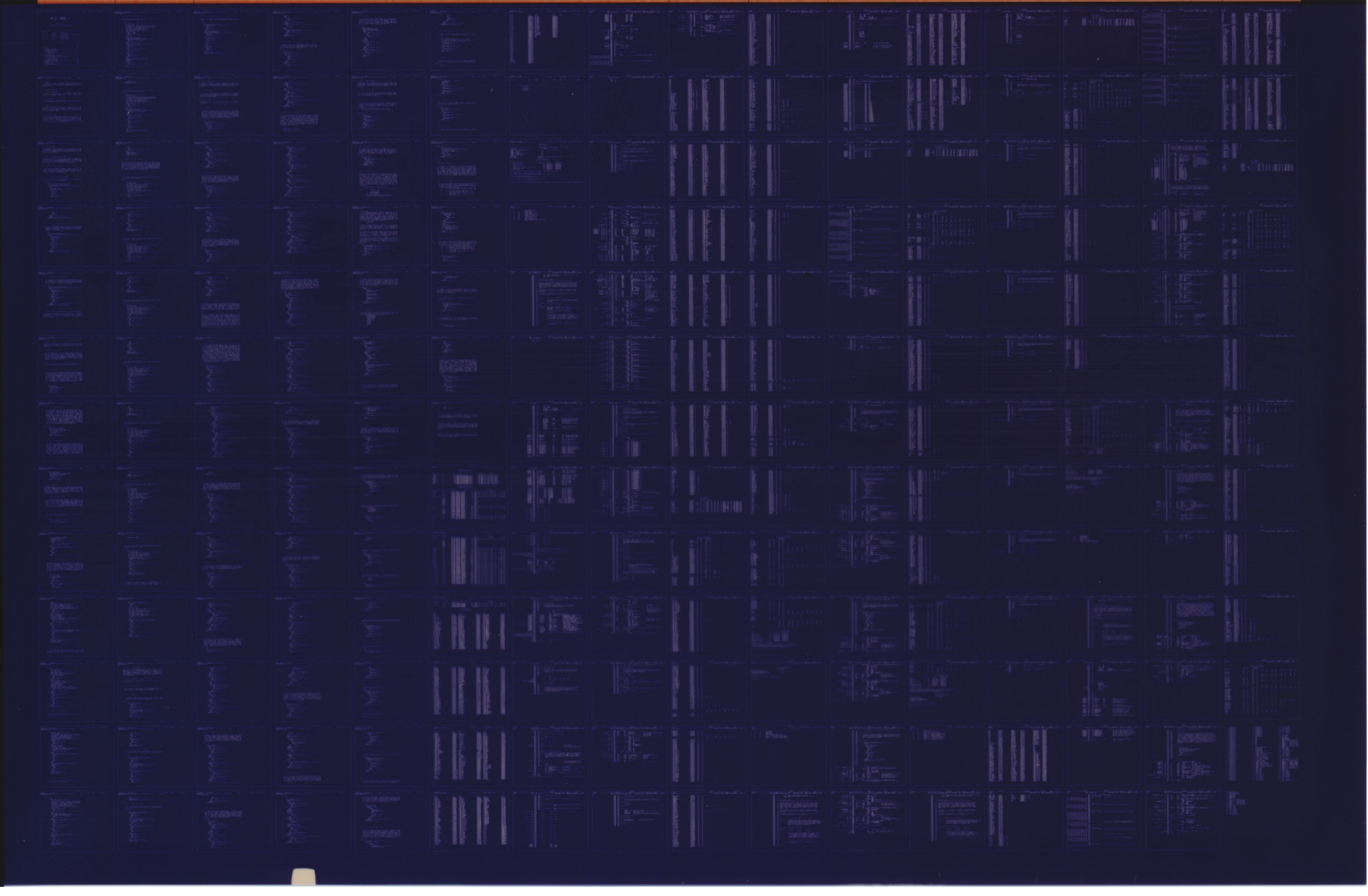
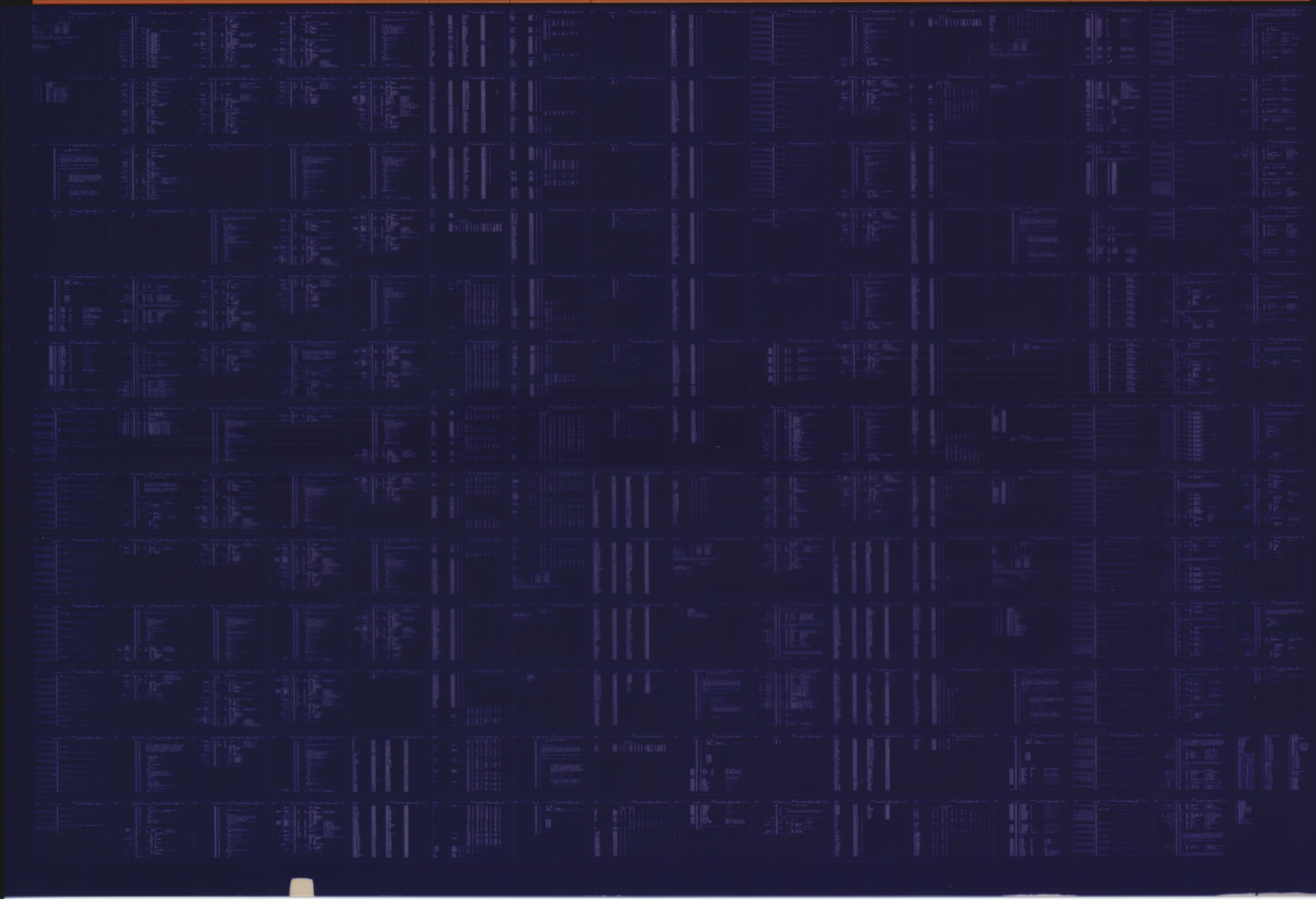
ENKAX-1.3 VAX 11730 CLUSTER EXERCISER

EP-ENKAX-DL-1.3 FICHE 1 OF 4



ENKAX-1.3 VAX 11730 CLUSTER EXERCISER

EP-ENKAX-DL-1.3
FICHE 2 OF 4



ENKAX-1.3 VAX 11730 CLUSTER EXERCISER

EP-ENKAX-DL-1.3 FICHE 3 OF 4

d i g i t a l

ENKAX-1.3 VAX 11730 CLUSTER EXERCISER EP-ENKAX-DL-1.3 FICHE 4 OF 4

dec

PRODUCT CODE: ZZ-ENKAX-1.3
DATE: 05-AUG-1983
CHARGE NUMBER: Q-098-05900

REVISION 1.3 OF

NEBULA Specific CPU Cluster Exerciser

REVISION HISTORY:

Rev. 1.0 on	03-MAR-82	by Rich Lewis
Rev. 1.1 on	09-NOV-82	by Kevin Martin
Rev. 1.2 on	23-MAY-83	by Tai-Chun Pan and Peter Green
Rev. 1.3 on	05-AUG-83	by Tai-Chun Pan

CONTENTS

1.0 ABSTRACT

This program tests portions of the NEBULA CPU cluster hardware which are not specified by the VAX architecture or require manual intervention. This program must run in a standalone environment under control of the Diagnostic Supervisor.

2.0 PROGRAMMING CONVENTIONS

2.1 Implementation Language

This program will be written mostly in VAX assembly language (MACRO) utilizing the Diagnostic Macro Library (DIAG.MLB). The program interfaces to the Diagnostic Supervisor.

2.2 Program Generation

The program modules are assembled individually with MACRO and linked into an executable image using the VAX linker (LINK).

2.3 Transportability

This program is not intended to be run on or transportable to any CPU except the VAX-11/730. Some portions may indeed be found to be transportable, but this program is design to test all parts of the CPU that fall outside the realm of the VAX architecture as defined by the System Reference Manual (SRM).

2.4 Event And Quick Flags

This program implements Event Flag 3 and the Quick Flag. The quick flag is used in Test 7: TU58 EXERCISOR and when set skips subtests 4 - 11 in Test 7 as documented in sections 3.7.2.4 - 3.7.2.11. Event flag 3 when set doesn't read the TU58 SCRAT.CH label and will override the label protection i.e.,write on the tape (providing the TU58 tab is in the write position). If APT is running and event flag 3 is clear and the TU58 is not labeled SCRAT.CH an error will occur.

3.0 DESIGN DESCRIPTION

The NEBULA Specific CPU Cluster Exerciser will be developed using as much as possible from ECKAX (COMET Specific CPU Cluster Exerciser). Due to differences in hardware, some tests will require redesign. All other tests will be developed from scratch because equivalent tests were not developed for the VAX-11/750.

3.1 ENKAXA: RESERVED PROCESSOR REGISTERS

This module tests that all processor register numbers not implemented on the VAX-11/730 will cause a reseved operand fault for a MTPR or MFPR instruction. Also internal processor registers which are write and read only and specific to the VAX-11/730 are checked to not cause reserved operand faults, except in the cases where they should fault.

3.1.1 Reserved Processor Registers

This test checks that processor numbers not used by the VAX-11/730 cause reserve operand faults for the MFPR and MTPR instructions. All read only and write only registers are checked to cause a reserved operand fault when the disallowed function is attempted. (i.e. - write to a read only or read a write only)

3.1.1.1 Reserved Processor Register Numbers

This subtest checks all the reserved processor register numbers that are not used in the VAX-11/730 processor.

Subtest steps:

```
SUBTEST RESERVED PROCESSOR REGISTER NUMBERS
:  Initialize pointer to address table
:  Get reserved operand fault vector to handle exception
:  REPEAT
:  :  Clear error flags
:  :  Get address of register to attempt access
:  :  Attempt MTPR on register
:  :  Check for errors
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  Attempt MFPR on register
:  :  Check for errors
```

```
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL All unused addresses done
:  ENDREPEAT
ENDSUBTEST RESERVED PROCESSOR REGISTER NUMBERS
```

3.1.1.2 Read only Processor Registers

This subtest checks processor-specific registers which are read only registers. This subtest attempts a read and checks that a reserved operand fault does not occur. Then an attempt to write the register is made which should cause a reserved operand fault. If an error is encountered it is reported.

Subtest steps:

```
SUBTEST READ ONLY PROCESSOR REGISTERS
:  Get reserved operand fault vector to handle exception
:  REPEAT
:  :  Clear error flags
:  :  Attempt MTPR on register
:  :  Check for errors
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  Attempt MFPR on register
:  :  Check for errors
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all read only registers are done
:  ENDREPEAT
ENDSUBTEST READ ONLY PROCESSOR REGISTERS
```

3.1.1.3 Write Only Processor Registers

This subtest checks processor-specific registers which are write only registers. This subtest attempts a write and checks that a reserved operand fault does not occur. Then an attempt to read each read only register is made which should cause a reserved operand fault. If an error is encountered it is reported.

Subtest steps:

```
SUBTEST WRITE ONLY PROCESSOR REGISTERS
:  Get reserved operand fault vector to handle exception
:  REPEAT
:  :  Clear error flags
:  :  Attempt MTPR on register
:  :  Check for errors
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  Attempt MFPR on register
:  :  Check for errors
:  :  IF any errors
:  :  :  THEN
:  :  :  Report error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all write only registers are done
:  ENDREPEAT
ENDSUBTEST WRITE ONLY PROCESSOR REGISTERS
```

3.2 ENKAXB: ARCHITECTURE "UNPREDICTABLE" AND "UNDEFINED" OPERATIONS

This module was not implemented for any VAX processor and it was agreed at the Nebula functional specification review that it will not be implemented for the VAX 11/730. The amount of effort required is excessive and the payback in terms of design verification coverage is felt to be minimal.

3.3 ENKAXC: POWER FAIL

This module tests that the VAX-11/730 power failure and recovery mechanism functions properly. The power fail tests instruct the operator to select console panel switch settings and force various power interruptions.

3.3.1 POWER FAIL

This test consists of a series of manual operator intervention subtests. These subtests will check that the Restart Parameter Block (RPB) is recognizable and initiates the proper sequence of events during and after a power failure. The power fail is induced by turning the key switch on the front panel of the 11/730 to the STAND BY position or by opening the BREAKER. With the battery backup on and the select switch in the restart position a warm restart will be attempted after the power is restored.

Assumptions:

For this test to function properly the front panel AUTO RESTART switch must be in the ON position and the battery backup must be on.

3.3.1.1 GOOD RPB SUBTEST

This subtest sets up a good RPB at location zero, which points to a power up routine that restores all pertinent processor registers, the argument pointer, frame pointer and stack pointer. The operator is instructed through the console terminal to turn the key switch on the front panel to the STAND BY position and then back to the local position or open the BREAKER to induce a CPU power failure. If the RPB is interpreted properly the instrucions for the next subtest will be printed on the console terminal. If an error occurred, either the processor will unexpectedly reboot or an error report will be generated by the diagnostic supervisor. This means the RPB was not recognized and a cold restart has been attempted.

SUBTEST STEPS:

```
SUBTEST GOOD RPB
:  Setup good RPB at location 0
:  Send instructions to operator
:  Recover from power fail
:  IF errors
:  :  THEN REPORT errors
:  ENDIF
ENDSUBTEST GOOD RPB
```

3.3.1.2 SEARCH FOR GOOD RPB

This subtest clears the first 1F (^X) memory locations starting with zero to assure that the string of zeros at location 0 will not be misinterpreted as a RPB even though all zeros fills basic criteria for a RPB (i.e., the starting address and checksum of all zeros would be good). It also sets up a good RPB at a higher page boundary which points to a power up routine that restores all pertinent processor registers, the argument pointer, frame pointer and stack pointer. This will test a search through memory to find a valid RPB. The operator is instructed through the console terminal to turn the key switch on the front panel to the STAND BY position or open the BREAKER to induce a CPU power failure. If location zero is misinter- preted as a RPB the processor will halt with code = 1. If the real RPB is interpreted properly the next subtest instructions will be printed on the console terminal. If an error occurred, either the processor will reboot unexpectedly or an error report will be generated by the diagnostic supervisor.

SUBTEST STEPS:

```
SUBTEST SEARCH FOR GOOD RPB
:  Clear first 1F (X) locations
:  Setup good RPB at higher page boundary
:  Send instructions to operator
:  Recover from Power Fail
:  IF any errors
:  :  THEN REPORT errors
:  ENDIF
ENDSUBTEST SEARCH FOR GOOD RPB
```

3.3.1.3 BAD CHECKSUM SUBTEST

This subtest sets up a RPB at location zero with a bad checksum. It points to a power up routine that restores all pertinent processor registers, the argument pointer, frame pointer and stack pointer and will set an error flag indicating that the checksum did not invalidate the RPB. Further into memory a good RPB will be located to warm restart normally without setting an error flag (this prevents an auto cold start). The operator is instructed through the console terminal to turn the key switch on the front panel to the STAND BY position or open the BREAKER to induce a CPU power failure. If the second RPB is not inter- preted properly, the system will attempt a cold restart and boot. If the RPB with the bad checksum causes the warm start an error will be reported.

SUBTEST STEPS:

```
SUBTEST BAD CHECKSUM
:  Setup RPB at location 0 with bad checksum
:  Setup good RPB at higher page boundary
:  Send instructions to operator
:  Recover from power fail
:  IF errors
:  :  THEN REPORT errors
:  ENDIF
ENDSUBTEST BAD CHECKSUM
```

3.4 ENKAXD: PROCESSOR HALT

This module consists of two tests that check the various ways the VAX-11/730 processor can be halted in an ARCHITECTURALLY DEFINED and an ARCHITECTURALLY UNDEFINED manner. These ways include executing a HALT instruction in kernel mode, attempting certain references to an invalid interrupt stack, executing a change mode instruction while on the interrupt stack, executing a change mode instruction which will be serviced on the interrupt stack, SCBB read error, invalid vector, and double machine check halts.

3.4.1 PROCESSOR HALTS ARCH. DEFINED

This test checks the various ways the VAX-11/730 processor can be halted in an architecturally defined manner. This includes executing a HALT instruction in kernel mode, attempting certain references to an invalid interrupt stack and executing change mode instructions when the IS bit of the PSL is set. INSTRUCTIONS FOR EARLY ABORT: If it is desired to exit the test before completion, one must wait for the console prompt (>>>) after the halt occurs and type in the sequence D PC adr <CR> C <CR>, where adr is the absolute address of the label TEST_003_X:: in the program listing.

3.4.1.1 HALT Inst. in Kernel Mode

This subtest checks that the HALT instruction halts the processor when executed in kernel mode (code = 6).

Subtest steps:

```
SUBTEST Halt Instruction in Kernel Mode
:  Initialize Flags
:  Setup code to set done flag at loc 100
:  Setup a jump to READ_GPR at next loc
```

```
:  Print instructions to operator
:  Load expected data for stacks
:  Load GPR's with data
:  Set ready to execute flag
:  Execute HALT
:  IF halt not executed properly
:  :  THEN
:  :  (will continue inline execution and get here)
:  :  Set halt execution error flag
:  ENDIF
:  Rentry point (restart will enter here)
:  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
:  IF any errors
:  :  THEN
:  :  Report errors
:  ENDIF
:  IF fatal error
:  :  THEN
:  :  Abort test
:  ENDIF
ENDSUBTEST Halt instruction in kernel mode
```

3.4.1.2 Interrupt Stack invalid

This subtest checks that an interrupt stack which is located on a page which is protected to kernel read only, will halt the processor when a write is attempted. An XFC instruction is executed to get onto the interrupt stack prior to invalidating the interrupt stack pointer. A second XFC instruction is executed with the vector specifying the interrupt stack for service. When the XFC instruction is executed there is an attempt to write the PC and PSL on the interrupt stack. When this happens an "Interrupt Stack not valid HALT" should occur (halt code = 4). Memory management is on for this subtest.

Subtest steps:

```
SUBTEST Interrupt Stack invalid
:  Initialize Flags
:  Turn memory management on
:  Set page0 protection to kernel write access
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort program
:  ENDIF
:  Get XFC vector to handle on interrupt stack
:  Execute XFC instruction
:  IF not on the interrupt stack
:  :  THEN
:  :  Report system error
:  :  Abort program
```

```
:  ENDIF
:  Release XFC vector
:  Get XFC vector to handle on interrupt stack
:  Setup code to set done flag at loc 100
:  Setup a jump to READ_GPR at next loc
:  Save current IS SP to reset after subtest is complete
:  Set interrupt stack pointer to buffer area
:  Invalidate buffer page (kernel read only access)
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort program
:  ENDIF
:  Print instructions to operator
:  Setup all expected data
:  Load GPR's with data
:  Set ready to execute flag
:  Execute XFC to attempt to write PC and PSL on ..
:  (this should cause an interrupt stack not valid HALT)
:  IF XFC not executed properly
:  :  THEN
:  :  (will continue in line execution and get here)
:  :  Set XFC execution error flag
:  ENDIF
:  Rentry point (restart will enter here)
:  Read PSL
:  IF on IS as expected
:  :  THEN
:  :  Restore IS SP
:  ENDIF
:  Go check FLAGS, PSL, KSP, ISP, SP and GPR'S for errors
:  (Either the KSP or ISP will be UNPREDICTABLE depending on the
:  current stack, this is machine dependant)
:  IF any errors
:  :  THEN
:  :  Report errors
:  ENDIF
:  IF fatal error
:  :  THEN
:  :  Abort program
:  ENDIF
:  Reset protection on buffer page to original protection
:  Turn memory management off
:  Release XFC vector
ENDSUBTEST Interrupt Stack invalid
```

3.4.1.3 IS SP points to NXM

This subtest checks that an interrupt stack, stack pointer which points to nonexistent memory will halt the processor. Memory management is off for this subtest and an Interrupt Stack not valid HALT should occur (code = 4).

Subtest steps:

```
SUBTEST IS points to NXM
:  Initialize Flags
:  Setup code to set done flag at loc 100
:  Save current IS SP
:  Setup a jump to RENTRY point at next loc
:  Get XFC vector to handle on interrupt stack
:  Execute XFC instruction
:  IF not on the interrupt stack
:  :  THEN
:  :  Report system error
:  :  Abort program
:  ENDIF
:  Release XFC vector
:  Get XFC vector for service on the interrupt stack
:  Print instructions to operator
:  Point interrupt stack pointer to NXM
:  Setup all expected data
:  Load GPR's with data
:  Set ready to execute flag
:  Execute XFC which will write PC and PSL on IS
:  (this should cause an interrupt stack not valid HALT)
:  IF XFC not executed (will continue and get here)
:  :  THEN
:  :  Set XFC execution error flag
:  ENDIF
:  RENTRY point (restart will enter here)
:  Read PSL
:  IF on IS stack as expected
:  :  THEN
:  :  Restore IS SP
:  ENDIF
:  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
:  IF any errors
:  :  THEN
:  :  Report errors
:  ENDIF
:  IF fatal error
:  :  THEN
:  :  Abort test
:  ENDIF
:  Release XFC vector
ENDSUBTEST IS points to NXM
```

3.4.1.4 CHMU inst. with PSL<IS>=1

This subtest checks that a CHMU instruction with PSL<IS>=1, will cause the processor to HALT (code = A).

Subtest steps:

```
SUBTEST CHMU instruction with PSL<IS>=1
:  Initialize Flags
:  Get XFC vector to handle exception on interrupt stack
:  Get CHMU vector in case CHMU instruction does execute
:  Setup code to set done flag at loc 100
:  Setup code to jump to READ_GPR point in next loc
:  Print restart instructions to operator
:  Execute an XFC instruction to get on interrupt stack
:  Read current PSL
:  IF NOT on interrupt stack
:  :  THEN
:  :  Report wrong stack error
:  :  Abort test
:  ENDIF
:  Setup all expected data
:  Load GPR's with data
:  Set ready to execute flag
:  Execute CHMU to cause processor to HALT
:  (this should cause a CHMU on interrupt stack HALT)
:  IF CHMU does not caues an exception
:  :  THEN
:  :  Set execution error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
:  IF any errors
:  :  THEN
:  :  Report errors
:  ENDIF
:  IF fatal error
:  :  THEN
:  :  Abort test
:  ENDIF
:  Modify PC on stack to point to second RENTRY point
:  Execute REI to get back to correct stack
:  Second RENTRY point
:  Read PSL
:  IF not on kernel stack
:  :  THEN
:  :  Report incorrect stack error
:  :  Abort test
:  ENDIF
:  Release interval timer vector
:  Release CHMU vector
:  Release XFC vector
ENDSUBTEST CHMU instruction with PSL<IS>=1
```

3.4.1.5 CHMS inst. with PSL<IS>=1

This subtest checks that a CHMS instruction with PSL<IS>=1, will cause the processor to HALT (code = A).

Subtest steps:

```
SUBTEST CHMS instruction with PSL<IS>=1
:  Initialize Flags
:  Get XFC vector to handle exception on interrupt stack
:  Get interval timer vector to prevent outside interrupts
:  Get CHMS vector in case CHMS instruction does execute
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Execute an XFC instruction to get on interrupt stack
:  Read current PSL
:  IF NOT on interrupt stack
:  :  THEN
:  :  Report wrong stack error
:  :  Exit test
:  ENDIF
:  Set ready to execute flag
:  Execute CHMS to cause processor to HALT
:  (this should cause a CHMS on interrupt stack HALT)
:  IF CHMS does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHMS exception error
:  :  Exit test
:  ENDIF
:  IF CHMS instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Modify PC on stack to point to RENTRY2
:  Execute REI to get back to correct stack
:  RENTRY2 point
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHMS vector
:  Release XFC vector
ENDSUBTEST CHMS instruction with PSL<IS>=1
```

3.4.1.6 CHME inst. with PSL<IS>=1

This subtest checks that a CHME instruction with PSL<IS>=1, will cause the processor to HALT (code = A).

Subtest steps:

```
SUBTEST CHME instruction with PSL<IS>=1
:  Initialize Flags
:  Get XFC vector to handle exception on interrupt stack
:  Get interval timer vector to prevent outside interrupts
:  Get CHME vector in case CHME instruction does execute
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Execute an XFC instruction to get on interrupt stack
:  Read current PSL
:  IF NOT on interrupt stack
:  :  THEN
:  :  Report wrong stack error
:  :  Exit test
:  ENDIF
:  Set ready to execute flag
:  Execute CHME to cause processor to HALT
:  (this should cause a CHME on interrupt stack HALT) ,
:  IF CHME does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHME exception error
:  :  Exit test
:  ENDIF
:  IF CHME instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Modify PC on stack to point to RENTRY2
:  Execute REI to get back to correct stack
:  RENTRY2 point
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
```

```
:  Release CHME vector
:  Release XFC vector
ENDSUBTEST CHME instruction with PSL<IS>=1
```

3.4.1.7 CHMK inst. with PSL<IS>=1

This subtest checks that a CHMK instruction with PSL<IS>=1, will cause the processor to HALT (code = A).

Subtest steps:

```
SUBTEST CHMK instruction with PSL<IS>=1
:  Initialize Flags
:  Get XFC vector to handle exception on interrupt stack
:  Get interval timer vector to prevent outside interrupts
:  Get CHMK vector in case CHMK instruction does execute
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Execute an XFC instruction to get on interrupt stack
:  Read current PSL
:  IF NOT on interrupt stack
:  :  THEN
:  :  Report wrong stack error
:  :  Exit test
:  ENDIF
:  Set ready to execute flag
:  Execute CHMK to cause processor to HALT
:  (this should cause a CHMK on interrupt stack HALT)
:  IF CHMK does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHMK exception error
:  :  Exit test
:  ENDIF
:  IF CHMK instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Modify PC on stack to point to RENTRY2
:  Execute REI to get back to correct stack
:  RENTRY2 point
```

```
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHMK vector
:  Release XFC vector
ENDSUBTEST CHMK instruction with PSL<IS>=1
```

3.4.2 PROCESSOR HALTS ARCH. UNDEFINED

This test checks the various ways the VAX-11/730 processor can be halted in an architecturally undefined manner. This includes change mode instructions with the vector specifying service on the interrupt stack, invalid vector halt, SCBB read error halt, double machine check halt, and transfer to NXM user WCS halt. INSTRUCTIONS FOR EARLY ABORT: If it is desired to exit the test before completion, one must wait for the console prompt (>>>) after the halt occurs and type in the sequence D PC adr <CR> C <CR>, where adr is the absolute address of the label TEST_004_X:: in the program listing.

3.4.2.1 CHMU inst. with vector service on IS

This subtest checks that a CHMU instruction with the SCB vector entry <1:0>=1 specifying service on the interrupt stack will cause the processor to HALT (code = B).

Subtest steps:

```
SUBTEST CHMU instruction with vector service on IS
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get CHMU vector specifying service on IS
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute CHMU to cause processor to HALT
:  (this should cause a CHMU service on interrupt stack HALT)
:  IF CHMU does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
```

```
:  :  Report CHMU exception error
:  :  Exit test
:  ENDIF
:  IF CHMU instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHMU vector
ENDSUBTEST CHMU instruction with
```

3.4.2.2 CHMS inst. with vector service on IS

This subtest checks that a CHMS instruction with the SCB vector entry <i:0>=1 specifying service on the interrupt stack will cause the processor to HALT (code = B).

Subtest steps:

```
SUBTEST CHMS instruction with vector service on IS
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get CHMS vector specifying service on IS
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute CHMS to cause processor to HALT
:  (this should cause a CHMS service on interrupt stack HALT)
:  IF CHMS does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHMS exception error
:  :  Exit test
:  ENDIF
:  IF CHMS instruction executed flag set
```

```
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHMS vector
ENDSUBTEST CHMS instruction with
```

3.4.2.3 CHME inst. with vector service on IS

This subtest checks that a CHME instruction with the SCB vector entry <1:0>=1 specifying service on the interrupt stack will cause the processor to HALT (code = B).

Subtest steps:

```
SUBTEST CHME instruction with vector service on IS
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get CHME vector specifying service on IS
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute CHME to cause processor to HALT
:  (this should cause a CHME service on interrupt stack HALT)
:  IF CHME does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHME exception error
:  :  Exit test
:  ENDIF
:  IF CHME instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
```

```
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHME vector
ENDSUBTEST CHME instruction with
```

3.4.2.4 CHMK inst. with vector service on IS

This subtest checks that a CHMK instruction with the SCB vector entry <1:0>=1 specifying service on the interrupt stack will cause the processor to HALT (code = B).

Subtest steps:

```
SUBTEST CHMK instruction with vector service on IS
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get CHMK vector specifying service on IS
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute CHMK to cause processor to HALT
:  (this should cause a CHMK service on interrupt stack HALT)
:  IF CHMK does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report CHMK exception error
:  :  Exit test
:  ENDIF
:  IF CHMK instruction executed flag set
:  :  THEN
:  :  Report change mode error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
```

```
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release CHMK vector
ENDSUBTEST CHMK instruction with
```

3.4.2.5 Invalid vector HALT

This subtest checks that an XFC instruction with the SCB vector entry <1:0>=3 specifying an invalid vector will cause the processor to HALT (code = 7).

Subtest steps:

```
SUBTEST Invalid vector HALT
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get XFC vector bits <1:0> = 3 (invalid vector)
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute XFC to cause processor to HALT
:  (this should cause a invalid vector HALT)
:  IF XFC does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report XFC exception error
:  :  Exit test
:  ENDIF
:  IF XFC instruction executed flag set
:  :  THEN
:  :  Report XFC error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
```

```
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
ENDSUBTEST Invalid vector HALT
```

3.4.2.6 SCBB read error HALT

This subtest checks that if the SCBB points to NXM, when an exception occurs the processor will HALT (code = C).

Subtest steps:

```
SUBTEST SCBB read error HALT
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get XFC vector
:  Point SCBB to NXM
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute XFC to cause processor to HALT
:  (this should cause an SCBB read error HALT)
:  IF XFC does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report XFC exception error
:  :  Exit test
:  ENDIF
:  IF XFC instruction executed flag set
:  :  THEN
:  :  Report SCBB error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
```

```
:  Release XFC vector
ENDSUBTEST SCBB read error HALT
```

3.4.2.7 Double machine check HALT

This subtest checks that a double machine check halt will be generated if a second machine check occurs before a previous machine check has been serviced. (code=5)

Subtest steps:

```
SUBTEST Double machine check HALT
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Point machine check vector to NXM location
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Reference NXM to cause machine check and thus a halt
:  IF NXM does not cause an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error flag set
:  :  THEN
:  :  Report NXM exception error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Release machine check vector
:  Release interval timer vector
ENDSUBTEST Double machine check HALT
```

3.4.2.8 Transfer to NXM USER WCS

This subtest will attempt to cause a transfer to NXM USER WCS HALT by checking the p-table for a response to "USER WCS loaded?". If a "no" response is found then this test will execute and attempt to envoke microcode that has not been loaded into WCS. (code=8)

Subtest steps:

```
SUBTEST Transfer to NXM USER WCS HALT
:  Check p table for USER WCS not loaded
:  IF loaded
:  :  THEN
:  :  Exit subtest
:  ENDIF
:  Initialize Flags
:  Get interval timer vector to prevent outside interrupts
:  Get XFC vector to service in WCS
:  Setup code to set done flag at loc 100
:  Setup code to jump to RENTRY point in next loc
:  Print restart instructions to operator
:  Set ready to execute flag
:  Execute XFC to cause processor to HALT
:  (this should cause a transfer to NXM USER WCS HALT)
:  IF XFC does not caues an exception
:  :  THEN
:  :  Set exception error flag
:  ENDIF
:  RENTRY point (restart enters here from exception)
:  IF exception error or instruction executed flags set
:  :  THEN
:  :  Report XFC exception error
:  :  Exit test
:  ENDIF
:  IF done flag not set
:  :  THEN
:  :  Report done error
:  :  Exit test
:  ENDIF
:  Read PSL
:  IF IS not clear
:  :  THEN
:  :  Report incorrect stack error in clean up
:  :  Exit test
:  ENDIF
:  Release interval timer vector
:  Release XFC vector
ENDSUBTEST Transfer to NXM USER WCS HALT
```

3.5 ENKAXE: WRITABLE CONTROL STORE

It was agreed at the Nebula functional specification review that this module will not be implemented for the VAX 11/730 due to the requirements to load WCS with executable code and the inablity for a MACRO program to access WCS. To implement this test would require a special TU58 micro code tape with an additional file containing any micro code that would be used in this test. If WCS were accessible to a MACRO program, as it is in the VAX 11/780 and VAX 11/750, this test would be more feasible.

3.6 ENKAXF: MACHINE CHECK EXCEPTIONS

This module tests some of the possible ways in which a machine check exception may occure.

3.6.1 Machine Check Exceptions

This test checks that a reference to NXM, unaligned reference to I/O space, illegal I/O space address, and illegal UNIBUS reference will cause machine checks with the proper logout information.

3.6.1.1 Reference to NXM

This subtest attempts to read a nonexistent memory location, which should cause a machine check. The machine check logout pushed on the stack is checked for the correct information.

Subtest steps:

```
SUBTEST Reference to NXM
:  Get Machine Check vector to capture exception
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
:  ENDIF
:  Setup expected data and load GPR's with data
:  Set ready to execute flag
:  Access nonexistent memory (to cause exception)
:  Set execution error flag
:  IF no exception
:  :  THEN
:  :  Set execution error flag
:  ENDIF
:  Check machine check logout and register data
:  IF error is detected
```

```
:  :  THEN
:  :  Report data error
:  ENDIF
:  IF fatal error flag is set
:  :  THEN
:  :  Abort test
:  ENDIF
:  Release Machine Check vector
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
:  ENDIF
:  Push return PC and PSL on stack
:  Execute an REI
ENDSUBTEST Reference to NXM
```

3.6.1.2 Unaligned reference to I/O space

This subtest attempts an unaligned reference to I/O space and checks that a machine check occurs with the correct logout information on the stack.

Subtest steps:

```
SUBTEST Unaligned reference to I/O space
:  Get Machine Check vector to capture exception
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
:  ENDIF
:  Setup expected data and load GPR's with data
:  Set ready to execute flag
:  Access unaligned I/O space (to cause exception)
:  Set execution error flag
:  IF no exception
:  :  THEN
:  :  Set execution error flag
:  ENDIF
:  Check machine check logout and register data
:  IF error is detected
:  :  THEN
:  :  Report data error
:  ENDIF
:  IF fatal error flag is set
:  :  THEN
:  :  Abort test
:  ENDIF
:  Release Machine Check vector
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
```

```
        :  ENDIF
        :  Push return PC and PSL on stack
        :  Execute an REI
        ENDSUBTEST Unaligned reference to I/O space
```

3.6.1.3 Illegal I/O space address

This subtest attempts an illegal reference to I/O space and checks that a machine check occurs with the correct logout information on the stack.

Subtest steps:

```
        SUBTEST Illegal I/O space address
        :  Get Machine Check vector to capture exception
        :  IF unsucessful
        :  :  THEN
        :  :  Report system error
        :  :  Abort test
        :  ENDIF
        :  Setup expected data and load GPR's with data
        :  Set ready to execute flag
        :  Access illegal I/O space (to cause exception)
        :  Set execution error flag
        :  IF no exception
        :  :  THEN
        :  :  Set execution error flag
        :  ENDIF
        :  Check machine check logout and register data
        :  IF error is detected
        :  :  THEN
        :  :  Report data error
        :  ENDIF
        :  IF fatal error flag is set
        :  :  THEN
        :  :  Abort test
        :  ENDIF
        :  Release Machine Check vector
        :  IF unsucessful
        :  :  THEN
        :  :  Report system error
        :  :  Abort test
        :  ENDIF
        :  Push return PC and PSL on stack
        :  Execute an REI
        ENDSUBTEST Illegal I/O space address
```

3.6.1.4 Illegal UNIBUS reference

This subtest attempts an illegal UNIBUS reference and checks that a machine check occurs with the correct logout information on the stack.

Subtest steps:

```
SUBTEST Illegal UNIBUS reference
:  Get Machine Check vector to capture exception
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
:  ENDIF
:  Setup expected data and load GPR's with data
:  Set ready to execute flag
:  Access illegal UNIBUS address (to cause exception)
:  Set execution error flag
:  IF no exception
:  :  THEN
:  :  Set execution error flag
:  ENDIF
:  Check machine check logout and register data
:  IF error is detected
:  :  THEN
:  :  Report data error
:  ENDIF
:  IF fatal error flag is set
:  :  THEN
:  :  Abort test
:  ENDIF
:  Release Machine Check vector
:  IF unsucessful
:  :  THEN
:  :  Report system error
:  :  Abort test
:  ENDIF
:  Push return PC and PSL on stack
:  Execute an REI
ENDSUBTEST Illegal UNIBUS reference
```

3.7 ENKAXG: TU58 EXERCISER

This module exercises the TU58 tape controller to assure basic functionality. A series of subtests, including the controller's self test are executed checking initialize, no operation, write, read, seek, write modified, read modified, head switching and positioning. This test will

3.7.1 TU58 EXERCISER

This test exercises the TU58 on the VAX-11/730 by performing initialization, self diagnostic and no operation. All information received from the TU58 is checked for errors. All errors detected are reported. All data transfers will be expected to execute in Modified Radial Serial Protocol (MRSP).

Assumptions:

This subtest does not write or read the TU58. All operations performed deal with the controller only. (NOTE: Test 2 requires a scratch tape).

3.7.1.1 INITIALIZE

This subtest sends the TU58 a command packet to perform an initialization operation which puts the controller into a known ready state. While sending each byte of the packet, a timer is set which will time out if the ready bit in the transmitter processor register does not set within the alloted time. The same is done for the done bit in the receiver processor register while reading a byte from the TU58. When this is completed successfully, the returned end packet is compared to an expected end packet. If a time out occurs or bad data is encountered an error will be reported.

Subtest steps:

```
SUBTEST INITIALIZE
:  Clear buffer
:  Point to initialize command packet
:  REPEAT
:  :  Send command packet byte at a time
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  ·  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
```

```
                :  ENDREPEAT
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  Point to expected and actual end packet
                :  REPEAT
                :  :  IF data expected NEQ data received
                :  :  :  THEN
                :  :  :  Report error
                :  :  :  Exit test
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST INITIALIZE
```

3.7.1.2 SELF DIAGNOSTIC

This subtest sends the TU58 a command packet to run its internal diagnostic. While sending each byte of the packet,a timer is set which will time out if the ready bit in the transmitter processor register does not set within the alloted time. The same is done for the done bit in the receiver processor register while reading a byte from the TU58. When this is completed successfully, the returned end packet is compared to an expected end packet. If a time out occurs or bad data is encountered an error will be reported.

Subtest steps:

```
        SUBTEST SELF DIAGNOSTIC
        :  Clear buffer
        :  Point to self diagnostic command packet
        :  REPEAT
        :  :  Send command packet byte at a time
        :  :  Set timer
        :  :  IF timer expired AND no response
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
```

```
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
                :  :  :  Report error
        #       :  :  :  Exit test
        #       :  ENDIF
                :  Point to expected and actual data
                :  REPEAT
                :  :  IF expected data NEQ received data
                :  :  :  THEN
                :  :  :  Report data error
                :  :  :  Exit test
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST SELF DIAGNOSTIC
```

3.7.1.3 NOP

This subtest sends the TU58 a command packet to perform a no operation. While sending each byte of the packet, a timer is set which will time out if the ready bit in the transmitter processor register does not set within the alloted time. The same is done for the done bit in the receiver processor register while reading a byte from the TU58. When this is completed successfully, the returned end packet is compared to an expected end packet. If a time out occurs while expecting a byte transfer or bad data is encountered an error will be reported.

Subtest steps:

```
        SUBTEST NOP
        :  Clear buffer
        :  Point to no operation command packet
        :  REPEAT
        :  :  Send command packet byte at a time
        :  :  Set timer
        :  :  IF timer expired AND no response
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
        :  REPEAT
```

```
                :   :  Set timer
                :   :  IF timer expired AND done set
                :   :  : THEN
                :   :  :  :  Read byte
                :   :  ENDIF
                :   :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  : THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  Point to expected and actual end packet
                :  REPEAT
                :  :  IF expected data NEQ received data
                :  :  : THEN
                :  :  :  Report data error
                :  :  :  Exit test
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST NOP
```

3.7.2 TU58 EXERCISER READ/WRITE

This test exercises the TU58 on the VAX-11/730 by performing position, read (normal and two versions of modified) and write (normal and two versions of modified) commands. Zero padding of blocks and records are also checked for write commands. All end and data packets are checked for errors. Timers are setup to time out if a byte transaction between the host and TU58 is not completed within a reasonable amount of time. If any errors are detected, they are reported with all available information pertaining to the failure.

Assumptions:

This test assumes the scratch tape cartridge used is in good condition. If a marginal tape is used bits may be dropped during the read modified test, which reads with an increased threshold and will consequently report false data errors. Any tape may be used, but existing data will be destroyed. Before the tape is written, the first directory will be read. If the directory read is not "scrat.ch", the operator will be prompted to either overwrite the tape or abort this test. If overwriting the tape is chosen, an attempt to write "scrat.ch" in the first directory area will be made. If an operational error occurs it will be reported, but error detection in this subtest is limited and the primary function is to give the operator an opportunity to remove a tape containing valuable data from the drive and inserting a scratch tape before it is destroyed. All transfers
will be expected to execute in Modified Radial Serial Protocol (MRSP).

3.7.2.1 READ DIRECTORY

This subtest sends the TU58 a command packet to read the first entry in the directory area of the tape. A timer is set which will time out if the ready bit in the transmitter processor register does not set within the allotted time. A second timer is set which will time out if the done bit in the receiver processor register does not set within the allotted amount of time. If either of these timers expires while expecting a data transfer an error will be reported stating the conditions existing at the time of failure. The returned data packet is examined to see if the directory on the tape is "scrat.ch". If it is a scratch tape testing continues; if not, the operator will be prompted to either abort the TU58 exerciser test or overwrite the tape in the drive. Before entering the appropriate response on the TTY, the tape cartridge may be replaced with a scratch tape. In the case of an overwrite, the TU58 will be sent a write command packet to write "scrat.ch" in the directory. Timers are set which will time out if the appropriate bits do not set within allotted times. If an error occurs all appropriate information will be reported. This subtest is not primarily used for error detection, if an error is encountered, testing should continue with the next subtest for diagnosis.

Subtest steps:

```
        SUBTEST READ DIRECTORY
        :  Clear buffer
        :  Point to read command packet
        :  REPEAT
        :  :  Send byte of command packet
        :  :  Set timer
        :  :  IF no response AND timer expired
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
        :  REPEAT
        :  :  Set timer
        :  :  IF timer expired AND done set
        :  :  :  THEN
        :  :  :  :  Read byte
        :  :  ENDIF
        :  :  UNTIL timer expired AND done clear
        :  ENDREPEAT
#       :  IF expected byte count NEQ actual byte count
#       :  :  THEN
#       :  :  :  Report error
        :  :  :  Exit test
#       :  ENDIF
        :  Examine data for first directory on tape
        :  IF directory NEQ scrat.ch
        :  :  THEN
        :  :  WHILE write directory flag EQL zero DO
```

```
: : : Prompt operator to overwrite directory
: : : IF response EQL no
: : : : THEN
: : : : Exit test
: : : ENDIF
: : : IF response EQL yes
: : : : THEN
: : : : Set write directory flag
: : : : Point to write directory command packet
: : : : REPEAT
: : : : : Send command packet byte at a time
: : : : : Set timer
: : : : : IF timer expired AND no response
: : : : : : THEN
: : : : : : Report no response error
: : : : : : Exit test
: : : : : ENDIF
: : : : : UNTIL all bytes sent
: : : : ENDREPEAT
: : : : Wait for continue
: : : : Set timer
: : : : IF timer expired AND no response
: : : : : THEN
: : : : : Report no response error
: : : : : Exit test
: : : : ENDIF
: : : : Point to data packet with "scrat.ch" directory
: : : : REPEAT
: : : : : Send data packet byte at a time
: : : : : Set timer
: : : : : IF timer expired AND no response
: : : : : : THEN
: : : : : : Report no response error
: : : : : : Exit test
: : : : : ENDIF
: : : : : UNTIL all bytes sent
: : : : ENDREPEAT
: : : : REPEAT
: : : : : Read end packet into buffer byte at a time
: : : : : Set timer
: : : : : IF timer expired AND no response
: : : : : : THEN
: : : : : : Report no response error
: : : : : : Exit test
: : : : : ENDIF
: : : : : : UNTIL all bytes read
: : : : ENDREPEAT
: : : : Point to expected and received end packets
: : : : REPEAT
: : : : : IF expected data NEQ received data
: : : : : : THEN
: : : : : : Report data error
: : : : : : Exit subtest
: : : : : ENDIF
: : : : : UNTIL all bytes checked
: : : : ENDREPEAT
```

```
:  :  :  ENDIF
:  :  ENDWHILE
:  ENDIF
ENDSUBTEST READ DIRECTORY
```

3.7.2.2 WRITE BLOCK

This subtest sends the TU58 a command packet to write one block with 512 bytes of data. A continue flag should be returned between sending each data packet and an end packet is then returned from the TU58. Four 128-byte data packets are sent to fill one block. All byte transactions between the host and TU58 are timed. If any timer times out while expecting a data transfer to occur an error is reported. The data packets contain alternating bytes of all ones and all zeroes. The end packet is checked and if any errors have occured, they are reported.

Subtest steps:

```
SUBTEST WRITE BLOCK
:  Clear buffer
:  Point to write command packet
:  REPEAT
:  :  Send command packet byte at a time
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
:  ENDREPEAT
:  DO for count = 1 to 4
:  :  Wait for continue
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  Point to data packet
:  :  REPEAT
:  :  :  Send data packet byte at a time
:  :  :  Set timer
:  :  :  IF timer expired AND no response
:  :  :  :  THEN
:  :  :  :  Report no response error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  UNTIL all bytes sent
:  :  ENDREPEAT
```

```
                :  ENDDO
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  REPEAT
                :  :  Point to expected end packet
                :  :  IF expected end packet data NEQ received end packet data
                :  :  :  THEN
                :  :  :  Report end packet data error
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST WRITE BLOCK
```

3.7.2.3 READ BLOCK

This subtest sends the TU58 a command packet to read the 512 bytes of data that were written in the previous subtest. A timer is set which will time out if there is no response from the TU58 between each byte transaction. The end packet and data packets are checked for errors. If any errors are encountered they are reported.

Subtest steps:

```
          SUBTEST READ BLOCK
          :  Clear buffer
          :  Point to read command packet
          :  REPEAT
          :  :  Send command packet byte at a time
          :  :  Set timer
          :  :  IF timer expired and no response
          :  :  :  THEN
          :  :  :  Report no response error
          :  :  :  Exit test
          :  :  ENDIF
          :  :  UNTIL all bytes sent
          :  ENDREPEAT
          :  REPEAT
          :  :  Set timer
          :  :  IF timer expired AND done set
          :  :  :  THEN
```

```
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
          #     :  IF expected byte count NEQ actual byte count
          #     :  :  THEN
          #     :  :  :  Report error
                :  :  :  Exit test
          #     :  ENDIF
                :  REPEAT
                :  :  Point to expected data packet
                :  :  REPEAT
                :  :  :  IF data expected NEQ data received
                :  :  :  :  THEN
                :  :  :  :  Report data error
                :  :  :  :  Exit subtest
                :  :  :  ENDIF
                :  :  :  Point to next byte of data
                :  :  :  UNTIL all data in packet checked
                :  :  ENDREPEAT
                :  :  UNTIL 4 data packets checked
                :  ENDREPEAT
                :  Point to expected end packet
                :  REPEAT
                :  :  IF data expected NEQ data received
                :  :  :  THEN
                :  :  :  Report data error
                :  :  :  Exit subtest
                :  :  ENDIF
                :  :  UNTIL all end packet bytes checked
                :  ENDREPEAT
               ENDSUBTEST READ BLOCK
```

3.7.2.4 WRITE MODIFIED

This subtest sends the TU58 a command packet to write modified 512 bytes of data. A timer is set which will time out if a response is not received within the allotted time on all byte transactions between the host and TU58. Four 128 byte data packets are sent. Before each data packet is sent a continue is read from the TU58 and when all data has been sent the end packet is read from the TU58 and checked. If any errors are encountered they are reported. This modified command will write all of the data and then back up and check the checksum for data just written.

Subtest steps:

```
SUBTEST WRITE MODIFIED
:  Clear buffer
:  Point to write command packet
:  REPEAT
```

```
                :  :  Send command packet byte at a time
                :  :  Set timer
                :  :  IF timer expired AND no response
                :  :  :  THEN
                :  :  :  Report no response error
                :  :  :  Exit test
                :  :  ENDIF
                :  :  UNTIL all bytes sent
                :  ENDREPEAT
                :  DO for count = 1 to 4
                :  :  Wait for continue
                :  :  Set timer
                :  :  IF timer expired AND no continue
                :  :  :  THEN
                :  :  :  Report no continue error
                :  :  :  Exit test
                :  :  ENDIF
                :  :  Point to data packet
                :  :  REPEAT
                :  :  :  Send data packet byte at a time
                :  :  :  Set timer
                :  :  :  IF timer expired AND no response
                :  :  :  :  THEN
                :  :  :  :  Report no response error
                :  :  :  :  Exit test
                :  :  :  ENDIF
                :  :  :  UNTIL all bytes sent
                :  :  ENDREPEAT
                :  ENDDO
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
          #     :  IF expected byte count NEQ actual byte count
          #     :  :  THEN
          #     :  :  :  Report error
                :  :  :  Exit test
          #     :  ENDIF
                :  Point to expected end packet
                :  REPEAT
                :  :  IF expected end packet data NEQ received end packet data
                :  :  :  THEN
                :  :  :  Report end packet data error
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST WRITE MODIFIED
```

3.7.2.5 READ MODIFIED

This subtest sends the TU58 a command packet to read modified the 512 bytes of data written in the previous subtest. This command will cause the TU58 to read the tape with an increased threshold. A timer is set which will time out if there is no response within the allotted time. After reading the first byte of data, another timer is set which will time out if an end packet is not received within the allotted time. If any data or timer errors are encountered, they are reported.

Subtest steps:

```
        SUBTEST READ MODIFIED
        :  Clear buffer
        :  Point to read command packet
        :  REPEAT
        :  :  Send command packet byte at a time
        :  :  Set timer
        :  :  IF timer expired and no response
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
        :  REPEAT
        :  :  Set timer
        :  :  IF timer expired AND done set
        :  :  :  THEN
        :  :  :  :  Read byte
        :  :  ENDIF
        :  :  UNTIL timer expired AND done clear
        :  ENDREPEAT
#       :  IF expected byte count NEQ actual byte count
#       :  :  THEN
#       :  :  :  Report error
        :  :  :  Exit test
#       :  ENDIF
        :  REPEAT
        :  :  Point to expected data packet
        :  :  REPEAT
        :  :  :  IF data expected NEQ data received
        :  :  :  :  THEN
        :  :  :  :  Report data error
        :  :  :  :  Exit subtest
        :  :  :  ENDIF
        :  :  :  Point to next byte of data
        :  :  :  UNTIL all data in packet checked
        :  :  ENDREPEAT
        :  :  UNTIL 4 data packets checked
        :  ENDREPEAT
        :  Point to expected end packet
        :  REPEAT
        :  :  IF end packet data expected NEQ end packet data received
```

```
:  :  :  THEN
:  :  :  Report end packet data error
:  :  :  Exit subtest
:  :  ENDIF
:  :  UNTIL all end packet bytes checked
:  ENDREPEAT
ENDSUBTEST READ MODIFIED
```

3.7.2.6 WRITE RECORD

This subtest sends the TU58 a command packet to write one record with 128 bytes of data. A continue flag is returned and the data packet is sent. All byte transactions between the host and TU58 are timed and will timeout if there is no response within the allotted time. If a timer expires while expecting a data transfer to occur, an error is reported. The data packet contains alternating bytes of all one's and all zero's. Both end and data packets are checked and if any errors have occured, they are reported.

Subtest steps:

```
SUBTEST WRITE RECORD
:  Clear buffer
:  Point to write command packet
:  REPEAT
:  :  Send command packet byte at a time
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
:  ENDREPEAT
:  Wait for continue
:  Set timer
:  IF timer expired AND no continue
:  :  THEN
:  :  Report no continue error
:  :  Exit test
:  ENDIF
:  Point to data packet
:  REPEAT
:  :  Send data packet byte at a time
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
```

```
                :  ENDREPEAT
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  REPEAT
                :  :  Point to expected end packet
                :  :  IF expected end packet data NEQ received end packet data
                :  :  :  THEN
                :  :  :  Report end packet data error
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
               ENDSUBTEST WRITE RECORD
```

3.7.2.7 READ RECORD

This subtest sends the TU58 a command packet to read the 128 bytes of data that were written in the previous subtest. A timer is set and times out if there is no response between all byte transactions. The end packet and data packet are checked for errors. If any errors are encountered they are reported.

Subtest steps:

```
        SUBTEST READ RECORD
        :  Clear buffer
        :  Point to read command packet
        :  REPEAT
        :  :  Send command packet byte at a time
        :  :  Set timer
        :  :  IF timer expired and no response
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
        :  REPEAT
        :  :  Set timer
        :  :  IF timer expired AND done set
        :  :  :  THEN
```

```
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  Point to expected data packet
                :  REPEAT
                :  :  IF data expected NEQ data received
                :  :  :  THEN
                :  :  :  Report data error
                :  :  :  Exit subtest
                :  :  ENDIF
                :  :  Point to next byte of data
                :  :  UNTIL all data in packet checked
                :  ENDREPEAT
                :  Point to expected end packet
                :  REPEAT
                :  :  IF expected end packet data NEQ received end packet data
                :  :  :  THEN
                :  :  :  Report end packet data error
                :  :  :  Exit subtest
                :  :  ENDIF
                :  :  UNTIL all end packet bytes checked
                :  ENDREPEAT
               ENDSUBTEST READ RECORD
```

3.7.2.8 WRITE BLOCK - PADDED ZEROS

This subtest sends the TU58 a series of command packets to check that a block written with less than 512 bytes of data, will pad the remainder of the block with data bytes of zeros. This is accomplished by writing two consecutive blocks each with 512 bytes of all ones. Then a command packet is sent to write the first block with one byte of alternating ones and zeroes. This command should write the first byte with 10101010 and the remaining 511 bytes with 00000000. The second block should still contain 512 bytes of all one's data. This data is read and checked. All end packets and data packets are checked as the subtest progresses. All byte transactions between host and TU58 are timed. If any errors are encountered, they are reported.

Subtest steps:

```
SUBTEST WRITE BLOCK PADDED ZEROS
:  Clear buffer
:  Point to first write command packet
:  DO for count = 1 to 3
```

```
                :  :  REPEAT
                :  :  :  Send command packet byte at a time
                :  :  :  Set timer
                :  :  :  IF timer expired AND no response
                :  :  :  :  THEN
                :  :  :  :  Report no response error
                :  :  :  :  Exit test
                :  :  :  ENDIF
                :  :  :  UNTIL all bytes sent
                :  :  ENDREPEAT
                :  :  DO for count = 1 to 4
                :  :  :  Wait for continue
                :  :  :  Set timer
                :  :  :  IF timer expired AND no continue
                :  :  :  :  THEN
                :  :  :  :  Report no continue error
                :  :  :  :  Exit test
                :  :  :  ENDIF
                :  :  :  Point to data packet
                :  :  :  REPEAT
                :  :  :  :  Send data packet byte at a time
                :  :  :  :  Set timer
                :  :  :  :  IF timer expired AND no response
                :  :  :  :  :  THEN
                :  :  :  :  :  Report no response error
                :  :  :  :  :  Exit test
                :  :  :  :  ENDIF
                :  :  :  :  UNTIL all bytes sent
                :  :  :  ENDREPEAT
                :  :  ENDDO
                :  :  REPEAT
                :  :  :  Set timer
                :  :  :  IF timer expired AND done set
                :  :  :  :  THEN
                :  :  :  :  :  Read byte
                :  :  :  ENDIF
                :  :  :  UNTIL timer expired AND done clear
                :  :  ENDREPEAT
            #   :  :  IF expected byte count NEQ actual byte count
            #   :  :  :  THEN
            #   :  :  :  :  Report error
            #   :  :  :  :  Exit test
            #   :  :  ENDIF
                :  :  REPEAT
                :  :  :  Point to expected end packet
                :  :  :  IF expected end data NEQ received end data
                :  :  :  :  THEN
                :  :  :  :  Report end packet data error
                :  :  :  ENDIF
                :  :  :  UNTIL all data checked
                :  :  ENDREPEAT
                :  :  Point to next write command
                :  ENDDO
                :  Clear buffer
                :  Point to first read command packet
                :  DO for count = 1 to 2
```

```
                :   :  REPEAT
                :   :  :  Send command packet byte at a time
                :   :  :  Set timer
                :   :  :  IF timer expired and no response
                :   :  :  :  THEN
                :   :  :  :  Report no response error
                :   :  :  :  Exit test
                :   :  :  ENDIF
                :   :  :  UNTIL all bytes sent
                :   :  ENDREPEAT
                :   :  :  REPEAT
                :   :  :  Set timer
                :   :  :  IF timer expired AND done set
                :   :  :  :  THEN
                :   :  :  :  :  Read byte
                :   :  :  ENDIF
                :   :  :  UNTIL timer expired AND done clear
                :   :  ENDREPEAT
          #     :   :  IF expected byte count NEQ actual byte count
          #     :   :  THEN
          #     :   :  :  Report error
                :   :  :  Exit test
          #     :   :  ENDIF
                :   :  REPEAT
                :   :  :  Point to expected data packet
                :   :  :  REPEAT
                :   :  :  :  IF data expected NEQ data received
                :   :  :  :  :  THEN
                :   :  :  :  :  Report data error
                :   :  :  :  :  Exit subtest
                :   :  :  :  ENDIF
                :   :  :  :  Point to next byte of data
                :   :  :  :  UNTIL all data in packet checked
                :   :  :  ENDREPEAT
                :   :  :  UNTIL 4 data packets checked
                :   :  ENDREPEAT
                :   :  Point to expected end packet
                :   :  REPEAT
                :   :  :  IF end data expected NEQ end data received
                :   :  :  :  THEN
                :   :  :  :  Report end packet data error
                :   :  :  :  Exit subtest
                :   :  :  ENDIF
                :   :  :  UNTIL all end packet bytes checked
                :   :  ENDREPEAT
                :   ENDDO ENDSUBTEST WRITE BLOCK PADDED ZEROS
```

3.7.2.9 WRITE RECORD - PADDED ZEROS

This subtest sends the TU58 a series of command packets to check that a record written with less than 128 bytes of data, will pad the remainder of the record with data bytes of zeroes. This is accomplished by writing two consecutive records each with 128 bytes of all ones. Then a command packet is sent to write the first record with one byte of alternating ones and zeroes data. This command should write the first byte with 10101010 and the remaining 127 bytes with 00000000. The second block should still contain 128 bytes of all ones data. This data is read and checked. All end packets and data packets are checked as the subtest progresses. All byte transactions between the host and TU58 are timed. If any errors are encountered, they are reported.

Subtest steps:

```
SUBTEST WRITE RECORD PADDED ZEROS
:  Clear buffer
:  Point to first write command packet
:  DO for count = 1 to 3
:  :  REPEAT
:  :  :  Send command packet byte at a time
:  :  :  Set timer
:  :  :  IF timer expired AND no response
:  :  :  :  THEN
:  :  :  :  Report no response error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  UNTIL all bytes sent
:  :  ENDREPEAT
:  :  Wait for continue
:  :  Set timer
:  :  IF timer expired AND no continue
:  :  :  THEN
:  :  :  Report no continue error
:  :  :  Exit test
:  :  ENDIF
:  :  Point to data packet
:  :  REPEAT
:  :  :  Send data packet byte at a time
:  :  :  Set timer
:  :  :  IF timer expired AND no response
:  :  :  :  THEN
:  :  :  :  Report no response error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  UNTIL all bytes sent
:  :  ENDREPEAT
:  :  REPEAT
:  :  :  Set timer
:  :  :  IF timer expired AND done set
:  :  :  :  THEN
:  :  :  :  :  Read byte
:  :  :  ENDIF
```

```
                :  :  :  UNTIL timer expired AND done clear
                :  :  ENDREPEAT
        #       :  :  IF expected byte count NEQ actual byte count
        #       :  :  : THEN
        #       :  :  :  : Report error
                :  :  :  : Exit test
        #       :  :  ENDIF
                :  :  REPEAT
                :  :  :  Point to expected end packet
                :  :  :  IF expected end data NEQ received end data
                :  :  :  : THEN
                :  :  :  : Report end packet data error
                :  :  :  ENDIF
                :  :  :  UNTIL all data checked
                :  :  ENDREPEAT
                :  :  Point to next write command
                :  ENDDO
                :  Clear buffer
                :  Point to first read command packet
                :  DO for count = 1 to 2
                :  :  REPEAT
                :  :  :  Send command packet byte at a time
                :  :  :  Set timer
                :  :  :  IF timer expired and no response
                :  :  :  : THEN
                :  :  :  : Report no response error
                :  :  :  : Exit test
                :  :  :  ENDIF
                :  :  :  UNTIL all bytes sent
                :  :  ENDREPEAT
                :  :  REPEAT
                :  :  :  Set timer
                :  :  :  IF timer expired AND done set
                :  :  :  : THEN
                :  :  :  : : Read byte
                :  :  :  ENDIF
                :  :  :  UNTIL timer expired AND done clear
                :  :  ENDREPEAT
        #       :  :  IF expected byte count NEQ actual byte count
        #       :  :  : THEN
        #       :  :  :  : Report error
                :  :  :  : Exit test
        #       :  :  ENDIF
                :  :  Point to expected data packet
                :  :  REPEAT
                :  :  :  IF data expected NEQ data received
                :  :  :  : THEN
                :  :  :  : Report data error
                :  :  :  : Exit subtest
                :  :  :  ENDIF
                :  :  :  Point to next byte of data
                :  :  :  UNTIL all data in packet checked
                :  :  ENDREPEAT
                :  :  Point to expected end packet
                :  :  REPEAT
                :  :  :  IF end data expected NEQ end data received
```

```
:  :  :  :  THEN
:  :  :  :  Report end packet data error
:  :  :  :  Exit subtest
:  :  :  ENDIF
:  :  :  UNTIL all end packet bytes checked
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST WRITE RECORD PADDED ZEROS
```

3.7.2.10 BLOCK WRITE - READ

This subtest sends the TU58 controller a series of command packets that will cause head switches, forward seeks, reverse seeks and seeks with head switches during read and write operations on blocks. All data written is read and checked. All end packets are checked as the subtest progresses. If any errors occur they are reported.

Subtest steps:

```
SUBTEST BLOCK WRITE - READ
:  DO for count = 1 to 5
:  :  Point to write command packet
:  :  Clear buffer
:  :  REPEAT
:  :  :  Send command packet byte at a time
:  :  :  Set timer
:  :  :  IF timer expired AND no response
:  :  :  :  THEN
:  :  :  :  Report no response error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  UNTIL all bytes sent
:  :  ENDREPEAT
:  :  DO for count = 1 to 4
:  :  :  Wait for continue
:  :  :  Set timer
:  :  :  IF timer expired AND no continue
:  :  :  :  THEN
:  :  :  :  Report no continue error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  Point to data packet
:  :  :  REPEAT
:  :  :  :  Send data packet byte at a time
:  :  :  :  Set timer
:  :  :  :  IF timer expired AND no response
:  :  :  :  :  THEN
:  :  :  :  :  Report no response error
:  :  :  :  :  Exit test
:  :  :  :  ENDIF
:  :  :  :  UNTIL all bytes sent
:  :  :  ENDREPEAT
```

```
                : :  ENDDO
                : :  REPEAT
                : :  :  Set timer
                : :  :  IF timer expired AND done set
                : :  :  :  THEN
                : :  :  :  :  Read byte
                : :  :  ENDIF
                : :  :  UNTIL timer expired AND done clear
                : :  ENDREPEAT
          #     : :  IF expected byte count NEQ actual byte count
          #     : :  :  THEN
          #     : :  :  :  Report error
                : :  :  :  Exit test
          #     : :  ENDIF
                : :  REPEAT
                : :  :  Point to expected end packet
                : :  :  IF expected end data NEQ received end data
                : :  :  :  THEN
                : :  :  :  Report end packet data error
                : :  :  ENDIF
                : :  :  UNTIL all data checked
                : :  ENDREPEAT
                : :  Point to read command packet
                : :  REPEAT
                : :  :  Send command packet byte at a time
                : :  :  Set timer
                : :  :  IF timer expired and no response
                : :  :  :  THEN
                : :  :  :  Report no response error
                : :  :  :  Exit test
                : :  :  ENDIF
                : :  :  UNTIL all bytes sent
                : :  ENDREPEAT
                : :  REPEAT
                : :  :  Set timer
                : :  :  IF timer expired AND done set
                : :  :  :  THEN
                : :  :  :  :  Read byte
                : :  :  ENDIF
                : :  :  UNTIL timer expired AND done clear
                : :  ENDREPEAT
          #     : :  IF expected byte count NEQ actual byte count
          #     : :  :  THEN
          #     : :  :  :  Report error
                : :  :  :  Exit test
          #     : :  ENDIF
                : :  REPEAT
                : :  :  Point to expected data packet
                : :  :  REPEAT
                : :  :  :  IF data expected NEQ data received
                : :  :  :  :  THEN
                : :  :  :  :  Report data error
                : :  :  :  :  Exit subtest
                : :  :  :  ENDIF
                : :  :  :  Point to next byte of data
                : :  :  :  UNTIL all data in packet checked
```

```
:  :  :  ENDREPEAT
:  :  :  UNTIL 4 data packets checked
:  :  ENDREPEAT
:  :  Point to expected end packet
:  :  REPEAT
:  :  :  IF data expected NEQ data received
:  :  :  :  THEN
:  :  :  :  Report data error
:  :  :  :  Exit subtest
:  :  :  ENDIF
:  :  :  UNTIL all end packet bytes checked
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST BLOCK WRITE - READ
```

3.7.2.11 RECORD WRITE - READ

This subtest sends the TU58 controller a series of command packets that will cause head switches, forward seeks, reverse seeks and seeks with head switches during read and write operations on records. All data written is read and checked. All end packets are checked as the subtest progresses. If any errors occur they are reported.

Subtest steps:

```
SUBTEST RECORD WRITE - READ
:  DO for count = 1 to 5
:  :  Point to write command packet
:  :  Clear buffer
:  :  REPEAT
:  :  :  Send command packet byte at a time
:  :  :  Set timer
:  :  :  IF timer expired AND no response
:  :  :  :  THEN
:  :  :  :  Report no response error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  UNTIL all bytes sent
:  :  ENDREPEAT
:  :  DO for count = 1 to 2
:  :  :  Wait for continue
:  :  :  Set timer
:  :  :  IF timer expired AND no continue
:  :  :  :  THEN
:  :  :  :  Report no continue error
:  :  :  :  Exit test
:  :  :  ENDIF
:  :  :  Point to data packet
:  :  :  REPEAT
:  :  :  :  Send data packet byte at a time
:  :  :  :  Set timer
```

```
                :  :  :  :  IF timer expired AND no response
                :  :  :  :  :  THEN
                :  :  :  :  :  Report no response error
                :  :  :  :  :  Exit test
                :  :  :  :  ENDIF
                :  :  :  :  UNTIL all bytes sent
                :  :  :  ENDREPEAT
                :  :  ENDDO
                :  :  REPEAT
                :  :  :  Set timer
                :  :  :  IF timer expired AND done set
                :  :  :  :  THEN
                :  :  :  :  :  Read byte
                :  :  :  ENDIF
                :  :  :  UNTIL timer expired AND done clear
                :  :  ENDREPEAT
          #     :  :  IF expected byte count NEQ actual byte count
          #     :  :  :  THEN
          #     :  :  :  :  Report error
                :  :  :  :  Exit test
          #     :  :  ENDIF
                :  :  REPEAT
                :  :  :  Point to expected end packet
                :  :  :  IF expected end data NEQ received end data
                :  :  :  :  THEN
                :  :  :  :  Report end packet data error
                :  :  :  ENDIF
                :  :  :  UNTIL all data checked
                :  :  ENDREPEAT
                :  :  Point to read command packet
                :  :  REPEAT
                :  :  :  Send command packet byte at a time
                :  :  :  Set timer
                :  :  :  IF timer expired and no response
                :  :  :  :  THEN
                :  :  :  :  Report no response error
                :  :  :  :  Exit test
                :  :  :  ENDIF
                :  :  :  UNTIL all bytes sent
                :  :  ENDREPEAT
                :  :  REPEAT
                :  :  :  Set timer
                :  :  :  IF timer expired AND done set
                :  :  :  :  THEN
                :  :  :  :  :  Read byte
                :  :  :  ENDIF
                :  :  :  UNTIL timer expired AND done clear
                :  :  ENDREPEAT
          #     :  :  IF expected byte count NEQ actual byte count
          #     :  :  :  THEN
          #     :  :  :  :  Report error
                :  :  :  :  Exit test
          #     :  :  ENDIF
                :  :  REPEAT
                :  :  :  Point to expected data packet
                :  :  :  REPEAT
```

```
:  :  :  :  IF data expected NEQ data received
:  :  :  :  :  THEN
:  :  :  :  :  Report data error
:  :  :  :  :  Exit subtest
:  :  :  :  ENDIF
:  :  :  :  Point to next byte of data
:  :  :  :  UNTIL all data in packet checked
:  :  :  ENDREPEAT
:  :  :  UNTIL 4 data packets checked
:  :  ENDREPEAT
:  :  Point to expected end packet
:  :  REPEAT
:  :  :  IF data expected NEQ data received
:  :  :  :  THEN
:  :  :  :  Report data error
:  :  :  :  Exit subtest
:  :  :  ENDIF
:  :  :  UNTIL all end packet bytes checked
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST RECORD WRITE - READ
```

3.7.2.12 WRITE RECORD INTERRUPT

This subtest sends the TU58 a command packet to write one record of 128 bytes in interrupt mode. When an interrupt occurs the write service routine sends the next byte. A timeout occurs after sufficient time has passed to send a full packet, at which time an end packet is read and checked. An error is reported if an interrupt fails to occur before the timer expires and the full packet has not been sent. The data packet sent contains alternating bytes of all one's and all zero's. The data will be read and checked for errors in the next subtest.

Subtest steps:

```
SUBTEST WRITE RECORD INTERRUPT
:  Clear buffer
:  Point to write command packet
:  REPEAT
:  :  Send command packet byte at a time
:  :  Set timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  Report no response error
:  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
:  ENDREPEAT
:  Wait for continue
:  Set timer
:  IF timer expired AND no continue
```

```
                :  :  THEN
                :  :  Report no continue error
                :  :  Exit test
                :  ENDIF
                :  Point to data packet
                :  Enable send interrupt
                :  REPEAT
                :  :  Send data packet byte at a time
                :  :  Set timer
                :  :  IF timer expired AND no response
                :  :  :  THEN
                :  :  :  :  DISABLE send interrupt
                :  :  :  :  Report no response error
                :  :  :  :  Exit test
                :  :  ENDIF
                :  :  UNTIL all bytes sent
                :  ENDREPEAT
                :  DISABLE send interrupt
                :  REPEAT
                :  :  Set timer
                :  :  IF timer expired AND done set
                :  :  :  THEN
                :  :  :  :  Read byte
                :  :  ENDIF
                :  :  UNTIL timer expired AND done clear
                :  ENDREPEAT
        #       :  IF expected byte count NEQ actual byte count
        #       :  :  THEN
        #       :  :  :  Report error
                :  :  :  Exit test
        #       :  ENDIF
                :  REPEAT
                :  :  Point to expected end packet
                :  :  IF expected end packet data NEQ received end packet data
                :  :  :  THEN
                :  :  :  Report end packet data error
                :  :  ENDIF
                :  :  UNTIL all data checked
                :  ENDREPEAT
            ENDSUBTEST WRITE RECORD INTERRUPT
```

3.7.2.13 READ RECORD INTERRUPT

This subtest sends the TU58 a command packet to read the last 128 bytes of data that were written in the previous subtest in the interrupt driven mode. A timer is set which should allow sufficient time to read a complete record and end packet. An error is reported if this does not occur within the allotted time. The end packet and data packet are checked for errors. If any errors are encountered they are reported.

Subtest steps:

```
        SUBTEST READ RECORD INTERRUPT
        :  Clear buffer
        :  Point to read command packet
        :  REPEAT
        :  :  Send command packet byte at a time
        :  :  Set timer
        :  :  IF timer expired and no response
        :  :  :  THEN
        :  :  :  Report no response error
        :  :  :  Exit test
        :  :  ENDIF
        :  :  UNTIL all bytes sent
        :  ENDREPEAT
        :  ENABLE read interrupt
        :  SET timer
        :  REPEAT
        :  :  IF interrupt occurs
        :  :  :  THEN
        :  :  :  :  Read byte
        :  :  ENDIF
        :  :  UNTIL timer expired
        :  ENDREPEAT
#       :  IF expected byte count NEQ actual byte count
#       :  :  THEN
        :  :  :  DISABLE read interrupt
#       :  :  :  Report error
        :  :  :  Exit test
#       :  ENDIF
        :  DISABLE read interrupt
        .  Point to expected data packet
        :  REPEAT
        :  :  IF data expected NEQ data received
        :  :  :  THEN
        :  :  :  Report data error
        :  :  :  Exit subtest
        :  :  ENDIF
        :  :  Point to next byte of data
        :  :  UNTIL all data in packet checked
        :  ENDREPEAT
        :  Point to expected end packet
        :  REPEAT
        :  :  IF expected end packet data NEQ received end packet data
        :  :  :  THEN
        :  :  :  Report end packet data error
        :  :  :  Exit subtest
        :  :  ENDIF
        :  :  UNTIL all end packet bytes checked
        :  ENDREPEAT
        ENDSUBTEST READ RECORD INTERRUPT
```

3.7.2.14 IPL CHECK

This subtest checks that the TU58 interrupt occurs at the correct interrupt priority level. The IPL is set one level above that at which a TU58 interrupt should occur. A NOP command is issued to cause the return of an end packet in interrupt mode. If an interrupt occurs an error is reported. The IPL is then decremented and if an interrupt does not occur within a reasonable interval an error is reported. There are no data checks made.

Subtest steps:

```
SUBTEST IPL CHECK
:  Point to NOP command packet
:  DISABLE TU58 Read Interrupt
:  RAISE IPL to prohibit TU58 Read Interrupt
:  REPEAT
:  :  SEND command packet byte at a time
:  :  SET timer
:  :  IF timer expired AND no response
:  :  :  THEN
:  :  :  :  REPORT no response error
:  :  :  :  Exit test
:  :  ENDIF
:  :  UNTIL all bytes sent
:  ENDREPEAT
:  SET timer
:  ENABLE read interrupt
:  IF interrupt occurs before timeout
:  :  THEN
:  :  :  DISABLE read interrupt
:  :  :  REPORT error
:  :  ENDIF
:  DISABLE read interrupt
:  DECR IPL to allow TU58 Read Interrupt
:  SET timer
:  ENABLE read interrupt
:  IF timer expires AND no interrupt
:  :  THEN
:  :  :  DISABLE read interrupt
:  :  :  REPORT error
:  ENDIF
:  DISABLE read interrupt
:  CLEAR IPL
ENDSUBTEST IPL CHECK
```

3.8 ENKAXH: UNIBUS INTERFACE TEST

This module consists of two tests that check the VAX-11/730 Unibus interface (UBI). This includes testing of the interface CSR, MAP's, and different types of data transactions and exercising UBE's. This program runs under the Diagnostic Supervisor and is intended for use on a VAX 11/730 system.

3.8.1 UNIBUS Functional Test

This test will be implemented to run using a Unibus Exerciser (UBE). This test checks the UBI CSR, MAP's and various data transactions using a Unibus Exerciser. The UBE must be set up at address 770000 (Oc- tal) with vector 510 (Octal).

3.8.1.1 CSR

This subtest checks that the UBI CSR register is accessible and that writing a one to the status bits results in the bits remaining cleared. In error reporting the UNUSED bits will be masked out since they are unpredictable. The bits are defined as follows:

CSR

```
##########31 30 17 16 15 14 13 0
        +-----+--------+-----+------------+--------+--------+
        | RDS | Unused | NXM | Map Parity | Wr Err | Unused |
        +-----+--------+-----+------------+--------+--------+
```

Subtest steps:

```
        SUBTEST CSR
        :  Attempt access to CSR
        :  IF unsuccessful
        :  :  THEN REPORT access error
        :  ENDIF
        :  CLEAR Status bits (with read)
        :  IF Status bits not clear
        :  :  THEN REPORT error
        :  ENDIF
        :  WRITE ones to CSR Status Bits
        :  READ CSR
        :  IF Status bits not clear
        :  :  THEN REPORT error
        :  ENDIF
        ENDSUBTEST CSR
Errors:
```

3.8.1.2 Map Data Bus

This subtest checks the integrity of the map data bus by floating a one through a field of zeros and a zero through a field of ones. If all bits can toggle from a one to a zero then the data bus is considered to be good. Only writable bits (31,25,<14:0>) are tested. The remain- ing bits are ignored. In error reporting the unused bits will be masked out since they are unpredictable.

Subtest steps:

```
SUBTEST Map Data Bus
:  REPEAT
:  :  WRITE data pattern to first Map Entry
:  :  READ pattern back
:  :  IF error
:  :  :  THEN REPORT error
:  :  Update pattern
:  :  UNTIL all patterns written
:  ENDREPEAT
ENDSUBTEST Map Data Bus
```

Errors:

3.8.1.3 Map Address Bus

This subtest checks that there are no map address bits stuck or shorted. A unique pattern is written into each map entry at maps num- bered 0, 1, 2, 4, 8, 16, 32, 64, and 128, (so that each address bit assumes both a zero and a one). Reading each of these map entries after all patterns are written will reveal any stuck or short- ed address bus bits because the last pattern written to a doubly ad- dressed map will overwrite the earlier pattern written. Unfortunately a data bit fault in one of the RAMs in a particular map entry may point erroneously to an address bus bit fault. Therefore, to make this test insensitive to a limited number of RAM data bit failures, it is necessary to encode each value prior to storing. The Reed-Muller (16,4) error-correcting code is used. In error reporting the unused bits are masked out since they are unpredictable.

EXPLANATION OF REED-MULLER ERROR-CORRECTING CODE

The basis of the Reed-Muller (16,4) coding space comprises the vectors:

```
I       1111111111111111
V1      0101010101010101
V2      0011001100110011
V3      0000111100001111
```

A word is encoded using the following expression:

CODEWORD = G0*I XOR G1*V1 XOR G2*V2 XOR G3*V3

where G0 is the least significant bit and G3 the most

significant bit.

Each codeword has 8 disjoint redundant relations due to the nature of the encoding vectors. In V1 the redundant relations are the bit pairs (31,30), (29,28), (27,26), etc. In V2, they are the bit pairs (31,29), (30,28), (27,25), etc. In V3, the relations are (31,27), (30,26), (29,25), etc. Thus, there are three sets of redundant relations in each codeword, and therefore, each bit is represented by 8 disjoint relations. To decode a particular bit, the value of each of the 8 redundant rela- tions is calculated. The position of the bit in the decoded word de- termines which bit pairs will be used. The original word is represent- as:

```
+--+--+--+--+
|G3|G2|G1|G0|
+--+--+--+--+
```

The bits G0 - G3 are referred to as information bits. The LSB has no inherent redundant relations, since it is represented by the product of itself and the identity vector. Here it is convenient to introduce the notion of pair separation. The pair separation for the redundant relations representing G1 is 1; for G2, 2, and for G3, 4.

After decoding the value of each redundant relation for a particular bit, these values are summed. If there was no corruption of the code- word, this sum will be 0 or 8, corresponding to an original value of 0 or 1. If there was any corruption, a majority decision must be made, i.e., if the sum is 4 or greater, the bit Gn is assumed to have been a 1, otherwise, it is assumed 0.

After decoding the three MSBs, the LSB is decoded by eliminating each of the terms G1*V1, G2*V2, and G3*V3, leaving simply G0*I. If there had been no corruption, this term would be 16 0s or 16 1s, representing 0 or 1 respectively. Otherwise, another majority decision must be made, i.e., if 8 or more bits are set, G0 is a 1, else, it is 0.

Subtest steps:

```
SUBTEST Map Address Bus
:  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
:  :  WRITE pattern to Map location
:  ENDDO
:  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
:  :  READ pattern from Map location
:  ENDDO
:  DECODE Data retrieved
:  IF error
:  :  THEN REPORT error
:  ENDIF
ENDSUBTEST Map Address Bus
```

Errors:

3.8.1.4 Map Entry

This subtest checks that all readable/writeable bits of each map entry are not stuck at zero or one. This is accomplished by writing all zeros to a map entry, then reading it to make certain the bits are not stuck at 1. Then a pattern of all ones is written to the entry and read again to verify that it contains all ones to check that there are no bits stuck at zero. This is repeated for all map entries. Again, in error reporting the unused bits will be masked out since they are unpredictable.

Subtest steps:

```
SUBTEST Map Entry
:  REPEAT
:  :  CLEAR Map Entry writable bits
:  :  IF Map Entry not clear
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  WRITE Mask to Map Entry
:  :  IF Map Entry NEQ Mask
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  CLEAR Map Entry writable bits
:  :  IF Map Entry not clear
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  UNTIL all Map Entries tested
:  ENDREPEAT
ENDSUBTEST Map Entry
```

Errors:

3.8.1.5 CPU Read/Write

This subtest checks the ability of the UBI to handle CPU reads and writes to the unibus. The UBE data and address registers are written and read, checking for stuck at one or zero faults. The following data patterns are used to perform this test:

```
01^1010101010101
1010101010101010
0011001100110011
0000111100001111
0000000011111111
```

to check for word writes, and

```
01010101
10101010
00110011
00001111
```

to check for byte writes.

Subtest steps:

```
SUBTEST CPU Read/Write
:  PROBE UBE Data Register
:  IF not present
:  :  THEN REPORT error
:  ENDIF
:  REPEAT
:  :  WRITE word pat ern to Address register
:  :  READ Address register
:  :  IF error
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  UNTIL patterns exhausted
:  ENDREPEAT
:  REPEAT
   :  WRITE word pattern to Data register
:  :  READ Data register
:  :  IF error
:  :  :  THEN DO
:  :  :  :  REPORT error
:  :  :  :  EXIT Test
:  :  :  ENDDO
:  :  ENDIF
:  :  UNTIL patterns exhausted
:  ENDREPEAT
:  REPEAT
:  :  WRITE byte pattern to Data register high byte
:  :  WRITE byte pattern to Data register low byte
:  :  READ Data register high byte
:  :  IF error
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  READ Data register low byte
:  :  IF error
:  :  :  THEN REPORT error
:  :  ENDIF
·  :  UNTIL patterns exhausted
:  ENDREPEAT
ENDSUBTEST CPU Read/Write
```

Errors:

3.8.1.6 Interrupts

This subtest checks the ability of the UBI to handle interrupts at BR4- BR7, using the UBE to initiate interrupts at each BR level. The service routine is set up to set flags if the interrupt occurs or if error conditions (at wrong IPL, wrong vector) exist. Any errors are reported.

Subtest steps:

```
SUBTEST INTERRUPTS
```

```
        : Set IPL at BR7 (17(X)) :  Set up UBE to interrupt upon
    becoming bus master at all 4 levels
        : DO FOR BR level = {4,5,6,7}
        : : Set IPL to prohibit BR interrupt
        : : IF interrupt occurs
        : : : THEN REPORT error
        : : ENDIF
        : : Decrement IPL
        : : If interrupt does not occur OR at wrong IPL OR wrong
    vector
        : : : THEN REPORT error
        : : ENDIF
        : ENDDO
        ENDSUBTEST INTERRUPTS
```

Errors:

3.8.1.7 DATI 1

This subtest checks the ability to execute a Unibus read (DATI) transaction. The UBE Data register is cleared with a CPU write and a unique data pattern is written to a main memory location. All usable Map Registers are validated and set up to access the main memory location. A DATI is executed and after a nominal wait, the UBE Data register is read. Transfers are made from longword- and word-aligned locations.

Subtest steps:

```
        SUBTEST DATI 1
        : Size UNIBUS
        : IF no usable map entries
        : : THEN DO
        : : : PRINT exit warning
        : : : EXIT test
        : : ENDDO
        : ENDIF
        : Validate and point all usable Map Entries to main memory
    address
        : DO FOR Alignment = {word, longword}
        : : Load UBE address registers
        : : Execute DATI
        : : Wait
        : : IF Error
        : : : THEN REPORT Error
        : : ENDIF
        : ENDDO
        ENDSUBTEST DATI 1
```

Errors:

3.8.1.8 Map Select

This subtest checks the ability of the UBI to correctly select a map from the page frame of the unibus address. The type of fault that could occur here is a stuck or shorted Unibus address bit, which would cause the wrong map entry to be selected. In order to detect this type of fault, the following steps are executed:

Subtest steps:

```
SUBTEST MAP SELECT
:  Invalidate all Maps
:  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
:  :  IF Map Entry usable
:  :  :  THEN DO
:  :  :  :  VALIDATE Map Entry
:  :  :  :  CLEAR UBE Data register
:  :  :  :  EXECUTE DATI
:  :  :  :  WAIT
:  :  :  :  IF error
:  :  :  :  :  THEN REPORT error
:  :  :  :  ENDIF
:  :  :  :  INVALIDATE Map Entry
:  :  :  ENDDO
:  :  ENDIF
:  ENDDO
ENDSUBTEST MAP SELECT
```

Errors:

3.8.1.9 DATI 2

This subtest checks the ability of the UBI to perform a DATI transaction and the integrity of the data path. The transactions are made from longword and word aligned addresses. The following data patterns are used:

0101010101010101
1010101010101010
0011001100110011
0000111100001111
0000000011111111

Subtest steps:

```
SUBTEST DATI 2
:  Invalidate all maps but one
:  DO for Alignment = {longword,word}
:  :  REPEAT
:  :  :  EXECUTE DATI
:  :  :  WAIT
:  :  :  IF error
:  :  :  :  THEN REPORT error
```

```
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST DATI 2
```

Errors:

3.8.1.10 DATO

This subtest checks the ability of the UBI to perform a DATO transaction. This test is similar to the DATI 2 subtest, with the data transfer in the opposite direction.

Subtest steps:

```
SUBTEST DATO
:  Load UBE address registers
:  DO FOR Alignment = {word, longword}
:  :  REPEAT
:  :  :  EXECUTE DATO
:  :  :  WAIT
:  :  :  IF error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST DATO
```

Errors:

3.8.1.11 DATOB

This subtest checks the ability of the UBI to perform a DATOB transac- tion. The transactions are made from longword- and word-aligned main memory addresses. The following data patterns are used:

```
01010101
10101010
00110011
00001111
```

Subtest steps:

```
SUBTEST DATOB
:  DO FOR Alignment = {word, longword}
:  CLEAR Main Memory location
:  :  REPEAT
```

```
:  :  :  Load pattern into UBE Data Register
:  :  :  Initiate DATOB
:  :  :  WAIT
:  :  :  IF error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST DATOB
```

Errors:

3.8.1.12 DATIP/DATO

This subtest checks the ability of the UBI to perform a DATIP/DATO transaction. This test is similar to the DATI 2 subtest, with the addition of checking that the DATO portion of the transaction also works.

Subtest steps:

```
SUBTEST DATIP/DATO
:  Invalidate all maps but one
:  Load UBE address registers
:  DO FOR Alignment = {word, longword}
:  :  REPEAT
:  :  :  EXECUTE DATIP/DATO
:  :  :  WAIT
:  :  :  IF error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST DATIP/DATO
```

Errors:

3.8.1.13 Address Offset DATI

This subtest checks the ability of the UBI to perform a DATI transaction in address offset mode.

Subtest steps:

```
SUBTEST Address Offset DATI
:  Set Byte offset bit in first valid Map Entry
:  DO for Alignment = {longword,word}
```

```
:  :  REPEAT
:  :  :  EXECUTE DATI
:  :  :  WAIT
:  :  :  IF Execution error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  IF Data error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST Address Offset DATI
```

Errors:

3.8.1.14 Address Offset DATO

This subtest checks the ability of the UBI to perform a DATO transaction in address offset mode.

Subtest steps:

```
SUBTEST Address Offset DATO
:  Invalidate all maps but one
:  DO FOR Alignment = {word, longword}
:  :  REPEAT
:  :  :  Initiate DATO
:  :  :  WAIT
:  :  :  IF Execution error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  IF Data error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST Address Offset DATO
```

Errors:

3.8.1.15 Address Offset DATOB

This subtest checks the ability of the UBI to perform a DATOB transaction in address offset mode.

Subtest steps:

```
SUBTEST Address Offset DATOB
```

```
:  Invalidate all maps but one
:  DO FOR Alignment = {word, longword}
:  :  REPEAT
:  :  :  Initiate DATOB
:  :  :  WAIT
:  :  :  IF Execution error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  IF Data error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST Address Offset DATOB
```

Errors:

3.8.1.16 Address Offset DATIP/DATO

This subtest checks the ability of the UBI to perform a DATIP/DATO transaction in address offset mode.

Subtest steps:

```
SUBTEST Address Offset DATIP/DATO
:  Invalidate all maps but one
:  DO FOR Alignment = {word, longword}
:  :  REPEAT
:  :  :  Initiate DATIP/DATO
:  :  :  WAIT
:  :  :  IF Execution error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  IF Data error
:  :  :  :  THEN REPORT error
:  :  :  ENDIF
:  :  :  UNTIL patterns exhausted
:  :  ENDREPEAT
:  ENDDO
ENDSUBTEST Address Offset DATIP/DATO
```

Errors:

3.8.1.17 Map Function

This subtest checks the ability of the UBI to correctly handle transactions using all useable map registers. 512 map registers are potentially useable, but of that set some may not be useable due to

the presence of Unibus Memory. Therefore, before the testing of the maps is started, the Unibus is sized for memory to ascertain which of the maps are useable. A bitvector is generated by the Unibus sizer routine. If a map is useable, its bit in the bitvector is set. For each map that is available for testing, a unique pattern is set up, executed and verified.

Subtest steps:

```
SUBTEST MAP FUNCTION
:  Invalidate all Maps
:  REPEAT
:  :  IF Map Entry usable
:  :  :  THEN DO
:  :  :  :  VALIDATE Map Entry
:  :  :  :  CLEAR UBE Data register
:  :  :  :  EXECUTE DATI
:  :  :  :  WAIT
:  :  :  :  IF error
:  :  :  :  :  THEN REPORT error
:  :  :  :  ENDIF
:  :  :  :  INVALIDATE Map Entry
:  :  :  ENDDO
:  :  ENDIF
:  :  UNTIL All Usable Map Entries exhausted
:  ENDREPEAT
ENDSUBTEST MAP FUNCTION
```

Errors:

3.8.1.18 Address Offset Map Function

This subtest checks the ability of the UBI to correctly handle byte-off- set transactions using all useable map registers. 512 map registers are potentially useable, but of that set some may not be useable due to the presence of Unibus Memory. Therefore, before the testing of the maps is started, the Unibus is sized for memory to ascertain which of the maps are useable. A bitvector is generated by the Unibus sizer routine. If a map is useable, its bit in the bitvector is set. For each map that is available for testing, a DATI with a unique pattern is set up, executed and verified.

Subtest steps:

```
SUBTEST ADDRESS OFFSET MAP FUNCTION
:  Invalidate all Maps
:  REPEAT
:  :  IF Map Entry usable
:  :  :  THEN DO
:  :  :  :  VALIDATE Map Entry
:  :  :  :  Set up Map Entry in Byte-offset mode
:  :  :  :  CLEAR UBE Data register
```

```
:  :  :  :  EXECUTE DATI
:  :  :  :  WAIT
:  :  :  :  IF error
:  :  :  :  :  THEN REPORT error
:  :  :  :  ENDIF
:  :  :  :  INVALIDATE Map Entry
:  :  :  ENDDO
:  :  ENDIF
:  :  UNTIL All Usable Map Entries exhausted
:  ENDREPEAT
ENDSUBTEST ADDRESS OFFSET MAP FUNCTION
```

Errors:

3.8.1.19 Page Boundary Write Error

This subtest checks that a byte offset write across a page boundary will cause an error if the second page is invalidated.

Subtest steps:

```
SUBTEST Page Boundary Write Error
:  Set up first valid Map Entry to point to valid page
:  Set up DATO to last byte in this page in byte-offset mode
:  Invalidate the page immediately following
:  Initiate DATO
:  IF no error condition generated
:  :  THEN REPORT error
:  ENDIF
:  IF UBI CSR not cleared when read
:  :  THEN REPORT error
:  ENDIF
:  IF unexpected SSYN asserted
:  :  THEN REPORT error
:  ENDIF
ENDSUBTEST Page Boundary Write Error
```

Errors:

3.8.1.20 Map Parity Error

This subtest checks that when the Unibus PB is asserted (simulating a Map Parity Error) and a transfer is attempted, the Map Parity Error bit in the UBI CSR will set and that it will clear when read.

Subtest steps:

```
SUBTEST Map Parity Error
```

```
:  Set FORCE TB PARITY in Memory CSR
:  LOAD Map Entry
:  CLEAR TB Parity bit
:  EXECUTE DATI
:  WAIT
:  IF Map Parity Error bit not set
:  :  THEN REPORT error
:  ENDIF
:  IF Map Parity Error did not clear when read
:  :  THEN REPORT error
:  ENDIF
:  IF unexpected SSYN asserted
:  :  THEN REPORT error
:  ENDIF
ENDSUBTEST Map Parity Error
```

Errors:

3.8.1.21 NXM Error

This subtest checks that when the UBE attempts to access a non-existent memory location, the NXM Error bit in the UBI CSR will set and that it will clear when read.

Subtest steps:

```
SUBTEST NXM Error
:  Set up DATI to access NXM address
:  Initiate DATI
:  WAIT
:  IF NXM bit did not set
:  :  THEN REPORT error
:  ENDIF
:  IF NXM bit did not clear when read
:  :  THEN REPORT error
:  ENDIF
:  IF unexpected SSYN asserted
:  :  THEN REPORT error
:  ENDIF
ENDSUBTEST NXM Error
```

Errors:

3.8.1.22 Map Invalid

This subtest checks that the UBI refuses to issue SSYN if the UBE at- tempts to access a map entry which has the valid bit turned off.

Subtest steps:

```
SUBTEST Map Invalid
:  Invalidate first valid Map Entry
:  Set up DATI to reference first valid Map Entry
:  CLEAR UBE Data register
:  Load pattern into memory location
:  Execute DATI
:  WAIT
:  IF SSYN received
:  :  THEN REPORT no-timeout error
:  ENDIF
ENDSUBTEST Map Invalid
```

Errors:

3.8.2 UBE Multi-transfer Test

This test supports up to four UBE's if they are attached and selected for testing. It consists of two subtests, the first is a multi-transfer subtest that starts traffic on the bus simultaneously and the second is a multi-transfer subtest that will have UBE's doing various trans- fers and/or interrupts at different request levels. Both subtests are designed to check CPU handling capabilities. This test is modeled after the existing PDP-11 Unibus System Exerciser, CZKUAE0. Note that the UBEs MUST be addressed and vectored as specified in the BUS EXERCISER ENGINEERING SPECIFICATION, i.e., the first device must be at address 777000 (octal) with vector 510 (octal), the second at address 777020 (octal) with vector 514 (octal), etc.

3.8.2.1 UBE Multi-transfer 1

This subtest will attempt to create traffic on the Unibus and check that the CPU can handle the data transactions without errors. All se- lected UBE's are set up and started simultaneously to cause the bus traffic. The functions cycle through the map entries using byte offset mode and are set up as follows:

UBE 0 - DATIP/DATOB single word BR6 transfer with a rotate data left after DATIP, interrupt on done, 2000 cycles.

UBE 1 - DATIP/DATO single word NPR transfer without rotate, interrupt on done BR7, 2000 cycles.

UBE 2 - DATO single word NPR transfer, interrupt on done BR7, 1000 cycles.

UBE 3 - DATO single word BR5 transfer, interrupt on done BR5, 1000 cycles.

Subtest steps:

```
BGNSUBTEST Multitransfer 1
:  DO FOR UBE Number = {0,1,2,3}
:  :  PROBE Data Register
:  :  IF NO ACCESS
:  :  :  THEN DO
:  :  :  :  REPORT ERROR
:  :  :  :  EXIT Test
:  :  :  ENDDO
:  :  ENDIF
:  ENDDO
:  DO FOR UBE Number = {0,1,2,3}
:  :  Set up UBE transfer
:  ENDDO
:  START Timer
:  SET Simultaneous GO bit
:  WAIT UNTIL All transfers stopped
:  DO FOR UBE Number = {0,1,2,3}
:  :  IF Data or Execution error
:  :  :  THEN REPORT error
:  :  ENDIF
:  ENDDO
ENDSUBTEST Multitransfer 1
```

Errors:

3.8.2.2 UBE Multi-transfer 2

This subtest will have all selected UBE's doing various transfer and/or interrupts at different request levels to further check CPU handling ablity. The functions cycle through the map entries using byte offset mode with each data pattern being rotated after each block transfer. The following functions are performed:

- UBE 0 - DATOB double word NPR transfers, data from cycle count register, interrupt on done BR6, 2000 cycles.
- UBE 1 - DATIP/DATO single word NPR transfers, with rotate after DATIP, interrupt on done BR7, 1000 cycles.
- UBE 2 - DATI double word BR5 transfers, 1000 cycles, interrupt on done BR5.
- UBE 3 - DATI single word BR7 transfers with an interrupt after each transfer, 3000 cycles.

Subtest steps:

```
BGNSUBTEST Multitransfer 2
:  DO FOR UBE Number = {0,1,2,3}
:  :  Set up UBE transfer
:  ENDDO
:  START Timer
:  SET Simultaneous GO bit
:  WAIT UNTIL All transfers stopped
```

```
        :  DO FOR UBE Number = {0,1,2,3}
        :  :  IF Data or Execution error
        :  :  :  THEN REPORT error
        :  :  ENDIF
        :  ENDDO
        ENDSUBTEST Multitransfer 2
```

Errors:

3.9 ENKAXI: MEMORY TEST

This module does a quick test of main memory for the VAX 11/730 CPU. First memory is sized and compared against the memory size stored in CSR2 by the micro code and then a march test is done on all existing memory above that which is allocated to the Diagnostic Supervisor or this diagnostic. Any gaps found

3.9.0.1 Memory Size

This subtest sizes memory and then compares memory found to the size stored in memory CSR2.

Subtest steps:

```
SUBTEST Memory Size
:  Get expected size from ka p table
:  Read memory CSR3 (memory found at power up by micro code)
:  Size memory
:  Compare two sizes
:  IF size is not the same
:  :  THEN
:  :  Report Memory size error
:  ENDIF
ENDSUBTEST Memory Size
```

3.9.0.2 Data Path

This subtest checks the data path for stuck at one, stuck at zero and shorted data lines. This is accomplished by floating a one and a zero through the data. If an error is detected it is reported with information describing the nature of the error.

Subtest steps:

```
SUBTEST Data Path
:  IF REPORT SB ERRORS
:  :  THEN SET Memory CSR1 Disable ECC
```

```
:  ENDIF
:  Initialize data pattern
:  REPEAT
:  :  Write pattern
:  :  Read pattern
:  :  IF pattern incorrect
:  :  :  THEN
:  :  :  Report error
:  :  ENDIF
:  :  Update pattern
:  :  UNTIL all patterns done
:  ENDREPEAT
ENDSUBTEST Data Path
```

3.9.0.3 Memory March

This subtest does a march thru all of memory above that which is allocated to the supervisor. A background pattern of all zeros is written into memory. Starting with location @A_FREE_MEM thru all available memory, the following is done. First, the background pattern is checked (i.e., checked for stuck or shorted address lines), then a pattern of all ones is written and checked. A reverse march is then executed, starting at the highest usable memory location, checking that the old pattern of all ones is still present (again, checking for stuck or shorted address lines) and loading a new pattern of all zeroes. The new pattern is also checked and the address is then decremented. This process assures also that there are no stuck data bits at any location. If a NXM machine check occurs while accessing expected good memory an error is reported. This test skips over "holes" in memory, using the map generated in the sizing routine.

Subtest steps:

```
SUBTEST Memory March
:  DO FOR each physical memory location beyond supervisor
:  :  Clear locations under test
:  ENDDO
:  DO FOR each physical memory location beyond supervisor
:  :  IF data error
:  :  :  THEN REPORT Error
:  :  ENDIF
:  :  SET all bits at location
:  ENDDO
:  REVERSE march direction
:  DO FOR each physical memory location beyond supervisor
:  :  IF data error
:  :  :  THEN REPORT error
:  :  ENDIF
:  :  CLEAR location
:  :  IF location not clear
:  :  :  THEN REPORT Error
```

```
:  :  ENDIF
:  ENDDO
ENDSUBTEST Memory March
```

3.10 ENKAXJ: FAST CONTROL STORE INTERRUPT

It was agreed at the Functional Specification Review that this module will not be implemented. The microcode requirements of the Fast Interrupt feature make testing this function not feasible. Possibly this issue could be addressed by IDC macro or micro diagnostics.

3.11 ENKAXK: LDPCTX (Bliss Module)

This module sets up each test case. The only legal mode for a LDPCTX instruction is kernel. A LDPCTX can only execute of a kernel or interrupt stack and always ends on the kernel stack. This module also sets up the initial and expected data for the Process Control Block and the registers involved are loaded with bakground data. This module also contains the error print and data compare routines for test verification.

3.12 ENKAXL: LDPCTX (Macro Module)

This module contains the control program which uses data set up in ENKAXK to run the different test cases. The routine Execute is called every test case and checks are made at the end of Execute to make sure the instruction is executed correctly.

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
ENKAX1	1-3	1961	WRKD$0:[PAN.ENKAX]ENKAX1.OBJ;1	4-AUG-1983 14:18	VAX-11 Macro V03-00
ENKAXA	1-3	1576	WRKD$0:[PAN.ENKAX]ENKAXA.OBJ;2	3-AUG-1983 13:42	VAX-11 Macro V03-00
ENKAXC	1-3	2317	WRKD$0:[PAN.ENKAX]ENKAXC.OBJ;2	3-AUG-1983 13:42	VAX-11 Macro V03-00
ENKAXD	1-3	9189	WRKD$0:[PAN.ENKAX]ENKAXD.OBJ;2	3-AUG-1983 13:43	VAX-11 Macro V03-00
ENKAXF	1-3	3204	WRKD$0:[PAN.ENKAX]ENKAXF.OBJ;2	3-AUG-1983 13:46	VAX-11 Macro V03-00
ENKAXG	1-3	18222	WRKD$0:[PAN.ENKAX]ENKAXG.OBJ;1	4-AUG-1983 11:08	VAX-11 Macro V03-00
ENKAXH	1-3	15423	WRKD$0:[PAN.ENKAX]ENKAXH.OBJ;4	3-AUG-1983 14:13	VAX-11 Macro V03-00
ENKAXI	1-3	2400	WRKD$0:[PAN.ENKAX]ENKAXI.OBJ;2	3-AUG-1983 13:56	VAX-11 Macro V03-00
ENKAXK	1.3	3140	WRKD$0:[PAN.ENKAX]ENKAXK.OBJ;3	4-Aug-1983 13:42	VAX-11 Bliss-32 V3-713
ENKAXL	1-3	1400	WRKD$0:[PAN.ENKAX]ENKAXL.OBJ;2	3-AUG-1983 13:59	VAX-11 Macro V03-00
DS	0	0	SYS$SYSROOT:[SYSLIB]DS.STB;1	5-JAN-1983 15:27	VAX-11 Linker V3A-16

Program Section Synopsis

Psect Name	Module Name	Base	End	Length		Align	Attributes
$HEADER		00000200	00000291	00000092	(146.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	ENKAX1	00000200	00000291	00000092	(146.)	PAGE 9	
OWN		00000294	000002A3	00000010	(16.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	ENKAXK	00000294	000002A3	00000010	(16.)	LONG 2	
$PLIT$		000002A4	0000058F	000002EC	(748.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	ENKAXK	000002A4	0000058F	000002EC	(748.)	LONG 2	
$TSTCNT		00000590	00000593	00000004	(4.)	LONG 2	NOPIC,USR,OVR,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	ENKAXA	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXC	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXD	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXF	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXG	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXH	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXI	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXK	00000590	00000593	00000004	(4.)	LONG 2	
	ENKAXL	00000590	00000593	00000004	(4.)	LONG 2	
. BLANK .		00000594	000007DF	0000024C	(588.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAXL	00000594	000007DF	0000024C	(588.)	BYTE 0	
ARGLIST		000007E0	00000B1B	0000033C	(828.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXA	000007E0	00000803	00000024	(36.)	LONG 2	
	ENKAXC	00000804	00000827	00000024	(36.)	LONG 2	
	ENKAXD	00000828	000008DB	000000B4	(180.)	LONG 2	
	ENKAXF	000008DC	0000090B	00000030	(48.)	LONG 2	
	ENKAXG	0000090C	000009D7	000000CC	(204.)	LONG 2	
	ENKAXH	000009D8	00000AF7	00000120	(288.)	LONG 2	
	ENKAXI	00000AF8	00000B1B	00000024	(36.)	LONG 2	
CLEANUP		00000B1C	00000C0E	000000F3	(243.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAX1	00000B1C	00000C0E	000000F3	(243.)	LONG 2	
CODE		00000C10	00001293	00000684	(1668.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXK	00000C10	00001293	00000684	(1668.)	LONG 2	
DATA		00001400	000015C4	000001C5	(453.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC

Psect Name	Module Name	Base	End	Length		Align	Attributes
DATA		00001400	000015C4	000001C5	(453.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAX1	00001400	000015C4	000001C5	(453.)	PAGE 9	
DISPATCH		000015C8	000016CF	00000108	(264.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	ENKAXA	000015C8	000015DF	00000018	(24.)	LONG 2	
	ENKAXC	000015E0	000015F7	00000018	(24.)	LONG 2	
	ENKAXD	000015F8	00001627	00000030	(48.)	LONG 2	
	ENKAXF	00001628	0000163F	00000018	(24.)	LONG 2	
	ENKAXG	00001640	0000166F	00000030	(48.)	LONG 2	
	ENKAXH	00001670	0000169F	00000030	(48.)	LONG 2	
	ENKAXI	000016A0	00001687	00000018	(24.)	LONG 2	
	ENKAXL	00001688	000016CF	00000018	(24.)	LONG 2	
DISPATCH_X		000016D0	000016E7	00000018	(24.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD,NOWRT,NOVEC
	ENKAX1	000016D0	000016E7	00000018	(24.)	LONG 2	
ENKAX1		00001800	0000198E	0000018F	(399.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAX1	00001800	0000198E	0000018F	(399.)	PAGE 9	
ENKAXA		00001A00	00001D10	00000311	(785.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXA	00001A00	00001D10	00000311	(785.)	PAGE 9	
ENKAXC		00001E00	0000244E	0000064F	(1615.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXC	00001E00	0000244E	0000064F	(1615.)	PAGE 9	
ENKAXD		00002450	000031FC	00000DAD	(3501.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAXD	00002450	000031FC	00000DAD	(3501.)	LONG 2	
ENKAXF		00003200	00003A4D	0000084E	(2126.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXF	00003200	00003A4D	0000084E	(2126.)	PAGE 9	
ENKAXG		00003C00	000058A4	00001CA5	(7333.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXG	00003C00	000058A4	00001CA5	(7333.)	PAGE 9	
ENKAXH		00005A00	000074B4	00001AB5	(6837.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXH	00005A00	000074B4	00001AB5	(6837.)	PAGE 9	
ENKAXI		00007600	00007B65	00000566	(1382.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXI	00007600	00007B65	00000566	(1382.)	PAGE 9	
GLOBALS		00007B68	00007E27	000002C0	(704.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	ENKAXK	00007B68	00007E27	000002C0	(704.)	LONG 2	
INITIALIZE		00007E28	00008091	0000026A	(618.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAX1	00007E28	00008091	0000026A	(618.)	LONG 2	
SUMMARY		00008094	000080E1	0000004E	(78.)	LONG 2	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ENKAX1	00008094	000080E1	0000004E	(78.)	LONG 2	
TEST_001		00008200	000084D6	000002D7	(727.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXA	00008200	000084D6	000002D7	(727.)	PAGE 9	
TEST_002		00008600	0000887D	0000027E	(638.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXC	00008600	0000887D	0000027E	(638.)	PAGE 9	
TEST_003		00008A00	00009567	00000B68	(2920.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXD	00008A00	00009567	00000B68	(2920.)	PAGE 9	
TEST_004		00009600	00009FE7	000009E8	(2536.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXD	00009600	00009FE7	000009E8	(2536.)	PAGE 9	
TEST_005		0000A000	0000A3E9	000003EA	(1002.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXF	0000A000	0000A3E9	000003EA	(1002.)	PAGE 9	
TEST_006		0000A400	0000A910	00000511	(1297.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXG	0000A400	0000A910	00000511	(1297.)	PAGE 9	
TEST_007		0000AA00	0000CE77	00002478	(9336.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXG	0000AA00	0000CE77	00002478	(9336.)	PAGE 9	
TEST_008		0000D000	0000E9BE	000019BF	(6591.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXH	0000D000	0000E9BE	000019BF	(6591.)	PAGE 9	
TEST_009		0000EA00	0000F076	00000677	(1655.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC

Psect Name	Module Name	Base	End	Length		Align	Attributes
TEST_009		0000EA00	0000F076	00000677	(1655.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXH	0000EA00	0000F076	00000677	(1655.)	PAGE 9	
TEST_010		0000F200	0000F5B9	000003BA	(954.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXI	0000F200	0000F5B9	000003BA	(954.)	PAGE 9	
TEST_011		0000F600	0000F90F	00000310	(784.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD,NOWRT,NOVEC
	ENKAXL	0000F600	0000F90F	00000310	(784.)	PAGE 9	

Symbols By Name

Symbol	Value	Symbol	Value	Symbol	Value
$ENV	00000001	CHI$V_CHNINT	00000000	CHS$M_PGMHDE	00000800
$MO	00000001	CHI$V_DEVINT	00000001	CHS$M_SYSERR	00000001
ACTUAL_EXCEPT	00007B68-R	CHI$V_IPL	00000002	CHS$M_SYSMEM	00000200
ACTUAL_PCB	00007CA0-R	CHM$_FORWARD	00000000	CHS$M_SYSSBI	00000400
ACTUAL_PSL	00007E20-R	CHM$_INVALIDATE	00000001	CHS$V_BUSIC	00000017
ACTUAL_REG	00007DC0-R	CHM$_MAP	00000002	CHS$V_BUSINIT	00000016
ALL	00000008	CHM$_MFWDN	00000002	CHS$V_BUSNXM	0000001B
A_BUFFER_BEGIN	00001800-R	CHM$_MFWDNO	0000000A	CHS$V_BUSPDN	00000018
A_BUFFER_END	00001804-R	CHM$_MFWDV	00000003	CHS$V_CHNDPE	00000006
A_FREE_MEM	00001848-R	CHM$_MFWDVO	0000000B	CHS$V_CHNERR	00000001
A_HALT_BUFFER	00001810-R	CHM$_MREVN	00000006	CHS$V_CHNMPE	00000007
A_MCHK_BUFFER	00001818-R	CHM$_MREVNO	0000000E	CHS$V_CHPFOT	00000008
A_PSLMNE	00001915-R	CHM$_MREVV	00000007	CHS$V_DEVBUS	00000004
A_PWR_BUFFER	00001808-R	CHM$_MREVVO	0000000F	CHS$V_DEVERR	00000002
A_UBE_BUFFER_0	00001828-R	CHM$_NFWDN	00000000	CHS$V_DEVTO	00000005
A_UBE_BUFFER_1	00001830-R	CHM$_NREVN	00000004	CHS$V_MBAATN	00000014
A_UBE_BUFFER_2	00001838-R	CHM$_OFFSET	00000008	CHS$V_MBACPE	00000015
A_UBE_BUFFER_3	00001840-R	CHM$_REVERSE	00000004	CHS$V_MBADTB	00000012
A_WCS_BUFFER	00001820-R	CHS$M_BUSIC	00800000	CHS$V_MBADTC	00000013
BE0_UNIT_NO	0000187D-R	CHS$M_BUSINIT	00400000	CHS$V_MBAEXC	00000010
BE1_UNIT_NO	00001881-R	CHS$M_BUSNXM	08000000	CHS$V_MBANED	00000011
BE2_UNIT_NO	00001885-R	CHS$M_BUSPDN	01000000	CHS$V_MBAWCKLWR	00000019
BE3_UNIT_NO	00001889-R	CHS$M_CHNDPE	00000040	CHS$V_MBAWCKUPR	0000001A
B_TU58_DRIVE	0000186C-R	CHS$M_CHNERR	00000002	CHS$V_PGMERR	00000003
CHC$_ABORT	00C00002	CHS$M_CHNMPE	00000080	CHS$V_PGMHDE	0000000B
CHC$_CLEAR	00000006	CHS$M_CHPFOT	00000100	CHS$V_SYSERR	00000000
CHC$_CLRDFT	00000009	CHS$M_DEVBUS	00000010	CHS$V_SYSMEM	00000009
CHC$_DSINT	00000005	CHS$M_DEVERR	00000004	CHS$V_SYSSBI	0000000A
CHC$_ENINT	00000004	CHS$M_DEVTO	00000020	CPU$V_COMET	00000002
CHC$_INITA	00000000	CHS$M_ERRANY	0000000F	CPU$V_HS	00000000
CHC$_INITB	00000001	CHS$M_MBAATN	00100000	CPU$V_STAR	00000001
CHC$_PURGE	00000003	CHS$M_MBACPE	00200000	DEFAULT	00000000
CHC$_SETDFT	00000008	CHS$M_MBADTB	00040000	DS$ABORT	00010020
CHC$_STATUS	00000007	CHS$M_MBADTC	00080000	DS$ASKADR	00010090
CHECK_STACK	00007B70-R	CHS$M_MBAEXC	00010000	DS$ASKDATA	00010080
CHI$M_CHNINT	00000001	CHS$M_MBANED	00020000	DS$ASKLGCL	00010098
CHI$M_DEVINT	00000002	CHS$M_MBAWCKLWR	02000000	DS$ASKSTR	000100A0
CHI$M_IPL	0000007C	CHS$M_MBAWCKUPR	04000000	DS$ASKVLD	00010088
CHI$S_IPL	00C00005	CHS$M_PGMERR	00000008	DS$ATTACH	000101A8

Symbol	Value	Symbol	Value	Symbol	Value
DS$BGNSUB	00010030	DS$K_BLEQ_M	0000000E	DS$_IHWE	00660060
DS$BRANCH	000100A8	DS$K_BLSSU_B	00000001	DS$_ILLCHAR	00660018
DS$BREAK	00010058	DS$K_BLSSU_M	0000000D	DS$_ILLPAGCNT	00660078
DS$CANWAIT	00010070	DS$K_BLSS_B	00000000	DS$_ILLUNIT	00660100
DS$CHANNEL	00010180	DS$K_BLSS_M	0000000C	DS$_INSFMEM	00660050
DS$CKLOOP	00010040	DS$K_BNEQU_B	0000000B	DS$_IPL2HI	006600B8
DS$CLRVEC	00010168	DS$K_BNEQU_M	00000017	DS$_IVADDR	00660040
DS$CNTRLC	00010078	DS$K_BNEQ_B	0000000A	DS$_IVVECT	00660038
DS$CVTREG	000100B0	DS$K_BNEQ_M	00000016	DS$_KRNLSTK	00660090
DS$ENDPASS	00010010	DS$K_BNERROR	00000027	DS$_LOGIC	00660070
DS$ENDSUB	00010038	DS$K_BVC_M	00000019	DS$_MCHK	00660088
DS$ERRDEV	000100C8	DS$K_BVS_M	00000018	DS$_MMOFF	00660058
DS$ERRHARD	000100D0	DS$K_ERROR	00000002	DS$_NEEDUNIT	006600F8
DS$ERRPREP	00010108	DS$K_NORMAL	00000001	DS$_NOPCS	00660110
DS$ERRSOFT	000100D8	DS$K_SEVERE	00000004	DS$_NORMAL	00660001
DS$ERRSYS	000100C0	DS$K_SUBSYS	00000066	DS$_NOSUPPORT	006600B1
DS$ESCAPE	00010050	DS$K_WARNING	00000000	DS$_NOTDON	00660030
DS$FREEDBGSYM	000101B8	DS$LOAD	00010198	DS$_NOTIMP	006600B0
DS$GETBUF	00010120	DS$LOADPCS	000101C0	DS$_NULLSTR	00660010
DS$GETMEM	00010130	DS$MAPDBGBLOCK	00010118	DS$_OVERFLOW	00660008
DS$GPHARD	00010018	DS$MMOFF	00010158	DS$_POWER	00660098
DS$HELP	000101B0	DS$MMON	00010150	DS$_PROGERR	00660020
DS$INITSCB	00010170	DS$MOVPHY	00010148	DS$_SEVERE	00660004
DS$INLOOP	00010048	DS$MOVVRT	00010140	DS$_TRANSL	006600A0
DS$K_BBC	0000001D	DS$PARSE	000100B8	DS$_TRUNCATE	00660028
DS$K_BBCC	00000021	DS$PRINTB	000100E0	DS$_UNEXPINT	006600D8
DS$K_BBCCI	00000023	DS$PRINTF	000100F0	DS$_VASFULL	00660048
DS$K_BBCS	0000001F	DS$PRINTS	000100F8	DS$_WARNING	00660000
DS$K_BBS	0000001C	DS$PRINTSIG	00010100	DSA$AL_APTMAIL	0000FE00
DS$K_BBSC	00000020	DS$PRINTX	000100E8	DSA$AT_APTTXT	0000FA00
DS$K_BBSS	0000001E	DS$PROBE	000101A0	DSA$GL_APTCOM	0000FE04
DS$K_BBSSI	00000022	DS$RELBUF	00010128	DSA$GL_DEVLEN	0000FE58
DS$K_BCC_M	0000001B	DS$RELMEM	00010138	DSA$GL_ERRNO	0000FE44
DS$K_BCS_M	0000001A	DS$SETIPL	00010178	DSA$GL_EVENT	0000FE48
DS$K_BEQLU_B	00000005	DS$SETMAP	00010188	DSA$GL_FLAGS	0000FE00
DS$K_BEQLU_M	00000011	DS$SETPRIEXV	00010110	DSA$GL_MSGTYP	0000FE40
DS$K_BEQL_B	00000004	DS$SETVEC	00010160	DSA$GL_PASSES	0000FE08
DS$K_BEQL_M	00000010	DS$SHOCHAN	00010190	DSA$GL_PASSNO	0000FE54
DS$K_BERROR	00000026	DS$SUMMARY	00010028	DSA$GL_SECTNO	0000FE10
DS$K_BGEQU_B	00000007	DS$WAITMS	00010060	DSA$GL_SID	0000FE14
DS$K_BGEQU_M	00000013	DS$WAITUS	00010068	DSA$GL_SUBTNO	0000FE4C
DS$K_BGEQ_B	00000006	DS$_ARITH	006600D0	DSA$GL_TESTNO	0000FE50
DS$K_BGEQ_M	00000012	DS$_BADLINK	006600F0	DSA$GL_UNITS	0000FE0C
DS$K_BGTRU_B	00000009	DS$_BADTYPE	006600E8	DSA$GQ_MSGPTR	0000FE68
DS$K_BGTRU_M	00000015	DS$_CHME	006600A8	DSA$GT_DEVNAM	0000FE5C
DS$K_BGTR_B	00000008	DS$_CHMK	006600E0	DSA$M_APT	80000000
DS$K_BGTR_M	00000014	DS$_DEVNAME	00660108	DSA$M_BELL	00000008
DS$K_BLBC	00000025	DS$_ERROR	00660002	DSA$M_BINARY	00000002
DS$K_BLBS	00000024	DS$_FHWE	00660068	DSA$M_CRD_AUTOTEST	00000800
DS$K_BLEQU_B	00000003	DS$_FRAGBUF	00660080	DSA$M_CRD_MENUTEST	00010000
DS$K_BLEQU_M	0000000F	DS$_ICBUSY	006600C8	DSA$M_CRD_TRACE	00020000
DS$K_BLEQ_B	00000002	DS$_ICERR	006600C0	DSA$M_DEBUG	04000000

Symbol	Value	Symbol	Value	Symbol	Value
DSA$M_HALT	00000001	IPR	00000006	PR$_ICR	0000001A
DSA$M_IE1	00000010	KA730_REV_54	03003600	PR$_IPL	00000012
DSA$M_IE2	00000020	KA_PTABLE	0000198B-R	PR$_ISP	00000004
DSA$M_IE3	00000040	KA_UNIT_NO	00001871-R	PR$_KSP	00000000
DSA$M_IES	00000080	LDPCTX	0000000B	PR$_MAPEN	00000038
DSA$M_LOAD_DEBUGGER	40000000	L_ACT_ISP	00001858-R	PR$_MCESR	00000026
DSA$M_LOOP	00000004	L_ACT_KSP	00001854-R	PR$_MCTL1	00000044
DSA$M_NORPT	08000000	L_ACT_PSL	0000185C-R	PR$_MCTL2	00000045
DSA$M_OPER	00001000	L_ACT_SP	00001850-R	PR$_MDCR1	00000049
DSA$M_PASSO	20000000	L_EXP_AP	000018C5-R	PR$_MEAR	00000048
DSA$M_PROMPT	00002000	L_EXP_FP	000018C9-R	PR$_MECCR	0000004B
DSA$M_QA	00008000	L_EXP_ISP	000018D5-R	PR$_MEDR	0000004A
DSA$M_QUICK	00000100	L_EXP_KSP	000018D1-R	PR$_MGEN	00000046
DSA$M_SEARCH	00004000	L_EXP_PSL	000018D9-R	PR$_MTBER	00000047
DSA$M_TRACE	00000400	L_EXP_SP	000018CD-R	PR$_NICR	00000019
DSA$M_USER	10000000	L_ISP	00001864-R	PR$_P0BR	00000008
DSA$M_VERIFY	00000200	L_KA_FLAGS	0000188D-R	PR$_P0LR	00000009
DSA$V_APT	0000001F	L_KSIZE_PTBL	00001891-R	PR$_P1BR	0000000A
DSA$V_BELL	00000003	L_KSP	00001860-R	PR$_P1LR	0000000B
DSA$V_BINARY	00000001	L_LAST_UBE	00001868-R	PR$_PAMACC	00000040
DSA$V_CRD_AUTOTEST	0000000B	MANUAL	C 00001	PR$_PAMLOC	00000041
DSA$V_CRD_MENUTEST	00000010	MCHK	00000005	PR$_PCBB	00000010
DSA$V_CRD_TRACE	00000011	MEM	00000009	PR$_PME	0000003D
DSA$V_DEBUG	0000001A	MODE	00007B71-R	PR$_RXCS	00000020
DSA$V_HALT	00000000	MODELIST	00001956-R	PR$_RXDB	00000021
DSA$V_IE1	00000004	OPR_ABORT	00007E24-R	PR$_SBIER	00000034
DSA$V_IE2	00000005	ORIG_PCBB	00007B7C-R	PR$_SBIFS	00000030
DSA$V_IE3	00000006	PART_INIT	0000158B-R	PR$_SBIMT	00000033
DSA$V_IES	00000007	PCB_ERR_FLAG	00007B74-R	PR$_SBIQC	00000036
DSA$V_LOAD_DEBUGGER	0000001E	POWER	00000003	PR$_SBIS	00000031
DSA$V_LOOP	00000002	PR$S_SID_ECO	00000009	PR$_SBISC	00000032
DSA$V_NORPT	0000001B	PR$S_SID_PL	00000003	PR$_SBITA	00000035
DSA$V_OPER	0000000C	PR$S_SID_SN	0000000C	PR$_SBR	0000000C
DSA$V_PASSO	0000001D	PR$S_SID_TYPE	00000008	PR$_SCBB	00000011
DSA$V_PROMPT	0000000D	PR$V_SID_ECO	0000000F	PR$_SID	0000003E
DSA$V_QA	0000000F	PR$V_SID_PL	0000000C	PR$_SID_TYP730	00000003
DSA$V_QUICK	00000008	PR$V_SID_SN	00000000	PR$_SID_TYP750	00000002
DSA$V_SEARCH	0000000E	PR$V_SID_TYPE	00000018	PR$_SID_TYP780	00000001
DSA$V_TRACE	0000000A	PR$_ACCR	00000029	PR$_SID_TYP7VV	00000004
DSA$V_USER	0000001C	PR$_ACCS	00000028	PR$_SID_TYPMAX	00000004
DSA$V_VERIFY	00000009	PR$_ASTLVL	00000013	PR$_SIRR	00000014
ERROR_CHECK	000030D5-R	PR$_CADR	00000025	PR$_SISR	00000015
EXECUTE	0000F675-R	PR$_CAER	00000027	PR$_SLR	0000000D
EXPECTED_EXCEPT	00007B6C-R	PR$_CMIERR	00000017	PR$_SSP	00000002
EXPECT_PCB	00007C40-R	PR$_CRBT	00000043	PR$_TBCHK	0000003F
EXPECT_REG	00007D60-R	PR$_CSRD	0000001D	PR$_TBDR	00000024
GPR_TABLE	00001895-R	PR$_CSRS	0000001C	PR$_TBIA	00000039
HALT	00000004	PR$_CSTD	0000001F	PR$_TBIS	0000003A
HANDLER	00000D0A-R	PR$_CSTS	0000001E	PR$_TODR	0000001B
HARDCNT	00001400-R	PR$_CSWP	00000042	PR$_TXCS	00000022
INIT_PCB	00007BE0-R	PR$_ESP	00000001	PR$_TXDB	00000023
INIT_REG	00007D00-R	PR$_ICCS	00000018	PR$_UBRESET	00000037

Symbol	Value	Symbol	Value	Symbol	Value
PR$_USP	00000003	PSL$V_Z	00000002	SYS$CLREF	00010298
PR$_WCSA	0000002C	PSL$V_ZBIT	00000002	SYS$CONNECT	000105C0
PR$_WCSD	0000002D	REG_DATA	00007B80-R	SYS$DALLOC	000102D8
PRINT_ERR	00000C10-R	REG_ERR_FLAG	00007B78-R	SYS$DASSGN	000102E0
PSL$C_EXEC	00000001	REG_TABLE	000018DD-R	SYS$DISCONNECT	000105D0
PSL$C_KERNEL	00000000	SAVE_RESULTS	0000F76E-R	SYS$FAO	00010350
PSL$C_SUPER	00000002	SCB$L_ACCESS	00000020	SYS$FAOL	00010358
PSL$C_USER	00000003	SCB$L_ARITH	00000034	SYS$GET	00010580
PSL$K_EXEC	00000001	SCB$L_BREAK	0000002C	SYS$GETCHN	000104C8
PSL$K_KERNEL	00000000	SCB$L_CHME	00000044	SYS$GETTIM	00010378
PSL$K_SUPER	00000002	SCB$L_CHMK	00000040	SYS$LKWSET	000103A0
PSL$K_USER	00000003	SCB$L_CHMS	00000048	SYS$NUMTIM	00010388
PSL$M_C	00000001	SCB$L_CHMU	0000004C	SYS$OPEN	00010608
PSL$M_CBIT	00000001	SCB$L_COMPAT	00000030	SYS$QIO	000103C8
PSL$M_CM	80000000	SCB$L_KNLSTK	00000008	SYS$QIOW	00010200
PSL$M_CURMOD	03000000	SCB$L_MACHCHK	00000004	SYS$READ	00010590
PSL$M_DV	00000080	SCB$L_OPCCUS	00000014	SYS$READEF	000103D0
PSL$M_FPD	08000000	SCB$L_OPCDEC	00000010	SYS$SETAST	000103F8
PSL$M_FU	00000040	SCB$L_POWER	0000000C	SYS$SETEF	00010400
PSL$M_IPL	001F0000	SCB$L_RADRMOD	0000001C	SYS$SETIMR	00010420
PSL$M_IS	04000000	SCB$L_ROPRAND	00000018	SYS$SETPRI	00010428
PSL$M_IV	00000020	SCB$L_RXDB	000000F8	SYS$SETPRT	00010430
PSL$M_N	00000008	SCB$L_SFTLVL1	00000084	SYS$SETRWM	00010438
PSL$M_NBIT	00000008	SCB$L_SFTLVL10	000000A8	SYS$SETSWM	00010448
PSL$M_PRVMOD	00C00000	SCB$L_SFTLVL11	000000AC	SYS$ULKPAG	00010460
PSL$M_SAFBITS	000037FF	SCB$L_SFTLVL12	000000B0	SYS$ULWSET	00010468
PSL$M_TBIT	00000010	SCB$L_SFTLVL13	000000B4	SYS$UNWIND	00010470
PSL$M_TP	40000000	SCB$L_SFTLVL14	000000B8	SYS$WAITFR	00010478
PSL$M_V	00000002	SCB$L_SFTLVL15	000000BC	SYS$WFLAND	00010488
PSL$M_VBIT	00000002	SCB$L_SFTLVL2	00000088	SYS$WFLOR	00010490
PSL$M_Z	00000004	SCB$L_SFTLVL3	0000008C	T10_S1	0000F22A-R
PSL$M_ZBIT	00000004	SCB$L_SFTLVL4	00000090	T10_S2	0000F30F-R
PSL$S_CURMOD	00000002	SCB$L_SFTLVL5	00000094	T10_S3	0000F3BD-R
PSL$S_IPL	00000005	SCB$L_SFTLVL6	00000098	T1_S1	0000823E-R
PSL$S_PRVMOD	00000002	SCB$L_SFTLVL7	0000009C	T1_S2	00008318-R
PSL$V_C	00000000	SCB$L_SFTLVL8	000000A0	T1_S3	000083F9-R
PSL$V_CBIT	00000000	SCB$L_SFTLVL9	000000A4	T2_S1	0000867F-R
PSL$V_CM	0000001F	SCB$L_TBIT	00000028	T2_S2	00008701-R
PSL$V_CURMOD	00000018	SCB$L_TIMER	000000C0	T2_S3	00008796-R
PSL$V_DV	00000007	SCB$L_TRANSL	00000024	T3_S1	00008AAD-R
PSL$V_FPD	0000001B	SCB$L_TXDB	000000FC	T3_S2	00008B99-R
PSL$V_FU	00000006	SCB$L_ZERO	00000000	T3_S3	00008E01-R
PSL$V_IPL	00000010	SUM_K_SIZE	0000000C	T3_S4	00008F81-R
PSL$V_IS	0000001A	SUM_L_HARDCNT	00000000	T3_S5	000090F4-R
PSL$V_IV	00000005	SUM_Q_RUNTIME	00000004	T3_S6	00009268-R
PSL$V_N	00000003	SYS$ALLOC	00010238	T3_S7	000093DC-R
PSL$V_NBIT	00000003	SYS$ASCTIM	00010248	T4_S1	000096B1-R
PSL$V_PRVMOD	00000016	SYS$ASSIGN	00010250	T4_S2	000097E3-R
PSL$V_TBIT	00000004	SYS$BINTIM	00010258	T4_S3	00009915-R
PSL$V_TP	0000001E	SYS$CANCEL	00010260	T4_S4	00009A47-R
PSL$V_V	00000001	SYS$CANTIM	00010268	T4_S5	00009B9D-R
PSL$V_VBIT	00000001	SYS$CLOSE	000105B8	T4_S6	00009C9E-R

Symbol	Value	Symbol	Value	Symbol	Value
T4_S7	00009DBD-R	TEST_003	00008A24-R		
T4_S8	00009EC4-R	TEST_003_X	00009560-R		
T5_S1	0000A056-R	TEST_004	00009628-R		
T5_S2	0000A135-R	TEST_004_X	00009FE0-R		
T5_S3	0000A214-R	TEST_005	0000A020-R		
T5_S4	0000A2F3-R	TEST_005_X	0000A3E2-R		
T6_S1	0000A5BE-R	TEST_006	0000A418-R		
T6_S2	0000A6CC-R	TEST_006_X	0000A909-R		
T6_S3	0000A7DA-R	TEST_007	0000AA20-R		
T7_S1	0000ABD2-R	TEST_007_X	0000CE70-R		
T7_S10	0000C0F5-R	TEST_008	0000D020-R		
T7_S11	0000C4B0-R	TEST_008_X	0000E998-R		
T7_S12	0000C845-R	TEST_009	0000EA20-R		
T7_S13	0000CAA8-R	TEST_009_X	0000F06F-R		
T7_S14	0000CC79-R	TEST_010	0000F210-R		
T7_S2	0000AF68-R	TEST_010_X	0000F5B2-R		
T7_S3	0000B16E-R	TEST_011	0000F61C-R		
T7_S4	0000B2DE-R	TEST_011_X	0000F908-R		
T7_S5	0000B4EB-R	TEST_CONTROL	00000EF2-R		
T7_S6	0000B669-R	TU0_UNIT_NO	00001875-R		
T7_S7	0000B86C-R	TU1_UNIT_NO	00001879-R		
T7_S8	0000B9DC-R	TU58	00000002		
T7_S9	0000BD81-R	UBI	0000000A		
T8_S1	0000D03C-R	WCS	00000007		
T8_S10	0000DA61-R				
T8_S11	0000DB81-R				
T8_S12	0000DCA5-R				
T8_S13	0000DE46-R				
T8_S14	0000DF74-R				
T8_S15	0000E090-R				
T8_S16	0000E1B2-R				
T8_S17	0000E336-R				
T8_S18	0000E48A-R				
T8_S19	0000E5DB-R				
T8_S2	0000D13A-R				
T8_S20	0000E6E1-R				
T8_S21	0000E7E8-R				
T8_S22	0000E8E0-R				
T8_S3	0000D1B1-R				
T8_S4	0000D24F-R				
T8_S5	0000D33D-R				
T8_S6	0000D4A9-R				
T8_S7	0000D65D-R				
T8_S8	0000D7E4-R				
T8_S9	0000D92A-R				
T9_S1	0000EAC0-R				
T9_S2	0000EDAE-R				
TEMP_PSL	000005F8-R				
TEST_001	00008224-R				
TEST_001_X	000084CF-R				
TEST_002	00008614-R				
TEST_002_X	00008876-R				

Symbol	Value	Symbol	Value	Symbol	Value

Key for special characters above:

*	- Undefined
U	- Universal
R	- Relocatable
X	- External

Image Synopsis

```
Virtual memory allocated:                   00000200 0000F9FF 0000F800 (63488. bytes, 124. pages)
Stack size:                                       0. pages
Image binary virtual block limits:                1.       124. (  124. blocks)
Image name and identification:              ENKAX 1-3
Number of files:                                 13.
Number of modules:                               12.
Number of program sections:                      37.
Number of global symbols:                       663.
Number of image sections:                         1.
Image type:                                 SYSTEM.
Map format:                                 DEFAULT in file WRKD$0:[PAN.ENKAX]ENKAX.MAP;1
Estimated map length:                       88. blocks
```

Link Run Statistics

Performance Indicators	Page Faults	CPU Time	Elapsed Time
Command processing:	53	00:00:00.25	00:00:03.42
Pass 1:	236	00:00:01.77	00:00:05.28
Allocation/Relocation:	29	00:00:00.08	00:00:00.32
Pass 2:	218	00:00:03.78	00:00:07.63
Map data after object module synopsis:	94	00:00:01.67	00:00:02.22
Symbol table output:	2	00:00:00.00	00:00:00.15
Total run values:	632	00:00:07.55	00:00:19.02

```
Using a working set limited to 600 pages and 105 pages of data storage (excluding image)

Total number object records read (both passes):   896
    of which 14 were in libraries and 28 were DEBUG data records containing 1647 bytes

Number of modules extracted explicitly             = 0
    with 1 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

/CONTIG/SYST=%X200/EXE=ENKAX/MAP=ENKAX ENKAX1+ENKAXA+ENKAXC+ENKAXD+ENKAXF+ENKAXG+ENKAXH+ENKAXI+ENKAXK+ENKAXL+SYS$LIBRARY:DS.STB
```

Table of contents

```
                        1          .SBTTL  ENKAX1: HEADER MODULE
                        2          .TITLE  ENKAX1  VAX 11/730 Specific CPU Cluster Exerciser
                        3          .IDENT  /1-3/
                        4
                        5 ;
                        6 ; Copyright (C) 1981, 1982
                        7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
                        8 ;
                        9 ; This software is furnished  under a license for use only on a single
                       10 ; computer  system and  may be  copied only with  the inclusion of the
                       11 ; above copyright notice.  This software, or any other copies thereof,
                       12 ; may not be provided or  otherwise made available to any other person
                       13 ; except for use on such system and to one who agrees to these license
                       14 ; terms.  Title to and  ownership of the  software  shall at all times
                       15 ; remain in DEC.
                       16 ;
                       17 ; The information in this software is subject to change without notice
                       18 ; and should  not be construed  as a commitment  by Digital  Equipment
                       19 ; Corporation.
                       20 ;
                       21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
                       22 ; software on equipment which is not supplied by DEC.
                       23 ;
                       24
                       25 ;++
                       26 ; Facility:      VAX Architecture Diagnostic.
                       27 ;
                       28 ; Abstract:      This module provides all the standard header informaton
                       29 ;                and routines required to interface to the VAX Diagnostic
                       30 ;                Supervisor.
                       31 ;
                       32 ; Environment:   VAX Diagnostic Supervisor.
                       33 ;
                       34 ; Author:        Rich Lewis      7-Jly-81        Version 1.0
                       35 ;
                       36 ; Modified by:   Kevin Martin    9-NOV-82        Version 1.1
                       37 ;       01       Added modules ENKAXK and ENKAXL - LDPCTX reserved operand
                       38 ;                test cases.
                       39 ;                Bug fixes in module ENKAX1 ENKAXI and ENKAXH
                       40 ;
                       41 ;                Tai-Chun Pan    01-Apr-83       Version 1.2
                       42 ;                Peter Green
                       43 ;
                       44 ;                Added quick flag on Test 7(TU58). If quick flag is set
                       45 ;                will skip subtest 4 through subtest 11.  Added event flag 3.
                       46 ;                If event flag 3 is set, will override protection,
                       47 ;                i.e., go ahead and write on tape.  If run APT & event flag 3
                       48 ;                is clear & tape is not scratch, will report an error.
                       49 ;                Modified Unibus sizer on Test 8 & 9. make test run faster.
                       50 ;
                       51 ;
                       52 ;                Tai-Chun Pan    03-Aug-83       Version 1.3
                       53 ;
                       54 ;                fixed bugs on TEST 7 and error summary routine call when
                       55 ;                running under APT.
                       56 ;
                       57 ;
```

```
                        0000     58 ;
                        0000     59 ;--
```

```
                        0000     61          .SBTTL  DECLARATIONS
                        0000     62          .LIST   MEB
                        0000     63          .NLIST  CND
                    00000000     64          .PSECT  ENKAX1, PAGE, NOWRT
                        0000     65          .LIBRARY        "SYS$LIBRARY:DIAG"              ; VAX family diagnostic libr
                        0000     66          $DS_BGNMOD      CEP_REPAIR
                        0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                        0000     67
                        0000     68 ;
                        0000     69 ; Include files:
                        0000     70 ;
                        0000     71          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                        0000     72
                        0000     76
                        0000     77          $DS_SCBDEF      GLOBAL          ; Defines system control block
                        0000     78          $DS_DSDEF       GLOBAL          ; Supervisor status, cond. codes
                        0000     79          $DS_DSSDEF      GLOBAL          ; Supervisor service entry points
                        0000     80          $DS_DSADEF                      ; Apt command, mailbox offsets
                        0000     81          $DS_PARDEF                      ; Define Psl bits
                        0000     82          $DS_PSLDEF      GLOBAL          ; Starlet definition of the Psl
                        0000     83          $PRDEF          GLOBAL          ; Defines processor registers
                        0000     84          $DS_CHDEF       GLOBAL          ;
                        0000     85          $CHFDEF                         ;
                        0000     86          $DS_HPODEF                      ; Hardware P-table offsets
                        0000     87          KA_DEF                          ; Set up for processor P-table
                        0000     88          $DS_QADEF       GLOBAL          ;
                    03003600    0000     89          KA730_REV_54 == ^X 03003600     ; KA730 micro rev ID value
                        0000     90
                        0000     91 ;
                        0000     92 ; Own storage and equated symbols:
                        0000     93 ;
                        0000     94
                    00000348    0000     95 SCB$L_UBE_0          =^X348          ; UBE 0 interrupt vector offset
                    0000034C    0000     96 SCB$L_UBE_1          =^X34C          ; UBE 1 interrupt vector offset
                    00000350    0000     97 SCB$L_UBE_2          =^X350          ; UBE 2 interrupt vector offset
                    00000354    0000     98 SCB$L_UBE_3          =^X354          ; UBE 3 interrupt vector offset
                        0000     99
                    02000000    0000    100 M_DSABL_ECC          =1@25           ; memory CSR disable ECC bit
                    00F20004    0000    101 A_MEM_CSR_1          =^XF20004       ; address of memory CSR
                        0000    102
                    0000000C    0000    103 HP$KA_L_FLAGS        =^XC            ; offset of flags in p table
                    00000044    0000    104 HP$KA_MEM_SIZE       =^X44           ; offset of memory size in p table
                        0000    105
                    00000004    0000    106 A_BUFFER_BEGIN::     .BLKL   1       ; address of data buffer
                    00000008    0004    107 A_BUFFER_END::       .BLKL   1
                    00000010    0008    108 A_PWR_BUFFER::       .BLKL   2       ; address of POWER FAIL data buffer
                    00000018    0010    109 A_HALT_BUFFER::      .BLKL   2       ; address of HALTS data buffer
                    00000020    0018    110 A_MCHK_BUFFER::      .BLKL   2       ; address of MACHINE CHECK data buffer
                    00000028    0020    111 A_WCS_BUFFER::       .BLKL   2       ; address of WCS data buffer
                    00000030    0028    112 A_UBE_BUFFER_0::     .BLKL   2       ; address of UBE buffer 0
                    00000038    0030    113 A_UBE_BUFFER_1::     .BLKL   2       ; address of UBE buffer 1
                    00000040    0038    114 A_UBE_BUFFER_2::     .BLKL   2       ; address of UBE buffer 2
                    00000048    0040    115 A_UBE_BUFFER_3::     .BLKL   2       ; address of UBE buffer 3
                    00000050    0048    116 A_FREE_MEM::         .BLKL   2       ; address of buffer above supervisor
                                0050    117
                    00000054    0050    118 L_ACT_SP::           .BLKL           ; used to store actual SP
                    00000058    0054    119 L_ACT_KSP::          .BLKL           ; used to store actual KSP
```

```
0000005C  0058   120 L_ACT_ISP::              .BLKL           ; used to store actual ISP
00000060  005C   121 L_ACT_PSL::              .BLKL           ; used to store actual PSL
00000064  0060   122 L_KSP::                  .BLKL           ; used to store initial KSP
00000068  0064   123 L_ISP::                  .BLKL           ; used to store initial ISP
          0068   124
FFFFFFFF  0068   125 L_LAST_UBE::             .LONG   -1      ; highest UBE number (0,1,2,or 3)
       00 006C   126 B_TU58_DRIVE::           .BYTE   0       ; represents TU58 drives attached
00000000  006D   127 A_P_TABLE:               .LONG           ; address of hardware p table
00000000  0071   128 KA_UNIT_NO::            .LONG           ; CPU unit number
00000000  0075   129 TU0_UNIT_NO::           .LONG           ; TU58 drive 0 unit number
00000000  0079   130 TU1_UNIT_NO::           .LONG           ; TU58 drive 1 unit number
00000000  007D   131 BE0_UNIT_NO::           .LONG           ; UBE 0 unit number
00000000  0081   132 BE1_UNIT_NO::           .LONG           ; UBE 1 unit number
00000000  0085   133 BE2_UNIT_NO::           .LONG           ; UBE 2 unit number
00000000  0089   134 BE3_UNIT_NO::           .LONG           ; UBE 3 unit number
00000000  008D   135 L_KA_FLAGS::            .LONG           ; KA flags
00000000  0091   136 L_KSIZE_PTBL::          .LONG           ; memory size from p table (kbytes)
          0095   137
          0095   138 GPR_TABLE::
00000000  0095   139         .LONG   ^X00000000              ; data for register 0
11111111  0099   140         .LONG   ^X11111111              ; data for register 1
22222222  009D   141         .LONG   ^X22222222              ; data for register 2
33333333  00A1   142         .LONG   ^X33333333              ; data for register 3
44444444  00A5   143         .LONG   ^X44444444              ; data for register 4
55555555  00A9   144         .LONG   ^X55555555              ; data for register 5
66666666  00AD   145         .LONG   ^X66666666              ; data for register 6
77777777  00B1   146         .LONG   ^X77777777              ; data for register 7
88888888  00B5   147         .LONG   ^X88888888              ; data for register 8
99999999  00B9   148         .LONG   ^X99999999              ; data for register 9
AAAAAAAA  00BD   149         .LONG   ^XAAAAAAAA              ; data for register A
BBBBBBBB  00C1   150         .LONG   ^XBBBBBBBB              ; data for register B
000000C9  00C5   151 L_EXP_AP::      .BLKL           ; location to store argument counter
000000CD  00C9   152 L_EXP_FP::      .BLKL           ; location to store frame pointer
000000D1  00CD   153 L_EXP_SP::      .BLKL           ; location to store expected SP
000000D5  00D1   154 L_EXP_KSP::     .BLKL           ; location to store expected KSP
000000D9  00D5   155 L_EXP_ISP::     .BLKL           ; location to store expected ISP
000000DD  00D9   156 L_EXP_PSL::     .BLKL           ; location to store expected PSL
          00DD   157
          00DD   158
          00DD   159
          00DD   160 REG_TABLE::
20 30 52 00'      00DD   161         .ASCIC  "R0 "
            03    00DD
20 31 52 00'      00E1   162         .ASCIC  "R1 "
            03    00E1
20 32 52 00'      00E5   163         .ASCIC  "R2 "
            03    00E5
20 33 52 00'      00E9   164         .ASCIC  "R3 "
            03    00E9
20 34 52 00'      00ED   165         .ASCIC  "R4 "
            03    00ED
20 35 52 00'      00F1   166         .ASCIC  "R5 "
            03    00F1
20 36 52 00'      00F5   167         .ASCIC  "R6 "
            03    00F5
20 37 52 00'      00F9   168         .ASCIC  "R7 "
            03    00F9
```

```
                        20 38 52 00' 00FD   169          .ASCIC  "R8 "
                                  03 00FD
                        20 39 52 00' 0101   170          .ASCIC  "R9 "
                                  03 0101
                        30 31 52 00' 0105   171          .ASCIC  "R10"
                                  03 0105
                        31 31 52 00' 0109   172          .ASCIC  "R11"
                                  03 0109
                        20 50 41 00' 010D   173          .ASCIC  "AP "
                                  03 010D
                        20 50 46 00' 0111   174          .ASCIC  "FP "
                                  03 0111
                                     0115   175
                                     0115   176
                                     0115   177 ; The following is used tc intrepret the PSL and to print out an English
                                     0115   178 ; language description of its contents.
                                     0115   179
                                     0115   180 A_PSLMNE::
44 50 46 2C 2C 2C 50 54 2C 4D 43 00' 0115   181          .ASCIC  "CM,TP,,,FPD,IS,CUR=2@,PRV=2@,,"-
50 2C 40 32 3D 52 55 43 2C 53 49 2C  0121
35 3D 4C 50 49 2C 2C 40 32 3D 56 52  012D
44 2C 2C 2C 2C 2C 2C 2C 2C 2C 58 5E  0139
2C 4E 2C 54 2C 56 49 2C 55 46 2C 56  0145
                  43 2C 56 2C 5A     0151
                              40     0115
                                     0156   182                  "IPL=5^X,,,,,,,,,DV,FU,IV,T,N,Z,V,C"
                                     0156   183
                           00000004  0156   184 MODELIST::       .LONG   4
                           0000016A' 015A   185          .ADDRESS        AKM
                           00000171' 015E   186          .ADDRESS        AEM
                           0000017B' 0162   187          .ADDRESS        ASM
                           00000186' 0166   188          .ADDRESS        AUM
                                     016A   189
                     4C 45 4E 52 45 4B 00' 016A   190 AKM:     .ASCIC  "KERNEL"
                                  06 016A
                  45 56 49 54 55 43 45 58 45 00' 0171   191 AEM:     .ASCIC  "EXECUTIVE"
                                  09 0171
               52 4F 53 49 56 52 45 50 55 53 00' 017B   192 ASM:     .ASCIC  "SUPERVISOR"
                                  0A 017B
                           52 45 53 55 00' 0186   193 AUM:     .ASCIC  "USER"
                                  04 0186
                                     018B   194
                                     018B   195
```

```
                                0188   200          .SBTTL  Program Header Data Block
                                0188   201 ;++
                                0188   202 ; Functional Description:
                                0188   203 ;
                                0188   204 ;       The program header data block contains the parameters which
                                0188   205 ;       allow the Diagnostic Supervisor to control the program.
                                0188   206 ;       The Diagnostic Supervisor looks for the header information
                                0188   207 ;       beginning at virtual address 200(hex).
                                0188   208 ;
                                0188   209 ;--
                                0188   210          $DS_HEADER  <VAX 11/730 Specific CPU Cluster Exerciser - ZZ-ENKAX-1.3 >, -
                                0188   211                      REV = 1, UPDATE = 3, -
                                0188   212                      NUNIT = 7
                                0188                        .SAVE
                             00000000                       .PSECT  $HEADER, PAGE, NOEXE, NOWRT
                    00000058' 0000      L$L_HEADLENGTH: .LONG    A_HEADEND-.     ; LENGTH OF THE HEADER DATA BLOCK
                    00000005  0004      L$L_ENVIRON:    .LONG    <ENV$_SUPER@ENV$V_SUPER>+$ENV  ; PROGRAM ENVIRONMENT
                    00000058' 0008      L$A_NAME:       .ADDRESS T_NAME          ; PROGRAM NAME TEXT ADDRESS
                    00000001  000C      L$L_REV:        .LONG    1               ; PROGRAM REVISION LEVEL
                    00000003  0010      L$L_UPDATE:     .LONG    3               ; DIAGNOSTIC ENGINEERING PATCH ORDER
                    00000000' 0014      L$A_LASTAD:     .ADDRESS LASTAD          ; FIRST FREE LOCATION AFTER PROGRAM
                    00000000' 0018      L$A_DTP:        .ADDRESS DISPATCH        ; TEST DISPATCH TABLE POINTER
                    0000009B' 001C      L$A_DEVP:       .ADDRESS AL_DEVTYP       ; DEVICE TYPE LIST POINTER
                    00000007  0020      L$L_UNIT:       .LONG    7               ; NUMBER OF UNITS THAT CAN BE TESTED
                    00000000' 0024      L$A_DREG:       .ADDRESS DEV_REG         ; DEVICE REGISTER CONTENTS TABLE POINTER
                    0000001C  0028                      .LONG    28              ; Ptable length for 4.05 compatability
00000000'00000000'00000000'00000000'  002C              .LONG    0[4]            ; UNUSED OFFSETS
                    00000000' 003C      L$A_ICP:        .ADDRESS INITIALIZE      ; INITIALIZATION CODE POINTER
                    00000000' 0040      L$A_CCP:        .ADDRESS CLEAN_UP        ; CLEAN-UP CODE POINTER
                    00000000' 0044      L$A_REPP:       .ADDRESS SUMMARY         ; SUMMARY REPORT CODE POINTER
                    00000000' 0048      L$A_STATAB:     .ADDRESS 0               ; STATISTIC TABLE POINTER
                    00000000  004C      L$L_ERRTYP:     .LONG    0               ; # OF TYPES OF $ERRSOFT & $ERRPREP
                    00000057' 0050      L$A_SECNAM:     .ADDRESS SECTION         ; LIST OF SECTION NAME ADDRESSES
                    00000000' 0054      L$A_TSTCNT:     .ADDRESS L_TSTCNT        ; POINTER TO NUMBER OF TESTS
                              0058      A_HEADEND:
20 30 33 37 2F 31 31 20 58 41 56 00'  0058  T_NAME:     .ASCIC  \VAX 11/730 Specific CPU Cluster Exerciser - ZZ-ENKAX-1.3 \
55 50 43 20 63 69 66 69 63 65 70 53   0064
65 78 45 20 72 65 74 73 75 6C 43 20   0070
2D 5A 5A 20 2D 20 72 65 73 69 63 72   007C
      20 33 2E 31 2D 58 41 4B 4E 45   0088
                                39    0058
                             00000000                   .PSECT  _LAST, PAGE
                                 0000   LASTAD:
                             00000000                   .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                                 0000   L_TSTCNT:
                             0000018B                   .RESTORE
```

```
                                 018B   214          .SBTTL  Dispatch Table
                                 018B   215 ;++
                                 018B   216 ; Functional Description:
                                 018B   217 ;
                                 018B   218 ;       The Dispatch Table is a collection of addresses generated at the
                                 018B   219 ;       beginning of each test and grouped together into a contiguous
                                 018B   220 ;       list by the linker.  This is done by defining a PSECT called
                                 018B   221 ;       DSIPATCH.
                                 018B   222 ;
                                 018B   223 ;--
                                 018B   224
                                 018B   225          $DS_DISPATCH
                                 018B                        .SAVE
                                00000000                     .PSECT  DISPATCH, LONG, NOWRT, NOEXE
                                 0000           DISPATCH:
                                00000000                     .PSECT  DISPATCH_X, LONG, NOWRT, NOEXE
00000000'00000000'00000000'00000000' 0000                    .LONG   0[6]
                 00000000'00000000' 0010
                                0000018B                     .RESTORE
                                 018B   226
                                 018B   227          .SBTTL  Hardware Parameter Table
                                 018B   228 ;++
                                 018B   229 ;
                                 018B   230 ;       The hardware parameter table holds all of the hardware related
                                 018B   231 ;       parameters which can be modified by the operator at run time.
                                 018B   232 ;       The Diagnostic Supervisor maintains a seperate copy of the
                                 018B   233 ;       hardware parameter table for each unit under test.
                                 018B   234 ;       The hardware parameter code section is used to actually supply
                                 018B   235 ;       the values to these locations.
                                 018B   236 ;--
                                 018B   237
                      0000018F   018B   238 KA_PTABLE::      .BLKL   1               ; pointer to the CPU parameter table.
                                 018F   239
```

```
                     018F    241          .SBTTL  Program Global Data Section
                     018F    242 ;++
                     018F    243 ;
                     018F    244 ; General data used by more than one module.
                     018F    245 ;
                     018F    246 ;--
                     018F    247          .SAVE
                  00000000   248          .PSECT  DATA,PAGE
                     0000    249
                     0000    250 DEV_REG:
                     0000    251
                     0000    252 ;
                     0000    253 ;       Miscellaneous data
                     0000    254 ;
                     0000    255
            00000000 0000    256 HARDCNT::        .LONG   0               ; Exerciser summary hard
                     0004    257                                          ;  error count.
            0000000C 0004    258 Q_TEST_TIME:     .BLKQ   1               ; Test start time, used to calculate
                     000C    259                                          ;  test run time for summary report.
                     000C    260
                     000C    261
                     000C    262 ;+
                     000C    263 ; Test summary information data block
                     000C    264 ;
                     000C    265 ;       The test summary information block is dynamically allocated, based
                     000C    266 ;       on the number of tests in the program.  There is one entry per test
                     000C    267 ;       which contains all the pertinent error and run time information.
                     000C    268 ;       This information is displayed by the SUMMARY procedure, which is
                     000C    269 ;       invoked at the end of a program run or by an operator 'SUMMARY'
                     000C    270 ;       command.
                     000C    271 ;
                     000C    272 ;--
                     000C    273
            00000014 000C    274 SUM_A_BLOCK:     .BLKL   2               ; Two longword array to receive the
                     0014    275                                          ;  start and end addresses of the
                     0014    276                                          ;  summary information data block.
                     0014    277          .SAVE
                     0014    278          .PSECT  $ABS$
            00000000 0610    279          . = 0
                     0000    280 $DEF     SUM_L_HARDCNT, .BLKL, 1         ; Hard error count entry displacement.
                     0000                 .IIF    NB,SUM_L_HARDCNT,       SUM_L_HARDCNT::
            00000004 0000                 .IIF    NB,.BLKL,       .BLKL   1
                     0004    281 $DEF     SUM_Q_RUNTIME, .BLKQ, 1         ; Run time entry displacement.
                     0004                 .IIF    NB,SUM_Q_RUNTIME,       SUM_Q_RUNTIME::
            0000000C 0004                 .IIF    NB,.BLKQ,       .BLKQ   1
                     000C    282 $DEF     SUM_K_SIZE                      ; Size of summary information block
                     000C                 .IIF    NB,SUM_K_SIZE,  SUM_K_SIZE::
                     000C    283                                          ;  for one test.
                  00000014   284          .RESTORE
                     0014    285
            00000018 0014    286 SUM_BLK_SIZE:    .BLKL   1               ; Number of pages for summary info block.
```

```
                                0018   288          .SBTTL  Program Text Section
                                0018   289 ;++
                                0018   290 ;
                                0018   291 ;       This section contains all the common character string type data
                                0018   292 ;       entries.
                                0018   293 ;
                                0018   294 ;--
                                0018   295
                                0018   296 ;+
                                0018   297 ; Program section names:
                                0018   298 ;                       MANUAL,TU58,POWER,HALT,MCHK,IPR,WCS,ALL,MEM,UBI,
                                0018   299 ;                       LDPCTX
                                0018   300 ;-
                                0018   301
                                0018   302          $DS_SECTION    MANUAL,TU58,POWER,HALT,MCHK,IPR,WCS,ALL,MEM,UBI,-
                                0018   303                         LDPCTX
                                0018       S_DEFAULT:
      54 4C 55 41 46 45 44 00'  0018                   .ASCIC  \DEFAULT\
                            07  0018
                                0020       S_MANUAL:
         4C 41 55 4E 41 4D 00'  0020                   .ASCIC  \MANUAL\
                            06  0020
                                0027       S_TU58:
               38 35 55 54 00'  0027                   .ASCIC  \TU58\
                            04  0027
                                002C       S_POWER:
            52 45 57 4F 50 00'  002C                   .ASCIC  \POWER\
                            05  002C
                                0032       S_HALT:
               54 4C 41 48 00'  0032                   .ASCIC  \HALT\
                            04  0032
                                0037       S_MCHK:
            4B 48 43 4D 00'     0037                   .ASCIC  \MCHK\
                            04  0037
                                003C       S_IPR:
               52 50 49 00'     003C                   .ASCIC  \IPR\
                            03  003C
                                0040       S_WCS:
               53 43 57 00'     0040                   .ASCIC  \WCS\
                            03  0040
                                0044       S_ALL:
               4C 4C 41 00'     0044                   .ASCIC  \ALL\
                            03  0044
                                0048       S_MEM:
               4D 45 4D 00'     0048                   .ASCIC  \MEM\
                            03  0048
                                004C       S_UBI:
               49 42 55 00'     004C                   .ASCIC  \UBI\
                            03  004C
                                0050       S_LDPCTX:
   58 54 43 50 44 4C 00'        0050                   .ASCIC  \LDPCTX\
                            06  0050
                                0057       SECTION:
                  0000000C      0057                   .LONG   12
                  00000018'     005B                   .LONG   S_DEFAULT
                  00000020'     005F                   .LONG   S_MANUAL
                  00000027'     0063                   .LONG   S_TU58
```

```
                          0000002C'  0067                          .LONG    S_POWER
                          00000032'  006B                          .LONG    S_HALT
                          00000037'  006F                          .LONG    S_MCHK
                          0000003C'  0073                          .LONG    S_IPR
                          00000040'  0077                          .LONG    S_WCS
                          00000044'  007B                          .LONG    S_ALL
                          00000048'  007F                          .LONG    S_MEM
                          0000004C'  0083                          .LONG    S_UBI
                          00000050'  0087                          .LONG    S_LDPCTX
                                     008B   304
                                     008B   305
                                     008B   306 ;+
                                     008B   307 ; Device mnemonics list:
                                     008B   308 ;       KA730 = VAX-11/730 CPU
                                     008B   309 ;-
                                     008B   310
                                     008B   311         $DS_DEVTYP          <KA730,TU58,UBE>
               30 33 37 41 4B 00'    008B       T_KA730:                    .ASCIC  \KA730\
                               05    008B
                  38 35 55 54 00'    0091       T_TU58:            .ASCIC   \TU58\
                               04    0091
                     45 42 55 00'    0096       T_UBE:             .ASCIC   \UBE\
                               03    0096
                               00    009A               .BYTE   0
                          00000003   009B       AL_DEVTYP:         .LONG    $$N
                          0000008B'  009F                          .LONG    T_KA730
                          00000091'  00A3                          .LONG    T_TU58
                          00000096'  00A7                          .LONG    T_UBE
                                     00AB   312
                                     00AB   313 ;+
                                     00AB   314 ; ASCII buffers
                                     00AB   315 ;-
                          0000000B   00AB   316 BUF_SIZE = 11
                                     00AB   317 QWD_TIME:
                         0000 000B   00AB   318         .WORD   BUF_SIZE, 0
                          000000B3'  00AF   319         .LONG   T_TIME
                                     00B3   320
                          000000BE   00B3   321 T_TIME: .BLKB   BUF_SIZE
                                     00BE   322
                                     00BE   323
                                     00BE   324 ;+
                                     00BE   325 ; Formatted ASCII output statements
                                     00BE   326 ;-
                                     00BE   327
                                     00BE   328 FMT_SUMMARY:                     ; FAO message for module summary printout.
20 57 5A 21 20 74 73 65 54 2F 21 00' 00BE   329         .ASCIC  "'!/Test !ZW had !ZL hard error!%s, !AS run time."'
64 72 61 68 20 4C 5A 21 20 64 61 68  00CA
21 20 2C 73 25 21 72 6F 72 72 65 20  00D6
2E 65 6D 69 74 20 6E 75 72 20 53 41  00E2
                               2F    00BE
                                     00EE   330
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 00' 00EE   331 MM_MSG: .ASCIC  '*********** UNABLE TO SHUT OFF MEMORY MANAGEMENT ***********'
53 20 4F 54 20 45 4C 42 41 4E 55 20  00FA
4F 4D 45 4D 20 46 46 4F 20 54 55 48  0106
4E 45 4D 45 47 41 4E 41 4D 20 59 52  0112
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 20 54  011E
                               2A    012A
```

```
                        3C  00EE
                            012B    332 ; [01]
                            012B    333
```

```
                     335          .SBTTL  Initialization Procedure
                     336 ;++
                     337 ; Functional description:
                     338 ;
                     339 ;       This routine will be executed at the beginning of each pass.
                     340 ;       This section is responsible for control of:
                     341 ;
                     342 ;       A) a double iteration of the exerciser to make up 1 pass;
                     343 ;       B) test summary information block initialization.
                     344 ;
                     345 ; Calling sequence:
                     346 ;
                     347 ;       The Diagnostic Supervisor automatically invokes this routine with
                     348 ;       a procedure call instruction.
                     349 ;
                     350 ; Input parameters:
                     351 ;
                     352 ;       INTERS_FLAG = 0 representing 1st iteration of a double iteration
                     353 ;               of the dispatch table to make up 1 pass.
                     354 ;
                     355 ; Implicit inputs:      None
                     356 ;
                     357 ; Output parameters:    None
                     358 ;
                     359 ; Implicit outputs:     None
                     360 ;
                     361 ; Completion codes:     None
                     362 ;
                     363 ; Side effects:         None
                     364 ;
                     365 ;--
```

```
                                012B   367          $DS_BGNINIT
                                012B                .SAVE
                            00000000                .PSECT  INITIALIZE,LONG
                                0000            INITIALIZE:
                              0000  0000                .WORD   ^M<>                ; ENTRY MASK
    00000000'EF                 D4  0002   368          CLRL    HARDCNT                     ; Initialize the test summary
                                0008   369                                              ;  hard errors counter.
                                0008   370          $DS_BPASS0      0$
                                1D  E0  0008                BBS     #DSA$V_PASS0,-      ; BR IF PASS 0
                  03 0000FE00 9F        000A                        @#DSA$GL_FLAGS, 0$
                          0248  31  0010   371          BRW     2000$                       ; br = not Pass 0
                                0013   372
                                0013   373 0$:       $DS_MMOFF_G                         ; Turn memory management off [01]
                00010158 9F   6E  FA  0013                CALLG   (SP), @#DS$MMOFF
                          14 50   E8  001A   374          BLBS    R0,2$                   ; If operator has turned it on [01]
                                001D   375          $DS_PRINTF_S    FORMAT=MM_MSG       ; then program is unable to turn [01]
                000000EE'EF     9F  001D                PUSHAB  MM_MSG
                000100F0 9F     01  FB  0023                CALLS   #$$N, @#DS$PRINTF
                                002A   376          $DS_ABORT       PROGRAM             ; it off , hence abort program [01]
                00010020 9F     6C  FA  002A                CALLG   (AP), @#DS$ABORT
                                0031   377
    0000007D'EF     FFFFFFFF    8F  D0  0031   378 2$:       MOVL            #-1,BE0_UNIT_NO         ; load default value
    00000081'EF     FFFFFFFF    8F  D0  003C   379           MOVL            #-1,BE1_UNIT_NO         ; load default value
    00000085'EF     FFFFFFFF    8F  D0  0047   380           MOVL            #-1,BE2_UNIT_NO         ; load default value
    00000089'EF     FFFFFFFF    8F  D0  0052   381           MOVL            #-1,BE3_UNIT_NO         ; load default value
    0000018B'EF     FFFFFFFF    8F  D0  005D   382           MOVL            #-1,KA_PTABLE           ; load default value
    00000068'EF     FFFFFFFF    8F  D0  0068   383           MOVL            #-1,L_LAST_UBE          ; load default value
                    0000006C'EF     94  0073   384           CLRB            B_TU58_DRIVE            ; clear drive flags
                                54  D4  0079   385           CLRL            R4                      ; clear unit number
                                    007B   386 10$:      $DS_GPHARD_S    R4,A_P_TABLE            ; get p table address
            0000006D'EF             DF  007B                        PUSHAL  A_P_TABLE
                                54  DD  0081                       PUSHL   R4
        00010018 9F             02  FB  0083                       CALLS   #2, @#DS$GPHARD
                                    008A   387          $DS_BNERROR     15$                     ; br = p tables not exhausted
                                50  DD  008A                       PUSHL   R0
                                01  DD  008C                       PUSHL   #$ER
                                27  DD  008E                       PUSHL   #DS$K_BNERROR
        000100A8 9F             03  FB  0090                       CALLS   #3, @#DS$BRANCH
                          03 50   E8  0097                       BLBS    R0, 15$
                           00CD   31  009A   388                BRW             170$                    ; br = p tables exhausted
    55      0000006D'EF         D0  009D   389 15$:       MOVL            A_P_TABLE,R5            ; load p table address
    00000096'EF     26 A5       D1  00A4   390           CMPL            HP$T_TYPE(R5),T_UBE     ; UBE?
                        3B      12  00AC   391           BNEQ            70$                     ; br = not UBE
            00000068'EF         D6  00AE   392           INCL            L_LAST_UBE              ; count UBE
     03     00      0B A5       8F  00B4   393           CASEB           HP$B_DRIVE(R5),#0,#3    ; br for UBE number
                            0008' 00B9   394 20$:                       .WORD   30$-20$         ; if UBE 0
                            0012' 00BB   395                            .WORD   40$-20$         ; if UBE 1
                            001C' 00BD   396                            .WORD   50$-20$         ; if UBE 2
                            0026' 00BF   397                            .WORD   60$-20$         ; if UBE 3
    0000007D'EF             54  D0  00C1   398 30$:       MOVL            R4,BE0_UNIT_NO          ; save unit number
                       009A     31  00C8   399           BRW             120$
    00000081'EF             54  D0  00CB   400 40$:       MOVL            R4,BE1_UNIT_NO          ; save unit number
                       0090     31  00D2   401           BRW             120$
    00000085'EF             54  D0  00D5   402 50$:       MOVL            R4,BE2_UNIT_NO          ; save unit number
                       0086     31  00DC   403           BRW             120$
    00000089'EF             54  D0  00DF   404 60$:       MOVL            R4,BE3_UNIT_NO          ; save unit number
                       007C     31  00E6   405           BRW             120$                    ; br to end of loop
```

```
                0000008B'EF   26 A5   D1  00E9   406 70$:    CMPL        HP$T_TYPE(R5),T_KA730   ; KA730?
                              2F   12  00F1   407         BNEQ        80$                     ; br = not KA730
                0000008F'9F   2A A5   B1  00F3   408         CMPW        <HP$T_TYPE+4>(R5),@#<T_KA730+4> ; KA730?
                              25   12  00FB   409         BNEQ        80$                     ; br = not KA730
                   00000071'EF    54   D0  00FD   410         MOVL        R4,KA_UNIT_NO           ; save CPU unit number
             0000018B'EF   0000006D'EF   D0  0104   411         MOVL        A_P_TABLE,KA_PTABLE     ; load KA730 p table address
                0000008D'EF   0C A5   D0  010F   412         MOVL        HP$KA_L_FLAGS(R5),L_KA_FLAGS ; save ka flags
                00000091'EF   44 A5   D0  0117   413         MOVL        HP$KA_MEM_SIZE(R5),L_KSIZE_PTBL ; save memory size
                                0043  31  011F   414         BRW         120$
                00000091'EF   26 A5   D1  0122   415 80$:    CMPL        HP$T_TYPE(R5),T_TU58    ; TU58?
                              39   12  012A   416         BNEQ        120$                    ; br = not TU58
                00000095'9F   2A A5   91  012C   417         CMPB        <HP$T_TYPE+4>(R5),@#<T_TU58+4> ; TU58?
                              2F   12  0134   418         BNEQ        120$                    ; br = not TU58
                   55   0000006D'EF    D0  0136   419         MOVL        A_P_TABLE,R5            ; load p table address
                    01   00   0B A5    8F  013D   420         CASEB       HP$B_DRIVE(R5),#0,#1    ; which drive?
                                     0004' 0142   421 90$:                .WORD   100$-90$        ; if drive 0
                                     0015' 0144   422                     .WORD   110$-90$        ; if drive 1
                0000006C'EF    01   C8  0146   423 100$:   BISL2       #1,B_TU58_DRIVE         ; set bit for drive 0
                00000075'EF    54   D0  014D   424         MOVL        R4,TU0_UNIT_NO          ; save drive 0 unit number
                                000E  31  0154   425         BRW         120$                    ; br to end of loop
                00000079'EF    54   D0  0157   426 110$:   MOVL        R4,TU1_UNIT_NO          ; save drive 1 unit number
                0000006C'EF    02   C8  015E   427         BISL2       #2,B_TU58_DRIVE         ; set bit for drive 1
                               54   D6  0165   428 120$:   INCL        R4                      ; incr unit number
                             FF11   31  0167   429         BRW         10$                     ; br to beginning of loop
                     0000012B'EF    16  016A   430 170$:   JSB     SUMMARY_INIT                ; Structure module summary
                                        0170   431                                             ; Information block
                               55   D4  0170   432         CLRL    R5                          ; Clear count
                                        0172   433                                             ; Look for free memory above
                                        0172   434 300$:   $DS_GETBUF_S    PAGCNT=#1, RETADR=A_FREE_MEM        ; supervisor
                               00   DD  0172                       PUSHL   #0
                               00   DD  0174                       PUSHL   #0
                     00000048'EF    7F  0176                       PUSHAQ  A_FREE_MEM
                               01   DD  017C                       PUSHL   #1
                00010120 9F    04   FB  017E                       CALLS   #4, @#DS$GETBUF
                                        0185   435         $DS_BNCOMPLETE  500$                ; br = unsuccessful
                                  11 50  E9  0185                   BLBC    R0, 500$
                               55   D6  0188   436         INCL    R5                          ; count buffers allocated
             00020000 8F   00000048'EF    D1  018A   437         CMPL    A_FREE_MEM,#^X20000         ; beyond 1st 128k ?
                               DB   19  0195   438         BLSS    300$                        ; br = not beyond 1st 128k
                               06   11  0197   439         BRB     900$                        ; br = successful
                     00000048'EF    7C  0199   440 500$:   CLRQ    A_FREE_MEM                  ; clear buffer address
                                        019F   441
                                        019F   442 900$:   $DS_RELBUF_S    PAGCNT=#1           ; release buffers
                               00   DD  019F                       PUSHL   #0
                               00   DD  01A1                       PUSHL   #0
                               01   DD  01A3                       PUSHL   #1
                00010128 9F    03   FB  01A5                       CALLS   #3, @#DS$RELBUF
                               55   D7  01AC   443         DECL    R5                          ; count page
                               EF   12  01AE   444         BNEQ    900$                        ; br = not done
                                        01B0   445
                                        01B0   446         $DS_GETBUF_S    PAGCNT=#3, RETADR=A_BUFFER_BEGIN
                               00   DD  01B0                       PUSHL   #0
                               00   DD  01B2                       PUSHL   #0
                     00000000'EF    7F  01B4                       PUSHAQ  A_BUFFER_BEGIN
                               03   DD  01BA                       PUSHL   #3
                00010120 9F    04   FB  01BC                       CALLS   #4, @#DS$GETBUF
                                        01C3   447         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_PWR_BUFFER
```

```
                    00    DD  01C3                        PUSHL   #0
                    00    DD  01C5                        PUSHL   #0
         00000008'EF    7F  01C7                        PUSHAQ  A_PWR_BUFFER
                    01    DD  01CD                        PUSHL   #1
     00010120 9F    04    FB  01CF                        CALLS   #4, @#DS$GETBUF
                              01D6    448         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_HALT_BUFFER
                    00    DD  01D6                        PUSHL   #0
                    00    DD  01D8                        PUSHL   #0
         00000010'EF    7F  01DA                        PUSHAQ  A_HALT_BUFFER
                    01    DD  01E0                        PUSHL   #1
     00010120 9F    04    FB  01E2                        CALLS   #4, @#DS$GETBUF
                              01E9    449         $DS_GETBUF_S    PAGCNT=#1, RETADR=A_MCHK_BUFFER
                    00    DD  01E9                        PUSHL   #0
                    00    DD  01EB                        PUSHL   #0
         00000018'EF    7F  01ED                        PUSHAQ  A_MCHK_BUFFER
                    01    DD  01F3                        PUSHL   #1
     00010120 9F    04    FB  01F5                        CALLS   #4, @#DS$GETBUF
                              01FC    450         $DS_GETBUF_S    PAGCNT=#10, RETADR=A_WCS_BUFFER
                    00    DD  01FC                        PUSHL   #0
                    00    DD  01FE                        PUSHL   #0
         00000020'EF    7F  0200                        PUSHAQ  A_WCS_BUFFER
                    0A    DD  0206                        PUSHL   #10
     00010120 9F    04    FB  0208                        CALLS   #4, @#DS$GETBUF
                              020F    451         $DS_GETBUF_S    PAGCNT=#8, RETADR=A_UBE_BUFFER_0
                    00    DD  020F                        PUSHL   #0
                    00    DD  0211                        PUSHL   #0
         00000028'EF    7F  0213                        PUSHAQ  A_UBE_BUFFER_0
                    08    DD  0219                        PUSHL   #8
     00010120 9F    04    FB  021B                        CALLS   #4, @#DS$GETBUF
                              0222    452         $DS_GETBUF_S    PAGCNT=#8, RETADR=A_UBE_BUFFER_1
                    00    DD  0222                        PUSHL   #0
                    00    DD  0224                        PUSHL   #0
         00000030'EF    7F  0226                        PUSHAQ  A_UBE_BUFFER_1
                    08    DD  022C                        PUSHL   #8
     00010120 9F    04    FB  022E                        CALLS   #4, @#DS$GETBUF
                              0235    453         $DS_GETBUF_S    PAGCNT=#8, RETADR=A_UBE_BUFFER_2
                    00    DD  0235                        PUSHL   #0
                    00    DD  0237                        PUSHL   #0
         00000038'EF    7F  0239                        PUSHAQ  A_UBE_BUFFER_2
                    08    DD  023F                        PUSHL   #8
     00010120 9F    04    FB  0241                        CALLS   #4, @#DS$GETBUF
                              0248    454         $DS_GETBUF_S    PAGCNT=#12, RETADR=A_UBE_BUFFER_3
                    00    DD  0248                        PUSHL   #0
                    00    DD  024A                        PUSHL   #0
         00000040'EF    7F  024C                        PUSHAQ  A_UBE_BUFFER_3
                    0C    DD  0252                        PUSHL   #12
     00010120 9F    04    FB  0254                        CALLS   #4, @#DS$GETBUF
                              025B    455 2000$:
                              025B    456
                              025B    457         $DS_ENDPASS_G
     00010010 9F    6E    FA  025B                        CALLG   (SP), @#DS$ENDPASS
                              0262    458         $DS_ENDINIT
                              0262        INITIALIZE_X:
     00010058 9F    6E    FA  0262                        CALLG   (SP), @#DS$BREAK
                        04  0269                        RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                    0000012B                            .RESTORE
```

```
                        012B   460          .SBTTL  Clean-up Procedure
                        012B   461 ;++
                        012B   462 ; Functional description:
                        012B   463 ;
                        012B   464 ;       This routine is executed at the completion of a program run.
                        012B   465 ;       This section is responsible for:
                        012B   466 ;
                        012B   467 ;       A) reinitialization of flags controlling double iteration per pass;
                        012B   468 ;       B) activation of a statistical summary report ($DS_SUMMARY_G);
                        012B   469 ;       C) releasing all temporary buffers.
                        012B   470 ;
                        012B   471 ; Calling sequence:
                        012B   472 ;
                        012B   473 ;       The Diagnostic Supervisor automatically invokes this routine with
                        012B   474 ;       a procedure call instruction.
                        012B   475 ;
                        012B   476 ; Input parameters:     None
                        012B   477 ;
                        012B   478 ; Implicit inputs:
                        012B   479 ;
                        012B   480 ;       The summary information block to be invoked by the $DS_SUMMARY_G
                        012B   481 ;       call has been dynamically filled at run time.
                        012B   482 ;
                        012B   483 ; Output parameters:    None
                        012B   484 ;
                        012B   485 ; Implicit outputs:     None
                        012B   486 ;
                        012B   487 ; Completion codes:     None
                        012B   488 ;
                        012B   489 ; Side effects:
                        012B   490 ;
                        012B   491 ;       The test summary information is destroyed after it is reported.
                        012B   492 ;
                        012B   493 ;--
                        012B   494
                        012B   495          $DS_BGNCLEAN
                        012B                        .SAVE
                    00000000                        .PSECT  CLEANUP,LONG
                        0000            CLEAN_UP:
                    0000  0000                      .WORD   ^M<>                ; ENTRY MASK
                        0002   496          $DS_CLRVEC_S   VECTOR=#SCB$L_TIMER
         000000C0 8F    DD   0002                       PUSHL   #SCB$L_TIMER
    00010168 9F    01    FB   0008                       CALLS   #1, @#DS$CLRVEC
                        000F   497          $DS_CLRVEC_S   VECTOR=#SCB$L_POWER
                 0C     DD   000F                       PUSHL   #SCB$L_POWER
    00010168 9F    01    FB   0011                       CALLS   #1, @#DS$CLRVEC
                        0018   498          $DS_CLRVEC_S   VECTOR=#SCB$L_CHME
         00000044 8F    DD   0018                       PUSHL   #SCB$L_CHME
    00010168 9F    01    FB   001E                       CALLS   #1, @#DS$CLRVEC
                        0025   499          $DS_CLRVEC_S   VECTOR=#SCB$L_CHMS
         00000048 8F    DD   0025                       PUSHL   #SCB$L_CHMS
    00010168 9F    01    FB   002B                       CALLS   #1, @#DS$CLRVEC
                        0032   500          $DS_CLRVEC_S   VECTOR=#SCB$L_CHMU
         0000004C 8F    DD   0032                       PUSHL   #SCB$L_CHMU
    00010168 9F    01    FB   0038                       CALLS   #1, @#DS$CLRVEC
                        003F   501          $DS_CLRVEC_S   VECTOR=#SCB$L_CHMK
         00000040 8F    DD   003F                       PUSHL   #SCB$L_CHMK
```

```
                00010168 9F    01    FB  0045                        CALLS   #1, @#DS$CLRVEC
                                         004C    502         $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                               04    DD  004C                        PUSHL   #SCB$L_MACHCHK
                00010168 9F    01    FB  004E                        CALLS   #1, @#DS$CLRVEC
                                         0055    503         $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                               14    DD  0055                        PUSHL   #SCB$L_OPCCUS
                00010168 9F    01    FB  0057                        CALLS   #1, @#DS$CLRVEC
                                         005E    504         $DS_CLRVEC_S    VECTOR=#SCB$L_OPCDEC
                               10    DD  005E                        PUSHL   #SCB$L_OPCDEC
                00010168 9F    01    FB  0060                        CALLS   #1, @#DS$CLRVEC
                                         0067    505         $DS_CLRVEC_S    VECTOR=#SCB$L_UBE_0
                     00000348 8F      DD  0067                        PUSHL   #SCB$L_UBE_0
                00010168 9F    01    FB  006D                        CALLS   #1, @#DS$CLRVEC
                                         0074    506         $DS_CLRVEC_S    VECTOR=#SCB$L_UBE_1
                     0000034C 8F      DD  0074                        PUSHL   #SCB$L_UBE_1
                00010168 9F    01    FB  007A                        CALLS   #1, @#DS$CLRVEC
                                         0081    507         $DS_CLRVEC_S    VECTOR=#SCB$L_UBE_2
                     00000350 8F      DD  0081                        PUSHL   #SCB$L_UBE_2
                00010168 9F    01    FB  0087                        CALLS   #1, @#DS$CLRVEC
                                         008E    508         $DS_CLRVEC_S    VECTOR=#SCB$L_UBE_3
                     00000354 8F      DD  008E                        PUSHL   #SCB$L_UBE_3
                00010168 9F    01    FB  0094                        CALLS   #1, @#DS$CLRVEC
                                         009B    509         $DS_MMOFF_G                     ; Turn memory management off
                00010158 9F    6E    FA  009B                        CALLG   (SP), @#DS$MMOFF
                         07 50 E8        00A2    510         BLBS    R0,2$                   ; If operator has turned it on [01]
                                         00A5    511         $DS_ABORT       PROGRAM         ; it off , hence abort program [01]
                00010020 9F    6C    FA  00A5                        CALLG   (AP), @#DS$ABORT
                                         00AC    512 2$:     $DS_INITSCB_G
                00010170 9F    6E    FA  00AC                        CALLG   (SP), @#DS$INITSCB
                                         00B3    513 ;       $DS_SUMMARY_G
                                         00B3    514         $DS_RELBUF_S    PAGCNT=#52      ; release all data buffers
                               00    DD  00B3                        PUSHL   #0
                               00    DD  00B5                        PUSHL   #0
                               34    DD  00B7                        PUSHL   #52
                00010128 9F    03    FB  00B9                        CALLS   #3, @#DS$RELBUF
                                         00C0    515         $DS_RELBUF_S    SUM_BLK_SIZE    ; Release summary data block.
                               00    DD  00C0                        PUSHL   #0
                               00    DD  00C2                        PUSHL   #0
                     00000014'EF      DD  00C4                        PUSHL   SUM_BLK_SIZE
                00010128 9F    03    FB  00CA                        CALLS   #3, @#DS$RELBUF
                          1C   00    DA  00D1    516         MTPR    #0,#PR$_CSRS            ; disable TU58 interrupt
                          1E   00    DA  00D4    517         MTPR    #0,#PR$_CSTS            ; disable TU58 interrupt
                                         00D7    518         $DS_SETIPL_S    LEVEL=#0        ; set interrupt level
                               00    DD  00D7                        PUSHL   #0
                00010178 9F    01    FB  00D9                        CALLS   #1, @#DS$SETIPL
00F20004'EF     02000000 8F    CA  00E0    519         BICL2   #M_DSABL_ECC,A_MEM_CSR_1 ; re-enable memory ECC
                                         00EB    520         $DS_ENDCLEAN
                                         00EB    CLEAN_UP_X:
                00010058 9F    6E    FA  00EB                        CALLG   (SP), @#DS$BREAK
                                   04    00F2                        RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                         0000012B                    .RESTORE
```

```
                        522          .SBTTL  Summary Report Procedure
                        523 ;++
                        524 ; Functional description:
                        525 ;
                        526 ;       This routine issues a summary report upon completion of a run or
                        527 ;       upon request from the operator.
                        528 ;       Information is taken from the 'head' of the test summary
                        529 ;       queue (SUMMARY_INFO) and printed out in the format shown
                        530 ;       below under ''output parameters:''.  The queue is accessed until
                        531 ;       empty.
                        532 ;
                        533 ; Calling sequence:
                        534 ;
                        535 ;       CALL    SUMMARY
                        536 ;
                        537 ; Input parameters:     None
                        538 ;
                        539 ; Implicit inputs:
                        540 ;
                        541 ;       The test summary information block has been dynamically filled at
                        542 ;       run time.
                        543 ;
                        544 ; Output parameters:
                        545 ;
                        546 ;       Any tests having errors will have information printed out in the
                        547 ;       following format:
                        548 ;*
                        549 ;       Test xx had yy hard errors; hh:mm:ss run time.
                        550 ;
                        551 ;       Where,
                        552 ;
                        553 ;       xx      represents the relevent test number.
                        554 ;       yy      represents the accumulated number of hard errors
                        555 ;                encountered in the course of fulfillment of the
                        556 ;                pass requirements.
                        557 ;
                        558 ;       hh:mm:ss represents the accumulated run time encountered for the
                        559 ;                test in the course of fulfillment of pass requirements.
                        560 ;
                        561 ;       Note!!!!! any test not encountering errors will not appear
                        562 ;       as part of the summary printout (i.e. if no errors existed
                        563 ;       in the execution of the exerciser there would be no summary
                        564 ;       printout).
                        565 ;*
                        566 ;
                        567 ; Implicit outputs:     None
                        568 ;
                        569 ; Completion codes:     None
                        570 ;
                        571 ; Side effects:         None
                        572 ;
                        573 ;--
```

```
                                012B   575          $DS_BGNSUMMARY  <R2, R3>
                                012B                .SAVE
                            00000000                .PSECT  SUMMARY,LONG
                                0000            SUMMARY:
                                000C   0000         .WORD   ^M<R2, R3>                 ; ENTRY MASK
    52   0000000C'EF            D0   0002   576      MOVL    SUM_A_BLOCK, R2           ; Get pointer to summary info.
         53   01                D0   0009   577      MOVL    #1, R3                    ; Test number.
                                000C   578 1$:                                         ; DO
              62                D5   000C   579      TSTL    SUM_L_HARDCNT(R2)         ; Check for any errors and
              2B                13   000E   580      BEQL    2$                        ;  skip report if none.
                                0010   581      $ASCTIM_S -                           ; Convert run time to ASCII:
                                0010   582              TIMBUF=QWD_TIME, -            ;  QWD of buffer for ASCII time;
                                0010   583              TIMADR=SUM_Q_RUNTIME(R2), -   ;  64-bit run time;
                - -             0010   584              CVTFLG=#1                     ;  hh:mm:ss only flag.
                01              DD   0010                PUSHL   #1
            04 A2               7F   0012                PUSHAQ  SUM_Q_RUNTIME(R2)
    000000AB'EF                 7F   0015                PUSHAQ  QWD_TIME
                00              DD   001B                PUSHL   #0
00010248'GF     04              FB   001D                CALLS   #4,G^SYS$ASCTIM
                                0024   585      $DS_PRINTS_S    FMT_SUMMARY, -        ; Display summary information:
                                0024   586              R3, -                         ;  Test number;
                                0024   587              SUM_L_HARDCNT(R2), -          ;  Number of hard errors;
                                0024   588              #QWD_TIME                     ;  Test run time.
    000000AB'8F                 DD   0024                PUSHL   #QWD_TIME
              62                DD   002A                PUSHL   SUM_L_HARDCNT(R2)
              53                DD   002C                PUSHL   R3
    000000BE'EF                 9F   002E                PUSHAB  FMT_SUMMARY
000100F8 9F                     04   FB   0034          CALLS   #$$N, @#DS$PRINTS
         52   0C                C0   0038   589 2$:     ADDL2   #SUM_K_SIZE, R2           ; Update block pointer to next
                                003E   590                                             ;  test.
C6 53    00000054'FF            F3   003E   591      AOBLEQ  @L$A_TSTCNT, R3, 1$       ;   WHILE test # LEQ last test.
                                0046   592                                             ; ENDDO
                                0046   593      $DS_ENDSUMMARY
                                0046       SUMMARY_X:
00010058 9F     6E              FA   0046            CALLG   (SP), @#DS$BREAK
                                04   004D            RET                               ; RETURN TO COMMAND MODE
                            0000012B                .RESTORE
```

```
                        595          .SBTTL  Summary Information Block Initialization Subroutine
                        596 ;++
                        597 ; Functional description:
                        598 ;
                        599 ;       This subroutine is responsible for building the structure for the
                        600 ;       exerciser test summary information block.  This block will contain
                        601 ;       the number of hard errors and run time for each test, which will be
                        602 ;       displayed when a SUMMARY is requested.
                        603 ;
                        604 ; Calling sequence:
                        605 ;
                        606 ;       JSB     SUMMARY_INIT
                        607 ;
                        608 ; Input parameters:     None
                        609 ;
                        610 ; Implicit inputs:
                        611 ;
                        612 ;       L$A_TSTCNT      = Pointer to the number of tests in this program;
                        613 ;       SUM_K_SIZE      = Number of bytes to be reserved for each test's data.
                        614 ;
                        615 ; Output parameters:    None
                        616 ;
                        617 ; Implicit outputs:
                        618 ;
                        619 ;       SUM_A_BLOCK     = Pointer to the memory reserved for the summary data;
                        620 ;       @SUM_A_BLOCK    = The summary data block itself is cleared.
                        621 ;
                        622 ; Completion codes:     None
                        623 ;
                        624 ; Side effects:         None
                        625 ;
                        626 ;--
```

```
                                        012B   628 SUMMARY_INIT:
                                  0C    BB  012B   629          PUSHR   #^M<R2, R3>
        52  0C  00000054'FF       C5    012D   630          MULL3   @L$A_TSTCNT, -          ; Compute the amount of buffer required
                                        0135   631                   #SUM_K_SIZE, R2       ;  for the summary information block.
        53  52  00000200 8F       C7    0135   632          DIVL3   #512, R2, R3          ; Convert to pages,
                                  53    D6  013D   633          INCL    R3                    ;  and round up.
                                        013F   634          $DS_GETBUF_S  R3, SUM_A_BLOCK ; Request the buffer from the Supervisor.
                                  00    DD  013F                        PUSHL   #0
                                  00    DD  0141                        PUSHL   #0
                              FEC5 CF   7F  0143                        PUSHAQ  SUM_A_BLOCK
                                  53    DD  0147                        PUSHL   R3
                00010120 9F       04    FB  0149                        CALLS   #4, @#DS$GETBUF
                                        0150   635          $DS_BNERROR     10$
                                  50    DD  0150                        PUSHL   R0
                                  01    DD  0152                        PUSHL   #$ER
                                  27    DD  0154                        PUSHL   #DS$K_BNERROR
                000100A8 9F       03    FB  0156                        CALLS   #3, @#DS$BRANCH
                             0F   50    E8  015D                        BLBS    R0, 10$
                                        0160   636          $DS_ERRSYS_S
                                  00    DD  0160                        PUSHL   #0
                                  00    DD  0162                        PUSHL   #0
                                  00    DD  0164                        PUSHL   #0
                                  01    DD  0166                        PUSHL   #$ER
                                        0168       ;;; GLOBAL ERROR 1
                000100C0 9F       04    FB  0168                        CALLS   #$$M, @#DS$ERRSYS
                                        016F   637 10$:
            FEA0 CF   53    D0  016F   638          MOVL    R3, SUM_BLK_SIZE      ; Save the page count.
            52   FE94 CF    C0  0174   639          ADDL2   SUM_A_BLOCK, R2       ; Compute end of block address
        FE91 CF   52    01    C3  0179   640          SUBL3   #1, R2, SUM_A_BLOCK+4 ;  and save it.
                        72    D4  017F   641 20$:     CLRL    -(R2)                 ; Clear out
            FE86 CF   52    D1  0181   642          CMPL    R2, SUM_A_BLOCK       ;  the summary information
                        F7    14  0186   643          BGTR    20$                   ;  block.
                                  0188   644
                        0C    BA  0188   645          POPR    #^M<R2, R3>
                              05  018A   646          RSB                           ; Return to mainline code.
```

```
                        018B   648          .SBTTL  Summary Information Input Subroutine
                        018B   649 ;++
                        018B   650 ; Functional description:
                        018B   651 ;
                        018B   652 ;       This subroutine is responsible for dynamically filling the test
                        018B   653 ;       summary information block with error data at the end of each test.
                        018B   654 ;
                        018B   655 ; Calling sequence:
                        018B   656 ;
                        018B   657 ;       JSB PART_INIT
                        018B   658 ;
                        018B   659 ; Input parameters:     None
                        018B   660 ;
                        018B   661 ;
                        018B   662 ; Implicit inputs:
                        018B   663 ;
                        018B   664 ;       DSA$GL_TESTNO = Current test number;
                        018B   665 ;       HARDCNT =       Number of hard errors, this test, this pass;
                        018B   666 ;       Q_TEST_TIME =   Start time of test;
                        018B   667 ;       SUM_A_BLOCK =   Base addresss of the summary information block;
                        018B   668 ;       SUM_K_SIZE =    Size of an entry in the summary information block.
                        018B   669 ;
                        018B   670 ; Output parameters:    None
                        018B   671 ;
                        018B   672 ; Implicit outputs:
                        018B   673 ;
                        018B   674 ;       SUM_L_HARDCNT(R2)       = Number of accumulated hard errors;
                        018B   675 ;       SUM_Q_RUNTIME(R2)       = Accumulated test run time.
                        018B   676 ;
                        018B   677 ; Completion codes:     None
                        018B   678 ;
                        018B   679 ; Side effects:         None
                        018B   680 ;
                        018B   681 ;--
```

```
                        018B   683 PART_INIT::
52   0C   0000FE50'EF    C5 018B   684         MULL3   DSA$GL_TESTNO, -         ; Grab the current test number and
                        0193   685                  #SUM_R_SIZE, R2          ;  compute displacement into the
     52   FE75 CF       C0 0193   686         ADDL2   SUM_A_BLOCK, R2          ;  summary information block.
     62   FE64 CF       C0 0198   687         ADDL2   HARDCNT, -               ; Update the tests hard count total.
                        019D   688                  SUM_L_HARDCNT(R2)
          FE5F CF       D4 019D   689         CLRL    HARDCNT                  ; Reinitialize exerciser hard
                        01A1   690                                           ;  error count for next test.
                        01A1   691 ;
                        01A1   692 ; Compute current run time and add it to previous passes' run time.
                        01A1   693 ;
      04 A2   FE5F CF   C0 01A1   694         ADDL2   Q_TEST_TIME, -           ; Factor in absolute start time.
                        01A7   695                  SUM_Q_RUNTIME(R2)
      04 A2   FE59 CF   D8 01A7   696         ADWC    Q_TEST_TIME, SUM_Q_RUNTIME(R2)
                        01AD   697         $GETTIM_S       Q_TEST_TIME      ; Get current time.
              FE53 CF   7F 01AD                 PUSHAQ  Q_TEST_TIME
00010378'GF     01      FB 01B1                 CALLS   #1,G^SYS$GETTIM
      04 A2   FE48 CF   C2 01B8   698         SUBL2   Q_TEST_TIME, -           ; Factor out absolute time;
                        01BE   699                  SUM_Q_RUNTIME(R2)        ;  result is delta time.
      04 A2   FE42 CF   D9 01BE   700         SBWC    Q_TEST_TIME, SUM_Q_RUNTIME(R2)
                        01C4   701
                        05 01C4   702         RSB                              ;Back into the stream of things!
                        01C5   703
                        01C5   704         $DS_ENDMOD
                        01C5   705         .END
```

```
$$E                             = 00000001
$$M                             = 00000004
$$N                             = 00000000
$$S                             = FFFFFFFF
$ENV                            = 00000001  G
$ER                             = 00000002
$MO                             = 00000001  G
$ST                             = 00000000
$TN                             = 00000000
AEM                               00000171 R      02
AKM                               0000016A R      02
ALL                             = 00000008  G
AL_DEVTYP                         0000009B R      09
ASM                               0000017B R      02
AUM                               00000186 R      02
A_BUFFER_BEGIN                    00000000 RG     02
A_BUFFER_END                      00000004 RG     02
A_FREE_MEM                        00000048 RG     02
A_HALT_BUFFER                     00000010 RG     02
A_HEADEND                         00000058 R      04
A_MCHK_BUFFER                     00000018 RG     02
A_MEM_CSR_1                     = 00F20004
A_PSL_MNE                         00000115 RG     02
A_PWR_BUFFER                      00000008 RG     02
A_P_TABLE                         0000006D R      02
A_UBE_BUFFER_0                    00000028 RG     02
A_UBE_BUFFER_1                    00000030 RG     02
A_UBE_BUFFER_2                    00000038 RG     02
A_UBE_BUFFER_3                    00000040 RG     02
A_WCS_BUFFER                      00000020 RG     02
BE0_UNIT_NO                       0000007D RG     02
BE1_UNIT_NO                       00000081 RG     02
BE2_UNIT_NO                       00000085 RG     02
BE3_UNIT_NO                       00000089 RG     02
BIT...                          = 00000028
BUF_SIZE                        = 0000000B
B_TU58_DRIVE                      0000006C RG     02
CEP_FUNCTIONAL                  = 00000000
CEP_REPAIR                      = 00000001
CHC$_ABORT                      = 00000002  G
CHC$_CLEAR                      = 00000006  G
CHC$_CLRDFT                     = 00000009  G
CHC$_DSINT                      = 00000005  G
CHC$_ENINT                      = 00000004  G
CHC$_INITA                      = 00000000  G
CHC$_INITB                      = 00000001  G
CHC$_PURGE                      = 00000003  G
CHC$_SETDFT                     = 00000008  G
CHC$_STATUS                     = 00000007  G
CHF$L_MCHARGLST                   00000008
CHF$L_MCH_ARGS                    00000000
CHF$L_MCH_DEPTH                   00000008
CHF$L_MCH_FRAME                   00000004
CHF$L_MCH_SAVR0                   0000000C
CHF$L_MCH_SAVR1                   00000010
CHF$L_SIGARGLST                   00000004
CHF$L_SIG_ARG1                    00000008
CHF$L_SIG_ARGS                    00000000
CHF$L_SIG_NAME                    00000004
CHI$M_CHNINT                    = 00000001  G
CHI$M_DEVINT                    = 00000002  G
CHI$M_IPL                       = 0000007C  G
CHI$S_IPL                       = 00000005  G
CHI$V_CHNINT                    = 00000000  G
CHI$V_DEVINT                    = 00000001  G
CHI$V_IPL                       = 00000002  G
CHM$_FORWARD                    = 00000000  G
CHM$_INVALIDATE                 = 00000001  G
CHM$_MAP                        = 00000002  G
CHM$_MFWDN                      = 00000002  G
CHM$_MFWDNO                     = 0000000A  G
CHM$_MFWDV                      = 00000003  G
CHM$_MFWDVO                     = 0000000B  G
CHM$_MREVN                      = 00000006  G
CHM$_MREVNO                     = 0000000E  G
CHM$_MREVV                      = 00000007  G
CHM$_MREVVO                     = 0000000F  G
CHM$_NFWDN                      = 00000000  G
CHM$_NREVN                      = 00000004  G
CHM$_OFFSET                     = 00000008  G
CHM$_REVERSE                    = 00000004  G
CHS$M_BUSIC                     = 00800000  G
CHS$M_BUSINIT                   = 00400000  G
CHS$M_BUSNXM                    = 08000000  G
CHS$M_BUSPDN                    = 01000000  G
CHS$M_CHNDPE                    = 00000040  G
CHS$M_CHNERR                    = 00000002  G
CHS$M_CHNMPE                    = 00000080  G
CHS$M_CHPFOT                    = 00000100  G
CHS$M_DEVBUS                    = 00000010  G
CHS$M_DEVERR                    = 00000004  G
CHS$M_DEVTO                     = 00000020  G
CHS$M_ERRANY                    = 0000000F  G
CHS$M_MBAATN                    = 00100000  G
CHS$M_MBACPE                    = 00200000  G
CHS$M_MBADTB                    = 00040000  G
CHS$M_MBADTC                    = 00080000  G
CHS$M_MBAEXC                    = 00010000  G
CHS$M_MBANED                    = 00020000  G
CHS$M_MBAWCKLWR                 = 02000000  G
CHS$M_MBAWCKUPR                 = 04000000  G
CHS$M_PGMERR                    = 00000008  G
CHS$M_PGMHDE                    = 00000800  G
CHS$M_SYSERR                    = 00000001  G
CHS$M_SYSMEM                    = 00000200  G
CHS$M_SYSSBI                    = 00000400  G
CHS$V_BUSIC                     = 00000017  G
CHS$V_BUSINIT                   = 00000016  G
CHS$V_BUSNXM                    = 0000001B  G
CHS$V_BUSPDN                    = 00000018  G
CHS$V_CHNDPE                    = 00000006  G
CHS$V_CHNERR                    = 00000001  G
CHS$V_CHNMPE                    = 00000007  G
CHS$V_CHPFOT                    = 00000008  G
```

```
CHS$V_DEVBUS                    = 00000004  G
CHS$V_DEVERR                    = 00000002  G
CHS$V_DEVTO                     = 00000005  G
CHS$V_MBAATN                    = 00000014  G
CHS$V_MBACPE                    = 00000015  G
CHS$V_MBADTB                    = 00000012  G
CHS$V_MBADTC                    = 00000013  G
CHS$V_MBAEXC                    = 00000010  G
CHS$V_MBANED                    = 00000011  G
CHS$V_MBAWCKLWR                 = 00000019  G
CHS$V_MBAWCKUPR                 = 0000001A  G
CHS$V_PGMERR                    = 00000003  G
CHS$V_PGMHDE                    = 0000000B  G
CHS$V_SYSERR                    = 00000000  G
CHS$V_SYSMEM                    = 00000009  G
CHS$V_SYSSBI                    = 0000000A  G
CLEAN_UP                          00000000 R      0B
CLEAN_UP_X                        000000EB R      0B
CPU$V_COMET                     = 00000002
CPU$V_HS                        = 00000000
CPU$V_STAR                      = 00000001
DEFAULT                         = 00000000  G
DEV_REG                           00000000 R      09
DISPATCH                          00000000 R      07
DS$ABORT                          00010020  G
DS$ASKADR                         00010090  G
DS$ASKDATA                        00010080  G
DS$ASKLGCL                        00010098  G
DS$ASKSTR                         000100A0  G
DS$ASKVLD                         00010088  G
DS$ATTACH                         000101A8  G
DS$BGNSUB                         00010030  G
DS$BRANCH                         000100A8  G
DS$BREAK                          00010058  G
DS$CANWAIT                        00010070  G
DS$CHANNEL                        00010180  G
DS$CKLOOP                         00010040  G
DS$CLRVEC                         00010168  G
DS$CNTRLC                         00010078  G
DS$CVTREG                         000100B0  G
DS$ENDPASS                        00010010  G
DS$ENDSUB                         00010038  G
DS$ERRDEV                         000100C8  G
DS$ERRHARD                        000100D0  G
DS$ERRPREP                        00010108  G
DS$ERRSOFT                        000100D8  G
DS$ERRSYS                         000100C0  G
DS$ESCAPE                         00010050  G
DS$FREEDBGSYM                     000101B8  G
DS$GETBUF                         00010120  G
DS$GETMEM                         00010130  G
DS$GPHARD                         00010018  G
DS$HELP                           000101B0  G
DS$INITSCB                        00010170  G
DS$INLOOP                         00010048  G
DS$K_BBC                        = 0000001D  G
DS$K_BBCC                       = 00000021  G
DS$K_BBCCI                      = 00000023  G
DS$K_BBCS                       = 0000001F  G
DS$K_BBS                        = 0000001C  G
DS$K_BBSC                       = 00000020  G
DS$K_BBSS                       = 0000001E  G
DS$K_BBSSI                      = 00000022  G
DS$K_BCC_M                      = 0000001B  G
DS$K_BCS_M                      = 0000001A  G
DS$K_BEQLU_B                    = 00000005  G
DS$K_BEQLU_M                    = 00000011  G
DS$K_BEQL_B                     = 00000004  G
DS$K_BEQL_M                     = 00000010  G
DS$K_BERROR                     = 00000026  G
DS$K_BGEQU_B                    = 00000007  G
DS$K_BGEQU_M                    = 00000013  G
DS$K_BGEQ_B                     = 00000006  G
DS$K_BGEQ_M                     = 00000012  G
DS$K_BGTRU_B                    = 00000009  G
DS$K_BGTRU_M                    = 00000015  G
DS$K_BGTR_B                     = 00000008  G
DS$K_BGTR_M                     = 00000014  G
DS$K_BLBC                       = 00000025  G
DS$K_BLBS                       = 00000024  G
DS$K_BLEQU_B                    = 00000003  G
DS$K_BLEQU_M                    = 0000000F  G
DS$K_BLEQ_B                     = 00000002  G
DS$K_BLEQ_M                     = 0000000E  G
DS$K_BLSSU_B                    = 00000001  G
DS$K_BLSSU_M                    = 0000000D  G
DS$K_BLSS_B                     = 00000000  G
DS$K_BLSS_M                     = 0000000C  G
DS$K_BNEQU_B                    = 0000000B  G
DS$K_BNEQU_M                    = 00000017  G
DS$K_BNEQ_B                     = 0000000A  G
DS$K_BNEQ_M                     = 00000016  G
DS$K_BNERROR                    = 00000027  G
DS$K_BVC_M                      = 00000019  G
DS$K_BVS_M                      = 00000018  G
DS$K_ERROR                      = 00000002  G
DS$K_NORMAL                     = 00000001  G
DS$K_SEVERE                     = 00000004  G
DS$K_SUBSYS                     = 00000066  G
DS$K_WARNING                    = 00000000  G
DS$LOAD                           00010198  G
DS$LOADPCS                        000101C0  G
DS$MAPDBGBLOCK                    00010118  G
DS$MMOFF                          00010158  G
DS$MMON                           00010150  G
DS$MOVPHY                         00010148  G
DS$MOVVRT                         00010140  G
DS$PARSE                          000100B8  G
DS$PRINTB                         000100E0  G
DS$PRINTF                         000100F0  G
DS$PRINTS                         000100F8  G
DS$PRINTSIG                       00010100  G
DS$PRINTX                         000100E8  G
DS$PROBE                          000101A0  G
```

DS$RELBUF		00010128	G	DSA$GL_PASSES		0000FE08	
DS$RELMEM		00010138	G	DSA$GL_PASSNO		0000FE54	
DS$SETIPL		00010178	G	DSA$GL_SECTNO		0000FE10	
DS$SETMAP		00010188	G	DSA$GL_SID		0000FE14	
DS$SETPRIEXV		00010110	G	DSA$GL_SUBTNO		0000FE4C	
DS$SETVEC		00010160	G	DSA$GL_TESTNO		0000FE50	
DS$SHOCHAN		00010190	G	DSA$GL_UNITS		0000FE0C	
DS$SUMMARY		00010028	G	DSA$GQ_MSGPTR		0000FE68	
DS$WAITMS		00010060	G	DSA$GT_DEVNAM		0000FE5C	
DS$WAITUS		00010068	G	DSA$M_APT	=	80000000	
DS$_ARITH	=	006600D0	G	DSA$M_BELL	=	00000008	
DS$_BADLINK	=	006600F0	G	DSA$M_BINARY	=	00000002	
DS$_BADTYPE	=	006600E8	G	DSA$M_CRD_AUTOTEST	=	00000800	
DS$_CHME	=	006600A8	G	DSA$M_CRD_MENUTEST	=	00010000	
DS$_CHMK	=	006600E0	G	DSA$M_CRD_TRACE	=	00020000	
DS$_DEVNAME	=	00660108	G	DSA$M_DEBUG	=	04000000	
DS$_ERROR	=	00660002	G	DSA$M_HALT	=	00000001	
DS$_FHWE	=	00660068	G	DSA$M_IE1	=	00000010	
DS$_FRAGBUF	=	00660080	G	DSA$M_IE2	=	00000020	
DS$_ICBUSY	=	006600C8	G	DSA$M_IE3	=	00000040	
DS$_ICERR	=	006600C0	G	DSA$M_IES	=	00000080	
DS$_IHWE	=	00660060	G	DSA$M_LOAD_DEBUGGER	=	40000000	
DS$_ILLCHAR	=	00660018	G	DSA$M_LOOP	=	00000004	
DS$_ILLPAGCNT	=	00660078	G	DSA$M_NORPT	=	08000000	
DS$_ILLUNIT	=	00660100	G	DSA$M_OPER	=	00001000	
DS$_INSFMEM	=	00660050	G	DSA$M_PASS0	=	20000000	
DS$_IPL2HI	=	006600B8	G	DSA$M_PROMPT	=	00002000	
DS$_IVADDR	=	00660040	G	DSA$M_QA	=	00008000	
DS$_IVVECT	=	00660038	G	DSA$M_QUICK	=	00000100	
DS$_KRNLSTK	=	00660090	G	DSA$M_SEARCH	=	00004000	
DS$_LOGIC	=	00660070	G	DSA$M_TRACE	=	00000400	
DS$_MCHK	=	00660088	G	DSA$M_USER	=	10000000	
DS$_MMOFF	=	00660058	G	DSA$M_VERIFY	=	00000200	
DS$_NEEDUNIT	=	006600F8	G	DSA$V_APT	=	0000001F	
DS$_NOPCS	=	00660110	G	DSA$V_BELL	=	00000003	
DS$_NORMAL	=	00660001	G	DSA$V_BINARY	=	00000001	
DS$_NOSUPPORT	=	006600B1	G	DSA$V_CRD_AUTOTEST	=	0000000B	
DS$_NOTDON	=	00660030	G	DSA$V_CRD_MENUTEST	=	00000010	
DS$_NOTIMP	=	006600B0	G	DSA$V_CRD_TRACE	=	00000011	
DS$_NULLSTR	=	00660010	G	DSA$V_DEBUG	=	0000001A	
DS$_OVERFLOW	=	00660008	G	DSA$V_HALT	=	00000000	
DS$_POWER	=	00660098	G	DSA$V_IE1	=	00000004	
DS$_PROGERR	=	00660020	G	DSA$V_IE2	=	00000005	
DS$_SEVERE	=	00660004	G	DSA$V_IE3	=	00000006	
DS$_TRANSL	=	006600A0	G	DSA$V_IES	=	00000007	
DS$_TRUNCATE	=	00660028	G	DSA$V_LOAD_DEBUGGER	=	0000001E	
DS$_UNEXPINT	=	006600D8	G	DSA$V_LOOP	=	00000002	
DS$_VASFULL	=	00660048	G	DSA$V_NORPT	=	0000001B	
DS$_WARNING	=	00660000	G	DSA$V_OPER	=	0000000C	
DSA$AL_APTMAIL		0000FE00		DSA$V_PASS0	=	0000001D	
DSA$AT_APTTXT		0000FA00		DSA$V_PROMPT	=	0000000D	
DSA$GL_APTCOM		0000FE04		DSA$V_QA	=	0000000F	
DSA$GL_DEVLEN		0000FE58		DSA$V_QUICK	=	00000008	
DSA$GL_ERRNO		0000FE44		DSA$V_SEARCH	=	0000000E	
DSA$GL_EVENT		0000FE48		DSA$V_TRACE	=	0000000A	
DSA$GL_FLAGS		0000FE00		DSA$V_USER	=	0000001C	
DSA$GL_MSGTYP		0000FE40		DSA$V_VERIFY	=	00000009	

```
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
FMT_SUMMARY                       000000BE R      09
GPR_TABLE                         00000095 RG     02
HALT                            = 00000004  G
HARDCNT                           00000000 RG     09
HP$A_DEPENDENT                    00000032
HP$A_DEVICE                       00000018
HP$A_DVA                          0000001C
HP$A_LINK                         00000020
HP$B_DRIVE                        0000000B
HP$B_FLAGS                        0000000A
HP$KA_L_FLAGS                   = 0000000C
HP$KA_MEM_SIZE                  = 00000044
HP$M_ALLOC                      = 00000001
HP$M_WASALL                     = 00000002
HP$Q_DEVICE                       00000000
HP$T_DEVICE                       0000000C
HP$T_TYPE                         00000026
HP$V_ALLOC                      = 00000000
HP$V_WASALL                     = 00000001
HP$W_SIZE                         00000008
HP$W_VECTOR                       00000024
INITIALIZE                        00000000 R      0A
INITIALIZE_X                      00000262 R      0A
IPR                             = 00000006  G
KA$B_ACC_TYPE                     00000042
KA$B_SID_TYPE                     0000003D
KA$K_LEN                          00000043
KA$L_FLAGS                        00000032
KA$L_SID                          0000003A
KA$M_CHAR                       = 00000100
KA$M_CRC                        = 00000010
KA$M_D_FLOAT                    = 00000002
KA$M_EDIT                       = 00000080
KA$M_F_FLOAT                    = 00000001
KA$M_G_FLOAT                    = 00000004
KA$M_H_FLOAT                    = 00000008
KA$M_PACKED                     = 00000020
KA$M_PACK_MUL                   = 00000040
KA$M_TODR                       = 00010000
KA$V_CHAR                       = 00000008
KA$V_CRC                        = 00000004
KA$V_D_FLOAT                    = 00000001
KA$V_EDIT                       = 00000007
KA$V_F_FLOAT                    = 00000000
KA$V_G_FLOAT                    = 00000002
KA$V_H_FLOAT                    = 00000003
KA$V_PACKED                     = 00000005
KA$V_PACK_MUL                   = 00000006
KA$V_TODR                       = 00000010
KA$W_LATENCY                      0000003E
KA$W_PROGRESS                     00000040
KA$W_RESERVED                     00000038
KA$W_WCS                          00000036
KA730                           = 00000001
KA730_REV_54                    = 03003600  G
KA_PTABLE                         0000018B RG     02
KA_UNIT_NO                        00000071 RG     02
L$A_CCP                           00000040 R      04
L$A_DEVP                          0000001C R      04
L$A_DREG                          00000024 R      04
L$A_DTP                           00000018 R      04
L$A_ICP                           0000003C R      04
L$A_LASTAD                        00000014 R      04
L$A_NAME                          00000008 R      04
L$A_REPP                          00000044 R      04
L$A_SECNAM                        00000050 R      04
L$A_STATAB                        00000048 R      04
L$A_TSTCNT                        00000054 R      04
L$L_ENVIRON                       00000004 R      04
L$L_ERRTYP                        0000004C R      04
L$L_HEADLENGTH                    00000000 R      04
L$L_REV                           0000000C R      04
L$L_UNIT                          00000020 R      04
L$L_UPDATE                        00000010 R      04
LASTAD                            00000000 R      05
LDPCTX                          = 0000000B  G
L_ACT_ISP                         00000058 RG     02
L_ACT_KSP                         00000054 RG     02
L_ACT_PSL                         0000005C RG     02
L_ACT_SP                          00000050 RG     02
L_EXP_AP                          000000C5 RG     02
L_EXP_FP                          000000C9 RG     02
L_EXP_ISP                         000000D5 RG     02
L_EXP_KSP                         000000D1 RG     02
L_EXP_PSL                         000000D9 RG     02
L_EXP_SP                          000000CD RG     02
L_ISP                             00000064 RG     02
L_KA_FLAGS                        0000008D RG     02
L_KSIZE_PTBL                      00000091 RG     02
L_KSP                             00000060 RG     02
L_LAST_UBE                        00000068 RG     02
L_TSTCNT                          00000000 R      06
MANUAL                          = 00000001  G
MCHK                            = 00000005  G
MEM                             = 00000009  G
MM_MSG                            000000EE R      09
MODELIST                          00000156 RG     02
M_DSABL_ECC                     = 02000000
PAR$M_ATDEF                     = 00000010
PAR$M_ATHI                      = 00000008
```

```
PAR$M_ATLO                      = 00000004
PAR$M_NODEF                     = 00000002
PAR$V_ATDEF                     = 00000004
PAR$V_ATHI                      = 00000003
PAR$V_ATLO                      = 00000002
PAR$V_NODEF                     = 00000001
PAR$_BIN                        = 00000002
PAR$_DEC                        = 0000000A
PAR$_HEX                        = 00000010
PAR$_NO                         = 00000000
PAR$_OCT                        = 00000008
PAR$_YES                        = 00000001
PART_INIT                         0000188B RG    09
POWER                           = 00000003  G
PR$S_SID_ECO                    = 00000009  G
PR$S_SID_PL                     = 00000003  G
PR$S_SID_SN                     = 0000000C  G
PR$S_SID_TYPE                   = 00000008  G
PR$V_SID_ECO                    = 0000000F  G
PR$V_SID_PL                     = 0000000C  G
PR$V_SID_SN                     = 00000000  G
PR$V_SID_TYPE                   = 00000018  G
PR$_ACCR                        = 00000029  G
PR$_ACCS                        = 00000028  G
PR$_ASTLVL                      = 00000013  G
PR$_CADR                        = 00000025  G
PR$_CAER                        = 00000027  G
PR$_CMIERR                      = 00000017  G
PR$_CRBT                        = 00000043  G
PR$_CSRD                        = 0000001D  G
PR$_CSRS                        = 0000001C  G
PR$_CSTD                        = 0000001F  G
PR$_CSTS                        = 0000001E  G
PR$_CSWP                        = 00000042  G
PR$_ESP                         = 00000001  G
PR$_ICCS                        = 00000018  G
PR$_ICR                         = 0000001A  G
PR$_IPL                         = 00000012  G
PR$_ISP                         = 00000004  G
PR$_KSP                         = 00000000  G
PR$_MAPEN                       = 00000038  G
PR$_MCESR                       = 00000026  G
PR$_MCTL1                       = 00000044  G
PR$_MCTL2                       = 00000045  G
PR$_MDCR1                       = 00000049  G
PR$_MEAR                        = 00000048  G
PR$_MECCR                       = 0000004B  G
PR$_MEDR                        = 0000004A  G
PR$_MGEN                        = 0C000046  G
PR$_MTBER                       = 00000047  G
PR$_NICR                        = 00000019  G
PR$_P0BR                        = 00000008  G
PR$_P0LR                        = 00000009  G
PR$_P1BR                        = 0000000A  G
PR$_P1LR                        = 0000000B  G
PR$_PAMACC                      = 00000040  G
PR$_PAMLOC                      = 00000041  G
PR$_PCBB                        = 00000010  G
PR$_PME                         = 0000003D  G
PR$_RXCS                        = 00000020  G
PR$_RXDB                        = 00000021  G
PR$_SBIER                       = 00000034  G
PR$_SBIFS                       = 00000030  G
PR$_SBIMT                       = 00000033  G
PR$_SBIQC                       = 00000036  G
PR$_SBIS                        = 00000031  G
PR$_SBISC                       = 00000032  G
PR$_SBITA                       = 00000035  G
PR$_SBR                         = 0000000C  G
PR$_SCBB                        = 00000011  G
PR$_SID                         = 0000003E  G
PR$_SID_TYP730                  = 00000003  G
PR$_SID_TYP750                  = 00000002  G
PR$_SID_TYP780                  = 00000001  G
PR$_SID_TYP7VV                  = 00000004  G
PR$_SID_TYPMAX                  = 00000004  G
PR$_SIRR                        = 00000014  G
PR$_SISR                        = 00000015  G
PR$_SLR                         = 0000000D  G
PR$_SSP                         = 00000002  G
PR$_TBCHK                       = 0000003F  G
PR$_TBDR                        = 00000024  G
PR$_TBIA                        = 00000039  G
PR$_TBIS                        = 0000003A  G
PR$_TODR                        = 0000001B  G
PR$_TXCS                        = 00000022  G
PR$_TXDB                        = 00000023  G
PR$_UBRESET                     = 00000037  G
PR$_USP                         = 00000003  G
PR$_WCSA                        = 0000002C  G
PR$_WCSD                        = 0000002D  G
PSL$C_EXEC                      = 00000001  G
PSL$C_KERNEL                    = 00000000  G
PSL$C_SUPER                     = 00000002  G
PSL$C_USER                      = 00000003  G
PSL$K_EXEC                      = 00000001  G
PSL$K_KERNEL                    = 00000000  G
PSL$K_SUPER                     = 00000002  G
PSL$K_USER                      = 00000003  G
PSL$M_C                         = 00000001  G
PSL$M_CBIT                      = 00000001  G
PSL$M_CM                        = 80000000  G
PSL$M_CURMOD                    = 03000000  G
PSL$M_DV                        = 00000080  G
PSL$M_FPD                       = 08000000  G
PSL$M_FU                        = 00000040  G
PSL$M_IPL                       = 001F0000  G
PSL$M_IS                        = 04000000  G
PSL$M_IV                        = 00000020  G
PSL$M_N                         = 00000008  G
PSL$M_NBIT                      = 00000008  G
PSL$M_PRVMOD                    = 00C00000  G
PSL$M_SAFBITS                   = 000037FF  G
PSL$M_TBIT                      = 00000010  G
```

```
PSL$M_TP                        = 40000000  G
PSL$M_V                         = 00000002  G
PSL$M_VBIT                      = 00000002  G
PSL$M_Z                         = 00000004  G
PSL$M_ZBIT                      = 00000004  G
PSL$S_CURMOD                    = 00000002  G
PSL$S_IPL                       = 00000005  G
PSL$S_PRVMOD                    = 00000002  G
PSL$V_C                         = 00000000  G
PSL$V_CBIT                      = 00000000  G
PSL$V_CM                        = 0000001F  G
PSL$V_CURMOD                    = 00000018  G
PSL$V_DV                        = 00000007  G
PSL$V_FPD                       = 0000001B  G
PSL$V_FU                        = 00000006  G
PSL$V_IPL                       = 00000010  G
PSL$V_IS                        = 0000001A  G
PSL$V_IV                        = 00000005  G
PSL$V_N                         = 00000003  G
PSL$V_NBIT                      = 00000003  G
PSL$V_PRVMOD                    = 00000016  G
PSL$V_TBIT                      = 00000004  G
PSL$V_TP                        = 0000001E  G
PSL$V_V                         = 00000001  G
PSL$V_VBIT                      = 00000001  G
PSL$V_Z                         = 00000002  G
PSL$V_ZBIT                      = 00000002  G
QWD_TIME                          000000AB R      09
Q_TEST_TIME                       00000004 R      09
REG_TABLE                         000000DD RG     02
SCB$L_ACCESS                      00000020  G
SCB$L_ARITH                       00000034  G
SCB$L_BREAK                       0000002C  G
SCB$L_CHME                        00000044  G
SCB$L_CHMK                        00000040  G
SCB$L_CHMS                        00000048  G
SCB$L_CHMU                        0000004C  G
SCB$L_COMPAT                      00000030  G
SCB$L_KNLSTK                      00000008  G
SCB$L_MACHCHK                     00000004  G
SCB$L_OPCCUS                      00000014  G
SCB$L_OPCDEC                      00000010  G
SCB$L_POWER                       0000000C  G
SCB$L_RADRMOD                     0000001C  G
SCB$L_ROPRAND                     00000018  G
SCB$L_RXDB                        000000F8  G
SCB$L_SFTLVL1                     00000084  G
SCB$L_SFTLVL10                    000000A8  G
SCB$L_SFTLVL11                    000000AC  G
SCB$L_SFTLVL12                    000000B0  G
SCB$L_SFTLVL13                    000000B4  G
SCB$L_SFTLVL14                    000000B8  G
SCB$L_SFTLVL15                    000000BC  G
SCB$L_SFTLVL2                     00000088  G
SCB$L_SFTLVL3                     0000008C  G
SCB$L_SFTLVL4                     00000090  G
SCB$L_SFTLVL5                     00000094  G
SCB$L_SFTLVL6                     00000098  G
SCB$L_SFTLVL7                     0000009C  G
SCB$L_SFTLVL8                     000000A0  G
SCB$L_SFTLVL9                     000000A4  G
SCB$L_TBIT                        00000028  G
SCB$L_TIMER                       000000C0  G
SCB$L_TRANSL                      00000024  G
SCB$L_TXDB                        000000FC  G
SCB$L_UBE_0                     = 00000348
SCB$L_UBE_1                     = 0000034C
SCB$L_UBE_2                     = 00000350
SCB$L_UBE_3                     = 00000354
SCB$L_ZERO                        00000000  G
SECTION                           00000057 R      09
SEP_FUNCTIONAL                  = 00000002
SEP_REPAIR                      = 00000003
SIZ...                          = 00000001
SUMMARY                           00000000 R      0C
SUMMARY_INIT                      0000012B R      09
SUMMARY_X                         00000046 R      0C
SUM_A_BLOCK                       0000000C R      09
SUM_BLK_SIZE                      00000014 R      09
SUM_K_SIZE                        0000000C  G
SUM_L_HARDCNT                     00000000  G
SUM_Q_RUNTIME                     00000004  G
SYS$ALLOC                         00010238  G
SYS$ASCTIM                        00010248  G
SYS$ASSIGN                        00010250  G
SYS$BINTIM                        00010258  G
SYS$CANCEL                        00010260  G
SYS$CANTIM                        00010268  G
SYS$CLOSE                         000105B8  G
SYS$CLREF                         00010298  G
SYS$CONNECT                       000105C0  G
SYS$DALLOC                        000102D8  G
SYS$DASSGN                        000102E0  G
SYS$DISCONNECT                    000105D0  G
SYS$FAO                           00010350  G
SYS$FAOL                          00010358  G
SYS$GET                           00010580  G
SYS$GETCHN                        000104C8  G
SYS$GETTIM                        00010378  G
SYS$LKWSET                        000103A0  G
SYS$NUMTIM                        000103B8  G
SYS$OPEN                          00010608  G
SYS$QIO                           000103C8  G
SYS$QIOW                          00010200  G
SYS$READ                          00010590  G
SYS$READEF                        000103D0  G
SYS$SETAST                        000103F8  G
SYS$SETEF                         00010400  G
SYS$SETIMR                        00010420  G
SYS$SETPRI                        00010428  G
SYS$SETPRT                        00010430  G
SYS$SETRWM                        00010438  G
SYS$SETSWM                        00010448  G
SYS$ULKPAG                        00010460  G
```

```
SYS$ULWSET                      00010468  G
SYS$UNWIND                      00010470  G
SYS$WAITFR                      00010478  G
SYS$WFLAND                      00010488  G
SYS$WFLOR                       00010490  G
S_ALL                           00000044 R      09
S_DEFAULT                       00000018 R      09
S_HALT                          00000032 R      09
S_IPR                           0000003C R      09
S_LDPCTX                        00000050 R      09
S_MANUAL                        00000020 R      09
S_MCHK                          00000037 R      09
S_MEM                           00000048 R      09
S_POWER                         0000002C R      09
S_TU58                          00000027 R      09
S_UBI                           0000004C R      09
S_WCS                           00000040 R      09
TU0_UNIT_NO                     00000075 RG     02
TU1_UNIT_NO                     00000079 RG     02
TU58                          = 00000002  G
T_KA730                         0000008B R      09
T_NAME                          00000058 R      04
T_TIME                          000000B3 R      09
T_TU58                          00000091 R      09
T_UBE                           00000096 R      09
UBE                           = 00000003
UBI                           = 0000000A  G
WCS                           = 00000007  G
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAX1	0000018F (399.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$HEADER	00000092 (146.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
_LAST	00000000 (0.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
$TSTCNT	00000000 (0.)	06 (6.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
DISPATCH	00000000 (0.)	07 (7.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
DISPATCH_X	00000018 (24.)	08 (8.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
DATA	000001C5 (453.)	09 (9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
INITIALIZE	0000026A (618.)	0A (10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
CLEANUP	000000F3 (243.)	0B (11.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
SUMMARY	0000004E (78.)	0C (12.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$E	=00000001	495 (8)	458 (7) 520 (8)
$$M	=00000004	636 (12)	636 (12)
$$N	=00000000	636 (12)	303 (5) 311 (5) #-375 (7) 588 (10) 636 (12)
$$S	=FFFFFFFF	66 (2)	
$ENV	=00000001	66 (2)	212 (3)
$ER	=00000002	636 (12)	367 (7) #-387 (7) 495 (8) #-635 (12) #-636 (12)
$MO	=00000001	704 (14)	704 (14)
$ST	=00000000	520 (8)	458 (7) 520 (8) 636 (12)
$TN	=00000000	66 (2)	367 (7) 495 (8) 575 (10) 704 (14)
AEM	00000171-R	191 (2)	186 (2)
AKM	0000016A-R	190 (2)	185 (2)
ALL	=00000008	303 (5)	
AL_DEVTYP	0000009B-R	311 (5)	212 (3)
ASM	0000017B-R	192 (2)	187 (2)
AUM	00000186-R	193 (2)	188 (2)
A_BUFFER_BEGIN	00000000-R	106 (2)	446 (7)
A_BUFFER_END	00000004-R	107 (2)	
A_FREE_MEM	00000048-R	116 (2)	434 (7) #-437 (7) #-440 (7)
A_HALT_BUFFER	00000010-R	109 (2)	448 (7)
A_HEADEND	00000058-R	212 (3)	212 (3)
A_MCHK_BUFFER	00000018-R	110 (2)	449 (7)
A_MEM_CSR_1	=00F20004	101 (2)	#-519 (8)
A_PSL_MNE	00000115-R	180 (2)	
A_PWR_BUFFER	00000008-R	108 (2)	447 (7)
A_P_TABLE	0000006D-R	127 (2)	386 (7) #-389 (7) #-411 (7) #-419 (7)
A_UBE_BUFFER_0	00000028-R	112 (2)	451 (7)
A_UBE_BUFFER_1	00000030-R	113 (2)	452 (7)
A_UBE_BUFFER_2	00000038-R	114 (2)	453 (7)
A_UBE_BUFFER_3	00000040-R	115 (2)	454 (7)
A_WCS_BUFFER	00000020-R	111 (2)	450 (7)
BE0_UNIT_NO	0000007D-R	131 (2)	#-378 (7) #-398 (7)
BE1_UNIT_NO	00000081-R	132 (2)	#-379 (7) #-400 (7)
BE2_UNIT_NO	00000085-R	133 (2)	#-380 (7) #-402 (7)
BE3_UNIT_NO	00000089-R	134 (2)	#-381 (7) #-404 (7)
BIT...	=00000028	88 (2)	66 (2) 78 (2) 80 (2) 81 (2) 82 (2) 84 (2) 86 (2) 87 (2) 88 (2)
BUF_SIZE	=0000000B	316 (5)	318 (5) 321 (5)
B_TU58_DRIVE	0000006C-R	126 (2)	#-384 (7) #-423 (7) #-427 (7)
CEP_FUNCTIONAL	=00000000	66 (2)	
CEP_REPAIR	=00000001	66 (2)	66 (2)
CHC$_ABORT	=00000002	84 (2)	
CHC$_CLEAR	=00000006	84 (2)	
CHC$_CLRDFT	=00000009	84 (2)	
CHC$_DSINT	=00000005	84 (2)	
CHC$_ENINT	=00000004	84 (2)	
CHC$_INITA	=00000000	84 (2)	
CHC$_INITB	=00000001	84 (2)	

```
CHC$_PURGE                      =00000003      84      (2)
CHC$_SETDFT                     =00000008      84      (2)
CHC$_STATUS                     =00000007      84      (2)
CHF$L_MCHARGLST                  00000008      85      (2)
CHF$L_MCH_ARGS                   00000000      85      (2)
CHF$L_MCH_DEPTH                  00000008      85      (2)
CHF$L_MCH_FRAME                  00000004      85      (2)
CHF$L_MCH_SAVR0                  0000000C      85      (2)
CHF$L_MCH_SAVR1                  00000010      85      (2)
CHF$L_SIGARGLST                  00000004      85      (2)
CHF$L_SIG_ARG1                   00000008      85      (2)
CHF$L_SIG_ARGS                   00000000      85      (2)
CHF$L_SIG_NAME                   00000004      85      (2)
CHI$M_CHNINT                    =00000001      84      (2)
CHI$M_DEVINT                    =00000002      84      (2)
CHI$M_IPL                       =0000007C      84      (2)
CHI$S_IPL                       =00000005      84      (2)
CHI$V_CHNINT                    =00000000      84      (2)
CHI$V_DEVINT                    =00000001      84      (2)
CHI$V_IPL                       =00000002      84      (2)
CHM$_FORWARD                    =00000000      84      (2)     84      (2)
CHM$_INVALIDATE                 =00000001      84      (2)     84      (2)
CHM$_MAP                        =00000002      84      (2)     84      (2)
CHM$_MFWDN                      =00000002      84      (2)
CHM$_MFWDNO                     =0000000A      84      (2)
CHM$_MFWDV                      =00000003      84      (2)
CHM$_MFWDVO                     =0000000B      84      (2)
CHM$_MREVN                      =00000006      84      (2)
CHM$_MREVNO                     =0000000E      84      (2)
CHM$_MREVV                      =00000007      84      (2)
CHM$_MREVVO                     =0000000F      84      (2)
CHM$_NFWDN                      =00000000      84      (2)
CHM$_NREVN                      =00000004      84      (2)
CHM$_OFFSET                     =00000008      84      (2)     84      (2)
CHM$_REVERSE                    =00000004      84      (2)     84      (2)
CHS$M_BUSIC                     =00800000      84      (2)
CHS$M_BUSINIT                   =00400000      84      (2)
CHS$M_BUSNXM                    =08000000      84      (2)
CHS$M_BUSPDN                    =01000000      84      (2)
CHS$M_CHNDPE                    =00000040      84      (2)
CHS$M_CHNERR                    =00000002      84      (2)     84      (2)
CHS$M_CHNMPE                    =00000080      84      (2)
CHS$M_CHPFOT                    =00000100      84      (2)
CHS$M_DEVBUS                    =00000010      84      (2)
CHS$M_DEVERR                    =00000004      84      (2)     84      (2)
CHS$M_DEVTO                     =00000020      84      (2)
CHS$M_ERRANY                    =0000000F      84      (2)
CHS$M_MBAATN                    =00100000      84      (2)
CHS$M_MBACPE                    =00200000      84      (2)
CHS$M_MBADTB                    =00040000      84      (2)
CHS$M_MBADTC                    =00080000      84      (2)
CHS$M_MBAEXC                    =00010000      84      (2)
CHS$M_MBANED                    =00020000      84      (2)
CHS$M_MBAWCKLWR                 =02000000      84      (2)
CHS$M_MBAWCKUPR                 =04000000      84      (2)
CHS$M_PGMERR                    =00000008      84      (2)     84      (2)
CHS$M_PGMHDE                    -00000800      84      (2)
```

```
CHS$M_SYSERR                    =00000001     84      (2)       84      (2)
CHS$M_SYSMEM                    =00000200     84      (2)
CHS$M_SYSSBI                    =00000400     84      (2)
CHS$V_BUSIC                     =00000017     84      (2)
CHS$V_BUSINIT                   =00000016     84      (2)
CHS$V_BUSNXM                    =0000001B     84      (2)
CHS$V_BUSPDN                    =00000018     84      (2)
CHS$V_CHNDPE                    =00000006     84      (2)
CHS$V_CHNERR                    =00000001     84      (2)
CHS$V_CHNMPE                    =00000007     84      (2)
CHS$V_CHPFOT                    =00000008     84      (2)
CHS$V_DEVBUS                    =00000004     84      (2)
CHS$V_DEVERR                    =00000002     84      (2)
CHS$V_DEVTO                     =00000005     84      (2)
CHS$V_MBAATN                    =00000014     84      (2)
CHS$V_MBACPE                    =00000015     84      (2)
CHS$V_MBADTB                    =00000012     84      (2)
CHS$V_MBADTC                    =00000013     84      (2)
CHS$V_MBAEXC                    =00000010     84      (2)
CHS$V_MBANED                    =00000011     84      (2)
CHS$V_MBAWCKLWR                 =00000019     84      (2)
CHS$V_MBAWCKUPR                 =0000001A     84      (2)
CHS$V_PGMERR                    =00000003     84      (2)
CHS$V_PGMHDE                    =0000000B     84      (2)
CHS$V_SYSERR                    =00000000     84      (2)
CHS$V_SYSMEM                    =00000009     84      (2)
CHS$V_SYSSBI                    =0000000A     84      (2)
CLEAN_UP                         00000000-R   495     (8)       212     (3)
CLEAN_UP_X                       000000EB-R   520     (8)
CPU$V_COMET                     =00000002     80      (2)
CPU$V_HS                        =00000000     80      (2)
CPU$V_STAR                      =00000001     80      (2)
DEFAULT                         =00000000     303     (5)
DEV_REG                          00000000-R   250     (4)       212     (3)
DISPATCH                         00000000-R   225     (3)       212     (3)
DS$ABORT                         00010020     79      (2)       376     (7)       511     (8)
DS$ASKADR                        00010090     79      (2)
DS$ASKDATA                       00010080     79      (2)
DS$ASKLGCL                       00010098     79      (2)
DS$ASKSTR                        000100A0     79      (2)
DS$ASKVLD                        00010088     79      (2)
DS$ATTACH                        000101A8     79      (2)
DS$BGNSUB                        00010030     79      (2)
DS$BRANCH                        000100A8     79      (2)       387     (7)       635     (12)
DS$BREAK                         00010058     79      (2)       458     (7)       520     (8)       593     (10)
DS$CANWAIT                       00010070     79      (2)
DS$CHANNEL                       00010180     79      (2)
DS$CKLOOP                        00010040     79      (2)
DS$CLRVEC                        00010168     79      (2)       496     (8)       497     (8)       498     (8)       499     (8)
                                                                 500     (8)       501     (8)       502     (8)       503     (8)
                                                                 504     (8)       505     (8)       506     (8)       507     (8)
                                                                 508     (8)
DS$CNTRLC                        00010078     79      (2)
DS$CVTREG                        000100B0     79      (2)
DS$ENDPASS                       00010010     79      (2)       457     (7)
DS$ENDSUB                        00010038     79      (2)
DS$ERRDEV                        000100C8     79      (2)
```

```
DS$ERRHARD                      000100D0      79      (2)
DS$ERRPREP                      00010108      79      (2)
DS$ERRSOFT                      000100D8      79      (2)
DS$ERRSYS                       000100C0      79      (2)       636     (12)
DS$ESCAPE                       00010050      79      (2)
DS$FREEDBGSYM                   000101B8      79      (2)
DS$GETBUF                       00010120      79      (2)       434     (7)       446     (7)       447     (7)       448     (7)
                                                                449     (7)       450     (7)       451     (7)       452     (7)
                                                                453     (7)       454     (7)       634     (12)
DS$GETMEM                       00010130      79      (2)
DS$GPHARD                       00010018      79      (2)       386     (7)
DS$HELP                         000101B0      79      (2)
DS$INITSCB                      00010170      79      (2)       512     (8)
DS$INLOOP                       00010048      79      (2)
DS$K_BBC                      =0000001D       88      (2)
DS$K_BBCC                     =00000021       88      (2)
DS$K_BBCCI                    =00000023       88      (2)
DS$K_BBCS                     =0000001F       88      (2)
DS$K_BBS                      =0000001C       88      (2)
DS$K_BBSC                     =00000020       88      (2)
DS$K_BBSS                     =0000001E       88      (2)
DS$K_BBSSI                    =00000022       88      (2)
DS$K_BCC_M                    =0000001B       88      (2)
DS$K_BCS_M                    =0000001A       88      (2)
DS$K_BEQLU_B                  =00000005       88      (2)
DS$K_BEQLU_M                  =00000011       88      (2)
DS$K_BEQL_B                   =00000004       88      (2)
DS$K_BEQL_M                   =00000010       88      (2)
DS$K_BERROR                   =00000026       88      (2)
DS$K_BGEQU_B                  =00000007       88      (2)
DS$K_BGEQU_M                  =00000013       88      (2)
DS$K_BGEQ_B                   =00000006       88      (2)
DS$K_BGEQ_M                   =00000012       88      (2)
DS$K_BGTRU_B                  =00000009       88      (2)
DS$K_BGTRU_M                  =00000015       88      (2)
DS$K_BGTR_B                   =00000008       88      (2)
DS$K_BGTR_M                   =00000014       88      (2)
DS$K_BLBC                     =00000025       88      (2)
DS$K_BLBS                     =00000024       88      (2)
DS$K_BLEQU_B                  =00000003       88      (2)
DS$K_BLEQU_M                  =0000000F       88      (2)
DS$K_BLEQ_B                   =00000002       88      (2)
DS$K_BLEQ_M                   =0000000E       88      (2)
DS$K_BLSSU_B                  =00000001       88      (2)
DS$K_BLSSU_M                  =0000000D       88      (2)
DS$K_BLSS_B                   =00000000       88      (2)
DS$K_BLSS_M                   =0000000C       88      (2)
DS$K_BNEQU_B                  =0000000B       88      (2)
DS$K_BNEQU_M                  =00000017       88      (2)
DS$K_BNEQ_B                   =0000000A       88      (2)
DS$K_BNEQ_M                   =00000016       88      (2)
DS$K_BNERROR                  =00000027       88      (2)     #-387     (7)     #-635     (12)
DS$K_BVC_M                    =00000019       88      (2)
DS$K_BVS_M                    =00000018       88      (2)
DS$K_ERROR                    =00000002       78      (2)
DS$K_NORMAL                   =00000001       78      (2)
DS$K_SEVERE                   =00000004       78      (2)
```

```
DS$K_SUBSYS                     =00000066     78     (2)       78     (2)
DS$K_WARNING                    =00000000     78     (2)
DS$LOAD                          00010198     79     (2)
DS$LOADPCS                       000101C0     79     (2)
DS$MAPDBGBLOCK                   00010118     79     (2)
DS$MMOFF                         00010158     79     (2)      373     (7)      509     (8)
DS$MMON                          00010150     79     (2)
DS$MOVPHY                        00010148     79     (2)
DS$MOVVRT                        00010140     79     (2)
DS$PARSE                         000100B8     79     (2)
DS$PRINTB                        000100E0     79     (2)
DS$PRINTF                        000100F0     79     (2)      375     (7)
DS$PRINTS                        000100F8     79     (2)      588     (10)
DS$PRINTSIG                      00010100     79     (2)
DS$PRINTX                        000100E8     79     (2)
DS$PROBE                         000101A0     79     (2)
DS$RELBUF                        00010128     79     (2)      442     (7)      514     (8)      515     (8)
DS$RELMEM                        00010138     79     (2)
DS$SETIPL                        00010178     79     (2)      518     (8)
DS$SETMAP                        00010188     79     (2)
DS$SETPRIEXV                     00010110     79     (2)
DS$SETVEC                        00010160     79     (2)
DS$SHOCHAN                       00010190     79     (2)
DS$SUMMARY                       00010028     79     (2)
DS$WAITMS                        00010060     79     (2)
DS$WAITUS                        00010068     79     (2)
DS$_ARITH                       =006600D0     78     (2)
DS$_BADLINK                     =006600F0     78     (2)
DS$_BADTYPE                     =006600E8     78     (2)
DS$_CHME                        =006600A8     78     (2)
DS$_CHMK                        =006600E0     78     (2)
DS$_DEVNAME                     =00660108     78     (2)
DS$_ERROR                       =00660002     78     (2)
DS$_FHWE                        =00660068     78     (2)
DS$_FRAGBUF                     =00660080     78     (2)
DS$_ICBUSY                      =006600C8     78     (2)
DS$_ICERR                       =006600C0     78     (2)
DS$_IHWE                        =00660060     78     (2)
DS$_ILLCHAR                     =00660018     78     (2)
DS$_ILLPAGCNT                   =00660078     78     (2)
DS$_ILLUNIT                     =00660100     78     (2)
DS$_INSFMEM                     =00660050     78     (2)
DS$_IPL2HI                      =006600B8     78     (2)
DS$_IVADDR                      =00660040     78     (2)
DS$_IVVECT                      =00660038     78     (2)
DS$_KRNLSTK                     =00660090     78     (2)
DS$_LOGIC                       =00660070     78     (2)
DS$_MCHK                        =00660088     78     (2)
DS$_MMOFF                       =00660058     78     (2)
DS$_NEEDUNIT                    =006600F8     78     (2)
DS$_NOPCS                       =00660110     78     (2)
DS$_NORMAL                      =00660001     78     (2)
DS$_NOSUPPORT                   =006600B1     78     (2)
DS$_NOTDON                      =00660030     78     (2)
DS$_NOTIMP                      =006600B0     78     (2)       78     (2)
DS$_NULLSTR                     =00660010     78     (2)
DS$_OVERFLOW                    =00660008     78     (2)
```

```
DS$_POWER                       =00660098      78      (2)
DS$_PROGERR                     =00660020      78      (2)
DS$_SEVERE                      =00660004      78      (2)
DS$_TRANSL                      =006600A0      78      (2)
DS$_TRUNCATE                    =00660028      78      (2)
DS$_UNEXPINT                    =006600D8      78      (2)
DS$_VASFULL                     =00660048      78      (2)
DS$_WARNING                     =00660000      78      (2)       78      (2)
DSA$AL_APTMAIL                   0000FE00      80      (2)
DSA$AT_APTTXT                    0000FA00      80      (2)
DSA$GL_APTCOM                    0000FE04      80      (2)
DSA$GL_DEVLEN                    0000FE58      80      (2)
DSA$GL_ERRNO                     0000FE44      80      (2)
DSA$GL_EVENT                     0000FE48      80      (2)
DSA$GL_FLAGS                     0000FE00      80      (2)       370     (7)
DSA$GL_MSGTYP                    0000FE40      80      (2)
DSA$GL_PASSES                    0000FE08      80      (2)
DSA$GL_PASSNO                    0000FE54      80      (2)
DSA$GL_SECTNO                    0000FE10      80      (2)
DSA$GL_SID                       0000FE14      80      (2)
DSA$GL_SUBTNO                    0000FE4C      80      (2)
DSA$GL_TESTNO                    0000FE50      80      (2)     #-684     (14)
DSA$GL_UNITS                     0000FE0C      80      (2)
DSA$GQ_MSGPTR                    0000FE68      80      (2)
DSA$GT_DEVNAM                    0000FE5C      80      (2)
DSA$M_APT                       =80000000      80      (2)
DSA$M_BELL                      =00000008      80      (2)
DSA$M_BINARY                    =00000002      80      (2)
DSA$M_CRD_AUTOTEST              =00000800      80      (2)
DSA$M_CRD_MENUTEST              =00010000      80      (2)
DSA$M_CRD_TRACE                 =00020000      80      (2)
DSA$M_DEBUG                     =04000000      80      (2)
DSA$M_HALT                      =00000001      80      (2)
DSA$M_IE1                       =00000010      80      (2)
DSA$M_IE2                       =00000020      80      (2)
DSA$M_IE3                       =00000040      80      (2)
DSA$M_IES                       =00000080      80      (2)
DSA$M_LOAD_DEBUGGER             =40000000      80      (2)
DSA$M_LOOP                      =00000004      80      (2)
DSA$M_NORPT                     =08000000      80      (2)
DSA$M_OPER                      =00001000      80      (2)
DSA$M_PASSO                     =20000000      80      (2)
DSA$M_PROMPT                    =00002000      80      (2)
DSA$M_QA                        =00008000      80      (2)
DSA$M_QUICK                     =00000100      80      (2)
DSA$M_SEARCH                    =00004000      80      (2)
DSA$M_TRACE                     =00000400      80      (2)
DSA$M_USER                      =10000000      80      (2)
DSA$M_VERIFY                    =00000200      80      (2)
DSA$V_APT                       =0000001F      80      (2)
DSA$V_BELL                      =00000003      80      (2)
DSA$V_BINARY                    =00000001      80      (2)
DSA$V_CRD_AUTOTEST              =0000000B      80      (2)
DSA$V_CRD_MENUTEST              =00000010      80      (2)
DSA$V_CRD_TRACE                 =00000011      80      (2)
DSA$V_DEBUG                     =0000001A      80      (2)
DSA$V_HALT                      =00000000      80      (2)
```

```
DSA$V_IE1                       =00000004     80      (2)
DSA$V_IE2                       =00000005     80      (2)
DSA$V_IE3                       =00000006     80      (2)
DSA$V_IES                       =00000007     80      (2)
DSA$V_LOAD_DEBUGGER             =0000001E     80      (2)
DSA$V_LOOP                      =00000002     80      (2)
DSA$V_NORPT                     =0000001B     80      (2)
DSA$V_OPER                      =0000000C     80      (2)
DSA$V_PASS0                     =0000001D     80      (2)     #-370     (7)
DSA$V_PROMPT                    =0000000D     80      (2)
DSA$V_QA                        =0000000F     80      (2)
DSA$V_QUICK                     =00000008     80      (2)
DSA$V_SEARCH                    =0000000E     80      (2)
DSA$V_TRACE                     =0000000A     80      (2)
DSA$V_USER                      =0000001C     80      (2)
DSA$V_VERIFY                    =00000009     80      (2)
ENV$M_DOMAIN                    =00000002     66      (2)
ENV$M_LEVEL                     =00000001     66      (2)
ENV$M_SUPER                     =000003FC     66      (2)
ENV$S_DOMAIN                    =00000001     66      (2)
ENV$S_LEVEL                     =00000001     66      (2)
ENV$S_SUPER                     =00000008     66      (2)
ENV$V_DOMAIN                    =00000001     66      (2)       66      (2)
ENV$V_LEVEL                     =00000000     66      (2)       66      (2)
ENV$V_SUPER                     =00000002     66      (2)       212     (3)
ENV$_CPU                        =00000000     66      (2)       66      (2)
ENV$_FUNCTIONAL                 =00000000     66      (2)       66      (2)
ENV$_REPAIR                     =00000001     66      (2)       66      (2)
ENV$_SUPER                      =00000001     66      (2)       212     (3)
ENV$_SYSTEM                     =00000001     66      (2)       66      (2)
FMT_SUMMARY                      000000BE-R   328     (5)       588     (10)
GPR_TABLE                        00000095-R   138     (2)
HALT                            =00000004     303     (5)
HARDCNT                          00000000-R   256     (4)     #-368     (7)     #-687     (14)    #-689     (14)
HP$A_DEPENDENT                   00000032     86      (2)       87      (2)
HP$A_DEVICE                      00000018     86      (2)
HP$A_DVA                         0000001C     86      (2)
HP$A_LINK                        00000020     86      (2)
HP$B_DRIVE                       0000000B     86      (2)     #-393     (7)     #-420     (7)
HP$B_FLAGS                       0000000A     86      (2)
HP$KA_L_FLAGS                   =0000000C     103     (2)     #-412     (7)
HP$KA_MEM_SIZE                  =00000044     104     (2)     #-413     (7)
HP$M_ALLOC                      =00000001     86      (2)
HP$M_WASALL                     =00000002     86      (2)
HP$Q_DEVICE                      00000000     86      (2)
HP$T_DEVICE                      0000000C     86      (2)
HP$T_TYPE                        00000026     86      (2)     #-390     (7)     #-406     (7)     #-408     (7)     #-415     (7)
                                                              #-417     (7)
HP$V_ALLOC                      =00000000     86      (2)
HP$V_WASALL                     =00000001     86      (2)
HP$W_SIZE                        00000008     86      (2)
HP$W_VECTOR                      00000024     86      (2)
INITIALIZE                       00000000-R   367     (7)       212     (3)
INITIALIZE_X                     00000262-R   458     (7)
IPR                             =00000006     303     (5)
KA$B_ACC_TYPE                    00000042     87      (2)
KA$B_SID_TYPE                    0000003D     87      (2)
```

```
KA$K_LEN                        00000043     87      (2)
KA$L_FLAGS                      00000032     87      (2)
KA$L_SID                        0000003A     87      (2)
KA$M_CHAR                      =00000100     87      (2)
KA$M_CRC                       =00000010     87      (2)
KA$M_D_FLOAT                   =00000002     87      (2)
KA$M_EDIT                      =00000080     87      (2)
KA$M_F_FLOAT                   =00000001     87      (2)
KA$M_G_FLOAT                   =00000004     87      (2)
KA$M_H_FLOAT                   =00000008     87      (2)
KA$M_PACKED                    =00000020     87      (2)
KA$M_PACK_MUL                  =00000040     87      (2)
KA$M_TODR                      =00010000     87      (2)
KA$V_CHAR                      =00000008     87      (2)
KA$V_CRC                       =00000004     87      (2)
KA$V_D_FLOAT                   =00000001     87      (2)
KA$V_EDIT                      =00000007     87      (2)
KA$V_F_FLOAT                   =00000000     87      (2)
KA$V_G_FLOAT                   =00000002     87      (2)
KA$V_H_FLOAT                   =00000003     87      (2)
KA$V_PACKED                    =00000005     87      (2)
KA$V_PACK_MUL                  =00000006     87      (2)
KA$V_TODR                      =00000010     87      (2)
KA$W_LATENCY                    0000003E     87      (2)
KA$W_PROGRESS                   00000040     87      (2)
KA$W_RESERVED                   00000038     87      (2)
KA$W_WCS                        00000036     87      (2)
KA730                          =00000001     311     (5)
KA730_REV_54                   =03003600     89      (2)
KA_PTABLE                       0000018B-R   238     (3)    #-382    (7)    #-411    (7)
KA_UNIT_NO                      00000071-R   128     (2)    #-410    (7)
L$A_CCP                         00000040-R   212     (3)
L$A_DEVP                        0000001C-R   212     (3)
L$A_DREG                        00000024-R   212     (3)
L$A_DTP                         00000018-R   212     (3)
L$A_ICP                         0000003C-R   212     (3)
L$A_LASTAD                      00000014-R   212     (3)
L$A_NAME                        00000008-R   212     (3)
L$A_REPP                        00000044-R   212     (3)
L$A_SECNAM                      00000050-R   212     (3)
L$A_STATAB                      00000048-R   212     (3)
L$A_TSTCNT                      00000054-R   212     (3)    #-591    (10)   #-630    (12)
L$L_ENVIRON                     00000004-R   212     (3)
L$L_ERRTYP                      0000004C-R   212     (3)
L$L_HEADLENGTH                  00000000-R   212     (3)
L$L_REV                         0000000C-R   212     (3)
L$L_UNIT                        00000020-R   212     (3)
L$L_UPDATE                      00000010-R   212     (3)
LASTAD                          00000000-R   212     (3)      212    (3)
LDPCTX                         =0000000B     303     (5)
L_ACT_ISP                       00000058-R   120     (2)
L_ACT_KSP                       00000054-R   119     (2)
L_ACT_PSL                       0000005C-R   121     (2)
L_ACT_SP                        00000050-R   118     (2)
L_EXP_AP                        000000C5-R   151     (2)
L_EXP_FP                        000000C9-R   152     (2)
L_EXP_ISP                       000000D5-R   155     (2)
```

```
L_EXP_KSP                       000000D1-R    154     (2)
L_EXP_PSL                       000000D9-R    156     (2)
L_EXP_SP                        000000CD-R    153     (2)
L_ISP                           00000064-R    123     (2)
L_KA_FLAGS                      0000008D-R    135     (2)     #-412     (7)
L_KSIZE_PTBL                    00000091-R    136     (2)     #-413     (7)
L_KSP                           00000060-R    122     (2)
L_LAST_UBE                      00000068-R    125     (2)     #-383     (7)     #-392     (7)
L_TSTCNT                        00000000-R    212     (3)       212     (3)
MANUAL                         =00000001      303     (5)
MCHK                           =00000005      303     (5)
MEM                            =00000009      303     (5)
MM_MSG                          000000EE-R    331     (5)       375     (7)
MODELIST                        00000156-R    184     (2)
M_DSABL_ECC                    =02000000      100     (2)     #-519     (8)
PAR$M_ATDEF                    =00000010      81      (2)
PAR$M_ATHI                     =00000008      81      (2)
PAR$M_ATLO                     =00000004      81      (2)
PAR$M_NODEF                    =00000002      81      (2)
PAR$V_ATDEF                    =00000004      81      (2)
PAR$V_ATHI                     =00000003      81      (2)
PAR$V_ATLO                     =00000002      81      (2)
PAR$V_NODEF                    =00000001      81      (2)
PAR$_BIN                       =00000002      81      (2)
PAR$_DEC                       =0000000A      81      (2)
PAR$_HEX                       =00000010      81      (2)
PAR$_NO                        =00000000      81      (2)
PAR$_OCT                       =00000008      81      (2)
PAR$_YES                       =00000001      81      (2)
PART_INIT                       0000188B-R    683     (14)
POWER                          =00000003      303     (5)
PR$S_SID_ECO                   =00000009      83      (2)
PR$S_SID_PL                    =00000003      83      (2)
PR$S_SID_SN                    =0000000C      83      (2)
PR$S_SID_TYPE                  =00000008      83      (2)
PR$V_SID_ECO                   =0000000F      83      (2)
PR$V_SID_PL                    =0000000C      83      (2)
PR$V_SID_SN                    =00000000      83      (2)
PR$V_SID_TYPE                  =00000018      83      (2)
PR$_ACCR                       =00000029      83      (2)
PR$_ACCS                       =00000028      83      (2)
PR$_ASTLVL                     =00000013      83      (2)
PR$_CADR                       =00000025      83      (2)
PR$_CAER                       =00000027      83      (2)
PR$_CMIERR                     =00000017      83      (2)
PR$_CRBT                       =00000043      83      (2)
PR$_CSRD                       =0000001D      83      (2)
PR$_CSRS                       =0000001C      83      (2)     #-516     (8)
PR$_CSTD                       =0000001F      83      (2)
PR$_CSTS                       =0000001E      83      (2)     #-517     (8)
PR$_CSWP                       =00000042      83      (2)
PR$_ESP                        =00000001      83      (2)
PR$_ICCS                       =00000018      83      (2)
PR$_ICR                        =0000001A      83      (2)
PR$_IPL                        =00000012      83      (2)
PR$_ISP                        =00000004      83      (2)
PR$_KSP                        =00000000      83      (2)
```

```
PR$_MAPEN                       =00000038     83      (2)
PR$_MCESR                       =00000026     83      (2)
PR$_MCTL1                       =00000044     83      (2)
PR$_MCTL2                       =00000045     83      (2)
PR$_MDCR1                       =00000049     83      (2)
PR$_MEAR                        =00000048     83      (2)
PR$_MECCR                       =0000004B     83      (2)
PR$_MEDR                        =0000004A     83      (2)
PR$_MGEN                        =00000046     83      (2)
PR$_MTBER                       =00000047     83      (2)
PR$_NICR                        =00000019     83      (2)
PR$_P0BR                        =00000008     83      (2)
PR$_P0LR                        =00000009     83      (2)
PR$_P1BR                        =0000000A     83      (2)
PR$_P1LR                        =0000000B     83      (2)
PR$_PAMACC                      =00000040     83      (2)
PR$_PAMLOC                      =00000041     83      (2)
PR$_PCBB                        =00000010     83      (2)
PR$_PME                         =0000003D     83      (2)
PR$_RXCS                        =00000020     83      (2)
PR$_RXDB                        =00000021     83      (2)
PR$_SBIER                       =00000034     83      (2)
PR$_SBIFS                       =00000030     83      (2)
PR$_SBIMT                       =00000033     83      (2)
PR$_SBIQC                       =00000036     83      (2)
PR$_SBIS                        =00000031     83      (2)
PR$_SBISC                       =00000032     83      (2)
PR$_SBITA                       =00000035     83      (2)
PR$_SBR                         =0000000C     83      (2)
PR$_SCBB                        =00000011     83      (2)
PR$_SID                         =0000003E     83      (2)
PR$_SID_TYP730                  =00000003     83      (2)
PR$_SID_TYP750                  =00000002     83      (2)
PR$_SID_TYP780                  =00000001     83      (2)
PR$_SID_TYP7VV                  =00000004     83      (2)
PR$_SID_TYPMAX                  =00000004     83      (2)
PR$_SIRR                        =00000014     83      (2)
PR$_SISR                        =00000015     83      (2)
PR$_SLR                         =0000000D     83      (2)
PR$_SSP                         =00000002     83      (2)
PR$_TBCHK                       =0000003F     83      (2)
PR$_TBDR                        =00000024     83      (2)
PR$_TBIA                        =00000039     83      (2)
PR$_TBIS                        =0000003A     83      (2)
PR$_TODR                        =0000001B     83      (2)
PR$_TXCS                        =00000022     83      (2)
PR$_TXDB                        =00000023     83      (2)
PR$_UBRESET                     =00000037     83      (2)
PR$_USP                         =00000003     83      (2)
PR$_WCSA                        =0000002C     83      (2)
PR$_WCSD                        =0000002D     83      (2)
PSL$C_EXEC                      =00000001     82      (2)
PSL$C_KERNEL                    =00000000     82      (2)
PSL$C_SUPER                     =00000002     82      (2)
PSL$C_USER                      =00000003     82      (2)
PSL$K_EXEC                      =00000001     82      (2)
PSL$K_KERNEL                    =00000000     82      (2)
```

```
PSL$K_SUPER                     =00000002     82     (2)
PSL$K_USER                      =00000003     82     (2)
PSL$M_C                         =00000001     82     (2)
PSL$M_CBIT                      =00000001     82     (2)
PSL$M_CM                        =80000000     82     (2)       82     (2)
PSL$M_CURMOD                    =03000000     82     (2)
PSL$M_DV                        =00000080     82     (2)
PSL$M_FPD                       =08000000     82     (2)       82     (2)
PSL$M_FU                        =00000040     82     (2)
PSL$M_IPL                       =001F0000     82     (2)
PSL$M_IS                        =04000000     82     (2)
PSL$M_IV                        =00000020     82     (2)
PSL$M_N                         =00000008     82     (2)
PSL$M_NBIT                      =00000008     82     (2)
PSL$M_PRVMOD                    =00C00000     82     (2)
PSL$M_SAFBITS                   =000037FF     82     (2)
PSL$M_TBIT                      =00000010     82     (2)
PSL$M_TP                        =40000000     82     (2)       82     (2)
PSL$M_V                         =00000002     82     (2)
PSL$M_VBIT                      =00000002     82     (2)
PSL$M_Z                         =00000004     82     (2)
PSL$M_ZBIT                      =00000004     82     (2)
PSL$S_CURMOD                    =00000002     82     (2)
PSL$S_IPL                       =00000005     82     (2)
PSL$S_PRVMOD                    =00000002     82     (2)
PSL$V_C                         =00000000     82     (2)
PSL$V_CBIT                      =00000000     82     (2)
PSL$V_CM                        =0000001F     82     (2)
PSL$V_CURMOD                    =00000018     82     (2)
PSL$V_DV                        =00000007     82     (2)
PSL$V_FPD                       =0000001B     82     (2)
PSL$V_FU                        =00000006     82     (2)
PSL$V_IPL                       =00000010     82     (2)
PSL$V_IS                        =0000001A     82     (2)
PSL$V_IV                        =00000005     82     (2)
PSL$V_N                         =00000003     82     (2)
PSL$V_NBIT                      =00000003     82     (2)
PSL$V_PRVMOD                    =00000016     82     (2)
PSL$V_TBIT                      =00000004     82     (2)
PSL$V_TP                        =0000001E     82     (2)
PSL$V_V                         =00000001     82     (2)
PSL$V_VBIT                      =00000001     82     (2)
PSL$V_Z                         =00000002     82     (2)
PSL$V_ZBIT                      =00000002     82     (2)
QWD_TIME                         000000AB-R   317    (5)        584    (10)  #-588    (10)
Q_TEST_TIME                      00000004-R   258    (4)  #-694    (14)  #-696    (14)  697    (14)  #-698    (14)
                                                         #-700    (14)
REG_TABLE                        000000DD-R   160    (2)
SCB$L_ACCESS                     00000020     77     (2)
SCB$L_ARITH                      00000034     77     (2)
SCB$L_BREAK                      0000002C     77     (2)
SCB$L_CHME                       00000044     77     (2)  #-498    (8)
SCB$L_CHMK                       00000040     77     (2)  #-501    (8)
SCB$L_CHMS                       00000048     77     (2)  #-499    (8)
SCB$L_CHMU                       0000004C     77     (2)  #-500    (8)
SCB$L_COMPAT                     00000030     77     (2)
SCB$L_KNLSTK                     00000008     77     (2)
```

```
SCB$L_MACHCHK                   00000004     77      (2)     #-502     (8)
SCB$L_OPCCUS                    00000014     77      (2)     #-503     (8)
SCB$L_OPCDEC                    00000010     77      (2)     #-504     (8)
SCB$L_POWER                     0000000C     77      (2)     #-497     (8)
SCB$L_RADRMOD                   0000001C     77      (2)
SCB$L_ROPRAND                   00000018     77      (2)
SCB$L_RXDB                      000000F8     77      (2)
SCB$L_SFTLVL1                   00000084     77      (2)
SCB$L_SFTLVL10                  000000A8     77      (2)
SCB$L_SFTLVL11                  000000AC     77      (2)
SCB$L_SFTLVL12                  000000B0     77      (2)
SCB$L_SFTLVL13                  000000B4     77      (2)
SCB$L_SFTLVL14                  000000B8     77      (2)
SCB$L_SFTLVL15                  000000BC     77      (2)
SCB$L_SFTLVL2                   00000088     77      (2)
SCB$L_SFTLVL3                   0000008C     77      (2)
SCB$L_SFTLVL4                   00000090     77      (2)
SCB$L_SFTLVL5                   00000094     77      (2)
SCB$L_SFTLVL6                   00000098     77      (2)
SCB$L_SFTLVL7                   0000009C     77      (2)
SCB$L_SFTLVL8                   000000A0     77      (2)
SCB$L_SFTLVL9                   000000A4     77      (2)
SCB$L_TBIT                      00000028     77      (2)
SCB$L_TIMER                     000000C0     77      (2)     #-496     (8)
SCB$L_TRANSL                    00000024     77      (2)
SCB$L_TXDB                      000000FC     77      (2)
SCB$L_UBE_0                    =00000348     95      (2)     #-505     (8)
SCB$L_UBE_1                    =0000034C     96      (2)     #-506     (8)
SCB$L_UBE_2                    =00000350     97      (2)     #-507     (8)
SCB$L_UBE_3                    =00000354     98      (2)     #-508     (8)
SCB$L_ZERO                      00000000     77      (2)
SECTION                         00000057-R   303     (5)       212     (3)
SEP_FUNCTIONAL                 =00000002     66      (2)
SEP_REPAIR                     =00000003     66      (2)
SIZ...                         =00000001     87      (2)       66      (2)       80      (2)       81      (2)       82      (2)
                                                                 84      (2)       86      (2)       87      (2)
SUMMARY                         00000000-R   575     (10)      212     (3)
SUMMARY_INIT                    0000012B-R   628     (12)      430     (7)
SUMMARY_X                       00000046-R   593     (10)
SUM_A_BLOCK                     0000000C-R   274     (4)     #-576     (10)      634     (12)    #-639     (12)    #-640     (12)
                                                             #-642     (12)    #-686     (14)
SUM_BLK_SIZE                    00000014-R   286     (4)     #-515     (8)     #-638     (12)
SUM_K_SIZE                      0000000C     282     (4)     #-589     (10)    #-631     (12)    #-685     (14)
SUM_L_HARDCNT                   00000000     280     (4)     #-579     (10)    #-588     (10)    #-688     (14)
SUM_Q_RUNTIME                   00000004     281     (4)       584     (10)    #-695     (14)    #-696     (14)    #-699     (14)
                                                             #-700     (14)
SYS$ALLOC                       00010238     79      (2)
SYS$ASCTIM                      00010248     79      (2)       584     (10)
SYS$ASSIGN                      00010250     79      (2)
SYS$BINTIM                      00010258     79      (2)
SYS$CANCEL                      00010260     79      (2)
SYS$CANTIM                      00010268     79      (2)
SYS$CLOSE                       000105B8     79      (2)
SYS$CLREF                       00010298     79      (2)
SYS$CONNECT                     000105C0     79      (2)
SYS$DALLOC                      000102D8     79      (2)
SYS$DASSGN                      000102E0     79      (2)
```

```
SYS$DISCONNECT                  000105D0      79      (2)
SYS$FAO                         00010350      79      (2)
SYS$FAOL                        00010358      79      (2)
SYS$GET                         00010580      79      (2)
SYS$GETCHN                      000104C8      79      (2)
SYS$GETTIM                      00010378      79      (2)       697     (14)
SYS$LKWSET                      000103A0      79      (2)
SYS$NUMTIM                      000103B8      79      (2)
SYS$OPEN                        00010608      79      (2)
SYS$QIO                         000103C8      79      (2)
SYS$QIOW                        00010200      79      (2)
SYS$READ                        00010590      79      (2)
SYS$READEF                      000103D0      79      (2)
SYS$SETAST                      000103F8      79      (2)
SYS$SETEF                       00010400      79      (2)
SYS$SETIMR                      00010420      79      (2)
SYS$SETPRI                      00010428      79      (2)
SYS$SETPRT                      00010430      79      (2)
SYS$SETRWM                      00010438      79      (2)
SYS$SETSWM                      00010448      79      (2)
SYS$ULKPAG                      00010460      79      (2)
SYS$ULWSET                      00010468      79      (2)
SYS$UNWIND                      00010470      79      (2)
SYS$WAITFR                      00010478      79      (2)
SYS$WFLAND                      00010488      79      (2)
SYS$WFLOR                       00010490      79      (2)
S_ALL                           00000044-R    303     (5)       303     (5)
S_DEFAULT                       00000018-R    303     (5)       303     (5)
S_HALT                          00000032-R    303     (5)       303     (5)
S_IPR                           0000003C-R    303     (5)       303     (5)
S_LDPCTX                        00000050-R    303     (5)       303     (5)
S_MANUAL                        00000020-R    303     (5)       303     (5)
S_MCHK                          00000037-R    303     (5)       303     (5)
S_MEM                           00000048-R    303     (5)       303     (5)
S_POWER                         0000002C-R    303     (5)       303     (5)
S_TU58                          00000027-R    303     (5)       303     (5)
S_UBI                           0000004C-R    303     (5)       303     (5)
S_WCS                           00000040-R    303     (5)       303     (5)
TU0_UNIT_NO                     00000075-R    129     (2)     #-424     (7)
TU1_UNIT_NO                     00000079-R    130     (2)     #-426     (7)
TU58                           =00000002      311     (5)
T_KA730                         0000008B-R    311     (5)       311     (5)     #-406     (7)     #-408     (7)
T_NAME                          00000058-R    212     (3)       212     (3)
T_TIME                          000000B3-R    321     (5)       319     (5)
T_TU58                          00000091-R    311     (5)       311     (5)     #-415     (7)     #-417     (7)
T_UBE                           00000096-R    311     (5)       311     (5)     #-390     (7)
UBE                            =00000003      311     (5)
UBI                            =0000000A      303     (5)
WCS                            =00000007      303     (5)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$ASCTIM_S	1	582 (10)	582 (10)
$CHFDEF	1	85 (2)	85 (2)
$D1_ABPROGRAM	1	376 (7)	376 (7) 511 (8)
$D1_ERR	1	636 (12)	636 (12)
$D1_ERR_S	1	636 (12)	636 (12)
$D1_PRINT_S	1	375 (7)	375 (7) 588 (10)
$D1_SEC	1	303 (5)	303 (5)
$DEF	1	88 (2)	280 (4) 281 (4) 282 (4) 77 (2) 79 (2)
			80 (2) 85 (2) 86 (2) 87 (2)
$DEFEND	1	77 (2)	77 (2) 79 (2) 80 (2) 82 (2) 83 (2)
			85 (2) 86 (2) 87 (2)
$DEFINI	1	77 (2)	77 (2) 79 (2) 80 (2) 82 (2) 83 (2)
			85 (2) 86 (2) 87 (2)
$DS_ABORT	1	376 (7)	376 (7) 511 (8)
$DS_BGNCLEAN	1	495 (8)	495 (8)
$DS_BGNINIT	1	367 (7)	367 (7)
$DS_BGNMOD	1	66 (2)	66 (2)
$DS_BGNSUMMARY	1	575 (10)	575 (10)
$DS_BNCOMPLETE	1	435 (7)	435 (7)
$DS_BNERROR	1	387 (7)	387 (7) 635 (12)
$DS_BPASS0	1	370 (7)	370 (7)
$DS_BREAK	1	458 (7)	458 (7) 520 (8) 593 (10)
$DS_CHCDEF	1	84 (2)	84 (2)
$DS_CHDEF	1	84 (2)	84 (2)
$DS_CHIDEF	1	84 (2)	84 (2)
$DS_CHMDEF	2	84 (2)	84 (2)
$DS_CHSDEF	3	84 (2)	84 (2)
$DS_CLRVEC_S	1	496 (8)	496 (8) 497 (8) 498 (8) 499 (8) 500 (8)
			501 (8) 502 (8) 503 (8) 504 (8) 505 (8)
			506 (8) 507 (8) 508 (8)
$DS_DEVTYP	1	311 (5)	311 (5)
$DS_DISPATCH	1	225 (3)	225 (3)
$DS_DSADEF	1	80 (2)	80 (2)
$DS_DSBDEF	1	80 (2)	
$DS_DSDEF	2	78 (2)	78 (2)
$DS_DSSDEF	1	79 (2)	79 (2)
$DS_ENDCLEAN	1	520 (8)	520 (8)
$DS_ENDINIT	1	458 (7)	458 (7)
$DS_ENDMOD	1	704 (14)	704 (14)
$DS_ENDPASS_G	1	457 (7)	457 (7)
$DS_ENDSUMMARY	1	593 (10)	593 (10)
$DS_ENVDEF	2	66 (2)	66 (2)
$DS_ERRSYS_S	1	636 (12)	636 (12)
$DS_GETBUF_S	1	434 (7)	434 (7) 446 (7) 447 (7) 448 (7) 449 (7)
			450 (7) 451 (7) 452 (7) 453 (7) 454 (7)
			634 (12)
$DS_GPHARD_S	1	386 (7)	386 (7)
$DS_HEADER	4	210 (3)	210 (3)
$DS_HPODEF	2	86 (2)	86 (2)
$DS_INITSCB_G	1	512 (8)	512 (8)

```
$DS_MMOFF_G         1       373     (7)     373     (7)     509     (8)
$DS_PARDEF          1       81      (2)     81      (2)
$DS_PRINTF_S        1       375     (7)     375     (7)
$DS_PRINTS_S        1       585     (10)    585     (10)
$DS_PSLDEF          1       82      (2)     82      (2)
$DS_QADEF           2       88      (2)     88      (2)
$DS_RELBUF_S        1       442     (7)     442     (7)     514     (8)     515     (8)
$DS_SCBDEF          1       77      (2)     77      (2)
$DS_SECTION         2       302     (5)     302     (5)
$DS_SETIPL_S        1       518     (8)     518     (8)
$EQU                1       88      (2)     66      (2)     78      (2)     81      (2)     82      (2)     83      (2)
                                            84      (2)     88      (2)
$EQULS1             1       88      (2)     66      (2)     78      (2)     82      (2)     84      (2)     88      (2)
$EQULST             1       66      (2)     66      (2)     78      (2)     82      (2)     84      (2)     88      (2)
$GBLINI             2       66      (2)     66      (2)     77      (2)     78      (2)     79      (2)     80      (2)
                                            81      (2)     82      (2)     83      (2)     84      (2)     85      (2)
                                            86      (2)     87      (2)     88      (2)
$GETTIM_S           1       697     (14)    697     (14)
$HP_DEFDEF          1       86      (2)
$KADEF              1       87      (2)
$PRDEF              1       83      (2)     83      (2)
$PSLDEF             1       82      (2)     82      (2)
$PUSHADR            1       434     (7)     434     (7)     442     (7)     446     (7)     447     (7)     448     (7)
                                            449     (7)     450     (7)     451     (7)     452     (7)     453     (7)
                                            454     (7)     514     (8)     515     (8)     584     (10)    634     (12)
                                            636     (12)    697     (14)
$VIELD              1       66      (2)     66      (2)     80      (2)     81      (2)     82      (2)     84      (2)
                                            86      (2)     87      (2)
$VIELD1             1       88      (2)     66      (2)     80      (2)     81      (2)     82      (2)     84      (2)
                                            86      (2)     87      (2)
KA_DEF              2       87      (2)     87      (2)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.06	00:00:00.41
Command processing	22	00:00:00.20	00:00:00.68
Pass 1	895	00:00:17.62	00:00:30.99
Symbol table sort	0	00:00:01.02	00:00:01.61
Pass 2	256	00:00:03.39	00:00:04.39
Symbol table output	89	00:00:00.55	00:00:00.63
Psect synopsis output	6	00:00:00.05	00:00:00.09
Cross-reference output	180	00:00:02.87	00:00:04.25
Assembler run totals	1479	00:00:25.77	00:00:43.06

```
The working set limit was 1000 pages.
108793 bytes (213 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 712 non-local and 25 local symbols.
705 source lines were read in Pass 1, producing 55 object records in Pass 2.
155 pages of virtual memory were used to define 44 macros.
```

```
                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                       Macros defined
------------------                       --------------
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72                         1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812                      47
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1                     11
TOTALS (all libraries)                                59

1082 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAX1
```

Table of contents

```
                        1          .SBTTL  ENKAXA: RESERVED PROCESSOR REGISTERS
                        2          .TITLE  ENKAXA  VAX-11/730 Specific CPU Cluster Exerciser
                        3          .IDENT  /1-3/
                        4
                        5 ;
                        6 ; Copyright (C) 1981
                        7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
                        8 ;
                        9 ; This software is furnished  under a license for use only on a single
                       10 ; computer  system and  may be  copied only with  the inclusion of the
                       11 ; above copyright notice.  This software, or any other copies thereof,
                       12 ; may not be provided or  otherwise made available to any other person
                       13 ; except for use on such system and to one who agrees to these license
                       14 ; terms.  Title to and  ownership of the  software  shall at all times
                       15 ; remain in DEC.
                       16 ;
                       17 ; The information in this software is subject to change without notice
                       18 ; and should  not be construed  as a commitment  by Digital  Equipment
                       19 ; Corporation.
                       20 ;
                       21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
                       22 ; software on equipment which is not supplied by DEC.
                       23 ;
                       24
                       25 ;++
                       26 ; Facility:      VAX Architecture Diagnostic.
                       27 ;
                       28 ; Abstract:      This module tests that all processor register numbers not
                       29 ;                implemented on the VAX-11/730 will cause a reseved operand
                       30 ;                fault for a MTPR or MFPR instruction. Also internal processor
                       31 ;                registers which are write and read only and specific to the
                       32 ;                VAX-11/730 are checked to not cause reserved operand faults,
                       33 ;                except in the cases where they should fault.
                       34 ;
                       35 ;
                       36 ; Environment:   VAX Diagnostic Supervisor.
                       37 ;
                       38 ; Author:        Rich Lewis      7-Jly-81        Version 1.0
                       39 ;
                       40 ;                Tai-Chun Pan    01-Apr-83       Version 1.2
                       41 ;
                       42 ;                Added quick flag on Test 7(TU58). If quick flag is set
                       43 ;                will skip subtest 4 through subtest 11.  Added event flag 3.
                       44 ;                If event flag 3 is set, will override protection,
                       45 ;                i.e., go ahead and write on tape.  If run APT & event flag 3
                       46 ;                is clear & tape is not scratch, will report an error.
                       47 ;                Modified Unibus sizer on Test 8 & 9. make test run faster.
                       48 ;
                       49 ;
                       50 ;                Tai-Chun Pan    03-Aug-83       Version 1.3
                       51 ;
                       52 ;                fixed bugs on TEST 7 and error summary routine call when
                       53 ;                running under APT.
                       54 ;
                       55 ;
                       56 ;--
```

```
                     0000     58          .SBTTL  DECLARATIONS
                     0000     59          .LIST   MEB
                     0000     60          .NLIST  CND
                00000000     61          .PSECT  ENKAXA, PAGE, NOWRT
                     0000     62          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     63          $DS_BGNMOD      CEP_REPAIR, TN=1
                     0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)''
                     0000     64
                     0000     65 ;
                     0000     66 ; Include files:
                     0000     67 ;
                     0000     68
                     0000     69          .LIBRARY        "ENKAX"                 ; CPU Cluster Exerciser library.
                     0000     70
                     0000     71
                     0000     72 ;
                     0000     73 ; Equated symbols:
                     0000     74 ;
                     0000     78
                     0000     79          $DS_SCBDEF
                     0000     80          $DS_DSDEF
                     0000     81          $DS_DSSDEF
                     0000     82          $DS_HPODEF
                     0000     83          KA_DEF
                     0000     84          $DS_ERRDEF
                     0000     85          ENKAX_SECDEF
                     0000     86
                     0000     87 ;
                     0000     88 ; Own storage:
                     0000     89 ;
                     0000     90
00000004             0000     91 L_REG_ADD:      .BLKL           ; contains the register address
00000005             0004     92 B_FLAGS:        .BLKB           ; contains the following flags
                     0005     93
00000000             0005     94 V_NO_FAULT=0                    ; bit position - fault did not occur
00000001             0005     95 M_NO_FAULT=1@0                  ; bit mask - expected fault did not occur
00000001             0005     96 V_FAULT=1                       ; bit position - fault sucess flag
00000002             0005     97 M_FAULT=1@1                     ; bit mask - fault sucess flag
                     0005     98
                     0005    102
```

```
                        0005   104          .SBTTL  Data Tables
                        0005   105
                        0005   106 ; This is a table that contains the unused internal processor register
                        0005   107 ; numbers. This table is terminated by a -1.
                        0005   108
00000005                0005   109 ADDRESS_TABLE:   .LONG    ^X5
00000006                0009   110                  .LONG    ^X6
00000007                000D   111                  .LONG    ^X7
0000000E                0011   112                  .LONG    ^XE
0000000F                0015   113                  .LONG    ^XF
00000016                0019   114                  .LONG    ^X16
00000017                001D   115                  .LONG    ^X17
00000029                0021   116                  .LONG    ^X29
0000002A                0025   117                  .LONG    ^X2A
0000002B                0029   118                  .LONG    ^X2B
0000002C                002D   119                  .LONG    ^X2C
0000002D                0031   120                  .LONG    ^X2D
0000002E                0035   121                  .LONG    ^X2E
0000002F                0039   122                  .LONG    ^X2F
0000003B                003D   123                  .LONG    ^X3B
0000003C                0041   124                  .LONG    ^X3C
00000040                0045   125                  .LONG    ^X40
00000080                0049   126                  .LONG    ^X80
00000100                004D   127                  .LONG    ^X100
00000200                0051   128                  .LONG    ^X200
00000400                0055   129                  .LONG    ^X400
00000800                0059   130                  .LONG    ^X800
00001000                005D   131                  .LONG    ^X1000
00002000                0061   132                  .LONG    ^X2000
00004000                0065   133                  .LONG    ^X4000
00008000                0069   134                  .LONG    ^X8000
00010000                006D   135                  .LONG    ^X10000
00020000                0071   136                  .LONG    ^X20000
00040000                0075   137                  .LONG    ^X40000
00080000                0079   138                  .LONG    ^X80000
00100000                007D   139                  .LONG    ^X100000
00200000                0081   140                  .LONG    ^X200000
00400000                0085   141                  .LONG    ^X400000
00800000                0089   142                  .LONG    ^X800000
01000000                008D   143                  .LONG    ^X1000000
02000000                0091   144                  .LONG    ^X2000000
04000000                0095   145                  .LONG    ^X4000000
08000000                0099   146                  .LONG    ^X8000000
10000000                009D   147                  .LONG    ^X10000000
20000000                00A1   148                  .LONG    ^X20000000
40000000                00A5   149                  .LONG    ^X40000000
80000000                00A9   150                  .LONG    ^X80000000
FFFFFFFF                00AD   151                  .LONG    ^XFFFFFFFF
                        00B1   152
                        00B1   153
                        00B1   154 ; This is a table that contains the read only internal processor register
                        00B1   155 ; numbers. This table is terminated by a -1.
                        00B1   156
0000001A                00B1   157 ADDRESS_TABLE1: .LONG    ^X1A
0000001D                00B5   158                  .LONG    ^X1D
00000021                00B9   159                  .LONG    ^X21
00000031                00BD   160                  .LONG    ^X31
```

```
0000003E  00C1   161                  .LONG   ^X3E
FFFFFFFF  00C5   162                  .LONG   ^XFFFFFFFF
          00C9   163
          00C9   164
          00C9   165 ; This is a table that contains the write only internal processor register
          00C9   166 ; numbers. This table is terminated by a -1.
          00C9   167
00000014  00C9   168 ADDRESS_TABLE2: .LONG   ^X14
00000019  00CD   169                  .LONG   ^X19
0000001F  00D1   170                  .LONG   ^X1F
00000023  00D5   171                  .LONG   ^X23
00000036  00D9   172                  .LONG   ^X36
00000037  00DD   173                  .LONG   ^X37
0000003F  00E1   174                  .LONG   ^X3F
FFFFFFFF  00E5   175                  .LONG   ^XFFFFFFFF
          00E9   176
          00E9   177
```

```
                                                00E9    179          .SBTTL  ERROR MESSAGES
                                                00E9    180 ;-----------------------------------------------------------------------
                                                00E9    181 T_NO_KA730:
69 76 65 64 20 6F 4E 20 20 2F 21 00'            00E9    182          .ASCIC  "!/  No devices of type KA730 selected.  Exiting test.!/"
20 65 70 79 74 20 66 6F 20 73 65 63             00F5
74 63 65 6C 65 73 20 30 33 37 41 4B             0101
67 6E 69 74 69 78 45 20 20 2E 64 65             010D
               2F 21 2E 74 73 65 74 20          0119
                                     37         00E9
                                                0121    183 ;-----------------------------------------------------------------------
                                                0121    184 T_NO_FAULT_TO:
45 52 20 64 65 74 63 65 70 78 45 00'            0121    185          .ASCIC  "Expected RESERVED OPERAND FAULT did not occur during!/"-
41 52 45 50 4F 20 44 45 56 52 45 53             012D
64 69 64 20 54 4C 55 41 46 20 44 4E             0139
64 20 72 75 63 63 6F 20 74 6F 6E 20             0145
20 65 76 6F 6D 2F 21 67 6E 69 72 75             0151
61 67 65 6C 6C 69 20 6E 61 20 6F 74             015D
20 72 6F 73 73 65 63 6F 72 70 20 6C             0169
64 64 61 20 72 65 74 73 69 67 65 72             0175
                  2F 21 73 73 65 72             0181
                                     65         0121
                                                0187    186                  "move to an illegal processor register address!/"
                                                0187    187 ;-----------------------------------------------------------------------
                                                0187    188 T_FAULT_ERR_F:
20 64 65 74 63 65 70 78 65 6E 55 00'            0187    189          .ASCIC  "Unexpected RESERVED OPERAND FAULT occured while attempting!/"-
45 50 4F 20 44 45 56 52 45 53 45 52             0193
6F 20 54 4C 55 41 46 20 44 4E 41 52             019F
65 6C 69 68 77 20 64 65 72 75 63 63             01AB
21 67 6E 69 74 70 6D 65 74 74 61 20             01B7
                                     2F         01C3
20 61 20 6E 6F 20 52 50 46 4D 20 61             01C4    190                  "a MFPR on a legal processor register address!/"
73 65 63 6F 72 70 20 6C 61 67 65 6C             01D0
72 65 74 73 69 67 65 72 20 72 6F 73             01DC
      2F 21 73 73 65 72 64 64 61 20             01E8
                                     6A         0187
                                                01F2    191 ;-----------------------------------------------------------------------
                                                01F2    192 T_NO_FAULT_FROM:
45 52 20 64 65 74 63 65 70 78 45 00'            01F2    193          .ASCIC  "Expected RESERVED OPERAND FAULT did not occur during!/"-
41 52 45 50 4F 20 44 45 56 52 45 53             01FE                                  `
64 69 64 20 54 4C 55 41 46 20 44 4E             020A
64 20 72 75 63 63 6F 20 74 6F 6E 20             0216
               2F 21 67 6E 69 72 75             0222
6E 61 20 6D 6F 72 66 20 65 76 6F 6D             0229    194                  "move from an illegal processor register address!/"
6F 72 70 20 6C 61 67 65 6C 6C 69 20             0235
73 69 67 65 72 20 72 6F 73 73 65 63             0241
21 73 73 65 72 64 64 61 20 72 65 74             024D
                                     2F         0259
                                     67         01F2
                                                025A    195 ;-----------------------------------------------------------------------
                                                025A    196 T_FAULT_ERR_T:
20 64 65 74 63 65 70 78 65 6E 55 00'            025A    197          .ASCIC  "Unexpected RESERVED OPERAND FAULT occured while attempting!/"-
45 50 4F 20 44 45 56 52 45 53 45 52             0266
6F 20 54 4C 55 41 46 20 44 4E 41 52             0272
65 6C 69 68 77 20 64 65 72 75 63 63             027E
21 67 6E 69 74 70 6D 65 74 74 61 20             028A
                                     2F         0296
20 61 20 6E 6F 20 52 50 54 4D 20 61             0297    198                  "a MTPR on a legal processor register address!/"
```

```
73 65 63 6F 72 70 20 6C 61 67 65 6C  02A3
72 65 74 73 69 67 65 72 20 72 6F 73  02AF
       2F 21 73 73 65 72 64 64 61 20  02BB
                                 6A  025A
                                     02C5    199 ;-----------------------------------------------------------------------
                                     02C5    200 T_REG_ADD:
41 20 72 65 74 73 69 67 65 52 20 00' 02C5    201         .ASCIC  " Register Addressed  =   ^X  !XL   !/"
20 3D 20 20 64 65 73 73 65 72 64 64  02D1
20 20 20 4C 58 21 20 20 58 5E 20 20  02DD
                              2F 21  02E9
                                 25  02C5
                                     02EB    202 ;-----------------------------------------------------------------------
                                     02EB    203
                                     02EB    204 ERR_MESSAGE:
                                     02EB    205         $DS_BGNMESSAGE
                                   0000  02EB                 .WORD   ^M<>                ; ENTRY MASK
                                     02ED    206         $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
                                   14 BC   9F  02ED                 PUSHAB  @ERR$_P1(AP)
                 000100E0 9F       01   FB  02F0                 CALLS   #$$N, @#DS$PRINTB
                                     02F7    207         $DS_PRINTB_S    FORMAT=T_REG_ADD,P0=L_REG_ADD
                           FD05 CF   DD  02F7                 PUSHL   L_REG_ADD
                             C7 AF   9F  02FB                 PUSHAB  T_REG_ADD
                 000100E0 9F       02   FB  02FE                 CALLS   #$$N, @#DS$PRINTB
                                     0305    208         $DS_ENDMESSAGE
                                   04  0305                 RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                                     0306    209
```

```
                        0306   211          .SBTTL SUBROUTINES AND SERVICE ROUTINES
                        0306   212
                        0306   213
                        0306   214 ; This routine will service all reserved operand faults.
                        0306   215
                        0306   216
                        0306   217          .ALIGN  LONG
                        0308   218 FAULT_SERVICE:
          6F    57  D0  0308   219          MOVL    R7,(SP)                 ; put return address on stack
    FCF4  CF    02  88  030B   220          BISB2   #M_FAULT,B_FLAGS        ; set the fault flag
                    02  0310   221          REI                             ; return
                        0311   222
                        0311   223          $DS_PAGE
```

```
                        0311                    .SBTTL  TEST 1:  Reserved Processor Registers
                    00000000                    .PSECT  TEST_001, PAGE, NOWRT
                        0020
                        0020    226 ;++
                        0020    227 ; Test description:
                        0020    228 ;
                        0020    229 ;       This test checks that processor numbers not used by the VAX-11/730
                        0020    230 ;       cause reserve operand faults for the MFPR and MTPR instructions.
                        0020    231 ;       All read only and write only registers are checked to cause a reserved
                        0020    232 ;       operand fault when the disallowed function is attempted. (i.e. - write
                        0020    233 ;       to a read only or read a write only)
                        0020    234 ;
                        0020    235 ;--
                        0020    236
                        0020    237         $DS_BGNTEST     <IPR,ALL,DEFAULT>
                        0020        DATA_001:
                    00000000    0020                .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                        0024        TEST_001::
                        0000    0024                .WORD   ^M<>            ; ENTRY MASK
                        0026    238
          00000000'EF   D5      0026    239         TSTL    KA_PTABLE                       ; CPU selected?
                  10    18      002C    240         BGEQ    5$                              ; br = CPU selected
                        002E    241         $DS_PRINTF_S    FORMAT=T_NO_KA730               ; print exit warning
          000000E9'EF   9F      002E                PUSHAB  T_NO_KA730
      000100F0 9F       01      FB  0034            CALLS   #1,@#DS$PRINTF
                        003B    242         $DS_EXIT        TEST                            ; exit test
                        0291    31  003B            BRW     TEST_001_X      ; EXIT TEST 1
                        003E    243 5$:
                        003E    244
                        003E    245         $DS_PAGE
```

```
                                                .SBTTL  SUBTEST 1.1:  Reserved Processor Register Numbers
                                        ;+
                                        ; Subtest description:
                                        ;
                                        ;       This subtest checks all the reserved processor register numbers that
                                        ;       are not used in the VAX-11/730 processor.
                                        ;
                                        ; Subtest steps:
                                        ;
                                        ;       SUBTEST RESERVED PROCESSOR REGISTER NUMBERS
                                        ;       :  Initialize pointer to address table
                                        ;       :  Get reserved operand fault vector to handle exception
                                        ;       :  REPEAT
                                        ;       :  :  Clear error flags
                                        ;       :  :  Get address of register to attempt access
                                        ;       :  :  Attempt MTPR on register
                                        ;       :  :  Check for errors
                                        ;       :  :  IF any errors
                                        ;       :  :  :  THEN
                                        ;       :  :  :  Report error
                                        ;       :  :  :  Exit test
                                        ;       :  :  ENDIF
                                        ;       :  :  Attempt MFPR on register
                                        ;       :  :  Check for errors
                                        ;       :  :  IF any errors
                                        ;       :  :  :  THEN
                                        ;       :  :  :  Report error
                                        ;       :  :  :  Exit test
                                        ;       :  :  ENDIF
                                        ;       :  : UNTIL All unused addresses done
                                        ;       :  ENDREPEAT
                                        ;       ENDSUBTEST RESERVED PROCESSOR REGISTER NUMBERS
                                        ;
                                        ;
                                        ;-
                                        $DS_BGNSUB
                                    T1_S1::
00010030 9F   00000000'EF   FA      003E                CALLG   $S$, @#DS$BGNSUB
         53   00000005'EF   9E      0049    284         MOVAB   ADDRESS_TABLE,R3        ; load R3 with address of table
                                    0050    285         $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                        00  DD      0050                PUSHL   #0
              00000308'EF   DF      0052                PUSHAL  FAULT_SERVICE
                        18  DD      0058                PUSHL   #SCB$L_ROPRAND
         00010160 9F    03  FB      005A                CALLS   #3, @#DS$SETVEC
         57   00000082'EF   9E      0061    286 NEXT:   MOVAB   RENTRY_1,R7             ; load R7 with address of rentry point
              00000004'EF   94      0068    287         CLRB    B_FLAGS                 ; initialize error flag
                   55   63  D0      006E    288         MOVL    (R3),R5                 ; load address of IPR to access
         00000000'EF   55   D0      0071    289         MOVL    R5,L_REG_ADD            ; save register address for error report
                   55   01  DA      0078    290         MTPR    #1,R5                   ; access IPR (should cause fault)
         00000004'EF   01   88      0078    291         BISB2   #M_NO_FAULT,B_FLAGS     ; flag no fault error
                                    0082    292 RENTRY_1:
      1D 00000004'EF   00   E1      0082    293         BBC     #V_NO_FAULT,B_FLAGS,3$  ; br = MTPR did cause fault
                                    008A    294         $DS_ERRHARD_S   UNIT=RA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
              00000121'8F   DD      008A                PUSHL   #T_NO_FAULT_TO
              000002EB'EF   DF      0090                PUSHAL  ERR_MESSAGE
                        00  DD      0096                PUSHL   #0
```

```
                00000000'EF   DD  0098                        PUSHL   KA_UNIT_NO
                          01  DD  009E                        PUSHL   #$ER
                              00A0          ;;; TEST 1, SUBTEST 1, ERROR 1
                000100D0 9F   05  FB  00A0                    CALLS   #$$M, @#DS$ERRHARD
                              00A7  295 3$:     $DS_CKLOOP      NEXT                    ; scope loop at NEXT
              00010040 9F   B7 AF  FA  00A7                    CALLG   NEXT, @#DS$CKLOOP
                 57   000000C9'EF  9E  00AF  296 SCOPE:  MOVAB   RENTRY_2,R7          ; load R7 with address of rentry point
                      00000004'EF  94  00B6  297         CLRB    B_FLAGS             ; initialize error flag
                           55   63  D0  00BC  298         MOVL    (R3),R5             ; load address of IPR to access
                           56   55  DB  00BF  299         MFPR    R5,R6               ; access IPR (should cause fault)
                00000004'EF   01  88  00C2  300         BISB2   #M_NO_FAULT,B_FLAGS ; set no fault error flag
                              00C9  301 RENTRY_2:
             1D 00000004'EF   00  E1  00C9  302         BBC     #V_NO_FAULT,B_FLAGS,4$  ; br = MFPR did cause fault
                              00D1  303         $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_FROM
                      000001F2'8F  DD  00D1                    PUSHL   #T_NO_FAULT_FROM
                      000002EB'EF  DF  00D7                    PUSHAL  ERR_MESSAGE
                           00      DD  00DD                    PUSHL   #0
                00000000'EF       DD  00DF                    PUSHL   KA_UNIT_NO
                          02      DD  00E5                    PUSHL   #$ER
                              00E7          ;;; TEST 1, SUBTEST 1, ERROR 2
                000100D0 9F   05  FB  00E7                    CALLS   #$$M, @#DS$ERRHARD
                              00EE  304 4$:     $DS_CKLOOP      SCOPE                   ; scope loop at SCOPE
              00010040 9F   BE AF  FA  00EE                    CALLG   SCOPE, @#DS$CKLOOP
                          83  D5  00F6  305 5$:     TSTL    (R3)+               ; advance pointer
                 63   FFFFFFFF 8F  D1  00F8  306         CMPL    #-1,(R3)            ; check for end of table
                          03  13  00FF  307         BEQL    6$                  ; br = finished with all addresses
                        FF5D  31  0101  308         BRW     NEXT                ; go do next address
                              0104  309 6$:     $DS_CLRVEC_S    VECTOR=#SCB$L_ROPRAND
                          18  DD  0104                    PUSHL   #SCB$L_ROPRAND
                000100168 9F  01  FB  0106                    CALLS   #1, @#DS$CLRVEC
                              010D  310
                              010D  311         $DS_ENDSUB
                              010D      T1_S1_X:
          00010038 9F   00000000'EF   FA  010D                    CALLG   $S$, @#DS$ENDSUB
                              0118  312
                              0118  313         $DS_PAGE
```

```
                                                .SBTTL  SUBTEST 1.2:  Read only Processor Registers
                                ;+
                                ; Subtest description:
                                ;
                                ;       This subtest checks processor-specific registers which are read only
                                ;       registers. This subtest attempts a read and checks that a reserved
                                ;       operand fault does not occur.  Then an attempt to write the register
                                ;       is made which should cause a reserved operand fault. If an error is
                                ;       encountered it is reported.
                                ;
                                ;
                                ; Subtest steps:
                                ;
                                ;       SUBTEST READ ONLY PROCESSOR REGISTERS
                                ;       :  Get reserved operand fault vector to handle exception
                                ;       :  REPEAT
                                ;       :  :  Clear error flags
                                ;       :  :  Attempt MTPR on register
                                ;       :  :  Check for errors
                                ;       :  :  IF any errors
                                ;       :  :  :  THEN
                                ;       :  :  :  Report error
                                ;       :  :  :  Exit test
                                ;       :  :  ENDIF
                                ;       :  :  Attempt MFPR on register
                                ;       :  :  Check for errors
                                ;       :  :  IF any errors
                                ;       :  :  :  THEN
                                ;       :  :  :  Report error
                                ;       :  :  :  Exit test
                                ;       :  :  ENDIF
                                ;       :  : UNTIL all read only registers are done
                                ;       :  ENDREPEAT
                                ;       ENDSUBTEST READ ONLY PROCESSOR REGISTERS
                                ;
                                ;
                                ;
                                ;-
                                        $DS_BGNSUB
                                T1_S2::
00010030 9F   0000000C'EF   FA          CALLG   $S$, a#DS$BGNSUB
         53   000000B1'EF   9E  354     MOVAB   ADDRESS_TABLE1,R3       ; load R3 with address of table
                                355     $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                        00    DD          PUSHL   #0
              00000308'EF     DF          PUSHAL  FAULT_SERVICE
                        18    DD          PUSHL   #SCB$L_ROPRAND
         00010160 9F    03    FB          CALLS   #3, a#DS$SETVEC
         57   0000015C'EF   9E  356 NEXT1:  MOVAB   RENTRY_3,R7            ; load R7 with address of rentry point
              00000004'EF   94  357     CLRB    B_FLAGS                 ; initialize error flag
                   55   63    D0  358     MOVL    (R3),R5                 ; load address of IPR to access
         00000000'EF    55    D0  359     MOVL    R5,L_REG_ADD            ; load reg address for error
                   55   01    DA  360     MTPR    #1,R5                   ; access IPR (should cause fault)
         00000004'EF    01    88  361     BISB2   #M_NO_FAULT,B_FLAGS     ; set no fault error flag
                                015C 362 RENTRY_3:
                                015C 363        $DS_BREAK                       ; check for ^C
         00010058 9F    6E    FA          CALLG   (SP), a#DS$BREAK
      1D 00000004'EF    00    E1  364     BBC     #V_NO_FAULT,B_FLAGS,3$  ; br = MTPR did cause fault
```

```
                                 016B   365          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
                 00000121'8F     DD  016B                    PUSHL   #T_NO_FAULT_TO
                 000002EB'EF     DF  0171                    PUSHAL  ERR_MESSAGE
                          00     DD  0177                    PUSHL   #0
                 00000000'EF     DD  0179                    PUSHL   KA_UNIT_NO
                          01     DD  017F                    PUSHL   #$ER
                                 0181       ;;; TEST 1, SUBTEST 2, ERROR 1
              000100D0 9F   05   FB  0181                    CALLS   #$$M, @#DS$ERRHARD
                                 0188   366 3$:       $DS_CKLOOP      NEXT1           ; scope loop at NEXT1
           00010040 9F   B0 AF   FA  0188                    CALLG   NEXT1, @#DS$CKLOOP
              57   000001A3'EF   9E  0190   367 SCOPE1: MOVAB   RENTRY_4,R7          ; load R7 with address of rentry point
                   00000004'EF   94  0197   368         CLRB    B_FLAGS             ; initialize error flag
                       55   63   D0  019D   369         MOVL    (R3),R5             ; load address of IPR to access
                       56   55   DB  01A0   370         MFPR    R5,R6               ; access IPR (should not cause fault)
                                 01A3   371 RENTRY_4:
                                 01A3   372         $DS_BREAK                       ; check for ^C
              00010058 9F   6E   FA  01A3                    CALLG   (SP), @#DS$BREAK
           1D 00000004'EF   01   E1  01AA   373         BBC     #V_FAULT,B_FLAGS,4$ ; br = exception rountine executed
                                 01B2   374         $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_F
                 00000187'8F     DD  01B2                    PUSHL   #T_FAULT_ERR_F
                 000002EB'EF     DF  01B8                    PUSHAL  ERR_MESSAGE
                          00     DD  01BE                    PUSHL   #0
                 00000000'EF     DD  01C0                    PUSHL   KA_UNIT_NO
                          02     DD  01C6                    PUSHL   #$ER
                                 01C8       ;;; TEST 1, SUBTEST 2, ERROR 2
              000100D0 9F   05   FB  01C8                    CALLS   #$$M, @#DS$ERRHARD
                                 01CF   375 4$:       $DS_CKLOOP      SCOPE1          ; scope loop at SCOPE1
           00010040 9F   BE AF   FA  01CF                    CALLG   SCOPE1, @#DS$CKLOOP
                          83     D5  01D7   376 5$:       TSTL    (R3)+               ; advance pointer
              63   FFFFFFFF 8F   D1  01D9   377         CMPL    #-1,(R3)            ; check for table terminator
                          03     13  01E0   378         BEQL    6$                  ; br = finished with all addresses
                       FF56      31  01E2   379         BRW     NEXT1               ; go do next address
                                 01E5   380 6$:       $DS_CLRVEC_S    VECTOR=#SCB$L_ROPRAND
                          18     DD  01E5                    PUSHL   #SCB$L_ROPRAND
              00010168 9F   01   FB  01E7                    CALLS   #1, @#DS$CLRVEC
                                 01EE   381         $DS_ENDSUB
                                 01EE       T1_S2_X:
        00010038 9F   0000000C'EF   FA  01EE                    CALLG   $S$, @#DS$ENDSUB
                                 01F9   382
                                 01F9   383         $DS_PAGE
```

```
                                01F9                    .SBTTL  SUBTEST 1.3:  Write Only Processor Registers
                                01F9    386 ;+
                                01F9    387 ; Subtest description:
                                01F9    388 ;
                                01F9    389 ;       This subtest checks processor-specific registers which are write only
                                01F9    390 ;       registers.  This subtest attempts a write and checks that a reserved
                                01F9    391 ;       operand fault does not occur. Then an attempt to read each read only
                                01F9    392 ;       register is made which should cause a reserved operand fault. If an
                                01F9    393 ;       error is encountered it is reported.
                                01F9    394 ;
                                01F9    395 ;
                                01F9    396 ; Subtest steps:
                                01F9    397 ;
                                01F9    398 ;       SUBTEST WRITE ONLY PROCESSOR REGISTERS
                                01F9    399 ;       :  Get reserved operand fault vector to handle exception
                                01F9    400 ;       :  REPEAT
                                01F9    401 ;       :  :  Clear error flags
                                01F9    402 ;       :  :  Attempt MTPR on register
                                01F9    403 ;       :  :  Check for errors
                                01F9    404 ;       :  :  IF any errors
                                01F9    405 ;       :  :  :  THEN
                                01F9    406 ;       :  :  :  Report error
                                01F9    407 ;       :  :  :  Exit test
                                01F9    408 ;       :  :  ENDIF
                                01F9    409 ;       :  :  Attempt MFPR on register
                                01F9    410 ;       :  :  Check for errors
                                01F9    411 ;       :  :  IF any errors
                                01F9    412 ;       :  :  :  THEN
                                01F9    413 ;       :  :  :  Report error
                                01F9    414 ;       :  :  :  Exit test
                                01F9    415 ;       :  :  ENDIF
                                01F9    416 ;       :  : UNTIL all write only registers are done
                                01F9    417 ;       :  ENDREPEAT
                                01F9    418 ;       ENDSUBTEST WRITE ONLY PROCESSOR REGISTERS
                                01F9    419 ;
                                01F9    420 ;
                                01F9    421 ;
                                01F9    422 ;-
                                01F9    423         $DS_BGNSUB
                                01F9        T1_S3::
00010030 9F   00000018'EF   FA  01F9                CALLG   $S$, @#DS$BGNSUB
         53   000000C9'EF   9E  0204    424         MOVAB   ADDRESS_TABLE2,R3       ; load R3 with address of table
                                020B    425         $DS_SETVEC_S    VECTOR=#SCB$L_ROPRAND   SRVADR=FAULT_SERVICE
                         00     DD  020B                PUSHL   #0
                00000308'EF     DF  020D                PUSHAL  FAULT_SERVICE
                         18     DD  0213                PUSHL   #SCB$L_ROPRAND
         00010160 9F    03     FB  0215                CALLS   #3, @#DS$SETVEC
         57   0000023D'EF   9E  021C    426 NEXT2:  MOVAB   RENTRY_5,R7             ; load R7 with address of rentry point
                00000004'EF   94  0223    427         CLRB    B_FLAGS             ; initialize error flag
                     55   63   D0  0229    428         MOVL    (R3),R5             ; load address of IPR to access
                00000000'EF   55   D0  022C    429         MOVL    R5,L_REG_ADD        ; load reg address for error
                     56   55   DB  0233    430         MFPR    R5,R6               ; access IPR (should cause fault)
                00000004'EF   01   88  0236    431         BISB2   #M_NO_FAULT,B_FLAGS     ; set no fault error flag
                                023D    432 RENTRY_5:
              1D 00000004'EF   00   E1  023D    433         BBC     #V_NO_FAULT,B_FLAGS,3$  ; br = MFPR did cause fault
                                0245    434         $DS_ERRHARD_S   UNIT=RA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_NO_FAULT_TO
                     00000121'8F   DD  0245                PUSHAL  #T_NO_FAULT_TO
```

```
                000002EB'EF   DF  024B                        PUSHAL  ERR_MESSAGE
                          00  DD  0251                        PUSHL   #0
                00000000'EF   DD  0253                        PUSHL   KA_UNIT_NO
                          01  DD  0259                        PUSHL   #$ER
                              025B        ;;; TEST 1, SUBTEST 3, ERROR 1
           000100D0 9F    05  FB  025B                        CALLS   #$$M, @#DS$ERRHARD
                              0262  435 3$:      $DS_CKLOOP      NEXT2           ; scope loop at NEXT2
        00010040 9F    B7 AF  FA  0262                        CALLG   NEXT2, @#DS$CKLOOP
           57   0000027D'EF   9E  026A  436 SCOPE2: MOVAB    RENTRY_6,R7     ; load R7 with address of rentry point
                00000004'EF   94  0271  437          CLRB    B_FLAGS         ; initialize error flag
                        55    63  D0  0277  438          MOVL    (R3),R5         ; load address of IPR to access
                        55    00  DA  027A  439          MTPR    #0,R5           ; access IPR (should not cause fault)
                              027D  440 RENTRY_6:
     1D 00000004'EF   01    E1  027D  441          BBC     #V_FAULT,B_FLAGS,4$     ; br = no exception occured
                              0285  442          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_FAULT_ERR_T
                0000025A'8F   DD  0285                        PUSHL   #T_FAULT_ERR_T
                000002EB'EF   DF  028B                        PUSHAL  ERR_MESSAGE
                          00  DD  0291                        PUSHL   #0
                00000000'EF   DD  0293                        PUSHL   KA_UNIT_NO
                          02  DD  0299                        PUSHL   #$ER
                              029B        ;;; TEST 1, SUBTEST 3, ERROR 2
           000100D0 9F    05  FB  029B                        CALLS   #$$M, @#DS$ERRHARD
                              02A2  443 4$:      $DS_CKLOOP      SCOPE2          ; scope loop at SCOPE2
        00010040 9F    C5 AF  FA  02A2                        CALLG   SCOPE2, @#DS$CKLOOP
                          83  D5  02AA  444 5$:      TSTL    (R3)+           ; advance pointer
           63   FFFFFFFF 8F   D1  02AC  445          CMPL    #-1,(R3)        ; check for table terminator
                          03  13  02B3  446          BEQL    6$              ; br = finished with all addresses
                      FF64    31  02B5  447          BRW     NEXT2           ; go do next address
                              02B8  448 6$:      $DS_CLRVEC_S    VECTOR=#SCB$L_ROPRAND
                          18  DD  02B8                        PUSHL   #SCB$L_ROPRAND
                00010168 9F   01  FB  02BA                        CALLS   #1, @#DS$CLRVEC
                              02C1  449          $DS_ENDSUB
                              02C1      T1_S3_X:
    00010038 9F    00000018'EF    FA  02C1                        CALLG   $S$, @#DS$ENDSUB
                              02CC  450
                              02CC  451          $DS_PAGE
                        50    01  D0  02CC                        MOVL    #1, R0          ; NORMAL EXIT
                              02CF      TEST_001_X::
                00010058 9F   6E  FA  02CF                        CALLG   (SP), @#DS$BREAK
                              04  02D6                        RET             ; RETURN TO TEST SEQUENCER
                                  00000000                    .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                          00000001  0000                      .LONG   $TN
                                    0004  454          .END
```

```
$$$                 = 00000018 R      06
$$ARGS              = 0000000A
$$E                 = 00000001
$$M                 = 00000005
$$N                 = 00000001
$$S                 = FFFFFFFF
$$T1                = 0000002C
$ENV                = 00000001   G
$ER                 = FFFFFFFF
$MO                 = 00000001   G
$S$                 = 00000018 R      06
$ST                 = 00000000
$TN                 = 00000001
ADDRESS_TABLE         00000005 R      02
ADDRESS_TABLE1        000000B1 R      02
ADDRESS_TABLE2        000000C9 R      02
ALL                 = 00000008   G
BASE_001              00000000 R      04
BIT...              = 00000011
B_FLAGS               00000004 R      02
CEP_FUNCTIONAL      = 00000000
CEP_REPAIR          = 00000001
DATA_001              00000020 R      04
DEFAULT             = 00000000   G
DS$ABORT              00010020
DS$ASKADR             00010090
DS$ASKDATA            00010080
DS$ASKLGCL            00010098
DS$ASKSTR             000100A0
DS$ASKVLD             00010088
DS$ATTACH             000101A8
DS$BGNSUB             00010030
DS$BRANCH             000100A8
DS$BREAK              00010058
DS$CANWAIT            00010070
DS$CHANNEL            00010180
DS$CKLOOP             00010040
DS$CLRVEC             00010168
DS$CNTRLC             00010078
DS$CVTREG             000100B0
DS$ENDPASS            00010010
DS$ENDSUB             00010038
DS$ERRDEV             000100C8
DS$ERRHARD            000100D0
DS$ERRPREP            00010108
DS$ERRSOFT            000100D8
DS$ERRSYS             000100C0
DS$ESCAPE             00010050
DS$FREEDBGSYM         000101B8
DS$GETBUF             00010120
DS$GETMEM             00010130
DS$GPHARD             00010018
DS$HELP               000101B0
DS$INITSCB            00010170
DS$INLOOP             00010048
DS$K_ERROR          = 00000002
DS$K_NORMAL         = 00000001
DS$K_SEVERE         = 00000004
DS$K_SUBSYS         = 00000066
DS$K_WARNING        = 00000000
DS$LOAD               00010198
DS$LOADPCS            000101C0
DS$MAPDBGBLOCK        00010118
DS$MMOFF              00010158
DS$MMON               00010150
DS$MOVPHY             00010148
DS$MOVVRT             00010140
DS$PARSE              000100B8
DS$PRINTB             000100E0
DS$PRINTF             000100F0
DS$PRINTS             000100F8
DS$PRINTSIG           00010100
DS$PRINTX             000100E8
DS$PROBE              000101A0
DS$RELBUF             00010128
DS$RELMEM             00010138
DS$SETIPL             00010178
DS$SETMAP             00010188
DS$SETPRIEXV          00010110
DS$SETVEC             00010160
DS$SHOCHAN            00010190
DS$SUMMARY            00010028
DS$WAITMS             00010060
DS$WAITUS             00010068
DS$_ARITH           = 006600D0
DS$_BADLINK         = 006600F0
DS$_BADTYPE         = 006600E8
DS$_CHME            = 006600A8
DS$_CHMK            = 006600E0
DS$_DEVNAME         = 00660108
DS$_ERROR           = 00660002
DS$_FHWE            = 00660068
DS$_FRAGBUF         = 00660080
DS$_ICBUSY          = 006600C8
DS$_ICERR           = 006600C0
DS$_IHWE            = 00660060
DS$_ILLCHAR         = 00660018
DS$_ILLPAGCNT       = 00660078
DS$_ILLUNIT         = 00660100
DS$_INSFMEM         = 00660050
DS$_IPL2HI          = 006600B8
DS$_IVADDR          = 00660040
DS$_IVVECT          = 00660038
DS$_KRNLSTK         = 00660090
DS$_LOGIC           = 00660070
DS$_MCHK            = 00660088
DS$_MMOFF           = 00660058
DS$_NEEDUNIT        = 006600F8
DS$_NOPCS           = 00660110
DS$_NORMAL          = 00660001
DS$_NOSUPPORT       = 006600B1
DS$_NOTDON          = 00660030
DS$_NOTIMP          = 006600B0
DS$_NULLSTR         = 00660010
DS$_OVERFLOW        = 00660008
DS$_POWER           = 00660098
DS$_PROGERR         = 00660020
DS$_SEVERE          = 00660004
DS$_TRANSL          = 006600A0
DS$_TRUNCATE        = 00660028
DS$_UNEXPINT        = 006600D8
DS$_VASFULL         = 00660048
DS$_WARNING         = 00660000
ENV$M_DOMAIN        = 00000002
ENV$M_LEVEL         = 00000001
ENV$M_SUPER         = 000003FC
ENV$S_DOMAIN        = 00000001
ENV$S_LEVEL         = 00000001
ENV$S_SUPER         = 00000008
ENV$V_DOMAIN        = 00000001
ENV$V_LEVEL         = 00000000
ENV$V_SUPER         = 00000002
ENV$_CPU            = 00000000
ENV$_FUNCTIONAL     = 00000000
ENV$_REPAIR         = 00000001
ENV$_SUPER          = 00000001
ENV$_SYSTEM         = 00000001
ERR$_MSGADR         = 0000000C
ERR$_NARGS          = 0000000A
ERR$_NUM            = 00000004
ERR$_P1             = 00000014
ERR$_P2             = 00000018
ERR$_P3             = 0000001C
ERR$_P4             = 00000020
ERR$_P5             = 00000024
ERR$_P6             = 00000028
ERR$_POINTER        = 00000010
ERR$_UNIT           = 00000008
ERR_MESSAGE           000002EB R      02
FAULT_SERVICE         00000308 R      02
HALT                = 00000004   G
HP$A_DEPENDENT        00000032
HP$A_DEVICE           00000018
HP$A_DVA              0000001C
HP$A_LINK             00000020
HP$B_DRIVE            0000000B
HP$B_FLAGS            0000000A
HP$M_ALLOC          = 00000001
HP$M_WASALL         = 00000002
HP$Q_DEVICE           00000000
HP$T_DEVICE           0000000C
HP$T_TYPE             00000026
HP$V_ALLOC          = 00000000
HP$V_WASALL         = 00000001
HP$W_SIZE             00000008
HP$W_VECTOR           00000024
IPR                 = 00000006   G
KA$B_ACC_TYPE         00000042
KA$B_SID_TYPE         0000003D
KA$K_LEN              00000043
KA$L_FLAGS            00000032
```

```
KA$L_SID              0000003A
KA$M_CHAR           = 00000100
KA$M_CRC            = 00000010
KA$M_D_FLOAT        = 00000002
KA$M_EDIT           = 00000080
KA$M_F_FLOAT        = 00000001
KA$M_G_FLOAT        = 00000004
KA$M_H_FLOAT        = 00000008
KA$M_PACKED         = 00000020
KA$M_PACK_MUL       = 00000040
KA$M_TODR           = 00010000
KA$V_CHAR           = 00000008
KA$V_CRC            = 00000004
KA$V_D_FLOAT        = 00000001
KA$V_EDIT           = 00000007
KA$V_F_FLOAT        = 00000000
KA$V_G_FLOAT        = 00000002
KA$V_H_FLOAT        = 00000003
KA$V_PACKED         = 00000005
KA$V_PACK_MUL       = 00000006
KA$V_TODR           = 00000010
KA$W_LATENCY          0000003E
KA$W_PROGRESS         00000040
KA$W_RESERVED         00000038
KA$W_WCS              00000036
KA_PTABLE             ********    X   04
KA_UNIT_NO            ********    X   04
LDPCTX              = 0000000B  G
L_REG_ADD             00000000 R      02
MANUAL              = 00000001  G
MCHK                = 00000005  G
MEM                 = 00000009  G
MSG_001               00000002 R      04
M_FAULT             = 00000002
M_NO_FAULT          = 00000001
NEXT                  00000061 R      04
NEXT1                 0000013B R      04
NEXT2                 0000021C R      04
POWER               = 00000003  G
RENTRY_1              00000082 R      04
RENTRY_2              000000C9 R      04
RENTRY_3              0000015C R      04
RENTRY_4              000001A3 R      04
RENTRY_5              0000023D R      04
RENTRY_6              0000027D R      04
SCB$L_ACCESS          00000020
SCB$L_ARITH           00000034
SCB$L_BREAK           0000002C
SCB$L_CHME            00000044
SCB$L_CHMK            00000040
SCB$L_CHMS            00000048
SCB$L_CHMU            0000004C
SCB$L_COMPAT          00000030
SCB$L_KNLSTK          00000008
SCB$L_MACHCHK         00000004
SCB$L_OPCCUS          00000014
SCB$L_OPCDEC          00000010
SCB$L_POWER           0000000C
SCB$L_RADRMOD         0000001C
SCB$L_ROPRAND         00000018
SCB$L_RXDB            000000F8
SCB$L_SFTLVL1         00000084
SCB$L_SFTLVL10        000000A8
SCB$L_SFTLVL11        000000AC
SCB$L_SFTLVL12        000000B0
SCB$L_SFTLVL13        000000B4
SCB$L_SFTLVL14        000000B8
SCB$L_SFTLVL15        000000BC
SCB$L_SFTLVL2         00000088
SCB$L_SFTLVL3         0000008C
SCB$L_SFTLVL4         00000090
SCB$L_SFTLVL5         00000094
SCB$L_SFTLVL6         00000098
SCB$L_SFTLVL7         0000009C
SCB$L_SFTLVL8         000000A0
SCB$L_SFTLVL9         000000A4
SCB$L_TBIT            00000028
SCB$L_TIMER           000000C0
SCB$L_TRANSL          00000024
SCB$L_TXDB            000000FC
SCB$L_ZERO            00000000
SCOPE                 000000AF R      04
SCOPE1                00000190 R      04
SCOPE2                0000026A R      04
SEP_FUNCTIONAI      = 00000002
SEP_REPAIR          = 00000003
SIZ...              = 00000001
SYS$ALLOC             00010238
SYS$ASCTIM            00010248
SYS$ASSIGN            00010250
SYS$BINTIM            00010258
SYS$CANCEL            00010260
SYS$CANTIM            00010268
SYS$CLOSE             000105B8
SYS$CLREF             00010298
SYS$CONNECT           000105C0
SYS$DALLOC            000102D8
SYS$DASSGN            000102E0
SYS$DISCONNECT        000105D0
SYS$FAO               00010350
SYS$FAOL              00010358
SYS$GET               00010580
SYS$GETCHN            000104C8
SYS$GETTIM            00010378
SYS$LKWSET            000103A0
SYS$NUMTIM            000103B8
SYS$OPEN              00010608
SYS$QIO               000103C8
SYS$QIOW              00010200
SYS$READ              00010590
SYS$READEF            000103D0
SYS$SETAST            000103F8
SYS$SETEF             00010400
SYS$SETIMR            00010420
SYS$SETPRI            00010428
SYS$SETPRT            00010430
SYS$SETRWM            00010438
SYS$SETSWM            00010448
SYS$ULKPAG            00010460
SYS$ULWSET            00010468
SYS$UNWIND            00010470
SYS$WAITFR            00010478
SYS$WFLAND            00010488
SYS$WFLOR             00010490
T1_S1                 0000003E RG     04
T1_S1_X               0000010D R      04
T1_S2                 00000118 RG     04
T1_S2_X               000001EE R      04
T1_S3                 000001F9 RG     04
T1_S3_X               000002C1 R      04
TEST_001              00000024 RG     04
TEST_001_X            000002CF RG     04
TU58                = 00000002  G
T_FAULT_ERR_F         00000187 R      02
T_FAULT_ERR_T         0000025A R      02
T_NO_FAULT_FROM       000001F2 R      02
T_NO_FAULT_TO         00000121 R      02
T_NO_KA730            000000E9 R      02
T_REG_ADD             000002C5 R      02
UBI                 = 0000000A  G
V_FAULT             = 00000001
V_NO_FAULT          = 00000000
WCS                 = 00000007  G
```

+-----------------+
' Psect synopsis !
+-----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXA	00000311 (785.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_001	000002D7 (727.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000018 (24.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	00000024 (36.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
$TSTCNT	00000004 (4.)	07 (7.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000018-R	423 (9)	283 (7) 353 (8) 423 (9)
$$ARGS	=0000000A	84 (2)	84 (2)
$$E	=00000001	423 (9)	311 (7) 381 (8) 449 (9)
$$M	=00000005	442 (9)	237 (6) 294 (7) 303 (7) 365 (8) 374 (8)
			434 (9) 442 (9)
$$N	=00000001	442 (9)	#-206 (4) 207 (4) #-241 (6) 294 (7) 303 (7)
			365 (8) 374 (8) 434 (9) 442 (9) 85 (2)
$$S	=FFFFFFFF	452 (9)	225 (6) 237 (6) 247 (7) 283 (7) 315 (8)
			353 (8) 385 (9) 423 (9)
$$T1	=0000002C	84 (2)	84 (2)
$ENV	=00000001	63 (2)	
$ER	=FFFFFFFF	452 (9)	283 (7) #-294 (7) #-303 (7) 353 (8) #-365 (8)
			#-374 (8) 423 (9) #-434 (9) #-442 (9)
$MO	=00000001	453 (9)	453 (9)
$SS	=00000018-R	423 (9)	283 (7) 311 (7) 353 (8) 381 (8) 423 (9)
			449 (9)
$ST	=00000000	452 (9)	237 (6) 247 (7) 283 (7) 294 (7) 295 (7)
			303 (7) 304 (7) 311 (7) 315 (8) 353 (8)
			365 (8) 366 (8) 374 (8) 375 (8) 381 (8)
			385 (9) 423 (9) 434 (9) 435 (9) 442 (9)
			443 (9) 449 (9) 452 (9)
$TN	=00000001	453 (9)	225 (6) 237 (6) 242 (6) 247 (7) 294 (7)
			303 (7) 315 (8) 365 (8) 374 (8) 385 (9)
			434 (9) 442 (9) 452 (9) 453 (9)
ADDRESS_TABLE	00000005-R	109 (3)	284 (7)
ADDRESS_TABLE1	000000B1-R	157 (3)	354 (8)
ADDRESS_TABLE2	000000C9-R	168 (3)	424 (9)
ALL	=00000008	85 (2)	237 (6)
BASE_001	00000000-R	225 (6)	237 (6)
BIT...	=00000011	83 (2)	63 (2) 80 (2) 82 (2) 83 (2)
B_FLAGS	00000004-R	92 (2)	#-220 (5) #-287 (7) #-291 (7) 293 (7) #-297 (7)
			#-300 (7) 302 (7) #-357 (8) #-361 (8) 364 (8)
			#-368 (8) 373 (8) #-427 (9) #-431 (9) 433 (9)
			#-437 (9) 441 (9)
CEP_FUNCTIONAL	=00000000	63 (2)	
CEP_REPAIR	=00000001	63 (2)	63 (2)
DATA_001	00000020-R	237 (6)	237 (6)
DEFAULT	=00000000	85 (2)	237 (6)
DS$ABORT	00010020	81 (2)	
DS$ASKADR	00010090	81 (2)	
DS$ASKDATA	00010080	81 (2)	
DS$ASKLGCL	00010098	81 (2)	
DS$ASKSTR	000100A0	81 (2)	
DS$ASKVLD	00010088	81 (2)	
DS$ATTACH	000101A8	81 (2)	
DS$BGNSUB	00010030	81 (2)	283 (7) 353 (8) 423 (9)
DS$BRANCH	000100A8	81 (2)	
DS$BREAK	00010058	81 (2)	363 (8) 372 (8) 452 (9)
DS$CANWAIT	00010070	81 (2)	
DS$CHANNEL	00010180	81 (2)	

```
DS$CKLOOP          00010040      81      (2)      295     (7)      304     (7)      366     (8)      375     (8)      435     (9)
                                                  443     (9)
DS$CLRVEC          00010168      81      (2)      309     (7)      380     (8)      448     (9)
DS$CNTRLC          00010078      81      (2)
DS$CVTREG          000100B0      81      (2)
DS$ENDPASS         00010010      81      (2)
DS$ENDSUB          00010038      81      (2)      311     (7)      381     (8)      449     (9)
DS$ERRDEV          000100C8      81      (2)
DS$ERRHARD         000100D0      81      (2)      294     (7)      303     (7)      365     (8)      374     (8)      434     (9)
                                                  442     (9)
DS$ERRPREP         00010108      81      (2)
DS$ERRSOFT         000100D8      81      (2)
DS$ERRSYS          000100C0      81      (2)
DS$ESCAPE          00010050      81      (2)
DS$FREEDBGSYM      00010188      81      (2)
DS$GETBUF          00010120      81      (2)
DS$GETMEM          00010130      81      (2)
DS$GPHARD          00010018      81      (2)
DS$HELP            000101B0      81      (2)
DS$INITSCB         00010170      81      (2)
DS$INLOOP          00010048      81      (2)
DS$K_ERROR        =00000002      80      (2)
DS$K_NORMAL       =00000001      80      (2)
DS$K_SEVERE       =00000004      80      (2)
DS$K_SUBSYS       =00000066      80      (2)      80      (2)
DS$K_WARNING      =00000000      80      (2)
DS$LOAD            00010198      81      (2)
DS$LOADPCS         000101C0      81      (2)
DS$MAPDBGBLOCK     00010118      81      (2)
DS$MMOFF           00010158      81      (2)
DS$MMON            00010150      81      (2)
DS$MOVPHY          00010148      81      (2)
DS$MOVVRT          00010140      81      (2)
DS$PARSE           000100B8      81      (2)
DS$PRINTB          000100E0      81      (2)      206     (4)      207     (4)
DS$PRINTF          000100F0      81      (2)      241     (6)
DS$PRINTS          000100F8      81      (2)
DS$PRINTSIG        00010100      81      (2)
DS$PRINTX          000100E8      81      (2)
DS$PROBE           000101A0      81      (2)
DS$RELBUF          00010128      81      (2)
DS$RELMEM          00010138      81      (2)
DS$SETIPL          00010178      81      (2)
DS$SETMAP          00010188      81      (2)
DS$SETPRIEXV       00010110      81      (2)
DS$SETVEC          00010160      81      (2)      285     (7)      355     (8)      425     (9)
DS$SHOCHAN         00010190      81      (2)
DS$SUMMARY         00010028      81      (2)
DS$WAITMS          00010060      81      (2)
DS$WAITUS          00010068      81      (2)
DS$_ARITH         =006600D0      80      (2)
DS$_BADLINK       =006600F0      80      (2)
DS$_BADTYPE       =006600E8      80      (2)
DS$_CHME          =006600A8      80      (2)
DS$_CHMK          =006600E0      80      (2)
DS$_DEVNAME       =00660108      80      (2)
DS$_ERROR         =00660002      80      (2)
```

```
DS$_FHWE          =00660068      80      (2)
DS$_FRAGBUF       =00660080      80      (2)
DS$_ICBUSY        =006600C8      80      (2)
DS$_ICERR         =006600C0      80      (2)
DS$_IHWE          =00660060      80      (2)
DS$_ILLCHAR       =00660018      80      (2)
DS$_ILLPAGCNT     =00660078      80      (2)
DS$_ILLUNIT       =00660100      80      (2)
DS$_INSFMEM       =00660050      80      (2)
DS$_IPL2HI        =006600B8      80      (2)
DS$_IVADDR        =00660040      80      (2)
DS$_IVVECT        =00660038      80      (2)
DS$_KRNLSTK       =00660090      80      (2)
DS$_LOGIC         =00660070      80      (2)
DS$_MCHK          =00660088      80      (2)
DS$_MMOFF         =00660058      80      (2)
DS$_NEEDUNIT      =006600F8      80      (2)
DS$_NOPCS         =00660110      80      (2)
DS$_NORMAL        =00660001      80      (2)
DS$_NOSUPPORT     =006600B1      80      (2)
DS$_NOTDON        =00660030      80      (2)
DS$_NOTIMP        =006600B0      80      (2)       80      (2)
DS$_NULLSTR       =00660010      80      (2)
DS$_OVERFLOW      =00660008      80      (2)
DS$_POWER         =00660098      80      (2)
DS$_PROGERR       =00660020      80      (2)
DS$_SEVERE        =00660004      80      (2)
DS$_TRANSL        =006600A0      80      (2)
DS$_TRUNCATE      =00660028      80      (2)
DS$_UNEXPINT      =006600D8      80      (2)
DS$_VASFULL       =00660048      80      (2)
DS$_WARNING       =00660000      80      (2)       80      (2)
ENV$M_DOMAIN      =00000002      63      (2)
ENV$M_LEVEL       =00000001      63      (2)
ENV$M_SUPER       =000003FC      63      (2)
ENV$S_DOMAIN      =00000001      63      (2)
ENV$S_LEVEL       =00000001      63      (2)
ENV$S_SUPER       =00000008      63      (2)
ENV$V_DOMAIN      =00000001      63      (2)       63      (2)
ENV$V_LEVEL       =00000000      63      (2)       63      (2)
ENV$V_SUPER       =00000002      63      (2)
ENV$_CPU          =00000000      63      (2)       63      (2)
ENV$_FUNCTIONAL   =00000000      63      (2)       63      (2)
ENV$_REPAIR       =00000001      63      (2)       63      (2)
ENV$_SUPER        =00000001      63      (2)
ENV$_SYSTEM       =00000001      63      (2)       63      (2)
ERR$_MSGADR       =0000000C      84      (2)
ERR$_NARGS        =0000000A      84      (2)
ERR$_NUM          =00000004      84      (2)
ERR$_P1           =00000014      84      (2)       206     (4)
ERR$_P2           =00000018      84      (2)
ERR$_P3           =0000001C      84      (2)
ERR$_P4           =00000020      84      (2)
ERR$_P5           =00000024      84      (2)
ERR$_P6           =00000028      84      (2)
ERR$_POINTER      =00000010      84      (2)
ERR$_UNIT         =00000008      84      (2)
```

```
ERR_MESSAGE     000002EB-R  204  (4)   294  (7)   303  (7)   365  (8)   374  (8)   434  (9)
                                       442  (9)
FAULT_SERVICE   00000308-R  218  (5)   285  (7)   355  (8)   425  (9)
HALT            =00000004    85  (2)
HP$A_DEPENDENT   00000032    82  (2)    83  (2)
HP$A_DEVICE      00000018    82  (2)
HP$A_DVA         0000001C    82  (2)
HP$A_LINK        00000020    82  (2)
HP$B_DRIVE       0000000B    82  (2)
HP$B_FLAGS       0000000A    82  (2)
HP$M_ALLOC      =00000001    82  (2)
HP$M_WASALL     =00000002    82  (2)
HP$Q_DEVICE      00000000    82  (2)
HP$T_DEVICE      0000000C    82  (2)
HP$T_TYPE        00000026    82  (2)
HP$V_ALLOC      =00000000    82  (2)
HP$V_WASALL     =00000001    82  (2)
HP$W_SIZE        00000008    82  (2)
HP$W_VECTOR      00000024    82  (2)
IPR             =00000006    85  (2)   237  (6)
KA$B_ACC_TYPE    00000042    83  (2)
KA$B_SID_TYPE    0000003D    83  (2)
KA$K_LEN         00000043    83  (2)
KA$L_FLAGS       00000032    83  (2)
KA$L_SID         0000003A    83  (2)
KA$M_CHAR       =00000100    83  (2)
KA$M_CRC        =00000010    83  (2)
KA$M_D_FLOAT    =00000002    83  (2)
KA$M_EDIT       =00000080    83  (2)
KA$M_F_FLOAT    =00000001    83  (2)
KA$M_G_FLOAT    =00000004    83  (2)
KA$M_H_FLOAT    =00000008    83  (2)
KA$M_PACKED     =00000020    83  (2)
KA$M_PACK_MUL   =00000040    83  (2)
KA$M_TODR       =00010000    83  (2)
KA$V_CHAR       =00000008    83  (2)
KA$V_CRC        =00000004    83  (2)
KA$V_D_FLOAT    =00000001    83  (2)
KA$V_EDIT       =00000007    83  (2)
KA$V_F_FLOAT    =00000000    83  (2)
KA$V_G_FLOAT    =00000002    83  (2)
KA$V_H_FLOAT    =00000003    83  (2)
KA$V_PACKED     =00000005    83  (2)
KA$V_PACK_MUL   =00000006    83  (2)
KA$V_TODR       =00000010    83  (2)
KA$W_LATENCY     0000003E    83  (2)
KA$W_PROGRESS    00000040    83  (2)
KA$W_RESERVED    00000038    83  (2)
KA$W_WCS         00000036    83  (2)
KA_PTABLE        00000000-XR        #-239  (6)
KA_UNIT_NO       00000000-XR        #-294  (7)  #-303  (7)  #-365  (8)  #-374  (8)  #-434  (9)
                                    #-442  (9)
LDPCTX          =0000000B    85  (2)
L_REG_ADD        00000000-R  91  (2)  #-207  (4)  #-289  (7)  #-359  (8)  #-429  (9)
MANUAL          =00000001    85  (2)
MCHK            =00000005    85  (2)
MEM             =00000009    85  (2)
```

```
MSG_001          00000002-R  225  (6)    237  (6)
M_FAULT          =00000002    97  (2)  #-220  (5)
M_NO_FAULT       =00000001    95  (2)  #-291  (7)  #-300  (7)  #-361  (8)  #-431  (9)
NEXT              00000061-R 286  (7)    295  (7)  #-308  (7)
NEXT1             0000013B-R 356  (8)    366  (8)  #-379  (8)
NEXT2             0000021C-R 426  (9)    435  (9)  #-447  (9)
POWER            =00000003    85  (2)
RENTRY_1          00000082-R 292  (7)    286  (7)
RENTRY_2          000000C9-R 301  (7)    296  (7)
RENTRY_3          0000015C-R 362  (8)    356  (8)
RENTRY_4          000001A3-R 371  (8)    367  (8)
RENTRY_5          0000023D-R 432  (9)    426  (9)
RENTRY_6          0000027D-R 440  (9)    436  (9)
SCB$L_ACCESS      00000020    79  (2)
SCB$L_ARITH       00000034    79  (2)
SCB$L_BREAK       0000002C    79  (2)
SCB$L_CHME        00000044    79  (2)
SCB$L_CHMK        00000040    79  (2)
SCB$L_CHMS        00000048    79  (2)
SCB$L_CHMU        0000004C    79  (2)
SCB$L_COMPAT      00000030    79  (2)
SCB$L_KNLSTK      00000008    79  (2)
SCB$L_MACHCHK     00000004    79  (2)
SCB$L_OPCCUS      00000014    79  (2)
SCB$L_OPCDEC      00000010    79  (2)
SCB$L_POWER       0000000C    79  (2)
SCB$L_RADRMOD     0000001C    79  (2)
SCB$L_ROPRAND     00000018    79  (2)  #-285  (7)  #-309  (7)  #-355  (8)  #-380  (8)  #-425  (9)
                                       #-448  (9)
SCB$L_RXDB        000000F8    79  (2)
SCB$L_SFTLVL1     00000084    79  (2)
SCB$L_SFTLVL10    000000A8    79  (2)
SCB$L_SFTLVL11    000000AC    79  (2)
SCB$L_SFTLVL12    000000B0    79  (2)
SCB$L_SFTLVL13    000000B4    79  (2)
SCB$L_SFTLVL14    000000B8    79  (2)
SCB$L_SFTLVL15    000000BC    79  (2)
SCB$L_SFTLVL2     00000088    79  (2)
SCB$L_SFTLVL3     0000008C    79  (2)
SCB$L_SFTLVL4     00000090    79  (2)
SCB$L_SFTLVL5     00000094    79  (2)
SCB$L_SFTLVL6     00000098    79  (2)
SCB$L_SFTLVL7     0000009C    79  (2)
SCB$L_SFTLVL8     000000A0    79  (2)
SCB$L_SFTLVL9     000000A4    79  (2)
SCB$L_TBIT        00000028    79  (2)
SCB$L_TIMER       000000C0    79  (2)
SCB$L_TRANSL      00000024    79  (2)
SCB$L_TXDB        000000FC    79  (2)
SCB$L_ZERO        00000000    79  (2)
SCOPE             000000AF-R 296  (7)    304  (7)
SCOPE1            00000190-R 367  (8)    375  (8)
SCOPE2            0000026A-R 436  (9)    443  (9)
SEP_FUNCTIONAL   =00000002    63  (2)
SEP_REPAIR       =00000003    63  (2)
SIZ...           =00000001    83  (2)    63  (2)    82  (2)    83  (2)
SYS$ALLOC         00010238    81  (2)
```

```
SYS$ASCTIM        00010248      81      (2)
SYS$ASSIGN        00010250      81      (2)
SYS$BINTIM        00010258      81      (2)
SYS$CANCEL        00010260      81      (2)
SYS$CANTIM        00010268      81      (2)
SYS$CLOSE         000105B8      81      (2)
SYS$CLREF         00010298      81      (2)
SYS$CONNECT       000105C0      81      (2)
SYS$DALLOC        000102D8      81      (2)
SYS$DASSGN        000102E0      81      (2)
SYS$DISCONNECT    000105D0      81      (2)
SYS$FAO           00010350      81      (2)
SYS$FAOL          00010358      81      (2)
SYS$GET           00010580      81      (2)
SYS$GETCHN        000104C8      81      (2)
SYS$GETTIM        00010378      81      (2)
SYS$LKWSET        000103A0      81      (2)
SYS$NUMTIM        000103B8      81      (2)
SYS$OPEN          00010608      81      (2)
SYS$QIO           000103C8      81      (2)
SYS$QIOW          00010200      81      (2)
SYS$READ          00010590      81      (2)
SYS$READEF        000103D0      81      (2)
SYS$SETAST        000103F8      81      (2)
SYS$SETEF         00010400      81      (2)
SYS$SETIMR        00010420      81      (2)
SYS$SETPRI        00010428      81      (2)
SYS$SETPRT        00010430      81      (2)
SYS$SETRWM        00010438      81      (2)
SYS$SETSWM        00010448      81      (2)
SYS$ULKPAG        00010460      81      (2)
SYS$ULWSET        00010468      81      (2)
SYS$UNWIND        00010470      81      (2)
SYS$WAITFR        00010478      81      (2)
SYS$WFLAND        00010488      81      (2)
SYS$WFLOR         00010490      81      (2)
T1_S1             0000003E-R    283     (7)
T1_S1_X           0000010D-R    311     (7)
T1_S2             00000118-R    353     (8)
T1_S2_X           000001EE-R    381     (8)
T1_S3             000001F9-R    423     (9)
T1_S3_X           000002C1-R    449     (9)
TEST_001          00000024-R    237     (6)     237     (6)
TEST_001_X        000002CF-R    452     (9)     #-242   (6)
TU58            =00000002       85      (2)
T_FAULT_ERR_F     00000187-R    188     (4)     #-374   (8)
T_FAULT_ERR_T     0000025A-R    196     (4)     #-442   (9)
T_NO_FAULT_FROM   000001F2-R    192     (4)     #-303   (7)
T_NO_FAULT_TO     00000121-R    184     (4)     #-294   (7)     #-365   (8)     #-434   (9)
T_NO_KA730        000000E9-R    181     (4)     241     (6)
T_REG_ADD         000002C5-R    200     (4)     207     (4)
UBI             =0000000A       85      (2)
V_FAULT         =00000001       96      (2)     #-373   (8)     #-441   (9)
V_NO_FAULT      =00000000       94      (2)     #-293   (7)     #-302   (7)     #-364   (8)     #-433   (9)
WCS             =00000007       85      (2)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_BSUBT	1	283 (7)	283 (7) 353 (8) 423 (9)
$D1_BTEST	1	237 (6)	237 (6)
$D1_ERR	1	294 (7)	294 (7) 303 (7) 365 (8) 374 (8) 434 (9) 442 (9)
$D1_ERR_S	1	294 (7)	294 (7) 303 (7) 365 (8) 374 (8) 434 (9) 442 (9)
$D1_ESUBT	1	311 (7)	311 (7) 381 (8) 449 (9)
$D1_ETEST	1	452 (9)	452 (9)
$D1_EXTEST	1	242 (6)	242 (6)
$D1_PRINT_S	1	206 (4)	206 (4) 207 (4) 241 (6)
$D1_SBTTL	2	225 (6)	225 (6) 247 (7) 315 (8) 385 (9)
$D2_BDATA	1	237 (6)	237 (6)
$D2_BTEST	1	237 (6)	237 (6)
$D2_SBTTL	1	225 (6)	225 (6)
$DEF	1	83 (2)	79 (2) 81 (2) 82 (2) 83 (2)
$DEFEND	1	79 (2)	79 (2) 81 (2) 82 (2) 83 (2)
$DEFINI	1	79 (2)	79 (2) 81 (2) 82 (2) 83 (2)
$DS_BGNMESSAGE	1	205 (4)	205 (4)
$DS_BGNMOD	1	63 (2)	63 (2)
$DS_BGNSUB	1	283 (7)	283 (7) 353 (8) 423 (9)
$DS_BGNTEST	1	237 (6)	237 (6)
$DS_BREAK	1	363 (8)	363 (8) 372 (8) 452 (9)
$DS_CKLOOP	1	295 (7)	295 (7) 304 (7) 366 (8) 375 (8) 435 (9) 443 (9)
$DS_CLRVEC_S	1	309 (7)	309 (7) 380 (8) 448 (9)
$DS_DSDEF	2	80 (2)	80 (2)
$DS_DSSDEF	1	81 (2)	81 (2)
$DS_ENDDATA	1	237 (6)	237 (6)
$DS_ENDMESSAGE	1	208 (4)	208 (4)
$DS_ENDMOD	1	453 (9)	453 (9)
$DS_ENDSUB	1	311 (7)	311 (7) 381 (8) 449 (9)
$DS_ENDTEST	1	452 (9)	452 (9)
$DS_ENVDEF	2	63 (2)	63 (2)
$DS_ERRDEF	1	84 (2)	84 (2)
$DS_ERRHARD_S	1	294 (7)	294 (7) 303 (7) 365 (8) 374 (8) 434 (9) 442 (9)
$DS_EXIT	1	242 (6)	242 (6)
$DS_HPODEF	2	82 (2)	82 (2)
$DS_PAGE	1	223 (5)	223 (5) 245 (6) 313 (7) 383 (8) 451 (9)
$DS_PRINTB_S	1	206 (4)	206 (4) 207 (4)
$DS_PRINTF_S	1	241 (6)	241 (6)
$DS_SBTTL	1	225 (6)	225 (6) 247 (7) 315 (8) 385 (9)
$DS_SCBDEF	1	79 (2)	79 (2)
$DS_SECDEF	1	85 (2)	85 (2)
$DS_SETVEC_S	1	285 (7)	285 (7) 355 (8) 425 (9)
$EQU	1	83 (2)	63 (2) 80 (2)
$EQULS1	1	80 (2)	63 (2) 80 (2)
$EQULST	1	63 (2)	63 (2) 80 (2)
$GBLINI	2	63 (2)	63 (2) 79 (2) 80 (2) 81 (2) 82 (2) 83 (2)

```
$HP_DEFDEF          1          82     (2)
$KADEF              1          83     (2)
$OFFDEF             1          84     (2)       84     (2)
$PUSHADR            1         294     (7)      294     (7)      303     (7)      365     (8)      374     (8)      434     (9)
                                                442     (9)
$VIELD              1          63     (2)       63     (2)       82     (2)       83     (2)
$VIELD1             1          83     (2)       63     (2)       82     (2)       83     (2)
ENKAX_SECDEF        1          85     (2)       85     (2)
KA_DEF              2          83     (2)       83     (2)
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	21	00:00:00.06	00:00:00.15
Command processing	29	00:00:00.20	00:00:00.65
Pass 1	752	00:00:11.08	00:00:32.18
Symbol table sort	0	00:00:00.43	00:00:01.10
Pass 2	169	00:00:02.19	00:00:04.89
Symbol table output	29	00:00:00.19	00:00:00.64
Psect synopsis output	6	00:00:00.04	00:00:00.13
Cross-reference output	96	00:00:01.43	00:00:04.98
Assembler run totals	1106	00:00:15.64	00:00:44.77

Performance indicators

```
The working set limit was 900 pages.
85368 bytes (167 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 314 non-local and 13 local symbols.
454 source lines were read in Pass 1, producing 25 object records in Pass 2.
93 pages of virtual memory were used to define 48 macros.
```

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	2
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	38
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	7
TOTALS (all libraries)	47

Macro library statistics

```
692 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXA
```

Table of contents

```
0000     1          .SBTTL  ENKAXB: ARCHITECTURE 'UNPREDICTABLE" AND 'UNDEFINED" OPERATIONS
0000     2          .TITLE  ENKAXB  VAX-11/730 Specific CPU Cluster Exerciser
0000     3          .IDENT  /1.2/
0000     4
0000     5 ;
0000     6 ; Copyright (C) 1980
0000     7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000     8 ;
0000     9 ; This software is furnished  under a license for use only on a single
0000    10 ; computer  system and  may be  copied only with  the inclusion of the
0000    11 ; above copyright notice.  This software, or any other copies thereof,
0000    12 ; may not be provided or  otherwise made available to any other person
0000    13 ; except for use on such system and to one who agrees to these license
0000    14 ; terms.  Title to and  ownership of the  software  shall at all times
0000    15 ; remain in DEC.
0000    16 ;
0000    17 ; The information in this software is subject to change without notice
0000    18 ; and should  not be construed  as a commitment  by Digital  Equipment
0000    19 ; Corporation.
0000    20 ;
0000    21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000    22 ; software on equipment which is not supplied by DEC.
0000    23 ;
0000    24
0000    25 ;++
0000    26 ; Facility:     VAX Architecture Diagnostic.
0000    27 ;
0000    28 ; Abstract:     This module was not implemented for any VAX processor and it was
0000    29 ;               agreed at the Nebula functional specification review that it
0000    30 ;               will not be implemented for the VAX 11/730. The amount of effort
0000    31 ;               required is excessive and the payback in terms of design
0000    32 ;               verification coverage is felt to be minimal.
0000    33 ;
0000    34 ;
0000    35 ; Environment:  VAX Diagnostic Supervisor.
0000    36 ;
0000    37 ; Author:       Rich Lewis      13-Dec-80       Version 1X
0000    38 ;
0000    39 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
0000    40 ;
0000    41 ;               Added quick flag on Test 7(TU58). If quick flag is set
0000    42 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
0000    43 ;               If event flag 3 is set, will override protection,
0000    44 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
0000    45 ;               is clear & tape is not scratch, will report an error.
0000    46 ;
0000    47 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
0000    48 ;
0000    49 ;
0000    50 ;
0000    51 ;--
```

```
                        53          .SBTTL  DECLARATIONS
                        54          .LIST   MEB
                        55          .NLIST  CND
                        56          .PSECT  ENKAXB, PAGE, NOWRT
                        57          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                        58          $DS_BGNMOD      CEP_REPAIR, IN=2
                                ;;;     Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                        59
                        60 ;
                        61 ; Include files:
                        62 ;
                        63
                        64          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                        65
                        66 ;
                        67 ; Macros:
                        68 ;
                        69
                        70
                        71 ;
                        72 ; Equated symbols:
                        73 ;
                        77
                        78          $DS_SCBDEF
                        79          $DS_DSDEF
                        80          $DS_DSSDEF
                        81          ENKAX_SECDEF
                        82
                        83 ;
                        84 ; Own storage:
                        85 ;
                        89
                        90          $DS_PAGE
```

```
                        0000                    .SBTTL  TEST 2:  UNPREDICTABLE Results
                    00000000                    .PSECT  TEST_002, PAGE, NOWRT
                        001C
                        001C     93 ; ++
                        001C     94 ; Test description:
                        001C     95 ;
                        001C     96 ;       This test checks most operations which are specified as producing
                        001C     97 ;       UNPREDICTABLE results.  The state of the processor is verified
                        001C     98 ;       after the operation to insure that no undefined side affects occurred.
                        001C     99 ;
                        001C    100 ;
                        001C    101 ;--
                        001C    102
                        001C    103         $DS_BGNTEST
                        001C        DATA_002:
                    00000000    001C                    .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                        0020        TEST_002::
                        0000    0020                    .WORD   ^M<>            ; ENTRY MASK
                        0022    104
                        0022    105         $DS_PAGE
```

```
                                0022                    .SBTTL  SUBTEST 2.1:  Writes to the Instruction Stream
                                0022    108 ; +
                                0022    109 ; Subtest description:
                                0022    110 ;
                                0022    111 ;       This subtest attempts carefully planned writes to the I-stream and
                                0022    112 ;       checks that the memory in the area of the instruction contains the
                                0022    113 ;       correct data.
                                0022    114 ;
                                0022    115 ;-
                                0022    116
                                0022    117         $DS_BGNSUB
                                0022        T2_S1::
00010030 9F   00000000'EF    FA 0022                        CALLG   $S$, a#DS$BGNSUB
                                002D    118
                                002D    119         $DS_ENDSUB
                                002D        T2_S1_X:
00010038 9F   00000000'EF    FA 002D                        CALLG   $S$, a#DS$ENDSUB
                                0038    120
                                0038    121         $DS_PAGE
```

```
                                0038                    .SBTTL  SUBTEST 2.2:  Register Mode Addressing Involving the PC
                                0038    124 ; +
                                0038    125 ; Subtest description:
                                0038    126 ;
                                0038    127 ;       This subtest does register mode addressing which involves the PC
                                0038    128 ;       using all the possible data types and checks that registers beyond
                                0038    129 ;       the range of the data type are not effected.
                                0038    130 ;
                                0038    131 ;       Note:  If the PC is modified in an undeterministic manner, an error
                                0038    132 ;       will probably result, but the report may not relate it to this test.
                                0038    133 ;
                                0038    134 ;-
                                0038    135
                                0038    136         $DS_BGNSUB
                                0038        T2_S2::
00010030 9F   0000000C'EF   FA  0038                CALLG   $S$, @#DS$BGNSUB
                                0043    137
                                0043    138         $DS_ENDSUB
                                0043        T2_S2_X:
00010038 9F   0000000C'EF   FA  0043                CALLG   $S$, @#DS$ENDSUB
                                004E    139
                                004E    140         $DS_PAGE
```

```
                                004E                    .SBTTL  SUBTEST 2.3:  Addressing Mode (PC) and -(PC)
                                004E    143 ; +
                                004E    144 ; Subtest description:
                                004E    145 ;
                                004E    146 ;       This subtest does register deferred and auto-decrement mode addressing
                                004E    147 ;       using the PC with all the possible data types and checks that the
                                004E    148 ;       memory containing the instructions are not undeterministicly modified.
                                004E    149 ;
                                004E    150 ;       Note:  If the PC is modified in an undeterministic manner, an error
                                004E    151 ;       will probably result, but the report may not relate it to this test.
                                004E    152 ;
                                004E    153 ;-
                                004E    154
                                004E    155             $DS_BGNSUB
                                004E            T2_S3::
00010030 9F   00000018'EF   FA   004E                            CALLG   $S$, @#DS$BGNSUB
                                0059    156
                                0059    157             $DS_ENDSUB
                                0059            T2_S3_X:
00010038 9F   00000018'EF   FA   0059                            CALLG   $S$, @#DS$ENDSUB
                                0064    158
                                0064    159             $DS_PAGE
```

```
                                0064                    .SBTTL  SUBTEST 2.4:  Addressing Modes 6, 7, or 8 Indexed, Same Register
                                0064    162 ; +
                                0064    163 ; Subtest description:
                                0064    164 ;
                                0064    165 ;       This subtest checks that auto-increment indexed, auto-increment
                                0064    166 ;       deferred indexed and auto-decrement indexed modes using the same
                                0064    167 ;       register as the both the bae and the index will not produce
                                0064    168 ;       undeterministic results.
                                0064    169 ;
                                0064    170 ;-
                                0064    171
                                0064    172             $DS_BGNSUB
                                0064        T2_S4::
00010030 9F   00000024'EF   FA  0064                            CALLG   $S$, @#DS$BGNSUB
                                006F    173
                                006F    174             $DS_ENDSUB
                                006F        T2_S4_X:
00010038 9F   00000024'EF   FA  006F                            CALLG   $S$, @#DS$ENDSUB
                                007A    175
                                007A    176             $DS_PAGE
```

```
                                007A                    .SBTTL  SUBTEST 2.5:  Floating Point Data Used as an Address
                                007A    179 ; +
                                007A    180 ; Subtest description:
                                007A    181 ;
                                007A    182 ;       This subtest checks that auto-increment, auto-increment deferred and
                                007A    183 ;       auto-decrement modes using a register that contains a floating point
                                007A    184 ;       value for the same instruction will not produce undeterministic
                                007A    185 ;       results.
                                007A    186 ;
                                007A    187 ;-
                                007A    188
                                007A    189         $DS_BGNSUB
                                007A        T2_S5::
00010030 9F   00000030'EF   FA  007A                CALLG   $S$, @#DS$BGNSUB
                                0085    190
                                0085    191         $DS_ENDSUB
                                0085        T2_S5_X:
00010038 9F   00000030'EF   FA  0085                CALLG   $S$, @#DS$ENDSUB
                                0090    192
                                0090    193         $DS_PAGE
```

```
                                0090                    .SBTTL  SUBTEST 2.6:  FPD Set by the Program
                                0090    196 ; +
                                0090    197 ; Subtest description:
                                0090    198 ;
                                0090    199 ;       This subtest sets the First Part Done bit in the PSL (Via an REI),
                                0090    200 ;       then tries to execute an instruction that doesn't allow FPD.  The
                                0090    201 ;       result should be a reserved instruction fault on the VAX-11/730.
                                0090    202 ;
                                0090    203 ;-
                                0090    204
                                0090    205             $DS_BGNSUB
                                0090            T2_S6::
00010030 9F   0000003C'EF   FA  0090                            CALLG   $S$, @#DS$BGNSUB
                                009B    206
                                009B    207             $DS_ENDSUB
                                009B            T2_S6_X:
00010038 9F   0000003C'EF   FA  009B                            CALLG   $S$, @#DS$ENDSUB
                                00A6    208
                                00A6    209             $DS_PAGE
```

```
                                00A6                    .SBTTL  SUBTEST 2.7:  Clearing TP and not T
                                00A6    212 ; +
                                00A6    213 ; Subtest description:
                                00A6    214 ;
                                00A6    215 ;       This subtest generates an exception while the T-bit is set which sets
                                00A6    216 ;       the TP-bit.  Then the TP-bit is cleared, the exception condition is
                                00A6    217 ;       cleared, and execution is resumed.  A trace trap is expected after the
                                00A6    218 ;       next instruction, at the latest, and the processor state is checked
                                00A6    219 ;       for possible side effects.
                                00A6    220 ;
                                00A6    221 ;-
                                00A6    222
                                00A6    223             $DS_BGNSUB
                                00A6            T2_S7::
00010030 9F   00000048'EF   FA  00A6                    CALLG.  $S$, @#DS$BGNSUB
                                00B1    224
                                00B1    225             $DS_ENDSUB
                                00B1            T2_S7_X:
00010038 9F   00000048'EF   FA  00B1                    CALLG   $S$, @#DS$ENDSUB
                                00BC    226
                                00BC    227             $DS_PAGE
```

```
                                00BC                    .SBTTL  SUBTEST 2.8:  Referrences to the Current SP Via MTPR or MFPR
                                00BC    230 ; +
                                00BC    231 ; Subtest description:
                                00BC    232 ;
                                00BC    233 ;       This subtest tries to read and write the current modes stack pointer
                                00BC    234 ;       using the MFPR and MTPR instructions.  This is done for both the
                                00BC    235 ;       kernel and interrupt stack pointers.
                                00BC    236 ;
                                00BC    237 ;-
                                00BC    238
                                00BC    239         $DS_BGNSUB
                                00BC        T2_S8::
00010030 9F   00000054'EF   FA  00BC                    CALLG   $S$, @#DS$BGNSUB
                                00C7    240
                                00C7    241         $DS_ENDSUB
                                00C7        T2_S8_X:
00010038 9F   00000054'EF   FA  00C7                    CALLG   $S$, @#DS$ENDSUB
                                00D2    242
                                00D2    243         $DS_PAGE
```

```
                     50    01    D0  00D2                          MOVL    #1, R0          ; NORMAL EXIT
                                     00D5          TEST_002_X::
           00010058  9F    6E    FA  00D5                          CALLG   (SP), @#DS$BREAK
                                     04  00DC                      RET                     ; RETURN TO TEST SEQUENCER
                                         00DD    246
                                     00000000                      .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                           00000002  0000                          .LONG   $TN
                                         0004    248
                                         0004    249          .END
```

```
$$$                 = 00000054 R     06      DS$MMOFF            00010158        DS$_UNEXPINT    = 006600D8
$$E                 = 00000001               DS$MMON             00010150        DS$_VASFULL     = 00660048
$$M                 = 00000001               DS$MOVPHY           00010148        DS$_WARNING     = 00660000
$$N                 = 00000000               DS$MOVVRT           00010140        ENV$M_DOMAIN    = 00000002
$$S                 = FFFFFFFF               DS$PARSE            000100B8        ENV$M_LEVEL     = 00000001
$ENV                = 00000001  G            DS$PRINTB           000100E0        ENV$M_SUPER     = 000003FC
$ER                 = FFFFFFFF               DS$PRINTF           000100F0        ENV$S_DOMAIN    = 00000001
$MO                 = 00000001  G            DS$PRINTS           000100F8        ENV$S_LEVEL     = 00000001
$S$                 = 00000054 R     06      DS$PRINTSIG         00010100        ENV$S_SUPER     = 00000008
$ST                 = 00000000               DS$PRINTX           000100E8        ENV$V_DOMAIN    = 00000001
$TN                 = 00000002               DS$PROBE            000101A0        ENV$V_LEVEL     = 00000000
ALL                 = 00000008  G            DS$RELBUF           00010128        ENV$V_SUPER     = 00000002
BASE_002              00000000 R     04      DS$RELMEM           00010138        ENV$_CPU        = 00000000
BIT...              = 00660118               DS$SETIPL           00010178        ENV$_FUNCTIONAL = 00000000
CEP_FUNCTIONAL      = 00000000               DS$SETMAP           00010188        ENV$_REPAIR     = 00000001
CEP_REPAIR          = 00000001               DS$SETPRIEXV        00010110        ENV$_SUPER      = 00000001
DATA_002              0000001C R     04      DS$SETVEC           00010160        ENV$_SYSTEM     = 00000001
DEFAULT             = 00000000  G            DS$SHOCHAN          00010190        HALT            = 00000004  G
DS$ABORT              00010020               DS$SUMMARY          00010028        IPR             = 00000006  G
DS$ASKADR             00010090               DS$WAITMS           00010060        LDPCTX          = 0000000B  G
DS$ASKDATA            00010080               DS$WAITUS           00010068        MANUAL          = 00000001  G
DS$ASKLGCL            00010098               DS$_ARITH       = 006600D0          MCHK            = 00000005  G
DS$ASKSTR             000100A0               DS$_BADLINK     = 006600F0          MEM             = 00000009  G
DS$ASKVLD             00010088               DS$_BADTYPE     = 006600E8          MSG_002           00000002 R     04
DS$ATTACH             000101A8               DS$_CHME        = 006600A8          POWER           = 00000003  G
DS$BGNSUB             00010030               DS$_CHMK        = 006600E0          SCB$L_ACCESS      00000020
DS$BRANCH             000100A8               DS$_DEVNAME     = 00660108          SCB$L_ARITH       00000034
DS$BREAK              00010058               DS$_ERROR       = 00660002          SCB$L_BREAK       0000002C
DS$CANWAIT            00010070               DS$_FHWE        = 00660068          SCB$L_CHME        00000044
DS$CHANNEL            00010180               DS$_FRAGBUF     = 00660080          SCB$L_CHMK        00000040
DS$CKLOOP             00010040               DS$_ICBUSY      = 006600C8          SCB$L_CHMS        00000048
DS$CLRVEC             00010168               DS$_ICERR       = 006600C0          SCB$L_CHMU        0000004C
DS$CNTRLC             00010078               DS$_IHWE        = 00660060          SCB$L_COMPAT      00000030
DS$CVTREG             000100B0               DS$_ILLCHAR     = 00660018          SCB$L_KNLSTK      00000008
DS$ENDPASS            00010010               DS$_ILLPAGCNT   = 00660078          SCB$L_MACHCHK     00000004
DS$ENDSUB             00010038               DS$_ILLUNIT     = 00660100          SCB$L_OPCCUS      00000014
DS$ERRDEV             000100C8               DS$_INSFMEM     = 00660050          SCB$L_OPCDEC      00000010
DS$ERRHARD            000100D0               DS$_IPL2HI      = 006600B8          SCB$L_POWER       0000000C
DS$ERRPREP            00010108               DS$_IVADDR      = 00660040          SCB$L_RADRMOD     0000001C
DS$ERRSOFT            000100D8               DS$_IVVECT      = 00660038          SCB$L_ROPRAND     00000018
DS$ERRSYS             000100C0               DS$_KRNLSTK     = 00660090          SCB$L_RXDB        000000F8
DS$ESCAPE             00010050               DS$_LOGIC       = 00660070          SCB$L_SFTLVL1     00000084
DS$FREEDBGSYM         000101B8               DS$_MCHK        = 00660088          SCB$L_SFTLVL10    000000A8
DS$GETBUF             00010120               DS$_MMOFF       = 00660058          SCB$L_SFTLVL11    000000AC
DS$GETMEM             00010130               DS$_NEEDUNIT    = 006600F8          SCB$L_SFTLVL12    000000B0
DS$GPHARD             00010018               DS$_NOPCS       = 00660110          SCB$L_SFTLVL13    000000B4
DS$HELP               000101B0               DS$_NORMAL      = 00660001          SCB$L_SFTLVL14    000000B8
DS$INITSCB            00010170               DS$_NOSUPPORT   = 006600B1          SCB$L_SFTLVL15    000000BC
DS$INLOOP             00010048               DS$_NOTDON      = 00660030          SCB$L_SFTLVL2     00000088
DS$K_ERROR          = 00000002               DS$_NOTIMP      = 006600B0          SCB$L_SFTLVL3     0000008C
DS$K_NORMAL         = 00000001               DS$_NULLSTR     = 00660010          SCB$L_SFTLVL4     00000090
DS$K_SEVERE         = 00000004               DS$_OVERFLOW    = 00660008          SCB$L_SFTLVL5     00000094
DS$K_SUBSYS         = 00000066               DS$_POWER       = 00660098          SCB$L_SFTLVL6     00000098
DS$K_WARNING        = 00000000               DS$_PROGERR     = 00660020          SCB$L_SFTLVL7     0000009C
DS$LOAD               00010198               DS$_SEVERE      = 00660004          SCB$L_SFTLVL8     000000A0
DS$LOADPCS            000101C0               DS$_TRANSL      = 006600A0          SCB$L_SFTLVL9     000000A4
DS$MAPDBGBLOCK        00010118               DS$_TRUNCATE    = 00660028          SCB$L_TBIT        00000028
```

```
SCB$L_TIMER       000000C0          T2_S7_X         000000B1 R     04
SCB$L_TRANSL      00000024          T2_S8           000000BC RG    04
SCB$L_TXDB        000000FC          T2_S8_X         000000C7 R     04
SCB$L_ZERO        00000000          TEST_002        00000020 RG    04
SEP_FUNCTIONAL  = 00000002          TEST_002_X      000000D5 RG    04
SEP_REPAIR      = 00000003          TU58          = 00000002  G
SIZ...          = 00000008          UBI           = 0000000A  G
SYS$ALLOC         00010238          WCS           = 00000007  G
SYS$ASCTIM        00010248
SYS$ASSIGN        00010250
SYS$BINTIM        00010258
SYS$CANCEL        00010260
SYS$CANTIM        00010268
SYS$CLOSE         000105B8
SYS$CLREF         00010298
SYS$CONNECT       000105C0
SYS$DALLOC        000102D8
SYS$DASSGN        000102E0
SYS$DISCONNECT    000105D0
SYS$FAO           00010350
SYS$FAOL          00010358
SYS$GET           00010580
SYS$GETCHN        000104C8
SYS$GETTIM        00010378
SYS$LKWSET        000103A0
SYS$NUMTIM        000103B8
SYS$OPEN          00010608
SYS$QIO           000103C8
SYS$QIOW          00010200
SYS$READ          00010590
SYS$READEF        000103D0
SYS$SETAST        000103F8
SYS$SETEF         00010400
SYS$SETIMR        00010420
SYS$SETPRI        00010428
SYS$SETPRT        00010430
SYS$SETRWM        00010438
SYS$SETSWM        00010448
SYS$ULKPAG        00010460
SYS$ULWSET        00010468
SYS$UNWIND        00010470
SYS$WAITFR        00010478
SYS$WFLAND        00010488
SYS$WFLOR         00010490
T2_S1             00000022 RG    04
T2_S1_X           0000002D R     04
T2_S2             00000038 RG    04
T2_S2_X           00000043 R     04
T2_S3             0000004E RG    04
T2_S3_X           00000059 R     04
T2_S4             00000064 RG    04
T2_S4_X           0000006F R     04
T2_S5             0000007A RG    04
T2_S5_X           00000085 R     04
T2_S6             00000090 RG    04
T2_S6_X           0000009B R     04
T2_S7             000000A6 RG    04
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes									
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
ENKAXB	00000000 (0.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
TEST_002	000000DD (221.)	04 (4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC PAGE
DISPATCH	00000018 (24.)	05 (5.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	NOWRT	NOVEC LONG
ARGLIST	00000060 (96.)	06 (6.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC LONG
$TSTCNT	00000004 (4.)	07 (7.)	NOPIC	USR	OVR	REL	LCL	NOSHR	NOEXE	RD	NOWRT	NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000054-R	239 (11)	117 (4) 136 (5) 155 (6) 172 (7) 189 (8) 205 (9) 223 (10) 239 (11)
$$E	=00000001	239 (11)	119 (4) 138 (5) 157 (6) 174 (7) 191 (8) 207 (9) 225 (10) 241 (11)
$$M	=00000001	103 (3)	103 (3)
$$N	=00000000	103 (3)	81 (2)
$$S	=FFFFFFFF	245 (12)	103 (3) 107 (4) 117 (4) 123 (5) 136 (5) 142 (6) 155 (6) 161 (7) 172 (7) 178 (8) 189 (8) 195 (9) 205 (9) 211 (10) 223 (10) 229 (11) 239 (11) 92 (3)
$ENV	=00000001	58 (2)	
$ER	=FFFFFFFF	245 (12)	117 (4) 136 (5) 155 (6) 172 (7) 189 (8) 205 (9) 223 (10) 239 (11)
$MO	=00000001	247 (12)	247 (12)
S	=00000054-R	239 (11)	117 (4) 119 (4) 136 (5) 138 (5) 155 (6) 157 (6) 172 (7) 174 (7) 189 (8) 191 (8) 205 (9) 207 (9) 223 (10) 225 (10) 239 (11) 241 (11)
$ST	=00000000	245 (12)	103 (3) 107 (4) 117 (4) 119 (4) 123 (5) 136 (5) 138 (5) 142 (6) 155 (6) 157 (6) 161 (7) 172 (7) 174 (7) 178 (8) 189 (8) 191 (8) 195 (9) 205 (9) 207 (9) 211 (10) 223 (10) 225 (10) 229 (11) 239 (11) 241 (11) 245 (12)
$TN	=00000002	247 (12)	103 (3) 107 (4) 123 (5) 142 (6) 161 (7) 178 (8) 195 (9) 211 (10) 229 (11) 245 (12) 247 (12) 92 (3)
ALL	=00000008	81 (2)	
BASE_002	00000000-R	92 (3)	103 (3)
BIT...	=00660118	79 (2)	58 (2) 79 (2)
CEP_FUNCTIONAL	=00000000	58 (2)	
CEP_REPAIR	=00000001	58 (2)	58 (2)
DATA_002	0000001C-R	103 (3)	103 (3)
DEFAULT	=00000000	81 (2)	
DS$ABORT	00010020	80 (2)	
DS$ASKADR	00010090	80 (2)	
DS$ASKDATA	00010080	80 (2)	
DS$ASKLGCL	00010098	80 (2)	
DS$ASKSTR	000100A0	80 (2)	
DS$ASKVLD	00010088	80 (2)	
DS$ATTACH	000101A8	80 (2)	
DS$BGNSUB	00010030	80 (2)	117 (4) 136 (5) 155 (6) 172 (7) 189 (8) 205 (9) 223 (10) 239 (11)
DS$BRANCH	000100A8	80 (2)	
DS$BREAK	00010058	80 (2)	245 (12)
DS$CANWAIT	00010070	80 (2)	
DS$CHANNEL	00010180	80 (2)	
DS$CKLOOP	00010040	80 (2)	
DS$CLRVEC	00010168	80 (2)	
DS$CNTRLC	00010078	80 (2)	

```
DS$CVTREG          000100B0      80      (2)
DS$ENDPASS         00010010      80      (2)
DS$ENDSUB          00010038      80      (2)       119     (4)       138     (5)       157     (6)       174     (7)       191     (8)
                                                   207     (9)       225     (10)      241     (11)
DS$ERRDEV          000100C8      80      (2)
DS$ERRHARD         000100D0      80      (2)
DS$ERRPREP         00010108      80      (2)
DS$ERRSOFT         000100D8      80      (2)
DS$ERRSYS          000100C0      80      (2)
DS$ESCAPE          00010050      80      (2)
DS$FREEDBGSYM      000101B8      80      (2)
DS$GETBUF          00010120      80      (2)
DS$GETMEM          00010130      80      (2)
DS$GPHARD          00010018      80      (2)
DS$HELP            000101B0      80      (2)
DS$INITSCB         00010170      80      (2)
DS$INLOOP          00010048      80      (2)
DS$K_ERROR        =00000002      79      (2)
DS$K_NORMAL       =00000001      79      (2)
DS$K_SEVERE       =00000004      79      (2)
DS$K_SUBSYS       =00000066      79      (2)       79      (2)
DS$K_WARNING      =00000000      79      (2)
DS$LOAD            00010198      80      (2)
DS$LOADPCS         000101C0      80      (2)
DS$MAPDBGBLOCK     00010118      80      (2)
DS$MMOFF           00010158      80      (2)
DS$MMON            00010150      80      (2)
DS$MOVPHY          00010148      80      (2)
DS$MOVVRT          00010140      80      (2)
DS$PARSE           000100B8      80      (2)
DS$PRINTB          000100E0      80      (2)
DS$PRINTF          000100F0      80      (2)
DS$PRINTS          000100F8      80      (2)
DS$PRINTSIG        00010100      80      (2)
DS$PRINTX          000100E8      80      (2)
DS$PROBE           000101A0      80      (2)
DS$RELBUF          00010128      80      (2)
DS$RELMEM          00010138      80      (2)
DS$SETIPL          00010178      80      (2)
DS$SETMAP          00010188      80      (2)
DS$SETPRIEXV       00010110      80      (2)
DS$SETVEC          00010160      80      (2)
DS$SHOCHAN         00010190      80      (2)
DS$SUMMARY         00010028      80      (2)
DS$WAITMS          00010060      80      (2)
DS$WAITUS          00010068      80      (2)
DS$_ARITH         =006600D0      79      (2)
DS$_BADLINK       =006600F0      79      (2)
DS$_BADTYPE       =006600E8      79      (2)
DS$_CHME          =006600A8      79      (2)
DS$_CHMK          =006600E0      79      (2)
DS$_DEVNAME       =00660108      79      (2)
DS$_ERROR         =00660002      79      (2)
DS$_FHWE          =00660068      79      (2)
DS$_FRAGBUF       =00660080      79      (2)
DS$_ICBUSY        =006600C8      79      (2)
DS$_ICERR         =006600C0      79      (2)
```

```
DS$_IHWE          =00660060      79      (2)
DS$_ILLCHAR       =00660018      79      (2)
DS$_ILLPAGCNT     =00660078      79      (2)
DS$_ILLUNIT       =00660100      79      (2)
DS$_INSFMEM       =00660050      79      (2)
DS$_IPL2HI        =006600B8      79      (2)
DS$_IVADDR        =00660040      79      (2)
DS$_IVVECT        =00660038      79      (2)
DS$_KRNLSTK       =00660090      79      (2)
DS$_LOGIC         =00660070      79      (2)
DS$_MCHK          =00660088      79      (2)
DS$_MMOFF         =00660058      79      (2)
DS$_NEEDUNIT      =006600F8      79      (2)
DS$_NOPCS         =00660110      79      (2)
DS$_NORMAL        =00660001      79      (2)
DS$_NOSUPPORT     =006600B1      79      (2)
DS$_NOTDON        =00660030      79      (2)
DS$_NOTIMP        =006600B0      79      (2)       79      (2)
DS$_NULLSTR       =00660010      79      (2)
DS$_OVERFLOW      =00660008      79      (2)
DS$_POWER         =00660098      79      (2)
DS$_PROGERR       =00660020      79      (2)
DS$_SEVERE        =00660004      79      (2)
DS$_TRANSL        =006600A0      79      (2)
DS$_TRUNCATE      =00660028      79      (2)
DS$_UNEXPINT      =006600D8      79      (2)
DS$_VASFULL       =00660048      79      (2)
DS$_WARNING       =00660000      79      (2)       79      (2)
ENV$M_DOMAIN      =00000002      58      (2)
ENV$M_LEVEL       =00000001      58      (2)
ENV$M_SUPER       =000003FC      58      (2)
ENV$S_DOMAIN      =00000001      58      (2)
ENV$S_LEVEL       =00000001      58      (2)
ENV$S_SUPER       =00000008      58      (2)
ENV$V_DOMAIN      =00000001      58      (2)       58      (2)
ENV$V_LEVEL       =00000000      58      (2)       58      (2)
ENV$V_SUPER       =00000002      58      (2)
ENV$_CPU          =00000000      58      (2)       58      (2)
ENV$_FUNCTIONAL   =00000000      58      (2)       58      (2)
ENV$_REPAIR       =00000001      58      (2)       58      (2)
ENV$_SUPER        =00000001      58      (2)
ENV$_SYSTEM       =00000001      58      (2)       58      (2)
HALT              =00000004      81      (2)
IPR               =00000006      81      (2)
LDPCTX            =0000000B      81      (2)
MANUAL            =00000001      81      (2)
MCHK              =00000005      81      (2)
MEM               =00000009      81      (2)
MSG_002            00000002-R    92      (3)       103     (3)
POWER             =00000003      81      (2)
SCB$L_ACCESS       00000020      78      (2)
SCB$L_ARITH        00000034      78      (2)
SCB$L_BREAK        0000002C      78      (2)
SCB$L_CHME         00000044      78      (2)
SCB$L_CHMK         00000040      78      (2)
SCB$L_CHMS         00000048      78      (2)
SCB$L_CHMU         0000004C      78      (2)
```

SCB$L_COMPAT	00000030	78	(2)		
SCB$L_KNLSTK	00000008	78	(2)		
SCB$L_MACHCHK	00000004	78	(2)		
SCB$L_OPCCUS	00000014	78	(2)		
SCB$L_OPCDEC	00000010	78	(2)		
SCB$L_POWER	0000000C	78	(2)		
SCB$L_RADRMOD	0000001C	78	(2)		
SCB$L_ROPRAND	00000018	78	(2)		
SCB$L_RXDB	000000F8	78	(2)		
SCB$L_SFTLVL1	00000084	78	(2)		
SCB$L_SFTLVL10	000000A8	78	(2)		
SCB$L_SFTLVL11	000000AC	78	(2)		
SCB$L_SFTLVL12	000000B0	78	(2)		
SCB$L_SFTLVL13	000000B4	78	(2)		
SCB$L_SFTLVL14	000000B8	78	(2)		
SCB$L_SFTLVL15	000000BC	78	(2)		
SCB$L_SFTLVL2	00000088	78	(2)		
SCB$L_SFTLVL3	0000008C	78	(2)		
SCB$L_SFTLVL4	00000090	78	(2)		
SCB$L_SFTLVL5	00000094	78	(2)		
SCB$L_SFTLVL6	00000098	78	(2)		
SCB$L_SFTLVL7	0000009C	78	(2)		
SCB$L_SFTLVL8	000000A0	78	(2)		
SCB$L_SFTLVL9	000000A4	78	(2)		
SCB$L_TBIT	00000028	78	(2)		
SCB$L_TIMER	000000C0	78	(2)		
SCB$L_TRANSL	00000024	78	(2)		
SCB$L_TXDB	000000FC	78	(2)		
SCB$L_ZERO	00000000	78	(2)		
SEP_FUNCTIONAL	=00000002	58	(2)		
SEP_REPAIR	=00000003	58	(2)		
SIZ...	=00000008	58	(2)	58	(2)
SYS$ALLOC	00010238	80	(2)		
SYS$ASCTIM	00010248	80	(2)		
SYS$ASSIGN	00010250	80	(2)		
SYS$BINTIM	00010258	80	(2)		
SYS$CANCEL	00010260	80	(2)		
SYS$CANTIM	00010268	80	(2)		
SYS$CLOSE	000105B8	80	(2)		
SYS$CLREF	00010298	80	(2)		
SYS$CONNECT	000105C0	80	(2)		
SYS$DALLOC	000102D8	80	(2)		
SYS$DASSGN	000102E0	80	(2)		
SYS$DISCONNECT	000105D0	80	(2)		
SYS$FAO	00010350	80	(2)		
SYS$FAOL	00010358	80	(2)		
SYS$GET	00010580	80	(2)		
SYS$GETCHN	000104C8	80	(2)		
SYS$GETTIM	00010378	80	(2)		
SYS$LKWSET	000103A0	80	(2)		
SYS$NUMTIM	000103B8	80	(2)		
SYS$OPEN	00010608	80	(2)		
SYS$QIO	000103C8	80	(2)		
SYS$QIOW	00010200	80	(2)		
SYS$READ	00010590	80	(2)		
SYS$READEF	000103D0	80	(2)		
SYS$SETAST	000103F8	80	(2)		

```
SYS$SETEF           00010400      80      (2)
SYS$SETIMR          00010420      80      (2)
SYS$SETPRI          00010428      80      (2)
SYS$SETPRT          00010430      80      (2)
SYS$SETRWM          00010438      80      (2)
SYS$SETSWM          00010448      80      (2)
SYS$ULKPAG          00010460      80      (2)
SYS$ULWSET          00010468      80      (2)
SYS$UNWIND          00010470      80      (2)
SYS$WAITFR          00010478      80      (2)
SYS$WFLAND          00010488      80      (2)
SYS$WFLOR           00010490      80      (2)
T2_S1               00000022-R    117     (4)
T2_S1_X             0000002D-R    119     (4)
T2_S2               00000038-R    136     (5)
T2_S2_X             00000043-R    138     (5)
T2_S3               0000004E-R    155     (6)
T2_S3_X             00000059-R    157     (6)
T2_S4               00000064-R    172     (7)
T2_S4_X             0000006F-R    174     (7)
T2_S5               0000007A-R    189     (8)
T2_S5_X             00000085-R    191     (8)
T2_S6               00000090-R    205     (9)
T2_S6_X             0000009B-R    207     (9)
T2_S7               000000A6-R    223     (10)
T2_S7_X             000000B1-R    225     (10)
T2_S8               000000BC-R    239     (11)
T2_S8_X             000000C7-R    241     (11)
TEST_002            00000020-R    103     (3)       103     (3)
TEST_002_X          000000D5-R    245     (12)
TU58              =00000002       81      (2)
UBI               =0000000A       81      (2)
WCS               =00000007       81      (2)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_BSUBT	1	117 (4)	117 (4) 136 (5) 155 (6) 172 (7) 189 (8) 205 (9) 223 (10) 239 (11)
$D1_BTEST	1	103 (3)	103 (3)
$D1_ESUBT	1	119 (4)	119 (4) 138 (5) 157 (6) 174 (7) 191 (8) 207 (9) 225 (10) 241 (11)
$D1_ETEST	1	245 (12)	245 (12)
$D1_SBTTL	2	92 (3)	107 (4) 123 (5) 142 (6) 161 (7) 178 (8) 195 (9) 211 (10) 229 (11) 92 (3)
$D2_BDATA	1	103 (3)	103 (3)
$D2_BTEST	1	103 (3)	103 (3)
$D2_SBTTL	1	92 (3)	92 (3)
$DEF	1	80 (2)	78 (2) 80 (2)
$DEFEND	1	78 (2)	78 (2) 80 (2)
$DEFINI	1	78 (2)	78 (2) 80 (2)
$DS_BGNMOD	1	58 (2)	58 (2)
$DS_BGNSUB	1	117 (4)	117 (4) 136 (5) 155 (6) 172 (7) 189 (8) 205 (9) 223 (10) 239 (11)
$DS_BGNTEST	1	103 (3)	103 (3)
$DS_BREAK	1	245 (12)	245 (12)
$DS_DSDEF	2	79 (2)	79 (2)
$DS_DSSDEF	1	80 (2)	80 (2)
$DS_ENDDATA	1	103 (3)	103 (3)
$DS_ENDMOD	1	247 (12)	247 (12)
$DS_ENDSUB	1	119 (4)	119 (4) 138 (5) 157 (6) 174 (7) 191 (8) 207 (9) 225 (10) 241 (11)
$DS_ENDTEST	1	245 (12)	245 (12)
$DS_ENVDEF	2	58 (2)	58 (2)
$DS_PAGE	1	90 (2)	105 (3) 121 (4) 140 (5) 159 (6) 176 (7) 193 (8) 209 (9) 227 (10) 243 (11) 90 (2)
$DS_SBTTL	1	92 (3)	107 (4) 123 (5) 142 (6) 161 (7) 178 (8) 195 (9) 211 (10) 229 (11) 92 (3)
$DS_SCBDEF	1	78 (2)	78 (2)
$DS_SECDEF	1	81 (2)	81 (2)
$EQU	1	80 (2)	58 (2) 79 (2)
$EQULS1	1	79 (2)	58 (2) 79 (2)
$EQULST	1	58 (2)	58 (2) 79 (2)
$GBLINI	2	58 (2)	58 (2) 78 (2) 79 (2) 80 (2)
$VIELD	1	58 (2)	58 (2)
$VIELD1	1	80 (2)	58 (2)
ENKAX_SECDEF	1	81 (2)	81 (2)

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.03	00:00:00.32
Command processing	23	00:00:00.18	00:00:00.48
Pass 1	776	00:00:07.50	00:00:17.89

Symbol table sort	7	00:00:00.25	00:00:00.51
Pass 2	149	00:00:01.22	00:00:02.78
Symbol table output	23	00:00:00.14	00:00:00.16
Psect synopsis output	6	00:00:00.04	00:00:00.23
Cross-reference output	100	00:00:01.03	00:00:02.63
Assembler run totals	1115	00:00:10.41	00:00:25.01

The working set limit was 750 pages.
55908 bytes (110 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 236 non-local and 0 local symbols.
249 source lines were read in Pass 1, producing 22 object records in Pass 2.
65 pages of virtual memory were used to define 28 macros.

! Macro library statistics !

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	23
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	5
TOTALS (all libraries)	29

527 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXB

Table of contents

```
                        1          .SBTTL  ENKAXC: POWER FAIL
                        2          .TITLE  ENKAXC  VAX-11/730 Specific CPU Cluster Exerciser
                        3          .IDENT  /1-3/
                        4
                        5 ;
                        6 ; Copyright (C) 1981
                        7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
                        8 ;
                        9 ; This software is furnished  under a license for use only on a single
                       10 ; computer  system and  may be  copied only with  the inclusion of the
                       11 ; above copyright notice.  This software, or any other copies thereof,
                       12 ; may not be provided or  otherwise made available to any other person
                       13 ; except for use on such system and to one who agrees to these license
                       14 ; terms.  Title to and  ownership of the  software  shall at all times
                       15 ; remain in DEC.
                       16 ;
                       17 ; The information in this software is subject to change without notice
                       18 ; and should  not be construed  as a commitment  by Digital  Equipment
                       19 ; Corporation.
                       20 ;
                       21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
                       22 ; software on equipment which is not supplied by DEC.
                       23 ;
                       24
                       25 ;++
                       26 ; Facility:     VAX Architecture Diagnostic.
                       27 ;
                       28 ; Abstract:     This module tests that the VAX-11/730 power failure and
                       29 ;               recovery mechanism functions properly.  The power fail
                       30 ;               tests instruct the operator to select console panel switch
                       31 ;               settings and force various power interruptions.
                       32 ;
                       33 ;
                       34 ; Environment:  VAX Diagnostic Supervisor.
                       35 ;
                       36 ; Author:       Rich Lewis      7-Jly-81        Version 1.0
                       37 ;
                       38 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
                       39 ;
                       40 ;               Added quick flag on Test 7(TU58). If quick flag is set
                       41 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
                       42 ;               If event flag 3 is set, will override protection,
                       43 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
                       44 ;               is clear & tape is not scratch, will report an error.
                       45 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
                       46 ;
                       47 ;
                       48 ;               Tai-Chun Pan    03-Aug-83       Version 1.3
                       49 ;
                       50 ;               fixed bugs on TEST 7 and error summary routine call when
                       51 ;               running under APT.
                       52 ;
                       53 ;
                       54 ;--
```

```
                     0000     56
                     0000     57          .SBTTL  DECLARATIONS
                     0000     58          .LIST   MEB
                     0000     59          .NLIST  CND
                 00000000     60          .PSECT  ENKAXC, PAGE, NOWRT
                     0000     61          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     62          $DS_BGNMOD      CEP_REPAIR, IN=2
                     0000        ;;;      Macro-32 Di.gnostic macro library Version 6.10 'DIAG.MLB (137)"
                     0000     63
                     0000     64 ;
                     0000     65 ; Include files:
                     0000     66 ;
                     0000     67
                     0000     68          .LIBRARY        "ENKAX"                 ; CPU Cluster Exerciser library.
                     0000     69
                     0000     70 ;
                     0000     71 ; Macros:
                     0000     72 ;
                     0000     73
                     0000     74          $DS_SCBDEF
                     0000     75          $DS_DSDEF
                     0000     76          $DS_DSSDEF
                     0000     77          $DS_PSLDEF
                     0000                 .SAVE   LOCAL_BLOCK
                     0000                 .PSECT  $ABS$,ABS
                00000000  0610            .=0
                00000000                  .RESTORE
                     0000     78          $DS_PARDEF
                     0000     79
                     0000     80          ENKAX_SECDEF
                     0000     81
                     0000     82
                     0000     83 ;
                     0000     84 ; Own storage:
                     0000     85 ;
                     0000     86
00000004             0000     87 L_BUF_BEGIN:     .BLKL   1       ; address of data stored on power failure
00000008             0004     88 L_COUNT:         .BLKL   1       ; counter
0000000C             0008     89 L_CHECKSUM:      .BLKL   1       ; contains good RPB checksum
00000010             000C     90 L_CHECKSUM1:     .BLKL   1       ; contains error RPB checksum
00000014             0010     91 L_PSL:           .BLKL   1       ; used to store PSL
00000018             0014     92 L_IS_SP:         .BLKL   1       ; used to store interrupt SP
0000001C             0018     93 L_KS_SP:         .BLKL   1       ; used to store kernel SP
                     001C     94
0000001D             001C     95 B_RESPONSE:      .BLKB   1       ; message response byte
0000001E             001D     96 B_FLAGS:         .BLKB   1       ; byte to keep power up and down done flags
0000001F             001E     97 B_FLAGS2:        .BLKB   1       ; byte to keep stack and mode flags
                     001F     98
00000000             001F     99 V_PWRDN_DONE=0                   ; power down done position in flags
00000001             001F    100 M_PWRDN_DONE=1@0                 ; power down done mask in flags
00000001             001F    101 V_PWRUP_DONE=1                   ; power up done position in flags
00000002             001F    102 M_PWRUP_DONE=1@1                 ; power up done mask in flags
00000002             001F    103 V_BAD_PC=2                       ; incorrect pc position in flags
00000004             001F    104 M_BAD_PC=1@2                     ; incorrect pc mask in flags
                     001F    105
00000000             001F    106 V_BADMODE_DN=0                   ; bad mode on power down position in flags2
00000001             001F    107 M_BADMODE_DN=1@0                 ; bad mode on power down mask in flags2
```

```
                    00000001  001F   108 V_BADMODE_UP=1                  ; bad mode on power up position in flags2
                    00000002  001F   109 M_BADMODE_UP=1@1                ; bad mode on power up mask in flags2
                    00000002  001F   110 V_BADSTACK_DN=2                  ; bad stack on power down position in flags2
                    00000004  001F   111 M_BADSTACK_DN=1@2                ; bad stack on power down mask in flags2
                    00000003  001F   112 V_BADSTACK_UP=3                  ; bad stack on power up position in flags2
                    00000008  001F   113 M_BADSTACK_UP=1@3                ; bad stack on power up mask in flags2
                    00000004  001F   114 V_PWRUP_ERR=4                    ; power up error position in flags2
                    00000010  001F   115 M_PWRUP_ERR=1@4                  ; power up error mask in flags2
                              001F   116
                              001F   117
            000004A0'9F    17 001F   118 JUMP:   JMP     @#POWER_UP        ; used to load instruction at loc 100 to
                              0025   119                                   ; restart from subtests which halt
                    00000100  0025   120 STRTUP=^X100                     ; label for loc 100
```

```
                                        0025    122 .SBTTL  INSTRUCTION MESSAGES
                                        0025    123 ;-----------------------------------------------------------------------
                                        0025    124 T_NO_KA730:
65 63 69 76 65 64 20 6F 4E 2F 21 00'    0025    125         .ASCIC  "!/No device of type KA730 selected.  Exiting test.!/"
37 41 4B 20 65 70 79 74 20 66 6F 20     0031
2E 64 65 74 63 65 6C 65 73 20 30 33     003D
65 74 20 67 6E 69 74 69 78 45 20 20     0049
                  2F 21 2E 74 73        0055
                              34        0025
                                        005A    126 ;-----------------------------------------------------------------------
                                        005A    127 T_INST:
6F 63 20 65 72 6F 66 65 42 2F 21 00'    005A    128         .ASCIC  "!/Before continuing, set the AUTO RESTART switch to ON!/"
65 73 20 2C 67 6E 69 75 6E 69 74 6E     0066
52 20 4F 54 55 41 20 65 68 74 20 74     0072
63 74 69 77 73 20 54 52 41 54 53 45     007E
         2F 21 4E 4F 20 6F 74 20 68     008A
                              38        005A
                                        0093    129 ;-----------------------------------------------------------------------
                                        0093    130 T_MESSAGE_1:
74 73 65 52 20 64 6F 6F 47 2F 21 00'    0093    131         .ASCIC  "!/Good Restart Parameter Block Subtest - This subtest!/"-
65 74 65 6D 61 72 61 50 20 74 72 61     009F
74 62 75 53 20 6B 63 6F 6C 42 20 72     00AB
73 20 73 69 68 54 20 2D 20 74 73 65     00B7
6C 6C 69 77 2F 21 74 73 65 74 62 75     00C3
73 74 69 20 74 72 61 74 73 65 72 20     00CF
52 20 65 68 74 20 66 69 20 66 6C 65     00DB
72 65 70 6F 72 70 20 73 69 20 42 50     00E7
74 65 72 70 72 65 74 6E 69 20 79 6C     00F3
                  2F 21 2E 64 65        00FF
                                        0104    132                 "will restart itself if the RPB is properly interpreted.!/"-
49 57 53 20 59 45 4B 20 6E 72 75 54     0104    133                 "Turn KEY SWITCH to STAND BY or open BREAKER momentarily..."
44 4E 41 54 53 20 6F 74 20 48 43 54     0110
20 6E 65 70 6F 20 72 6F 20 59 42 20     011C
65 6D 6F 6D 20 52 45 4B 41 45 52 42     0128
         2E 2E 2E 79 6C 69 72 61 74 6E  0134
                              AA        0093
                                        013E    134 ;-----------------------------------------------------------------------
                                        013E    135 T_MESSAGE_2:
6B 63 65 68 43 20 64 61 42 2F 21 00'    013E    136         .ASCIC  "!/Bad Checksum Subtest - This subtest will restart itself!/"-
20 74 73 65 74 62 75 53 20 6D 75 73     014A
65 74 62 75 73 20 73 69 68 54 20 2D     0156
74 73 65 72 20 6C 6C 69 77 20 74 73     0162
2F 21 66 6C 65 73 74 69 20 74 72 61     016E
69 20 42 50 52 20 65 68 74 20 66 69     017A    137                 "if the RPB is properly interpreted.!/"-
69 20 79 6C 72 65 70 6F 72 70 20 73     0186
21 2E 64 65 74 65 72 70 72 65 74 6E     0192
                              2F        019E
49 57 53 20 59 45 4B 20 6E 72 75 54     019F    138                 "Turn KEY SWITCH to STAND BY or open BREAKER momentarily..."
44 4E 41 54 53 20 6F 74 20 48 43 54     01AB
20 6E 65 70 6F 20 72 6F 20 59 42 20     01B7
65 6D 6F 6D 20 52 45 4B 41 45 52 42     01C3
         2E 2E 2E 79 6C 69 72 61 74 6E  01CF
                              9A        013E
                                        01D9    139 ;-----------------------------------------------------------------------
                                        01D9    140 T_MESSAGE_3:
6F 66 20 68 63 72 61 65 53 2F 21 00'    01D9    141         .ASCIC  "!/Search for Good RPB - This subtest will restart itself!/"-
2D 20 42 50 52 20 64 6F 6F 47 20 72     01E5
```

```
73 65 74 62 75 73 20 73 69 68 54 20  01F1
61 74 73 65 72 20 6C 6C 69 77 20 74  01FD
   2F 21 66 6C 65 73 74 69 20 74 72  0209
69 20 42 50 52 20 65 68 74 20 66 69  0214    142                 "if the RPB is properly interpreted.!/"-
69 20 79 6C 72 65 70 6F 72 70 20 73  0220
21 2E 64 65 74 65 72 70 72 65 74 6E  022C
                                 2F  0238
49 57 53 20 59 45 4B 20 6E 72 75 54  0239    143                 "Turn KEY SWITCH to STAND BY or open BREAKER momentarily..."
44 4E 41 54 53 20 6F 74 20 48 43 54  0245
20 6E 65 70 6F 20 72 6F 20 59 42 20  0251
65 6D 6F 6D 20 52 45 4B 41 45 52 42  025D
      2E 2E 2E 79 6C 69 72 61 74 6E  0269
                              99 01D9
                                 0273    144 ;-----------------------------------------------------------------------
                                 0273    145
                                 0273    146 .SBTTL  ERROR MESSAGES
                                 0273    147 ;-----------------------------------------------------------------------
                                 0273    148 T_PWRDN_ERROR:
20 6E 77 6F 64 20 72 65 77 6F 50 00' 0273    149         .ASCIC  "Power down sequence did not complete"
64 69 64 20 65 63 6E 65 71 65 73  027F
74 65 6C 70 6D 6F 63 20 74 6F 6E 20  028B
                                 65  0297
                                 24  0273
                                 0298    150
                                 0298    151 ;-----------------------------------------------------------------------
                                 0298    152 T_PWRUP_ERROR:
65 73 20 70 75 20 72 65 77 6F 50 00' 0298    153         .ASCIC  "Power up sequence did not complete"
6E 20 64 69 64 20 65 63 6E 65 75 71  02A4
   65 74 65 6C 70 6D 6F 63 20 74 6F  02B0
                                 22  0298
                                 02BB    154
                                 02BB    155 ;-----------------------------------------------------------------------
                                 02BB    156 T_BADMODE:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 02BB    157         .ASCIC  "Wrong mode in PSL for power down and up"
70 20 72 6F 66 20 4C 53 50 20 6E 69  02C7
6E 61 20 6E 77 6F 64 20 72 65 77 6F  02D3
                        70 75 20 64  02DF
                                 27  02BB
                                 02E3    158
                                 02E3    159 ;-----------------------------------------------------------------------
                                 02E3    160 T_BADMODE_DN:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 02E3    161         .ASCIC  "Wrong mode in PSL for power down sequence"
70 20 72 6F 66 20 4C 53 50 20 6E 69  02EF
65 73 20 6E 77 6F 64 20 72 65 77 6F  02FB
                     65 63 6E 65 75  0307
                                 29  02E3
                                 030D    162
                                 030D    163 ;-----------------------------------------------------------------------
                                 030D    164 T_BADMODE_UP:
20 65 64 6F 6D 20 67 6E 6F 72 57 00' 030D    165         .ASCIC  "Wrong mode in PSL for power up sequence"
70 20 72 6F 66 20 4C 53 50 20 6E 69  0319
75 71 65 73 20 70 75 20 72 65 77 6F  0325
                        65 63 6E 65  0331
                                 27  030D
                                 0335    166
                                 0335    167 ;-----------------------------------------------------------------------
                                 0335    168 T_BADSTACK:
```

```
73 20 74 70 75 72 72 65 74 6E 49 00' 0335    169          .ASCIC  "Interrupt stack not used for power down and up"
65 73 75 20 74 6F 6E 20 6B 63 61 74  0341
20 72 65 77 6F 70 20 72 6F 66 20 64  034D
   70 75 20 64 6E 61 20 6E 77 6F 64  0359
                             2E      0335
                                     0364    170
                                     0364    171 ;-----------------------------------------------------------------------
                                     0364    172 T_BADSTACK_DN:
73 20 74 70 75 72 72 65 74 6E 49 00' 0364    173          .ASCIC  "Interrupt stack not used for power down sequence"
65 73 75 20 74 6F 6E 20 6B 63 61 74  0370
20 72 65 77 6F 70 20 72 6F 66 20 64  037C
63 6E 65 75 71 65 73 20 6E 77 6F 64  0388
                             65      0394
                             30      0364
                                     0395    174
                                     0395    175 ;-----------------------------------------------------------------------
                                     0395    176 T_BADSTACK_UP:
73 20 74 70 75 72 72 65 74 6E 49 00' 0395    177          .ASCIC  "Interrupt stack not used for power up sequence"
65 73 75 20 74 6F 6E 20 6B 63 61 74  03A1
20 72 65 77 6F 70 20 72 6F 66 20 64  03AD
   65 63 6E 65 75 71 65 73 20 70 75  03B9
                             2E      0395
                                     03C4    178
                                     03C4    179 ;-----------------------------------------------------------------------
                                     03C4    180 T_CHECKSUM_ERR:
61 62 20 68 74 69 77 20 42 50 52 00' 03C4    181          .ASCIC  "RPB with bad checksum used to restart"
75 20 6D 75 73 6B 63 65 68 63 20 64  03D0
61 74 73 65 72 20 6F 74 20 64 65 73  03DC
                                74 72 03E8
                             25      03C4
                                     03EA    182 ;-----------------------------------------------------------------------
```

```
                                03EA   184 .SBTTL  SUBROUTINES
                                03EA   185
                                03EA   186 ; This is the Power Fail interrupt service routine which is entered
                                03EA   187 ; upon an interrupt due to loss of power. The internal processor
                                03EA   188 ; registers, frame pointer, stack pointer, argument pointer and
                                03EA   189 ; all general purpose registers are saved in memory before all power
                                03EA   190 ; is lost. If this is completed sucessfully a flag is set and then
                                03EA   191 ; this routine goes into a loop, waiting for power to be totaly lost.
                                03EA   192
                                03EA   193          .ALIGN  LONG
                                03EC   194 POWER_DOWN:
      58   00000000'FF   DE   03EC   195          MOVAL   @A_PWR_BUFFER,R8        ; load R8 with address of buffer
           FC19 CF       DC   03F3   196          MOVPSL  L_PSL                   ; save contents of the current PSL
           02   18       EC   03F7   197          CMPV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
        00   FC13 CF            03FA   198                  L_PSL,#PSL$K_KERNEL     ; check PSL for current mode = kernel
                  05   13   03FE   199          BEQL    1$                      ; br = kernel mode
           FC19 CF    01   88   0400   200          BISB2   #M_BADMODE_DN,B_FLAGS2  ; flag bad mode on power down
        15 FC06 CF    1A   E0   0405   201 1$:      BBS     #PSL$V_IS,L_PSL,2$      ; br = on interrupt stack
           FC0E CF    04   88   040B   202          BISB2   #M_BADSTACK_DN,B_FLAGS2 ; flag bad stack on power down
  FBFB CF   00000000'8F   DB   0410   203          MFPR    #PR$_ISP,L_IS_SP        ; save interrupt stack pointer
           FBFA CF    5E   D0   0419   204          MOVL    SP,L_KS_SP              ; save kernel stack pointer
                      0E   11   041E   205          BRB     3$                      ; go save other processor registers
           FBEF CF    5E   D0   0420   206 2$:      MOVL    SP,L_IS_SP              ; save interrupt stack pointer
  FBEA CF   00000000'8F   DB   0425   207          MFPR    #PR$_KSP,L_KS_SP        ; save kernel stack pointer
                88    5C   D0   042E   208 3$:      MOVL    AP,(R8)+                ; save AP
                88    5D   D0   0431   209          MOVL    FP,(R8)+                ; save FP
        88   00000000'8F   DB   0434   210          MFPR    #PR$_ESP,(R8)+          ; save processor registers
        88   00000000'8F   DB   043B   211          MFPR    #PR$_SSP,(R8)+
        88   00000000'8F   DB   0442   212          MFPR    #PR$_USP,(R8)+
        88   00000000'8F   DB   0449   213          MFPR    #PR$_SBR,(R8)+
        88   00000000'8F   DB   0450   214          MFPR    #PR$_SLR,(R8)+
        88   00000000'8F   DB   0457   215          MFPR    #PR$_PCBB,(R8)+
        88   00000000'8F   DB   045E   216          MFPR    #PR$_P0BR,(R8)+
        88   00000000'8F   DB   0465   217          MFPR    #PR$_P0LR,(R8)+
        88   00000000'8F   DB   046C   218          MFPR    #PR$_P1BR,(R8)+
        88   00000000'8F   DB   0473   219          MFPR    #PR$_P1LR,(R8)+
        88   00000000'8F   DB   047A   220          MFPR    #PR$_ASTLVL,(R8)+
        88   00000000'8F   DB   0481   221          MFPR    #PR$_SCBB,(R8)+
     FB73 CF    58   D0   0488   222          MOVL    R8,L_BUF_BEGIN          ; save address of data in memory
     FB8B CF    01   88   048D   223          BISB2   #M_PWRDN_DONE,B_FLAGS   ; set done flag
                     0492   224 4$:      $DS_BREAK
 00010058 9F    6E   FA   0492                      CALLG   (SP), @#DS$BREAK
                F7   11   0499   225          BRB     4$                      ; wait
                     049B   226
                     049B   227
                     049B   228
                     049B   229 ; This is the Power Up routine which is pointed to by either the Restart
                     049B   230 ; Parameter Block (RPB) or is entered through the restart code when a
                     049B   231 ; nonrecoverable power fail has occurred. This routine will restore all
                     049B   232 ; of the information which was saved in the power down routine and also
                     049B   233 ; sets a done flag to indicate that the power up sequence was sucessful.
                     049B   234
                     049B   235 POWER_UP_ERR:
     FB7E CF    10   C8   049B   236          BISL2   #M_PWRUP_ERR,B_FLAGS2   ; set error flag if starting here
                     04A0   237 POWER_UP:
     58   FB5C CF    D0   04A0   238          MOVL    L_BUF_BEGIN,R8          ; load address of data
 00000000'8F    78   DA   04A5   239          MTPR    -(R8),#PR$_SCBB         ; load SCBB to handle any exceptions
```

```
                FB60 CF    DC  04AC   240          MOVPSL  L_PSL                          ; save current PSL
                02   18    EC  04B0   241          CMPV    #PSL$V_CURMOD,#PSL$S_CURMOD,-
          00    FB5A CF          04B3   242                  L_PSL,#PSL$K_KERNEL          ; check current mode = kernel
                     05    13  04B7   243          BEQL    1$                           ; br = current mode kernel
             FB60 CF    02    88  04B9   244          BISB2   #M_BADMODE_UP,B_FLAGS2       ; flag bad mode on power up
       15    FB4D CF    1A    E0  04BE   245 1$:      BBS     #PSL$V_IS,L_PSL,2$           ; br = on interrupt stack
             FB55 CF    08    88  04C4   246          BISB2   #M_BADSTACK_UP,B_FLAGS2      ; flag bad stack on power up
00000000'8F  FB47 CF    DA  04C9   247          MTPR    L_IS_SP,#PR$_ISP             ; load interrupt SP
          5E FB42 CF    D0  04D2   248          MOVL    L_KS_SP,SP                   ; load kernel SP
                     17    11  04D7   249          BRB     3$                           ; restore other processor registers
00000000'8F  FB37 CF    DA  04D9   250 2$:      MTPR    L_IS_SP,#PR$_ISP             ; load interrupt SP
          5E FB2E CF    D0  04E2   251          MOVL    L_IS_SP,SP                   ; load interrupt SP
00000000'8F  FB2D CF    DA  04E7   252          MTPR    L_KS_SP,#PR$_KSP             ; load kernel SP
     00000000'8F    78    DA  04F0   253 3$:      MTPR    -(R8),#PR$_ASTLVL            ; restore other processor registers
     00000000'8F    78    DA  04F7   254          MTPR    -(R8),#PR$_P1LR
     00000000'8F    78    DA  04FE   255          MTPR    -(R8),#PR$_P1BR
     00000000'8F    78    DA  0505   256          MTPR    -(R8),#PR$_P0LR
     00000000'8F    78    DA  050C   257          MTPR    -(R8),#PR$_P0BR
     00000000'8F    78    DA  0513   258          MTPR    -(R8),#PR$_PCBB
     00000000'8F    78    DA  051A   259          MTPR    -(R8),#PR$_SLR
     00000000'8F    78    DA  0521   260          MTPR    -(R8),#PR$_SBR
     00000000'8F    78    DA  0528   261          MTPR    -(R8),#PR$_USP
     00000000'8F    78    DA  052F   262          MTPR    -(R8),#PR$_SSP
     00000000'8F    78    DA  0536   263          MTPR    -(R8),#PR$_ESP
                5D  78    D0  053D   264          MOVL    -(R8),FP                     ; restore FP
                5C  78    D0  0540   265          MOVL    -(R8),AP                     ; restore AP
                           0543   266
 22 FAD6 CF    00    E0  0543   267          BBS     #V_BADMODE_DN,B_FLAGS2,5$ ; br = bad mode on power down
 5A FAD0 CF    01    E1  0549   268          BBC     #V_BADMODE_UP,B_FLAGS2,9$ ; br = both modes good
                           054F   269          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADMODE_UP
                     00    DD  054F                   PUSHL   #0
             FDB8 CF    DF  0551                   PUSHAL  T_BADMODE_UP
     00000000'EF    DD  0555                   PUSHL   KA_UNIT_NO
                     01    DD  0558                   PUSHL   #$ER
                           055D       ;;; TEST 2, SUBTEST 0, ERROR 1
000100D0 9F    04    FB  055D                   CALLS   #$$M, @#DS$ERRHARD
                           0564   270         $DS_ABORT
00010020 9F    6C    FA  0564                   CALLG   (AP), @#DS$ABORT
 1C FAAE CF    01    E1  056B   271 5$:      BBC     #V_BADMODE_UP,B_FLAGS2,7$ ; br = bad mode on power down only
                           0571   272          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADMODE
                     00    DD  0571                   PUSHL   #0
             FD44 CF    DF  0573                   PUSHAL  T_BADMODE
     00000000'EF    DD  0577                   PUSHL   KA_UNIT_NO
                     02    DD  057D                   PUSHL   #$ER
                           057F       ;;; TEST 2, SUBTEST 0, ERROR 2
000100D0 9F    04    FB  057F                   CALLS   #$$M, @#DS$ERRHARD
                           0586   273         $DS_ABORT
00010020 9F    6C    FA  0586                   CALLG   (AP), @#DS$ABORT
                           058D   274 7$:      $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADMODE_DN
                     00    DD  058D                   PUSHL   #0
             FD50 CF    DF  058F                   PUSHAL  T_BADMODE_DN
     00000000'EF    DD  0593                   PUSHL   KA_UNIT_NO
                     03    DD  0599                   PUSHL   #$ER
                           059B       ;;; TEST 2, SUBTEST 0, ERROR 3
000100D0 9F    04    FB  059B                   CALLS   #$$M, @#DS$ERRHARD
                           05A2   275         $DS_ABORT
00010020 9F    6C    FA  05A2                   CALLG   (AP), @#DS$ABORT
```

```
                 22 FA70 CF    02    E0  05A9   276 9$:     BBS     #V_BADSTACK_DN,B_FLAGS2,11$ ; br = bad stack on power down
                 5A FA6A CF    03    E1  05AF   277         BBC     #V_BADSTACK_UP,B_FLAGS2,15$ ; br = both stacks good
                                           05B5   278         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADSTACK_UP
                                     00    DD  05B5                   PUSHL   #0
                                   FDDA CF     DF  05B7                   PUSHAL  T_BADSTACK_UP
                          00000000'EF          DD  05BB                   PUSHL   KA_UNIT_NO
                                     04    DD  05C1                   PUSHL   #$ER
                                           05C3       ;;; TEST 2, SUBTEST 0, ERROR 4
                          000100D0 9F    04    FB  05C3                   CALLS   #$$M, @#DS$ERRHARD
                                           05CA   279         $DS_ABORT
                          00010020 9F    6C    FA  05CA                   CALLG   (AP), @#DS$ABORT
                           1C FA48 CF    03    E1  05D1   280 11$:    BBC     #V_BADSTACK_UP,B_FLAGS2,13$ ; br = bad stack power down only
                                           05D7   281         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADSTACK
                                     00    DD  05D7                   PUSHL   #0
                                   FD58 CF     DF  05D9                   PUSHAL  T_BADSTACK
                          00000000'EF          DD  05DD                   PUSHL   KA_UNIT_NO
                                     05    DD  05E3                   PUSHL   #$ER
                                           05E5       ;;; TEST 2, SUBTEST 0, ERROR 5
                          000100D0 9F    04    FB  05E5                   CALLS   #$$M, @#DS$ERRHARD
                                           05EC   282         $DS_ABORT
                          00010020 9F    6C    FA  05EC                   CALLG   (AP), @#DS$ABORT
                                           05F3   283 13$:    $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_BADSTACK_DN
                                     00    DD  05F3                   PUSHL   #0
                                   FD6B CF     DF  05F5                   PUSHAL  T_BADSTACK_DN
                          00000000'EF          DD  05F9                   PUSHL   KA_UNIT_NO
                                     06    DD  05FF                   PUSHL   #$ER
                                           0601       ;;; TEST 2, SUBTEST 0, ERROR 6
                          000100D0 9F    04    FB  0601                   CALLS   #$$M, @#DS$ERRHARD
                                           0608   284         $DS_ABORT
                          00010020 9F    6C    FA  0608                   CALLG   (AP), @#DS$ABORT
                              FA09 CF    02    88  060F   285 15$:    BISB2   #M_PWRUP_DONE,B_FLAGS   ; set done flag
                00000000'8F    00000F03 8F    DA  0614   286         MTPR    #^XF03,#PR$_TXDB        ; clear warm start flag
                                     02    02  061F   287         REI                             ; return to test
                                           0620   288
                                           0620   289
                                           0620   290 ; This routine will calculate the checksum for the power up routine to
                                           0620   291 ; be stored in the restart parameter block.
                                           0620   292 ;
                                           0620   293 ;       R6 contains the address of the power up routine
                                           0620   294 ;       R7 gets the checksum for that routine
                                           0620   295 ;
                                           0620   296
                                           0620   297 CALC_CHECKSUM:
                                     54    D4  0620   298         CLRL    R4                      ; init R4 to store checksum
                                 F9DD CF     01    D0  0622   299         MOVL    #1,L_COUNT              ; initialize counter
                                     53    56    D0  0627   300         MOVL    R6,R3                   ; get address of power up routine
                                     54    83    C0  062A   301 1$:     ADDL2   (R3)+,R4                ; add long word checksum
                                F7 F9D2 CF     1F    F3  062D   302         AOBLEQ  #^X1F,L_COUNT,1$        ; br = not finished
                                     57    54    D0  0633   303         MOVL    R4,R7                   ; save checksum
                                           05  0636   304         RSB                             ; return
                                           0637   305
                                           0637   306 ; This routine will clear the first page of buffer area.
                                           0637   307
                                           0637   308 CLEAR_BUFFER:
                          55   00000000'FF    DE  0637   309         MOVAL   @A_BUFFER_BEGIN,R5      ; get starting buffer address
                                 F9C2 CF     D4  063E   310         CLRL    L_COUNT                 ; init counter
                                     85    D4  0642   311 1$:     CLRL    (R5)+                   ; clear buffer location
```

```
      F4 F9B7 CF    00000080 8F    F3  0644    312          AOBLEQ  #128,L_COUNT,1$          ; br = not finished
                                   05  064E    313          RSB                              ; return
                                       064F    314
                                       064F    315
                                       064F    316          $DS_PAGE
```

```
                                064F                    .SBTTL  TEST 2:   POWER FAIL
                               00000000                 .PSECT  TEST_002, PAGE, NOWRT
                                   0010
                                   0010    319 ;++
                                   0010    320 ; Test description:
                                   0010    321 ;
                                   0010    322 ;       This test consists of a series of manual operator intervention
                                   0010    323 ;       subtests. These subtests will check that the Restart Parameter Block
                                   0010    324 ;       (RPB) is recognizable and initiates the proper sequence of events
                                   0010    325 ;       during and after a power failure. The power fail is induced by turning
                                   0010    326 ;       the key switch on the front panel of the 11/730 to the STAND BY
                                   0010    327 ;       position or by oper ng the BREAKER. With the battery backup on and the
                                   0010    328 ;       select switch in th  restart position a warm restart will be attempted
                                   0010    329 ;       after the power is restored.
                                   0010    330 ;
                                   0010    331 ;
                                   0010    332 ; Assumptions:
                                   0010    333 ;
                                   0010    334 ;       For this test to function properly the front panel AUTO RESTART
                                   0010    335 ;       switch must be in the ON position and the battery backup must
                                   0010    336 ;       be on.
                                   0010    337 ;
                                   0010    338 ;--
                                   0010    339
                                   0010    340         $DS_BGNTEST     <MANUAL,POWER,ALL>
                                   0010        DATA_002:
                         00000000  0010                .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                   0014        TEST_002::
                             0000  0014                .WORD   ^M<>            ; ENTRY MASK
                                   0016    341
                 00000000'EF   D5  0016    342         TSTL    KA_PTABLE                       ; CPU selected?
                               10  18  001C    343         BGEQ    5$                              ; br = CPU selected
                                   001E    344         $DS_PRINTF_S    FORMAT=T_NO_KA730       ; print exit warning
                      00000025'EF  9F  001E                PUSHAB  T_NO_KA730
                 000100F0 9F       01  FB  0024                CALLS   #$$N, @#DS$PRINTF
                                   002B    345         $DS_EXIT        TEST                    ; exit test
                                 0248  31  002B                BRW     TEST_002_X      ; EXIT TEST 2
                                   002E    346 5$:
                                   002E    347         $DS_SETVEC_S    VECTOR=#SCB$L_POWER, SRVADR=POWER_DOWN, CODE=#1
                                   01  DD  002E                PUSHL   #1
                      000003EC'EF  DF  0030                PUSHAL  POWER_DOWN
                                   0C  DD  0036                PUSHL   #SCB$L_POWER
                 00010160 9F       03  FB  0038                CALLS   #3, @#DS$SETVEC
00000000'8F   00000F03 8F  DA  003F  348         MTPR    #^XF03,#PR$_TXDB ; clear warm start flag
              56   000004A0'EF  DE  004A  349         MOVAL   POWER_UP,R6     ; load address of good power up routine
                   00000620'EF  16  0051  350         JSB     CALC_CHECKSUM   ; calculate good checksum for RPB.
              00000008'EF   57  D0  0057  351         MOVL    R7,L_CHECKSUM   ; load checksum for good power up
              56   0000049B'EF  DE  005E  352         MOVAL   POWER_UP_ERR,R6 ; load address of error power up routine
                   00000620'EF  16  0065  353         JSB     CALC_CHECKSUM   ; calculate good checksum for RPB.
              0000000C'EF   57  D0  006B  354         MOVL    R7,L_CHECKSUM1  ; load checksum for error power up
                                   0072  355         $DS_PRINTF_S    FORMAT=T_INST
                   0000005A'EF  9F  0072                PUSHAB  T_INST
                 000100F0 9F       01  FB  0078                CALLS   #$$N, @#DS$PRINTF
                                   007F  356         $DS_PAGE
```

```
                                                        .SBTTL  SUBTEST 2.1:  GOOD RPB SUBTEST
                                                    ;+
                                                    ; SUBTEST DESCRIPTION:
                                                    ;
                                                    ;       This subtest sets up a good RPB at location zero, which points to
                                                    ;       a power up routine that restores all pertinent processor registers,
                                                    ;       the argument pointer, frame pointer and stack pointer. The operator
                                                    ;       is instructed through the console terminal to turn the key switch
                                                    ;       on the front panel to the STAND BY position and then back to the local
                                                    ;       position or open the BREAKER to induce a CPU power failure. If the RPB
                                                    ;       is interpreted properly the instrucions for the next subtest will be
                                                    ;       printed on the console terminal. If an error occurred, either the
                                                    ;       processor will unexpectedly reboot or an error report will be generated
                                                    ;       by the diagnostic supervisor. This means the RPB was not recognized and
                                                    ;       a cold restart has been attempted.
                                                    ;
                                                    ;
                                                    ; SUBTEST STEPS:
                                                    ;
                                                    ;       SUBTEST GOOD RPB
                                                    ;       :  Setup good RPB at location 0
                                                    ;       :  Send instructions to operator
                                                    ;       :  Recover from power fail
                                                    ;       :  IF errors
                                                    ;       :  :  THEN REPORT errors
                                                    ;       :  ENDIF
                                                    ;       ENDSUBTEST GOOD RPB
                                                    ;
                                                    ;
                                                    ;
                                                    ;-
                                                            $DS_BGNSUB
                                                        T2_S1::
00010030 9F    00000000'EF    FA                            CALLG   $S$, @#DS$BGNSUB
         00000637'EF    16   008A  391                JSB     CLEAR_BUFFER    ; clear buffer area
         0000001D'EF    94   0090  392                CLRB    B_FLAGS         ; initialize flags
         0000001E'EF    94   0096  393                CLRB    B_FLAGS2        ; init flags2
             55    00   D0   009C  394                MOVL    #0,R5           ; load address to build RPB
             85    00   D0   009F  395                MOVL    #0,(R5)+        ; load first longword with the address
      85 000004A0'EF    DE   00A2  396                MOVAL   POWER_UP,(R5)+  ; load address of restart routine
      85 00000008'EF    D0   00A9  397                MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                 65    D4   00B0  398                CLRL    (R5)            ; clear warm start flag
                                00B2  399                $DS_PRINTF_S    FORMAT=T_MESSAGE_1
         00000093'EF    9F   00B2                             PUSHAB  T_MESSAGE_1
      000100F0 9F    01   FB   00B8                             CALLS   #$$N, @#DS$PRINTF
                                00BF  400
                                00BF  401 POWER_DOWN_1:
                                00BF  402                $DS_BREAK
      00010058 9F    6E   FA   00BF                             CALLG   (SP), @#DS$BREAK
   02 0000001D'EF    01   E0   00C6  403                BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up done
                     EF   11   00CE  404                BRB     POWER_DOWN_1    ; wait for power fail interrupt
                                00D0  405
   1E 0000001D'EF    00   E0   00D0  406 1$:            BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                00D8  407                $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_PWRDN_ERROR
                     00   DD   00D8                             PUSHL   #0
             00000273'EF    DF   00DA                             PUSHAL  T_PWRDN_ERROR
```

```
                  00000000'EF   DD  00E0                        PUSHL   KA_UNIT_NO
                              01  DD  00E6                        PUSHL   #$ER
                                  00E8          ;;; TEST 2, SUBTEST 1, ERROR 1
                  000100D0 9F   04  FB  00E8                        CALLS   #$$M, @#DS$ERRHARD
                                  00EF   408          $DS_ABORT
                  00010020 9F   6C  FA  00EF                        CALLG   (AP), @#DS$ABORT
                                  00F6   409 3$:
                                  00F6   410          $DS_ENDSUB
                                  00F6           T2_S1_X:
      00010038 9F   00000000'EF   FA  00F6                        CALLG   $S$, @#DS$ENDSUB
                                  0101   411          $DS_PAGE
                                  0101   412
```

```
                                0101                    .SBTTL  SUBTEST 2.2:  SEARCH FOR GOOD RPB
                                0101    415 ;+
                                0101    416 ; SUBTEST DESCRIPTION:
                                0101    417 ;
                                0101    418 ;       This subtest clears the first 1F (^X) memory locations starting with
                                0101    419 ;       zero to assure that the string of zeros at location 0 will not be
                                0101    420 ;       misinterpreted as a RPB even though all zeros fills basic criteria for
                                0101    421 ;       a RPB (i.e., the starting address and checksum of all zeros would be
                                0101    422 ;       good).  It also sets up a good RPB at a higher page boundary which
                                0101    423 ;       points to a power up routine that restores all pertinent processor
                                0101    424 ;       registers, the argument pointer, frame pointer and stack pointer.
                                0101    425 ;       This will test a search through memory to find a valid RPB.
                                0101    426 ;       The operator is instructed through the console terminal to turn the
                                0101    427 ;       key switch on the front panel to the STAND BY position or open the
                                0101    428 ;       BREAKER to induce a CPU power failure. If location zero is misinter-
                                0101    429 ;       preted as a RPB the processor will halt with code = 1. If the real RPB
                                0101    430 ;       is interpreted properly the next subtest instructions will be printed
                                0101    431 ;       on the console terminal. If an error occurred, either the processor
                                0101    432 ;       will reboot unexpectedly or an error report will be generated by the
                                0101    433 ;       diagnostic supervisor.
                                0101    434 ;
                                0101    435 ;
                                0101    436 ; SUBTEST STEPS:
                                0101    437 ;
                                0101    438 ;       SUBTEST SEARCH FOR GOOD RPB
                                0101    439 ;       :  Clear first 1F (X) locations
                                0101    440 ;       :  Setup good RPB at higher page boundary
                                0101    441 ;       :  Send instructions to operator
                                0101    442 ;       :  Recover from Power Fail
                                0101    443 ;       :  IF any errors
                                0101    444 ;       :  :  THEN REPORT errors
                                0101    445 ;       :  ENDIF
                                0101    446 ;       ENDSUBTEST SEARCH FOR GOOD RPB
                                0101    447 ;
                                0101    448 ;
                                0101    449 ;-
                                0101    450
                                0101    451             $DS_BGNSUB
                                0101            T2_S2::
00010030 9F    0000000C'EF   FA  0101                    CALLG   $S$, @#DS$BGNSUB
               00000637'EF   16  010C    452             JSB     CLEAR_BUFFER    ; clear buffer area
               0000001D'EF   94  0112    453             CLRB    B_FLAGS         ; initialize flags
               0000001E'EF   94  0118    454             CLRB    B_FLAGS2        ; init flags2
                       54    D4  011E    455             CLRL    R4              ; init counter
                    55   00    D0  0120    456             MOVL    #0,R5           ; load address to start clearing
                         85    7C  0123    457 1$:         CLRQ    (R5)+           ; clear a quad word
                     FA 54     10  F3  0125    458         AOBLEQ  #16,R4,1$       ; br = not finished
          55   00000000'FF      DE  0129    459             MOVAL   @A_BUFFER_BEGIN,R5 ; load address to build good RPB
          85   00000000'FF      DE  0130    460             MOVAL   @A_BUFFER_BEGIN,(R5)+ ; load first longword with the address
          85   000004A0'EF      DE  0137    461             MOVAL   POWER_UP,(R5)+  ; load address of restart routine
          85   00000008'EF      D0  013E    462             MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                         65    D4  0145    463             CLRL    (R5)            ; clear warm start flag
                                   0147    464             $DS_PRINTF_S    FORMAT=T_MESSAGE_3
               000001D9'EF   9F  0147                    PUSHAB  T_MESSAGE_3
          000100F0 9F    01    FB  014D                    CALLS   #$$N, @#DS$PRINTF
                                   0154    465
                                   0154    466 POWER_DOWN_2:
```

```
                                0154   467          $DS_BREAK
          00010058 9F   6E    FA 0154                      CALLG   (SP), @#DS$BREAK
       02 0000001D'EF   01    E0 015B   468          BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                        EF    11 0163   469          BRB     POWER_DOWN_2    ; wait for power fail interrupt
                                0165   470
       1E 0000001D'EF   00    E0 0165   471 1$:      BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                016D   472          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_PWRDN_ERROR
                        00    DD 016D                      PUSHL   #0
                00000273'EF   DF 016F                      PUSHAL  T_PWRDN_ERROR
                00000000'EF   DD 0175                      PUSHL   KA_UNIT_NO
                        01    DD 017B                      PUSHL   #$ER
                                017D       ;;; TEST 2, SUBTEST 2, ERROR 1
          000100D0 9F   04    FB 017D                      CALLS   #$$M, @#DS$ERRHARD
                                0184   473          $DS_ABORT
          00010020 9F   6C    FA 0184                      CALLG   (AP), @#DS$ABORT
                                018B   474 3$:
                                018B   475          $DS_ENDSUB
                                018B       T2_S2_X:
    00010038 9F   0000000C'EF   FA 018B                      CALLG   $S$, @#DS$ENDSUB
                                0196   476
                                0196   477          $DS_PAGE
```

```
                                0196                    .SBTTL  SUBTEST 2.3:  BAD CHECKSUM SUBTEST
                                0196    480 ;+
                                0196    481 ; SUBTEST DESCRIPTION:
                                0196    482 ;
                                0196    483 ;       This subtest sets up a RPB at location zero with a bad checksum.
                                0196    484 ;       It points to a power up routine that restores all pertinent processor
                                0196    485 ;       registers, the argument pointer, frame pointer and stack pointer and
                                0196    486 ;       will set an error flag indicating that the checksum did not invalidate
                                0196    487 ;       the RPB. Further into memory a good RPB will be located to warm restart
                                0196    488 ;       normally without setting an error flag (this prevents an auto cold
                                0196    489 ;       start). The operator is instructed through the console terminal to turn
                                0196    490 ;       the key switch on the front panel to the STAND BY position or open the
                                0196    491 ;       BREAKER to induce a CPU power failure. If the second RPB is not inter-
                                0196    492 ;       preted properly, the system will attempt a cold restart and boot. If
                                0196    493 ;       the RPB with the bad checksum causes the warm start an error will be
                                0196    494 ;       reported.
                                0196    495 ;
                                0196    496 ;
                                0196    497 ; SUBTEST STEPS:
                                0196    498 ;
                                0196    499 ;       SUBTEST BAD CHECKSUM
                                0196    500 ;       :  Setup RPB at location 0 with bad checksum
                                0196    501 ;       :  Setup good RPB at higher page boundary
                                0196    502 ;       :  Send instructions to operator
                                0196    503 ;       :  Recover from  power fail
                                0196    504 ;       :  IF errors
                                0196    505 ;       :  :  THEN REPORT errors
                                0196    506 ;       :  ENDIF
                                0196    507 ;       ENDSUBTEST BAD CHECKSUM
                                0196    508 ;
                                0196    509 ;
                                0196    510 ;-
                                0196    511
                                0196    512             $DS_BGNSUB
                                0196        T2_S3::
00010030 9F   00000018'EF   FA  0196                    CALLG   $S$, @#DS$BGNSUB
              00000637'EF   16  01A1    513             JSB     CLEAR_BUFFER    ; clear buffer area
              0000001D'EF   94  01A7    514             CLRB    B_FLAGS         ; initialize flags
              0000001E'EF   94  01AD    515             CLRB    B_FLAGS2        ; init flags2
                  55    00    D0  01B3    516             MOVL    #0,R5           ; load address to build RPB
                  85    00    D0  01B6    517             MOVL    #0,(R5)+        ; load first longword with the address
          85  0000049B'EF   DE  01B9    518             MOVAL   POWER_UP_ERR,(R5)+ ; load address of restart routine
          65  0000000C'EF   D0  01C0    519             MOVL    L_CHECKSUM1,(R5) ; load error restart routine checksum
                  85    01    C2  01C7    520             SUBL2   #1,(R5)+        ; make the checksum bad
                        65    D4  01CA    521             CLRL    (R5)            ; clear warm start flag
          55  00000000'FF   DE  01CC    522             MOVAL   @A_BUFFER_BEGIN,R5 ; load address to build good RPB
          85  00000000'FF   DE  01D3    523             MOVAL   @A_BUFFER_BEGIN,(R5)+ ; load first longword with the address
          85  000004A0'EF   DE  01DA    524             MOVAL   POWER_UP,(R5)+  ; load address of restart routine
          85  00000008'EF   D0  01E1    525             MOVL    L_CHECKSUM,(R5)+ ; load good restart routine checksum
                        65    D4  01E8    526             CLRL    (R5)            ; clear warm start flag
00000100'EF   0000001F'EF   B0  01EA    527             MOVW    JUMP,STRTUP     ; load addr 100 with JMP to POWER_UP to restart
                                01F5    528             $DS_PRINTF_S    FORMAT=T_MESSAGE_2
              0000013E'EF   9F  01F5                    PUSHAB  T_MESSAGE_2
          000100F0 9F   01    FB  01FB                    CALLS   #$$N, @#DS$PRINTF
                                0202    529
                                0202    530 POWER_DOWN_3:
                                0202    531             $DS_BREAK
```

```
                00010058 9F   6E   FA  0202                        CALLG   (SP), @#DS$BREAK
             02 0000001D'EF   01   E0  0209   532          BBS     #V_PWRUP_DONE,B_FLAGS,1$ ; br = power up sequence complete
                              EF   11  0211   533          BRB     POWER_DOWN_3    ; wait for power fail interrupt
                                       0213   534
             1E 0000001E'EF   04   E1  0213   535 1$:      BBC     #V_PWRUP_ERR,B_FLAGS2,2$ ; br = restart was from good RPB
                                       021B   536          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_CHECKSUM_ERR
                                   00  DD  021B                        PUSHL   #0
                        000003C4'EF    DF  021D                        PUSHAL  T_CHECKSUM_ERR
                        00000000'EF    DD  0223                        PUSHL   KA_UNIT_NO
                                   01  DD  0229                        PUSHL   #$ER
                                       022B       ;;; TEST 2, SUBTEST 3, ERROR 1
                00010DD0 9F    04   FB  022B                        CALLS   #$$M, @#DS$ERRHARD
                                       0232   537          $DS_ABORT
                00010020 9F    6C   FA  0232                        CALLG   (AP), @#DS$ABORT
             1E 0000001D'EF    00   E0  0239   538 2$:      BBS     #V_PWRDN_DONE,B_FLAGS,3$ ; br = power down sequence complete
                                       0241   539          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_PWRDN_ERROR
                                   00  DD  0241                        PUSHL   #0
                        00000273'EF    DF  0243                        PUSHAL  T_PWRDN_ERROR
                        00000000'EF    DD  0249                        PUSHL   KA_UNIT_NO
                                   02  DD  024F                        PUSHL   #$ER
                                       0251       ;;; TEST 2, SUBTEST 3, ERROR 2
                0001C0D0 9F    04   FB  0251                        CALLS   #$$M, @#DS$ERRHARD
                                       0258   540          $DS_ABORT
                00010020 9F    6C   FA  0258                        CALLG   (AP), @#DS$ABORT
                                       025F   541 3$:
                                       025F   542          $DS_ENDSUB
                                       025F       T2_S3_X:
    00010038 9F    00000018'EF    FA  025F                        CALLG   $S$, @#DS$ENDSUB
                                       026A   543
                                       026A   544          $DS_CLRVEC_S    VECTOR=#SCB$L_POWER
                                   0C  DD  026A                        PUSHL   #SCB$L_POWER
                00010168 9F    01   FB  026C                        CALLS   #1, @#DS$CLRVEC
                                       0273   545
                                       0273   546          $DS_ENDTEST
                          50   01   D0  0273                        MOVL    #1, R0          ; NORMAL EXIT
                                       0276       TEST_002_X::
                00010058 9F    6E   FA  0276                        CALLG   (SP), @#DS$BREAK
                                   04  027D                        RET                     ; RETURN TO TEST SEQUENCER
                                       027E   547
                                       027E   548          $DS_ENDMOD
                                 00000000                          .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                             00000002  0000                        .LONG   $TN
                                       0004   549
                                       0004   550          .END
```

```
$$$                 = 00000018 R      06      DS$K_NORMAL     = 00000001          DS$_NULLSTR     = 00660010
$$E                 = 00000001                DS$K_SEVERE     = 00000004          DS$_OVERFLOW    = 00660008
$$M                 = 00000004                DS$K_SUBSYS     = 00000066          DS$_POWER       = 00660098
$$N                 = 00000000                DS$K_WARNING    = 00000000          DS$_PROGERR     = 00660020
$$S                 = FFFFFFFF                DS$LOAD           00010198          DS$_SEVERE      = 00660004
$ENV                = 00000001  G             DS$LOADPCS        000101C0          DS$_TRANSL      = 006600A0
$ER                 = FFFFFFFF                DS$MAPDBGBLOCK    00010118          DS$_TRUNCATE    = 00660028
$MO                 = 00000001  G             DS$MMOFF          00010158          DS$_UNEXPINT    = 006600D8
$SS                 = 00000018 R      06      DS$MMON           00010150          DS$_VASFULL     = 00660048
$ST                 = 00000000                DS$MOVPHY         00010148          DS$_WARNING     = 00660000
$TN                 = 00000002                DS$MOVVRT         00010140          ENV$M_DOMAIN    = 00000002
ALL                 = 00000008  G             DS$PARSE          000100B8          ENV$M_LEVEL     = 00000001
A_BUFFER_BEGIN        ********    X   02      DS$PRINTB         000100E0          ENV$M_SUPER     = 000003FC
A_PWR_BUFFER          ********    X   02      DS$PRINTF         000100F0          ENV$S_DOMAIN    = 00000001
BASE_002              00000000 R      04      DS$PRINTS         000100F8          ENV$S_LEVEL     = 00000001
BIT...              = 00000005                DS$PRINTSIG       00010100          ENV$S_SUPER     = 00000008
B_FLAGS               0000001D R      02      DS$PRINTX         000100E8          ENV$V_DOMAIN    = 00000001
B_FLAGS2              0000001E R      02      DS$PROBE          000101A0          ENV$V_LEVEL     = 00000000
B_RESPONSE            0000001C R      02      DS$RELBUF         00010128          ENV$V_SUPER     = 00000002
CALC_CHECKSUM         00000620 R      02      DS$RELMEM         00010138          ENV$_CPU        = 00000000
CEP_FUNCTIONAL      = 00000000                DS$SETIPL         00010178          ENV$_FUNCTIONAL = 00000000
CEP_REPAIR          = 00000001                DS$SETMAP         00010188          ENV$_REPAIR     = 00000001
CLEAR_BUFFER          00000637 R      02      DS$SETPRIEXV      00010110          ENV$_SUPER      = 00000001
DATA_002              00000010 R      04      DS$SETVEC         00010160          ENV$_SYSTEM     = 00000001
DEFAULT             = 00000000  G             DS$SHOCHAN        00010190          HALT            = 00000004  G
DS$ABORT              00010020                DS$SUMMARY        00010028          IPR             = 00000006  G
DS$ASKADR             00010090                DS$WAITMS         00010060          JUMP              0000001F R      02
DS$ASKDATA            00010080                DS$WAITUS         00010068          KA_PTABLE         ********    X   04
DS$ASKLGCL            00010098                DS$_ARITH       = 006600D0          KA_UNIT_NO        ********    X   02
DS$ASKSTR             000100A0                DS$_BADLINK     = 006600F0          LDPCTX          = 0000000B  G
DS$ASKVLD             00010088                DS$_BADTYPE     = 006600E8          L_BUF_BEGIN       00000000 R      02
DS$ATTACH             000101A8                DS$_CHME        = 006600A8          L_CHECKSUM        00000008 R      02
DS$BGNSUB             00010030                DS$_CHMK        = 006600E0          L_CHECKSUM1       0000000C R      02
DS$BRANCH             000100A8                DS$_DEVNAME     = 00660108          L_COUNT           00000004 R      02
DS$BREAK              00010058                DS$_ERROR       = 00660002          L_IS_SP           00000014 R      02
DS$CANWAIT            00010070                DS$_FHWE        = 00660068          L_KS_SP           00000018 R      02
DS$CHANNEL            00010180                DS$_FRAGBUF     = 00660080          L_PSL             00000010 R      02
DS$CKLOOP             00010040                DS$_ICBUSY      = 006600C8          MANUAL          = 00000001  G
DS$CLRVEC             00010168                DS$_ICERR       = 006600C0          MCHK            = 00000005  G
DS$CNTRLC             00010078                DS$_IHWE        = 00660060          MEM             = 00000009  G
DS$CVTREG             000100B0                DS$_ILLCHAR     = 00660018          MSG_002           00000002 R      04
DS$ENDPASS            00010010                DS$_ILLPAGCNT   = 00660078          M_BADMODE_DN    = 00000001
DS$ENDSUB             00010038                DS$_ILLUNIT     = 00660100          M_BADMODE_UP    = 00000002
DS$ERRDEV             000100C8                DS$_INSFMEM     = 00660050          M_BADSTACK_DN   = 00000004
DS$ERRHARD            00010000                DS$_IPL2HI      = 006600B8          M_BADSTACK_UP   = 00000008
DS$ERRPREP            00010108                DS$_IVADDR      = 00660040          M_BAD_PC        = 00000004
DS$ERRSOFT            000100D8                DS$_IVVECT      = 00660038          M_PWRDN_DONE    = 00000001
DS$ERRSYS             000100C0                DS$_KRNLSTK     = 00660090          M_PWRUP_DONE    = 00000002
DS$ESCAPE             00010050                DS$_LOGIC       = 00660070          M_PWRUP_ERR     = 00000010
DS$FREEDBGSYM         000101B8                DS$_MCHK        = 00660088          PAR$M_ATDEF     = 00000010
DS$GETBUF             00010120                DS$_MMOFF       = 00660058          PAR$M_ATHI      = 00000008
DS$GETMEM             00010130                DS$_NEEDUNIT    = 006600F8          PAR$M_ATLO      = 00000004
DS$GPHARD             00010018                DS$_NOPCS       = 00660110          PAR$M_NODEF     = 00000002
DS$HELP               000101B0                DS$_NORMAL      = 00660001          PAR$V_ATDEF     = 00000004
DS$INITSCB            00010170                DS$_NOSUPPORT   = 006600B1          PAR$V_ATHI      = 00000003
DS$INLOOP             00010048                DS$_NOTDON      = 00660030          PAR$V_ATLO      = 00000002
DS$K_ERROR          = 00000002                DS$_NOTIMP      = 006600B0          PAR$V_NODEF     = 00000001
```

```
PAR$_BIN        = 00000002
PAR$_DEC        = 0000000A
PAR$_HEX        = 00000010
PAR$_NO         = 00000000
PAR$_OCT        = 00000008
PAR$_YES        = 00000001
POWER           = 00000003  G
POWER_DOWN        000003EC R      02
POWER_DOWN_1      000000BF R      04
POWER_DOWN_2      00000154 R      04
POWER_DOWN_3      00000202 R      04
POWER_UP          000004A0 R      02
POWER_UP_ERR      0000049B R      02
PR$_ASTLVL        ********   X    02
PR$_ESP           ********   X    02
PR$_ISP           ********   X    02
PR$_KSP           ********   X    02
PR$_P0BR          ********   X    02
PR$_P0LR          ********   X    02
PR$_P1BR          ********   X    02
PR$_P1LR          ********   X    02
PR$_PCBB          ********   X    02
PR$_SBR           ********   X    02
PR$_SCBB          ********   X    02
PR$_SLR           ********   X    02
PR$_SSP           ********   X    02
PR$_TXDB          ********   X    02
PR$_USP           ********   X    02
PSL$C_EXEC      = 00000001
PSL$C_KERNEL    = 00000000
PSL$C_SUPER     = 00000002
PSL$C_USER      = 00000003
PSL$K_EXEC      = 00000001
PSL$K_KERNEL    = 00000000
PSL$K_SUPER     = 00000002
PSL$K_USER      = 00000003
PSL$M_C         = 00000001
PSL$M_CBIT      = 00000001
PSL$M_CM        = 80000000
PSL$M_CURMOD    = 03000000
PSL$M_DV        = 00000080
PSL$M_FPD       = 08000000
PSL$M_FU        = 00000040
PSL$M_IPL       = 001F0000
PSL$M_IS        = 04000000
PSL$M_IV        = 00000020
PSL$M_N         = 00000008
PSL$M_NBIT      = 00000008
PSL$M_PRVMOD    = 00C00000
PSL$M_SAFBITS   = 000037FF
PSL$M_TBIT      = 00000010
PSL$M_TP        = 40000000
PSL$M_V         = 00000002
PSL$M_VBIT      = 00000002
PSL$M_Z         = 00000004
PSL$M_ZBIT      = 00000004
PSL$S_CURMOD    - 00000002

PSL$S_IPL       = 00000005
PSL$S_PRVMOD    = 00000002
PSL$V_C         = 00000000
PSL$V_CBIT      = 00000000
PSL$V_CM        = 0000001F
PSL$V_CURMOD    = 00000018
PSL$V_DV        = 00000007
PSL$V_FPD       = 0000001B
PSL$V_FU        = 00000006
PSL$V_IPL       = 00000010
PSL$V_IS        = 0000001A
PSL$V_IV        = 00000005
PSL$V_N         = 00000003
PSL$V_NBIT      = 00000003
PSL$V_PRVMOD    = 00000016
PSL$V_TBIT      = 00000004
PSL$V_TP        = 0000001E
PSL$V_V         = 00000001
PSL$V_VBIT      = 00000001
PSL$V_Z         = 00000002
PSL$V_ZBIT      = 00000002
SCB$L_ACCESS      00000020
SCB$L_ARITH       00000034
SCB$L_BREAK       0000002C
SCB$L_CHME        00000044
SCB$L_CHMK        00000040
SCB$L_CHMS        00000048
SCB$L_CHMU        0000004C
SCB$L_COMPAT      00000030
SCB$L_KNLSTK      00000008
SCB$L_MACHCHK     00000004
SCB$L_OPCCUS      00000014
SCB$L_OPCDEC      00000010
SCB$L_POWER       0000000C
SCB$L_RADRMOD     0000001C
SCB$L_ROPRAND     00000018
SCB$L_RXDB        000000F8
SCB$L_SFTLVL1     00000084
SCB$L_SFTLVL10    000000A8
SCB$L_SFTLVL11    000000AC
SCB$L_SFTLVL12    000000B0
SCB$L_SFTLVL13    000000B4
SCB$L_SFTLVL14    000000B8
SCB$L_SFTLVL15    000000BC
SCB$L_SFTLVL2     00000088
SCB$L_SFTLVL3     0000008C
SCB$L_SFTLVL4     00000090
SCB$L_SFTLVL5     00000094
SCB$L_SFTLVL6     00000098
SCB$L_SFTLVL7     0000009C
SCB$L_SFTLVL8     000000A0
SCB$L_SFTLVL9     000000A4
SCB$L_TBIT        00000028
SCB$L_TIMER       000000C0
SCB$L_TRANSL      00000024
SCB$L_TXDB        000000FC
SCB$L_ZERO        00000000

SEP_FUNCTIONAL  = 00000002
SEP_REPAIR      = 00000003
SIZ...          = 00000001
STRTUP          = 00000100
SYS$ALLOC         00010238
SYS$ASCTIM        00010248
SYS$ASSIGN        00010250
SYS$BINTIM        00010258
SYS$CANCEL        00010260
SYS$CANTIM        00010268
SYS$CLOSE         000105B8
SYS$CLREF         00010298
SYS$CONNECT       000105C0
SYS$DALLOC        000102D8
SYS$DASSGN        000102E0
SYS$DISCONNECT    000105D0
SYS$FAO           00010350
SYS$FAOL          00010358
SYS$GET           00010580
SYS$GETCHN        000104C8
SYS$GETTIM        00010378
SYS$LKWSET        000103A0
SYS$NUMTIM        000103B8
SYS$OPEN          00010608
SYS$QIO           000103C8
SYS$QIOW          00010200
SYS$READ          00010590
SYS$READEF        000103D0
SYS$SETAST        000103F8
SYS$SETEF         00010400
SYS$SETIMR        00010420
SYS$SETPRI        00010428
SYS$SETPRT        00010430
SYS$SETRWM        00010438
SYS$SETSWM        00010448
SYS$ULKPAG        00010460
SYS$ULWSET        00010468
SYS$UNWIND        00010470
SYS$WAITFR        00010478
SYS$WFLAND        00010488
SYS$WFLOR         00010490
T2_S1             0000007F RG     04
T2_S1_X           000000F6 R      04
T2_S2             00000101 RG     04
T2_S2_X           0000018B R      04
T2_S3             00000196 RG     04
T2_S3_X           0000025F R      04
TEST_002          00000014 RG     04
TEST_002_X        00000276 RG     04
TU58            = 00000002  G
T_BADMODE         000002BB R      02
T_BADMODE_DN      000002E3 R      02
T_BADMODE_UP      0000030D R      02
T_BADSTACK        00000335 R      02
T_BADSTACK_DN     00000364 R      02
T_BADSTACK_UP     00000395 R      02
T_CHECKSUM_ERR    000003C4 R      02
```

```
T_INST              0000005A R      02
T_MESSAGE_1         00000093 R      02
T_MESSAGE_2         0000013E R      02
T_MESSAGE_3         000001D9 R      02
T_NO_KA730          00000025 R      02
T_PWRDN_ERROR       00000273 R      02
T_PWRUP_ERROR       00000298 R      02
UBI               = 0000000A  G
V_BADMODE_DN      = 00000000
V_BADMODE_UP      = 00000001
V_BADSTACK_DN     = 00000002
V_BADSTACK_UP     = 00000003
V_BAD_PC          = 00000002
V_PWRDN_DONE      = 00000000
V_PWRUP_DONE      = 00000001
V_PWRUP_ERR       = 00000004
WCS               = 00000007  G
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXC	0000064F (1615.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_002	0000027E (638.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000018 (24.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	00000024 (36.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
$TSTCNT	00000004 (4.)	07 (7.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000018-R	512 (9)	390 (7) 451 (8) 512 (9)
$$E	=00000001	512 (9)	410 (7) 475 (8) 542 (9)
$$M	=00000004	539 (9)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5) 283 (5) 340 (6) 407 (7) 472 (8) 536 (9) 539 (9)
$$N	=00000000	539 (9)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5) 283 (5) #-344 (6) #-355 (6) #-399 (7) 407 (7) #-464 (8) 472 (8) #-528 (9) 536 (9) 539 (9) 80 (3)
$$S	=FFFFFFFF	546 (9)	318 (6) 340 (6) 358 (7) 390 (7) 414 (8) 451 (8) 479 (9) 512 (9)
$ENV	=00000001	62 (3)	
$ER	=FFFFFFFF	546 (9)	#-269 (5) #-272 (5) #-274 (5) #-278 (5) #-281 (5) #-283 (5) 390 (7) #-407 (7) 451 (8) #-472 (8) 512 (9) #-536 (9) #-539 (9)
$MO	=00000001	548 (9)	548 (9)
$SS	=00000018-R	512 (9)	390 (7) 410 (7) 451 (8) 475 (8) 512 (9) 542 (9)
$ST	=00000000	546 (9)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5) 283 (5) 340 (6) 358 (7) 390 (7) 407 (7) 410 (7) 414 (8) 451 (8) 472 (8) 475 (8) 479 (9) 512 (9) 536 (9) 539 (9) 542 (9) 546 (9)
$TN	=00000002	548 (9)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5) 283 (5) 318 (6) 340 (6) 345 (6) 358 (7) 407 (7) 414 (8) 472 (8) 479 (9) 536 (9) 539 (9) 546 (9) 548 (9)
ALL	=00000008	80 (3)	340 (6)
A_BUFFER_BEGIN	00000000-XR		309 (5) 459 (8) 460 (8) 522 (9) 523 (9)
A_PWR_BUFFER	00000000-XR		195 (5)
BASE_002	00000000-R	318 (6)	340 (6)
BIT...	=00000005	78 (3)	62 (3) 75 (3) 77 (3) 78 (3)
B_FLAGS	0000001D-R	96 (3)	#-223 (5) #-285 (5) #-392 (7) 403 (7) 406 (7) #-453 (8) 468 (8) 471 (8) #-514 (9) 532 (9) 538 (9)
B_FLAGS2	0000001E-R	97 (3)	#-200 (5) #-202 (5) #-236 (5) #-244 (5) #-246 (5) 267 (5) 268 (5) 271 (5) 276 (5) 277 (5) 280 (5) #-393 (7) #-454 (8) #-515 (9) 535 (9)
B_RESPONSE	0000001C-R	95 (3)	
CALC_CHECKSUM	00000620-R	297 (5)	350 (6) 353 (6)
CEP_FUNCTIONAL	=00000000	62 (3)	
CEP_REPAIR	=00000001	62 (3)	62 (3)
CLEAR_BUFFER	00000637-R	308 (5)	391 (7) 452 (8) 513 (9)
DATA_002	00000010-R	340 (6)	340 (6)
DEFAULT	=00000000	80 (3)	
DS$ABORT	00010020	76 (3)	270 (5) 273 (5) 275 (5) 279 (5) 282 (5) 284 (5) 408 (7) 473 (8) 537 (9) 540 (9)
DS$ASKADR	00010090	76 (3)	
DS$ASKDATA	00010080	76 (3)	
DS$ASKLGCL	00010098	76 (3)	

```
DS$ASKSTR          000100A0      76      (3)
DS$ASKVLD          00010088      76      (3)
DS$ATTACH          000101A8      76      (3)
DS$BGNSUB          00010030      76      (3)       390   (7)       451   (8)       512   (9)
DS$BRANCH          000100A8      76      (3)
DS$BREAK           00010058      76      (3)       224   (5)       402   (7)       467   (8)       531   (9)       546   (9)
DS$CANWAIT         00010070      76      (3)
DS$CHANNEL         00010180      76      (3)
DS$CKLOOP          00010040      76      (3)
DS$CLRVEC          00010168      76      (3)       544   (9)
DS$CNTRLC          00010078      76      (3)
DS$CVTREG          000100B0      76      (3)
DS$ENDPASS         00010010      76      (3)
DS$ENDSUB          00010038      76      (3)       410   (7)       475   (8)       542   (9)
DS$ERRDEV          000100C8      76      (3)
DS$ERRHARD         000100D0      76      (3)       269   (5)       272   (5)       274   (5)       278   (5)       281   (5)
                                                   283   (5)       407   (7)       472   (8)       536   (9)       539   (9)
DS$ERRPREP         00010108      76      (3)
DS$ERRSOFT         000100D8      76      (3)
DS$ERRSYS          000100C0      76      (3)
DS$ESCAPE          00010050      76      (3)
DS$FREEDBGSYM      000101B8      76      (3)
DS$GETBUF          00010120      76      (3)
DS$GETMEM          00010130      76      (3)
DS$GPHARD          00010018      76      (3)
DS$HELP            000101B0      76      (3)
DS$INITSCB         00010170      76      (3)
DS$INLOOP          00010048      76      (3)
DS$K_ERROR        =00000002      75      (3)
DS$K_NORMAL       =00000001      75      (3)
DS$K_SEVERE       =00000004      75      (3)
DS$K_SUBSYS       =00000066      75      (3)        75   (3)
DS$K_WARNING      =00000000      75      (3)
DS$LOAD            00010198      76      (3)
DS$LOADPCS         000101C0      76      (3)
DS$MAPDBGBLOCK     00010118      76      (3)
DS$MMOFF           00010158      76      (3)
DS$MMON            00010150      76      (3)
DS$MOVPHY          00010148      76      (3)
DS$MOVVRT          00010140      76      (3)
DS$PARSE           000100B8      76      (3)
DS$PRINTB          000100E0      76      (3)
DS$PRINTF          000100F0      76      (3)       344   (6)       355   (6)       399   (7)       464   (8)       528   (9)
DS$PRINTS          000100F8      76      (3)
DS$PRINTSIG        00010100      76      (3)
DS$PRINTX          000100E8      76      (3)
DS$PROBE           000101A0      76      (3)
DS$RELBUF          00010128      76      (3)
DS$RELMEM          00010138      76      (3)
DS$SETIPL          00010178      76      (3)
DS$SETMAP          00010188      76      (3)
DS$SETPRIEXV       00010110      76      (3)
DS$SETVEC          00010160      76      (3)       347   (6)
DS$SHOCHAN         00010190      76      (3)
DS$SUMMARY         00010028      76      (3)
DS$WAITMS          00010060      76      (3)
DS$WAITUS          00010068      76      (3)
```

```
DS$_ARITH        =006600D0      75      (3)
DS$_BADLINK      =006600F0      75      (3)
DS$_BADTYPE      =006600E8      75      (3)
DS$_CHME         =006600A8      75      (3)
DS$_CHMK         =006600E0      75      (3)
DS$_DEVNAME      =00660108      75      (3)
DS$_ERROR        =00660002      75      (3)
DS$_FHWE         =00660068      75      (3)
DS$_FRAGBUF      =00660080      75      (3)
DS$_ICBUSY       =006600C8      75      (3)
DS$_ICERR        =006600C0      75      (3)
DS$_IHWE         =00660060      75      (3)
DS$_ILLCHAR      =00660018      75      (3)
DS$_ILLPAGCNT    =00660078      75      (3)
DS$_ILLUNIT      =00660100      75      (3)
DS$_INSFMEM      =00660050      75      (3)
DS$_IPL2HI       =006600B8      75      (3)
DS$_IVADDR       =00660040      75      (3)
DS$_IVVECT       =00660038      75      (3)
DS$_KRNLSTK      =00660090      75      (3)
DS$_LOGIC        =00660070      75      (3)
DS$_MCHK         =00660088      75      (3)
DS$_MMOFF        =00660058      75      (3)
DS$_NEEDUNIT     =006600F8      75      (3)
DS$_NOPCS        =00660110      75      (3)
DS$_NORMAL       =00660001      75      (3)
DS$_NOSUPPORT    =006600B1      75      (3)
DS$_NOTDON       =00660030      75      (3)
DS$_NOTIMP       =006600B0      75      (3)       75      (3)
DS$_NULLSTR      =00660010      75      (3)
DS$_OVERFLOW     =00660008      75      (3)
DS$_POWER        =00660098      75      (3)
DS$_PROGERR      =00660020      75      (3)
DS$_SEVERE       =00660004      75      (3)
DS$_TRANSL       =006600A0      75      (3)
DS$_TRUNCATE     =00660028      75      (3)
DS$_UNEXPINT     =006600D8      75      (3)
DS$_VASFULL      =00660048      75      (3)
DS$_WARNING      =00660000      75      (3)       75      (3)
ENV$M_DOMAIN     =00000002      62      (3)
ENV$M_LEVEL      =00000001      62      (3)
ENV$M_SUPER      =000003FC      62      (3)
ENV$S_DOMAIN     =00000001      62      (3)
ENV$S_LEVEL      =00000001      62      (3)
ENV$S_SUPER      =00000008      62      (3)
ENV$V_DOMAIN     =00000001      62      (3)       62      (3)
ENV$V_LEVEL      =00000000      62      (3)       62      (3)
ENV$V_SUPER      =00000002      62      (3)
ENV$_CPU         =00000000      62      (3)       62      (3)
ENV$_FUNCTIONAL  =00000000      62      (3)       62      (3)
ENV$_REPAIR      =00000001      62      (3)       62      (3)
ENV$_SUPER       =00000001      62      (3)
ENV$_SYSTEM      =00000001      62      (3)       62      (3)
HALT             =00000004      80      (3)
IPR              =00000006      80      (3)
JUMP              C000001F-R    118     (3)     #-527     (9)
KA_PTABLE         00000000-XR                   #-342     (6)
```

```
KA_UNIT_NO      00000000-XR                     #-269   (5)   #-272   (5)   #-274   (5)   #-278   (5)   #-281   (5)
                                                #-283   (5)   #-407   (7)   #-472   (8)   #-536   (9)   #-539   (9)
LDPCTX          =0000000B       80      (3)
L_BUF_BEGIN      00000000-R     87      (3)     #-222   (5)   #-238   (5)
L_CHECKSUM       00000008-R     89      (3)     #-351   (6)   #-397   (7)   #-462   (8)   #-525   (9)
L_CHECKSUM1      0000000C-R     90      (3)     #-354   (6)   #-519   (9)
L_COUNT          00000004-R     88      (3)     #-299   (5)   #-302   (5)   #-310   (5)   #-312   (5)
L_IS_SP          00000014-R     92      (3)     #-203   (5)   #-206   (5)   #-247   (5)   #-250   (5)   #-251   (5)
L_KS_SP          00000018-R     93      (3)     #-204   (5)   #-207   (5)   #-248   (5)   #-252   (5)
L_PSL            00000010-R     91      (3)     #-196   (5)     198   (5)     201   (5)   #-240   (5)     242   (5)
                                                  245   (5)
MANUAL          =00000001       80      (3)       340   (6)
MCHK            =00000005       80      (3)
MEM             =00000009       80      (3)
MSG_002          00000002-R     318     (6)       340   (6)
M_BADMODE_DN    =00000001       107     (3)     #-200   (5)
M_BADMODE_UP    =00000002       109     (3)     #-244   (5)
M_BADSTACK_DN   =00000004       111     (3)     #-202   (5)
M_BADSTACK_UP   =00000008       113     (3)     #-246   (5)
M_BAD_PC        =00000004       104     (3)
M_PWRDN_DONE    =00000001       100     (3)     #-223   (5)
M_PWRUP_DONE    =00000002       102     (3)     #-285   (5)
M_PWRUP_ERR     =00000010       115     (3)     #-236   (5)
PAR$M_ATDEF     =00000010       78      (3)
PAR$M_ATHI      =00000008       78      (3)
PAR$M_ATLO      =00000004       78      (3)
PAR$M_NODEF     =00000002       78      (3)
PAR$V_ATDEF     =00000004       78      (3)
PAR$V_ATHI      =00000003       78      (3)
PAR$V_ATLO      =00000002       78      (3)
PAR$V_NODEF     =00000001       78      (3)
PAR$_BIN        =00000002       78      (3)
PAR$_DEC        =0000000A       78      (3)
PAR$_HEX        =00000010       78      (3)
PAR$_NO         =00000000       78      (3)
PAR$_OCT        =00000008       78      (3)
PAR$_YES        =00000001       78      (3)
POWER           =00000003       80      (3)       340   (6)
POWER_DOWN       000003EC-R     194     (5)       347   (6)
POWER_DOWN_1     000000BF-R     401     (7)     #-404   (7)
POWER_DOWN_2     00000154-R     466     (8)     #-469   (8)
POWER_DOWN_3     00000202-R     530     (9)     #-533   (9)
POWER_UP         000004A0-R     237     (5)       118   (3)     349   (6)     396   (7)     461   (8)     524   (9)
POWER_UP_ERR     0000049B-R     235     (5)       352   (6)     518   (9)
PR$_ASTLVL      00000000-XR                     #-220   (5)   #-253   (5)
PR$_ESP         00000000-XR                     #-210   (5)   #-263   (5)
PR$_ISP         00000000-XR                     #-203   (5)   #-247   (5)   #-250   (5)
PR$_KSP         00000000-XR                     #-207   (5)   #-252   (5)
PR$_P0BR        00000000-XR                     #-216   (5)   #-257   (5)
PR$_P0LR        00000000-XR                     #-217   (5)   #-256   (5)
PR$_P1BR        00000000-XR                     #-218   (5)   #-255   (5)
PR$_P1LR        00000000-XR                     #-219   (5)   #-254   (5)
PR$_PCBB        00000000-XR                     #-215   (5)   #-258   (5)
PR$_SBR         00000000-XR                     #-213   (5)   #-260   (5)
PR$_SCBB        00000000-XR                     #-221   (5)   #-239   (5)
PR$_SLR         00000000-XR                     #-214   (5)   #-259   (5)
PR$_SSP         00000000-XR                     #-211   (5)   #-262   (5)
```

```
PR$_TXDB         00000000-XR                    #-286    (5)    #-348    (6)
PR$_USP          00000000-XR                    #-212    (5)    #-261    (5)
PSL$C_EXEC      =00000001      77      (3)
PSL$C_KERNEL    =00000000      77      (3)
PSL$C_SUPER     =00000002      77      (3)
PSL$C_USER      =00000003      77      (3)
PSL$K_EXEC      =00000001      77      (3)
PSL$K_KERNEL    =00000000      77      (3)    #-198    (5)    #-242    (5)
PSL$K_SUPER     =00000002      77      (3)
PSL$K_USER      =00000003      77      (3)
PSL$M_C         =00000001      77      (3)
PSL$M_CBIT      =00000001      77      (3)
PSL$M_CM        =80000000      77      (3)      77      (3)
PSL$M_CURMOD    =03000000      77      (3)
PSL$M_DV        =00000080      77      (3)
PSL$M_FPD       =08000000      77      (3)      77      (3)
PSL$M_FU        =00000040      77      (3)
PSL$M_IPL       =001F0000      77      (3)
PSL$M_IS        =04000000      77      (3)
PSL$M_IV        =00000020      77      (3)
PSL$M_N         =00000008      77      (3)
PSL$M_NBIT      =00000008      77      (3)
PSL$M_PRVMOD    =00C00000      77      (3)
PSL$M_SAFBITS   =000037FF      77      (3)
PSL$M_TBIT      =00000010      77      (3)
PSL$M_TP        =40000000      77      (3)      77      (3)
PSL$M_V         =00000002      77      (3)
PSL$M_VBIT      =00000002      77      (3)
PSL$M_Z         =00000004      77      (3)
PSL$M_ZBIT      =00000004      77      (3)
PSL$S_CURMOD    =00000002      77      (3)    #-197    (5)    #-241    (5)
PSL$S_IPL       =00000005      77      (3)
PSL$S_PRVMOD    =00000002      77      (3)
PSL$V_C         =00000000      77      (3)
PSL$V_CBIT      =00000000      77      (3)
PSL$V_CM        =0000001F      77      (3)
PSL$V_CURMOD    =00000018      77      (3)    #-197    (5)    #-241    (5)
PSL$V_DV        =00000007      77      (3)
PSL$V_FPD       =0000001B      77      (3)
PSL$V_FU        =00000006      77      (3)
PSL$V_IPL       =00000010      77      (3)
PSL$V_IS        =0000001A      77      (3)    #-201    (5)    #-245    (5)
PSL$V_IV        =00000005      77      (3)
PSL$V_N         =00000003      77      (3)
PSL$V_NBIT      =00000003      77      (3)
PSL$V_PRVMOD    =00000016      77      (3)
PSL$V_TBIT      =00000004      77      (3)
PSL$V_TP        =0000001E      77      (3)
PSL$V_V         =00000001      77      (3)
PSL$V_VBIT      =00000001      77      (3)
PSL$V_Z         =00000002      77      (3)
PSL$V_ZBIT      =00000002      77      (3)
SCB$L_ACCESS     00000020      74      (3)
SCB$L_ARITH      00000034      74      (3)
SCB$L_BREAK      0000002C      74      (3)
SCB$L_CHME       00000044      74      (3)
SCB$L_CHMK       00000040      74      (3)
```

```
SCB$L_CHMS          00000048      74      (3)
SCB$L_CHMU          0000004C      74      (3)
SCB$L_COMPAT        00000030      74      (3)
SCB$L_KNLSTK        00000008      74      (3)
SCB$L_MACHCHK       00000004      74      (3)
SCB$L_OPCCUS        00000014      74      (3)
SCB$L_OPCDEC        00000010      74      (3)
SCB$L_POWER         0000000C      74      (3)     #-347     (6)     #-544     (9)
SCB$L_RADRMOD       0000001C      74      (3)
SCB$L_ROPRAND       00000018      74      (3)
SCB$L_RXDB          000000F8      74      (3)
SCB$L_SFTLVL1       00000084      74      (3)
SCB$L_SFTLVL10      000000A8      74      (3)
SCB$L_SFTLVL11      000000AC      74      (3)
SCB$L_SFTLVL12      000000B0      74      (3)
SCB$L_SFTLVL13      000000B4      74      (3)
SCB$L_SFTLVL14      000000B8      74      (3)
SCB$L_SFTLVL15      000000BC      74      (3)
SCB$L_SFTLVL2       00000088      74      (3)
SCB$L_SFTLVL3       0000008C      74      (3)
SCB$L_SFTLVL4       00000090      74      (3)
SCB$L_SFTLVL5       00000094      74      (3)
SCB$L_SFTLVL6       00000098      74      (3)
SCB$L_SFTLVL7       0000009C      74      (3)
SCB$L_SFTLVL8       000000A0      74      (3)
SCB$L_SFTLVL9       000000A4      74      (3)
SCB$L_TBIT          00000028      74      (3)
SCB$L_TIMER         000000C0      74      (3)
SCB$L_TRANSL        00000024      74      (3)
SCB$L_TXDB          000000FC      74      (3)
SCB$L_ZERO          00000000      74      (3)
SEP_FUNCTIONAL     =00000002      62      (3)
SEP_REPAIR         =00000003      62      (3)
SIZ...             =00000001      78      (3)       62      (3)       77      (3)       78      (3)
STRTUP             =00000100     120      (3)     #-527     (9)
SYS$ALLOC           00010238      76      (3)
SYS$ASCTIM          00010248      76      (3)
SYS$ASSIGN          00010250      76      (3)
SYS$BINTIM          00010258      76      (3)
SYS$CANCEL          00010260      76      (3)
SYS$CANTIM          00010268      76      (3)
SYS$CLOSE           000105B8      76      (3)
SYS$CLREF           00010298      76      (3)
SYS$CONNECT         000105C0      76      (3)
SYS$DALLOC          000102D8      76      (3)
SYS$DASSGN          000102E0      76      (3)
SYS$DISCONNECT      000105D0      76      (3)
SYS$FAO             00010350      76      (3)
SYS$FAOL            00010358      76      (3)
SYS$GET             00010580      76      (3)
SYS$GETCHN          000104C8      76      (3)
SYS$GETTIM          00010378      76      (3)
SYS$LKWSET          000103A0      76      (3)
SYS$NUMTIM          000103B8      76      (3)
SYS$OPEN            00010608      76      (3)
SYS$QIO             000103C8      76      (3)
SYS$QIOW            00010200      76      (3)
```

```
SYS$READ            00010590     76      (3)
SYS$READEF          000103D0     76      (3)
SYS$SETAST          000103F8     76      (3)
SYS$SETEF           00010400     76      (3)
SYS$SETIMR          00010420     76      (3)
SYS$SETPRI          00010428     76      (3)
SYS$SETPRT          00010430     76      (3)
SYS$SETRWM          00010438     76      (3)
SYS$SETSWM          00010448     76      (3)
SYS$ULKPAG          00010460     76      (3)
SYS$ULWSET          00010468     76      (3)
SYS$UNWIND          00010470     76      (3)
SYS$WAITFR          00010478     76      (3)
SYS$WFLAND          00010488     76      (3)
SYS$WFLOR           00010490     76      (3)
T2_S1               0000007F-R   390     (7)
T2_S1_X             000000F6-R   410     (7)
T2_S2               00000101-R   451     (8)
T2_S2_X             0000018B-R   475     (8)
T2_S3               00000196-R   512     (9)
T2_S3_X             0000025F-R   542     (9)
TEST_002            00000014-R   340     (6)      340     (6)
TEST_002_X          00000276-R   546     (9)    #-345     (6)
TU58              =00000002      80      (3)
T_BADMODE           000002BB-R   156     (4)      272     (5)
T_BADMODE_DN        000002E3-R   160     (4)      274     (5)
T_BADMODE_UP        0000030D-R   164     (4)      269     (5)
T_BADSTACK          00000335-R   168     (4)      281     (5)
T_BADSTACK_DN       00000364-R   172     (4)      283     (5)
T_BADSTACK_UP       00000395-R   176     (4)      278     (5)
T_CHECKSUM_ERR      000003C4-R   180     (4)      536     (9)
T_INST              0000005A-R   127     (4)      355     (6)
T_MESSAGE_1         00000093-R   130     (4)      399     (7)
T_MESSAGE_2         0000013E-R   135     (4)      528     (9)
T_MESSAGE_3         000001D9-R   140     (4)      464     (8)
T_NO_KA730          00000025-R   124     (4)      344     (6)
T_PWRDN_ERROR       00000273-R   148     (4)      407     (7)      472     (8)      539     (9)
T_PWRUP_ERROR       00000298-R   152     (4)
UBI               =0000000A      80      (3)
V_BADMODE_DN      =00000000      106     (3)    #-267     (5)
V_BADMODE_UP      =00000001      108     (3)    #-268     (5)    #-271     (5)
V_BADSTACK_DN     =00000002      110     (3)    #-276     (5)
V_BADSTACK_UP     =00000003      112     (3)    #-277     (5)    #-280     (5)
V_BAD_PC          =00000002      103     (3)
V_PWRDN_DONE      =00000000      99      (3)    #-406     (7)    #-471     (8)    #-538     (9)
V_PWRUP_DONE      =00000001      101     (3)    #-403     (7)    #-468     (8)    #-532     (9)
V_PWRUP_ERR       =00000004      114     (3)    #-535     (9)
WCS               -00000007      80      (3)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_ABPROGRAM	1	270 (5)	270 (5) 273 (5) 275 (5) 279 (5) 282 (5)
			284 (5) 408 (7) 473 (8) 537 (9) 540 (9)
$D1_BSUBT	1	390 (7)	390 (7) 451 (8) 512 (9)
$D1_BTEST	1	340 (6)	340 (6)
$D1_ERR	1	269 (5)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5)
			283 (5) 407 (7) 472 (8) 536 (9) 539 (9)
$D1_ERR_S	1	269 (5)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5)
			283 (5) 407 (7) 472 (8) 536 (9) 539 (9)
$D1_ESUBT	1	410 (7)	410 (7) 475 (8) 542 (9)
$D1_ETEST	1	546 (9)	546 (9)
$D1_EXTEST	1	345 (6)	345 (6)
$D1_PRINT_S	1	344 (6)	344 (6) 355 (6) 399 (7) 464 (8) 528 (9)
$D1_SBTTL	2	318 (6)	318 (6) 358 (7) 414 (8) 479 (9)
$D2_BDATA	1	340 (6)	340 (6)
$D2_BTEST	1	340 (6)	340 (6)
$D2_SBTTL	1	318 (6)	318 (6)
$DEF	1	78 (3)	74 (3) 76 (3)
$DEFEND	1	74 (3)	74 (3) 76 (3) 77 (3)
$DEFINI	1	74 (3)	74 (3) 76 (3) 77 (3)
$DS_ABORT	1	270 (5)	270 (5) 273 (5) 275 (5) 279 (5) 282 (5)
			284 (5) 408 (7) 473 (8) 537 (9) 540 (9)
$DS_BGNMOD	1	62 (3)	62 (3)
$DS_BGNSUB	1	390 (7)	390 (7) 451 (8) 512 (9)
$DS_BGNTEST	1	340 (6)	340 (6)
$DS_BREAK	1	224 (5)	224 (5) 402 (7) 467 (8) 531 (9) 546 (9)
$DS_CLRVEC_S	1	544 (9)	544 (9)
$DS_DSDEF	2	75 (3)	75 (3)
$DS_DSSDEF	1	76 (3)	76 (3)
$DS_ENDDATA	1	340 (6)	340 (6)
$DS_ENDMOD	1	548 (9)	548 (9)
$DS_ENDSUB	1	410 (7)	410 (7) 475 (8) 542 (9)
$DS_ENDTEST	1	546 (9)	546 (9)
$DS_ENVDEF	2	62 (3)	62 (3)
$DS_ERRHARD_S	1	269 (5)	269 (5) 272 (5) 274 (5) 278 (5) 281 (5)
			283 (5) 407 (7) 472 (8) 536 (9) 539 (9)
$DS_EXIT	1	345 (6)	345 (6)
$DS_PAGE	1	316 (5)	316 (5) 356 (6) 411 (7) 477 (8)
$DS_PARDEF	1	78 (3)	78 (3)
$DS_PRINTF_S	1	344 (6)	344 (6) 355 (6) 399 (7) 464 (8) 528 (9)
$DS_PSLDEF	1	77 (3)	77 (3)
$DS_SBTTL	1	318 (6)	318 (6) 358 (7) 414 (8) 479 (9)
$DS_SCBDEF	1	74 (3)	74 (3)
$DS_SECDEF	1	80 (3)	80 (3)
$DS_SETVEC_S	1	347 (6)	347 (6)
$EQU	1	78 (3)	62 (3) 75 (3) 77 (3) 78 (3)
$EQULS1	1	77 (3)	62 (3) 75 (3) 77 (3)
$EQULST	1	62 (3)	62 (3) 75 (3) 77 (3)
$GBLINI	2	62 (3)	62 (3) 74 (3) 75 (3) 76 (3) 77 (3)
			78 (3)
$PSLDEF	1	77 (3)	77 (3)

$PUSHADR	1	269	(5)	269	(5)	272	(5)	274	(5)	278	(5)	281	(5)
				283	(5)	407	(7)	472	(8)	536	(9)	539	(9)
$VIELD	1	62	(3)	62	(3)	77	(3)	78	(3)				
$VIELD1	1	78	(3)	62	(3)	77	(3)	78	(3)				
ENKAX_SECDEF	1	80	(3)	80	(3)								

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	26	00:00:00.04	00:00:00.26
Command processing	26	00:00:00.22	00:00:00.51
Pass 1	1072	00:00:12.53	00:00:27.16
Symbol table sort	2	00:00:00.51	00:00:01.22
Pass 2	386	00:00:02.90	00:00:08.40
Symbol table output	33	00:00:00.22	00:00:00.68
Psect synopsis output	5	00:00:00.04	00:00:00.09
Cross-reference output	177	00:00:01.83	00:00:07.35
Assembler run totals	1731	00:00:18.31	00:00:45.69

The working set limit was 900 pages.
137402 bytes (269 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 359 non-local and 24 local symbols.
550 source lines were read in Pass 1, producing 28 object records in Pass 2.
94 pages of virtual memory were used to define 41 macros.

! Macro library statistics !

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	36
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	7
TOTALS (all libraries)	44

705 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXC

Table of contents

```
0000     1
0000     2
0000     3          .SBTTL  ENKAXD: PROCESSOR HALT
0000     4          .TITLE  ENKAXD  VAX-11/730 Specific CPU Cluster Exerciser
0000     5          .IDENT  /1-3/
0000     6
0000     7 ;
0000     8 ; Copyright (C) 1981
0000     9 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000    10 ;
0000    11 ; This software is furnished  under a license for use only on a single
0000    12 ; computer  system and  may be  copied only with  the inclusion of the
0000    13 ; above copyright notice.  This software, or any other copies thereof,
0000    14 ; may not be provided or  otherwise made available to any other person
0000    15 ; except for use on such system and to one who agrees to these license
0000    16 ; terms.  Title to and  ownership of the  software  shall at all times
0000    17 ; remain in DEC.
0000    18 ;
0000    19 ; The information in this software is subject to change without notice
0000    20 ; and should  not be construed  as a commitment  by Digital  Equipment
0000    21 ; Corporation.
0000    22 ;
0000    23 ; Dec assumes  no  responsibility  for  the use or  reliability of its
0000    24 ; software on equipment which is not supplied by DEC.
0000    25 ;
0000    26
0000    27 ;++
0000    28 ; Facility:      VAX Architecture Diagnostic.
0000    29 ;
0000    30 ; Abstract:      This module consists of two tests that check the various ways
0000    31 ;                the VAX-11/730 processor can be halted in an ARCHITECTURALLY
0000    32 ;                DEFINED and an ARCHITECTURALLY UNDEFINED manner. These ways
0000    33 ;                include executing a HALT instruction in kernel mode, attempting
0000    34 ;                certain references to an invalid interrupt stack, executing
0000    35 ;                a change mode instruction while on the interrupt stack,
0000    36 ;                executing a change mode instruction which will be serviced on
0000    37 ;                the interrupt stack, SCBB read error, invalid vector, and
0000    38 ;                double machine check halts.
0000    39 ;
0000    40 ;
0000    41 ; Environment:   VAX Diagnostic Supervisor.
0000    42 ;
0000    43 ; Author:        Rich Lewis      7-Jly-81        Version 1.0
0000    44 ;
0000    45 ;                Tai-Chun Pan    01-Apr-83       Version 1.2
0000    46 ;
0000    47 ;                Added quick flag on Test 7(TU58). If quick flag is set
0000    48 ;                will skip subtest 4 through subtest 11.  Added event flag 3.
0000    49 ;                If event flag 3 is set, will override protection,
0000    50 ;                i.e., go ahead and write on tape.  If run APT & event flag 3
0000    51 ;                is clear & tape is not scratch, will report an error.
0000    52 ;                Modified Unibus sizer on Test 8 & 9. make test run faster.
0000    53 ;
0000    54 ;
0000    55 ;
0000    56 ;                Tai-Chun Pan    03-Aug-83       Version 1.3
0000    57 ;
```

```
                     0000     58 ;                fixed bugs on TEST 7 and error summary routine call when
                     0000     59 ;                running under APT.
                     0000     60 ;
                     0000     61 ;--
```

```
                     0000     63          .SBTTL  DECLARATIONS    .LIST   MEB
                     0000     64          .NLIST  CND
                 00000000     65          .PSECT ENKAXD,  LONG
                     0000     66          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     67          $DS_BGNMOD      CEP_REPAIR, IN=3
                     0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                     0000     68
                     0000     69 ;
                     0000     70 ; Include files:
                     0000     71 ;
                     0000     72
                     0000     73          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                     0000     74
                     0000     75 ;
                     0000     76 ; Equated symbols:
                     0000     77 ;
                     0000     81
                     0000     82          $DS_SCBDEF
                     0000     83          $DS_DSDEF
                     0000     84          $DS_ERRDEF
                     0000     85          $DS_DSSDEF
                     0000     86          $DS_PSLDEF
                     0000     87          $DS_DSADEF
                     0000     88          $DS_PARDEF
                     0000     89
                     0000     90          ENKAX_SECDEF
                     0000     91
                     0000     92
                     0000     93
                     0000     94
 00000004            0000     95 B_ANSWER:       .BLKL                   ; used to store answer (for asklgcl)
 00000008            0004     96 RENTRY_POINT:   .BLKL                   ; used to store rentry address
 0000000C            0008     97 L_PAGESTRT:     .BLKL                   ; first of 2 longword array to store
 00000010            000C     98 L_PAGEND:       .BLKL                   ; page boundary addresses to setprot
 00000014            0010     99 L_PAGEZERO:     .BLKL                   ; first of 2 longword array to store
 00000018            0014    100 L_PAGE0END:     .BLKL                   ; page 0 boundary addresses to setprot
                     0018    101
 0000001C            0018    102 L_COUNT:        .BLKL                   ; used as a counter
 00000020            001C    103 L_ADDRESS:      .BLKL                   ; used to temporary store addresses
 00000024            0020    104 L_CUR_ISP:      .BLKL                   ; used to store initial ISP
 00000028            0024    105 L_EXP_MOD:      .BLKL                   ; temp storage of expected mode
 0000002C            0028    106 L_EXP_STACK:    .BLKL                   ; temp storage of expected stack
 00000030            002C    107 L_OLDPROT:      .BLKL                   ; byte to store old page protection
 00000034            0030    108                 .BLKL
 00000038            0034    109 L_TEMP:         .BLKL                   ; temporary storage
 0000003C            0038    110 L_CUR_SCBB:     .BLKL                   ; location to store SCBB
                     003C    111
 00000040            003C    112 L_FLAGS:        .BLKL                   ; contains the following flags
                     0040    113                                         ; bit positions and masks
 00000000            0040    114 V_READY         =0                      ; ready to execute flag
 00000001            0040    115 M_READY         =1@0                    ;
 00000001            0040    116 V_DONE          =1                      ; restart done flag
 00000002            0040    117 M_DONE          =1@1                    ;
 00000002            0040    118 V_EXECUTE_ERR   =2                      ; execution error flag
 00000004            0040    119 M_EXECUTE_ERR   =1@2                    ;
 00000003            0040    120 V_CHMX_ERR      =3                      ; change mode error
 00000008            0040    121 M_CHMX_ERR      =1@3                    ;
```

```
                00000004  0040   122 V_XFC_ERR       =4                      ; XFC error
                00000010  0040   123 M_XFC_ERR       =1@4                    ;
                00000005  0040   124 V_TIMER         =5                      ; timer interrupt flag
                00000020  0040   125 M_TIMER         =1@5                    ;
                00000006  0040   126 V_BAD_MODE      =6                      ; bad mode, abort test
                00000040  0040   127 M_BAD_MODE      =1@6                    ;
                00000007  0040   128 V_BAD_STK       =7                      ; stack bad, abort test
                00000080  0040   129 M_BAD_STK       =1@7                    ;
                00000008  0040   130 V_FATAL_ERR     =8                      ; flag fatal error
                00000100  0040   131 M_FATAL_ERR     =1@8                    ;
                00000009  0040   132 V_PSL_ERR       =9                      ; flag PSL error
                00000200  0040   133 M_PSL_ERR       =1@9                    ;
                0000000A  0040   134 V_GPR_ERR       =10                     ; flag GPR error
                00000400  0040   135 M_GPR_ERR       =1@10                   ;
                0000000B  0040   136 V_XFC           =11                     ; expected XFC
                00000800  0040   137 M_XFC           =1@11                   ;
                0000000C  0040   138 V_ERROR_FLAG    =12                     ; flag any error
                00001000  0040   139 M_ERROR_FLAG    =1@12                   ;
                0000000D  0040   140 V_KSP_ERR       =13                     ; flag KSP error
                00002000  0040   141 M_KSP_ERR       =1@13                   ;
                0000000E  0040   142 V_ISP_ERR       =14                     ; flag ISP error
                00004000  0040   143 M_ISP_ERR       =1@14                   ;
                0000000F  0040   144 V_SP_ERR        =15                     ; flag SP error
                00008000  0040   145 M_SP_ERR        =1@15                   ;
                00000010  0040   146 V_ISTK          =16                     ; flag interrupt stack
                00010000  0040   147 M_ISTK          =1@16                   ;
                00000011  0040   148 V_MCHK_FLAG     =17                     ; machine check flag
                00500200  0040   149 L_NXM_ADR       =^X500200               ; address of non-existent memory
                00000019  0040   150 V_USR_WCS       =25                     ; user WCS loaded flag field
                          0040   151
                          0040   152
                00000000  0040   153 CODE_00         =^X0                    ; halt code equates
                00000001  0040   154 CODE_01         =^X1
                00000002  0040   155 CODE_02         =^X2
                00000003  0040   156 CODE_03         =^X3
                00000004  0040   157 CODE_04         =^X4
                00000005  0040   158 CODE_05         =^X5
                00000006  0040   159 CODE_06         =^X6
                00000007  0040   160 CODE_07         =^X7
                00000008  0040   161 CODE_08         =^X8
                00000009  0040   162 CODE_09         =^X9
                0000000A  0040   163 CODE_0A         =^XA
                0000000B  0040   164 CODE_0B         =^XB
                0000000C  0040   165 CODE_0C         =^XC
                          0040   166
                00000080  0040   167 BUFSIZ          =128
                          0040   168
                00000100  0040   169 RESTART        =^X100                  ; label for memory loc 100
0000003C'9F   02   C8     0040   170 JUMP:   BISL2   #M_DONE,@#L_FLAGS      ; code to set done and go to routine
    00000C06'9F   17      0047   171       JMP     @#READ_GPR              ; to read all GPR's after a halt
                          004D   172
```

```
                                                004D    177 .SBTTL INSTRUCION MESSAGES
                                                004D    178 ;-----------------------------------------------------------------------
                                                004D    179 T_NO_KA730:
69 76 65 64 20 6F 4E 20 20 2F 21 00'            004D    180         .ASCIC  "!/  No devices of type KA730 selected.  Exiting test.!/"
20 65 70 79 74 20 66 6F 20 73 65 63             0059
74 63 65 6C 65 73 20 30 33 37 41 4B             0065
67 6E 69 74 69 78 45 20 20 2E 64 65             0071
               2F 21 2E 74 73 65 74 20          007D
                                     37         004D
                                                0085    181 ;-----------------------------------------------------------------------
                                                0085    182 T_INSTRUCTIONS:
65 70 78 65 20 54 4C 41 48 2F 21 00'            0085    183         .ASCIC  "!/HALT expected with the following printout:!/"-
68 74 20 68 74 69 77 20 64 65 74 63             0091
20 67 6E 69 77 6F 6C 6C 6F 66 20 65             009D
3F 2F 21 3A 74 75 6F 74 6E 69 72 70             00A9
20 4C 58 21 3D 43 50 20 20 42 58 21             00B5
                        2F 21                   00C1
                              3D                0085
                                                00C3    184                 "?!XB  PC=!XL !/"
                                                00C3    185
                                                00C3    186 ;-----------------------------------------------------------------------
                                                00C3    187 T_CONTINUE:
65 75 6E 69 74 6E 6F 63 20 6F 54 00'            00C3    188         .ASCIC  "To continue from console mode do the following:!/"-
6C 6F 73 6E 6F 63 20 6D 6F 72 66 20             00CF
68 74 20 6F 64 20 65 64 6F 6D 20 65             00DB
3A 67 6E 69 77 6F 6C 6C 6F 66 20 65             00E7
                                  2F 21         00F3
43 3C 20 30 30 31 20 46 20 47 2F 44     00F5    189                 'D/G F 100 <CR> and then C <CR> "
20 6E 65 68 74 20 64 6E 61 20 3E 52     0101
                  20 3E 52 43 3C 20 43  010D
                                     50 00C3
                                        0114    190 ;-----------------------------------------------------------------------
                                        0114    191 T_EXITING:
4C 41 48 20 67 6E 69 74 69 78 45 00'    0114    192         .ASCIC  "Exiting HALT test.!/"
               2F 21 2E 74 73 65 74 20  0120
                                     54 0114
                                        0129    193 ;-----------------------------------------------------------------------
                                        0129    194 T_WARNING:
20 3A 47 4E 49 4E 52 41 57 2F 21 00'    0129    195         .ASCIC  "!/WARNING: The AUTO RESTART switch must be in the OFF!/"-
53 45 52 20 4F 54 55 41 20 65 68 54     0135
20 68 63 74 69 77 73 20 54 52 41 54     0141
74 20 6E 69 20 65 62 20 74 73 75 6D     014D
               2F 21 46 46 4F 20 65 68  0159
20 6E 6F 69 74 69 73 6F 70 20 20 09     0161    196                 "       position for this test to operate properly.!/"
73 65 74 20 73 69 68 74 20 72 6F 66     016D
65 74 61 72 65 70 6F 20 6F 74 20 74     0179
2F 21 2E 79 6C 72 65 70 6F 72 70 20     0185
                                  67    0129
                                        0191    197 ;-----------------------------------------------------------------------
                                        0191    198 T_REST_OFF:
4F 54 55 41 20 65 68 74 20 73 49 00'    0191    199         .ASCIC  "Is the AUTO RESTART switch in the off position?"
69 77 73 20 54 52 41 54 53 45 52 20     019D
6F 20 65 68 74 20 6E 69 20 68 63 74     01A9
3F 6E 6F 69 74 69 73 6F 70 20 66 66     01B5
                                  2F    0191
                                        01C1    200 ;-----------------------------------------------------------------------
```

```
                                                01C1    202 .SBTTL  ERROR MESSAGES
                                                01C1    203 ;-----------------------------------------------------------------------
                                                01C1    204 T_HALT_MESS:
20 64 65 74 70 6D 65 74 74 41 20 00'            01C1    205         .ASCIC  " Attempted to halt the processor by executing!/"
20 65 68 74 20 74 6C 61 68 20 6F 74             01CD
79 62 20 72 6F 73 73 65 63 6F 72 70             01D9
2F 21 67 6E 69 74 75 63 65 78 65 20             01E5
                                        2F      01C1
                                                01F1    206 ;-----------------------------------------------------------------------
                                                01F1    207 T_HALT_ERR:
6F 63 20 6F 72 63 61 6D 20 61 20 00'            01F1    208         .ASCIC  " a macro code HALT instruction.!/!/"
74 73 6E 69 20 54 4C 41 48 20 65 64             01FD
2F 21 2F 21 2E 6E 6F 69 74 63 75 72             0209
                                        23      01F1
                                                0215    209 ;-----------------------------------------------------------------------
                                                0215    210 T_INVLDIS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'            0215    211         .ASCIC  " an XFC inst with the IS pointing to a protected page.!/!/"
49 20 65 68 74 20 68 74 69 77 20 74             0221
74 20 67 6E 69 74 6E 69 6F 70 20 53             022D
65 74 63 65 74 6F 72 70 20 61 20 6F             0239
   2F 21 2F 21 2E 65 67 61 70 20 64             0245
                                        3A      0215
                                                0250    212 ;-----------------------------------------------------------------------
                                                0250    213 T_NXMIS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'            0250    214         .ASCIC  " an XFC inst with the IS pointing to nonexistent memory.!/!/"
49 20 65 68 74 20 68 74 69 77 20 74             025C
74 20 67 6E 69 74 6E 69 6F 70 20 53             0268
6E 65 74 73 69 78 65 6E 6F 6E 20 6F             0274
21 2F 21 2E 79 72 6F 6D 65 6D 20 74             0280
                                        2F      028C
                                        3C      0250
                                                028D    215 ;-----------------------------------------------------------------------
                                                028D    216 T_CHMU_ERR:
73 6E 69 20 55 4D 48 43 20 61 20 00'            028D    217         .ASCIC  " a CHMU instruction with PSL<IS>=1.!/!/"
74 69 77 20 6E 6F 69 74 63 75 72 74             0299
2E 31 3D 3E 53 49 3C 4C 53 50 20 68             02A5
                        2F 21 2F 21             02B1
                                        27      028D
                                                02B5    218 ;-----------------------------------------------------------------------
                                                02B5    219 T_CHMS_ERR:
73 6E 69 20 53 4D 48 43 20 61 20 00'            02B5    220         .ASCIC  " a CHMS instruction with PSL<IS>=1.!/!/"
74 69 77 20 6E 6F 69 74 63 75 72 74             02C1
2E 31 3D 3E 53 49 3C 4C 53 50 20 68             02CD
                        2F 21 2F 21             02D9
                                        27      02B5
                                                02DD    221 ;-----------------------------------------------------------------------
                                                02DD    222 T_CHME_ERR:
73 6E 69 20 45 4D 48 43 20 61 20 00'            02DD    223         .ASCIC  " a CHME instruction with PSL<IS>=1.!/!/"
74 69 77 20 6E 6F 69 74 63 75 72 74             02E9
2E 31 3D 3E 53 49 3C 4C 53 50 20 68             02F5
                        2F 21 2F 21             0301
                                        27      02DD
                                                0305    224 ;-----------------------------------------------------------------------
                                                0305    225 T_CHMK_ERR:
73 6E 69 20 4B 4D 48 43 20 61 20 00'            0305    226         .ASCIC  " a CHMK instruction with PSL<IS>=1.!/!/"
74 69 77 20 6E 6F 69 74 63 75 72 74             0311
2E 31 3D 3E 53 49 3C 4C 53 50 20 68             031D
```

```
                                2F 21 2F 21  0329
                                        27   0305
                                             032D    227 ;-----------------------------------------------------------------------
                                             032D    228 T_SCB_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'         032D    229         .ASCIC  " an XFC inst with the SCB vector entry <1:0> = 3.!/!/"
53 20 65 68 74 20 68 74 69 77 20 74          0339
6E 35 20 72 6F 74 63 65 76 20 42 43          0345
20 3D 20 3E 30 3A 31 3C 20 79 72 74          0351
                  2F 21 2F 21 2E 33          035D
                                        35   032D
                                             0363    230 ;-----------------------------------------------------------------------
                                             0363    231 T_SCBBNXM_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'         0363    232         .ASCIC  " an XFC inst with SCBB pointing to non-existent memory.!/!/"
20 42 42 43 53 20 68 74 69 77 20 74          036F
20 6F 74 20 67 6E 69 74 6E 69 6F 70          037B
74 6E 65 74 73 69 78 65 2D 6E 6F 6E          0387
2F 21 2F 21 2E 79 72 6F 6D 65 6D 20          0393
                                        3B   0363
                                             039F    233 ;-----------------------------------------------------------------------
                                             039F    234 T_WCS_ERR:
73 6E 69 20 43 46 58 20 6E 61 20 00'         039F    235         .ASCIC  " an XFC inst with SCB vector entry <1:0>=2!/"-
76 20 42 43 53 20 68 74 69 77 20 74          03AB
20 79 72 74 6E 65 20 72 6F 74 63 65          03B7
            2F 21 32 3D 3E 30 3A 31 3C       03C3
73 69 64 20 53 43 57 20 64 6E 61 20          03CC    236                 " and WCS disabled or not present!/!/."
74 6F 6E 20 72 6F 20 64 65 6C 62 61          03D8
2F 21 2F 21 74 6E 65 73 65 72 70 20          03E4
                                        2E   03F0
                                        51   039F
                                             03F1    237 ;-----------------------------------------------------------------------
                                             03F1    238 T_CHMUSCB_ERR:
73 6E 69 20 55 4D 48 43 20 61 20 00'         03F1    239         .ASCIC  " a CHMU inst with the SCB vector entry <1:0> = 1 .!/!/"
53 20 65 68 74 20 68 74 69 77 20 74          03FD
6E 65 20 72 6F 74 63 65 76 20 42 43          0409
20 3D 20 3E 30 3A 31 3C 20 79 72 74          0415
                  2F 21 2F 21 2E 20 31       0421
                                        36   03F1
                                             0428    240 ;-----------------------------------------------------------------------
                                             0428    241 T_CHMSSCB_ERR:
73 6E 69 20 53 4D 48 43 20 61 20 00'         0428    242         .ASCIC  " a CHMS inst with the SCB vector entry <1:0> = 1 .!/!/"
53 20 65 68 74 20 68 74 69 77 20 74          0434
6E 65 20 72 6F 74 63 65 76 20 42 43          0440
20 3D 20 3E 30 3A 31 3C 20 79 72 74          044C
                  2F 21 2F 21 2E 20 31       0458
                                        36   0428
                                             045F    243 ;-----------------------------------------------------------------------
                                             045F    244 T_CHMESCB_ERR:
73 6E 69 20 45 4D 48 43 20 61 20 00'         045F    245         .ASCIC  " a CHME inst with the SCB vector entry <1:0> = 1 .!/!/"
53 20 65 68 74 20 68 74 69 77 20 74          046B
6E 65 20 72 6F 74 63 65 76 20 42 43          0477
20 3D 20 3E 30 3A 31 3C 20 79 72 74          0483
                  2F 21 2F 21 2E 20 31       048F
                                        36   045F
                                             0496    246 ;-----------------------------------------------------------------------
                                             0496    247 T_CHMKSCB_ERR:
73 6E 69 20 4B 4D 48 43 20 61 20 00'         0496    248         .ASCIC  " a CHMK inst with the SCB vector entry <1:0> = 1 .!/!/"
53 20 65 68 74 20 68 74 69 77 20 74          04A2
```

```
6E 65 20 72 6F 74 63 65 76 20 42 43  04AE
20 3D 20 3E 30 3A 31 3C 20 79 72 74  04BA
               2F 21 2F 21 2E 20 31  04C6
                                 36  0496
                                     04CD    249 ;-----------------------------------------------------------------------
                                     04CD    250 T_DBL_MCHK_ERR:
63 6E 65 72 65 66 65 72 20 61 20 00' 04CD    251         .ASCIC  " a reference to NXM with vector pointing to NXM location.!/!/"
74 69 77 20 4D 58 4E 20 6F 74 20 65  04D9
69 6F 70 20 72 6F 74 63 65 76 20 68  04E5
4D 58 4E 20 6F 74 20 67 6E 69 74 6E  04F1
2F 21 2E 6E 6F 69 74 61 63 6F 6C 20  04FD
                           2F 21      0509
                                 3D  04CD
                                     050B    252 ;-----------------------------------------------------------------------
                                     050B    253 T_NXM_WCS_ERR:
72 65 66 73 6E 61 72 74 20 61 20 00' 050B    254         .ASCIC  " a transfer to user WCS which has not been loaded.!/!/"
53 43 57 20 72 65 73 75 20 6F 74 20  0517
6E 20 73 61 68 20 68 63 69 68 77 20  0523
64 61 6F 6C 20 6E 65 65 62 20 74 6F  052F
               2F 21 2F 21 2E 64 65  053B
                                 36  050B
                                     0542    255 ;-----------------------------------------------------------------------
                                     0542    256 T_NOT_READY:
72 74 73 6E 69 20 74 73 65 54 20 00' 0542    257         .ASCIC  " Test instruction was not ready to be executed.!/"
6E 20 73 61 77 20 6E 6F 69 74 63 75  054E
20 6F 74 20 79 64 61 65 72 20 74 6F  055A
2E 64 65 74 75 63 65 78 65 20 65 62  0566
                           2F 21      0572
                                 31  0542
                                     0574    258 ;-----------------------------------------------------------------------
                                     0574    259 T_NOT_DONE:
6C 66 20 74 72 61 74 73 65 52 20 00' 0574    260         .ASCIC  " Restart flag was not set while restarting.!/"
73 20 74 6F 6E 20 73 61 77 20 67 61  0580
73 65 72 20 65 6C 69 68 77 20 74 65  058C
      2F 21 2E 67 6E 69 74 72 61 74  0598
                                 2D  0574
                                     05A2    261 ;-----------------------------------------------------------------------
                                     05A2    262 T_EXECUTE:
72 74 73 6E 69 20 74 73 65 54 20 00' 05A2    263         .ASCIC  " Test instruction execution error.!/"
75 63 65 78 65 20 6E 6F 69 74 63 75  05AE
21 2E 72 6F 72 72 65 20 6E 6F 69 74  05BA
                                 2F  05C6
                                 24  05A2
                                     05C7    264 ;-----------------------------------------------------------------------
                                     05C7    265 T_XFC:
75 72 74 73 6E 69 20 43 46 58 20 00' 05C7    266         .ASCIC  " XFC instruction executed instead of causing halt.!/"
74 75 63 65 78 65 20 6E 6F 69 74 63  05D3
6F 20 64 61 65 74 73 6E 69 20 64 65  05DF
61 68 20 67 6E 69 73 75 61 63 20 66  05EB
                     2F 21 2E 74 6C  05F7
                                 34  05C7
                                     05FC    267 ;-----------------------------------------------------------------------
                                     05FC    268 T_CHMX_ERR:
72 74 73 6E 69 20 78 4D 48 43 20 00' 05FC    269         .ASCIC  " CHMx instruction executed instead of causing halt.!/"
75 63 65 78 65 20 6E 6F 69 74 63 75  0608
20 64 61 65 74 73 6E 69 20 64 65 74  0614
68 20 67 6E 69 73 75 61 63 20 66 6F  0620
```

```
                    2F 21 2E 74 6C 61  062C
                                 35    05FC
                                       0632    270 ;-----------------------------------------------------------------------
                                       0632    271 T_TIMER:
64 65 74 63 65 70 78 65 6E 55 20 00'   0632    272         .ASCIC  " Unexpected interval timer interrupt.!/"
69 74 20 6C 61 76 72 65 74 6E 69 20    063E
70 75 72 72 65 74 6E 69 20 72 65 6D    064A
                     2F 21 2E 74       0656
                                 27    0632
                                       065A    273 ;-----------------------------------------------------------------------
                                       065A    274 T_KSP_ERR1:
63 6E 69 20 73 69 20 50 53 4B 20 00'   065A    275         .ASCIC  " KSP is incorrect  -   EXP = !XL  ACT = !XL!/"
20 20 20 2D 20 20 74 63 65 72 72 6F    0666
41 20 20 4C 58 21 20 3D 20 50 58 45    0672
      2F 21 4C 58 21 20 3D 20 54 43    067E
                                 2D    065A
                                       0688    276 ;-----------------------------------------------------------------------
                                       0688    277 T_KSP_ERR2:
20 20 20 2D 20 20 20 50 53 4B 20 00'   0688    278         .ASCIC  " KSP   -   EXP =(UNPREDICTABLE)   ACT = !XL!/"
44 45 52 50 4E 55 28 3D 20 50 58 45    0694
41 20 20 20 29 45 4C 42 41 54 43 49    06A0
      2F 21 4C 58 21 20 3D 20 54 43    06AC
                                 2D    0688
                                       06B6    279 ;-----------------------------------------------------------------------
                                       06B6    280 T_ISP_ERR1:
63 6E 69 20 73 69 20 50 53 49 20 00'   06B6    281         .ASCIC  " ISP is incorrect  -   EXP = !XL  ACT = !XL!/"
20 20 20 2D 20 20 74 63 65 72 72 6F    06C2
41 20 20 4C 58 21 20 3D 20 50 58 45    06CE
      2F 21 4C 58 21 20 3D 20 54 43    06DA
                                 2D    06B6
                                       06E4    282 ;-----------------------------------------------------------------------
                                       06E4    283 T_ISP_ERR2:
20 20 20 2D 20 20 20 50 53 49 20 00'   06E4    284         .ASCIC  " ISP   -   EXP =(UNPREDICTABLE)   ACT = !XL!/"
44 45 52 50 4E 55 28 3D 20 50 58 45    06F0
41 20 20 20 29 45 4C 42 41 54 43 49    06FC
      2F 21 4C 58 21 20 3D 20 54 43    0708
                                 2D    06E4
                                       0712    285 ;-----------------------------------------------------------------------
                                       0712    286 T_SP_ERR1:
6F 63 6E 69 20 73 69 20 50 53 20 00'   0712    287         .ASCIC  " SP is incorrect   -   EXP = !XL  ACT = !XL!/"
20 20 20 2D 20 20 20 74 63 65 72 72    071E
41 20 20 4C 58 21 20 3D 20 50 58 45    072A
      2F 21 4C 58 21 20 3D 20 54 43    0736
                                 2D    0712
                                       0740    288 ;-----------------------------------------------------------------------
                                       0740    289 T_SP_ERR2:
2F 21 4C 58 21 20 3D 20 50 53 20 00'   0740    290         .ASCIC  " SP = !XL!/"
                                 0B    0740
                                       074C    291 ;-----------------------------------------------------------------------
                                       074C    292 T_GPR_ERR:
69 20 61 74 61 64 20 64 61 42 20 00'   074C    293         .ASCIC  " Bad data in general purpose register:!/"
75 70 20 6C 61 72 65 6E 65 67 20 6E    0758
74 73 69 67 65 72 20 65 73 6F 70 72    0764
                     2F 21 3A 72 65    0770
                                 28    074C
                                       0775    294 ;-----------------------------------------------------------------------
                                       0775    295 T_BAD_DATA:
```

```
20 4C 58 21 20 3D 20 50 58 45 20 00' 0775   296          .ASCIC  " EXP = !XL    ACT = !XL    XOR= !XL !/"
4C 58 21 20 3D 20 54 43 41 20 20 20  0781
4C 58 21 20 3D 52 4F 58 20 20 20 20  078D
                  2F 21 20           0799
                        26           0775
                                     079C   297 ;---------------------------------------------------------------------------
                                     079C   298 T_GOOD_DATA:
20 4C 58 21 20 3D 20 50 58 45 20 00' 079C   299          .ASCIC  " EXP = !XL    ACT = !XL !/"
4C 58 21 20 3D 20 54 43 41 20 20 20  07A8
                  2F 21 20           07B4
                        1A           079C
                                     07B7   300 ;---------------------------------------------------------------------------
                                     07B7   301 T_PAGE0PROT:
20 6F 74 20 65 6C 62 61 6E 55 20 00' 07B7   302          .ASCIC  " Unable to set proper protection on page 0.!/"
70 20 72 65 70 6F 72 70 20 74 65 73  07C3
6E 6F 20 6E 6F 69 74 63 65 74 6F 72  07CF
      2F 21 2E 30 20 65 67 61 70 20  07DB
                        2D           07B7
                                     07E5   303 ;---------------------------------------------------------------------------
                                     07E5   304 T_PAGEPROT:
20 6F 74 20 65 6C 62 61 6E 55 20 00' 07E5   305          .ASCIC  " Unable to set proper protection on buffer page.!/"
70 20 72 65 70 6F 72 70 20 74 65 73  07F1
6E 6F 20 6E 6F 69 74 63 65 74 6F 72  07FD
65 67 61 70 20 72 65 66 66 75 62 20  0809
                  2F 21 2E           0815
                        32           07E5
                                     0818   306 ;---------------------------------------------------------------------------
                                     0818   307 T_NOT_ON_KS:
72 65 4B 20 6E 6F 20 74 6F 4E 20 00' 0818   308          .ASCIC  " Not on Kernel Stack as expected.!/"
73 61 20 6B 63 61 74 53 20 6C 65 6E  0824
2F 21 2E 64 65 74 63 65 70 78 65 20  0830
                        23           0818
                                     083C   309 ;---------------------------------------------------------------------------
                                     083C   310 T_NOT_ON_IS:
74 6E 49 20 6E 6F 20 74 6F 4E 20 00' 083C   311          .ASCIC  " Not on Interrupt Stack as expected.!/"
6B 63 61 74 53 20 74 70 75 72 72 65  0848
64 65 74 63 65 70 78 65 20 73 61 20  0854
                  2F 21 2E           0860
                        26           083C
                                     0863   312 ;---------------------------------------------------------------------------
                                     0863   313 T_EXP_PSL:
50 20 64 65 74 63 65 70 78 45 20 00' 0863   314          .ASCIC  " Expected PSL    =    !_!XL(X) ; !AC!/"
21 5F 21 20 20 20 3D 20 20 20 4C 53  086F
21 43 41 21 20 3B 20 29 58 28 4C 58  087B
                        2F           0887
                        24           0863
                                     0888   315 ;---------------------------------------------------------------------------
                                     0888   316 T_ACT_PSL:
4C 53 50 20 6C 61 75 74 63 41 20 00' 0888   317          .ASCIC  " Actual PSL      =    !_!XL(X) ; !AC!/"
21 5F 21 20 20 20 3D 20 20 20 20 20  0894
21 43 41 21 20 3B 20 29 58 28 4C 58  08A0
                        2F           08AC
                        24           0888
                                     08AD   318 ;---------------------------------------------------------------------------
                                     08AD   319 T_PSL:
20 20 20 3D 20 20 20 4C 53 50 20 00' 08AD   320          .ASCIC  " PSL    =    !_!XL (X) ; !AC'/"
20 3B 20 29 58 28 20 4C 58 21 5F 21  08B9
```

```
                                2F 21 43 41 21  08C5
                                         1C  08AD
                                             08CA    321 ;-----------------------------------------------------------------------
                                             08CA    322 T_GPR:
20 20 72 65 74 73 69 67 65 52 20 00'         08CA    323         .ASCIC  " Register  Expected    Actual     XOR!/"
20 20 20 20 64 65 74 63 65 70 78 45          08D6
58 20 20 20 20 20 6C 61 75 74 63 41          08E2
                        2F 21 52 4F          08EE
                                 27          08CA
                                             08F2    324 ;-----------------------------------------------------------------------
                                             08F2    325 T_XOR:
4C 58 21 20 20 5F 21 43 41 21 20 00'         08F2    326         .ASCIC  " !AC!_  !XL   !XL   !XL!/"
4C 58 21 20 20 20 4C 58 21 20 20 20          08FE
                              2F 21          090A
                                 19          08F2
                                             090C    327 ;-----------------------------------------------------------------------
                                             090C    328 T_NOXOR:
4C 58 21 20 20 5F 21 43 41 21 20 00'         090C    329         .ASCIC  " !AC!_  !XL   !XL!/"
              2F 21 4C 58 21 20 20 20        0918
                                 13          090C
                                             0920    330 ;-----------------------------------------------------------------------
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 00'         0920    331 MM_MSG: .ASCIC  "********** UNABLE TO TURN MEMORY MANAGEMENT OFF *********"
4F 54 20 45 4C 42 41 4E 55 20 2A 2A          092C
59 52 4F 4D 45 4D 20 4E 52 55 54 20          0938
20 54 4E 45 4D 45 47 41 4E 41 4D 20          0944
2A 2A 2A 2A 2A 2A 2A 2A 20 46 46 4F          0950
                                 2A          095C
                                 3C          0920
                                             095D    332
                                             095D    333 ;-----------------------------------------------------------------------
```

```
                                 095D   335 .SBTTL  DATA TABLES
                                 095D   336
                                 095D   337 ; Table for test 1
                                 095D   338
                        00000000 095D   339 DATA_1:          .LONG   ^X0             ; expected manual restart PSL
                                 0961   340
                        041F0000 0961   341 DATA_2:          .LONG   ^X41F0000       ; expected PSL
                                 0965   342
                        041F0008 0965   343 DATA_3:          .LONG   ^X41F0008       ; expected PSL
                                 0969   344
                                 0969   345
                                 0969   346
                                 0969   347
                                 0969   348 ERR_MESSAGE:
                                 0969   349         $DS_BGNMESSAGE
                             0000 0969                 .WORD   ^M<>            ; ENTRY MASK
5E   00000084 8F     C2  096B   350         SUBL2   #BUFSIZ+4,SP            ; Reserve room on stack for cvtreg
         52   5E     D0  0972   351         MOVL    SP,R2                   ; Save starting address of buffer
                          0975   352         $DS_PRINTB_S    FORMAT=T_HALT_MESS
         F848 CF     9F  0975                 PUSHAB  T_HALT_MESS
000100E0 9F     01   FB  0979                 CALLS   #$$N, @#DS$PRINTB
                          0980   353         $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
           14 BC     9F  0980                 PUSHAB  @ERR$_P1(AP)
000100E0 9F     01   FB  0983                 CALLS   #$$N, @#DS$PRINTB
 0B F6AD CF     00   E0  098A   354         BBS     #V_READY,L_FLAGS,1$     ; br = instruction ready to execute
                          0990   355         $DS_PRINTB_S    FORMAT=T_NOT_READY
         FBAE CF     9F  0990                 PUSHAB  T_NOT_READY
000100E0 9F     01   FB  0994                 CALLS   #$$N, @#DS$PRINTB
 0B F69C CF     01   E0  099B   356 1$:     BBS     #V_DONE,L_FLAGS,2$      ; br = normal restart
                          09A1   357         $DS_PRINTB_S    FORMAT=T_NOT_DONE
         FBCF CF     9F  09A1                 PUSHAB  T_NOT_DONE
000100E0 9F     01   FB  09A5                 CALLS   #$$N, @#DS$PRINTB
 0B F68B CF     02   E1  09AC   358 2$:     BBC     #V_EXECUTE_ERR,L_FLAGS,3$ ; br = instruction executed
                          09B2   359         $DS_PRINTB_S    FORMAT=T_EXECUTE
         FBEC CF     9F  09B2                 PUSHAB  T_EXECUTE
000100E0 9F     01   FB  09B6                 CALLS   #$$N, @#DS$PRINTB
 0B F67A CF     03   E1  09BD   360 3$:     BBC     #V_CHMX_ERR,L_FLAGS,4$ ; br = no CHMx error
                          09C3   361         $DS_PRINTB_S    FORMAT=T_CHMX_ERR
         FC35 CF     9F  09C3                 PUSHAB  T_CHMX_ERR
000100E0 9F     01   FB  09C7                 CALLS   #$$N, @#DS$PRINTB
 0B F669 CF     04   E1  09CE   362 4$:     BBC     #V_XFC_ERR,L_FLAGS,5$   ; br = no XFC error
                          09D4   363         $DS_PRINTB_S    FORMAT=T_XFC
         FBEF CF     9F  09D4                 PUSHAB  T_XFC
000100E0 9F     01   FB  09D8                 CALLS   #$$N, @#DS$PRINTB
 1F F658 CF     10   E1  09DF   364 5$:     BBC     #V_ISTK,L_FLAGS,6$      ; br = on kernel stack
 2A F652 CF     0D   E1  09E5   365         BBC     #V_KSP_ERR,L_FLAGS,7$   ; br = no KSP error
                          09EB   366         $DS_PRINTB_S    FORMAT=T_KSP_ERR1,P0=L_EXP_KSP,P1=L_ACT_KSP
     00000000'EF     DD  09EB                 PUSHL   L_ACT_KSP
     00000000'EF     DD  09F1                 PUSHL   L_EXP_KSP
         FC5F CF     9F  09F7                 PUSHAB  T_KSP_ERR1
000100E0 9F     03   FB  09FB                 CALLS   #$$N, @#DS$PRINTB
                 11   11  0A02   367         BRB     7$
                          0A04   368 6$:     $DS_PRINTX_S    FORMAT=T_KSP_ERR2,P0=L_ACT_KSP
     00000000'EF     DD  0A04                 PUSHL   L_ACT_KSP
         FC7A CF     9F  0A0A                 PUSHAB  T_KSP_ERR2
000100E8 9F     02   FB  0A0E                 CALLS   #$$N, @#DS$PRINTX
 1F F622 CF     10   E0  0A15   369 7$:     BBS     #V_ISTK,L_FLAGS,8$      ; br = on interrupt stack
```

```
                2A F61C CF    0E    E1  0A1B  370          BBC     #V_ISP_ERR,L_FLAGS,9$    ; br = no ISP error
                                        0A21  371          $DS_PRINTB_S    FORMAT=T_ISP_ERR1,P0=L_EXP_ISP,P1=L_ACT_ISP
                    00000000'EF    DD   0A21                       PUSHL   L_ACT_ISP
                    00000000'EF    DD   0A27                       PUSHL   L_EXP_ISP
                        FC85 CF    9F   0A2D                       PUSHAB  T_ISP_ERR1
               000100E0 9F    03   FB   0A31                       CALLS   #$$N, @#DS$PRINTB
                             11    11   0A38  372          BRB     9$                       ; skip extened error report
                                        0A3A  373 8$:      $DS_PRINTX_S    FORMAT=T_ISP_ERR2,P0=L_ACT_ISP
                    00000000'EF    DD   0A3A                       PUSHL   L_ACT_ISP
                        FCA0 CF    9F   0A40                       PUSHAB  T_ISP_ERR2
               000100E8 9F    02   FB   0A44                       CALLS   #$$N, @#DS$PRINTX
                19 F5EC CF    0F   E1   0A4B  374 9$:      BBC     #V_SP_ERR,L_FLAGS,10$    ; br = no SP error
                                        0A51  375          $DS_PRINTB_S    FORMAT=T_SP_ERR1,P0=L_EXP_SP,P1=L_ACT_SP
                    00000000'EF    DD   0A51                       PUSHL   L_ACT_SP
                    00000000'EF    DD   0A57                       PUSHL   L_EXP_SP
                        FCB1 CF    9F   0A5D                       PUSHAB  T_SP_ERR1
               000100E0 9F    03   FB   0A61                       CALLS   #$$N, @#DS$PRINTB
                             11    11   0A68  376          BRB     11$                      ; skip extended error report
                                        0A6A  377 10$:     $DS_PRINTX_S    FORMAT=T_SP_ERR2,P0=L_ACT_SP
                    00000000'EF    DD   0A6A                       PUSHL   L_ACT_SP
                        FCCC CF    9F   0A70                       PUSHAB  T_SP_ERR2
               000100E8 9F    02   FB   0A74                       CALLS   #$$N, @#DS$PRINTX
                 03 F5BC CF    09   E0   0A7B  378 11$:     BBS     #V_PSL_ERR,L_FLAGS,12$   ; br = PSL error
                                 008A  31   0A81  379          BRW     13$                      ; no PSL error
                                        0A84  380 12$:     $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                                        0A84  381                          STRBUF=(R2),MAXLEN=#BUFSIZ,-
                                        0A84  382                          V1=MODELIST,V2=MODELIST
                                     00    DD   0A84                       PUSHL   #0
                                     00    DD   0A86                       PUSHL   #0
                                     00    DD   0A88                       PUSHL   #0
                                     00    DD   0A8A                       PUSHL   #0
                    00000000'EF    DF   0A8C                       PUSHAL  MODELIST
                    00000000'EF    DF   0A92                       PUSHAL  MODELIST
                    00000080 8F    DD   0A98                       PUSHL   #BUFSIZ
                                     62    9F   0A9E                       PUSHAB  (R2)
                    00000000'EF    9F   0AA0                       PUSHAB  A_PSLMNE
                    00000000'EF    DD   0AA6                       PUSHL   L_EXP_PSL
                                     1F    DD   0AAC                       PUSHL   #31
               000100B0 9F    0B   FB   0AAE                       CALLS   #11, @#DS$CVTREG
                                        0AB5  383          $DS_PRINTB_S    FORMAT=T_EXP_PSL,P0=L_EXP_PSL,P1=R2
                                     52    DD   0AB5                       PUSHL   R2
                    00000000'EF    DD   0AB7                       PUSHL   L_EXP_PSL
                        FDA2 CF    9F   0ABD                       PUSHAB  T_EXP_PSL
               000100E0 9F    03   FB   0AC1                       CALLS   #$$N, @#DS$PRINTB
                                        0AC8  384          $DS_CVTREG_S    MSB=#31,DATA=L_ACT_PSL,MNEADR=A_PSLMNE,-
                                        0AC8  385                          STRBUF=(R2),MAXLEN=#BUFSIZ,-
                                        0AC8  386                          V1=MODELIST,V2=MODELIST
                                     00    DD   0AC8                       PUSHL   #0
                                     00    DD   0ACA                       PUSHL   #0
                                     00    DD   0ACC                       PUSHL   #0
                                     00    DD   0ACE                       PUSHL   #0
                    00000000'EF    DF   0AD0                       PUSHAL  MODELIST
                    00000000'EF    DF   0AD6                       PUSHAL  MODELIST
                    00000080 8F    DD   0ADC                       PUSHL   #BUFSIZ
                                     62    9F   0AE2                       PUSHAB  (R2)
                    00000000'EF    9F   0AE4                       PUSHAB  A_PSLMNE
                    00000000'EF    DD   0AEA                       PUSHL   L_ACT_PSL
```

```
                1F   DD  0AF0                        PUSHL   #31
000100B0 9F     0B   FB  0AF2                        CALLS   #11, @#DS$CVTREG
                     0AF9   387          $DS_PRINTB_S    FORMAT=T_ACT_PSL,P0=L_ACT_PSL,P1=R2
                52   DD  0AF9                        PUSHL   R2
      00000000'EF   DD  0AFB                        PUSHL   L_ACT_PSL
          FD83 CF   9F  0B01                        PUSHAB  T_ACT_PSL
000100E0 9F     03   FB  0B05                        CALLS   #3$N, @#DS$PRINTB
                44   11  0B0C   388          BRB     14$
                     0B0E   389 13$:      $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                     0B0E   390                           STRBUF=(R2),MAXLEN=#BUFSIZ,-
                     0B0E   391                           V1=MODELIST,V2=MODELIST
                00   DD  0B0E                        PUSHL   #0
                00   DD  0B10                        PUSHL   #0
                00   DD  0B12                        PUSHL   #0
                00   DD  0B14                        PUSHL   #0
      00000000'EF   DF  0B16                        PUSHAL  MODELIST
      00000000'EF   DF  0B1C                        PUSHAL  MODELIST
      00000080 8F   DD  0B22                        PUSHL   #BUFSIZ
                62   9F  0B28                        PUSHAB  (R2)
      00000000'EF   9F  0B2A                        PUSHAB  A_PSLMNE
      00000000'EF   DD  0B30                        PUSHL   L_EXP_PSL
                1F   DD  0B36                        PUSHL   #31
00010080 9F     0B   FB  0B38                        CALLS   #11, @#DS$CVTREG
                     0B3F   392          $DS_PRINTX_S    FORMAT=T_PSL,P0=L_EXP_PSL,P1=R2
                52   DD  0B3F                        PUSHL   R2
      00000000'EF   DD  0B41                        PUSHL   L_EXP_PSL
          FD62 CF   9F  0B47                        PUSHAB  T_PSL
000100E8 9F     03   FB  0B4B                        CALLS   #3$N, @#DS$PRINTX
 5C F4E5 CF     0A   E1  0B52   393 14$:      BBC     #V_GPR_ERR,L_FLAGS,18$ ; br = no GPR error
                     0B58   394          $DS_PRINTB_S    FORMAT=T_GPR
          FD6E CF   9F  0B58                        PUSHAB  T_GPR
000100E0 9F     01   FB  0B5C                        CALLS   #3$N, @#DS$PRINTB
53    00000000'EF   DE  0B63   395          MOVAL   GPR_TABLE,R3            ; load address of expected data
54    00000000'EF   D0  0B6A   396          MOVL    A_HALT_BUFFER,R4        ; load address of actual data
                55   7C  0B71   397          CLRQ    R5                      ; init R5 and R6 for counter and data
             57   0D   D0  0B73   398          MOVL    #13,R7                  ; load limit
58    00000000'EF45   DE  0B76   399 15$:      MOVAL   REG_TABLE[R5],R8        ; load address of ascic string
 56    6445   6345   CD  0B7E   400          XORL3   (R3)[R5],(R4)[R5],R6    ; get XOR of data if bad
                17   13  0B84   401          BEQL    16$                     ; br = good data
                     0B86   402          $DS_PRINTB_S    FORMAT=T_XOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5],P3=R6
                56   DD  0B86                        PUSHL   R6
              6445   DD  0B88                        PUSHL   (R4)[R5]
              6345   DD  0B8B                        PUSHL   (R3)[R5]
                58   DD  0B8E                        PUSHL   R8
          FD5E CF   9F  0B90                        PUSHAB  T_XOR
000100E0 9F     05   FB  0B94                        CALLS   #3$N, @#DS$PRINTB
                13   11  0B9B   403          BRB     17$                     ; check if finished
                     0B9D   404 16$:      $DS_PRINTX_S    FORMAT=T_NOXOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5]
              6445   DD  0B9D                        PUSHL   (R4)[R5]
              6345   DD  0BA0                        PUSHL   (R3)[R5]
                58   DD  0BA3                        PUSHL   R8
          FD63 CF   9F  0BA5                        PUSHAB  T_NOXOR
000100E8 9F     04   FB  0BA9                        CALLS   #3$N, @#DS$PRINTX
      C2 55     57   F3  0BB0   405 17$:      AOBLEQ  R7,R5,15$
 0B F483 CF     05   E1  0BB4   406 18$:      BBC     #V_TIMER,L_FLAGS,19$    ; br = no timer error
                     0BBA   407          $DS_PRINTB_S    FORMAT=T_TIMER
          FA74 CF   9F  0BBA                        PUSHAB  T_TIMER
```

```
                000100E0 9F    01    FB  0BBE                        CALLS   #$$N, a#DS$PRINTB
                                     0BC5   408 19$:
                                     0BC5   409          $DS_ENDMESSAGE
                                  04 0BC5                RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                     0BC6   410
```

```
                                0BC6   412 .SBTTL  SUBROUTINES
                                0BC6   413
                                0BC6   414
                                0BC6   415 ; This routine will clear the buffer area.
                                0BC6   416
                                0BC6   417 CLEAR_BUFFER:
          55   00000000'FF    DE  0BC6   418          MOVAL   @A_HALT_BUFFER,R5 ; get starting buffer address
                    F447 CF    D4  0BCD   419          CLRL    L_COUNT           ; initialize counter
                         85    D4  0BD1   420 1$:      CLRL    (R5)+             ; clear a buffer location
      F4 F43C CF    00000100 8F  F3  0BD3   421          AOBLEQ  #256,L_COUNT,1$ ; br = not finished
                                05  0BDD   422          RSB                       ; return
                                0BDE   423
                                0BDE   424 ; This routine will load general purpose registers R0 to R11 with data and
                                0BDE   425 ; read the AP and FP befor a halt is tested.
                                0BDE   426
                                0BDE   427 LOAD_GPR:
          5B   00000000'EF    DE  0BDE   428          MOVAL   GPR_TABLE,R11     ; load address of data table to load GPR's
                         50   8B  7D  0BE5   429          MOVQ    (R11)+,R0         ; move data into R0 and R1
                         52   8B  7D  0BE8   430          MOVQ    (R11)+,R2         ; move data into R2 and R3
                         54   8B  7D  0BEB   431          MOVQ    (R11)+,R4         ; move data into R4 and R5
                         56   8B  7D  0BEE   432          MOVQ    (R11)+,R6         ; move data into R6 and R7
                         58   8B  7D  0BF1   433          MOVQ    (R11)+,R8         ; move data into R8 and R9
                         5A   8B  7D  0BF4   434          MOVQ    (R11)+,R10        ; move data into R10 and R11
          00000000'EF    5C  D0  0BF7   435          MOVL    AP,L_EXP_AP       ; move contents of AP into expected data table
          00000000'EF    5D  D0  0BFE   436          MOVL    FP,L_EXP_FP       ; move contents of FP into expected data table
                                05  0C05   437          RSB                       ; return
                                0C06   438
                                0C06   439 ; This routine will read general purpose registers R0 to R11 , the AP, FP, SP
                                0C06   440 ; KSP, ISP and PSL into a buffer area (A_HALT_BUFFER) to check data patterns
                                0C06   441 ; and data after a halt. It will then retrun to to address contained in
                                0C06   442 ; RENTRY_POINT.
                                0C06   443
                                0C06   444 READ_GPR:
       00000000'FF    50   7D  0C06   445          MOVQ    R0,@A_HALT_BUFFER ; move contents of R0 and R1 into buffer
       50   00000000'EF    D0  0C0D   446          MOVL    A_HALT_BUFFER,R0 ; load R0 with address of buffer
                 50   08   C0  0C14   447          ADDL2   #8,R0             ; get over first two words
                 80   52   7D  0C17   448          MOVQ    R2,(R0)+          ; move contents of R2 and R3 into buffer
                 80   54   7D  0C1A   449          MOVQ    R4,(R0)+          ; move R4 and R5
                 80   56   7D  0C1D   450          MOVQ    R6,(R0)+          ; move R6 and R7
                 80   58   7D  0C20   451          MOVQ    R8,(R0)+          ; move R8 and R9
                 80   5A   7D  0C23   452          MOVQ    R10,(R0)+         ; move R10 and R11
                 80   5C   D0  0C26   453          MOVL    AP,(R0)+          ; move AP
                 80   5D   D0  0C29   454          MOVL    FP,(R0)+          ; move FP
       00000000'EF    5E   D0  0C2C   455          MOVL    SP,L_ACT_SP       ; move SP
00000000'EF    00000000'8F  DB  0C33   456          MFPR    #PR$_KSP,L_ACT_KSP ; move KSP
00000000'EF    00000000'8F  DB  0C3E   457          MFPR    #PR$_ISP,L_ACT_ISP ; move ISP
          00000000'EF    DC  0C49   458          MOVPSL  L_ACT_PSL         ; move PSL
               F3B1 DF    17  0C4F   459          JMP     @RENTRY_POINT     ; return to subtest
                                0C53   460
                                0C53   461
                                0C53   462 ; This routine will service unexpected interval timer interrupts and set an
                                0C53   463 ; error flag.
                                0C53   464
                                0C53   465          .ALIGN  LONG
                                0C54   466 TIMER_SERVICE:
            F3E3 CF    20   C8  0C54   467          BISL2   #M_TIMER,L_FLAGS ; set error flag
                                02  0C59   468          REI                       ; return
```

```
                                0C5A   469
                                0C5A   470
                                0C5A   471 ; This routine will service unexpected CHANGE MODE exceptions and set
                                0C5A   472 ; an error flag.
                                0C5A   473
                                0C5A   474          .ALIGN  LONG
                                0C5C   475 CHMX_SERVICE:
                F3D3 DF   01    CA     0C5C   476          BICL2   #1,@L_TEMP              ; set service on inter. stack
                F3D6 CF   08    C8     0C61   477          BISL2   #M_CHMX_ERR,L_FLAGS ; set error flag
                          8E    D5     0C66   478          TSTL    (SP)+                   ; get change mode argument off stack
                          02    0C68   479          REI                             ; return
                                0C69   480
                                0C69   481 ; This routine will service unexpected XFC exceptions and set a flag
                                0C69   482 ; to indicate the error.
                                0C69   483
                                0C69   484          .ALIGN  LONG
                                0C6C   485 XFC_ERROR:
                F3CB CF   10    C8     0C6C   486          BISL2   #M_XFC_ERR,L_FLAGS ; set error flag
                     6E   01    C0     0C71   487          ADDL2   #1,(SP)                 ; update the pc for the REI
                          02    0C74   488          REI                             ; return
                                0C75   489
                                0C75   490 ; This routine will service expected XFC exceptions and set a flag to
                                0C75   491 ; indicate successful completion of the operation.
                                0C75   492
                                0C75   493          .ALIGN  LONG
                                0C78   494 XFC_SERVICE:
          F3BB CF   00000800    8F     C8     0C78   495          BISL2   #M_XFC,L_FLAGS  ; set flag
                     6E   01    C0     0C81   496          ADDL2   #1,(SP)                 ; update the pc for the REI
                          02    0C84   497          REI                             ; return
                                0C85   498
                                0C85   499
                                0C85   500 ; This routine is called after execution of an instruction which should cause
                                0C85   501 ; a halt. The PSL, SP, KSP, ISP, GENERAL PURPOSE REGISTERS and FLAG BYTE are
                                0C85   502 ; examined for expected information. If an error exists, a flag is set to
                                0C85   503 ; indicate that it should be reported.
                                0C85   504
                                0C85   505 ERROR_CHECK::
                          57    D4     0C85   506          CLRL    R7                      ; initialize counter
                     55   0D    D0     0C87   507          MOVL    #13,R5                  ; set limit
               53   00000000'EF    D0     0C8A   508          MOVL    A_HALT_BUFFER,R3 ; load address of actual data to check
               54   00000000'EF    DE     0C91   509          MOVAL   GPR_TABLE,R4    ; load address of expected data
                     84   83    D1     0C98   510 1$:       CMPL    (R3)+,(R4)+     ; check data
                          09    13     0C9B   511          BEQL    2$                      ; br = data is good
          F396 CF   00001400    8F     C8     0C9D   512          BISL2   #M_ERROR_FLAG!M_GPR_ERR,L_FLAGS ; set GPR and error flag
                      EE 57    55     F3     0CA6   513 2$:       AOBLEQ  R5,R7,1$        ; br = not finished
00000000'EF   00000000'EF    D1     0CAA   514          CMPL    L_ACT_SP,L_EXP_SP ; check SP
                          09    13     0CB5   515          BEQL    3$                      ; br = SP is good
          F37C CF   00009000    8F     C8     0CB7   516          BISL2   #M_SP_ERR!M_ERROR_FLAG,L_FLAGS ; set SP and error flag
            1F 00000000'EF    1A     E1     0CC0   517 3$:       BBC     #PSL$V_IS,L_ACT_PSL,4$ ; br = on KS no possible error
          F36B CF   00010000    8F     C8     0CC8   518          BISL2   #M_ISTK,L_FLAGS ; set interrupt stack flag
00000000'EF   00000000'EF    D1     0CD1   519          CMPL    L_ACT_KSP,L_EXP_KSP ; check KSP
                          09    13     0CDC   520          BEQL    4$                      ; br = KSP is good
          F355 CF   00003000    8F     C8     0CDE   521          BISL2   #M_KSP_ERR!M_ERROR_FLAG,L_FLAGS ; set KSP and error flag
                16 F350 CF    10     E0     0CE7   522 4$:       BBS     #V_ISTK,L_FLAGS,5$ ; br = on IS no possible error
00000000'EF   00000000'EF    D1     0CED   523          CMPL    L_ACT_ISP,L_EXP_ISP ; check ISP
                          09    13     0CF8   524          BEQL    5$                      ; br = ISP is good
          F339 CF   00005000    8F     C8     0CFA   525          BISL2   #M_ISP_ERR!M_ERROR_FLAG,L_FLAGS ; set ISP and error flag
```

```
00000000'EF   00000000'EF   D1  0D03   526 5$:     CMPL    L_ACT_PSL,L_EXP_PSL ; check PSL
                      42    13  0D0E   527         BEQL    8$                  ; br = PSL is good
                   02   18   EF  0D10   528         EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,-
      F309 CF      00000000'EF       0D13   529                 L_EXP_PSL,L_EXP_MOD ; extract current mode from expected psl
                   02   18   ED  0D1B   530         CMPZV   #PSL$V_CURMOD,#PSL$S_CURMOD,-
      F2FE CF      00000000'EF       0D1E   531                 L_ACT_PSL,L_EXP_MOD ; check current mode for error
                      18    12  0D26   532         BNEQ    6$                  ; br = wrong mode
                   01   1A   EF  0D28   533         EXTZV   #PSL$V_IS,#1,-
      F2F5 CF      00000000'EF       0D2B   534                 L_EXP_PSL,L_EXP_STACK ; extract stack from expected PSL
                   01   1A   ED  0D33   535         CMPZV   #PSL$V_IS,#1,-
      F2EA CF      00000000'EF       0D36   536                 L_ACT_PSL,L_EXP_STACK ; check current stack for error
                      09    13  0D3E   537         BEQL    7$                  ; br = mode is good
      F2F3 CF      00000100 8F   C8  0D40   538 6$:     BISL2   #M_FATAL_ERR,L_FLAGS ; signal fatal error
      F2EA CF      00001200 8F   C8  0D49   539 7$:     BISL2   #M_PSL_ERR!M_ERROR_FLAG,L_FLAGS ; set PSL and error flag
            09 F2E5 CF    00   E0  0D52   540 8$:     BBS     #V_READY,L_FLAGS,9$ ; br = no ready error
      F2DB CF      00001000 8F   C8  0D58   541         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
            09 F2D6 CF    02   E1  0D61   542 9$:     BBC     #V_EXECUTE_ERR,L_FLAGS,10$ ; br = no execute error
      F2CC CF      00001000 8F   C8  0D67   543         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
            09 F2C7 CF    01   E0  0D70   544 10$:    BBS     #V_DONE,L_FLAGS,11$ ; br = no done error
      F2BD CF      00001000 8F   C8  0D76   545         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
            09 F2B8 CF    03   E1  0D7F   546 11$:    BBC     #V_CHMX_ERR,L_FLAGS,12$ ; br = no unexpected CHMx exception
      F2AE CF      00001000 8F   C8  0D85   547         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
            09 F2A9 CF    04   E1  0D8E   548 12$:    BBC     #V_XFC_ERR,L_FLAGS,13$ ; br = no unexpected XFC exception
      F29F CF      00001000 8F   C8  0D94   549         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
            09 F29A CF    05   E1  0D9D   550 13$:    BBC     #V_TIMER,L_FLAGS,14$ ; br = no unexpected timer interrupt
      F290 CF      00001000 8F   C8  0DA3   551         BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
                                   05  0DAC   552 14$:    RSB                 ; return
                                       0DAD   553
                                       0DAD   554         $DS_PAGE
```

```
                        0DAD                    .SBTTL  TEST 3:  PROCESSOR HALTS ARCH. DEFINED
                     00000000                   .PSECT  TEST_003, PAGE, NOWRT
                        0020
                        0020    557 ;++
                        0020    558 ; Test description:
                        0020    559 ;
                        0020    560 ;       This test checks the various ways the VAX-11/730 processor can
                        0020    561 ;       be halted in an architecturally defined manner. This includes
                        0020    562 ;       executing a HALT instruction in kernel mode, attempting
                        0020    563 ;       certain references to an invalid interrupt stack and executing
                        0020    564 ;       change mode instructions when the IS bit of the PSL is set.
                        0020    565 ;
                        0020    566 ;       INSTRUCTIONS FOR EARLY ABORT:  If it is desired to exit the test
                        0020    567 ;       before completion, one must wait for the console prompt ( >>> )
                        0020    568 ;       after the halt occurs and type in the sequence D PC adr <CR>
                        0020    569 ;       C <CR>, where adr is the absolute address of the label
                        0020    570 ;       TEST_003_X:: in the program listing.
                        0020    571 ;
                        0020    572 ;--
                        0020    573
                        0020    574
                        0020    575         $DS_BGNTEST     <MANUAL,HALT,ALL>
                        0020        DATA_003:
                     00000000   0020                .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                        0024        TEST_003::
                        0000    0024                .WORD   ^M<>            ; ENTRY MASK
                        0026    576
          00000000'EF   D5      0026    577         TSTL    KA_PTABLE       ; CPU selected?
                  10    18      002C    578         BGEQ    5$              ; br = CPU selected
                        002E    579         $DS_PRINTF_S    FORMAT=T_NO_KA730 ; print exit warning
          0000004D'EF   9F      002E                PUSHAB  T_NO_KA730
000100F0  9F    01    FB      0034                CALLS   #5$N, @#DS$PRINTF
                        003B    580         $DS_EXIT        TEST            ; exit test
                  0B22  31      003B                BRW     TEST_003_X      ; EXIT TEST 3
                        003E    581 ;
                        003E    582 ; Ask operator if Auto Restart Switch is in off position, continue if YES, exit
                        003E    583 ; test if NO.
                        003E    584 ;
                        003E    585 5$:     $DS_PRINTF_S    FORMAT=T_WARNING        ; print warning
          00000129'EF   9F      003E                PUSHAB  T_WARNING
000100F0  9F    01    FB      0044                CALLS   #5$N, @#DS$PRINTF
                        004B    586
                        004B    587         $DS_ASKLGCL_S   MSGADR=T_REST_OFF,-      ; ask operator
                        004B    588                         DATADR=B_ANSWER,-       ; location of answer
                        004B    589                         YEXFER=7$,-             ; br to 7$ if YES
                        004B    590                         DEFALT=PAR$_NO          ; default is NO
                     00   DD      004B                PUSHL   #0
          00000000'EF   DD      004D                PUSHL   PAR$_NO
                     00   DD      0053                PUSHL   #0
          00000080'EF   DF      0055                PUSHAL  7$
                     00   DD      005B                PUSHL   #0
          00000000'EF   DF      005D                PUSHAL  B_ANSWER
          00000191'EF   DF      0063                PUSHAL  T_REST_OFF
00010098  9F    07    FB      0069                CALLS   #7, @#DS$ASKLGCL
                        0070    591
                        0070    592         $DS_PRINTF_S    FORMAT=T_EXITING        ; print warning
          00000114'EF   9F      0070                PUSHAB  T_EXITING
```

```
                000100F0 9F    01    FB  0076                        CALLS   #$$N, a#DS$PRINTF
                                         007D   593          $DS_EXIT        TEST                    ; exit test if NO
                         0AE0  31        007D                        BRW     TEST_003_X      ; EXIT TEST 3
                                         0080   594
                00000000'EF    5E    D0  0080   595 7$:      MOVL    SP,L_KSP                ; save initial KSP
     00000000'EF    00000000'8F    DB  0087   596          MFPR    #PR$_ISP,L_ISP          ; save initial ISP
                00000BC6'EF    16    92  0098   597          JSB     CLEAR_BUFFER            ; clear buffer area
                                         0098   598          $DS_SETVEC_S    VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                         00    DD    98                              PUSHL   #0
                00000C54'EF    DF    9A                              PUSHAL  TIMER_SERVICE
                000000C0 8F    DD    A0                              PUSHL   #SCB$L_TIMER
                00010160 9F    03    FB  00A6                        CALLS   #3, a#DS$SETVEC
                                         00AD   599 10$:
                                         00AD   600          $DS_PAGE
```

```
                                00AD                    .SBTTL  SUBTEST 3.1:  HALT Inst. in Kernel Mode
                                00AD    603 ;+
                                00AD    604 ; Subtest description:
                                00AD    605 ;
                                00AD    606 ;       This subtest checks that the HALT instruction halts the processor
                                00AD    607 ;       when executed in kernel mode (code = 6).
                                00AD    608 ;
                                00AD    609 ;
                                00AD    610 ; Subtest steps:
                                00AD    611 ;
                                00AD    612 ;       SUBTEST Halt Instruction in Kernel Mode
                                00AD    613 ;       :  Initialize Flags
                                00AD    614 ;       :  Setup code to set done flag at loc 100
                                00AD    615 ;       :  Setup a jump to READ_GPR at next loc
                                00AD    616 ;       :  Print instructions to operator
                                00AD    617 ;       :  Load expected data for stacks
                                00AD    618 ;       :  Load GPR's with data
                                00AD    619 ;       :  Set ready to execute flag
                                00AD    620 ;       :  Execute HALT
                                00AD    621 ;       :  IF halt not executed properly
                                00AD    622 ;       :  :  THEN
                                00AD    623 ;       :  :  (will continue inline execution and get here)
                                00AD    624 ;       :  :  Set halt execution error flag
                                00AD    625 ;       :  ENDIF
                                00AD    626 ;       :  Rentry point (restart will enter here)
                                00AD    627 ;       :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
                                00AD    628 ;       :  IF any errors
                                00AD    629 ;       :  :  THEN
                                00AD    630 ;       :  :  Report errors
                                00AD    631 ;       :  ENDIF
                                00AD    632 ;       :  IF fatal error
                                00AD    633 ;       :  :  THEN
                                00AD    634 ;       :  :  Abort test
                                00AD    635 ;       :  ENDIF
                                00AD    636 ;       ENDSUBTEST Halt instruction in kernel mode
                                00AD    637 ;
                                00AD    638 ;-
                                00AD    639
                                00AD    640             $DS_BGNSUB
                                00AD            T3_S1::
00010030 9F   00000000'EF   FA  00AD                    CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  00B8    641             JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  00BE    642             CLRL    L_FLAGS                 ; initialize flags
00000100'EF   00000040'EF   7D  00C4    643             MOVQ    JUMP,RESTART            ; load loc 100 with rentry code
00000108'EF   00000048'EF   7D  00CF    644             MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
00000004'EF   00000143'EF   DE  00DA    645             MOVAL   RENTRY1,RENTRY_POINT    ; load rentry address
                                00E5    646             $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS, P0=#CODE_06,P1=#4$
              0000013C'8F   DD  00E5                    PUSHL   #4$
                    06      DD  00EB                    PUSHL   #CODE_06
              00000085'EF   9F  00ED                    PUSHAB  T_INSTRUCTIONS
         000100F0 9F    03  FB  00F3                    CALLS   #$$N, @#DS$PRINTF
                                00FA    647             $DS_PRINTF_S    FORMAT=T_CONTINUE
              000000C3'EF   9F  00FA                    PUSHAB  T_CONTINUE
         000100F0 9F    01  FB  0100                    CALLS   #$$N, @#DS$PRINTF
00000000'EF   00000000'EF   D0  0107    648             MOVL    L_KSP,L_EXP_SP          ; load expected SP
              00000000'EF   D4  0112    649             CLRL    L_EXP_KSP               ; KSP unpredictable if on kernel stack
00000000'EF   00000000'EF   D0  0118    650             MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
```

```
                00000000'EF   0000095D'EF   D0  0123   651          MOVL    DATA_1,L_EXP_PSL          ; load expected PSL
                     00000BDE'EF   16  012E   652          JSB     LOAD_GPR                  ; load GPR data and read AP & FP
                0000003C'EF   01   C8  0134   653          BISL2   #M_READY,L_FLAGS          ; set ready to execute flag
                              00   00  0138   654          HALT                              ; execute HALT instruction (should
                                        013C   655                                            ; halt and continue at rentry point)
                0000003C'EF   04   C8  013C   656 4$:      BISL2   #M_EXECUTE_ERR,L_FLAGS    ; halt not executed, set error flag
                                        0143   657 RENTRY1:                                  ; rentry point after executing halt
                5E   00000000'EF   D0  0143   658          MOVL    L_KSP,SP                  ; reset SP
                     00000C85'EF   16  014A   659          JSB     ERROR_CHECK               ; check for errors
             27 0000003C'EF   0C   E1  0150   660          BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no errors
                                        0158   661          $DS_ERRHARD_S   UNIT =KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_HALT_ERR
                     000001F1'8F   DD  0158                    PUSHL   #T_HALT_ERR
                     00000969'EF   DF  015E                    PUSHAL  ERR_MESSAGE
                               00   DD  0164                    PUSHL   #0
                     00000000'EF   DD  0166                    PUSHL   KA_UNIT_NO
                               01   DD  016C                    PUSHL   #$ER
                                        016E              ;;; TEST 3, SUBTEST 1, ERROR 1
                0001000D0 9F   05   FB  016E                    CALLS   #$$M, @#DS$ERRHARD
                                        0175   662          $DS_BCOMPLETE   5$
                           07 50   E8  0175                    BLBS    R0, 5$
                                        0178   663          $DS_ABORT
                00010020 9F   6C   FA  0178                    CALLG   (AP), @#DS$ABORT
             07 0000003C'EF   08   E1  017F   664 5$:      BBC     #V_FATAL_ERR,L_FLAGS,6$ ; br = no fatal errors
                                        0187   665          $DS_ABORT
                00010020 9F   6C   FA  0187                    CALLG   (AP), @#DS$ABORT
                                        018E   666 6$:      $DS_ENDSUB
                                        018E              T3_S1_X:
        00010038 9F   00000000'EF   FA  018E                    CALLG   $S$, @#DS$ENDSUB
                                        0199   667          $DS_PAGE
```

```
0199                    .SBTTL  SUBTEST 3.2:  Interrupt Stack invalid
0199    670 ;+
0199    671 ; Subtest description:
0199    672 ;
0199    673 ;       This subtest checks that an interrupt stack which is located on
0199    674 ;       a page which is protected to kernel read only, will halt the processor
0199    675 ;       when a write is attempted. An XFC instruction is executed to get onto
0199    676 ;       the interrupt stack prior to invalidating the interrupt stack pointer.
0199    677 ;       A second XFC instruction is executed with the vector specifying the
0199    678 ;       interrupt stack for service. When the XFC instruction is executed
0199    679 ;       there is an attempt to write the PC and PSL on the interrupt stack.
0199    680 ;       When this happens an "Interrupt Stack not valid HALT" should occur
0199    681 ;       (halt code = 4). Memory management is on for this subtest.
0199    682 ;
0199    683 ; Subtest steps:
0199    684 ;
0199    685 ;       SUBTEST Interrupt Stack invalid
0199    686 ;       :  Initialize Flags
0199    687 ;       :  Turn memory management on
0199    688 ;       :  Set page0 protection to kernel write access
0199    689 ;       :  IF unsucessful
0199    690 ;       :  :  THEN
0199    691 ;       :  :  Report system error
0199    692 ;       :  :  Abort program
0199    693 ;       :  ENDIF
0199    694 ;       :  Get XFC vector to handle on interrupt stack
0199    695 ;       :  Execute XFC instruction
0199    696 ;       :  IF not on the interrupt stack
0199    697 ;       :  :  THEN
0199    698 ;       :  :  Report system error
0199    699 ;       :  :  Abort program
0199    700 ;       :  ENDIF
0199    701 ;       :  Release XFC vector
0199    702 ;       :  Get XFC vector to handle on interrupt stack
0199    703 ;       :  Setup code to set done flag at loc 100
0199    704 ;       :  Setup a jump to READ_GPR at next loc
0199    705 ;       :  Save current IS SP to reset after subtest is complete
0199    706 ;       :  Set interrupt stack pointer to buffer area
0199    707 ;       :  Invalidate buffer page (kernel read only access)
0199    708 ;       :  IF unsucessful
0199    709 ;       :  :  THEN
0199    710 ;       :  :  Report system error
0199    711 ;       :  :  Abort program
0199    712 ;       :  ENDIF
0199    713 ;       :  Print instructions to operator
0199    714 ;       :  Setup all expected data
0199    715 ;       :  Load GPR's with data
0199    716 ;       :  Set ready to execute flag
0199    717 ;       :  Execute XFC to attempt to write PC and PSL on IS
0199    718 ;       :  (this should cause an interrupt stack not valid HALT)
0199    719 ;       :  IF XFC not executed properly
0199    720 ;       :  :  THEN
0199    721 ;       :  :  (will continue in line execution and get here)
0199    722 ;       :  :  Set XFC execution error flag
0199    723 ;       :  ENDIF
0199    724 ;       :  Rentry point (restart will enter here)
0199    725 ;       :  Read  PSL
```

```
                                0199   726 ;       :  IF on IS as expected
                                0199   727 ;       :  :  THEN
                                0199   728 ;       :  :  Restore IS SP
                                0199   729 ;       :  ENDIF
                                0199   730 ;       :  Go check FLAGS, PSL, KSP, ISP, SP and GPR'S for errors
                                0199   731 ;       :   (Either the KSP or ISP will be UNPREDICTABLE depending on the
                                0199   732 ;       :   current stack, this is machine dependant)
                                0199   733 ;       :  IF any errors
                                0199   734 ;       :  :  THEN
                                0199   735 ;       :  :  Report errors
                                0199   736 ;       :  ENDIF
                                0199   737 ;       :  IF fatal error
                                0199   738 ;       :  :  THEN
                                0199   739 ;       :  :  Abort program
                                0199   740 ;       :  ENDIF
                                0199   741 ;       :  Reset protection on buffer page to original protection
                                0199   742 ;       :  Turn memory management off
                                0199   743 ;       :  Release XFC vector
                                0199   744 ;       ENDSUBTEST Interrupt Stack invalid
                                0199   745 ;
                                0199   746 ;-
                                0199   747
                                0199   748         $DS_BGNSUB
                                0199        T3_S2::
    00010030 9F   0000000C'EF   FA  0199                    CALLG   $S$, @#DS$BGNSUB
                  00000BC6'EF   16  01A4   749         JSB     CLEAR_BUFFER            ; clear data buffers
                  0000003C'EF   D4  01AA   750         CLRL    L_FLAGS                 ; initialize flags
                  00000010'EF   D4  01B0   751         CLRL    L_PAGEZERO              ; set page zero to 0
                                01B6   752         $DS_MMON_S                      ; turn memory management on
    00010150 9F   00            FB  01B6                    CALLS   #0, @#DS$MMON
                                01BD   753         $SETPRT_S       INADR=L_PAGEZERO,PROT=#PRT$C_KW
                  00            DD  01BD                    PUSHL   #0
              00000000'8F       DD  01BF                    PUSHL   #PRT$C_KW
                  00            DD  01C5                    PUSHL   #0
                  00            DD  01C7                    PUSHL   #0
              00000010'EF       7F  01C9                    PUSHAQ  L_PAGEZERO
    00010430'GF   05            FB  01CF                    CALLS   #5,G^SYS$SETPRT
                                01D6   754         $DS_BCOMPLETE   1$
                1A 50           E8  01D6                    BLBS    R0, 1$
                                01D9   755         $DS_ERRSYS_S    UNIT=#0,MSGADR=T_PAGEOPROT
                  00            DD  01D9                    PUSHL   #0
              000007B7'EF       DF  01DB                    PUSHAL  T_PAGEOPROT
                  00            DD  01E1                    PUSHL   #0
                  01            DD  01E3                    PUSHL   #$ER
                                01E5       ;;; TEST 3, SUBTEST 2, ERROR 1
    000100C0 9F   04            FB  01E5                    CALLS   #$$M, @#DS$ERRSYS
                                01EC   756         $DS_ABORT
    00010020 9F   6C            FA  01EC                    CALLG   (AP), @#DS$ABORT
                                01F3   757 1$:      $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY2,CODE=#1
                  01            DD  01F3                    PUSHL   #1
              00000208'EF       DF  01F5                    PUSHAL  RENTRY2
                  14            DD  01FB                    PUSHL   #SCB$L_OPCCUS
    00010160 9F   03            FB  01FD                    CALLS   #3, @#DS$SETVEC
                                FC  0204   758 2$:      XFC
                                0205   759         .ALIGN  LONG
                                0208   760 RENTRY2:
              00000000'EF       DC  0208   761         MOVPSL  L_ACT_PSL
```

```
                1E 00000000'EF    1A    E0  020E   762          BBS     #PSL$V_IS,L_ACT_PSL,3$
                                                0216   763          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                                   00    DD  0216                    PUSHL   #0
                        0000083C'EF    DF  0218                    PUSHAL  T_NOT_ON_IS
                        00000000'EF    DD  021E                    PUSHL   KA_UNIT_NO
                                   02    DD  0224                    PUSHL   #$ER
                                                0226       ;;; TEST 3, SUBTEST 2, ERROR 2
                00100D0 9F      04    FB  0226                    CALLS   #$$M, @#DS$ERRHARD
                                                022D   764          $DS_ABORT
                00010020 9F     6C    FA  022D                    CALLG   (AP), @#DS$ABORT
                                                0234   765 3$:      $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                                   14    DD  0234                    PUSHL   #SCB$L_OPCCUS
                00010168 9F     01    FB  0236                    CALLS   #1, @#DS$CLRVEC
                                                023D   766          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#1
                                   01    DD  023D                    PUSHL   #1
                        00000C78'EF    DF  023F                    PUSHAL  XFC_SERVICE
                                   14    DD  0245                    PUSHL   #SCB$L_OPCCUS
                00010160 9F     03    FB  0247                    CALLS   #3, @#DS$SETVEC
    00000100'EF    00000040'EF    7D  024E   767 4$:      MOVQ    JUMP,RESTART                ; setup code to renter after halt
    00000108'EF    00000048'EF    7D  0259   768          MOVQ    JUMP+8,RESTART+8            ; load rest of rentry code
    00000004'EF    00000343'EF    DE  0264   769          MOVAL   RENTRY3,RENTRY_POINT        ; load rentry address
                00000020'EF    5E    D0  026F   770          MOVL    SP,L_CUR_ISP                ; read the current IS SP
                                                0276   771 5$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_04,P1=#8$
                        0000033B'8F    DD  0276                    PUSHL   #8$
                                   04    DD  027C                    PUSHL   #CODE_04
                        00000085'EF    9F  027E                    PUSHAB  T_INSTRUCTIONS
                00100F0 9F      03    FB  0284                    CALLS   #$$N, @#DS$PRINTF
                                                028B   772          $DS_PRINTF_S    FORMAT=T_CONTINUE
                        000000C3'EF    9F  028B                    PUSHAB  T_CONTINUE
                00100F0 9F      01    FB  0291                    CALLS   #$$N, @#DS$PRINTF
    0000001C'EF    00000000'EF    D0  0298   773 6$:      MOVL    A_BUFFER_BEGIN,L_ADDRESS ; get starting address of buffer
    00000008'EF    0000001C'EF    D0  02A3   774          MOVL    L_ADDRESS,L_PAGESTRT        ; load address of page to change
    0000000C'EF    0000001C'EF    D0  02AE   775          MOVL    L_ADDRESS,L_PAGEND          ; protection in 2 longword array
                0000001C'EF    28    C0  02B9   776          ADDL2   #40,L_ADDRESS               ; point into protected page
                                                02C0   777          $SETPRT_S INADR=L_PAGESTRT,PROT=#PRT$C_KR,PRVPRT=L_OLDPROT
                        0000002C'EF    DF  02C0                    PUSHAL  L_OLDPROT
                        00000000'8F    DD  02C6                    PUSHL   #PRT$C_KR
                                   00    DD  02CC                    PUSHL   #0
                                   00    DD  02CE                    PUSHL   #0
                        00000008'EF    7F  02D0                    PUSHAQ  L_PAGESTRT
                00010430'GF     05    FB  02D6                    CALLS   #5,G^SYS$SETPRT
                                                02DD   778          $DS_BCOMPLETE   7$
                                1E 50   E8  02DD                    BLBS    R0, 7$
                                                02E0   779          $DS_ERRSYS_S    UNIT=KA_UNIT_NO, MSGADR=T_PAGEPROT
                                   00    DD  02E0                    PUSHL   #0
                        000007E5'EF    DF  02E2                    PUSHAL  T_PAGEPROT
                        00000000'EF    DD  02E8                    PUSHL   KA_UNIT_NO
                                   03    DD  02EE                    PUSHL   #$ER
                                                02F0       ;;; TEST 3, SUBTEST 2, ERROR 3
                00100C0 9F      04    FB  02F0                    CALLS   #$$M, @#DS$ERRSYS
                                                02F7   780          $DS_ABORT
                00010020 9F     6C    FA  02F7                    CALLG   (AP), @#DS$ABORT
                                                02FE   781
00000000'EF    0000001C'EF      04    C3  02FE   782 7$:      SUBL3   #4,L_ADDRESS,L_EXP_SP   ; load expected SP
     00000000'EF    00000000'EF    D0  030A   783          MOVL    L_KSP,L_EXP_KSP             ; load expected KSP
                00000000'EF    D4  0315   784          CLRL    L_EXP_ISP                   ; ISP unpredictable if on int. stack
     00000000'EF    00000961'EF    D0  031B   785          MOVL    DATA_2,L_EXP_PSL            ; load expected PSL
```

```
                                0326   786
                00000BDE'EF   16  0326   787          JSB     LOAD_GPR                        ; load GPR's with data and read AP, FP
           0000003C'EF   01   C8  032C   788          BISL2   #M_READY,L_FLAGS                 ; set ready to execute flag
           5E   0000001C'EF   D0  0333   789          MOVL    L_ADDRESS,SP                    ; point IS SP to that address
                              01  033A   790          NOP                                     ; wait a little
                              FC  033B   791 8$:       XFC                                     ; execute XFC instruction to cause
                                  033C   792                                                  ; exception which will attempt to push
                                  033C   793                                                  ; on an invalid IS, renter at RENTRY2
                                  033C   794
           0000003C'EF   04   C8  033C   795          BISL2   #M_EXECUTE_ERR,L_FLAGS          ; set flag, XFC did not work properly
                                  0343   796
                                  0343   797 RENTRY3:
                                  0343   798
        11 00000000'EF   1A   E1  0343   799          BBC     #PSL$V_IS,L_ACT_PSL,10$         ; br = not on IS as expected
           5E   00000020'EF   D0  034B   800          MOVL    L_CUR_ISP,SP                    ; reset IS SP
           6E   0000035A'8F   D0  0352   801          MOVL    #9$,(SP)                        ; setup return address
                              02  0359   802          REI                                     ; get off interrupt stack
                           0B   11  035A   803 9$:     BRB     11$                             ; skip IPR load
     00000000'8F   00000020'EF   DA  035C   804 10$:    MTPR    L_CUR_ISP,#PR$_ISP              ; load IS SP into IPR
                00000C85'EF   16  0367   805 11$:    JSB     ERROR_CHECK                     ; check for errors (GPR's,AP,FP,PSL)
        24 0000003C'EF   0C   E1  036D   806          BBC     #V_ERROR_FLAG,L_FLAGS,12$       ; br = no errors
                                  0375   807          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_INVLDIS_ERR
                00000215'8F   DD  0375                       PUSHL   #T_INVLDIS_ERR
                00000969'EF   DF  037B                       PUSHAL  ERR_MESSAGE
                           00   DD  0381                       PUSHL   #0
                00000000'EF   DD  0383                       PUSHL   KA_UNIT_NO
                           04   DD  0389                       PUSHL   #$ER
                                  038B       ;;; TEST 3, SUBTEST 2, ERROR 4
        000100D0 9F   05   FB  038B                       CALLS   #$$M, @#DS$ERRHARD
                                  0392   808          $DS_ABORT
        00010020 9F   6C   FA  0392                       CALLG   (AP), @#DS$ABORT
     07 0000003C'EF   08   E1  0399   809 12$:    BBC     #V_FATAL_ERR,L_FLAGS,13$ ; br = no fatal errors
                                  03A1   810          $DS_ABORT
        00010020 9F   6C   FA  03A1                       CALLG   (AP), @#DS$ABORT
                                  03A8   811 13$:    $SETPRT_S INADR=L_PAGESTRT,PROT=#PRT$C_KW,PRVPRT=L_OLDPROT
             0000002C'EF   DF  03A8                       PUSHAL  L_OLDPROT
             00000000'8F   DD  03AE                       PUSHL   #PRT$C_KW
                        00   DD  03B4                       PUSHL   #0
                        00   DD  03B6                       PUSHL   #0
             00000008'EF   7F  03B8                       PUSHAQ  L_PAGESTRT
        00010430'GF   05   FB  03BE                       CALLS   #5,G^SYS$SETPRT
                                  03C5   812          $DS_BCOMPLETE   14$
                     1E 50   E8  03C5                       BLBS    R0, 14$
                                  03C8   813          $DS_ERRSYS_S    UNIT=KA_UNIT_NO, MSGADR=T_PAGEPROT
                        00   DD  03C8                       PUSHL   #0
             000007E5'EF   DF  03CA                       PUSHAL  T_PAGEPROT
             00000000'EF   DD  03D0                       PUSHL   KA_UNIT_NO
                        05   DD  03D6                       PUSHL   #$ER
                                  03D8       ;;; TEST 3, SUBTEST 2, ERROR 5
        000100C0 9F   04   FB  03D8                       CALLS   #$$M, @#DS$ERRSYS
                                  03DF   814          $DS_ABORT
        00010020 9F   6C   FA  03DF                       CALLG   (AP), @#DS$ABORT
                                  03E6   815 14$:    $DS_MMOFF_S                             ; turn memory management off
        00010158 9F   00   FB  03E6                       CALLS   #0, @#DS$MMOFF
                                  03ED   816          $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                        14   DD  03ED                       PUSHL   #SCB$L_OPCCUS
        00010168 9F   01   FB  03EF                       CALLS   #1, @#DS$CLRVEC
```

```
                                03F6   817          $DS_ENDSUB
                                03F6        T3_S2_X:
     00010038 9F    0000000C'EF    FA  03F6                    CALLG   $S$, @#DS$ENDSUB
                                0401   818          $DS_PAGE
```

```
0401                    .SBTTL  SUBTEST 3.3:  IS SP points to NXM
0401    821 ;+
0401    822 ; Subtest description:
0401    823 ;
0401    824 ;       This subtest checks that an interrupt stack, stack pointer which
0401    825 ;       points to nonexistent memory will halt the processor. Memory
0401    826 ;       management is off for this subtest and an Interrupt Stack not valid
0401    827 ;       HALT should occur (code = 4).
0401    828 ;
0401    829 ; Subtest steps:
0401    830 ;
0401    831 ;       SUBTEST IS points to NXM
0401    832 ;       :  Initialize flags
0401    833 ;       :  Setup code to set done flag at loc 100
0401    834 ;       :  Save current IS SP
0401    835 ;       :  Setup a jump to RENTRY point at next loc
0401    836 ;       :  Get XFC vector to handle on interrupt stack
0401    837 ;       :  Execute XFC instruction
0401    838 ;       :  IF not on the interrupt stack
0401    839 ;       :  :  THEN
0401    840 ;       :  :  Report system error
0401    841 ;       :  :  Abort program
0401    842 ;       :  ENDIF
0401    843 ;       :  Release XFC vector
0401    844 ;       :  Get XFC vector for service on the interrupt stack
0401    845 ;       :  Print instructions to operator
0401    846 ;       :  Point interrupt stack pointer to NXM
0401    847 ;       :  Setup all expected data
0401    848 ;       :  Load GPR's with data
0401    849 ;       :  Set ready to execute flag
0401    850 ;       :  Execute XFC which will write PC and PSL on IS
0401    851 ;       :  (this should cause an interrupt stack not valid HALT)
0401    852 ;       :  IF XFC not executed (will continue and get here)
0401    853 ;       :  :  THEN
0401    854 ;       :  :  Set XFC execution error flag
0401    855 ;       :  ENDIF
0401    856 ;       :  RENTRY point (restart will enter here)
0401    857 ;       :  Read PSL
0401    858 ;       :  IF on IS stack as expected
0401    859 ;       :  :  THEN
0401    860 ;       :  :  Restore IS SP
0401    861 ;       :  ENDIF
0401    862 ;       :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
0401    863 ;       :  IF any errors
0401    864 ;       :  :  THEN
0401    865 ;       :  :  Report errors
0401    866 ;       :  ENDIF
0401    867 ;       :  IF fatal error
0401    868 ;       :  :  THEN
0401    869 ;       :  :  Abort test
0401    870 ;       :  ENDIF
0401    871 ;       :  Release XFC vector
0401    872 ;       ENDSUBTEST IS points to NXM
0401    873 ;
0401    874 ;-
0401    875
0401    876         $DS_BGNSUB
```

```
                                        0401        T3_S3::
00010030 9F   00000018'EF    FA 0401                        CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF    16 040C   877          JSB     CLEAR_BUFFER                ; clear data buffers
              0000003C'EF    D4 0412   878          CLRL    L_FLAGS                     ; initialize flags
                                        0418   879          $DS_MMON_S
         00010150 9F   00    FB 0418                        CALLS   #0, @#DS$MMON
                                        041F   880          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY4,CODE=#1
                       01   DD 041F                        PUSHL   #1
              00000434'EF    DF 0421                        PUSHAL  RENTRY4
                       14   DD 0427                        PUSHL   #SCB$L_OPCCUS
         00010160 9F   03    FB 0429                        CALLS   #3, @#DS$SETVEC
                             FC 0430   881 1$:      XFC                                 ; get onto the IS
                                        0431   882          .ALIGN  LONG
                                        0434   883 RENTRY4:
              00000000'EF    DC 0434   884          MOVPSL  L_ACT_PSL
        1E 00000000'EF    1A E0 043A   885          BBS     #PSL$V_IS,L_ACT_PSL,2$
                                        0442   886          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                       00   DD 0442                        PUSHL   #0
              0000083C'EF    DF 0444                        PUSHAL  T_NOT_ON_IS
              00000000'EF    DD 044A                        PUSHL   KA_UNIT_NO
                       01   DD 0450                        PUSHL   #$ER
                                        0452        ;;; TEST 3, SUBTEST 3, ERROR 1
         000100D0 9F   04    FB 0452                        CALLS   #$$M, @#DS$ERRHARD
                                        0459   887          $DS_ABORT
         00010020 9F   6C    FA 0459                        CALLG   (AP), @#DS$ABORT
                             0460   888 2$:      $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                       14   DD 0460                        PUSHL   #SCB$L_OPCCUS
         00010168 9F   01    FB 0462                        CALLS   #1, @#DS$CLRVEC
                                        0469   889
                                        0469   890          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#1
                       01   DD 0469                        PUSHL   #1
              00000C78'EF    DF 046B                        PUSHAL  XFC_SERVICE
                       14   DD 0471                        PUSHL   #SCB$L_OPCCUS
         00010160 9F   03    FB 0473                        CALLS   #3, @#DS$SETVEC
00000100'EF   00000040'EF    7D 047A   891 3$:      MOVQ    JUMP,RESTART                ; setup code to renter after halt
00000108'EF   00000048'EF    7D 0485   892          MOVQ    JUMP+8,RESTART+8            ; load rest of rentry code
00000004'EF   00000508'EF    DE 0490   893          MOVAL   RENTRY5,RENTRY_POINT        ; load rentry address
         00000020'EF   5E    D0 049B   894          MOVL    SP,L_CUR_ISP                ; read the current IS SP
                             04A2   895 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_04,P1=#6$
              00000500'8F    DD 04A2                        PUSHL   #6$
                       04   DD 04A8                        PUSHL   #CODE_04
              00000085'EF    9F 04AA                        PUSHAB  T_INSTRUCTIONS
         000100F0 9F   03    FB 04B0                        CALLS   #$$N, @#DS$PRINTF
                                        04B7   896          $DS_PRINTF_S    FORMAT=T_CONTINUE
              000000C3'EF    9F 04B7                        PUSHAB  T_CONTINUE
         000100F0 9F   01    FB 04BD                        CALLS   #$$N, @#DS$PRINTF
00000000'EF   FFFFFFF8 8F    D0 04C4   897 5$:      MOVL    #-8,L_EXP_SP                ; load expected SP
00000000'EF   00000000'EF    D0 04CF   898          MOVL    L_KSP,L_EXP_KSP             ; load expected KSP
              00000000'EF    D4 04DA   899          CLRL    L_EXP_ISP                   ; ISP unpredictable if on int. stack
00000000'EF   00000965'EF    D0 04E0   900          MOVL    DATA_3,L_EXP_PSL            ; load expected PSL
              00000BDE'EF    16 04EB   901          JSB     LOAD_GPR                    ; load GPR'S and read AP & FP
         5E   FFFFFFFC 8F    D0 04F1   902          MOVL    #-4,SP                      ; point IS sp to a NXM address
         0000003C'EF   01    C8 04F8   903          BISL2   #M_READY,L_FLAGS            ; set ready to execute flag
                             01 04FF   904          NOP                                 ; wait a little
                             FC 0500   905 6$:      XFC                                 ; execute XFC instruction to cause
                                0501   906                                             ; exception which will push on IS
                                0501   907                                             ; and cause a not valid halt
```

```
                0000003C'EF   04   C8  0501   908          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                      0508   909
                                      0508   910 RENTRY5:                                 ; will renter here when completed
                                      0508   911
            11 00000000'EF    1A    E1  0508   912          BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
               5E   00000020'EF    D0  0510   913          MOVL    L_CUR_ISP,SP            ; reset IS SP
               6E   0000051F'8F    D0  0517   914          MOVL    #7$,(SP)                ; setup return address
                                   02  051E   915          REI                             ; get off interrupt stack
                                   0B  051F   916 7$:      BRB     9$                      ; skip IPR load
     00000000'8F   00000020'EF    DA  0521   917 8$:      MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                                      052C   918
                00000C85'EF    16  052C   919 9$:      JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
            2C 0000003C'EF    0C   E1  0532   920          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                      053A   921          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_NXMIS_ERR
                00000250'8F    DD  053A                        PUSHL   #T_NXMIS_ERR
                00000969'EF    DF  0540                        PUSHAL  ERR_MESSAGE
                           00  DD  0546                        PUSHL   #0
                00000000'EF    DD  0548                        PUSHL   KA_UNIT_NO
                           02  DD  054E                        PUSHL   #$ER
                                      0550       ;;; TEST 3, SUBTEST 3, ERROR 2
                00100D0 9F    05   FB  0550                    CALLS   #$$M, @#DS$ERRHARD
            07 0000003C'EF    08   E1  0557   922          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                      055F   923          $DS_ABORT
               00010020 9F    6C   FA  055F                    CALLG   (AP), @#DS$ABORT
                                      0566   924 10$:     $DS_MMOFF_S                     ; turn memory management off
               00010158 9F    00   FB  0566                    CALLS   #0, @#DS$MMOFF
                                      056D   925          $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                           14  DD  056D                        PUSHL   #SCB$L_OPCCUS
               00010168 9F    01   FB  056F                    CALLS   #1, @#DS$CLRVEC
                                      0576   926          $DS_ENDSUB
                                      0576       T3_S3_X:
     00010038 9F   00000018'EF    FA  0576                    CALLG   $S$, @#DS$ENDSUB
                                      0581   927          $DS_PAGE
```

```
0581                    .SBTTL  SUBTEST 3.4:  CHMU inst. with PSL<IS>=1
0581    930 ;+
0581    931 ; Subtest description:
0581    932 ;
0581    933 ;       This subtest checks that a CHMU instruction with PSL<IS>=1, will cause
0581    934 ;       the processor to HALT (code = A).
0581    935 ;
0581    936 ; Subtest steps:
0581    937 ;
0581    938 ;       SUBTEST CHMU instruction with PSL<IS>=1
0581    939 ;       :  Initialize Flags
0581    940 ;       :  Get XFC vector to handle exception on interrupt stack
0581    941 ;       :  Get CHMU vector in case CHMU instruction does execute
0581    942 ;       :  Setup code to set done flag at loc 100
0581    943 ;       :  Setup code to jump to READ_GPR point in next loc
0581    944 ;       :  Print restart instructions to operator
0581    945 ;       :  Execute an XFC instruction to get on interrupt stack
0581    946 ;       :  Read current PSL
0581    947 ;       :  IF NOT on interrupt stack
0581    948 ;       :  :  THEN
0581    949 ;       :  :  Report wrong stack error
0581    950 ;       :  :  Abort test
0581    951 ;       :  ENDIF
0581    952 ;       :  Setup all expected data
0581    953 ;       :  Load GPR's with data
0581    954 ;       :  Set ready to execute flag
0581    955 ;       :  Execute CHMU to cause processor to HALT
0581    956 ;       :  (this should cause a CHMU on interrupt stack HALT)
0581    957 ;       :  IF CHMU does not caues an exception
0581    958 ;       :  :  THEN
0581    959 ;       :  :  Set execution error flag
0581    960 ;       :  ENDIF
0581    961 ;       :  RENTRY point (restart enters here from exception)
0581    962 ;       :  Go check FLAGS, PSL, KSP, ISP, SP and GPR's for errors
0581    963 ;       :  IF any errors
0581    964 ;       :  :  THEN
0581    965 ;       :  :  Report errors
0581    966 ;       :  ENDIF
0581    967 ;       :  IF fatal error
0581    968 ;       :  :  THEN
0581    969 ;       :  :  Abort test
0581    970 ;       :  ENDIF
0581    971 ;       :  Modify PC on stack to point to second RENTRY point
0581    972 ;       :  Execute REI to get back to correct stack
0581    973 ;       :  Second RENTRY point
0581    974 ;       :  Read PSL
0581    975 ;       :  IF not on kernel stack
0581    976 ;       :  :  THEN
0581    977 ;       :  :  Report incorrect stack error
0581    978 ;       :  :  Abort test
0581    979 ;       :  ENDIF
0581    980 ;       :  Release interval timer vector
0581    981 ;       :  Release CHMU vector
0581    982 ;       :  Release XFC vector
0581    983 ;       ENDSUBTEST CHMU instruction with PSL<IS>=1
0581    984 ;
0581    985 ;-
```

```
                                0581   986
                                0581   987          $DS_BGNSUB
                                0581          T3_S4::
00010030 9F   00000024'EF   FA 0581                 CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16 058C   988          JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4 0592   989          CLRL    L_FLAGS                 ; initialize flags
                            0598   990          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY6,CODE=#1
                         01 DD 0598                 PUSHL   #1
                   000005C0'EF DF 059A              PUSHAL  RENTRY6
                         14 DD 05A0                 PUSHL   #SCB$L_OPCCUS
              00010160 9F   03 FB 05A2              CALLS   #3, @#DS$SETVEC
                            05A9   991 1$:       $DS_SETVEC_S    VECTOR=#SCB$L_CHMU,SRVADR=CHMX_SERVICE
                         00 DD 05A9                 PUSHL   #0
                   00000C5C'EF DF 05AB              PUSHAL  CHMX_SERVICE
                   0000004C 8F DD 05B1              PUSHL   #SCB$L_CHMU
              00010160 9F   03 FB 05B7              CALLS   #3, @#DS$SETVEC
                            05BE   992
                         FC 05BE   993 2$:       XFC                             ; execute XFC instruction to cause
                            05BF   994                                          ; exception which will get on IS
                            05BF   995          .ALIGN  LONG
                            05C0   996 RENTRY6:
                   00000000'EF DC 05C0   997          MOVPSL  L_ACT_PSL               ; read PSL
                1E 00000000'EF 1A E0 05C6   998          BBS     #PSL$V_IS,L_ACT_PSL,3$  ; br = on interrupt stack
                            05CE   999          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                         00 DD 05CE                 PUSHL   #0
                   0000083C'EF DF 05D0              PUSHAL  T_NOT_ON_IS
                   00000000'EF DD 05D6              PUSHL   KA_UNIT_NO
                         01 DD 05DC                 PUSHL   #SER
                            05DE       ;;; TEST 3, SUBTEST 4, ERROR 1
              000100D0 9F   04 FB 05DE              CALLS   #$$M, @#DS$ERRHARD
                            05E5   1000         $DS_ABORT
              00010020 9F   6C FA 05E5                  CALLG   (AP), @#DS$ABORT
   00000100'EF   00000040'EF 7D 05EC   1001 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
   00000108'EF   00000048'EF 7D 05F7   1002         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
   00000004'EF   00000675'EF DE 0602   1003         MOVAL   RENTRY7,RENTRY_POINT    ; load rentry address
                            060D   1004 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0A,P1=#6$
                   0000066C'8F DD 060D              PUSHL   #6$
                         0A DD 0613                 PUSHL   #CODE_0A
                   00000085'EF 9F 0615              PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03 FB 061B              CALLS   #$$N, @#DS$PRINTF
                            0622   1005         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   000000C3'EF 9F 0622              PUSHAB  T_CONTINUE
              000100F0 9F   01 FB 0628              CALLS   #$$N, @#DS$PRINTF
              00000020'EF   5E D0 062F   1006 5$:      MOVL    SP,L_CUR_ISP
00000000'EF   00000000'EF   08 C3 0636   1007         SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
     00000000'EF   00000000'EF D0 0642   1008         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF   00 D0 064D   1009         MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
     00000000'EF   00000961'EF D0 0654   1010         MOVL    DATA_2,L_EXP_PSL        ; load expected PSL
              00000BDE'EF   16 065F   1011         JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF   01 C8 0665   1012         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                         00 BF 066C   1013 6$:      CHMU    #0                      ; execute change mode instruction
              0000003C'EF   04 C8 066E   1014         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                            0675   1015 RENTRY7:                                  ; will renter here when completed
          11 00000000'EF   1A E1 0675   1016         BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
             5E   00000020'EF D0 067D   1017         MOVL    L_CUR_ISP,SP            ; reset IS SP
             6E   0000068C'8F D0 0684   1018         MOVL    #7$,(SP)                ; setup return address
                            02 068B   1019         REI                             ; get off interrupt stack
```

```
                          0B   11  068C  1020 7$:     BRB     9$                              ; skip IPR load
00000000'8F    00000020'EF    DA  068E  1021 8$:     MTPR    L_CUR_ISP,#PR$_ISP              ; load IS SP into IPR
               00000C85'EF    16  0699  1022 9$:     JSB     ERROR_CHECK                     ; check for errors (GPR's,AP,FP,PSL)
        2C 0000003C'EF    0C   E1  069F  1023         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                  06A7  1024         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMU_ERR
               0000028D'8F    DD  06A7                        PUSHL   #T_CHMU_ERR
               00000969'EF    DF  06AD                        PUSHAL  ERR_MESSAGE
                          00   DD  06B3                        PUSHL   #0
               00000000'EF    DD  06B5                        PUSHL   KA_UNIT_NO
                          02   DD  06BB                        PUSHL   #$ER
                                  06BD             ;;; TEST 3, SUBTEST 4, ERROR 2
                               FB  06BD                        CALLS   #$$M, @#DS$ERRHARD
        07 0000003C'EF    08   E1  06C4  1025         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                  06CC  1026         $DS_ABORT
                               FA  06CC                        CALLG   (AP), @#DS$ABORT
                                  06D3  1027 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                          14   DD  06D3                        PUSHL   #SCB$L_OPCCUS
                               FB  06D5                        CALLS   #1, @#DS$CLRVEC
                                  06DC  1028         $DS_CLRVEC_S    VECTOR=#SCB$L_CHMU
               0000004C 8F    DD  06DC                        PUSHL   #SCB$L_CHMU
                               FB  06E2                        CALLS   #1, @#DS$CLRVEC
                                  06E9  1029         $DS_ENDSUB
                                  06E9             T3_S4_X:
00010038 9F    00000024'EF    FA  06E9                        CALLG   $S$, @#DS$ENDSUB
                                  06F4  1030         $DS_PAGE
                                  06F4  1031
```

```
                        .SBTTL  SUBTEST 3.5:  CHMS inst. with PSL<IS>=1
06F4   1034 ;+
06F4   1035 ; Subtest description:
06F4   1036 ;
06F4   1037 ;       This subtest checks that a CHMS instruction with PSL<IS>=1, will cause
06F4   1038 ;       the processor to HALT (code = A).
06F4   1039 ;
06F4   1040 ; Subtest steps:
06F4   1041 ;
06F4   1042 ;       SUBTEST CHMS instruction with PSL<IS>=1
06F4   1043 ;       :  Initialize Flags
06F4   1044 ;       :  Get XFC vector to handle exception on interrupt stack
06F4   1045 ;       :  Get interval timer vector to prevent outside interrupts
06F4   1046 ;       :  Get CHMS vector in case CHMS instruction does execute
06F4   1047 ;       :  Setup code to set done flag at loc 100
06F4   1048 ;       :  Setup code to jump to RENTRY point in next loc
06F4   1049 ;       :  Print restart instructions to operator
06F4   1050 ;       :  Execute an XFC instruction to get on interrupt stack
06F4   1051 ;       :  Read current PSL
06F4   1052 ;       :  IF NOT on interrupt stack
06F4   1053 ;       :  :  THEN
06F4   1054 ;       :  :  Report wrong stack error
06F4   1055 ;       :  :  Exit test
06F4   1056 ;       :  ENDIF
06F4   1057 ;       :  Set ready to execute flag
06F4   1058 ;       :  Execute CHMS to cause processor to HALT
06F4   1059 ;       :  (this should cause a CHMS on interrupt stack HALT)
06F4   1060 ;       :  IF CHMS does not caues an exception
06F4   1061 ;       :  :  THEN
06F4   1062 ;       :  :  Set exception error flag
06F4   1063 ;       :  ENDIF
06F4   1064 ;       :  RENTRY point (restart enters here from exception)
06F4   1065 ;       :  IF exception error flag set
06F4   1066 ;       :  :  THEN
06F4   1067 ;       :  :  Report CHMS exception error
06F4   1068 ;       :  :  Exit test
06F4   1069 ;       :  ENDIF
06F4   1070 ;       :  IF CHMS instruction executed flag set
06F4   1071 ;       :  :  THEN
06F4   1072 ;       :  :  Report change mode error
06F4   1073 ;       :  :  Exit test
06F4   1074 ;       :  ENDIF
06F4   1075 ;       :  IF done flag not set
06F4   1076 ;       :  :  THEN
06F4   1077 ;       :  :  Report done error
06F4   1078 ;       :  :  Exit test
06F4   1079 ;       :  ENDIF
06F4   1080 ;       :  Mcdify PC on stack to point to RENTRY2
06F4   1081 ;       :  Execute REI to get back to correct stack
06F4   1082 ;       :  RENTRY2 point
06F4   1083 ;       :  Read PSL
06F4   1084 ;       :  IF IS not clear
06F4   1085 ;       :  :  THEN
06F4   1086 ;       :  :  Report incorrect stack error in clean up
06F4   1087 ;       :  :  Exit test
06F4   1088 ;       :  ENDIF
06F4   1089 ;       :  Release interval timer vector
```

```
                                    06F4  1090 ;       :  Release CHMS vector
                                    06F4  1091 ;       :  Release XFC vector
                                    06F4  1092 ;       ENDSUBTEST CHMS instruction with PSL<IS>=1
                                    06F4  1093 ;
                                    06F4  1094 ;-
                                    06F4  1095         $DS_BGNSUB
                                    06F4       T3_S5::
    00010030 9F   00000030'EF   FA  06F4                       CALLG   $S$, @#DS$BGNSUB
                  00000BC6'EF   16  06FF  1096         JSB     CLEAR_BUFFER            ; clear data buffers
                  0000003C'EF   D4  0705  1097         CLRL    L_FLAGS                 ; initialize flags
                                    070B  1098         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY8,CODE=#1
                              01    DD  070B                       PUSHL   #1
                  00000734'EF       DF  070D                       PUSHAL  RENTRY8
                              14    DD  0713                       PUSHL   #SCB$L_OPCCUS
              00010160 9F   03    FB  0715                       CALLS   #3, @#DS$SETVEC
                                    071C  1099 1$:       $DS_SETVEC_S    VECTOR=#SCB$L_CHMS,SRVADR=CHMX_SERVICE
                              00    DD  071C                       PUSHL   #0
                  00000C5C'EF       DF  071E                       PUSHAL  CHMX_SERVICE
                  00000048 8F       DD  0724                       PUSHL   #SCB$L_CHMS
              00010160 9F   03    FB  072A                       CALLS   #3, @#DS$SETVEC
                                    0731  1100
                                    0731  1101 2$:       XFC                             ; execute XFC instruction to cause
                                    0732  1102                                         ; exception which will get on IS
                                    0732  1103         .ALIGN  LONG
                                    0734  1104 RENTRY8:
                  00000000'EF       DC  0734  1105         MOVPSL  L_ACT_PSL               ; read PSL
            1E 00000000'EF      1A    E0  073A  1106         BBS     #PSL$V_IS,L_ACT_PSL,3$  ; br = on interrupt stack
                                    0742  1107         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                              00    DD  0742                       PUSHL   #0
                  0000083C'EF       DF  0744                       PUSHAL  T_NOT_ON_IS
                  00000000'EF       DD  074A                       PUSHL   KA_UNIT_NO
                              01    DD  0750                       PUSHL   #$ER
                                    0752       ;;; TEST 3, SUBTEST 5, ERROR 1
              000100D0 9F   04    FB  0752                       CALLS   #$$M, @#DS$ERRHARD
                                    0759  1108         $DS_ABORT
              00010020 9F   6C    FA  0759                       CALLG   (AP), @#DS$ABORT
    00000100'EF   00000040'EF   7D  0760  1109 3$:      MOVQ    JUMP,RESTART            ; setup code to renter after halt
    00000108'EF   00000048'EF   7D  076B  1110         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
    00000004'EF   000007E9'EF   DE  0776  1111         MOVAL   RENTRY9,RENTRY_POINT    ; load rentry address
                                    0781  1112 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0A,P1=#6$
                  000007E0'8F       DD  0781                       PUSHL   #6$
                              0A    DD  0787                       PUSHL   #CODE_0A
                  00000085'EF       9F  0789                       PUSHAB  T_INSTRUCTIONS
              000100F0 9F   03    FB  078F                       CALLS   #$$N, @#DS$PRINTF
                                    0796  1113         $DS_PRINTF_S    FORMAT=T_CONTINUE
                  000000C3'EF       9F  0796                       PUSHAB  T_CONTINUE
              000100F0 9F   01    FB  079C                       CALLS   #$$N, @#DS$PRINTF
              00000020'EF   5E    D0  07A3  1114 5$:      MOVL    SP,L_CUR_ISP
00000000'EF   000000C0'EF   08    C3  07AA  1115         SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
    00000000'EF   00000000'EF   D0  07B6  1116         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF   00    D0  07C1  1117         MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
    00000000'EF   00000961'EF   D0  07C8  1118         MOVL    DATA_2,L_EXP_PSL        ; load expected PSL
                  00000BDE'EF   16  07D3  1119         JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF   01    C8  07D9  1120         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                              00    BE  07E0  1121 6$:      CHMS    #0                      ; execute change mode instruction
              0000003C'EF   04    C8  07E2  1122         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                    07E9  1123 RENTRY9:                                 ; will renter here when completed
```

```
         11 00000000'EF    1A   E1  07E9       BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
            5E    00000020'EF   D0  07F1       MOVL    L_CUR_ISP,SP            ; reset IS SP
            6E    00000800'8F   D0  07F8       MOVL    #7$,(SP)                ; setup return address
                          02   07FF            REI                             ; get off interrupt stack
                     0B   11   0800  7$:       BRB     9$                      ; skip IPR load
   00000000'8F    00000020'EF   DA  0802  8$:       MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                  00000C85'EF   16  080D  9$:       JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
         2C 0000003C'EF    0C   E1  0813       BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                0818       $DS_ERRHARD_S    UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMS_ERR
                  000002B5'8F   DD  081B                PUSHL   #T_CHMS_ERR
                  00000969'EF   DF  0821                PUSHAL  ERR_MESSAGE
                           00   DD  0827                PUSHL   #0
                  00000000'EF   DD  0829                PUSHL   KA_UNIT_NO
                           02   DD  082F                PUSHL   #$ER
                                0831   ;;; TEST 3, SUBTEST 5, ERROR 2
            000100D0 9F    05   FB  0831                CALLS   #$$M, @#DS$ERRHARD
         07 0000003C'EF    08   E1  0838       BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                0840       $DS_ABORT
            00010020 9F    6C   FA  0840                CALLG   (AP), @#DS$ABORT
                                0847  10$:     $DS_CLRVEC_S     VECTOR=#SCB$L_OPCCUS
                           14   DD  0847                PUSHL   #SCB$L_OPCCUS
            00010168 9F    01   FB  0849                CALLS   #1, @#DS$CLRVEC
                                0850       $DS_CLRVEC_S     VECTOR=#SCB$L_CHMS
                  00000048 8F   DD  0850                PUSHL   #SCB$L_CHMS
            00010168 9F    01   FB  0856                CALLS   #1, @#DS$CLRVEC
                                085D       $DS_ENDSUB
                                085D   T3_S5_X:
   00010038 9F    00000030'EF   FA  085D                CALLG   $S$, @#DS$ENDSUB
                                0868       $DS_PAGE
```

```
                        .SBTTL  SUBTEST 3.6:  CHME inst. with PSL<IS>=1

                ;+
                ; Subtest description:
                ;
                ;       This subtest checks that a CHME instruction with PSL<IS>=1, will cause
                ;       the processor to HALT (code = A).
                ;
                ; Subtest steps:
                ;
                ;       SUBTEST CHME instruction with PSL<IS>=1
                ;       :  Initialize Flags
                ;       :  Get XFC vector to handle exception on interrupt stack
                ;       :  Get interval timer vector to prevent outside interrupts
                ;       :  Get CHME vector in case CHME instruction does execute
                ;       :  Setup code to set done flag at loc 100
                ;       :  Setup code to jump to RENTRY point in next loc
                ;       :  Print restart instructions to operator
                ;       :  Execute an XFC instruction to get on interrupt stack
                ;       :  Read current PSL
                ;       :  IF NOT on interrupt stack
                ;       :  :  THEN
                ;       :  :  Report wrong stack error
                ;       :  :  Exit test
                ;       :  ENDIF
                ;       :  Set ready to execute flag
                ;       :  Execute CHME to cause processor to HALT
                ;       :  (this should cause a CHME on interrupt stack HALT)
                ;       :  IF CHME does not caues an exception
                ;       :  :  THEN
                ;       :  :  Set exception error flag
                ;       :  ENDIF
                ;       :  RENTRY point (restart enters here from exception)
                ;       :  IF exception error flag set
                ;       :  :  THEN
                ;       :  :  Report CHME exception error
                ;       :  :  Exit test
                ;       :  ENDIF
                ;       :  IF CHME instruction executed flag set
                ;       :  :  THEN
                ;       :  :  Report change mode error
                ;       :  :  Exit test
                ;       :  ENDIF
                ;       :  IF done flag not set
                ;       :  :  THEN
                ;       :  :  Report done error
                ;       :  :  Exit test
                ;       :  ENDIF
                ;       :  Mcdify PC on stack to point to RENTRY2
                ;       :  Execute REI to get back to correct stack
                ;       :  RENTRY2 point
                ;       :  Read PSL
                ;       :  IF IS not clear
                ;       :  :  THEN
                ;       :  :  Report incorrect stack error in clean up
                ;       :  :  Exit test
                ;       :  ENDIF
                ;       :  Release interval timer vector
```

```
                                0868  1197 ;       :  Release CHME vector
                                0868  1198 ;       :  Release XFC vector
                                0868  1199 ;       ENDSUBTEST CHME instruction with PSL<IS>=1
                                0868  1200 ;
                                0868  1201 ;-
                                0868  1202         $DS_BGNSUB
                                0868        T3_S6::
00010030 9F   0000003C'EF   FA  0868                CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  0873  1203          JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  0879  1204          CLRL    L_FLAGS                 ; initialize flags
                            087F  1205          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY10,CODE=#1
                        01  DD  087F                PUSHL   #1
              000008A8'EF   DF  0881                PUSHAL  RENTRY10
                        14  DD  0887                PUSHL   #SCB$L_OPCCUS
         00010160 9F    03  FB  0889                CALLS   #3, @#DS$SETVEC
                            0890  1206 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_CHME,SRVADR=CHMX_SERVICE
                        00  DD  0890                PUSHL   #0
              00000C5C'EF   DF  0892                PUSHAL  CHMX_SERVICE
              00000044 8F   DD  0898                PUSHL   #SCB$L_CHME
         00010160 9F    03  FB  089E                CALLS   #3, @#DS$SETVEC
                            08A5  1207
                        FC  08A5  1208 2$:      XFC                             ; execute XFC instruction to cause
                            08A6  1209                                          ; exception which will get on IS
                            08A6  1210          .ALIGN  LONG
                            08A8  1211 RENTRY10:
              00000000'EF   DC  08A8  1212          MOVPSL  L_ACT_PSL               ; read PSL
        1E 00000000'EF      1A  E0  08AE  1213          BBS     #PSL$V_IS,L_ACT_PSL,3$  ; br = on interrupt stack
                            08B6  1214          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                        00  DD  08B6                PUSHL   #0
              0000083C'EF   DF  08B8                PUSHAL  T_NOT_ON_IS
              00000000'EF   DD  08BE                PUSHL   KA_UNIT_NO
                        01  DD  08C4                PUSHL   #$ER
                            08C6        ;;; TEST 3, SUBTEST 6, ERROR 1
         000100D0 9F    04  FB  08C6                CALLS   #$$M, @#DS$ERRHARD
                            08CD  1215          $DS_ABORT
         00010020 9F    6C  FA  08CD                CALLG   (AP), @#DS$ABORT
   00000100'EF   00000040'EF  7D  08D4  1216 3$:      MOVQ    JUMP,RESTART            ; setup code to renter after halt
   00000108'EF   00000048'EF  7D  08DF  1217          MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
   00000004'EF   0000095D'EF  DE  08EA  1218          MOVAL   RENTRY11,RENTRY_POINT   ; load rentry address
                            08F5  1219 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0A,P1=#6$
              00000954'8F   DD  08F5                PUSHL   #6$
                        0A  DD  08FB                PUSHL   #CODE_0A
              00000085'EF   9F  08FD                PUSHAB  T_INSTRUCTIONS
         000100F0 9F    03  FB  0903                CALLS   #$$N, @#DS$PRINTF
                            090A  1220          $DS_PRINTF_S    FORMAT=T_CONTINUE
              000000C3'EF   9F  090A                PUSHAB  T_CONTINUE
         000100F0 9F    01  FB  0910                CALLS   #$$N, @#DS$PRINTF
              00000020'EF   5E  D0  0917  1221 5$:      MOVL    SP,L_CUR_ISP
00000000'EF   00000000'EF   08  C3  091E  1222          SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
   00000000'EF   00000000'EF  D0  092A  1223          MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
              00000000'EF   00  D0  0935  1224          MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
   00000000'EF   00000961'EF  D0  093C  1225          MOVL    DATA_2,L_EXP_PSL        ; load expected PSL
              00000BDE'EF   16  0947  1226          JSB     LOAD_GPR                ; load GPR's with data
              0000003C'EF   01  C8  094D  1227          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                        00  BD  0954  1228 6$:      CHME    #0                      ; execute change mode instruction
              0000003C'EF   04  C8  0956  1229          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                            095D  1230 RENTRY11:                                    ; will renter here when completed
```

```
          11 00000000'EF   1A   E1  095D  1231          BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
             5E   00000020'EF   D0  0965  1232          MOVL    L_CUR_ISP,SP            ; reset IS SP
             6E   00000974'8F   D0  096C  1233          MOVL    #7$,(SP)                ; setup return address
                                02  0973  1234          REI                             ; get off interrupt stack
                           0B   11  0974  1235 7$:      BRB     9$                      ; skip IPR load
    00000000'8F   00000020'EF   DA  0976  1236 8$:      MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                  00000C85'EF   16  0981  1237 9$:      JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
          2C 0000003C'EF   0C   E1  0987  1238          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                098F  1239          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHME_ERR
                    000002DD'8F   DD  098F                      PUSHL   #T_CHME_ERR
                    00000969'EF   DF  0995                      PUSHAL  ERR_MESSAGE
                           00   DD  099B                      PUSHL   #0
                    00000000'EF   DD  099D                      PUSHL   KA_UNIT_NO
                           02   DD  09A3                      PUSHL   #$ER
                                09A5          ;;; TEST 3, SUBTEST 6, ERROR 2
             000100D0 9F   05   FB  09A5                      CALLS   #$$M, @#DS$ERRHARD
          07 0000003C'EF   08   E1  09AC  1240          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                09B4  1241          $DS_ABORT
             00010020 9F   6C   FA  09B4                      CALLG   (AP), @#DS$ABORT
                                09BB  1242 10$:      $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                           14   DD  09BB                      PUSHL   #SCB$L_OPCCUS
             00010168 9F   01   FB  09BD                      CALLS   #1, @#DS$CLRVEC
                                09C4  1243          $DS_CLRVEC_S    VECTOR=#SCB$L_CHME
                  00000044 8F   DD  09C4                      PUSHL   #SCB$L_CHME
             00010168 9F   01   FB  09CA                      CALLS   #1, @#DS$CLRVEC
                                09D1  1244          $DS_ENDSUB
                                09D1          T3_S6_X:
    00010038 9F   0000003C'EF   FA  09D1                      CALLG   $S$, @#DS$ENDSUB
                                09DC  1245          $DS_PAGE
```

```
09DC                    .SBTTL  SUBTEST 3.7:  CHMK inst. with PSL<IS>=1
09DC  1248 ;+
09DC  1249 ; Subtest description:
09DC  1250 ;
09DC  1251 ;       This subtest checks that a CHMK instruction with PSL<IS>=1, will cause
09DC  1252 ;       the processor to HALT (code = A).
09DC  1253 ;
09DC  1254 ; Subtest steps:
09DC  1255 ;
09DC  1256 ;       SUBTEST CHMK instruction with PSL<IS>=1
09DC  1257 ;       :  Initialize Flags
09DC  1258 ;       :  Get XFC vector to handle exception on interrupt stack
09DC  1259 ;       :  Get interval timer vector to prevent outside interrupts
09DC  1260 ;       :  Get CHMK vector in case CHMK instruction does execute
09DC  1261 ;       :  Setup code to set done flag at loc 100
09DC  1262 ;       :  Setup code to jump to RENTRY point in next loc
09DC  1263 ;       :  Print restart instructions to operator
09DC  1264 ;       :  Execute an XFC instruction to get on interrupt stack
09DC  1265 ;       :  Read current PSL
09DC  1266 ;       :  IF NOT on interrupt stack
09DC  1267 ;       :  :  THEN
09DC  1268 ;       :  :  Report wrong stack error
09DC  1269 ;       :  :  Exit test
09DC  1270 ;       :  ENDIF
09DC  1271 ;       :  Set ready to execute flag
09DC  1272 ;       :  Execute CHMK to cause processor to HALT
09DC  1273 ;       :  (this should cause a CHMK on interrupt stack HALT)
09DC  1274 ;       :  IF CHMK does not caues an exception
09DC  1275 ;       :  :  THEN
09DC  1276 ;       :  :  Set exception error flag
09DC  1277 ;       :  ENDIF
09DC  1278 ;       :  RENTRY point (restart enters here from exception)
09DC  1279 ;       :  IF exception error flag set
09DC  1280 ;       :  :  THEN
09DC  1281 ;       :  :  Report CHMK exception error
09DC  1282 ;       :  :  Exit test
09DC  1283 ;       :  ENDIF
09DC  1284 ;       :  IF CHMK instruction executed flag set
09DC  1285 ;       :  :  THEN
09DC  1286 ;       :  :  Report change mode error
09DC  1287 ;       :  :  Exit test
09DC  1288 ;       :  ENDIF
09DC  1289 ;       :  IF done flag not set
09DC  1290 ;       :  :  THEN
09DC  1291 ;       :  :  Report done error
09DC  1292 ;       :  :  Exit test
09DC  1293 ;       :  ENDIF
09DC  1294 ;       :  Modify PC on stack to point to RENTRY2
09DC  1295 ;       :  Execute REI to get back to correct stack
09DC  1296 ;       :  RENTRY2 point
09DC  1297 ;       :  Read PSL
09DC  1298 ;       :  IF IS not clear
09DC  1299 ;       :  :  THEN
09DC  1300 ;       :  :  Report incorrect stack error in clean up
09DC  1301 ;       :  :  Exit test
09DC  1302 ;       :  ENDIF
09DC  1303 ;       :  Release interval timer vector
```

```
                                09DC  1304 ;       :  Release CHMK vector
                                09DC  1305 ;       :  Release XFC vector
                                09DC  1306 ;       ENDSUBTEST CHMK instruction with PSL<IS>=1
                                09DC  1307 ;
                                09DC  1308 ;-
                                09DC  1309         $DS_BGNSUB
                                09DC       T3_S7::
00010030 9F   00000048'EF   FA  09DC                       CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  09E7  1310         JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  09ED  1311         CLRL    L_FLAGS                 ; initialize flags
                                09F3  1312         $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=RENTRY12,CODE=#1
                          01    DD  09F3                       PUSHL   #1
              00000A1C'EF       DF  09F5                       PUSHAL  RENTRY12
                          14    DD  09FB                       PUSHL   #SCB$L_OPCCUS
           00010160 9F    03    FB  09FD                       CALLS   #3, @#DS$SETVEC
                                0A04  1313 1$:       $DS_SETVEC_S    VECTOR=#SCB$L_CHMK,SRVADR=CHMX_SERVICE
                          00    DD  0A04                       PUSHL   #0
              00000C5C'EF       DF  0A06                       PUSHAL  CHMX_SERVICE
              00000040 8F       DD  0A0C                       PUSHL   #SCB$L_CHMK
           00010160 9F    03    FB  0A12                       CALLS   #3, @#DS$SETVEC
                                0A19  1314
                             FC 0A19  1315 2$:       XFC                             ; execute XFC instruction to cause
                                0A1A  1316                                           ; exception which will get on IS
                                0A1A  1317         .ALIGN  LONG
                                0A1C  1318 RENTRY12:
              00000000'EF       DC  0A1C  1319         MOVPSL  L_ACT_PSL               ; read PSL
        1E 00000000'EF    1A    E0  0A22  1320         BBS     #PSL$V_IS,L_ACT_PSL,3$  ; br = on interrupt stack
                                0A2A  1321         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, MSGADR=T_NOT_ON_IS
                          00    DD  0A2A                       PUSHL   #0
              0000083C'EF       DF  0A2C                       PUSHAL  T_NOT_ON_IS
              00000000'EF       DD  0A32                       PUSHL   KA_UNIT_NO
                          01    DD  0A38                       PUSHL   #$ER
                                0A3A       ;;; TEST 3, SUBTEST 7, ERROR 1
           000100D0 9F    04    FB  0A3A                       CALLS   #$$M, @#DS$ERRHARD
                                0A41  1322         $DS_ABORT
           00010020 9F    6C    FA  0A41                       CALLG   (AP), @#DS$ABORT
     00000100'EF   00000040'EF   7D  0A48  1323 3$:       MOVQ    JUMP,RESTART            ; setup code to renter after halt
     00000108'EF   00000048'EF   7D  0A53  1324           MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
     00000004'EF   00000AD1'EF   DE  0A5E  1325           MOVAL   RENTRY13,RENTRY_POINT   ; load rentry address
                                0A69  1326 4$:       $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0A,P1=#6$
              00000AC8'8F       DD  0A69                       PUSHL   #6$
                          0A    DD  0A6F                       PUSHL   #CODE_0A
              00000085'EF       9F  0A71                       PUSHAB  T_INSTRUCTIONS
           000100F0 9F    03    FB  0A77                       CALLS   #$$N, @#DS$PRINTF
                                0A7E  1327         $DS_PRINTF_S    FORMAT=T_CONTINUE
              000000C3'EF       9F  0A7E                       PUSHAB  T_CONTINUE
           000100F0 9F    01    FB  0A84                       CALLS   #$$N, @#DS$PRINTF
           00000020'EF    5E    D0  0A8B  1328 5$:       MOVL    SP,L_CUR_ISP
00000000'EF   00000000'EF    08    C3  0A92  1329           SUBL3   #8,L_ISP,L_EXP_SP       ; load expected SP
     00000000'EF   00000000'EF    D0  0A9E  1330           MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
           00000000'EF    00    D0  0AA9  1331           MOVL    #0,L_EXP_ISP            ; ISP unpredictable if on int. stack
     00000000'EF   00000961'EF   D0  0AB0  1332           MOVL    DATA_2,L_EXP_PSL        ; load expected PSL
              00000BDE'EF       16  0ABB  1333           JSB     LOAD_GPR                ; load GPR's with data
           0000003C'EF    01    C8  0AC1  1334           BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                          00    BC  0AC8  1335 6$:       CHMK    #0                      ; execute change mode instruction
           0000003C'EF    04    C8  0ACA  1336           BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                0AD1  1337 RENTRY13:                                 ; will renter here when completed
```

```
                                                0AD1  1338          BBC     #PSL$V_IS,L_ACT_PSL,8$  ; br = not on IS as expected
                                                0AD9  1339          MOVL    L_CUR_ISP,SP            ; reset IS SP
                                                0AE0  1340          MOVL    #7$,(SP)                ; setup return address
                                                0AE7  1341          REI                             ; get off interrupt stack
                                                0AE8  1342 7$:      BRB     9$                      ; skip IPR load
                                                0AEA  1343 8$:      MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                                                0AF5  1344 9$:      JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
                                                0AFB  1345          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                                0B03  1346          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMK_ERR
                                                0B03                PUSHL   #T_CHMK_ERR
                                                0B09                PUSHAL  ERR_MESSAGE
                                                0B0F                PUSHL   #0
                                                0B11                PUSHL   KA_UNIT_NO
                                                0B17                PUSHL   #$ER
                                                0B19            ;;; TEST 3, SUBTEST 7, ERROR 2
                                                0B19                CALLS   #$$M, @#DS$ERRHARD
                                                0B20  1347          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                                0B28  1348          $DS_ABORT
                                                0B28                CALLG   (AP), @#DS$ABORT
                                                0B2F  1349 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                                                0B2F                PUSHL   #SCB$L_OPCCUS
                                                0B31                CALLS   #1, @#DS$CLRVEC
                                                0B38  1350          $DS_CLRVEC_S    VECTOR=#SCB$L_CHMK
                                                0B38                PUSHL   #SCB$L_CHMK
                                                0B3E                CALLS   #1, @#DS$CLRVEC
                                                0B45  1351          $DS_ENDSUB
                                                0B45            T3_S7_X:
                                                0B45                CALLG   $S$, @#DS$ENDSUB
                                                0B50  1352          $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                                                0B50                PUSHL   #SCB$L_TIMER
                                                0B56                CALLS   #1, @#DS$CLRVEC
                                                0B5D  1353          $DS_ENDTEST
                                                0B5D                MOVL    #1, R0                  ; NORMAL EXIT
                                                0B60            TEST_003_X::
                                                0B60                CALLG   (SP), @#DS$BREAK
                                                0B67                RET                             ; RETURN TO TEST SEQUENCER
                                                0B68  1354          $DS_PAGE
                                                0B68  1355
```

```
                     0B68                       .SBTTL  TEST 4:  PROCESSOR HALTS ARCH. UNDEFINED
                  00000000                      .PSECT  TEST_004, PAGE, NOWRT
                      0024
                      0024  1358 ;++
                      0024  1359 ; Test description:
                      0024  1360 ;
                      0024  1361 ;       This test checks the various ways the VAX-11/730 processor can
                      0024  1362 ;       be halted in an architecturally undefined manner. This includes
                      0024  1363 ;       change mode instructions with the vector specifying service on
                      0024  1364 ;       the interrupt stack, invalid vector halt, SCBB read error halt,
                      0024  1365 ;       double machine check halt, and transfer to NXM user WCS halt.
                      0024  1366 ;
                      0024  1367 ;       INSTRUCTIONS FOR EARLY ABORT:  If it is desired to exit the test
                      0024  1368 ;       before completion, one must wait for the console prompt ( >>> )
                      0024  1369 ;       after the halt occurs and type in the sequence D PC adr <CR>
                      0024  1370 ;       C <CR>, where adr is the absolute address of the label
                      0024  1371 ;       TEST_004_X:: in the program listing.
                      0024  1372 ;
                      0024  1373 ;--
                      0024  1374
                      0024  1375
                      0024  1376          $DS_BGNTEST     <MANUAL,HALT,ALL>
                      0024          DATA_004:
                  00000000  0024                  .LONG   0                       ; TEST ARGUMENT TABLE TERMINATOR
                      0028          TEST_004::
                      0000  0028                  .WORD   ^M<>                    ; ENTRY MASK
                      002A  1377
      00000000'EF     D5  002A  1378          TSTL    KA_PTABLE                   ; CPU selected?
                  10      18  0030  1379          BGEQ    5$                      ; br = CPU selected
                      0032  1380          $DS_PRINTF_S    FORMAT=T_NO_KA730 ; print exit warning
      0000004D'EF     9F  0032                  PUSHAB  T_NO_KA730
 000100F0 9F      01     FB  0038                  CALLS   #^$N, @#DS$PRINTF
                      003F  1381          $DS_EXIT        TEST                    ; exit test
                  099E    31  003F                  BRW     TEST_004_X      ; EXIT TEST 4
                      0042  1382 ;
                      0042  1383 ; Ask operator if Auto Restart Switch is in off position, continue if YES, exit
                      0042  1384 ; test if NO.
                      0042  1385 ;
                      0042  1386 5$:      $DS_PRINTF_S    FORMAT=T_WARNING        ; print warning
      00000129'EF     9F  0042                  PUSHAB  T_WARNING
 000100F0 9F      01     FB  0048                  CALLS   #^$N, @#DS$PRINTF
                      004F  1387
                      004F  1388          $DS_ASKLGCL_S   MSGADR=T_REST_OFF,-      ; ask operator
                      004F  1389                          DATADR=B_ANSWER,-       ; location of answer
                      004F  1390                          YEXFER=7$,-             ; br to 7$ if YES
                      004F  1391                          DEFALT=PAR$_NO          ; default is NO
                  00      DD  004F                  PUSHL   #0
      00000000'EF     DD  0051                  PUSHL   PAR$_NO
                  00      DD  0057                  PUSHL   #0
      00000084'EF     DF  0059                  PUSHAL  7$
                  00      DD  005F                  PUSHL   #0
      00000000'EF     DF  0061                  PUSHAL  B_ANSWER
      00000191'EF     DF  0067                  PUSHAL  T_REST_OFF
 00010098 9F      07     FB  006D                  CALLS   #7, @#DS$ASKLGCL
                      0074  1392
                      0074  1393          $DS_PRINTF_S    FORMAT=T_EXITING        ; print warning
      00000114'EF     9F  0074                  PUSHAB  T_EXITING
```

```
                 000100F0 9F    01    FB  007A                          CALLS   #$$N, a#DS$PRINTF
                                      0081  1394          $DS_EXIT        TEST                    ; exit test if NO
                                095C  31    0081                        BRW     TEST_004_X      ; EXIT TEST 4
                                      0084  1395
                 00000000'EF    5E    D0  0084  1396 7$:     MOVL    SP,L_KSP                ; save initial KSP
     00000000'EF 00000000'8F    DB  008B  1397              MFPR    #PR$_ISP,L_ISP          ; save initial ISP
                      00000BC6'EF  16  0096  1398              JSB     CLEAR_BUFFER            ; clear buffer area
                                      009C  1399              $DS_SETVEC_S    VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                                   00 DD  009C                          PUSHL   #0
                           00000C54'EF  DF  009E                        PUSHAL  TIMER_SERVICE
                           000000C0 8F  DD  00A4                        PUSHL   #SCB$L_TIMER
                 00010160 9F    03    FB  00AA                          CALLS   #3, a#DS$SETVEC
                                      00B1  1400 10$:
                                      00B1  1401              $DS_PAGE
```

```
                        00B1                    .SBTTL  SUBTEST 4.1:  CHMU inst. with vector service on IS
                        00B1  1404 ;+
                        00B1  1405 ; Subtest description:
                        00B1  1406 ;
                        00B1  1407 ;       This subtest checks that a CHMU instruction with the SCB vector entry
                        00B1  1408 ;       <1:0>=1 specifying service on the interrupt stack will cause
                        00B1  1409 ;       the processor to HALT (code = B).
                        00B1  1410 ;
                        00B1  1411 ; Subtest steps:
                        00B1  1412 ;
                        00B1  1413 ;       SUBTEST CHMU instruction with vector service on IS
                        00B1  1414 ;       :  Initialize Flags
                        00B1  1415 ;       :  Get interval timer vector to prevent outside interrupts
                        00B1  1416 ;       :  Get CHMU vector specifying service on IS
                        00B1  1417 ;       :  Setup code to set done flag at loc 100
                        00B1  1418 ;       :  Setup code to jump to RENTRY point in next loc
                        00B1  1419 ;       :  Print restart instructions to operator
                        00B1  1420 ;       :  Set ready to execute flag
                        00B1  1421 ;       :  Execute CHMU to cause processor to HALT
                        00B1  1422 ;       :  (this should cause a CHMU service on interrupt stack HALT)
                        00B1  1423 ;       :  IF CHMU does not caues an exception
                        00B1  1424 ;       :  :  THEN
                        00B1  1425 ;       :  :  Set exception error flag
                        00B1  1426 ;       :  ENDIF
                        00B1  1427 ;       :  RENTRY point (restart enters here from exception)
                        00B1  1428 ;       :  IF exception error flag set
                        00B1  1429 ;       :  :  THEN
                        00B1  1430 ;       :  :  Report CHMU exception error
                        00B1  1431 ;       :  :  Exit test
                        00B1  1432 ;       :  ENDIF
                        00B1  1433 ;       :  IF CHMU instruction executed flag set
                        00B1  1434 ;       :  :  THEN
                        00B1  1435 ;       :  :  Report change mode error
                        00B1  1436 ;       :  :  Exit test
                        00B1  1437 ;       :  ENDIF
                        00B1  1438 ;       :  IF done flag not set
                        00B1  1439 ;       :  :  THEN
                        00B1  1440 ;       :  :  Report done error
                        00B1  1441 ;       :  :  Exit test
                        00B1  1442 ;       :  ENDIF
                        00B1  1443 ;       :  Read PSL
                        00B1  1444 ;       :  IF IS not clear
                        00B1  1445 ;       :  :  THEN
                        00B1  1446 ;       :  :  Report incorrect stack error in clean up
                        00B1  1447 ;       :  :  Exit test
                        00B1  1448 ;       :  ENDIF
                        00B1  1449 ;       :  Release interval timer vector
                        00B1  1450 ;       :  Release CHMU vector
                        00B1  1451 ;       ENDSUBTEST CHMU instruction with
                        00B1  1452 ;
                        00B1  1453 ;-
                        00B1  1454          $DS_BGNSUB
                        00B1          T4_S1::
00010030 9F  00000054'EF  FA  00B1                  CALLG   $S$, @#DS$BGNSUB
             00000BC6'EF  16  00BC  1455          JSB     CLEAR_BUFFER                ; clear data buffers
             0000003C'EF  D4  00C2  1456          CLRL    L_FLAGS                     ; initialize flags
                              00C8  1457 1$:      $DS_SETVEC_S    VECTOR=#SCB$L_CHMU,SRVADR=CHMX_SERVICE,CODE=#1
```

```
                       01   DD  00C8                        PUSHL   #1
             00000C5C'EF    DF  00CA                        PUSHAL  CHMX_SERVICE
                 0000004C 8F  DD  00D0                        PUSHL   #SCB$L_CHMU
           00010160 9F    03   FB  00D6                        CALLS   #3, @#DS$SETVEC
                                  00DD  1458
    00000100'EF    00000040'EF   7D  00DD  1459 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
    00000108'EF    00000048'EF   7D  00E8  1460         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
    00000004'EF    0000016D'EF   DE  00F3  1461         MOVAL   RENTRY15,RENTRY_POINT   ; load rentry address
                                  00FE  1462 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0B,P1=#6$
                   00000164'8F   DD  00FE                        PUSHL   #6$
                           0B    DD  0104                        PUSHL   #CODE_0B
                   00000085'EF   9F  0106                        PUSHAB  T_INSTRUCTIONS
           000100F0 9F    03   FB  010C                        CALLS   #$$N, @#DS$PRINTF
                                  0113  1463         $DS_PRINTF_S    FORMAT=T_CONTINUE
                   000000C3'EF   9F  0113                        PUSHAB  T_CONTINUE
           000100F0 9F    01   FB  0119                        CALLS   #$$N, @#DS$PRINTF
    00000020'EF    00000000'EF   D0  0120  1464 5$:     MOVL    L_ISP,L_CUR_ISP         ; save current IS SP
    00000000'EF    00000000'EF   D0  012B  1465         MOVL    L_KSP,L_EXP_SP          ; load expected SP
    00000000'EF    00000000'EF   D0  0136  1466         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
    00000000'EF    00000000'EF   D0  0141  1467         MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
    00000000'EF    0000095D'EF   D0  014C  1468         MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                   00000BDE'EF   16  0157  1469         JSB     LOAD_GPR                ; load GPR's with data
           0000003C'EF    01   C8  015D  1470         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                           00   BF  0164  1471 6$:     CHMU    #0                      ; execute change mode instruction
           0000003C'EF    04   C8  0166  1472         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                  016D  1473 RENTRY15:                                ; will renter here when completed
         1C 00000000'EF    1A   E1  016D  1474         BBC     #PSL$V_IS,L_ACT_PSL,9$  ; br = not on IS as expected
            5E  00000020'EF    D0  0175  1475         MOVL    L_CUR_ISP,SP            ; reset IS SP
            6E  00000184'8F    D0  017C  1476         MOVL    #7$,(SP)                ; setup return address
                           02   02  0183  1477         REI                             ; get off interrupt stack
                           0B   11  0184  1478 7$:     BRB     9$                      ; skip IPR load
    00000000'8F    00000020'EF   DA  0186  1479 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                   00000C85'EF   16  0191  1480 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
         2C 0000003C'EF    0C   E1  0197  1481         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                  019F  1482         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMUSCB_ERR
                   000003F1'8F   DD  019F                        PUSHL   #T_CHMUSCB_ERR
                   00000969'EF   DF  01A5                        PUSHAL  ERR_MESSAGE
                           00    DD  01AB                        PUSHL   #0
                   00000000'EF   DD  01AD                        PUSHL   KA_UNIT_NO
                           01    DD  01B3                        PUSHL   #$ER
                                  01B5        ;;; TEST 4, SUBTEST 1, ERROR 1
           000100D0 9F    05   FB  01B5                        CALLS   #$$M, @#DS$ERRHARD
         07 0000003C'EF    08   E1  01BC  1483         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                  01C4  1484         $DS_ABORT
            00010020 9F    6C   FA  01C4                        CALLG   (AP), @#DS$ABORT
                                  01CB  1485 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_CHMU
                 0000004C 8F   DD  01CB                        PUSHL   #SCB$L_CHMU
           00010168 9F    01   FB  01D1                        CALLS   #1, @#DS$CLRVEC
                                  01D8  1486         $DS_ENDSUB
                                  01D8        T4_S1_X:
    00010038 9F    00000054'EF   FA  01D8                        CALLG   $S$, @#DS$ENDSUB
                                  01E3  1487         $DS_PAGE
```

```
                        01E3                    .SBTTL  SUBTEST 4.2:  CHMS inst. with vector service on IS
                        01E3  1490 ;+
                        01E3  1491 ; Subtest description:
                        01E3  1492 ;
                        01E3  1493 ;       This subtest checks that a CHMS instruction with the SCB vector entry
                        01E3  1494 ;       <1:0>=1 specifying service on the interrupt stack will cause
                        01E3  1495 ;       the processor to HALT (code = B).
                        01E3  1496 ;
                        01E3  1497 ; Subtest steps:
                        01E3  1498 ;
                        01E3  1499 ;       SUBTEST CHMS instruction with vector service on IS
                        01E3  1500 ;       :  Initialize flags
                        01E3  1501 ;       :  Get interval timer vector to prevent outside interrupts
                        01E3  1502 ;       :  Get CHMS vector specifying service on IS
                        01E3  1503 ;       :  Setup code to set done flag at loc 100
                        01E3  1504 ;       :  Setup code to jump to RENTRY point in next loc
                        01E3  1505 ;       :  Print restart instructions to operator
                        01E3  1506 ;       :  Set ready to execute flag
                        01E3  1507 ;       :  Execute CHMS to cause processor to HALT
                        01E3  1508 ;       :  (this should cause a CHMS service on interrupt stack HALT)
                        01E3  1509 ;       :  IF CHMS does not caues an exception
                        01E3  1510 ;       :  :  THEN
                        01E3  1511 ;       :  :  Set exception error flag
                        01E3  1512 ;       :  ENDIF
                        01E3  1513 ;       :  RENTRY point (restart enters here from exception)
                        01E3  1514 ;       :  IF exception error flag set
                        01E3  1515 ;       :  :  THEN
                        01E3  1516 ;       :  :  Report CHMS exception error
                        01E3  1517 ;       :  :  Exit test
                        01E3  1518 ;       :  ENDIF
                        01E3  1519 ;       :  IF CHMS instruction executed flag set
                        01E3  1520 ;       :  :  THEN
                        01E3  1521 ;       :  :  Report change mode error
                        01E3  1522 ;       :  :  Exit test
                        01E3  1523 ;       :  ENDIF
                        01E3  1524 ;       :  IF done flag not set
                        01E3  1525 ;       :  :  THEN
                        01E3  1526 ;       :  :  Report done error
                        01E3  1527 ;       :  :  Exit test
                        01E3  1528 ;       :  ENDIF
                        01E3  1529 ;       :  Read PSL
                        01E3  1530 ;       :  IF IS not clear
                        01E3  1531 ;       :  :  THEN
                        01E3  1532 ;       :  :  Report incorrect stack error in clean up
                        01E3  1533 ;       :  :  Exit test
                        01E3  1534 ;       :  ENDIF
                        01E3  1535 ;       :  Release interval timer vector
                        01E3  1536 ;       :  Release CHMS vector
                        01E3  1537 ;       ENDSUBTEST CHMS instruction with
                        01E3  1538 ;
                        01E3  1539 ;-
                        01E3  1540          $DS_BGNSUB
                        01E3          T4_S2::
00010030 9F   00000060'EF   FA  01E3                  CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  01EE  1541            JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  01F4  1542            CLRL    L_FLAGS                 ; initialize flags
                                01FA  1543 1$:        $DS_SETVEC_S    VECTOR=#SCB$L_CHMS,SRVADR=CHMX_SERVICE,CODE=#1
```

```
                        01    DD  01FA                        PUSHL   #1
                00000C5C'EF   DF  01FC                        PUSHAL  CHMX_SERVICE
                00000048 8F   DD  0202                        PUSHL   #SCB$L_CHMS
            00010160 9F   03  FB  0208                        CALLS   #3, @#DS$SETVEC
                              020F  1544
    00000100'EF   00000040'EF   7D  020F  1545 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
    00000108'EF   00000048'EF   7D  021A  1546         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
    00000004'EF   0000029F'EF   DE  0225  1547         MOVAL   RENTRY16,RENTRY_POINT   ; load rentry address
                              0230  1548 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0B,P1=#6$
                00000296'8F   DD  0230                        PUSHL   #6$
                        0B    DD  0236                        PUSHL   #CODE_0B
                00000085'EF   9F  0238                        PUSHAB  T_INSTRUCTIONS
            000100F0 9F   03  FB  023E                        CALLS   #3$N, @#DS$PRINTF
                              0245  1549         $DS_PRINTF_S    FORMAT=T_CONTINUE
                000000C3'EF   9F  0245                        PUSHAB  T_CONTINUE
            000100F0 9F   01  FB  024B                        CALLS   #3$N, @#DS$PRINTF
    00000020'EF   00000000'EF   D0  0252  1550 5$:     MOVL    L_ISP,L_CUR_ISP         ; save current IS SP
    00000000'EF   00000000'EF   D0  025D  1551         MOVL    L_KSP,L_EXP_SP          ; load expected SP
    00000000'EF   00000000'EF   D0  0268  1552         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
    00000000'EF   00000000'EF   D0  0273  1553         MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
    00000000'EF   0000095D'EF   D0  027E  1554         MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                00000BDE'EF   16  0289  1555         JSB     LOAD_GPR                ; load GPR's with data
            0000003C'EF   01  C8  028F  1556         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                        00    BE  0296  1557 6$:     CHMS    #0                      ; execute change mode instruction
            0000003C'EF   04  C8  0298  1558         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                              029F  1559 RENTRY16:                                  ; will renter here when completed
        1C 00000000'EF   1A  E1  029F  1560         BBC     #PSL$V_IS,L_ACT_PSL,9$  ; br = not on IS as expected
           5E   00000020'EF   D0  02A7  1561         MOVL    L_CUR_ISP,SP            ; reset IS SP
           6E   000002B6'8F   D0  02AE  1562         MOVL    #7$,(SP)                ; setup return address
                        02    02  02B5  1563         REI                             ; get off interrupt stack
                        0B    11  02B6  1564 7$:     BRB     9$                      ; skip IPR load
    00000000'8F   00000020'EF   DA  02B8  1565 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                00000C85'EF   16  02C3  1566 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
          2C 0000003C'EF   0C  E1  02C9  1567         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                              02D1  1568         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMSSCB_ERR
                00000428'8F   DD  02D1                        PUSHL   #T_CHMSSCB_ERR
                00000969'EF   DF  02D7                        PUSHAL  ERR_MESSAGE
                        00    DD  02DD                        PUSHL   #0
                00000000'EF   DD  02DF                        PUSHL   KA_UNIT_NO
                        01    DD  02E5                        PUSHL   #3ER
                              02E7      ;;; TEST 4, SUBTEST 2, ERROR 1
            000100D0 9F   05  FB  02E7                        CALLS   #3$M, @#DS$ERRHARD
          07 0000003C'EF   08  E1  02EE  1569         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                              02F6  1570         $DS_ABORT
            00010020 9F   6C  FA  02F6                        CALLG   (AP), @#DS$ABORT
                              02FD  1571 10$:    $DS_CLRVEC_S    VECTOR=#SCB$L_CHMS
                00000048 8F   DD  02FD                        PUSHL   #SCB$L_CHMS
            00010168 9F   01  FB  0303                        CALLS   #1, @#DS$CLRVEC
                              030A  1572         $DS_ENDSUB
                              030A      T4_S2_X:
    00010038 9F   00000060'EF   FA  030A                        CALLG   $S$, @#DS$ENDSUB
                              0315  1573         $DS_PAGE
```

```
                        0315                    .SBTTL  SUBTEST 4.3:  CHME inst. with vector service on IS
                        0315  1576 ;+
                        0315  1577 ; Subtest description:
                        0315  1578 ;
                        0315  1579 ;       This subtest checks that a CHME instruction with the SCB vector entry
                        0315  1580 ;       <1:0>=1 specifying service on the interrupt stack will cause
                        0315  1581 ;       the processor to HALT (code = B).
                        0315  1582 ;
                        0315  1583 ; Subtest steps:
                        0315  1584 ;
                        0315  1585 ;       SUBTEST CHME instruction with vector service on IS
                        0315  1586 ;       :  Initialize Flags
                        0315  1587 ;       :  Get interval timer vector to prevent outside interrupts
                        0315  1588 ;       :  Get CHME vector specifying service on IS
                        0315  1589 ;       :  Setup code to set done flag at loc 100
                        0315  1590 ;       :  Setup code to jump to RENTRY point in next loc
                        0315  1591 ;       :  Print restart instructions to operator
                        0315  1592 ;       :  Set ready to execute flag
                        0315  1593 ;       :  Execute CHME to cause processor to HALT
                        0315  1594 ;       :  (this should cause a CHME service on interrupt stack HALT)
                        0315  1595 ;       :  IF CHME does not caues an exception
                        0315  1596 ;       :  :  THEN
                        0315  1597 ;       :  :  Set exception error flag
                        0315  1598 ;       :  ENDIF
                        0315  1599 ;       :  RENTRY point (restart enters here from exception)
                        0315  1600 ;       :  IF exception error flag set
                        0315  1601 ;       :  :  THEN
                        0315  1602 ;       :  :  Report CHME exception error
                        0315  1603 ;       :  :  Exit test
                        0315  1604 ;       :  ENDIF
                        0315  1605 ;       :  IF CHME instruction executed flag set
                        0315  1606 ;       :  :  THEN
                        0315  1607 ;       :  :  Report change mode error
                        0315  1608 ;       :  :  Exit test
                        0315  1609 ;       :  ENDIF
                        0315  1610 ;       :  IF done flag not set
                        0315  1611 ;       :  :  THEN
                        0315  1612 ;       :  :  Report done error
                        0315  1613 ;       :  :  Exit test
                        0315  1614 ;       :  ENDIF
                        0315  1615 ;       :  Read PSL
                        0315  1616 ;       :  IF IS not clear
                        0315  1617 ;       :  :  THEN
                        0315  1618 ;       :  :  Report incorrect stack error in clean up
                        0315  1619 ;       :  :  Exit test
                        0315  1620 ;       :  ENDIF
                        0315  1621 ;       :  Release interval timer vector
                        0315  1622 ;       :  Release CHME vector
                        0315  1623 ;       ENDSUBTEST CHME instruction with
                        0315  1624 ;
                        0315  1625 ;-
                        0315  1626          $DS_BGNSUB
                        0315          T4_S3::
00010030 9F   0000006C'EF   FA  0315                  CALLG   $S$, a#DS$BGNSUB
              00000BC6'EF   16  0320  1627          JSB     CLEAR_BUFFER                ; clear data buffers
              0000003C'EF   D4  0326  1628          CLRL    L_FLAGS                     ; initialize flags
                            032C  1629 1$:        $DS_SETVEC_S    VECTOR=#SCB$L_CHME,SRVADR=CHMX_SERVICE,CODE=#1
```

```
                        01   DD  032C                        PUSHL   #1
                00000C5C'EF  DF  032E                        PUSHAL  CHMX_SERVICE
                00000044 8F  DD  0334                        PUSHL   #SCB$L_CHME
            00010160 9F  03  FB  033A                        CALLS   #3, @#DS$SETVEC
                             0341  1630
00000100'EF     00000040'EF  7D  0341  1631 3$:     MOVQ    JUMP,RESTART            ; setup code to renter after halt
00000108'EF     00000048'EF  7D  034C  1632         MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
00000004'EF     000003D1'EF  DE  0357  1633         MOVAL   RENTRY17,RENTRY_POINT   ; load rentry address
                             0362  1634 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0B,P1=#6$
                000003C8'8F  DD  0362                        PUSHL   #6$
                        0B   DD  0368                        PUSHL   #CODE_0B
                00000085'EF  9F  036A                        PUSHAB  T_INSTRUCTIONS
            000100F0 9F  03  FB  0370                        CALLS   #3$N, @#DS$PRINTF
                             0377  1635         $DS_PRINTF_S    FORMAT=T_CONTINUE
                000000C3'EF  9F  0377                        PUSHAB  T_CONTINUE
            000100F0 9F  01  FB  037D                        CALLS   #3$N, @#DS$PRINTF
00000020'EF     00000000'EF  D0  0384  1636 5$:     MOVL    L_ISP,L_CUR_ISP         ; save current IS SP
00000000'EF     00000000'EF  D0  038F  1637         MOVL    L_KSP,L_EXP_SP          ; load expected SP
00000000'EF     00000000'EF  D0  039A  1638         MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
00000000'EF     00000000'EF  D0  03A5  1639         MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
00000000'EF     0000095D'EF  D0  03B0  1640         MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                00000BDE'EF  16  03BB  1641         JSB     LOAD_GPR                ; load GPR's with data
            0000003C'EF  01  C8  03C1  1642         BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                        00   BD  03C8  1643 6$:     CHME    #0                      ; execute change mode instruction
            0000003C'EF  04  C8  03CA  1644         BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                             03D1  1645 RENTRY17:                                ; will renter here when completed
        1C 00000000'EF   1A  E1  03D1  1646         BBC     #PSL$V_IS,L_ACT_PSL,9$  ; br = not on IS as expected
           5E   00000020'EF  D0  03D9  1647         MOVL    L_CUR_ISP,SP            ; reset IS SP
           6E   000003E8'8F  D0  03E0  1648         MOVL    #7$,(SP)                ; setup return address
                        02   02  03E7  1649         REI                             ; get off interrupt stack
                        0B   11  03E8  1650 7$:     BRB     9$                      ; skip IPR load
00000000'8F     00000020'EF  DA  03EA  1651 8$:     MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
                00000C85'EF  16  03F5  1652 9$:     JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
        2C 0000003C'EF   0C  E1  03FB  1653         BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                             0403  1654         $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMESCB_ERR
                0000045F'8F  DD  0403                        PUSHL   #T_CHMESCB_ERR
                00000969'EF  DF  0409                        PUSHAL  ERR_MESSAGE
                        00   DD  040F                        PUSHL   #0
                00000000'EF  DD  0411                        PUSHL   KA_UNIT_NO
                        01   DD  0417                        PUSHL   #$ER
                             0419      ;;; TEST 4, SUBTEST 3, ERROR 1
           000100D0 9F  05  FB  0419                        CALLS   #3$M, @#DS$ERRHARD
        07 0000003C'EF   08  E1  0420  1655         BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                             0428  1656         $DS_ABORT
           00010020 9F  6C  FA  0428                        CALLG   (AP), @#DS$ABORT
                             042F  1657 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_CHME
                00000044 8F  DD  042F                        PUSHL   #SCB$L_CHME
           00010168 9F  01  FB  0435                        CALLS   #1, @#DS$CLRVEC
                             043C  1658         $DS_ENDSUB
                             043C      T4_S3_X:
00010038 9F     0000006C'EF  FA  043C                        CALLG   $S$, @#DS$ENDSUB
                             0447  1659         $DS_PAGE
```

```
                                0447                    .SBTTL  SUBTEST 4.4:  CHMK inst. with vector service on IS
                                0447  1662 ;+
                                0447  1663 ; Subtest description:
                                0447  1664 ;
                                0447  1665 ;       This subtest checks that a CHMK instruction with the SCB vector entry
                                0447  1666 ;       <1:0>=1 specifying service on the interrupt stack will cause
                                0447  1667 ;       the processor to HALT (code = B).
                                0447  1668 ;
                                0447  1669 ; Subtest steps:
                                0447  1670 ;
                                0447  1671 ;       SUBTEST CHMK instruction with vector service on IS
                                0447  1672 ;       :  Initialize Flags
                                0447  1673 ;       :  Get interval timer vector to prevent outside interrupts
                                0447  1674 ;       :  Get CHMK vector specifying service on IS
                                0447  1675 ;       :  Setup code to set done flag at loc 100
                                0447  1676 ;       :  Setup code to jump to RENTRY point in next loc
                                0447  1677 ;       :  Print restart instructions to operator
                                0447  1678 ;       :  Set ready to execute flag
                                0447  1679 ;       :  Execute CHMK to cause processor to HALT
                                0447  1680 ;       :  (this should cause a CHMK service on interrupt stack HALT)
                                0447  1681 ;       :  IF CHMK does not caues an exception
                                0447  1682 ;       :  :  THEN
                                0447  1683 ;       :  :  Set exception error flag
                                0447  1684 ;       :  ENDIF
                                0447  1685 ;       :  RENTRY point (restart enters here from exception)
                                0447  1686 ;       :  IF exception error flag set
                                0447  1687 ;       :  :  THEN
                                0447  1688 ;       :  :  Report CHMK exception error
                                0447  1689 ;       :  :  Exit test
                                0447  1690 ;       :  ENDIF
                                0447  1691 ;       :  IF CHMK instruction executed flag set
                                0447  1692 ;       :  :  THEN
                                0447  1693 ;       :  :  Report change mode error
                                0447  1694 ;       :  :  Exit test
                                0447  1695 ;       :  ENDIF
                                0447  1696 ;       :  IF done flag not set
                                0447  1697 ;       :  :  THEN
                                0447  1698 ;       :  :  Report done error
                                0447  1699 ;       :  :  Exit test
                                0447  1700 ;       :  ENDIF
                                0447  1701 ;       :  Read PSL
                                0447  1702 ;       :  IF IS not clear
                                0447  1703 ;       :  :  THEN
                                0447  1704 ;       :  :  Report incorrect stack error in clean up
                                0447  1705 ;       :  :  Exit test
                                0447  1706 ;       :  ENDIF
                                0447  1707 ;       :  Release interval timer vector
                                0447  1708 ;       :  Release CHMK vector
                                0447  1709 ;       ENDSUBTEST CHMK instruction with
                                0447  1710 ;
                                0447  1711 ;-
                                0447  1712              $DS_BGNSUB
                                0447            T4_S4::
00010030 9F   00000078'EF   FA  0447                        CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  0452  1713              JSB     CLEAR_BUFFER                ; clear data buffers
              0000003C'EF   D4  0458  1714              CLRL    L_FLAGS                     ; initialize flags
                                045E  1715 1$:          $DS_SETVEC_S    VECTOR=#SCB$L_CHMK,SRVADR=CHMX_SERVICE,CODE=#0
```

```
                      00    DD  045E                        PUSHL   #0
            00000C5C'EF     DF  0460                        PUSHAL  CHMX_SERVICE
            00000040 8F     DD  0466                        PUSHL   #SCB$L_CHMK
        00010160 9F   03    FB  046C                        CALLS   #3, @#DS$SETVEC
                            0473  1716
00000100'EF 00000040'EF     7D  0473  1717 3$:      MOVQ    JUMP,RESTART            ; setup code to renter after halt
00000108'EF 00000048'EF     7D  047E  1718          MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
00000004'EF 00000520'EF     DE  0489  1719          MOVAL   RENTRY18,RENTRY_POINT   ; load rentry address
                            0494  1720 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0B,P1=#6$
            00000517'8F     DD  0494                        PUSHL   #6$
                      0B    DD  049A                        PUSHL   #CODE_0B
            00000085'EF     9F  049C                        PUSHAB  T_INSTRUCTIONS
        000100F0 9F   03    FB  04A2                        CALLS   #$$N, @#DS$PRINTF
                            04A9  1721          $DS_PRINTF_S    FORMAT=T_CONTINUE
            000000C3'EF     9F  04A9                        PUSHAB  T_CONTINUE
        000100F0 9F   01    FB  04AF                        CALLS   #$$N, @#DS$PRINTF
00000020'EF 00000000'EF     D0  04B6  1722 5$:      MOVL    L_ISP,L_CUR_ISP         ; save current IS SP
00000000'EF 00000000'EF     D0  04C1  1723          MOVL    L_KSP,L_EXP_SP          ; load expected SP
00000000'EF 00000000'EF     D0  04CC  1724          MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
00000000'EF 00000000'EF     D0  04D7  1725          MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
00000000'EF 0000095D'EF     D0  04E2  1726          MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
            00000BDE'EF     16  04ED  1727          JSB     LOAD_GPR                ; load GPR's with data
00000034'EF 00000000'8F     DB  04F3  1728          MFPR    #PR$_SCBB,L_TEMP        ; load SCB base
00000034'EF 00000040 8F     C0  04FE  1729          ADDL2   #SCB$L_CHMK,L_TEMP      ; offset of CHMK vector
        00000034'FF   01    C8  0509  1730          BISL2   #1,@L_TEMP              ; set service on inter. stack
        0000003C'EF   01    C8  0510  1731          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                      00    BC  0517  1732 6$:      CHMK    #0                      ; execute change mode instruction
        0000003C'EF   04    C8  0519  1733          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                            0520  1734 RENTRY18:                                  ; will renter here when completed
        00000034'FF   01    CA  0520  1735          BICL2   #1,@L_TEMP              ; set service on inter. stack
     1C 00000000'EF   1A    E1  0527  1736          BBC     #PSL$V_IS,L_ACT_PSL,9$  ; br = not on IS as expected
        5E   00000020'EF    D0  052F  1737          MOVL    L_CUR_ISP,SP            ; reset IS SP
        6E   0000053E'8F    D0  0536  1738          MOVL    #7$,(SP)                ; setup return address
                            02  053D  1739          REI                             ; get off interrupt stack
                      0B    11  053E  1740 7$:      BRB     9$                      ; skip IPR load
00000000'8F 00000020'EF     DA  0540  1741 8$:      MTPR    L_CUR_ISP,#PR$_ISP      ; load IS SP into IPR
            00000C85'EF     16  054B  1742 9$:      JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
     2C 0000003C'EF   0C    E1  0551  1743          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                            0559  1744          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_CHMKSCB_ERR
            00000496'8F     DD  0559                        PUSHL   #T_CHMKSCB_ERR
            00000969'EF     DF  055F                        PUSHAL  ERR_MESSAGE
                      00    DD  0565                        PUSHL   #0
            00000000'EF     DD  0567                        PUSHL   KA_UNIT_NO
                      01    DD  056D                        PUSHL   #$ER
                            056F        ;;; TEST 4, SUBTEST 4, ERROR 1
        000100D0 9F   05    FB  056F                        CALLS   #$$M, @#DS$ERRHARD
     07 0000003C'EF   08    E1  0576  1745          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                            057E  1746          $DS_ABORT
        00010020 9F   6C    FA  057E                        CALLG   (AP), @#DS$ABORT
                            0585  1747 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_CHMK
            00000040 8F     DD  0585                        PUSHL   #SCB$L_CHMK
        00010168 9F   01    FB  058B                        CALLS   #1, @#DS$CLRVEC
                            0592  1748          $DS_ENDSUB
                            0592        T4_S4_X:
00010038 9F   00000078'EF   FA  0592                        CALLG   $S$, @#DS$ENDSUB
                            059D  1749          $DS_PAGE
```

```
                        059D                    .SBTTL  SUBTEST 4.5:  Invalid vector HALT
                        059D  1752 ;+
                        059D  1753 ; Subtest description:
                        059D  1754 ;
                        059D  1755 ;       This subtest checks that an XFC instruction with the SCB vector entry
                        059D  1756 ;       <1:0>=3 specifying an invalid vector will cause the processor to HALT
                        059D  1757 ;       (code = 7).
                        059D  1758 ;
                        059D  1759 ; Subtest steps:
                        059D  1760 ;
                        059D  1761 ;       SUBTEST Invalid vector HALT
                        059D  1762 ;       :  Initialize Flags
                        059D  1763 ;       :  Get interval timer vector to prevent outside interrupts
                        059D  1764 ;       :  Get XFC vector bits <1:0> = 3 (invalid vector)
                        059D  1765 ;       :  Setup code to set done flag at loc 100
                        059D  1766 ;       :  Setup code to jump to RENTRY point in next loc
                        059D  1767 ;       :  Print restart instructions to operator
                        059D  1768 ;       :  Set ready to execute flag
                        059D  1769 ;       :  Execute XFC to cause processor to HALT
                        059D  1770 ;       :  (this should cause a invalid vector HALT)
                        059D  1771 ;       :  IF XFC does not caues an exception
                        059D  1772 ;       :  :  THEN
                        059D  1773 ;       :  :  Set exception error flag
                        059D  1774 ;       :  ENDIF
                        059D  1775 ;       :  RENTRY point (restart enters here from exception)
                        059D  1776 ;       :  IF exception error flag set
                        059D  1777 ;       :  :  THEN
                        059D  1778 ;       :  :  Report XFC exception error
                        059D  1779 ;       :  :  Exit test
                        059D  1780 ;       :  ENDIF
                        059D  1781 ;       :  IF XFC instruction executed flag set
                        059D  1782 ;       :  :  THEN
                        059D  1783 ;       :  :  Report XFC error
                        059D  1784 ;       :  :  Exit test
                        059D  1785 ;       :  ENDIF
                        059D  1786 ;       :  IF done flag not set
                        059D  1787 ;       :  :  THEN
                        059D  1788 ;       :  :  Report done error
                        059D  1789 ;       :  :  Exit test
                        059D  1790 ;       :  ENDIF
                        059D  1791 ;       :  Read PSL
                        059D  1792 ;       :  IF IS not clear
                        059D  1793 ;       :  :  THEN
                        059D  1794 ;       :  :  Report incorrect stack error in clean up
                        059D  1795 ;       :  :  Exit test
                        059D  1796 ;       :  ENDIF
                        059D  1797 ;       :  Release interval timer vector
                        059D  1798 ;       :  Release XFC vector
                        059D  1799 ;       ENDSUBTEST Invalid vector HALT
                        059D  1800 ;
                        059D  1801 ;-
                        059D  1802          $DS_BGNSUB
                        059D            T4_S5::
00010030 9F   00000084'EF   FA  059D                    CALLG   $S$, @#DS$BGNSUB
              00000BC6'EF   16  05A8  1803          JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  05AE  1804          CLRL    L_FLAGS                 ; initialize flags
                                05B4  1805          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_ERROR,CODE=#3
```

```
                          03   DD  05B4                        PUSHL   #3
                00000C6C'EF    DF  05B6                        PUSHAL  XFC_ERROR
                          14   DD  05BC                        PUSHL   #SCB$L_OPCCUS
             00010160 9F   03   FB  05BE                        CALLS   #3, @#DS$SETVEC
00000100'EF    00000040'EF    7D  05C5  1806 3$:      MOVQ    JUMP,RESTART            ; setup code to renter after halt
00000108'EF    00000048'EF    7D  05D0  1807          MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
00000004'EF    00000649'EF    DE  05DB  1808          MOVAL   RENTRY19,RENTRY_POINT   ; load rentry address
                              05E6  1809 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_07,P1=#6$
                00000641'8F    DD  05E6                        PUSHL   #6$
                          07   DD  05EC                        PUSHL   #CODE_07
                00000085'EF    9F  05EE                        PUSHAB  T_INSTRUCTIONS
             000100F0 9F   03   FB  05F4                        CALLS   #3,@#DS$PRINTF
                              05FB  1810          $DS_PRINTF_S    FORMAT=T_CONTINUE
                000000C3'EF    9F  05FB                        PUSHAB  T_CONTINUE
             000100F0 9F   01   FB  0601                        CALLS   #3,@#DS$PRINTF
00000000'EF    00000000'EF    D0  0608  1811 5$:      MOVL    L_KSP,L_EXP_SP          ; load expected SP
00000000'EF    00000000'EF    D0  0613  1812          MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
00000000'EF    00000000'EF    D0  061E  1813          MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
00000000'EF    0000095D'EF    D0  0629  1814          MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                00000BDE'EF    16  0634  1815          JSB     LOAD_GPR                ; load GPR's with data
             0000003C'EF   01   C8  063A  1816          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                              FC  0641  1817 6$:      XFC                             ; execute XFC to cause exception
             0000003C'EF   04   C8  0642  1818          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                              0649  1819 RENTRY19:
             5E   00000000'EF    D0  0649  1820 9$:      MOVL    L_KSP,SP                ; restore SP
                00000C85'EF    16  0650  1821          JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
          2C 0000003C'EF   0C   E1  0656  1822          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                              065E  1823          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_SCB_ERR
                0000032D'8F    DD  065E                        PUSHL   #T_SCB_ERR
                00000969'EF    DF  0664                        PUSHAL  ERR_MESSAGE
                          00   DD  066A                        PUSHL   #0
                00000000'EF    DD  066C                        PUSHL   KA_UNIT_NO
                          01   DD  0672                        PUSHL   #$ER
                              0674      ;;; TEST 4, SUBTEST 5, ERROR 1
             000100D0 9F   05   FB  0674                        CALLS   #$$M, @#DS$ERRHARD
          07 0000003C'EF   08   E1  067B  1824          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                              0683  1825          $DS_ABORT
             00010020 9F   6C   FA  0683                        CALLG   (AP), @#DS$ABORT
                              068A  1826 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                          14   DD  068A                        PUSHL   #SCB$L_OPCCUS
             00010168 9F   01   FB  068C                        CALLS   #1, @#DS$CLRVEC
                              0693  1827          $DS_ENDSUB
                              0693      T4_S5_X:
00010038 9F    00000084'EF    FA  0693                        CALLG   $S$, @#DS$ENDSUB
                              069E  1828          $DS_PAGE
```

```
                                        .SBTTL  SUBTEST 4.6:  SCBB read error HALT
                                ;+
                                ; Subtest description:
                                ;
                                ;       This subtest checks that if the SCBB points to NXM, when an exception
                                ;       occurs the processor will HALT  (code = C).
                                ;
                                ; Subtest steps:
                                ;
                                ;       SUBTEST SCBB read error HALT
                                ;       :  Initialize Flags
                                ;       :  Get interval timer vector to prevent outside interrupts
                                ;       :  Get XFC vector
                                ;       :  Point SCBB to NXM
                                ;       :  Setup code to set done flag at loc 100
                                ;       :  Setup code to jump to RENTRY point in next loc
                                ;       :  Print restart instructions to operator
                                ;       :  Set ready to execute flag
                                ;       :  Execute XFC to cause processor to HALT
                                ;       :  (this should cause an SCBB read error HALT)
                                ;       :  IF XFC does not caues an exception
                                ;       :  :  THEN
                                ;       :  :  Set exception error flag
                                ;       :  ENDIF
                                ;       :  RENTRY point (restart enters here from exception)
                                ;       :  IF exception error flag set
                                ;       :  :  THEN
                                ;       :  :  Report XFC exception error
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  IF XFC instruction executed flag set
                                ;       :  :  THEN
                                ;       :  :  Report SCBB error
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  IF done flag not set
                                ;       :  :  THEN
                                ;       :  :  Report done error
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  Read PSL
                                ;       :  IF IS not clear
                                ;       :  :  THEN
                                ;       :  :  Report incorrect stack error in clean up
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  Release interval timer vector
                                ;       :  Release XFC vector
                                ;       ENDSUBTEST SCBB read error HALT
                                ;
                                ;-

                                $DS_BGNSUB
                                T4_S6::
                                        CALLG   $S$, @#DS$BGNSUB
00010030 9F   00000090'EF   FA  069E
              00000BC6'EF   16  06A9  1883      JSB     CLEAR_BUFFER            ; clear data buffers
              0000003C'EF   D4  06AF  1884      CLRL    L_FLAGS                 ; initialize flags
```

```
                                    06B5  1885 1$:     $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#1
                         01   DD    06B5                       PUSHL   #1
                00000C78'EF   DF    06B7                       PUSHAL  XFC_SERVICE
                         14   DD    06BD                       PUSHL   #SCB$L_OPCCUS
            00010160 9F      03   FB  06BF                       CALLS   #3, @#DS$SETVEC
    00000038'EF  00000000'8F   DB    06C6  1886           MFPR    #PR$_SCBB,L_CUR_SCBB    ; save SCBB
    00000100'EF  00000040'EF   7D    06D1  1887 3$:       MOVQ    JUMP,RESTART            ; setup code to renter after halt
    00000108'EF  00000048'EF   7D    06DC  1888           MOVQ    JUMP+8,RESTART+8        ; load rest of rentry code
    00000004'EF  0000075D'EF   DE    06E7  1889           MOVAL   RENTRY20,RENTRY_POINT   ; load rentry address
                                    06F2  1890 4$:       $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_0C,P1=#6$
                00000755'8F   DD    06F2                       PUSHL   #6$
                         0C   DD    06F8                       PUSHL   #CODE_0C
                00000085'EF   9F    06FA                       PUSHAB  T_INSTRUCTIONS
            000100F0 9F      03   FB  0700                       CALLS   #$$N, @#DS$PRINTF
                                    0707  1891           $DS_PRINTF_S    FORMAT=T_CONTINUE
                0000000'EF   9F    0707                       PUSHAB  T_CONTINUE
            000100F0 9F      01   FB  070D                       CALLS   #$$N, @#DS$PRINTF
            00000000'EF      5E   D0  0714  1892 5$:       MOVL    SP,L_EXP_SP             ; load expected SP
    00000000'EF  00000000'EF   D0    071B  1893           MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
    00000000'EF  00000000'EF   D0    0726  1894           MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
    00000000'EF  0000095D'EF   D0    0731  1895           MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                00000BDE'EF   16    073C  1896           JSB     LOAD_GPR                ; load GPR'S and read AP & FP
    00000000'8F  00500200 8F   DA    0742  1897           MTPR    #L_NXM_ADR,#PR$_SCBB    ; point SCBB to NXM
            0000003C'EF      01   C8  074D  1898           BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                                    0754  1899           NOP                             ; wait a little
                                    0755  1900 6$:       XFC                             ; execute XFC instruction to cause halt
            0000003C'EF      04   C8  0756  1901           BISL2   #M_EXECUTE_ERR,L_FLAGS  ; if no exception occurs will get here
                                    075D  1902
                                    075D  1903 RENTRY20:                                 ; will renter here when completed
            5E   00000000'EF   D0    075D  1904           MOVL    L_KSP,SP                ; restore SP
00000000'8F  00000038'EF   DA    0764  1905           MTPR    L_CUR_SCBB,#PR$_SCBB    ; restore saved SCBB
            00000C85'EF   16    076F  1906 9$:       JSB     ERROR_CHECK             ; check for errors (GPR's,AP,FP,PSL)
        2C 0000003C'EF      0C   E1  0775  1907           BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                    077D  1908           $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_SCBBNXM_ERR
                00000363'8F   DD    077D                       PUSHL   #T_SCBBNXM_ERR
                00000969'EF   DF    0783                       PUSHAL  ERR_MESSAGE
                         00   DD    0789                       PUSHL   #0
                00000000'EF   DD    078B                       PUSHL   KA_UNIT_NO
                         01   DD    0791                       PUSHL   #$ER
                                    0793       ;;; TEST 4, SUBTEST 6, ERROR 1
            000100D0 9F      05   FB  0793                       CALLS   #$$M, @#DS$ERRHARD
        07 0000003C'EF      08   E1  079A  1909           BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                    07A2  1910           $DS_ABORT
            00010020 9F      6C   FA  07A2                       CALLG   (AP), @#DS$ABORT
                                    07A9  1911 10$:      $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                         14   DD    07A9                       PUSHL   #SCB$L_OPCCUS
            00010168 9F      01   FB  07AB                       CALLS   #1, @#DS$CLRVEC
                                    07B2  1912           $DS_ENDSUB
                                    07B2       T4_S6_X:
00010038 9F  00000090'EF   FA    07B2                       CALLG   $S$, @#DS$ENDSUB
                                    07BD  1913           $DS_PAGE
```

```
                                07BD                    .SBTTL  SUBTEST 4.7:  Double machine check HALT
                                07BD  1916 ;+
                                07BD  1917 ; Subtest description:
                                07BD  1918 ;
                                07BD  1919 ;       This subtest checks that a double machine check halt will be generated
                                07BD  1920 ;       if a second machine check occurs before a previous machine check has
                                07BD  1921 ;       been serviced. (code=5)
                                07BD  1922 ;
                                07BD  1923 ; Subtest steps:
                                07BD  1924 ;
                                07BD  1925 ;       SUBTEST Double machine check HALT
                                07BD  1926 ;       :  Initialize Flags
                                07BD  1927 ;       :  Get interval timer vector to prevent outside interrupts
                                07BD  1928 ;       :  Point machine check vector to NXM location
                                07BD  1929 ;       :  Setup code to set done flag at loc 100
                                07BD  1930 ;       :  Setup code to jump to RENTRY point in next loc
                                07BD  1931 ;       :  Print restart instructions to operator
                                07BD  1932 ;       :  Set ready to execute flag
                                07BD  1933 ;       :  Reference NXM to cause machine check and thus a halt
                                07BD  1934 ;       :  IF NXM does not cause an exception
                                07BD  1935 ;       :  :  THEN
                                07BD  1936 ;       :  :  Set exception error flag
                                07BD  1937 ;       :  ENDIF
                                07BD  1938 ;       :  RENTRY point (restart enters here from exception)
                                07BD  1939 ;       :  IF exception error flag set
                                07BD  1940 ;       :  :  THEN
                                07BD  1941 ;       :  :  Report NXM exception error
                                07BD  1942 ;       :  :  Exit test
                                07BD  1943 ;       :  ENDIF
                                07BD  1944 ;       :  IF done flag not set
                                07BD  1945 ;       :  :  THEN
                                07BD  1946 ;       :  :  Report done error
                                07BD  1947 ;       :  :  Exit test
                                07BD  1948 ;       :  ENDIF
                                07BD  1949 ;       :  Release machine check vector
                                07BD  1950 ;       :  Release interval timer vector
                                07BD  1951 ;       ENDSUBTEST Double machine check HALT
                                07BD  1952 ;
                                07BD  1953 ;-
                                07BD  1954          $DS_BGNSUB
                                07BD             T4_S7::
00010030 9F    0000009C'EF   FA  07BD                  CALLG   $S$, @#DS$BGNSUB
         00000BC6'EF   16  07C8  1955          JSB     CLEAR_BUFFER                ; clear data buffers
         0000003C'EF   D4  07CE  1956          CLRL    L_FLAGS                     ; initialize flags
      54 00000000'8F   DB  07D4  1957          MFPR    #PR$_SCBB,R4                ; load SCBB into R4
            54    04   C0  07DB  1958          ADDL2   #SCB$L_MACHCHK,R4           ; add m.c. vector offset
      64 00500200 8F   D0  07DE  1959          MOVL    #L_NXM_ADR,(R4)             ; load NXM address into vector
00000100'EF 00000040'EF   7D  07E5  1960 3$:     MOVQ    JUMP,RESTART                ; setup code to renter after halt
00000108'EF 00000048'EF   7D  07F0  1961          MOVQ    JUMP+8,RESTART+8            ; load rest of rentry code
00000004'EF 0000086F'EF   DE  07FB  1962          MOVAL   RENTRY21,RENTRY_POINT       ; load rentry address
                          0806  1963 4$:     $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_05,P1=#L_NXM_ADR
         00500200 8F   DD  0806                  PUSHL   #L_NXM_ADR
                  05   DD  080C                  PUSHL   #CODE_05
         00000085'EF   9F  080E                  PUSHAB  T_INSTRUCTIONS
      000100F0 9F    03   FB  0814                  CALLS   #$$N, @#DS$PRINTF
                          081B  1964          $DS_PRINTF_S    FORMAT=T_CONTINUE
         000000C3'EF   9F  081B                  PUSHAB  T_CONTINUE
```

```
                000100F0 9F   01    FB  0821                          CALLS   #$$N, @#DS$PRINTF
00000000'EF     00000000'EF   18    C3  0828  1965 5$:       SUBL3   #^X18,L_KSP,L_EXP_SP     ; load expected SP
     00000000'EF     00000000'EF   D0  0834  1966             MOVL    L_KSP,L_EXP_KSP         ; load expected KSP
     00000000'EF     00000000'EF   D0  083F  1967             MOVL    L_ISP,L_EXP_ISP         ; load expected ISP
     00000000'EF     0000095D'EF   D0  084A  1968             MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                00000BDE'EF   16    0855  1969             JSB     LOAD_GPR                ; load GPR's with data
                0000003C'EF   01    C8  085B  1970             BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
                     00500200 9F    D5  0862  1971 6$:       TSTL    @#L_NXM_ADR             ; reference NXM to cause first m.c.
                0000003C'EF   04    C8  0868  1972             BISL2   #M_EXECUTE_ERR,L_FLAGS  ; set flag if no halt occurs
                                        086F  1973 RENTRY21:
                5E   00000000'EF   D0  086F  1974             MOVL    L_KSP,SP                ; restore SP
                     00000C85'EF   16    0876  1975             JSB     ERROR_CHECK             ; check for errors
           2C 0000003C'EF   0C    E1  087C  1976             BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                        0884  1977             $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_DBL_MCHK_ERR -
                     000004CD'8F   DD    0884                          PUSHL   #T_DBL_MCHK_ERR
                     00000969'EF   DF    088A                          PUSHAL  ERR_MESSAGE
                                00 DD    0890                          PUSHL   #0
                     00000000'EF   DD    0892                          PUSHL   KA_UNIT_NO
                                01 DD    0898                          PUSHL   #$ER
                                        089A       ;;; TEST 4, SUBTEST 7, ERROR 1
                000100D0 9F   05    FB  089A                          CALLS   #$$M, @#DS$ERRHARD
           07 0000003C'EF   08    E1  08A1  1978             BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                        08A9  1979             $DS_ABORT
                00010020 9F   6C    FA  08A9                          CALLG   (AP), @#DS$ABORT
                                        08B0  1980 10$:       $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                                04 DD    08B0                          PUSHL   #SCB$L_MACHCHK
                00010168 9F   01    FB  08B2                          CALLS   #1, @#DS$CLRVEC
                                        08B9  1981             $DS_ENDSUB
                                        08B9       T4_S7_X:
00010038 9F     0000009C'EF   FA    08B9                          CALLG   $S$, @#DS$ENDSUB
                                        08C4  1982             $DS_PAGE
```

```
                                        .SBTTL  SUBTEST 4.8:  Transfer to NXM USER WCS
                                ;+
                                ; Subtest description:
                                ;
                                ;       This subtest will attempt to cause a transfer to NXM USER WCS HALT
                                ;       by checking the p-table for a response to 'USER WCS loaded?'. If a 'no'
                                ;       response  is found then this test will execute and attempt to envoke
                                ;       microcode that has not been loaded into WCS. (code=8)
                                ;
                                ; Subtest steps:
                                ;
                                ;       SUBTEST Transfer to NXM USER WCS HALT
                                ;       :  Check p table for USER WCS not loaded
                                ;       :  IF loaded
                                ;       :  :  THEN
                                ;       :  :  Exit subtest
                                ;       :  ENDIF
                                ;       :  Initialize Flags
                                ;       :  Get interval timer vector to prevent outside interrupts
                                ;       :  Get XFC vector to service in WCS
                                ;       :  Setup code to set done flag at loc 100
                                ;       :  Setup code to jump to RENTRY point in next loc
                                ;       :  Print restart instructions to operator
                                ;       :  Set ready to execute flag
                                ;       :  Execute XFC to cause processor to HALT
                                ;       :  (this should cause a transfer to NXM USER WCS HALT)
                                ;       :  IF XFC does not caues an exception
                                ;       :  :  THEN
                                ;       :  :  Set exception error flag
                                ;       :  ENDIF
                                ;       :  RENTRY point (restart enters here from exception)
                                ;       :  IF exception error or instruction executed flags set
                                ;       :  :  THEN
                                ;       :  :  Report XFC exception error
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  IF done flag not set
                                ;       :  :  THEN
                                ;       :  :  Report done error
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  Read PSL
                                ;       :  IF IS not clear
                                ;       :  :  THEN
                                ;       :  :  Report incorrect stack error in clean up
                                ;       :  :  Exit test
                                ;       :  ENDIF
                                ;       :  Release interval timer vector
                                ;       :  Release XFC vector
                                ;       ENDSUBTEST Transfer to NXM USER WCS HALT
                                ;
                                ;-
                                        $DS_BGNSUB
                                T4_S8::
00010030 9F   000000A8'EF   FA                  CALLG   $S$, @#DS$BGNSUB
                                                        ; check p table for WCS loaded
      03 00000000'EF   19   E1          BBC     #V_USR_WCS,L_KA_FLAGS,1$ ; br = user wcs not loaded
```

```
                                        08D7  2039          $DS_EXIT        TEST                    ; exit test if user wcs loaded
                                        08D7                        BRW     TEST_004_X              ; EXIT TEST 4
                    00000BC6'EF   16    08DA  2040 1$:      JSB     CLEAR_BUFFER                    ; clear data buffers
                    0000003C'EF   D4    08E0  2041          CLRL    L_FLAGS                         ; initialize flags
                                        08E6  2042          $DS_SETVEC_S    VECTOR=#SCB$L_OPCCUS,SRVADR=XFC_SERVICE,CODE=#2
                              02    DD    08E6                        PUSHL   #2
                    00000C78'EF   DF    08E8                        PUSHAL  XFC_SERVICE
                              14    DD    08EE                        PUSHL   #SCB$L_OPCCUS
                00010160 9F   03    FB    08F0                        CALLS   #3, @#DS$SETVEC
    00000100'EF 00000040'EF   7D    08F7  2043 3$:      MOVQ    JUMP,RESTART                    ; setup code to renter after halt
    00000108'EF 00000048'EF   7D    0902  2044          MOVQ    JUMP+8,RESTART+8                ; load rest of rentry code
    00000004'EF 0000097B'EF   DE    090D  2045          MOVAL   RENTRY22,RENTRY_POINT           ; load rentry address
                                        0918  2046 4$:      $DS_PRINTF_S    FORMAT=T_INSTRUCTIONS,P0=#CODE_08,P1=#6$
                    00000973'8F   DD    0918                        PUSHL   #6$
                              08    DD    091E                        PUSHL   #CODE_08
                    00000085'EF   9F    0920                        PUSHAB  T_INSTRUCTIONS
                000100F0 9F   03    FB    0926                        CALLS   #3$N, @#DS$PRINTF
                                        092D  2047          $DS_PRINTF_S    FORMAT=T_CONTINUE
                    000000C3'EF   9F    092D                        PUSHAB  T_CONTINUE
                000100F0 9F   01    FB    0933                        CALLS   #3$N, @#DS$PRINTF
    00000000'EF 00000000'EF   D0    093A  2048 5$:      MOVL    L_KSP,L_EXP_SP                  ; load expected SP
    00000000'EF 00000000'EF   D0    0945  2049          MOVL    L_KSP,L_EXP_KSP                 ; load expected KSP
    00000000'EF 00000000'EF   D0    0950  2050          MOVL    L_ISP,L_EXP_ISP                 ; load expected ISP
    00000000'EF 0000095D'EF   D0    095B  2051          MOVL    DATA_1,L_EXP_PSL                ; load expected PSL
                00000BDE'EF   16    0966  2052          JSB     LOAD_GPR                        ; load GPR's with data
                0000003C'EF   01    C8    096C  2053          BISL2   #M_READY,L_FLAGS                ; set ready to execute flag
                                        FC    0973  2054 6$:      XFC                             ; execute XFC to cause halt
                0000003C'EF   04    C8    0974  2055          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; set flag if no halt occurs
                                        0978  2056 RENTRY22:
                5E  00000000'EF   D0    0978  2057          MOVL    L_KSP,SP                        ; restore SP
                    00000C85'EF   16    0982  2058          JSB     ERROR_CHECK                     ; check for errors
            2C 0000003C'EF    0C    E1    0988  2059          BBC     #V_ERROR_FLAG,L_FLAGS,10$ ; br = no errors
                                        0990  2060          $DS_ERRHARD_S   UNIT=KA_UNIT_NO, PRLINK=ERR_MESSAGE,P1=#T_NXM_WCS_ERR
                    0000050B'8F   DD    0990                        PUSHL   #T_NXM_WCS_ERR
                    00000969'EF   DF    0996                        PUSHAL  ERR_MESSAGE
                              00    DD    099C                        PUSHL   #0
                    00000000'EF   DD    099E                        PUSHL   KA_UNIT_NO
                              01    DD    09A4                        PUSHL   #$ER
                                        09A6          ;;; TEST 4, SUBTEST 8, ERROR 1
                000100D0 9F   05    FB    09A6                        CALLS   #3$M, @#DS$ERRHARD
        07 0000003C'EF    08    E1    09AD  2061          BBC     #V_FATAL_ERR,L_FLAGS,10$ ; br = no fatal errors
                                        09B5  2062          $DS_ABORT
                00010020 9F   6C    FA    09B5                        CALLG   (AP), @#DS$ABORT
                                        09BC  2063 10$:     $DS_CLRVEC_S    VECTOR=#SCB$L_OPCCUS
                              14    DD    09BC                        PUSHL   #SCB$L_OPCCUS
                00010168 9F   01    FB    09BE                        CALLS   #1, @#DS$CLRVEC
                                        09C5  2064          $DS_ENDSUB
                                        09C5          T4_S8_X:
    00010038 9F 000000A8'EF   FA    09C5                        CALLG   $S$, @#DS$ENDSUB
                                        09D0  2065
                                        09D0  2066          $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
                    000000C0 8F   DD    09D0                        PUSHL   #SCB$L_TIMER
                00010168 9F   01    FB    09D6                        CALLS   #1, @#DS$CLRVEC
                                        09DD  2067          $DS_ENDTEST
                         50   01    D0    09DD                        MOVL    #1, R0                  ; NORMAL EXIT
                                        09E0          TEST_004_X::
                00010058 9F   6E    FA    09E0                        CALLG   (SP), @#DS$BREAK
```

```
                    04  09E7                           RET                     ; RETURN TO TEST SEQUENCER
                        09E8  2068          $DS_PAGE
                        09E8  2069
                    00000000                           .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                00000004  0000                         .LONG   $TN
                        0004  2071          .END
```

```
$$$                     = 000000A8 R     06        DS$BREAK                        00010058
$$ARGS                  = 0000000A               DS$CANWAIT                      00010070
$$E                     = 00000001               DS$CHANNEL                      00010180
$$M                     = 00000005               DS$CKLOOP                       00010040
$$N                     = 00000001               DS$CLRVEC                       00010168
$$S                     = FFFFFFFF               DS$CNTRLC                       00010078
$$T1                    = 0000002C               DS$CVTREG                       00010080
$ENV                    = 00000001  G            DS$ENDPASS                      00010010
$ER                     = FFFFFFFF               DS$ENDSUB                       00010038
$MO                     = 00000001  G            DS$ERRDEV                       000100C8
$S$                     = 000000A8 R     06        DS$ERRHARD                      000100D0
$ST                     = 00000000               DS$ERRPREP                      00010108
$TN                     = 00000004               DS$ERRSOFT                      000100D8
ALL                     = 00000008  G            DS$ERRSYS                       000100C0
A_BUFFER_BEGIN             ********   X  04        DS$ESCAPE                       00010050
A_HALT_BUFFER              ********   X  02        DS$FREEDBGSYM                   000101B8
A_PSLMNE                   ********   X  02        DS$GETBUF                       00010120
BASE_003                   00000000 R     04        DS$GETMEM                       00010130
BASE_004                   00000000 R     07        DS$GPHARD                       00010018
BIT...                  = 00000005               DS$HELP                         000101B0
BUFSIZ                  = 00000080               DS$INITSCB                      00010170
B_ANSWER                   00000000 R     02        DS$INLOOP                       00010048
CEP_FUNCTIONAL          = 00000000               DS$K_ERROR                  = 00000002
CEP_REPAIR              = 00000001               DS$K_NORMAL                 = 00000001
CHMX_SERVICE               00000C5C R     02        DS$K_SEVERE                 = 00000004
CLEAR_BUFFER               00000BC6 R     02        DS$K_SUBSYS                 = 00000066
CODE_00                 = 00000000               DS$K_WARNING                = 00000000
CODE_01                 = 00000001               DS$LOAD                         00010198
CODE_02                 = 00000002               DS$LOADPCS                      000101C0
CODE_03                 = 00000003               DS$MAPDBGBLOCK                  00010118
CODE_04                 = 00000004               DS$MMOFF                        00010158
CODE_05                 = 00000005               DS$MMON                         00010150
CODE_06                 = 00000006               DS$MOVPHY                       00010148
CODE_07                 = 00000007               DS$MOVVRT                       00010140
CODE_08                 = 00000008               DS$PARSE                        000100B8
CODE_09                 = 00000009               DS$PRINTB                       000100E0
CODE_0A                 = 0000000A               DS$PRINTF                       000100F0
CODE_0B                 = 0000000B               DS$PRINTS                       000100F8
CODE_0C                 = 0000000C               DS$PRINTSIG                     00010100
CPU$V_COMET             = 00000002               DS$PRINTX                       000100E8
CPU$V_HS                = 00000000               DS$PROBE                        000101A0
CPU$V_STAR              = 00000001               DS$RELBUF                       00010128
DATA_003                   00000020 R     04        DS$RELMEM                       00010138
DATA_004                   00000024 R     07        DS$SETIPL                       00010178
DATA_1                     0000095D R     02        DS$SETMAP                       00010188
DATA_2                     00000961 R     02        DS$SETPRIEXV                    00010110
DATA_3                     00000965 R     02        DS$SETVEC                       00010160
DEFAULT                 = 00000000  G            DS$SHOCHAN                      00010190
DS$ABORT                   00010020                 DS$SUMMARY                      00010028
DS$ASKADR                  00010090                 DS$WAITMS                       00010060
DS$ASKDATA                 00010080                 DS$WAITUS                       00010068
DS$ASKLGCL                 00010098                 DS$_ARITH                   = 006600D0
DS$ASKSTR                  000100A0                 DS$_BADLINK                 = 006600F0
DS$ASKVLD                  00010088                 DS$_BADTYPE                 = 006600E8
DS$ATTACH                  000101A8                 DS$_CHME                    = 006600A8
DS$BGNSUB                  00010030                 DS$_CHMK                    = 006600E0
DS$BRANCH                  000100A8                 DS$_DEVNAME                 = 00660108
```

```
DS$_ERROR                       = 00660002
DS$_FHWE                        = 00660068
DS$_FRAGBUF                     = 00660080
DS$_ICBUSY                      = 006600C8
DS$_ICERR                       = 006600C0
DS$_IHWE                        = 00660060
DS$_ILLCHAR                     = 00660018
DS$_ILLPAGCNT                   = 00660078
DS$_ILLUNIT                     = 00660100
DS$_INSFMEM                     = 00660050
DS$_IPL2HI                      = 006600B8
DS$_IVADDR                      = 00660040
DS$_IVVECT                      = 00660038
DS$_KRNLSTK                     = 00660090
DS$_LOGIC                       = 00660070
DS$_MCHK                        = 00660088
DS$_MMOFF                       = 00660058
DS$_NEEDUNIT                    = 006600F8
DS$_NOPCS                       = 00660110
DS$_NORMAL                      = 00660001
DS$_NOSUPPORT                   = 006600B1
DS$_NOTDON                      = 00660030
DS$_NOTIMP                      = 006600B0
DS$_NULLSTR                     = 00660010
DS$_OVERFLOW                    = 00660008
DS$_POWER                       = 00660098
DS$_PROGERR                     = 00660020
DS$_SEVERE                      = 00660004
DS$_TRANSL                      = 006600A0
DS$_TRUNCATE                    = 00660028
DS$_UNEXPINT                    = 006600D8
DS$_VASFULL                     = 00660048
DS$_WARNING                     = 00660000
DSA$AL_APTMAIL                    0000FE00
DSA$AT_APTTXT                     0000FA00
DSA$GL_APTCOM                     0000FE04
DSA$GL_DEVLEN                     0000FE58
DSA$GL_ERRNO                      0000FE44
DSA$GL_EVENT                      0000FE48
DSA$GL_FLAGS                      0000FE00
DSA$GL_MSGTYP                     0000FE40
DSA$GL_PASSES                     0000FE08
DSA$GL_PASSNO                     0000FE54
DSA$GL_SECTNO                     0000FE10
DSA$GL_SID                        0000FE14
DSA$GL_SUBTNO                     0000FE4C
DSA$GL_TESTNO                     0000FE50
DSA$GL_UNITS                      0000FE0C
DSA$GQ_MSGPTR                     0000FE68
DSA$GT_DEVNAM                     0000FE5C
DSA$M_APT                       = 80000000
DSA$M_BELL                      = 00000008
DSA$M_BINARY                    = 00000002
DSA$M_CRD_AUTOTEST              = 00000800
DSA$M_CRD_MENUTEST              = 00010000
DSA$M_CRD_TRACE                 = 00020000
DSA$M_DEBUG                     = 04000000
DSA$M_HALT                      = 00000001
DSA$M_IE1                       = 00000010
DSA$M_IE2                       = 00000020
DSA$M_IE3                       = 00000040
DSA$M_IES                       = 00000080
DSA$M_LOAD_DEBUGGER             = 40000000
DSA$M_LOOP                      = 00000004
DSA$M_NORPT                     = 08000000
DSA$M_OPER                      = 00001000
DSA$M_PASS0                     = 20000000
DSA$M_PROMPT                    = 00002000
DSA$M_QA                        = 00008000
DSA$M_QUICK                     = 00000100
DSA$M_SEARCH                    = 00004000
DSA$M_TRACE                     = 00000400
DSA$M_USER                      = 10000000
DSA$M_VERIFY                    = 00000200
DSA$V_APT                       = 0000001F
DSA$V_BELL                      = 00000003
DSA$V_BINARY                    = 00000001
DSA$V_CRD_AUTOTEST              = 0000000B
DSA$V_CRD_MENUTEST              = 00000010
DSA$V_CRD_TRACE                 = 00000011
DSA$V_DEBUG                     = 0000001A
DSA$V_HALT                      = 00000000
DSA$V_IE1                       = 00000004
DSA$V_IE2                       = 00000005
DSA$V_IE3                       = 00000006
DSA$V_IES                       = 00000007
DSA$V_LOAD_DEBUGGER             = 0000001E
DSA$V_LOOP                      = 00000002
DSA$V_NORPT                     = 0000001B
DSA$V_OPER                      = 0000000C
DSA$V_PASS0                     = 0000001D
DSA$V_PROMPT                    = 0000000D
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
ERR$_MSGADR                     = 0000000C
ERR$_NARGS                      = 0000000A
```

```
ERR$_NUM                = 00000004
ERR$_P1                 = 00000014
ERR$_P2                 = 00000018
ERR$_P3                 = 0000001C
ERR$_P4                 = 00000020
ERR$_P5                 = 00000024
ERR$_P6                 = 00000028
ERR$_POINTER            = 00000010
ERR$_UNIT               = 00000008
ERROR_CHECK               00000C85 RG     02
ERR_MESSAGE               00000969 R      02
GPR_TABLE                 ******** X      02
HALT                    = 00000004 G
IPR                     = 00000006 G
JUMP                      00000040 R      02
KA_PTABLE                 ******** X      04
KA_UNIT_NO                ******** X      04
LDPCTX                  = 0000000B G
LOAD_GPR                  00000BDE R      02
L_ACT_ISP                 ******** X      02
L_ACT_KSP                 ******** X      02
L_ACT_PSL                 ******** X      02
L_ACT_SP                  ******** X      02
L_ADDRESS                 0000001C R      02
L_COUNT                   00000018 R      02
L_CUR_ISP                 00000020 R      02
L_CUR_SCBB                00000038 R      02
L_EXP_AP                  ******** X      02
L_EXP_FP                  ******** X      02
L_EXP_ISP                 ******** X      02
L_EXP_KSP                 ******** X      02
L_EXP_MOD                 00000024 R      02
L_EXP_PSL                 ******** X      02
L_EXP_SP                  ******** X      02
L_EXP_STACK               00000028 R      02
L_FLAGS                   0000003C R      02
L_ISP                     ******** X      04
L_KA_FLAGS                ******** X      07
L_KSP                     ******** X      04
L_NXM_ADR               = 00500200
L_OLDPROT                 0000002C R      02
L_PAGE0END                00000014 R      02
L_PAGEND                  0000000C R      02
L_PAGESTRT                00000008 R      02
L_PAGEZERO                00000010 R      02
L_TEMP                    00000034 R      02
MANUAL                  = 00000001 G
MCHK                    = 00000005 G
MEM                     = 00000009 G
MM_MSG                    00000920 R      02
MODELIST                  ******** X      02
MSG_003                   00000002 R      04
MSG_004                   00000002 R      07
M_BAD_MODE              = 00000040
M_BAD_STK               = 00000080
M_CHMX_ERR              = 00000008
M_DONE                  = 00000002
M_ERROR_FLAG            = 00001000
M_EXECUTE_ERR           = 00000004
M_FATAL_ERR             = 00000100
M_GPR_ERR               = 00000400
M_ISP_ERR               = 00004000
M_ISTR                  = 00010000
M_KSP_ERR               = 00002000
M_PSL_ERR               = 00000200
M_READY                 = 00000001
M_SP_ERR                = 00008000
M_TIMER                 = 00000020
M_XFC                   = 00000800
M_XFC_ERR               = 00000010
PAR$M_ATDEF             = 00000010
PAR$M_ATHI              = 00000008
PAR$M_ATLO              = 00000004
PAR$M_NODEF             = 00000002
PAR$V_ATDEF             = 00000004
PAR$V_ATHI              = 00000003
PAR$V_ATLO              = 00000002
PAR$V_NODEF             = 00000001
PAR$_BIN                = 00000002
PAR$_DEC                = 0000000A
PAR$_HEX                = 00000010
PAR$_NO                 = 00000000
PAR$_OCT                = 00000008
PAR$_YES                = 00000001
POWER                   = 00000003 G
PR$_ISP                   ******** X    02
PR$_KSP                   ******** X    02
PR$_SCBB                  ******** X    07
PRT$C_KR                  ******** X    04
PRT$C_KW                  ******** X    04
PSL$C_EXEC              = 00000001
PSL$C_KERNEL            = 00000000
PSL$C_SUPER             = 00000002
PSL$C_USER              = 00000003
PSL$K_EXEC              = 00000001
PSL$K_KERNEL            = 00000000
PSL$K_SUPER             = 00000002
PSL$K_USER              = 00000003
PSL$M_C                 = 00000001
PSL$M_CBIT              = 00000001
PSL$M_CM                = 80000000
PSL$M_CURMOD            = 03000000
PSL$M_DV                = 00000080
PSL$M_FPD               = 08000000
PSL$M_FU                = 00000040
PSL$M_IPL               = 001F0000
PSL$M_IS                = 04000000
PSL$M_IV                = 00000020
PSL$M_N                 = 00000008
PSL$M_NBIT              = 00000008
PSL$M_PRVMOD            = 00C00000
PSL$M_SAFBITS           = 000037FF
PSL$M_TBIT              = 00000010
PSL$M_TP                = 40000000
```

```
PSL$M_V                         = 00000002
PSL$M_VBIT                      = 00000002
PSL$M_Z                         = 00000004
PSL$M_ZBIT                      = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CBIT                      = 00000000
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_NBIT                      = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_VBIT                      = 00000001
PSL$V_Z                         = 00000002
PSL$V_ZBIT                      = 00000002
READ_GPR                          00000C06 R      02
REG_TABLE                         ******** X    02
RENTRY1                           00000143 R      04
RENTRY10                          000008A8 R      04
RENTRY11                          0000095D R      04
RENTRY12                          00000A1C R      04
RENTRY13                          00000AD1 R      04
RENTRY15                          0000016D R      07
RENTRY16                          0000029F R      07
RENTRY17                          000003D1 R      07
RENTRY18                          00000520 R      07
RENTRY19                          00000649 R      07
RENTRY2                           00000208 R      04
RENTRY20                          0000075D R      07
RENTRY21                          0000086F R      07
RENTRY22                          0000097B R      07
RENTRY3                           00000343 R      04
RENTRY4                           00000434 R      04
RENTRY5                           00000508 R      04
RENTRY6                           000005C0 R      04
RENTRY7                           00000675 R      04
RENTRY8                           00000734 R      04
RENTRY9                           000007E9 R      04
RENTRY_POINT                      00000004 R      02
RESTART                         = 00000100
SCB$L_ACCESS                      00000020
SCB$L_ARITH                       00000034
SCB$L_BREAK                       0000002C
SCB$L_CHME                        00000044
SCB$L_CHMK                        00000040
SCB$L_CHMS                        00000048
SCB$L_CHMU                        0000004C
SCB$L_COMPAT                      00000030
SCB$L_KNLSTK                      00000008
SCB$L_MACHCHK                     00000004
SCB$L_OPCCUS                      00000014
SCB$L_OPCDEC                      00000010
SCB$L_POWER                       0000000C
SCB$L_RADRMOD                     0000001C
SCB$L_ROPRAND                     00000018
SCB$L_RXDB                        000000F8
SCB$L_SFTLVL1                     00000084
SCB$L_SFTLVL10                    000000A8
SCB$L_SFTLVL11                    000000AC
SCB$L_SFTLVL12                    000000B0
SCB$L_SFTLVL13                    000000B4
SCB$L_SFTLVL14                    000000B8
SCB$L_SFTLVL15                    000000BC
SCB$L_SFTLVL2                     00000088
SCB$L_SFTLVL3                     0000008C
SCB$L_SFTLVL4                     00000090
SCB$L_SFTLVL5                     00000094
SCB$L_SFTLVL6                     00000098
SCB$L_SFTLVL7                     0000009C
SCB$L_SFTLVL8                     000000A0
SCB$L_SFTLVL9                     000000A4
SCB$L_TBIT                        00000028
SCB$L_TIMER                       000000C0
SCB$L_TRANSL                      00000024
SCB$L_TXDB                        000000FC
SCB$L_ZERO                        00000000
SEP_FUNCTIONAL                  = 00000002
SEP_REPAIR                      = 00000003
SIZ...                          = 00000001
SYS$ALLOC                         00010238
SYS$ASCTIM                        00010248
SYS$ASSIGN                        00010250
SYS$BINTIM                        00010258
SYS$CANCEL                        00010260
SYS$CANTIM                        00010268
SYS$CLOSE                         000105B8
SYS$CLREF                         00010298
SYS$CONNECT                       000105C0
SYS$DALLOC                        000102D8
SYS$DASSGN                        000102E0
SYS$DISCONNECT                    000105D0
SYS$FAO                           00010350
SYS$FAOL                          00010358
SYS$GET                           00010580
SYS$GETCHN                        000104C8
SYS$GETTIM                        00010378
SYS$LKWSET                        000103A0
SYS$NUMTIM                        000103B8
SYS$OPEN                          00010608
SYS$QIO                           000103C8
SYS$QIOW                          00010200
SYS$READ                          00010590
SYS$READEF                        000103D0
```

```
SYS$SETAST                      000103F8
SYS$SETEF                       00010400
SYS$SETIMR                      00010420
SYS$SETPRI                      00010428
SYS$SETPRT                      00010430  G
SYS$SETRWM                      00010438
SYS$SETSWM                      00010448
SYS$ULKPAG                      00010460
SYS$ULWSET                      00010468
SYS$UNWIND                      00010470
SYS$WAITFR                      00010478
SYS$WFLAND                      00010488
SYS$WFLOR                       00010490
T3_S1                           000000AD RG     04
T3_S1_X                         0000018E R      04
T3_S2                           00000199 RG     04
T3_S2_X                         000003F6 R      04
T3_S3                           00000401 RG     04
T3_S3_X                         00000576 R      04
T3_S4                           00000581 RG     04
T3_S4_X                         000006E9 R      04
T3_S5                           000006F4 RG     04
T3_S5_X                         0000085D R      04
T3_S6                           00000868 RG     04
T3_S6_X                         000009D1 R      04
T3_S7                           000009DC RG     04
T3_S7_X                         00000B45 R      04
T4_S1                           000000B1 RG     07
T4_S1_X                         000001D8 R      07
T4_S2                           000001E3 RG     07
T4_S2_X                         0000030A R      07
T4_S3                           00000315 RG     07
T4_S3_X                         0000043C R      07
T4_S4                           00000447 RG     07
T4_S4_X                         00000592 R      07
T4_S5                           0000059D RG     07
T4_S5_X                         00000693 R      07
T4_S6                           0000069E RG     07
T4_S6_X                         000007B2 R      07
T4_S7                           000007BD RG     07
T4_S7_X                         000008B9 R      07
T4_S8                           000008C4 RG     07
T4_S8_X                         000009C5 R      07
TEST_003                        00000024 RG     04
TEST_003_X                      00000B60 RG     04
TEST_004                        00000028 RG     07
TEST_004_X                      000009E0 RG     07
TIMER_SERVICE                   00000C54 R      02
TU58                          = 00000002  G
T_ACT_PSL                       00000888 R      02
T_BAD_DATA                      00000775 R      02
T_CHME_SCB_ERR                  0000045F R      02
T_CHME_ERR                      000002DD R      02
T_CHMK_SCB_ERR                  00000496 R      02
T_CHMK_ERR                      00000305 R      02
T_CHMS_SCB_ERR                  00000428 R      02
T_CHMS_ERR                      000002B5 R      02
T_CHMU_SCB_ERR                  000003F1 R      02
T_CHMU_ERR                      0000028D R      02
T_CHMX_ERR                      000005FC R      02
T_CONTINUE                      000000C3 R      02
T_DBL_MCHK_ERR                  000004CD R      02
T_EXECUTE                       000005A2 R      02
T_EXITING                       00000114 R      02
T_EXP_PSL                       00000863 R      02
T_GOOD_DATA                     0000079C R      02
T_GPR                           000008CA R      02
T_GPR_ERR                       0000074C R      02
T_HALT_ERR                      000001F1 R      02
T_HALT_MESS                     000001C1 R      02
T_INSTRUCTIONS                  00000085 R      02
T_INVLDIS_ERR                   00000215 R      02
T_ISP_ERR1                      000006B6 R      02
T_ISP_ERR2                      000006E4 R      02
T_KSP_ERR1                      0000065A R      02
T_KSP_ERR2                      00000688 R      02
T_NOT_DONE                      00000574 R      02
T_NOT_ON_IS                     0000083C R      02
T_NOT_ON_KS                     00000818 R      02
T_NOT_READY                     00000542 R      02
T_NOXOR                         0000090C R      02
T_NO_KA730                      0000004D R      02
T_NXMIS_ERR                     00000250 R      02
T_NXM_WCS_ERR                   0000050B R      02
T_PAGEOPROT                     000007B7 R      02
T_PAGEPROT                      000007E5 R      02
T_PSL                           000008AD R      02
T_REST_OFF                      00000191 R      02
T_SCBBNXM_ERR                   00000363 R      02
T_SCB_ERR                       0000032D R      02
T_SP_ERR1                       00000712 R      02
T_SP_ERR2                       00000740 R      02
T_TIMER                         00000632 R      02
T_WARNING                       00000129 R      02
T_WCS_ERR                       0000039F R      02
T_XFC                           000005C7 R      02
T_XOR                           000008F2 R      02
UBI                           = 0000000A  G
V_BAD_MODE                    = 00000006
V_BAD_STK                     = 00000007
V_CHMX_ERR                    = 00000003
V_DONE                        = 00000001
V_ERROR_FLAG                  = 0000000C
V_EXECUTE_ERR                 = 00000002
V_FATAL_ERR                   = 00000008
V_GPR_ERR                     = 0000000A
V_ISP_ERR                     = 0000000E
V_ISTK                        = 00000010
V_KSP_ERR                     = 0000000D
V_MCHK_FLAG                   = 0C000011
V_PSL_ERR                     = 00000009
V_READY                       = 00000000
V_SP_ERR                      = 0000000F
V_TIMER                       = 00000005
```

```
V_USR_WCS                       = 00000019
V_XFC                           = 0000000B
V_XFC_ERR                       = 00000004
WCS                             = 00000007  G
XFC_ERROR                         00000C6C R      02
XFC_SERVICE                       00000C78 R      02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXD	00000DAD (3501.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_003	00000B68 (2920.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000030 (48.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	000000B4 (180.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
TEST_004	000009E8 (2536.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
$TSTCNT	00000004 (4.)	08 (8.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION		REFERENCES...							
$$$	=000000A8-R	2036	(23)	1095	(12)	1202	(13)	1309	(14)	1454	(16)
				1540	(17)	1626	(18)	1712	(19)	1802	(20)
				1882	(21)	1954	(22)	2036	(23)	640	(8)
				748	(9)	876	(10)	987	(11)		
$$ARGS	=0000000A	84	(2)	84	(2)						
$$E	=00000001	2036	(23)	1029	(11)	1137	(12)	1244	(13)	1351	(14)
				1486	(16)	1572	(17)	1658	(18)	1748	(19)
				1827	(20)	1912	(21)	1981	(22)	2064	(23)
				666	(8)	817	(9)	926	(10)		
$$M	=00000005	2060	(23)	1024	(11)	1107	(12)	1132	(12)	1214	(13)
				1239	(13)	1321	(14)	1346	(14)	1376	(15)
				1482	(16)	1568	(17)	1654	(18)	1744	(19)
				1823	(20)	1908	(21)	1977	(22)	2060	(23)
				575	(7)	661	(8)	755	(9)	763	(9)
				779	(9)	807	(9)	813	(9)	886	(10)
				921	(10)	999	(11)				
$$N	=00000001	2060	(23)	1004	(11)	#-1005	(11)	1024	(11)	1107	(12)
				1112	(12)	#-1113	(12)	1132	(12)	1214	(13)
				1219	(13)	#-1220	(13)	1239	(13)	1321	(14)
				1326	(14)	#-1327	(14)	1346	(14)	#-1380	(15)
				#-1386	(15)	#-1393	(15)	1462	(16)	#-1463	(16)
				1482	(16)	1548	(17)	#-1549	(17)	1568	(17)
				1634	(18)	#-1635	(18)	1654	(18)	1720	(19)
				#-1721	(19)	1744	(19)	1809	(20)	#-1810	(20)
				1823	(20)	1890	(21)	#-1891	(21)	1908	(21)
				1963	(22)	#-1964	(22)	1977	(22)	2046	(23)
				#-2047	(23)	2060	(23)	#-352	(5)	#-353	(5)
				#-355	(5)	#-357	(5)	#-359	(5)	#-361	(5)
				#-363	(5)	366	(5)	368	(5)	371	(5)
				373	(5)	375	(5)	377	(5)	383	(5)
				387	(5)	392	(5)	#-394	(5)	402	(5)
				404	(5)	#-407	(5)	#-579	(7)	#-585	(7)
				#-592	(7)	646	(8)	#-647	(8)	661	(8)
				755	(9)	763	(9)	771	(9)	#-772	(9)
				779	(9)	807	(9)	813	(9)	886	(10)
				895	(10)	#-896	(10)	90	(2)	921	(10)
				999	(11)						
$$S	=FFFFFFFF	2067	(23)	1033	(12)	1095	(12)	1140	(13)	1202	(13)
				1247	(14)	1309	(14)	1357	(15)	1376	(15)
				1403	(16)	1454	(16)	1489	(17)	1540	(17)
				1575	(18)	1626	(18)	1661	(19)	1712	(19)
				1751	(20)	1802	(20)	1830	(21)	1882	(21)
				1915	(22)	1954	(22)	1984	(23)	2036	(23)
				556	(7)	575	(7)	602	(8)	640	(8)
				669	(9)	748	(9)	820	(10)	876	(10)
				929	(11)	987	(11)				
$$T1	=0000002C	84	(2)	84	(2)						
$ENV	=00000001	67	(2)								
$ER	=FFFFFFFF	2067	(23)	#-1024	(11)	1095	(12)	#-1107	(12)	#-1132	(12)
				1202	(13)	#-1214	(13)	#-1239	(13)	1309	(14)

```
                                                  #-1321  (14)    #-1346  (14)      1454  (16)    #-1482  (16)
                                                    1540  (17)    #-1568  (17)      1626  (18)    #-1654  (18)
                                                    1712  (19)    #-1744  (19)      1802  (20)    #-1823  (20)
                                                    1882  (21)    #-1908  (21)      1954  (22)    #-1977  (22)
                                                    2036  (23)    #-2060  (23)       640  (8)     #-661   (8)
                                                     748  (9)     #-755   (9)     #-763  (9)     #-779   (9)
                                                  #-807   (9)     #-813   (9)       876  (10)    #-886   (10)
                                                  #-921   (10)      987   (11)   #-999  (11)
$MO                 =00000001       2070  (23)      2070  (23)
$S$                 =00000A8-R      2036  (23)      1029  (11)      1095  (12)      1137  (12)      1202  (13)
                                                      1244  (13)      1309  (14)      1351  (14)      1454  (16)
                                                      1486  (16)      1540  (17)      1572  (17)      1626  (18)
                                                      1658  (18)      1712  (19)      1748  (19)      1802  (20)
                                                      1827  (20)      1882  (21)      1912  (21)      1954  (22)
                                                      1981  (22)      2036  (23)      2064  (23)       640  (8)
                                                       666  (8)        748  (9)       817  (9)       876  (10)
                                                       926  (10)       987  (11)
$ST                 =00000000       2067  (23)      1024  (11)      1029  (11)      1033  (12)      1095  (12)
                                                      1107  (12)      1132  (12)      1137  (12)      1140  (13)
                                                      1202  (13)      1214  (13)      1239  (13)      1244  (13)
                                                      1247  (14)      1309  (14)      1321  (14)      1346  (14)
                                                      1351  (14)      1353  (14)      1376  (15)      1403  (16)
                                                      1454  (16)      1482  (16)      1486  (16)      1489  (17)
                                                      1540  (17)      1568  (17)      1572  (17)      1575  (18)
                                                      1626  (18)      1654  (18)      1658  (18)      1661  (19)
                                                      1712  (19)      1744  (19)      1748  (19)      1751  (20)
                                                      1802  (20)      1823  (20)      1827  (20)      1830  (21)
                                                      1882  (21)      1908  (21)      1912  (21)      1915  (22)
                                                      1954  (22)      1977  (22)      1981  (22)      1984  (23)
                                                      2036  (23)      2060  (23)      2064  (23)      2067  (23)
                                                       575  (7)        602  (8)       640  (8)       661  (8)
                                                       666  (8)        669  (9)       748  (9)       755  (9)
                                                       763  (9)        779  (9)       807  (9)       813  (9)
                                                       817  (9)        820  (10)      876  (10)      886  (10)
                                                       921  (10)       926  (10)      929  (11)      987  (11)
                                                       999  (11)
$TN                 =00000004       2070  (23)      1024  (11)      1033  (12)      1107  (12)      1132  (12)
                                                      1140  (13)      1214  (13)      1239  (13)      1247  (14)
                                                      1321  (14)      1346  (14)      1353  (14)      1357  (15)
                                                      1376  (15)      1381  (15)      1394  (15)      1403  (16)
                                                      1482  (16)      1489  (17)      1568  (17)      1575  (18)
                                                      1654  (18)      1661  (19)      1744  (19)      1751  (20)
                                                      1823  (20)      1830  (21)      1908  (21)      1915  (22)
                                                      1977  (22)      1984  (23)      2039  (23)      2060  (23)
                                                      2067  (23)      2070  (23)       556  (7)       575  (7)
                                                       580  (7)        593  (7)       602  (8)       661  (8)
                                                       669  (9)        763  (9)       807  (9)       820  (10)
                                                       886  (10)       921  (10)      929  (11)      999  (11)
ALL                 =00000008       90    (2)       1376  (15)      575   (7)
A_BUFFER_BEGIN       00000000-XR                    #-773   (9)
A_HALT_BUFFER        00000000-XR                    #-396   (5)       418   (6)     #-445  (6)     #-446   (6)
                                                    #-508   (6)
A_PSLMNE             00000000-XR                      382   (5)       386   (5)       391  (5)
BASE_003             00000000-R      556   (7)        575   (7)
BASE_004             00000000-R      1357  (15)      1376  (15)
BIT...              =00000005       88    (2)         67   (2)       83    (2)       86   (2)       87   (2)
                                                       88   (2)
```

```
BUFSIZ                          =00000080     167     (2)     #-350   (5)     #-382   (5)     #-386   (5)     #-391   (5)
B_ANSWER                         00000000-R   95      (2)       1391  (15)      590   (7)
CEP_FUNCTIONAL                  =00000000     67      (2)
CEP_REPAIR                      =00000001     67      (2)       67    (2)
CHMX_SERVICE                     00000C5C-R   475     (6)       1099  (12)     1206   (13)     1313   (14)     1457   (16)
                                                                 1543  (17)     1629   (18)     1715   (19)      991   (11)
CLEAR_BUFFER                     00000BC6-R   417     (6)       1096  (12)     1203   (13)     1310   (14)     1398   (15)
                                                                 1455  (16)     1541   (17)     1627   (18)     1713   (19)
                                                                 1803  (20)     1883   (21)     1955   (22)     2040   (23)
                                                                  597   (7)      641   (8)      749   (9)      877   (10)
                                                                  988  (11)
CODE_00                         =00000000     153     (2)
CODE_01                         =00000001     154     (2)
CODE_02                         =00000002     155     (2)
CODE_03                         =00000003     156     (2)
CODE_04                         =00000004     157     (2)     #-771   (9)     #-895   (10)
CODE_05                         =00000005     158     (2)     #-1963  (22)
CODE_06                         =00000006     159     (2)     #-646   (8)
CODE_07                         =00000007     160     (2)     #-1809  (20)
CODE_08                         =00000008     161     (2)     #-2046  (23)
CODE_09                         =00000009     162     (2)
CODE_0A                         =0000000A     163     (2)     #-1004  (11)     #-1112  (12)     #-1219  (13)     #-1326  (14)
CODE_0B                         =0000000B     164     (2)     #-1462  (16)     #-1548  (17)     #-1634  (18)     #-1720  (19)
CODE_0C                         =0000000C     165     (2)     #-1890  (21)
CPU$V_COMET                     =00000002     87      (2)
CPU$V_HS                        =00000000     87      (2)
CPU$V_STAR                      =00000001     87      (2)
DATA_003                         00000020-R   575     (7)       575   (7)
DATA_004                         00000024-R   1376    (15)      1376  (15)
DATA_1                           0000095D-R   339     (5)     #-1468  (16)     #-1554  (17)     #-1640  (18)     #-1726  (19)
                                                             #-1814  (20)     #-1895  (21)     #-1968  (22)     #-2051  (23)
                                                             #-651   (8)
DATA_2                           00000961-R   341     (5)     #-1010  (11)     #-1118  (12)     #-1225  (13)     #-1332  (14)
                                                             #-785   (9)
DATA_3                           00000965-R   343     (5)     #-900   (10)
DEFAULT                         =00000000     90      (2)
DS$ABORT                         00010020     85      (2)       1000  (11)     1026   (11)     1108   (12)     1134   (12)
                                                                 1215  (13)     1241   (13)     1322   (14)     1348   (14)
                                                                 1484  (16)     1570   (17)     1656   (18)     1746   (19)
                                                                 1825  (20)     1910   (21)     1979   (22)     2062   (23)
                                                                  663   (8)      665   (8)      756   (9)      764   (9)
                                                                  780   (9)      808   (9)      810   (9)      814   (9)
                                                                  887  (10)      923   (10)
DS$ASKADR                        00010090     85      (2)
DS$ASKDATA                       00010080     85      (2)
DS$ASKLGCL                       00010098     85      (2)       1391  (15)      590   (7)
DS$ASKSTR                        000100A0     85      (2)
DS$ASKVLD                        00010088     85      (2)
DS$ATTACH                        000101A8     85      (2)
DS$BGNSUB                        00010030     85      (2)       1095  (12)     1202   (13)     1309   (14)     1454   (16)
                                                                 1540  (17)     1626   (18)     1712   (19)     1802   (20)
                                                                 1882  (21)     1954   (22)     2036   (23)      640   (8)
                                                                  748   (9)      876   (10)      987   (11)
DS$BRANCH                        000100A8     85      (2)
DS$BREAK                         00010058     85      (2)       1353  (14)     2067   (23)
DS$CANWAIT                       00010070     85      (2)
DS$CHANNEL                       00010180     85      (2)
```

DS$CKLOOP	00010040	85	(2)							
DS$CLRVEC	00010168	85	(2)	1027	(11)	1028	(11)	1135	(12)	1136 (12)
				1242	(13)	1243	(13)	1349	(14)	1350 (14)
				1352	(14)	1485	(16)	1571	(17)	1657 (18)
				1747	(19)	1826	(20)	1911	(21)	1980 (22)
				2063	(23)	2066	(23)	765	(9)	816 (9)
				888	(10)	925	(10)			
DS$CNTRLC	00010078	85	(2)							
DS$CVTREG	000100B0	85	(2)	382	(5)	386	(5)	391	(5)	
DS$ENDPASS	00010010	85	(2)							
DS$ENDSUB	00010038	85	(2)	1029	(11)	1137	(12)	1244	(13)	1351 (14)
				1486	(16)	1572	(17)	1658	(18)	1748 (19)
				1827	(20)	1912	(21)	1981	(22)	2064 (23)
				666	(8)	817	(9)	926	(10)	
DS$ERRDEV	000100C8	85	(2)							
DS$ERRHARD	000100D0	85	(2)	1024	(11)	1107	(12)	1132	(12)	1214 (13)
				1239	(13)	1321	(14)	1346	(14)	1482 (16)
				1568	(17)	1654	(18)	1744	(19)	1823 (20)
				1908	(21)	1977	(22)	2060	(23)	661 (8)
				763	(9)	807	(9)	886	(10)	921 (10)
				999	(11)					
DS$ERRPREP	00010108	85	(2)							
DS$ERRSOFT	000100D8	85	(2)							
DS$ERRSYS	000100C0	85	(2)	755	(9)	779	(9)	813	(9)	
DS$ESCAPE	00010050	85	(2)							
DS$FREEDBGSYM	000101B8	85	(2)							
DS$GETBUF	00010120	85	(2)							
DS$GETMEM	00010130	85	(2)							
DS$GPHARD	00010018	85	(2)							
DS$HELP	000101B0	85	(2)							
DS$INITSCB	00010170	85	(2)							
DS$INLOOP	00010048	85	(2)							
DS$K_ERROR	=00000002	83	(2)							
DS$K_NORMAL	=00000001	83	(2)							
DS$K_SEVERE	=00000004	83	(2)							
DS$K_SUBSYS	=00000066	83	(2)	83	(2)					
DS$K_WARNING	=00000000	83	(2)							
DS$LOAD	00010198	85	(2)							
DS$LOADPCS	000101C0	85	(2)							
DS$MAPDBGBLOCK	00010118	85	(2)							
DS$MMOFF	00010158	85	(2)	815	(9)	924	(10)			
DS$MMON	00010150	85	(2)	752	(9)	879	(10)			
DS$MOVPHY	00010148	85	(2)							
DS$MOVVRT	00010140	85	(2)							
DS$PARSE	000100B8	85	(2)							
DS$PRINTB	000100E0	85	(2)	352	(5)	353	(5)	355	(5)	357 (5)
				359	(5)	361	(5)	363	(5)	366 (5)
				371	(5)	375	(5)	383	(5)	387 (5)
				394	(5)	402	(5)	407	(5)	
DS$PRINTF	000100F0	85	(2)	1004	(11)	1005	(11)	1112	(12)	1113 (12)
				1219	(13)	1220	(13)	1326	(14)	1327 (14)
				1380	(15)	1386	(15)	1393	(15)	1462 (16)
				1463	(16)	1548	(17)	1549	(17)	1634 (18)
				1635	(18)	1720	(19)	1721	(19)	1809 (20)
				1810	(20)	1890	(21)	1891	(21)	1963 (22)
				1964	(22)	2046	(23)	2047	(23)	579 (7)
				585	(7)	592	(7)	646	(8)	647 (8)

				771	(9)	772	(9)	895	(10)	896	(10)
DS$PRINTS	000100F8	85	(2)								
DS$PRINTSIG	00010100	85	(2)								
DS$PRINTX	000100E8	85	(2)	368	(5)	373	(5)	377	(5)	392	(5)
				404	(5)						
DS$PROBE	000101A0	85	(2)								
DS$RELBUF	00010128	85	(2)								
DS$RELMEM	00010138	85	(2)								
DS$SETIPL	00010178	85	(2)								
DS$SETMAP	00010188	85	(2)								
DS$SETPRIEXV	00010110	85	(2)								
DS$SETVEC	00010160	85	(2)	1098	(12)	1099	(12)	1205	(13)	1206	(13)
				1312	(14)	1313	(14)	1399	(15)	1457	(16)
				1543	(17)	1629	(18)	1715	(19)	1805	(20)
				1885	(21)	2042	(23)	598	(7)	757	(9)
				766	(9)	880	(10)	890	(10)	990	(11)
				991	(11)						
DS$SHOCHAN	00010190	85	(2)								
DS$SUMMARY	00010028	85	(2)								
DS$WAITMS	00010060	85	(2)								
DS$WAITUS	00010068	85	(2)								
DS$_ARITH	=006600D0	83	(2)								
DS$_BADLINK	=006600F0	83	(2)								
DS$_BADTYPE	=006600E8	83	(2)								
DS$_CHME	=006600A8	83	(2)								
DS$_CHMK	=006600E0	83	(2)								
DS$_DEVNAME	=00660108	83	(2)								
DS$_ERROR	=00660002	83	(2)								
DS$_FHWE	=00660068	83	(2)								
DS$_FRAGBUF	=00660080	83	(2)								
DS$_ICBUSY	=006600C8	83	(2)								
DS$_ICERR	=006600C0	83	(2)								
DS$_IHWE	=00660060	83	(2)								
DS$_ILLCHAR	=00660018	83	(2)								
DS$_ILLPAGCNT	=00660078	83	(2)								
DS$_ILLUNIT	=00660100	83	(2)								
DS$_INSFMEM	=00660050	83	(2)								
DS$_IPL2HI	=006600B8	83	(2)								
DS$_IVADDR	=00660040	83	(2)								
DS$_IVVECT	=00660038	83	(2)								
DS$_KRNLSTK	=00660090	83	(2)								
DS$_LOGIC	=00660070	83	(2)								
DS$_MCHK	=00660088	83	(2)								
DS$_MMOFF	=00660058	83	(2)								
DS$_NEEDUNIT	=006600F8	83	(2)								
DS$_NOPCS	=00660110	83	(2)								
DS$_NORMAL	=00660001	83	(2)								
DS$_NOSUPPORT	=006600B1	83	(2)								
DS$_NOTDON	=00660030	83	(2)								
DS$_NOTIMP	=006600B0	83	(2)	83	(2)						
DS$_NULLSTR	=00660010	83	(2)								
DS$_OVERFLOW	=00660008	83	(2)								
DS$_POWER	=00660098	83	(2)								
DS$_PROGERR	=00660020	83	(2)								
DS$_SEVERE	=00660004	83	(2)								
DS$_TRANSL	=006600A0	83	(2)								
DS$_TRUNCATE	=00660028	83	(2)								

```
DS$_UNEXPINT                    =006600D8      83      (2)
DS$_VASFULL                     =00660048      83      (2)
DS$_WARNING                     =00660000      83      (2)       83      (2)
DSA$AL_APTMAIL                   0000FE00      87      (2)
DSA$AT_APTTXT                    0000FA00      87      (2)
DSA$GL_APTCOM                    0000FE04      87      (2)
DSA$GL_DEVLEN                    0000FE58      87      (2)
DSA$GL_ERRNO                     0000FE44      87      (2)
DSA$GL_EVENT                     0000FE48      87      (2)
DSA$GL_FLAGS                     0000FE00      87      (2)
DSA$GL_MSGTYP                    0000FE40      87      (2)
DSA$GL_PASSES                    0000FE08      87      (2)
DSA$GL_PASSNO                    0000FE54      87      (2)
DSA$GL_SECTNO                    0000FE10      87      (2)
DSA$GL_SID                       0000FE14      87      (2)
DSA$GL_SUBTNO                    0000FE4C      87      (2)
DSA$GL_TESTNO                    0000FE50      87      (2)
DSA$GL_UNITS                     0000FE0C      87      (2)
DSA$GQ_MSGPTR                    0000FE68      87      (2)
DSA$GT_DEVNAM                    0000FE5C      87      (2)
DSA$M_APT                       =80000000      87      (2)
DSA$M_BELL                      =00000008      87      (2)
DSA$M_BINARY                    =00000002      87      (2)
DSA$M_CRD_AUTOTEST              =00000800      87      (2)
DSA$M_CRD_MENUTEST              =00010000      87      (2)
DSA$M_CRD_TRACE                 =00020000      87      (2)
DSA$M_DEBUG                     =04000000      87      (2)
DSA$M_HALT                      =00000001      87      (2)
DSA$M_IE1                       =00000010      87      (2)
DSA$M_IE2                       =00000020      87      (2)
DSA$M_IE3                       =00000040      87      (2)
DSA$M_IES                       =00000080      87      (2)
DSA$M_LOAD_DEBUGGER             =40000000      87      (2)
DSA$M_LOOP                      =00000004      87      (2)
DSA$M_NORPT                     =08000000      87      (2)
DSA$M_OPER                      =00001000      87      (2)
DSA$M_PASS0                     =20000000      87      (2)
DSA$M_PROMPT                    =00002000      87      (2)
DSA$M_QA                        =00008000      87      (2)
DSA$M_QUICK                     =00000100      87      (2)
DSA$M_SEARCH                    =00004000      87      (2)
DSA$M_TRACE                     =00000400      87      (2)
DSA$M_USER                      =10000000      87      (2)
DSA$M_VERIFY                    =00000200      87      (2)
DSA$V_APT                       =0000001F      87      (2)
DSA$V_BELL                      =00000003      87      (2)
DSA$V_BINARY                    =00000001      87      (2)
DSA$V_CRD_AUTOTEST              =0000000B      87      (2)
DSA$V_CRD_MENUTEST              =00000010      87      (2)
DSA$V_CRD_TRACE                 =00000011      87      (2)
DSA$V_DEBUG                     =0000001A      87      (2)
DSA$V_HALT                      =00000000      87      (2)
DSA$V_IE1                       =00000004      87      (2)
DSA$V_IE2                       =00000005      87      (2)
DSA$V_IE3                       =00000006      87      (2)
DSA$V_IES                       =00000007      87      (2)
DSA$V_LOAD_DEBUGGER             =0000001E      87      (2)
```

```
DSA$V_LOOP                      =00000002      87      (2)
DSA$V_NORPT                     =0000001B      87      (2)
DSA$V_OPER                      =0000000C      87      (2)
DSA$V_PASSO                     =0000001D      87      (2)
DSA$V_PROMPT                    =0000000D      87      (2)
DSA$V_QA                        =0000000F      87      (2)
DSA$V_QUICK                     =00000008      87      (2)
DSA$V_SEARCH                    =0000000E      87      (2)
DSA$V_TRACE                     =0000000A      87      (2)
DSA$V_USER                      =0000001C      87      (2)
DSA$V_VERIFY                    =00000009      87      (2)
ENV$M_DOMAIN                    =00000002      67      (2)
ENV$M_LEVEL                     =00000001      67      (2)
ENV$M_SUPER                     =000003FC      67      (2)
ENV$S_DOMAIN                    =00000001      67      (2)
ENV$S_LEVEL                     =00000001      67      (2)
ENV$S_SUPER                     =00000008      67      (2)
ENV$V_DOMAIN                    =00000001      67      (2)     67      (2)
ENV$V_LEVEL                     =00000000      67      (2)     67      (2)
ENV$V_SUPER                     =00000002      67      (2)
ENV$_CPU                        =00000C00      67      (2)     67      (2)
ENV$_FUNCTIONAL                 =00000000      67      (2)     67      (2)
ENV$_REPAIR                     =00000001      67      (2)     67      (2)
ENV$_SUPER                      =00000001      67      (2)
ENV$_SYSTEM                     =00000001      67      (2)     67      (2)
ERR$_MSGADR                     =0000000C      84      (2)
ERR$_NARGS                      =0000000A      84      (2)
ERR$_NUM                        =00000004      84      (2)
ERR$_P1                         =00000014      84      (2)     353     (5)
ERR$_P2                         =00000018      84      (2)
ERR$_P3                         =0000001C      84      (2)
ERR$_P4                         =00000020      84      (2)
ERR$_P5                         =00000024      84      (2)
ERR$_P6                         =00000028      84      (2)
ERR$_POINTER                    =00000010      84      (2)
ERR$_UNIT                       =00000008      84      (2)
ERROR_CHECK                      00000C85-R    505     (6)     1022    (11)    1130    (12)    1237    (13)    1344    (14)
                                                               1480    (16)    1566    (17)    1652    (18)    1742    (19)
                                                               1821    (20)    1906    (21)    1975    (22)    2058    (23)
                                                               659     (8)     805     (9)     919     (10)
ERR_MESSAGE                      00000969-R    348     (5)     1024    (11)    1132    (12)    1239    (13)    1346    (14)
                                                               1482    (16)    1568    (17)    1654    (18)    1744    (19)
                                                               1823    (20)    1908    (21)    1977    (22)    2060    (23)
                                                               661     (8)     807     (9)     921     (10)
GPR_TABLE                        00000000-XR                   395     (5)     428     (6)     509     (6)
HALT                            =00000004      90      (2)     1376    (15)    575     (7)
IPR                             =00000006      90      (2)
JUMP                             00000040-R    170     (2)   #-1001    (11)   #-1002    (11)   #-1109    (12)   #-1110    (12)
                                                             #-1216    (13)   #-1217    (13)   #-1323    (14)   #-1324    (14)
                                                             #-1459    (16)   #-1460    (16)   #-1545    (17)   #-1546    (17)
                                                             #-1631    (18)   #-1632    (18)   #-1717    (19)   #-1718    (19)
                                                             #-1806    (20)   #-1807    (20)   #-1887    (21)   #-1888    (21)
                                                             #-1960    (22)   #-1961    (22)   #-2043    (23)   #-2044    (23)
                                                             #-643     (8)    #-644     (8)    #-767     (9)    #-768     (9)
                                                             #-891     (10)   #-892     (10)
KA_PTABLE                        00000000-XR                 #-1378    (15)   #-577     (7)
KA_UNIT_NO                       00000000-XR                 #-1024    (11)   #-1107    (12)   #-1132    (12)   #-1214    (13)
```

```
                                                #-1239  (13)  #-1321  (14)  #-1346  (14)  #-1482  (16)
                                                #-1568  (17)  #-1654  (18)  #-1744  (19)  #-1823  (20)
                                                #-1908  (21)  #-1977  (22)  #-2060  (23)  #-661   (8)
                                                #-763   (9)   #-779   (9)   #-807   (9)   #-813   (9)
                                                #-886   (10)  #-921   (10)  #-999   (11)
LDPCTX                  =0000000B       90  (2)
LOAD_GPR                00000BDE-R     427  (6)    1011  (11)    1119  (12)    1226  (13)    1333  (14)
                                                  1469  (16)    1555  (17)    1641  (18)    1727  (19)
                                                  1815  (20)    1896  (21)    1969  (22)    2052  (23)
                                                  652   (8)     787   (9)     901   (10)
L_ACT_ISP               00000000-XR             #-371   (5)   #-373   (5)   #-457   (6)   #-523   (6)
L_ACT_KSP               00000000-XR             #-366   (5)   #-368   (5)   #-456   (6)   #-519   (6)
L_ACT_PSL               00000000-XR               1016  (11)  #-1105  (12)    1106  (12)    1124  (12)
                                                #-1212  (13)    1213  (13)    1231  (13)  #-1319  (14)
                                                  1320  (14)    1338  (14)    1474  (16)    1560  (17)
                                                  1646  (18)    1736  (19)  #-386   (5)   #-387   (5)
                                                #-458   (6)     517   (6)   #-526   (6)     531   (6)
                                                  536   (6)   #-761   (9)     762   (9)     799   (9)
                                                #-884   (10)    885   (10)    912   (10)  #-997   (11)
                                                  998   (11)
L_ACT_SP                00000000-XR             #-375   (5)   #-377   (5)   #-455   (6)   #-514   (6)
L_ADDRESS               0000001C-R     103  (2)  #-773   (9)   #-774   (9)   #-775   (9)   #-776   (9)
                                                #-782   (9)   #-789   (9)
L_COUNT                 00000018-R     102  (2)  #-419   (6)   #-421   (6)
L_CUR_ISP               00000020-R     104  (2)  #-1006  (11)  #-1017  (11)  #-1021  (11)  #-1114  (12)
                                                #-1125  (12)  #-1129  (12)  #-1221  (13)  #-1232  (13)
                                                #-1236  (13)  #-1328  (14)  #-1339  (14)  #-1343  (14)
                                                #-1464  (16)  #-1475  (16)  #-1479  (16)  #-1550  (17)
                                                #-1561  (17)  #-1565  (17)  #-1636  (18)  #-1647  (18)
                                                #-1651  (18)  #-1722  (19)  #-1737  (19)  #-1741  (19)
                                                #-770   (9)   #-800   (9)   #-804   (9)   #-894   (10)
                                                #-913   (10)  #-917   (10)
L_CUR_SCBB              00000038-R     110  (2)  #-1886  (21)  #-1905  (21)
L_EXP_AP                00000000-XR             #-435   (6)
L_EXP_FP                00000000-XR             #-436   (6)
L_EXP_ISP               00000000-XR             #-1009  (11)  #-1117  (12)  #-1224  (13)  #-1331  (14)
                                                #-1467  (16)  #-1553  (17)  #-1639  (18)  #-1725  (19)
                                                #-1813  (20)  #-1894  (21)  #-1967  (22)  #-2050  (23)
                                                #-371   (5)   #-523   (6)   #-650   (8)   #-784   (9)
                                                #-899   (10)
L_EXP_KSP               00000000-XR             #-1008  (11)  #-1116  (12)  #-1223  (13)  #-1330  (14)
                                                #-1466  (16)  #-1552  (17)  #-1638  (18)  #-1724  (19)
                                                #-1812  (20)  #-1893  (21)  #-1966  (22)  #-2049  (23)
                                                #-366   (5)   #-519   (6)   #-649   (8)   #-783   (9)
                                                #-898   (10)
L_EXP_MOD               00000024-R     105  (2)  #-529   (6)   #-531   (6)
L_EXP_PSL               00000000-XR             #-1010  (11)  #-1118  (12)  #-1225  (13)  #-1332  (14)
                                                #-1468  (16)  #-1554  (17)  #-1640  (18)  #-1726  (19)
                                                #-1814  (20)  #-1895  (21)  #-1968  (22)  #-2051  (23)
                                                #-382   (5)   #-383   (5)   #-391   (5)   #-392   (5)
                                                #-526   (6)     529   (6)     534   (6)   #-651   (8)
                                                #-785   (9)   #-900   (10)
L_EXP_SP                00000000-XR             #-1007  (11)  #-1115  (12)  #-1222  (13)  #-1329  (14)
                                                #-1465  (16)  #-1551  (17)  #-1637  (18)  #-1723  (19)
                                                #-1811  (20)  #-1892  (21)  #-1965  (22)  #-2048  (23)
                                                #-375   (5)   #-514   (6)   #-648   (8)   #-782   (9)
                                                #-897   (10)
```

```
L_EXP_STACK                     00000028-R    106     (2)     #-534    (6)     #-536    (6)
L_FLAGS                         0000003C-R    112     (2)     #-1012   (11)    #-1014   (11)    1023     (11)    1025     (11)
                                                              #-1097   (12)    #-1120   (12)    #-1122   (12)    1131     (12)
                                                              1133     (12)    #-1204   (13)    #-1227   (13)    #-1229   (13)
                                                              1238     (13)    1240     (13)    #-1311   (14)    #-1334   (14)
                                                              #-1336   (14)    1345     (14)    1347     (14)    #-1456   (16)
                                                              #-1470   (16)    #-1472   (16)    1481     (16)    1483     (16)
                                                              #-1542   (17)    #-1556   (17)    #-1558   (17)    1567     (17)
                                                              1569     (17)    #-1628   (18)    #-1642   (18)    #-1644   (18)
                                                              1653     (18)    1655     (18)    #-170    (2)     #-1714   (19)
                                                              #-1731   (19)    #-1733   (19)    1743     (19)    1745     (19)
                                                              #-1804   (20)    #-1816   (20)    #-1818   (20)    1822     (20)
                                                              1824     (20)    #-1884   (21)    #-1898   (21)    #-1901   (21)
                                                              1907     (21)    1909     (21)    #-1956   (22)    #-1970   (22)
                                                              #-1972   (22)    1976     (22)    1978     (22)    #-2041   (23)
                                                              #-2053   (23)    #-2055   (23)    2059     (23)    2061     (23)
                                                              354      (5)     356      (5)     358      (5)     360      (5)
                                                              362      (5)     364      (5)     365      (5)     369      (5)
                                                              370      (5)     374      (5)     378      (5)     393      (5)
                                                              406      (5)     #-467    (6)     #-477    (6)     #-486    (6)
                                                              #-495    (6)     #-512    (6)     #-516    (6)     #-518    (6)
                                                              #-521    (6)     522      (6)     #-525    (6)     #-538    (6)
                                                              #-539    (6)     540      (6)     #-541    (6)     542      (6)
                                                              #-543    (6)     544      (6)     #-545    (6)     546      (6)
                                                              #-547    (6)     548      (6)     #-549    (6)     550      (6)
                                                              #-551    (6)     #-642    (8)     #-653    (8)     #-656    (8)
                                                              660      (8)     664      (8)     #-750    (9)     #-788    (9)
                                                              #-795    (9)     806      (9)     809      (9)     #-878    (10)
                                                              #-903    (10)    #-908    (10)    920      (10)    922      (10)
                                                              #-989    (11)
L_ISP                           00000000-XR                   #-1007   (11)    #-1115   (12)    #-1222   (13)    #-1329   (14)
                                                              #-1397   (15)    #-1464   (16)    #-1467   (16)    #-1550   (17)
                                                              #-1553   (17)    #-1636   (18)    #-1639   (18)    #-1722   (19)
                                                              #-1725   (19)    #-1813   (20)    #-1894   (21)    #-1967   (22)
                                                              #-2050   (23)    #-596    (7)     #-650    (8)
L_KA_FLAGS                      00000000-XR                   2038     (23)
L_KSP                           00000000-XR                   #-1008   (11)    #-1116   (12)    #-1223   (13)    #-1330   (14)
                                                              #-1396   (15)    #-1465   (16)    #-1466   (16)    #-1551   (17)
                                                              #-1552   (17)    #-1637   (18)    #-1638   (18)    #-1723   (19)
                                                              #-1724   (19)    #-1811   (20)    #-1812   (20)    #-1820   (20)
                                                              #-1893   (21)    #-1904   (21)    #-1965   (22)    #-1966   (22)
                                                              #-1974   (22)    #-2048   (23)    #-2049   (23)    #-2057   (23)
                                                              #-595    (7)     #-648    (8)     #-658    (8)     #-783    (9)
                                                              #-898    (10)
L_NXM_ADR                       =00500200     149     (2)     #-1897   (21)    #-1959   (22)    #-1963   (22)    #-1971   (22)
L_OLDPROT                        0000002C-R   107     (2)     777      (9)     811      (9)
L_PAGEOEND                       00000014-R   100     (2)
L_PAGEND                         0000000C-R   98      (2)     #-775    (9)
L_PAGESTRT                       00000008-R   97      (2)     #-774    (9)     777      (9)     811      (9)
L_PAGEZERO                       00000010-R   99      (2)     #-751    (9)     753      (9)
L_TEMP                           00000034-R   109     (2)     #-1728   (19)    #-1729   (19)    #-1730   (19)    #-1735   (19)
                                                              #-476    (6)
MANUAL                          =00000001     90      (2)     1376     (15)    575      (7)
MCHK                            =00000005     90      (2)
MEM                             =00000009     90      (2)
MM_MSG                           00000920-R   331     (4)
MODELIST                        00000000-XR                   382      (5)     386      (5)     391      (5)
```

```
MSG_003                          00000002-R   556   (7)      575   (7)
MSG_004                          00000002-R  1357   (15)    1376   (15)
M_BAD_MODE                      =00000040    127   (2)
M_BAD_STK                       =00000080    129   (2)
M_CHMX_ERR                      =00000008    121   (2)      #-477   (6)
M_DONE                          =00000002    117   (2)      #-170   (2)
M_ERROR_FLAG                    =00001000    139   (2)      #-512   (6)    #-516   (6)    #-521   (6)    #-525   (6)
                                                            #-539   (6)    #-541   (6)    #-543   (6)    #-545   (6)
                                                            #-547   (6)    #-549   (6)    #-551   (6)
M_EXECUTE_ERR                   =00000004    119   (2)      #-1014  (11)   #-1122  (12)   #-1229  (13)   #-1336  (14)
                                                            #-1472  (16)   #-1558  (17)   #-1644  (18)   #-1733  (19)
                                                            #-1818  (20)   #-1901  (21)   #-1972  (22)   #-2055  (23)
                                                            #-656   (8)    #-795   (9)    #-908   (10)
M_FATAL_ERR                     =00000100    131   (2)      #-538   (6)
M_GPR_ERR                       =00000400    135   (2)      #-512   (6)
M_ISP_ERR                       =00004000    143   (2)      #-525   (6)
M_ISTK                          =00010000    147   (2)      #-518   (6)
M_KSP_ERR                       =00002000    141   (2)      #-521   (6)
M_PSL_ERR                       =00000200    133   (2)      #-539   (6)
M_READY                         =00000001    115   (2)      #-1012  (11)   #-1120  (12)   #-1227  (13)   #-1334  (14)
                                                            #-1470  (16)   #-1556  (17)   #-1642  (18)   #-1731  (19)
                                                            #-1816  (20)   #-1898  (21)   #-1970  (22)   #-2053  (23)
                                                            #-653   (8)    #-788   (9)    #-903   (10)
M_SP_ERR                        =00008000    145   (2)      #-516   (6)
M_TIMER                         =00000020    125   (2)      #-467   (6)
M_XFC                           =00000800    137   (2)      #-495   (6)
M_XFC_ERR                       =00000010    123   (2)      #-486   (6)
PAR$M_ATDEF                     =00000010    88    (2)
PAR$M_ATHI                      =00000008    88    (2)
PAR$M_ATLO                      =00000004    88    (2)
PAR$M_NODEF                     =00000002    88    (2)
PAR$V_ATDEF                     =00000004    88    (2)
PAR$V_ATHI                      =00000003    88    (2)
PAR$V_ATLO                      =00000002    88    (2)
PAR$V_NODEF                     =00000001    88    (2)
PAR$_BIN                        =00000002    88    (2)
PAR$_DEC                        =0000000A    88    (2)
PAR$_HEX                        =00000010    88    (2)
PAR$_NO                         =00000000    88    (2)      #-1391  (15)   #-590   (7)
PAR$_OCT                        =00000008    88    (2)
PAR$_YES                        =00000001    88    (2)
POWER                           =00000003    90    (2)
PR$_ISP                          00000000-XR          #-1021  (11)   #-1129  (12)   #-1236  (13)   #-1343  (14)
                                                            #-1397  (15)   #-1479  (16)   #-1565  (17)   #-1651  (18)
                                                            #-1741  (19)   #-457   (6)    #-596   (7)    #-804   (9)
                                                            #-917   (10)
PR$_KSP                          00000000-XR          #-456   (6)
PR$_SCBB                         00000000-XR          #-1728  (19)   #-1886  (21)   #-1897  (21)   #-1905  (21)
                                                            #-1957  (22)
PRT$C_KR                         00000000-XR          #-777   (9)
PRT$C_KW                         00000000-XR          #-753   (9)    #-811   (9)
PSL$C_EXEC                      =00000001    86    (2)
PSL$C_KERNEL                    =00000000    86    (2)
PSL$C_SUPER                     =00000002    86    (2)
PSL$C_USER                      =00000003    86    (2)
PSL$K_EXEC                      =00000001    86    (2)
PSL$K_KERNEL                    =00000000    86    (2)
```

```
PSL$K_SUPER                     =00000002     86      (2)
PSL$K_USER                      =00000003     86      (2)
PSL$M_C                         =00000001     86      (2)
PSL$M_CBIT                      =00000001     86      (2)
PSL$M_CM                        =80000000     86      (2)     86      (2)
PSL$M_CURMOD                    =03000000     86      (2)
PSL$M_DV                        =00000080     86      (2)
PSL$M_FPD                       =08000000     86      (2)     86      (2)
PSL$M_FU                        =00000040     86      (2)
PSL$M_IPL                       =001F0000     86      (2)
PSL$M_IS                        =04000000     86      (2)
PSL$M_IV                        =00000020     86      (2)
PSL$M_N                         =00000008     86      (2)
PSL$M_NBIT                      =00000008     86      (2)
PSL$M_PRVMOD                    =00C00000     86      (2)
PSL$M_SAFBITS                   =000037FF     86      (2)
PSL$M_TBIT                      =00000010     86      (2)
PSL$M_TP                        =40000000     86      (2)     86      (2)
PSL$M_V                         =00000002     86      (2)
PSL$M_VBIT                      =00000002     86      (2)
PSL$M_Z                         =00000004     86      (2)
PSL$M_ZBIT                      =00000004     86      (2)
PSL$S_CURMOD                    =00000002     86      (2)     #-528   (6)     #-530   (6)
PSL$S_IPL                       =00000005     86      (2)
PSL$S_PRVMOD                    =00000002     86      (2)
PSL$V_C                         =00000000     86      (2)
PSL$V_CBIT                      =00000000     86      (2)
PSL$V_CM                        =0000001F     86      (2)
PSL$V_CURMOD                    =00000018     86      (2)     #-528   (6)     #-530   (6)
PSL$V_DV                        =00000007     86      (2)
PSL$V_FPD                       =0000001B     86      (2)
PSL$V_FU                        =00000006     86      (2)
PSL$V_IPL                       =00000010     86      (2)
PSL$V_IS                        =0000001A     86      (2)     #-1016  (11)    #-1106  (12)    #-1124  (12)    #-1213  (13)
                                                              #-1231  (13)    #-1320  (14)    #-1338  (14)    #-1474  (16)
                                                              #-1560  (17)    #-1646  (18)    #-1736  (19)    #-517   (6)
                                                              #-533   (6)     #-535   (6)     #-762   (9)     #-799   (9)
                                                              #-885   (10)    #-912   (10)    #-998   (11)
PSL$V_IV                        =00000005     86      (2)
PSL$V_N                         =00000003     86      (2)
PSL$V_NBIT                      =00000003     86      (2)
PSL$V_PRVMOD                    =00000016     86      (2)
PSL$V_TBIT                      =00000004     86      (2)
PSL$V_TP                        =0000001E     86      (2)
PSL$V_V                         =00000001     86      (2)
PSL$V_VBIT                      =00000001     86      (2)
PSL$V_Z                         =00000002     86      (2)
PSL$V_ZBIT                      =00000002     86      (2)
READ_GPR                         00000C06-R   444     (6)     171     (2)
REG_TABLE                        00000000-XR                  399     (5)
RENTRY1                          00000143-R   657     (8)     645     (8)
RENTRY10                         000008A8-R   1211    (13)    1205    (13)
RENTRY11                         0000095D-R   1230    (13)    1218    (13)
RENTRY12                         00000A1C-R   1318    (14)    1312    (14)
RENTRY13                         00000AD1-R   1337    (14)    1325    (14)
RENTRY15                         0000016D-R   1473    (16)    1461    (16)
RENTRY16                         0000029F-R   1559    (17)    1547    (17)
```

```
RENTRY17                        000003D1-R    1645    (18)     1633    (18)
RENTRY18                        00000520-R    1734    (19)     1719    (19)
RENTRY19                        00000649-R    1819    (20)     1808    (20)
RENTRY2                         00000208-R    760     (9)      757     (9)
RENTRY20                        0000075D-R    1903    (21)     1889    (21)
RENTRY21                        0000086F-R    1973    (22)     1962    (22)
RENTRY22                        0000097B-R    2056    (23)     2045    (23)
RENTRY3                         00000343-R    797     (9)      769     (9)
RENTRY4                         00000434-R    883     (10)     880     (10)
RENTRY5                         00000508-R    910     (10)     893     (10)
RENTRY6                         000005C0-R    996     (11)     990     (11)
RENTRY7                         00000675-R    1015    (11)     1003    (11)
RENTRY8                         00000734-R    1104    (12)     1098    (12)
RENTRY9                         000007E9-R    1123    (12)     1111    (12)
RENTRY_POINT                    00000004-R    96      (2)    #-1003    (11)  #-1111    (12)  #-1218    (13)  #-1325    (14)
                                                             #-1461    (16)  #-1547    (17)  #-1633    (18)  #-1719    (19)
                                                             #-1808    (20)  #-1889    (21)  #-1962    (22)  #-2045    (23)
                                                               459     (6)   #-645     (8)   #-769     (9)   #-893     (10)
RESTART                         =00000100     169     (2)    #-1001    (11)  #-1002    (11)  #-1109    (12)  #-1110    (12)
                                                             #-1216    (13)  #-1217    (13)  #-1323    (14)  #-1324    (14)
                                                             #-1459    (16)  #-1460    (16)  #-1545    (17)  #-1546    (17)
                                                             #-1631    (18)  #-1632    (18)  #-1717    (19)  #-1718    (19)
                                                             #-1806    (20)  #-1807    (20)  #-1887    (21)  #-1888    (21)
                                                             #-1960    (22)  #-1961    (22)  #-2043    (23)  #-2044    (23)
                                                             #-643     (8)   #-644     (8)   #-767     (9)   #-768     (9)
                                                             #-891     (10)  #-892     (10)
SCB$L_ACCESS                    00000020      82      (2)
SCB$L_ARITH                     00000034      82      (2)
SCB$L_BREAK                     0000002C      82      (2)
SCB$L_CHME                      00000044      82      (2)    #-1206    (13)  #-1243    (13)  #-1629    (18)  #-1657    (18)
SCB$L_CHMK                      00000040      82      (2)    #-1313    (14)  #-1350    (14)  #-1715    (19)  #-1729    (19)
                                                             #-1747    (19)
SCB$L_CHMS                      00000048      82      (2)    #-1099    (12)  #-1136    (12)  #-1543    (17)  #-1571    (17)
SCB$L_CHMU                      0000004C      82      (2)    #-1028    (11)  #-1457    (16)  #-1485    (16)  #-991     (11)
SCB$L_COMPAT                    00000030      82      (2)
SCB$L_KNLSTK                    00000008      82      (2)
SCB$L_MACHCHK                   00000004      82      (2)    #-1958    (22)  #-1980    (22)
SCB$L_OPCCUS                    00000014      82      (2)    #-1027    (11)  #-1098    (12)  #-1135    (12)  #-1205    (13)
                                                             #-1242    (13)  #-1312    (14)  #-1349    (14)  #-1805    (20)
                                                             #-1826    (20)  #-1885    (21)  #-1911    (21)  #-2042    (23)
                                                             #-2063    (23)  #-757     (9)   #-765     (9)   #-766     (9)
                                                             #-816     (9)   #-880     (10)  #-888     (10)  #-890     (10)
                                                             #-925     (10)  #-990     (11)
SCB$L_OPCDEC                    00000010      82      (2)
SCB$L_POWER                     0000000C      82      (2)
SCB$L_RADRMOD                   0000001C      82      (2)
SCB$L_ROPRAND                   00000018      82      (2)
SCB$L_RXDB                      000000F8      82      (2)
SCB$L_SFTLVL1                   00000084      82      (2)
SCB$L_SFTLVL10                  000000A8      82      (2)
SCB$L_SFTLVL11                  000000AC      82      (2)
SCB$L_SFTLVL12                  000000B0      82      (2)
SCB$L_SFTLVL13                  000000B4      82      (2)
SCB$L_SFTLVL14                  000000B8      82      (2)
SCB$L_SFTLVL15                  000000BC      82      (2)
SCB$L_SFTLVL2                   00000088      82      (2)
SCB$L_SFTLVL3                   0000008C      82      (2)
```

```
SCB$L_SFTLVL4                   00000090      82      (2)
SCB$L_SFTLVL5                   00000094      82      (2)
SCB$L_SFTLVL6                   00000098      82      (2)
SCB$L_SFTLVL7                   0000009C      82      (2)
SCB$L_SFTLVL8                   000000A0      82      (2)
SCB$L_SFTLVL9                   000000A4      82      (2)
SCB$L_TBIT                      00000028      82      (2)
SCB$L_TIMER                     000000C0      82      (2)    #-1352    (14)    #-1399    (15)    #-2066    (23)    #-598     (7)
SCB$L_TRANSL                    00000024      82      (2)
SCB$L_TXDB                      000000FC      82      (2)
SCB$L_ZERO                      00000000      82      (2)
SEP_FUNCTIONAL                  =00000002     67      (2)
SEP_REPAIR                      =00000003     67      (2)
SIZ...                          =00000001     88      (2)      67      (2)       86      (2)       87      (2)       88      (2)
SYS$ALLOC                        00010238     85      (2)
SYS$ASCTIM                       00010248     85      (2)
SYS$ASSIGN                       00010250     85      (2)
SYS$BINTIM                       00010258     85      (2)
SYS$CANCEL                       00010260     85      (2)
SYS$CANTIM                       00010268     85      (2)
SYS$CLOSE                        000105B8     85      (2)
SYS$CLREF                        00010298     85      (2)
SYS$CONNECT                      000105C0     85      (2)
SYS$DALLOC                       000102D8     85      (2)
SYS$DASSGN                       000102E0     85      (2)
SYS$DISCONNECT                   000105D0     85      (2)
SYS$FAO                          00010350     85      (2)
SYS$FAOL                         00010358     85      (2)
SYS$GET                          00010580     85      (2)
SYS$GETCHN                       000104C8     85      (2)
SYS$GETTIM                       00010378     85      (2)
SYS$LKWSET                       000103A0     85      (2)
SYS$NUMTIM                       000103B8     85      (2)
SYS$OPEN                         00010608     85      (2)
SYS$QIO                          000103C8     85      (2)
SYS$QIOW                         00010200     85      (2)
SYS$READ                         00010590     85      (2)
SYS$READEF                       000103D0     85      (2)
SYS$SETAST                       000103F8     85      (2)
SYS$SETEF                        00010400     85      (2)
SYS$SETIMR                       00010420     85      (2)
SYS$SETPRI                       00010428     85      (2)
SYS$SETPRT                       00010430     85      (2)      753     (9)       777     (9)       811     (9)
SYS$SETRWM                       00010438     85      (2)
SYS$SETSWM                       00010448     85      (2)
SYS$ULKPAG                       00010460     85      (2)
SYS$ULWSET                       00010468     85      (2)
SYS$UNWIND                       00010470     85      (2)
SYS$WAITFR                       00010478     85      (2)
SYS$WFLAND                       00010488     85      (2)
SYS$WFLOR                        00010490     85      (2)
T3_S1                           000000AD-R    640     (8)
T3_S1_X                         000001BE-R    666     (8)
T3_S2                           00000199-R    748     (9)
T3_S2_X                         000003F6-R    817     (9)
T3_S3                           00000401-R    876     (10)
T3_S3_X                         00000576-R    926     (10)
```

```
T3_S4                           00000581-R   987    (11)
T3_S4_X                         000006E9-R   1029   (11)
T3_S5                           000006F4-R   1095   (12)
T3_S5_X                         0000085D-R   1137   (12)
T3_S6                           00000868-R   1202   (13)
T3_S6_X                         000009D1-R   1244   (13)
T3_S7                           000009DC-R   1309   (14)
T3_S7_X                         00000B45-R   1351   (14)
T4_S1                           000000B1-R   1454   (16)
T4_S1_X                         000001D8-R   1486   (16)
T4_S2                           000001E3-R   1540   (17)
T4_S2_X                         0000030A-R   1572   (17)
T4_S3                           00000315-R   1626   (18)
T4_S3_X                         0000043C-R   1658   (18)
T4_S4                           00000447-R   1712   (19)
T4_S4_X                         00000592-R   1748   (19)
T4_S5                           0000059D-R   1802   (20)
T4_S5_X                         00000693-R   1827   (20)
T4_S6                           0000069E-R   1882   (21)
T4_S6_X                         000007B2-R   1912   (21)
T4_S7                           000007BD-R   1954   (22)
T4_S7_X                         000008B9-R   1981   (22)
T4_S8                           000008C4-R   2036   (23)
T4_S8_X                         000009C5-R   2064   (23)
TEST_003                        00000024-R   575    (7)     575    (7)
TEST_003_X                      00000B60-R   1353   (14)  #-580    (7)   #-593    (7)
TEST_004                        00000028-R   1376   (15)    1376   (15)
TEST_004_X                      000009E0-R   2067   (23)  #-1381   (15)  #-1394   (15)  #-2039   (23)
TIMER_SERVICE                   00000C54-R   466    (6)     1399   (15)    598    (7)
TU58                           =00000002     90     (2)
T_ACT_PSL                       00000888-R   316    (4)     387    (5)
T_BAD_DATA                      00000775-R   295    (4)
T_CHMESCB_ERR                   0000045F-R   244    (4)  #-1654   (18)
T_CHME_ERR                      000002DD-R   222    (4)  #-1239   (13)
T_CHMKSCB_ERR                   00000496-R   247    (4)  #-1744   (19)
T_CHMK_ERR                      00000305-R   225    (4)  #-1346   (14)
T_CHMSSCB_ERR                   00000428-R   241    (4)  #-1568   (17)
T_CHMS_ERR                      000002B5-R   219    (4)  #-1132   (12)
T_CHMUSCB_ERR                   000003F1-R   238    (4)  #-1482   (16)
T_CHMU_ERR                      0000028D-R   216    (4)  #-1024   (11)
T_CHMX_ERR                      000005FC-R   268    (4)    361    (5)
T_CONTINUE                      000000C3-R   187    (3)    1005   (11)  1113   (12)  1220   (13)  1327   (14)
                                                          1463   (16)  1549   (17)  1635   (18)  1721   (19)
                                                          1810   (20)  1891   (21)  1964   (22)  2047   (23)
                                                           647   (8)    772    (9)   896    (10)
T_DBL_MCHK_ERR                  000004CD-R   250    (4)  #-1977   (22)
T_EXECUTE                       000005A2-R   262    (4)    359    (5)
T_EXITING                       00000114-R   191    (3)    1393   (15)  592    (7)
T_EXP_PSL                       00000863-R   313    (4)    383    (5)
T_GOOD_DATA                     0000079C-R   298    (4)
T_GPR                           000008CA-R   322    (4)    394    (5)
T_GPR_ERR                       0000074C-R   292    (4)
T_HALT_ERR                      000001F1-R   207    (4)  #-661    (8)
T_HALT_MESS                     000001C1-R   204    (4)    352    (5)
T_INSTRUCTIONS                  00000085-R   182    (3)    1004   (11)  1112   (12)  1219   (13)  1326   (14)
                                                          1462   (16)  1548   (17)  1634   (18)  1720   (19)
                                                          1809   (20)  1890   (21)  1963   (22)  2046   (23)
```

```
                                                    646     (8)     771     (9)     895     (10)
T_INVLDIS_ERR                   00000215-R  210     (4)   #-807     (9)
T_ISP_ERR1                      000006B6-R  280     (4)     371     (5)
T_ISP_ERR2                      000006E4-R  283     (4)     373     (5)
T_KSP_ERR1                      0000065A-R  274     (4)     366     (5)
T_KSP_ERR2                      00000688-R  277     (4)     368     (5)
T_NOT_DONE                      00000574-R  259     (4)     357     (5)
T_NOT_ON_IS                     0000083C-R  310     (4)    1107     (12)    1214    (13)    1321    (14)    763     (9)
                                                        886     (10)    999     (11)
T_NOT_ON_KS                     00000818-R  307     (4)
T_NOT_READY                     00000542-R  256     (4)     355     (5)
T_NOXOR                         0000090C-R  328     (4)     404     (5)
T_NO_KA730                      0000004D-R  179     (3)    1380     (15)    579     (7)
T_NXMIS_ERR                     00000250-R  213     (4)   #-921     (10)
T_NXM_WCS_ERR                   0000050B-R  253     (4)   #-2060    (23)
T_PAGEOPROT                     000007B7-R  301     (4)     755     (9)
T_PAGEPROT                      000007E5-R  304     (4)     779     (9)     813     (9)
T_PSL                           000008AD-R  319     (4)     392     (5)
T_REST_OFF                      00000191-R  198     (3)    1391     (15)    590     (7)
T_SCBBNXM_ERR                   00000363-R  231     (4)   #-1908    (21)
T_SCB_ERR                       0000032D-R  228     (4)   #-1823    (20)
T_SP_ERR1                       00000712-R  286     (4)     375     (5)
T_SP_ERR2                       00000740-R  289     (4)     377     (5)
T_TIMER                         00000632-R  271     (4)     407     (5)
T_WARNING                       00000129-R  194     (3)    1386     (15)    585     (7)
T_WCS_ERR                       0000039F-R  234     (4)
T_XFC                           000005C7-R  265     (4)     363     (5)
T_XOR                           000008F2-R  325     (4)     402     (5)
UBI                           =0000000A     90      (2)
V_BAD_MODE                    =00000006     126     (2)
V_BAD_STK                     =00000007     128     (2)
V_CHMX_ERR                    =00000003     120     (2)   #-360     (5)     #-546     (6)
V_DONE                        =00000001     116     (2)   #-356     (5)     #-544     (6)
V_ERROR_FLAG                  =0000000C     138     (2)   #-1023    (11)    #-1131    (12)    #-1238    (13)    #-1345    (14)
                                                        #-1481    (16)    #-1567    (17)    #-1653    (18)    #-1743    (19)
                                                        #-1822    (20)    #-1907    (21)    #-1976    (22)    #-2059    (23)
                                                        #-660     (8)     #-806     (9)     #-920     (10)
V_EXECUTE_ERR                 =00000002     118     (2)   #-358     (5)     #-542     (6)
V_FATAL_ERR                   =00000008     130     (2)   #-1025    (11)    #-1133    (12)    #-1240    (13)    #-1347    (14)
                                                        #-1483    (16)    #-1569    (17)    #-1655    (18)    #-1745    (19)
                                                        #-1824    (20)    #-1909    (21)    #-1978    (22)    #-2061    (23)
                                                        #-664     (8)     #-809     (9)     #-922     (10)
V_GPR_ERR                     =0000000A     134     (2)   #-393     (5)
V_ISP_ERR                     =0000000E     142     (2)   #-370     (5)
V_ISTK                        =00000010     146     (2)   #-364     (5)     #-369     (5)     #-522     (6)
V_KSP_ERR                     =0000000D     140     (2)   #-365     (5)
V_MCHK_FLAG                   =00000011     148     (2)
V_PSL_ERR                     =00000009     132     (2)   #-378     (5)
V_READY                       =00000000     114     (2)   #-354     (5)     #-540     (6)
V_SP_ERR                      =0000000F     144     (2)   #-374     (5)
V_TIMER                       =00000005     124     (2)   #-406     (5)     #-550     (6)
V_USR_WCS                     =00000019     150     (2)   #-2038    (23)
V_XFC                         =0000000B     136     (2)
V_XFC_ERR                     =00000004     122     (2)   #-362     (5)     #-548     (6)
WCS                           =00000007     90      (2)
XFC_ERROR                      00000C6C-R   485     (6)    1805     (20)
XFC_SERVICE                    00000C78-R   494     (6)    1885     (21)    2042     (23)    766     (9)     890     (10)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...									
$D1_ABPROGRAM	1	663 (8)	1000	(11)	1026	(11)	1108	(12)	1134	(12)	1215	(13)
			1241	(13)	1322	(14)	1348	(14)	1484	(16)	1570	(17)
			1656	(18)	1746	(19)	1825	(20)	1910	(21)	1979	(22)
			2062	(23)	663	(8)	665	(8)	756	(9)	764	(9)
			780	(9)	808	(9)	810	(9)	814	(9)	887	(10)
			923	(10)								
$D1_BSUBT	1	640 (8)	1095	(12)	1202	(13)	1309	(14)	1454	(16)	1540	(17)
			1626	(18)	1712	(19)	1802	(20)	1882	(21)	1954	(22)
			2036	(23)	640	(8)	748	(9)	876	(10)	987	(11)
$D1_BTEST	1	575 (7)	1376	(15)	575	(7)						
$D1_ERR	1	661 (8)	1024	(11)	1107	(12)	1132	(12)	1214	(13)	1239	(13)
			1321	(14)	1346	(14)	1482	(16)	1568	(17)	1654	(18)
			1744	(19)	1823	(20)	1908	(21)	1977	(22)	2060	(23)
			661	(8)	755	(9)	763	(9)	779	(9)	807	(9)
			813	(9)	886	(10)	921	(10)	999	(11)		
$D1_ERR_S	1	661 (8)	1024	(11)	1107	(12)	1132	(12)	1214	(13)	1239	(13)
			1321	(14)	1346	(14)	1482	(16)	1568	(17)	1654	(18)
			1744	(19)	1823	(20)	1908	(21)	1977	(22)	2060	(23)
			661	(8)	755	(9)	763	(9)	779	(9)	807	(9)
			813	(9)	886	(10)	921	(10)	999	(11)		
$D1_ESUBT	1	666 (8)	1029	(11)	1137	(12)	1244	(13)	1351	(14)	1486	(16)
			1572	(17)	1658	(18)	1748	(19)	1827	(20)	1912	(21)
			1981	(22)	2064	(23)	666	(8)	817	(9)	926	(10)
$D1_ETEST	1	1353 (14)	1353	(14)	2067	(23)						
$D1_EXTEST	1	580 (7)	1381	(15)	1394	(15)	2039	(23)	580	(7)	593	(7)
$D1_PRINT_S	1	352 (5)	1004	(11)	1005	(11)	1112	(12)	1113	(12)	1219	(13)
			1220	(13)	1326	(14)	1327	(14)	1380	(15)	1386	(15)
			1393	(15)	1462	(16)	1463	(16)	1548	(17)	1549	(17)
			1634	(18)	1635	(18)	1720	(19)	1721	(19)	1809	(20)
			1810	(20)	1890	(21)	1891	(21)	1963	(22)	1964	(22)
			2046	(23)	2047	(23)	352	(5)	353	(5)	355	(5)
			357	(5)	359	(5)	361	(5)	363	(5)	366	(5)
			368	(5)	371	(5)	373	(5)	375	(5)	377	(5)
			383	(5)	387	(5)	392	(5)	394	(5)	402	(5)
			404	(5)	407	(5)	579	(7)	585	(7)	592	(7)
			646	(8)	647	(8)	771	(9)	772	(9)	895	(10)
			896	(10)								
$D1_SBTTL	2	556 (7)	1033	(12)	1140	(13)	1247	(14)	1357	(15)	1403	(16)
			1489	(17)	1575	(18)	1661	(19)	1751	(20)	1830	(21)
			1915	(22)	1984	(23)	556	(7)	602	(8)	669	(9)
			820	(10)	929	(11)						
$D2_BDATA	1	575 (7)	1376	(15)	575	(7)						
$D2_BTEST	1	575 (7)	1376	(15)	575	(7)						
$D2_SBTTL	1	556 (7)	1357	(15)	556	(7)						
$DEF	1	88 (2)	82	(2)	85	(2)	87	(2)				
$DEFEND	1	82 (2)	82	(2)	85	(2)	86	(2)	87	(2)		
$DEFINI	1	82 (2)	82	(2)	85	(2)	86	(2)	87	(2)		
$DS_ABORT	1	663 (8)	1000	(11)	1026	(11)	1108	(12)	1134	(12)	1215	(13)
			1241	(13)	1322	(14)	1348	(14)	1484	(16)	1570	(17)
			1656	(18)	1746	(19)	1825	(20)	1910	(21)	1979	(22)

				2062	(23)	663	(8)	665	(8)	756	(9)	764	(9)
				780	(9)	808	(9)	810	(9)	814	(9)	887	(10)
				923	(10)								
$DS_ASKLGCL_S	1	587	(7)	1388	(15)	587	(7)						
$DS_BCOMPLETE	1	662	(8)	662	(8)	754	(9)	778	(9)	812	(9)		
$DS_BGNMESSAGE	1	349	(5)	349	(5)								
$DS_BGNMOD	1	67	(2)	67	(2)								
$DS_BGNSUB	1	640	(8)	1095	(12)	1202	(13)	1309	(14)	1454	(16)	1540	(17)
				1626	(18)	1712	(19)	1802	(20)	1882	(21)	1954	(22)
				2036	(23)	640	(8)	748	(9)	876	(10)	987	(11)
$DS_BGNTEST	1	575	(7)	1376	(15)	575	(7)						
$DS_BREAK	1	1353	(14)	1353	(14)	2067	(23)						
$DS_CLRVEC_S	1	765	(9)	1027	(11)	1028	(11)	1135	(12)	1136	(12)	1242	(13)
				1243	(13)	1349	(14)	1350	(14)	1352	(14)	1485	(16)
				1571	(17)	1657	(18)	1747	(19)	1826	(20)	1911	(21)
				1980	(22)	2063	(23)	2066	(23)	765	(9)	816	(9)
				888	(10)	925	(10)						
$DS_CVTREG_S	1	380	(5)	380	(5)	384	(5)	389	(5)				
$DS_DSADEF	1	87	(2)	87	(2)								
$DS_DSBDEF	1	87	(2)										
$DS_DSDEF	2	83	(2)	83	(2)								
$DS_DSSDEF	1	85	(2)	85	(2)								
$DS_ENDDATA	1	575	(7)	1376	(15)	575	(7)						
$DS_ENDMESSAGE	1	409	(5)	409	(5)								
$DS_ENDMOD	1	2070	(23)	2070	(23)								
$DS_ENDSUB	1	666	(8)	1029	(11)	1137	(12)	1244	(13)	1351	(14)	1486	(16)
				1572	(17)	1658	(18)	1748	(19)	1827	(20)	1912	(21)
				1981	(22)	2064	(23)	666	(8)	817	(9)	926	(10)
$DS_ENDTEST	1	1353	(14)	1353	(14)	2067	(23)						
$DS_ENVDEF	2	67	(2)	67	(2)								
$DS_ERRDEF	1	84	(2)	84	(2)								
$DS_ERRHARD_S	1	661	(8)	1024	(11)	1107	(12)	1132	(12)	1214	(13)	1239	(13)
				1321	(14)	1346	(14)	1482	(16)	1568	(17)	1654	(18)
				1744	(19)	1823	(20)	1908	(21)	1977	(22)	2060	(23)
				661	(8)	763	(9)	807	(9)	886	(10)	921	(10)
				999	(11)								
$DS_ERRSYS_S	1	755	(9)	755	(9)	779	(9)	813	(9)				
$DS_EXIT	1	580	(7)	1381	(15)	1394	(15)	2039	(23)	580	(7)	593	(7)
$DS_MMOFF_S	1	815	(9)	815	(9)	924	(10)						
$DS_MMON_S	1	752	(9)	752	(9)	879	(10)						
$DS_PAGE	1	554	(6)	1030	(11)	1138	(12)	1245	(13)	1354	(14)	1401	(15)
				1487	(16)	1573	(17)	1659	(18)	1749	(19)	1828	(20)
				1913	(21)	1982	(22)	2068	(23)	554	(6)	600	(7)
				667	(8)	818	(9)	927	(10)				
$DS_PARDEF	1	88	(2)	88	(2)								
$DS_PRINTB_S	1	352	(5)	352	(5)	353	(5)	355	(5)	357	(5)	359	(5)
				361	(5)	363	(5)	366	(5)	371	(5)	375	(5)
				383	(5)	387	(5)	394	(5)	402	(5)	407	(5)
$DS_PRINTF_S	1	579	(7)	1004	(11)	1005	(11)	1112	(12)	1113	(12)	1219	(13)
				1220	(13)	1326	(14)	1327	(14)	1380	(15)	1386	(15)
				1393	(15)	1462	(16)	1463	(16)	1548	(17)	1549	(17)
				1634	(18)	1635	(18)	1720	(19)	1721	(19)	1809	(20)
				1810	(20)	1890	(21)	1891	(21)	1963	(22)	1964	(22)
				2046	(23)	2047	(23)	579	(7)	585	(7)	592	(7)
				646	(8)	647	(8)	771	(9)	772	(9)	895	(10)
				896	(10)								
$DS_PRINTX_S	1	368	(5)	368	(5)	373	(5)	377	(5)	392	(5)	404	(5)

$DS_PSLDEF	1	86	(2)	86	(2)								
$DS_SBTTL	1	556	(7)	1033	(12)	1140	(13)	1247	(14)	1357	(15)	1403	(16)
				1489	(17)	1575	(18)	1661	(19)	1751	(20)	1830	(21)
				1915	(22)	1984	(23)	556	(7)	602	(8)	669	(9)
				820	(10)	929	(11)						
$DS_SCBDEF	1	82	(2)	82	(2)								
$DS_SECDEF	1	90	(2)	90	(2)								
$DS_SETVEC_S	1	598	(7)	1098	(12)	1099	(12)	1205	(13)	1206	(13)	1312	(14)
				1313	(14)	1399	(15)	1457	(16)	1543	(17)	1629	(18)
				1715	(19)	1805	(20)	1885	(21)	2042	(23)	598	(7)
				757	(9)	766	(9)	880	(10)	890	(10)	990	(11)
				991	(11)								
$EQU	1	88	(2)	67	(2)	83	(2)	86	(2)	88	(2)		
$EQULS1	1	86	(2)	67	(2)	83	(2)	86	(2)				
$EQULST	1	67	(2)	67	(2)	83	(2)	86	(2)				
$GBLINI	2	67	(2)	67	(2)	82	(2)	83	(2)	85	(2)	86	(2)
				87	(2)	88	(2)						
$OFFDEF	1	84	(2)	84	(2)								
$PSLDEF	1	86	(2)	86	(2)								
$PUSHADR	1	382	(5)	1024	(11)	1107	(12)	1132	(12)	1214	(13)	1239	(13)
				1321	(14)	1346	(14)	1391	(15)	1482	(16)	1568	(17)
				1654	(18)	1744	(19)	1823	(20)	1908	(21)	1977	(22)
				2060	(23)	382	(5)	386	(5)	391	(5)	590	(7)
				661	(8)	753	(9)	755	(9)	763	(9)	777	(9)
				779	(9)	807	(9)	811	(9)	813	(9)	886	(10)
				921	(10)	999	(11)						
$SETPRT_S	1	753	(9)	753	(9)	777	(9)	811	(9)				
$VIELD	1	67	(2)	67	(2)	86	(2)	87	(2)	88	(2)		
$VIELD1	1	88	(2)	67	(2)	86	(2)	87	(2)	88	(2)		
ENKAX_SECDEF	1	90	(2)	90	(2)								

+-------------------------+
! Performance indicators !
+-------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.05	00:00:00.27
Command processing	25	00:00:00.23	00:00:01.04
Pass 1	1939	00:00:32.09	00:02:07.40
Symbol table sort	1	00:00:00.91	00:00:02.32
Pass 2	1413	00:00:09.75	00:00:39.59
Symbol table output	27	00:00:00.41	00:00:01.53
Psect synopsis output	3	00:00:00.02	00:00:00.10
Cross-reference output	258	00:00:04.63	00:00:13.09
Assembler run totals	3695	00:00:48.10	00:03:05.34

The working set limit was 1000 pages.
363263 bytes (710 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 576 non-local and 167 local symbols.
2071 source lines were read in Pass 1, producing 52 object records in Pass 2.
116 pages of virtual memory were used to define 56 macros.

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	48
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	9
TOTALS (all libraries)	58

Macro library statistics

842 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXD

Table of contents

```
                        .SBTTL  ENKAXE: WRITABLE CONTROL STORE
                        .TITLE  ENKAXE  VAX-11/730 Specific CPU Cluster Exerciser
                        .IDENT  /1.2/

;
; Copyright (C) 1980
; Digital Equipment Corporation, Maynard, Massachusetts 01754
;
; This software is furnished  under a license for use only on a single
; computer  system and  may be  copied only with  the inclusion of the
; above copyright notice.  This software, or any other copies thereof,
; may not be provided or  otherwise made available to any other person
; except for use on such system and to one who agrees to these license
; terms.  Title to and  ownership of the  software  shall at all times
; remain in DEC.
;
; The information in this software is subject to change without notice
; and should  not be construed  as a commitment  by Digital  Equipment
; Corporation.
;
; DEC assumes  no  responsibility  for  the use or  reliability of its
; software on equipment which is not supplied by DEC.
;

;++
; Facility:     VAX Architecture Diagnostic.
;
; Abstract:     It was agreed at the Nebula functional specification review
;               that this module will not be implemented for the VAX 11/730
;               due to the requirements to load WCS with executable code
;               and the inablity for a MACRO program to access WCS.
;               To implement this test would require a special TU58 micro
;               code tape with an additional file containing any micro code
;               that would be used in this test. If WCS were accessible to
;               a MACRO program, as it is in the VAX 11/780 and VAX 11/750,
;               this test would be more feasible.
;
; Environment:  VAX Diagnostic Supervisor.
;
; Author:       Rich Lewis      13-Dec-80       Version 1X
;
;               Tai-Chun Pan    01-Apr-83       Version 1.2
;
;               Added quick flag on Test 7(TU58). If quick flag is set
;               will skip subtest 4 through subtest 11.  Added event flag 3.
;               If event flag 3 is set, will override protection,
;               i.e., go ahead and write on tape.  If run APT & event flag 3
;               is clear & tape is not scratch, will report an error.
;
;               Modified Unibus sizer on Test 8 & 9. make test run faster.
;
;
;
;--
```

```
                        56          .SBTTL  DECLARATIONS
                        57          .LIST   MEB
                        58          .NLIST  CND
                    00000000        59          .PSECT  ENKAXE, PAGE, NOWRT
                        60          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                        61          $DS_BGNMOD      CEP_REPAIR, TN=5
                                ;;;     Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)'
                        62
                        63 ;
                        64 ; Include files:
                        65 ;
                        66
                        67          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                        68
                        72
                        73          $DS_SCBDEF
                        74          $DS_DSADEF
                        75          $DS_DSDEF
                        76          $DS_PARDEF
                        77          $DS_PSLDEF
                        78          $DS_ERRDEF
                        79          $DS_DSSDEF
                        80          $DS_HPODEF
                        81          $DS_KA_DEF
                        82          ENKAX_SECDEF
                        83
                        84 ;
                        85 ; Own storage:
                        86 ;
                        87
                        88
                        89
```

```
                        .SBTTL  SUBROUTINES
```

```
0000     99
0000    100 .SBTTL  ERROR MESSAGES
0000    101
0000    102 ;-----------------------------------------------------------------------
0000    103
```

```
                        0000    105
                        0000    106 .SBTTL  ERROR REPORTING ROUTINES
                        0000    107
                        0000    108 ;----------------------------------------------------------------------------
                        0000    109
                        0000    110         $DS_PAGE
```

```
                                        .SBTTL  TEST 5:  Writable Control Store Test <NYI>
                                        .PSECT  TEST_005, PAGE, NOWRT

                                ; ++
                                ; Test description:
                                ;
                                ;
                                ;
                                ;--

                                        $DS_BGNTEST     <WCS,DEFAULT,ALL>
                                DATA_005:
                                        .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                TEST_005::
                                        .WORD   ^M<>            ; ENTRY MASK

                                        $DS_PAGE
```

```
                                002E                    .SBTTL  SUBTEST 5.1:
                                002E    125 ; +
                                002E    126 ; Subtest description:
                                002E    127 ;
                                002E    128 ;
                                002E    129 ; Subtest steps:
                                002E    130 ;
                                002E    131 ;
                                002E    132 ; Errors:
                                002E    133 ;
                                002E    134 ;
                                002E    135 ;-
                                002E    136
                                002E    137
                                002E    138             $DS_BGNSUB
                                002E            T5_S1::
00010030 9F   00000000'EF   FA  002E                            CALLG   $S$, @#DS$BGNSUB
                                0039    139
                                0039    140             $DS_ENDSUB
                                0039            T5_S1_X:
00010038 9F   00000000'EF   FA  0039                            CALLG   $S$, @#DS$ENDSUB
                                0044    141
                                0044    142             $DS_PAGE
```

```
                                0044                    .SBTTL  SUBTEST 5.2:
                                0044    145 ; +
                                0044    146 ; Subtest description:
                                0044    147 ;
                                0044    148 ;
                                0044    149 ; Subtest steps:
                                0044    150 ;
                                0044    151 ;
                                0044    152 ; Errors:
                                0044    153 ;
                                0044    154 ;-
                                0044    155             $DS_BGNSUB
                                0044            T5_S2::
00010030 9F   0000000C'EF   FA  0044                    CALLG   $S$, @#DS$BGNSUB
                                004F    156
                                004F    157             $DS_ENDSUB
                                004F            T5_S2_X:
00010038 9F   0000000C'EF   FA  004F                    CALLG   $S$, @#DS$ENDSUB
                                005A    158
                                005A    159             $DS_PAGE
```

```
                                005A                    .SBTTL  SUBTEST 5.3:
                                005A    162 ; +
                                005A    163 ; Subtest description:
                                005A    164 ;
                                005A    165 ;
                                005A    166 ; Subtest steps:
                                005A    167 ;
                                005A    168 ; Errors:
                                005A    169 ;
                                005A    170 ;-
                                005A    171
                                005A    172
                                005A    173             $DS_BGNSUB
                                005A                T5_S3::
00010030 9F   00000018'EF   FA  005A                        CALLG   $S$, @#DS$BGNSUB
                                0065    174
                                0065    175             $DS_ENDSUB
                                0065                T5_S3_X:
00010038 9F   00000018'EF   FA  0065                        CALLG   $S$, @#DS$ENDSUB
                                0070    176
                                0070    177             $DS_ENDTEST
                     50   01    D0  0070                        MOVL    #1, R0          ; NORMAL EXIT
                                0073                TEST_005_X::
          00010058 9F   6E    FA  0073                        CALLG   (SP), @#DS$BREAK
                                04  007A                        RET                     ; RETURN TO TEST SEQUENCER
                                007B    178
                                007B    179             $DS_ENDMOD
                           00000000                             .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                     00000005   0000                            .LONG   $TN
                                0004    180             .END
```

```
$$$                     = 00000018 R     06
$$ARGS                  = 0000000A
$$E                     = 00000001
$$M                     = 00000181
$$N                     = 00000000
$$S                     = FFFFFFFF
$$T1                    = 0000002C
$ENV                    = 00000001   G
$ER                     = FFFFFFFF
$MO                     = 00000001   G
$S$                     = 00000018 R     06
$ST                     = 00000000
$TN                     = 00000005
ALL                     = 00000008   G
BASE_005                  00000000 R     04
BIT...                  = 0000001A
CEP_FUNCTIONAL          = 00000000
CEP_REPAIR              = 00000001
CPU$V_COMET             = 00000002
CPU$V_HS                = 00000000
CPU$V_STAR              = 00000001
DATA_005                  00000028 R     04
DEFAULT                 = 00000000   G
DS$ABORT                  00010020
DS$ASKADR                 00010090
DS$ASKDATA                00010080
DS$ASKLGCL                00010098
DS$ASKSTR                 000100A0
DS$ASKVLD                 00010088
DS$ATTACH                 000101A8
DS$BGNSUB                 00010030
DS$BRANCH                 000100A8
DS$BREAK                  00010058
DS$CANWAIT                00010070
DS$CHANNEL                00010180
DS$CKLOOP                 00010040
DS$CLRVEC                 00010168
DS$CNTRLC                 00010078
DS$CVTREG                 000100B0
DS$ENDPASS                00010010
DS$ENDSUB                 00010038
DS$ERRDEV                 000100C8
DS$ERRHARD                000100D0
DS$ERRPREP                00010108
DS$ERRSOFT                000100D8
DS$ERRSYS                 000100C0
DS$ESCAPE                 00010050
DS$FREEDBGSYM             000101B8
DS$GETBUF                 00010120
DS$GETMEM                 00010130
DS$GPHARD                 00010018
DS$HELP                   000101B0
DS$INITSCB                00010170
DS$INLOOP                 00010048
DS$K_ERROR              = 00000002
DS$K_NORMAL             = 00000001
DS$K_SEVERE             = 00000004
DS$K_SUBSYS             = 00000066
DS$K_WARNING            = 00000000
DS$LOAD                   00010198
DS$LOADPCS                000101C0
DS$MAPDBGBLOCK            00010118
DS$MMOFF                  00010158
DS$MMON                   00010150
DS$MOVPHY                 00010148
DS$MOVVRT                 00010140
DS$PARSE                  000100B8
DS$PRINTB                 000100E0
DS$PRINTF                 000100F0
DS$PRINTS                 000100F8
DS$PRINTSIG               00010100
DS$PRINTX                 000100E8
DS$PROBE                  000101A0
DS$RELBUF                 00010128
DS$RELMEM                 00010138
DS$SETIPL                 00010178
DS$SETMAP                 00010188
DS$SETPRIEXV              00010110
DS$SETVEC                 00010160
DS$SHOCHAN                00010190
DS$SUMMARY                00010028
DS$WAITMS                 00010060
DS$WAITUS                 00010068
DS$_ARITH               = 006600D0
DS$_BADLINK             = 006600F0
DS$_BADTYPE             = 006600E8
DS$_CHME                = 006600A8
DS$_CHMK                = 006600E0
DS$_DEVNAME             = 00660108
DS$_ERROR               = 00660002
DS$_FHWE                = 00660068
DS$_FRAGBUF             = 00660080
DS$_ICBUSY              = 006600C8
DS$_ICERR               = 006600C0
DS$_IHWE                = 00660060
DS$_ILLCHAR             = 00660018
DS$_ILLPAGCNT           = 00660078
DS$_ILLUNIT             = 00660100
DS$_INSFMEM             = 00660050
DS$_IPL2HI              = 006600B8
DS$_IVADDR              = 00660040
DS$_IVVECT              = 00660038
DS$_KRNLSTK             = 00660090
DS$_LOGIC               = 00660070
DS$_MCHK                = 00660088
DS$_MMOFF               = 00660058
DS$_NEEDUNIT            = 006600F8
DS$_NOPCS               = 00660110
DS$_NORMAL              = 00660001
DS$_NOSUPPORT           = 006600B1
DS$_NOTDON              = 00660030
DS$_NOTIMP              = 006600B0
DS$_NULLSTR             = 00660010
DS$_OVERFLOW            = 00660008
```

```
DS$_POWER                       = 00660098
DS$_PROGERR                     = 00660020
DS$_SEVERE                      = 00660004
DS$_TRANSL                      = 006600A0
DS$_TRUNCATE                    = 00660028
DS$_UNEXPINT                    = 006600D8
DS$_VASFULL                     = 00660048
DS$_WARNING                     = 00660000
DSA$AL_APTMAIL                    0000FE00
DSA$AT_APTTXT                     0000FA00
DSA$GL_APTCOM                     0000FE04
DSA$GL_DEVLEN                     0000FE58
DSA$GL_ERRNO                      0000FE44
DSA$GL_EVENT                      0000FE48
DSA$GL_FLAGS                      0000FE00
DSA$GL_MSGTYP                     0000FE40
DSA$GL_PASSES                     0000FE08
DSA$GL_PASSNO                     0000FE54
DSA$GL_SECTNO                     0000FE10
DSA$GL_SID                        0000FE14
DSA$GL_SUBTNO                     0000FE4C
DSA$GL_TESTNO                     0000FE50
DSA$GL_UNITS                      0000FE0C
DSA$GQ_MSGPTR                     0000FE68
DSA$GT_DEVNAM                     0000FE5C
DSA$M_APT                       = 80000000
DSA$M_BELL                      = 00000008
DSA$M_BINARY                    = 00000002
DSA$M_CRD_AUTOTEST              = 00000800
DSA$M_CRD_MENUTEST              = 00010000
DSA$M_CRD_TRACE                 = 00020000
DSA$M_DEBUG                     = 04000000
DSA$M_HALT                      = 00000001
DSA$M_IE1                       = 00000010
DSA$M_IE2                       = 00000020
DSA$M_IE3                       = 00000040
DSA$M_IES                       = 00000080
DSA$M_LOAD_DEBUGGER             = 40000000
DSA$M_LOOP                      = 00000004
DSA$M_NORPT                     = 08000000
DSA$M_OPER                      = 00001000
DSA$M_PASS0                     = 20000000
DSA$M_PROMPT                    = 00002000
DSA$M_QA                        = 00008000
DSA$M_QUICK                     = 00000100
DSA$M_SEARCH                    = 00004000
DSA$M_TRACE                     = 00000400
DSA$M_USER                      = 10000000
DSA$M_VERIFY                    = 00000200
DSA$V_APT                       = 0000001F
DSA$V_BELL                      = 00000003
DSA$V_BINARY                    = 00000001
DSA$V_CRD_AUTOTEST              = 0000000B
DSA$V_CRD_MENUTEST              = 00000010
DSA$V_CRD_TRACE                 = 00000011
DSA$V_DEBUG                     = 0000001A
DSA$V_HALT                      = 00000000
DSA$V_IE1                       = 00000004
DSA$V_IE2                       = 00000005
DSA$V_IE3                       = 00000006
DSA$V_IES                       = 00000007
DSA$V_LOAD_DEBUGGER             = 0000001E
DSA$V_LOOP                      = 00000002
DSA$V_NORPT                     = 0000001B
DSA$V_OPER                      = 0000000C
DSA$V_PASS0                     = 0000001D
DSA$V_PROMPT                    = 0000000D
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
ERR$_MSGADR                     = 0000000C
ERR$_NARGS                      = 0000000A
ERR$_NUM                        = 00000004
ERR$_P1                         = 00000014
ERR$_P2                         = 00000018
ERR$_P3                         = 0000001C
ERR$_P4                         = 00000020
ERR$_P5                         = 00000024
ERR$_P6                         = 00000028
ERR$_POINTER                    = 00000010
ERR$_UNIT                       = 00000008
HALT                            = 00000004   G
HP$A_DEPENDENT                    00000032
HP$A_DEVICE                       00000018
HP$A_DVA                          0000001C
HP$A_LINK                         00000020
HP$B_DRIVE                        0000000B
HP$B_FLAGS                        0000000A
HP$M_ALLOC                      = 00000001
HP$M_WASALL                     = 00000002
HP$Q_DEVICE                       00000000
HP$T_DEVICE                       0000000C
HP$T_TYPE                         00000026
HP$V_ALLOC                      = 00000000
HP$V_WASALL                     = 00000001
HP$W_SIZE                         00000008
HP$W_VECTOR                       00000024
```

```
IPR                             = 00000006  G
KA$B_ACC_TYPE                     00000042
KA$B_RESERVED                     00000043
KA$B_SID_TYPE                     0000003D
KA$K_LEN                          00000046
KA$L_FLAGS                        00000032
KA$L_SID                          0000003A
KA$M_CHAR                       = 00000100
KA$M_CRC                        = 00000010
KA$M_D_FLOAT                    = 00000002
KA$M_EDIT                       = 00000080
KA$M_F_FLOAT                    = 00000001
KA$M_G_FLOAT                    = 00000004
KA$M_H_FLOAT                    = 00000008
KA$M_PACKED                     = 00000020
KA$M_PACK_MUL                   = 00000040
KA$M_SB_ERR                     = 02000000
KA$M_TODR                       = 00010000
KA$M_WCS_LD                     = 01000000
KA$V_CHAR                       = 00000008
KA$V_CRC                        = 00000004
KA$V_D_FLOAT                    = 00000001
KA$V_EDIT                       = 00000007
KA$V_F_FLOAT                    = 00000000
KA$V_G_FLOAT                    = 00000002
KA$V_H_FLOAT                    = 00000003
KA$V_PACKED                     = 00000005
KA$V_PACK_MUL                   = 00000006
KA$V_SB_ERR                     = 00000019
KA$V_TODR                       = 00000010
KA$V_WCS_LD                     = 00000018
KA$W_LATENCY                      0000003E
KA$W_MEM_SIZE                     00000044
KA$W_PROGRESS                     00000040
KA$W_RESERVED                     00000038
KA$W_WCS                          00000036
LDPCTX                          = 0000000B  G
MANUAL                          = 00000001  G
MCHK                            = 00000005  G
MEM                             = 00000009  G
MSG_005                           00000002  R       04
PAR$M_ATDEF                     = 00000010
PAR$M_ATHI                      = 00000008
PAR$M_ATLO                      = 00000004
PAR$M_NODEF                     = 00000002
PAR$V_ATDEF                     = 00000004
PAR$V_ATHI                      = 00000003
PAR$V_ATLO                      = 00000002
PAR$V_NODEF                     = 00000001
PAR$_BIN                        = 00000002
PAR$_DEC                        = 0000000A
PAR$_HEX                        = 00000010
PAR$_NO                         = 00000000
PAR$_OCT                        = 00000008
PAR$_YES                        = 00000001
POWER                           = 00000003  G
PSL$C_EXEC                      = 00000001
PSL$C_KERNEL                    = 00000000
PSL$C_SUPER                     = 00000002
PSL$C_USER                      = 00000003
PSL$K_EXEC                      = 00000001
PSL$K_KERNEL                    = 00000000
PSL$K_SUPER                     = 00000002
PSL$K_USER                      = 00000003
PSL$M_C                         = 00000001
PSL$M_CBIT                      = 00000001
PSL$M_CM                        = 80000000
PSL$M_CURMOD                    = 03000000
PSL$M_DV                        = 00000080
PSL$M_FPD                       = 08000000
PSL$M_FU                        = 00000040
PSL$M_IPL                       = 001F0000
PSL$M_IS                        = 04000000
PSL$M_IV                        = 00000020
PSL$M_N                         = 00000008
PSL$M_NBIT                      = 00000008
PSL$M_PRVMOD                    = 00C00000
PSL$M_SAFBITS                   = 000037FF
PSL$M_TBIT                      = 00000010
PSL$M_TP                        = 40000000
PSL$M_V                         = 00000002
PSL$M_VBIT                      = 00000002
PSL$M_Z                         = 00000004
PSL$M_ZBIT                      = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CBIT                      = 00000000
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_NBIT                      = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_VBIT                      = 00000001
PSL$V_Z                         = 00000002
PSL$V_ZBIT                      = 00000002
SCB$L_ACCESS                      00000020
SCB$L_ARITH                       00000034
SCB$L_BREAK                       0000002C
SCB$L_CHME                        00000044
SCB$L_CHMK                        00000040
SCB$L_CHMS                        00000048
SCB$L_CHMU                        0000004C
SCB$L_COMPAT                      00000030
```

```
SCB$L_KNLSTK                    00000008
SCB$L_MACHCHK                   00000004
SCB$L_OPCCUS                    00000014
SCB$L_OPCDEC                    00000010
SCB$L_POWER                     0000000C
SCB$L_RADRMOD                   0000001C
SCB$L_ROPRAND                   00000018
SCB$L_RXDB                      000000F8
SCB$L_SFTLVL1                   00000084
SCB$L_SFTLVL10                  000000A8
SCB$L_SFTLVL11                  000000AC
SCB$L_SFTLVL12                  000000B0
SCB$L_SFTLVL13                  000000B4
SCB$L_SFTLVL14                  000000B8
SCB$L_SFTLVL15                  000000BC
SCB$L_SFTLVL2                   00000088
SCB$L_SFTLVL3                   0000008C
SCB$L_SFTLVL4                   00000090
SCB$L_SFTLVL5                   00000094
SCB$L_SFTLVL6                   00000098
SCB$L_SFTLVL7                   0000009C
SCB$L_SFTLVL8                   000000A0
SCB$L_SFTLVL9                   000000A4
SCB$L_TBIT                      00000028
SCB$L_TIMER                     000000C0
SCB$L_TRANSL                    00000024
SCB$L_TXDB                      000000FC
SCB$L_ZERO                      00000000
SEP_FUNCTIONAL                = 00000002
SEP_REPAIR                    = 00000003
SIZ...                        = 00000001
SYS$ALLOC                       00010238
SYS$ASCTIM                      00010248
SYS$ASSIGN                      00010250
SYS$BINTIM                      00010258
SYS$CANCEL                      00010260
SYS$CANTIM                      00010268
SYS$CLOSE                       000105B8
SYS$CLREF                       00010298
SYS$CONNECT                     000105C0
SYS$DALLOC                      000102D8
SYS$DASSGN                      000102E0
SYS$DISCONNECT                  000105D0
SYS$FAO                         00010350
SYS$FAOL                        00010358
SYS$GET                         00010580
SYS$GETCHN                      000104C8
SYS$GETTIM                      00010378
SYS$LKWSET                      000103A0
SYS$NUMTIM                      000103B8
SYS$OPEN                        00010608
SYS$QIO                         000103C8
SYS$QIOW                        00010200
SYS$READ                        00010590
SYS$READEF                      000103D0
SYS$SETAST                      000103F8
SYS$SETEF                       00010400
SYS$SETIMR                      00010420
SYS$SETPRI                      00010428
SYS$SETPRT                      00010430
SYS$SETRWM                      00010438
SYS$SETSWM                      00010448
SYS$ULKPAG                      00010460
SYS$ULWSET                      00010468
SYS$UNWIND                      00010470
SYS$WAITFR                      00010478
SYS$WFLAND                      00010488
SYS$WFLOR                       00010490
T5_S1                           0000002E RG     04
T5_S1_X                         00000039 R      04
T5_S2                           00000044 RG     04
T5_S2_X                         0000004F R      04
T5_S3                           0000005A RG     04
T5_S3_X                         00000065 R      04
TEST_005                        0000002C RG     04
TEST_005_X                      00000073 RG     04
TU58                          = 00000002  G
UBI                           = 0000000A  G
WCS                           = 00000007  G
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation		PSECT No.		Attributes									
. ABS .	00000000	(0.)	00	(0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC BYTE
. BLANK .	00000000	(0.)	01	(1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
ENKAXE	00000000	(0.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC PAGE
ABS	00010610	(67088.)	03	(3.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
TEST_005	0000007B	(123.)	04	(4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC PAGE
DISPATCH	00000018	(24.)	05	(5.)	NOPIC	USR	CON	REL	LCL	NOSHR	NOEXE	RD	NOWRT	NOVEC LONG
ARGLIST	00000024	(36.)	06	(6.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC LONG
$TSTCNT	00000004	(4.)	07	(7.)	NOPIC	USR	OVR	REL	LCL	NOSHR	NOEXE	RD	NOWRT	NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000018-R	173 (11)	138 (9) 155 (10) 173 (11)
$$ARGS	=0000000A	78 (2)	78 (2)
$$E	=00000001	173 (11)	140 (9) 157 (10) 175 (11)
$$M	=00000181	120 (8)	120 (8)
$$N	=00000000	120 (8)	82 (2)
$$S	=FFFFFFFF	177 (11)	112 (8) 120 (8) 124 (9) 138 (9) 144 (10) 155 (10) 161 (11) 173 (11)
$$T1	=0000002C	78 (2)	78 (2)
$ENV	=00000001	61 (2)	
$ER	=FFFFFFFF	177 (11)	138 (9) 155 (10) 173 (11)
$MO	=00000001	179 (11)	179 (11)
S	=00000018-R	173 (11)	138 (9) 140 (9) 155 (10) 157 (10) 173 (11) 175 (11)
$ST	=00000000	177 (11)	120 (8) 124 (9) 138 (9) 140 (9) 144 (10) 155 (10) 157 (10) 161 (11) 173 (11) 175 (11) 177 (11)
$TN	=00000005	179 (11)	112 (8) 120 (8) 124 (9) 144 (10) 161 (11) 177 (11) 179 (11)
ALL	=00000008	82 (2)	120 (8)
BASE_005	00000000-R	112 (8)	120 (8)
BIT...	=0000001A	81 (2)	61 (2) 74 (2) 75 (2) 76 (2) 77 (2) 80 (2) 81 (2)
CEP_FUNCTIONAL	=00000000	61 (2)	
CEP_REPAIR	=00000001	61 (2)	61 (2)
CPU$V_COMET	=00000002	74 (2)	
CPU$V_HS	=00000000	74 (2)	
CPU$V_STAR	=00000001	74 (2)	
DATA_005	00000028-R	120 (8)	120 (8)
DEFAULT	=00000000	82 (2)	120 (8)
DS$ABORT	00010020	79 (2)	
DS$ASKADR	00010090	79 (2)	
DS$ASKDATA	00010080	79 (2)	
DS$ASKLGCL	00010098	79 (2)	
DS$ASKSTR	000100A0	79 (2)	
DS$ASKVLD	00010088	79 (2)	
DS$ATTACH	000101A8	79 (2)	
DS$BGNSUB	00010030	79 (2)	138 (9) 155 (10) 173 (11)
DS$BRANCH	000100A8	79 (2)	
DS$BREAK	00010058	79 (2)	177 (11)
DS$CANWAIT	00010070	79 (2)	
DS$CHANNEL	00010180	79 (2)	
DS$CKLOOP	00010040	79 (2)	
DS$CLRVEC	00010168	79 (2)	
DS$CNTRLC	00010078	79 (2)	
DS$CVTREG	000100B0	79 (2)	
DS$ENDPASS	00010010	79 (2)	
DS$ENDSUB	00010038	79 (2)	140 (9) 157 (10) 175 (11)
DS$ERRDEV	000100C8	79 (2)	
DS$ERRHARD	000100D0	79 (2)	
DS$ERRPREP	00010108	79 (2)	

```
DS$ERRSOFT                      000100D8      79      (2)
DS$ERRSYS                       000100C0      79      (2)
DS$ESCAPE                       00010050      79      (2)
DS$FREEDBGSYM                   000101B8      79      (2)
DS$GETBUF                       00010120      79      (2)
DS$GETMEM                       00010130      79      (2)
DS$GPHARD                       00010018      79      (2)
DS$HELP                         000101B0      79      (2)
DS$INITSCB                      00010170      79      (2)
DS$INLOOP                       00010048      79      (2)
DS$K_ERROR                     =00000002      75      (2)
DS$K_NORMAL                    =00000001      75      (2)
DS$K_SEVERE                    =00000004      75      (2)
DS$K_SUBSYS                    =00000066      75      (2)       75      (2)
DS$K_WARNING                   =00000000      75      (2)
DS$LOAD                         00010198      79      (2)
DS$LOADPCS                      000101C0      79      (2)
DS$MAPDBGBLOCK                  00010118      79      (2)
DS$MMOFF                        00010158      79      (2)
DS$MMON                         00010150      79      (2)
DS$MOVPHY                       00010148      79      (2)
DS$MOVVRT                       00010140      79      (2)
DS$PARSE                        000100B8      79      (2)
DS$PRINTB                       000100E0      79      (2)
DS$PRINTF                       000100F0      79      (2)
DS$PRINTS                       000100F8      79      (2)
DS$PRINTSIG                     00010100      79      (2)
DS$PRINTX                       000100E8      79      (2)
DS$PROBE                        000101A0      79      (2)
DS$RELBUF                       00010128      79      (2)
DS$RELMEM                       00010138      79      (2)
DS$SETIPL                       00010178      79      (2)
DS$SETMAP                       00010188      79      (2)
DS$SETPRIEXV                    00010110      79      (2)
DS$SETVEC                       00010160      79      (2)
DS$SHOCHAN                      00010190      79      (2)
DS$SUMMARY                      00010028      79      (2)
DS$WAITMS                       00010060      79      (2)
DS$WAITUS                       00010068      79      (2)
DS$_ARITH                      =006600D0      75      (2)
DS$_BADLINK                    =006600F0      75      (2)
DS$_BADTYPE                    =006600E8      75      (2)
DS$_CHME                       =006600A8      75      (2)
DS$_CHMK                       =006600E0      75      (2)
DS$_DEVNAME                    =00660108      75      (2)
DS$_ERROR                      =00660002      75      (2)
DS$_FHWE                       =00660068      75      (2)
DS$_FRAGBUF                    =00660080      75      (2)
DS$_ICBUSY                     =006600C8      75      (2)
DS$_ICERR                      =006600C0      75      (2)
DS$_IHWE                       =00660060      75      (2)
DS$_ILLCHAR                    =00660018      75      (2)
DS$_ILLPAGCNT                  =00660078      75      (2)
DS$_ILLUNIT                    =00660100      75      (2)
DS$_INSFMEM                    =00660050      75      (2)
DS$_IPL2HI                     =006600B8      75      (2)
DS$_IVADDR                     =00660040      75      (2)
```

```
DS$_IVVECT                      =00660038      75      (2)
DS$_KRNLSTK                     =00660090      75      (2)
DS$_LOGIC                       =00660070      75      (2)
DS$_MCHK                        =00660088      75      (2)
DS$_MMOFF                       =00660058      75      (2)
DS$_NEEDUNIT                    =006600F8      75      (2)
DS$_NOPCS                       =00660110      75      (2)
DS$_NORMAL                      =00660001      75      (2)
DS$_NOSUPPORT                   =006600B1      75      (2)
DS$_NOTDON                      =00660030      75      (2)
DS$_NOTIMP                      =006600B0      75      (2)       75      (2)
DS$_NULLSTR                     =00660010      75      (2)
DS$_OVERFLOW                    =00660008      75      (2)
DS$_POWER                       =00660098      75      (2)
DS$_PROGERR                     =00660020      75      (2)
DS$_SEVERE                      =00660004      75      (2)
DS$_TRANSL                      =006600A0      75      (2)
DS$_TRUNCATE                    =00660028      75      (2)
DS$_UNEXPINT                    =006600D8      75      (2)
DS$_VASFULL                     =00660048      75      (2)
DS$_WARNING                     =00660000      75      (2)       75      (2)
DSA$AL_APTMAIL                   0000FE00      74      (2)
DSA$AT_APTTXT                    0000FA00      74      (2)
DSA$GL_APTCOM                    0000FE04      74      (2)
DSA$GL_DEVLEN                    0000FE58      74      (2)
DSA$GL_ERRNO                     0000FE44      74      (2)
DSA$GL_EVENT                     0000FE48      74      (2)
DSA$GL_FLAGS                     0000FE00      74      (2)
DSA$GL_MSGTYP                    0000FE40      74      (2)
DSA$GL_PASSES                    0000FE08      74      (2)
DSA$GL_PASSNO                    0000FE54      74      (2)
DSA$GL_SECTNO                    0000FE10      74      (2)
DSA$GL_SID                       0000FE14      74      (2)
DSA$GL_SUBTNO                    0000FE4C      74      (2)
DSA$GL_TESTNO                    0000FE50      74      (2)
DSA$GL_UNITS                     0000FE0C      74      (2)
DSA$GQ_MSGPTR                    0000FE68      74      (2)
DSA$GT_DEVNAM                    0000FE5C      74      (2)
DSA$M_APT                       =80000000      74      (2)
DSA$M_BELL                      =00000008      74      (2)
DSA$M_BINARY                    =00000002      74      (2)
DSA$M_CRD_AUTOTEST              =00000800      74      (2)
DSA$M_CRD_MENUTEST              =00010000      74      (2)
DSA$M_CRD_TRACE                 =00020000      74      (2)
DSA$M_DEBUG                     =04000000      74      (2)
DSA$M_HALT                      =00000001      74      (2)
DSA$M_IE1                       =00000010      74      (2)
DSA$M_IE2                       =00000020      74      (2)
DSA$M_IE3                       =00000040      74      (2)
DSA$M_IES                       =00000080      74      (2)
DSA$M_LOAD_DEBUGGER             =40000000      74      (2)
DSA$M_LOOP                      =00000004      74      (2)
DSA$M_NORPT                     =08000000      74      (2)
DSA$M_OPER                      =00001000      74      (2)
DSA$M_PASSO                     =20000000      74      (2)
DSA$M_PROMPT                    =00002000      74      (2)
DSA$M_QA                        =00008000      74      (2)
```

```
ZZ-
ENK
1-3

64
63
6C
6D

72
6E
20
2E

72
75
21

64
69
70

20
72
20
21

6F
20
3D

2F

69
75
74
```

```
DSA$M_QUICK                     =00000100     74      (2)
DSA$M_SEARCH                    =00004000     74      (2)
DSA$M_TRACE                     =00000400     74      (2)
DSA$M_USER                      =10000000     74      (2)
DSA$M_VERIFY                    =00000200     74      (2)
DSA$V_APT                       =0000001F     74      (2)
DSA$V_BELL                      =00000003     74      (2)
DSA$V_BINARY                    =00000001     74      (2)
DSA$V_CRD_AUTOTEST              =0000000B     74      (2)
DSA$V_CRD_MENUTEST              =00000010     74      (2)
DSA$V_CRD_TRACE                 =00000011     74      (2)
DSA$V_DEBUG                     =0000001A     74      (2)
DSA$V_HALT                      =00000000     74      (2)
DSA$V_IE1                       =00000004     74      (2)
DSA$V_IE2                       =00000005     74      (2)
DSA$V_IE3                       =00000006     74      (2)
DSA$V_IES                       =00000007     74      (2)
DSA$V_LOAD_DEBUGGER             =0000001E     74      (2)
DSA$V_LOOP                      =00000002     74      (2)
DSA$V_NORPT                     =0000001B     74      (2)
DSA$V_OPER                      =0000000C     74      (2)
DSA$V_PASSO                     =0000001D     74      (2)
DSA$V_PROMPT                    =0000000D     74      (2)
DSA$V_QA                        =0000000F     74      (2)
DSA$V_QUICK                     =00000008     74      (2)
DSA$V_SEARCH                    =0000000E     74      (2)
DSA$V_TRACE                     =0000000A     74      (2)
DSA$V_USER                      =0000001C     74      (2)
DSA$V_VERIFY                    =00000009     74      (2)
ENV$M_DOMAIN                    =00000002     61      (2)
ENV$M_LEVEL                     =00000001     61      (2)
ENV$M_SUPER                     =000003FC     61      (2)
ENV$S_DOMAIN                    =00000001     61      (2)
ENV$S_LEVEL                     =00000001     61      (2)
ENV$S_SUPER                     =00000008     61      (2)
ENV$V_DOMAIN                    =00000001     61      (2)     61      (2)
ENV$V_LEVEL                     =00000000     61      (2)     61      (2)
ENV$V_SUPER                     =00000002     61      (2)
ENV$_CPU                        =00000000     61      (2)     61      (2)
ENV$_FUNCTIONAL                 =00000000     61      (2)     61      (2)
ENV$_REPAIR                     =00000001     61      (2)     61      (2)
ENV$_SUPER                      =00000001     61      (2)
ENV$_SYSTEM                     =00000001     61      (2)     61      (2)
ERR$_MSGADR                     =0000000C     78      (2)
ERR$_NARGS                      =0000000A     78      (2)
ERR$_NUM                        =00000004     78      (2)
ERR$_P1                         =00000014     78      (2)
ERR$_P2                         =00000018     78      (2)
ERR$_P3                         =0000001C     78      (2)
ERR$_P4                         =00000020     78      (2)
ERR$_P5                         =00000024     78      (2)
ERR$_P6                         =00000028     78      (2)
ERR$_POINTER                    =00000010     78      (2)
ERR$_UNIT                       =00000008     78      (2)
HALT                            =00000004     82      (2)
HP$A_DEPENDENT                   00000032     80      (2)     81      (2)
HP$A_DEVICE                      00000018     80      (2)
```

```
HP$A_DVA                         0000001C     80     (2)
HP$A_LINK                        00000020     80     (2)
HP$B_DRIVE                       0000000B     80     (2)
HP$B_FLAGS                       0000000A     80     (2)
HP$M_ALLOC                      =00000001     80     (2)
HP$M_WASALL                     =00000002     80     (2)
HP$Q_DEVICE                      00000000     80     (2)
HP$T_DEVICE                      0000000C     80     (2)
HP$T_TYPE                        00000026     80     (2)
HP$V_ALLOC                      =00000000     80     (2)
HP$V_WASALL                     =00000001     80     (2)
HP$W_SIZE                        00000008     80     (2)
HP$W_VECTOR                      00000024     80     (2)
IPR                             =00000006     82     (2)
KA$B_ACC_TYPE                    00000042     81     (2)
KA$B_RESERVED                    00000043     81     (2)
KA$B_SID_TYPE                    0000003D     81     (2)
KA$K_LEN                         00000046     81     (2)
KA$L_FLAGS                       00000032     81     (2)
KA$L_SID                         0000003A     81     (2)
KA$M_CHAR                       =00000100     81     (2)
KA$M_CRC                        =00000010     81     (2)
KA$M_D_FLOAT                    =00000002     81     (2)
KA$M_EDIT                       =00000080     81     (2)
KA$M_F_FLOAT                    =00000001     81     (2)
KA$M_G_FLOAT                    =00000004     81     (2)
KA$M_H_FLOAT                    =00000008     81     (2)
KA$M_PACKED                     =00000020     81     (2)
KA$M_PACK_MUL                   =00000040     81     (2)
KA$M_SB_ERR                     =02000000     81     (2)
KA$M_TODR                       =00010000     81     (2)
KA$M_WCS_LD                     =01000000     81     (2)
KA$V_CHAR                       =00000008     81     (2)
KA$V_CRC                        =00000004     81     (2)
KA$V_D_FLOAT                    =00000001     81     (2)
KA$V_EDIT                       =00000007     81     (2)
KA$V_F_FLOAT                    =00000000     81     (2)
KA$V_G_FLOAT                    =00000002     81     (2)
KA$V_H_FLOAT                    =00000003     81     (2)
KA$V_PACKED                     =00000005     81     (2)
KA$V_PACK_MUL                   =00000006     81     (2)
KA$V_SB_ERR                     =00000019     81     (2)
KA$V_TODR                       =00000010     81     (2)
KA$V_WCS_LD                     =00000018     81     (2)
KA$W_LATENCY                     0000003E     81     (2)
KA$W_MEM_SIZE                    00000044     81     (2)
KA$W_PROGRESS                    00000040     81     (2)
KA$W_RESERVED                    00000038     81     (2)
KA$W_WCS                         00000036     81     (2)
LDPCTX                          =0000000B     82     (2)
MANUAL                          =00000001     82     (2)
MCHK                            =00000005     82     (2)
MEM                             =00000009     82     (2)
MSG_005                          00000002-R   112    (8)       120    (8)
PAR$M_ATDEF                     =00000010     76     (2)
PAR$M_ATHI                      =00000008     76     (2)
PAR$M_ATLO                      =00000004     76     (2)
```

```
PAR$M_NODEF                     =00000002     76      (2)
PAR$V_ATDEF                     =00000004     76      (2)
PAR$V_ATHI                      =00000003     76      (2)
PAR$V_ATLO                      =00000002     76      (2)
PAR$V_NODEF                     =00000001     76      (2)
PAR$_BIN                        =00000002     76      (2)
PAR$_DEC                        =0000000A     76      (2)
PAR$_HEX                        =00000010     76      (2)
PAR$_NO                         =00000000     76      (2)
PAR$_OCT                        =00000008     76      (2)
PAR$_YES                        =00000001     76      (2)
POWER                           =00000003     82      (2)
PSL$C_EXEC                      =00000001     77      (2)
PSL$C_KERNEL                    =00000000     77      (2)
PSL$C_SUPER                     =00000002     77      (2)
PSL$C_USER                      =00000003     77      (2)
PSL$K_EXEC                      =00000001     77      (2)
PSL$K_KERNEL                    =00000000     77      (2)
PSL$K_SUPER                     =00000002     77      (2)
PSL$K_USER                      =00000003     77      (2)
PSL$M_C                         =00000001     77      (2)
PSL$M_CBIT                      =00000001     77      (2)
PSL$M_CM                        =80000000     77      (2)       77      (2)
PSL$M_CURMOD                    =03000000     77      (2)
PSL$M_DV                        =00000080     77      (2)
PSL$M_FPD                       =08000000     77      (2)       77      (2)
PSL$M_FU                        =00000040     77      (2)
PSL$M_IPL                       =001F0000     77      (2)
PSL$M_IS                        =04000000     77      (2)
PSL$M_IV                        =00000020     77      (2)
PSL$M_N                         =00000008     77      (2)
PSL$M_NBIT                      =00000008     77      (2)
PSL$M_PRVMOD                    =00C00000     77      (2)
PSL$M_SAFBITS                   =000037FF     77      (2)
PSL$M_TBIT                      =00000010     77      (2)
PSL$M_TP                        =40000000     77      (2)       77      (2)
PSL$M_V                         =00000002     77      (2)
PSL$M_VBIT                      =00000002     77      (2)
PSL$M_Z                         =00000004     77      (2)
PSL$M_ZBIT                      =00000004     77      (2)
PSL$S_CURMOD                    =00000002     77      (2)
PSL$S_IPL                       =00000005     77      (2)
PSL$S_PRVMOD                    =00000002     77      (2)
PSL$V_C                         =00000000     77      (2)
PSL$V_CBIT                      =00000000     77      (2)
PSL$V_CM                        =0000001F     77      (2)
PSL$V_CURMOD                    =00000018     77      (2)
PSL$V_DV                        =00000007     77      (2)
PSL$V_FPD                       =0000001B     77      (2)
PSL$V_FU                        =00000006     77      (2)
PSL$V_IPL                       =00000010     77      (2)
PSL$V_IS                        =0000001A     77      (2)
PSL$V_IV                        =00000005     77      (2)
PSL$V_N                         =00000003     77      (2)
PSL$V_NBIT                      =00000003     77      (2)
PSL$V_PRVMOD                    =00000016     77      (2)
PSL$V_TBIT                      =00000004     77      (2)
```

```
PSL$V_TP                        =0000001E     77     (2)
PSL$V_V                         =00000001     77     (2)
PSL$V_VBIT                      =00000001     77     (2)
PSL$V_Z                         =00000002     77     (2)
PSL$V_ZBIT                      =00000002     77     (2)
SCB$L_ACCESS                     00000020     73     (2)
SCB$L_ARITH                      00000034     73     (2)
SCB$L_BREAK                      0000002C     73     (2)
SCB$L_CHME                       00000044     73     (2)
SCB$L_CHMK                       00000040     73     (2)
SCB$L_CHMS                       00000048     73     (2)
SCB$L_CHMU                       0000004C     73     (2)
SCB$L_COMPAT                     00000030     73     (2)
SCB$L_KNLSTK                     00000008     73     (2)
SCB$L_MACHCHK                    00000004     73     (2)
SCB$L_OPCCUS                     00000014     73     (2)
SCB$L_OPCDEC                     00000010     73     (2)
SCB$L_POWER                      0000000C     73     (2)
SCB$L_RADRMOD                    0000001C     73     (2)
SCB$L_ROPRAND                    00000018     73     (2)
SCB$L_RXDB                       000000F8     73     (2)
SCB$L_SFTLVL1                    00000084     73     (2)
SCB$L_SFTLVL10                   000000A8     73     (2)
SCB$L_SFTLVL11                   000000AC     73     (2)
SCB$L_SFTLVL12                   000000B0     73     (2)
SCB$L_SFTLVL13                   000000B4     73     (2)
SCB$L_SFTLVL14                   000000B8     73     (2)
SCB$L_SFTLVL15                   000000BC     73     (2)
SCB$L_SFTLVL2                    00000088     73     (2)
SCB$L_SFTLVL3                    0000008C     73     (2)
SCB$L_SFTLVL4                    00000090     73     (2)
SCB$L_SFTLVL5                    00000094     73     (2)
SCB$L_SFTLVL6                    00000098     73     (2)
SCB$L_SFTLVL7                    0000009C     73     (2)
SCB$L_SFTLVL8                    000000A0     73     (2)
SCB$L_SFTLVL9                    000000A4     73     (2)
SCB$L_TBIT                       00000028     73     (2)
SCB$L_TIMER                      000000C0     73     (2)
SCB$L_TRANSL                     00000024     73     (2)
SCB$L_TXDB                       000000FC     73     (2)
SCB$L_ZERO                       00000000     73     (2)
SEP_FUNCTIONAL                  =00000002     61     (2)
SEP_REPAIR                      =00000003     61     (2)
SIZ...                          =00000001     81     (2)    61 (2)  74 (2)  76 (2)  77 (2)
                                                            80 (2)  81 (2)
SYS$ALLOC                        00010238     79     (2)
SYS$ASCTIM                       00010248     79     (2)
SYS$ASSIGN                       00010250     79     (2)
SYS$BINTIM                       00010258     79     (2)
SYS$CANCEL                       00010260     79     (2)
SYS$CANTIM                       00010268     79     (2)
SYS$CLOSE                        000105B8     79     (2)
SYS$CLREF                        00010298     79     (2)
SYS$CONNECT                      000105C0     79     (2)
SYS$DALLOC                       000102D8     79     (2)
SYS$DASSGN                       000102E0     79     (2)
SYS$DISCONNECT                   000105D0     79     (2)
```

```
SYS$FAO                         00010350      79      (2)
SYS$FAOL                        00010358      79      (2)
SYS$GET                         00010580      79      (2)
SYS$GETCHN                      000104C8      79      (2)
SYS$GETTIM                      00010378      79      (2)
SYS$LKWSET                      000103A0      79      (2)
SYS$NUMTIM                      000103B8      79      (2)
SYS$OPEN                        00010608      79      (2)
SYS$QIO                         000103C8      79      (2)
SYS$QIOW                        00010200      79      (2)
SYS$READ                        00010590      79      (2)
SYS$READEF                      000103D0      79      (2)
SYS$SETAST                      000103F8      79      (2)
SYS$SETEF                       00010400      79      (2)
SYS$SETIMR                      00010420      79      (2)
SYS$SETPRI                      00010428      79      (2)
SYS$SETPRT                      00010430      79      (2)
SYS$SETRWM                      00010438      79      (2)
SYS$SETSWM                      00010448      79      (2)
SYS$ULKPAG                      00010460      79      (2)
SYS$ULWSET                      00010468      79      (2)
SYS$UNWIND                      00010470      79      (2)
SYS$WAITFR                      00010478      79      (2)
SYS$WFLAND                      00010488      79      (2)
SYS$WFLOR                       00010490      79      (2)
T5_S1                           0000002E-R    138     (9)
T5_S1_X                         00000039-R    140     (9)
T5_S2                           00000044-R    155     (10)
T5_S2_X                         0000004F-R    157     (10)
T5_S3                           0000005A-R    173     (11)
T5_S3_X                         00000065-R    175     (11)
TEST_005                        0000002C-R    120     (8)       120     (8)
TEST_005_X                      00000073-R    177     (11)
TU58                            =00000002     82      (2)
UBI                             =0000000A     82      (2)
WCS                             =00000007     82      (2)       120     (8)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_BSUBT	1	138 (9)	138 (9) 155 (10) 173 (11)
$D1_BTEST	1	120 (8)	120 (8)
$D1_ESUBT	1	140 (9)	140 (9) 157 (10) 175 (11)
$D1_ETEST	1	177 (11)	177 (11)
$D1_SBTTL	2	112 (8)	112 (8) 124 (9) 144 (10) 161 (11)
$D2_BDATA	1	120 (8)	120 (8)
$D2_BTEST	1	120 (8)	120 (8)
$D2_SBTTL	1	112 (8)	112 (8)
$DEF	1	81 (2)	73 (2) 74 (2) 79 (2) 80 (2) 81 (2)
$DEFEND	1	73 (2)	73 (2) 74 (2) 77 (2) 79 (2) 80 (2)
			81 (2)
$DEFINI	1	73 (2)	73 (2) 74 (2) 77 (2) 79 (2) 80 (2)
			81 (2)
$DS_BGNMOD	1	61 (2)	61 (2)
$DS_BGNSUB	1	138 (9)	138 (9) 155 (10) 173 (11)
$DS_BGNTEST	1	120 (8)	120 (8)
$DS_BREAK	1	177 (11)	177 (11)
$DS_DSADEF	1	74 (2)	74 (2)
$DS_DSBDEF	1	74 (2)	
$DS_DSDEF	2	75 (2)	75 (2)
$DS_DSSDEF	1	79 (2)	79 (2)
$DS_ENDDATA	1	120 (8)	120 (8)
$DS_ENDMOD	1	179 (11)	179 (11)
$DS_ENDSUB	1	140 (9)	140 (9) 157 (10) 175 (11)
$DS_ENDTEST	1	177 (11)	177 (11)
$DS_ENVDEF	2	61 (2)	61 (2)
$DS_ERRDEF	1	78 (2)	78 (2)
$DS_HPODEF	2	80 (2)	80 (2)
$DS_KA_DEF	3	81 (2)	81 (2)
$DS_PAGE	1	110 (7)	110 (7) 122 (8) 142 (9) 159 (10)
$DS_PARDEF	1	76 (2)	76 (2)
$DS_PSLDEF	1	77 (2)	77 (2)
$DS_SBTTL	1	112 (8)	112 (8) 124 (9) 144 (10) 161 (11)
$DS_SCBDEF	1	73 (2)	73 (2)
$DS_SECDEF	1	82 (2)	82 (2)
$EQU	1	81 (2)	61 (2) 75 (2) 76 (2) 77 (2)
$EQULST	1	77 (2)	61 (2) 75 (2) 77 (2)
$EQULST	1	61 (2)	61 (2) 75 (2) 77 (2)
$GBLINI	2	61 (2)	61 (2) 73 (2) 74 (2) 75 (2) 76 (2)
			77 (2) 79 (2) 80 (2) 81 (2)
$HP_DEFDEF	1	80 (2)	
$KADEF	1	81 (2)	
$OFFDEF	1	78 (2)	78 (2)
$PSLDEF	1	77 (2)	77 (2)
$VIELD	1	61 (2)	61 (2) 74 (2) 76 (2) 77 (2) 80 (2)
			81 (2)
$VIELD1	1	81 (2)	61 (2) 74 (2) 76 (2) 77 (2) 80 (2)
			81 (2)
ENKAX_SECDEF	1	82 (2)	82 (2)

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.04	00:00:00.23
Command processing	16	00:00:00.18	00:00:00.59
Pass 1	688	00:00:10.86	00:00:16.16
Symbol table sort	2	00:00:00.55	00:00:00.57
Pass 2	116	00:00:01.40	00:00:01.66
Symbol table output	56	00:00:00.32	00:00:00.36
Psect synopsis output	5	00:00:00.04	00:00:00.06
Cross-reference output	150	00:00:01.61	00:00:02.50
Assembler run totals	1063	00:00:15.02	00:00:22.17

The working set limit was 900 pages.
63383 bytes (124 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 421 non-local and 0 local symbols.
180 source lines were read in Pass 1, producing 22 object records in Pass 2.
106 pages of virtual memory were used to define 37 macros.

! Macro library statistics !

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	29
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	7
TOTALS (all libraries)	37

734 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXE

Table of contents

```
                        .SBTTL  ENKAXF: MACHINE CHECK EXCEPTIONS
                        .TITLE  ENKAXF  VAX-11/730 Specific CPU Cluster Exerciser
                        .IDENT  /1-3/

;
; Copyright (C) 1981
; Digital Equipment Corporation, Maynard, Massachusetts 01754
;
; This software is furnished  under a license for use only on a single
; computer  system and  may be  copied only with  the inclusion of the
; above copyright notice.  This software, or any other copies thereof,
; may not be provided or  otherwise made available to any other person
; except for use on such system and to one who agrees to these license
; terms.  Title to and  ownership of the  software  shall at all times
; remain in DEC.
;
; The information in this software is subject to change without notice
; and should  not be construed  as a commitment  by Digital  Equipment
; Corporation.
;
; DEC assumes  no  responsibility  for  the use or  reliability of its
; software on equipment which is not supplied by DEC.
;

;++
; Facility:     VAX Architecture Diagnostic.
;
; Abstract:     This module tests some of the possible ways in which a machine
;               check exception may occure.
;
; Environment:  VAX Diagnostic Supervisor.
;
; Author:       Rich Lewis      7-Jly-81        Version 1.0
;
;               Tai-Chun Pan    01-Apr-83       Version 1.2
;
;               Added quick flag on Test 7(TU58). If quick flag is set
;               will skip subtest 4 through subtest 11.  Added event flag 3.
;               If event flag 3 is set, will override protection,
;               i.e., go ahead and write on tape.  If run APT & event flag 3
;               is clear & tape is not scratch, will report an error.
;               Modified Unibus sizer on Test 8 & 9. make test run faster.
;
;
;
;               Tai-Chun Pan    03-Aug-83       Version 1.3
;
;               fixed bugs on TEST 7 and error summary routine call when
;               running under APT.
;
;--
```

```
                     0000     53          .SBTTL  DECLARATIONS
                     0000     54          .LIST   MEB
                     0000     55          .NLIST  CND
                 00000000     56          .PSECT  ENKAXF, PAGE, NOWRT
                     0000     57          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     58          $DS_BGNMOD      CEP_REPAIR, TN=5
                     0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                     0000     59
                     0000     60 ;
                     0000     61 ; Include files:
                     0000     62 ;
                     0000     63
                     0000     64          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                     0000     65
                     0000     66
                     0000     70
                     0000     71          $DS_SCBDEF
                     0000     72          $DS_DSDEF
                     0000     73          $DS_DSADEF
                     0000     74          $DS_DSSDEF
                     0000     75          $DS_ERRDEF
                     0000     76          $DS_PARDEF
                     0000     77          $DS_PSLDEF
                     0000     78          $DS_HPODEF
                     0000     79          $DS_KA_DEF
                     0000     80          ENKAX_SECDEF
                     0000     81
                     0000     82 ;
                     0000     83 ; Own storage:
                     0000     84 ;
                     0000     85
                     0000     86
 00000004            0000     87 L_COUNT:         .BLKL                   ; used as a counter
 00000008            0004     88 L_ADDRESS:       .BLKL                   ; used to temporary store addresses
 0000000C            0008     89 L_CUR_ISP:       .BLKL                   ; used to store initial ISP
 00000010            000C     90 RENTRY_POINT:    .BLKL                   ; used to store rentry address
 00000014            0010     91 L_EXP_MOD:       .BLKL                   ; temp storage of expected mode
 00000018            0014     92 L_EXP_STACK:     .BLKL                   ; temp storage of expected stack
 0000001C            0018     93                  .BLKL
 00000020            001C     94 L_TIMER:         .BLKL                   ; used as a timer for interrupts
                     0020     95
 00000024            0020     96 L_FLAGS:         .BLKL                   ; contains the following flags
                     0024     97                                          ; bit positions and masks
 00000000            0024     98 V_READY=0                                ; ready to execute flag
 00000001            0024     99 M_READY=1@0                              ;
 00000001            0024    100 V_DONE=1                                 ; restart done flag
 00000002            0024    101 M_DONE=1@1                               ;
 00000002            0024    102 V_EXECUTE_ERR=2                          ; execution error flag
 00000004            0024    103 M_EXECUTE_ERR=1@2                        ;
 00000003            0024    104 V_TIMER=3                                ; timer interrupt flag
 00000008            0024    105 M_TIMER=1@3                              ;
 00000004            0024    106 V_FATAL_ERR=4                            ; flag fatal error
 00000010            0024    107 M_FATAL_ERR=1@4                          ;
 00000005            0024    108 V_PSL_ERR=5                              ; flag PSL error
 00000020            0024    109 M_PSL_ERR=1@5                            ;
 00000006            0024    110 V_GPR_ERR=6                              ; flag GPR error
 00000040            0024    111 M_GPR_ERR=1@6                            ;
```

```
                    00000007  0024    112 V_ERROR_FLAG=7                          ; flag any error
                    00000080  0024    113 M_ERROR_FLAG=1@7                        ;
                    00000009  0024    114 V_ISP_ERR=9                             ; flag ISP error
                    00000200  0024    115 M_ISP_ERR=1@9                           ;
                    0000000A  0024    116 V_SP_ERR=10                             ; flag SP error
                    00000400  0024    117 M_SP_ERR=1@10                           ;
                    0000000C  0024    118 V_MC_ERR=12                             ; flag machine check error
                    00001000  0024    119 M_MC_ERR=1@12                           ;
                    0000000D  0024    120 V_INT=13                                ; flag interrupt
                    00002000  0024    121 M_INT=1@13                              ;
                    0000000E  0024    122 V_MCINT=14                              ; flag machine check interrupt
                    00004000  0024    123 M_MCINT=1@14                            ;
                              0024    124
                    00080000  0024    125 M_BIT19=1@19
                    00100000  0024    126 M_BIT20=1@20
                    00000080  0024    127 BUFSIZ=128                              ; buffer size
                    00500200  0024    128 A_NXM=^X500200                          ; address for nonexistent memory
                    00F00001  0024    129 A_UNAL_IO=^XF00001                      ; address for unaligned I/O reference
                    00F2A000  0024    130 A_ILL_IO=^XF2A000                       ; address for illegal I/O reference
                    00FC0000  0024    131 A_ILL_UB=^XFC0000                       ; address for illegal UNIBUS reference
                    00000000  0024    132 L_DATA=0                                ; data used to write nonexistent mem.
                              0024    133
                              0024    134
                    00F00000  0024    135 A_WCS_ADDRESS:  .LONG   ^XF00000        ; starting address of wcs
                              0028    136
                              0028    137
                              0028    141
```

```
                                        0028    143 .SBTTL  ERROR MESSAGES
                                        0028    144 ;-----------------------------------------------------------------------
                                        0028    145 T_NO_KA730:
69 76 65 64 20 6F 4E 20 20 2F 21 00'    0028    146         .ASCIC  "!/  No devices of type KA730 selected.  Exiting test.!/"
20 65 70 79 74 20 66 6F 20 73 65 63     0034
74 63 65 6C 65 73 20 30 33 37 41 4B     0040
67 6E 69 74 69 78 45 20 20 2E 64 65     004C
              2F 21 2E 74 73 65 74 20   0058
                                    37  0028
                                        0060    147 ;-----------------------------------------------------------------------
                                        0060    148 T_MC_MESS:
20 64 65 74 70 6D 65 74 74 41 20 00'    0060    149         .ASCIC  " Attempted to cause a Machine Check by!/"
4D 20 61 20 65 73 75 61 63 20 6F 74     006C
6B 63 65 68 43 20 65 6E 69 68 63 61     0078
                  2F 21 79 62 20        0084
                                    28  0060
                                        0089    150 ;-----------------------------------------------------------------------
                                        0089    151 T_BUSERR:
20 67 6E 69 73 73 65 63 63 61 20 00'    0089    152         .ASCIC  " accessing a nonexistent memory location!/!/"
6E 65 74 73 69 78 65 6E 6F 6E 20 61     0095
63 6F 6C 20 79 72 6F 6D 65 6D 20 74     00A1
           2F 21 2F 21 6E 6F 69 74 61   00AD
                                    2C  0089
                                        00B6    153 ;-----------------------------------------------------------------------
                                        00B6    154 T_UNALIO_ERR:
20 67 6E 69 73 73 65 63 63 61 20 00'    00B6    155         .ASCIC  " accessing an unaligned I/O address!/!/"
64 65 6E 67 69 6C 61 6E 75 20 6E 61     00C2
73 73 65 72 64 64 61 20 4F 2F 49 20     00CE
                     2F 21 2F 21        00DA
                                    27  00B6
                                        00DE    156 ;-----------------------------------------------------------------------
                                        00DE    157 T_ILLIO_ERR:
20 67 6E 69 73 73 65 63 63 61 20 00'    00DE    158         .ASCIC  " accessing an illegal I/O address!/!/"
49 20 6C 61 67 65 6C 6C 69 20 6E 61     00EA
2F 21 73 73 65 72 64 64 61 20 4F 2F     00F6
                              2F 21     0102
                                    25  00DE
                                        0104    159 ;-----------------------------------------------------------------------
                                        0104    160 T_ILLUB_ERR:
20 67 6E 69 73 73 65 63 63 61 20 00'    0104    161         .ASCIC  " accessing an illegal UNIBUS address!/!/"
55 20 6C 61 67 65 6C 6C 69 20 6E 61     0110
73 65 72 64 64 61 20 53 55 42 49 4E     011C
                     2F 21 2F 21 73     0128
                                    28  0104
                                        012D    162 ;-----------------------------------------------------------------------
                                        012D    163 T_MC_ERR:
6D 20 6E 69 20 72 6F 72 72 45 20 00'    012D    164         .ASCIC  " Error in machine check logout stack!/"-
6B 63 65 68 63 20 65 6E 69 68 63 61     0139
63 61 74 73 20 74 75 6F 67 6F 6C 20     0145
21 3A 4C 41 55 54 43 41 20 2F 21 6B     0151
                                 2F     015D
                                    30  012D
                                        015E    165                     " ACTUAL:!/"
                                        015E    166 ;-----------------------------------------------------------------------
                                        015E    167 T_EXPECTED:
21 3A 44 45 54 43 45 50 58 45 20 00'    015E    168         .ASCIC  " EXPECTED:!/"
                                    2F  016A
```

```
                                        0C  015E
                                            016B   169 ;-----------------------------------------------------------------------
                                            016B   170 T_UNEXP_DATA:
64 65 74 63 65 70 78 65 6E 55 20 00'        016B   171         .ASCIC  " Unexpected data in machine check logout information"
63 61 6D 20 6E 69 20 61 74 61 64 20         0177
6C 20 6B 63 65 68 63 20 65 6E 69 68         0183
6D 72 6F 66 6E 69 20 74 75 6F 67 6F         018F
                     6E 6F 69 74 61         019B
                                 34         016B
                                            01A0   172 ;-----------------------------------------------------------------------
                                            01A0   173 T_NOT_READY:
72 74 73 6E 69 20 74 73 65 54 20 00'        01A0   174         .ASCIC  " Test instruction was not ready to be executed.!/"
6E 20 73 61 77 20 6E 6F 69 74 63 75         01AC
20 6F 74 20 79 64 61 65 72 20 74 6F         01B8
2E 64 65 74 75 63 65 78 65 20 65 62         01C4
                              2F 21         01D0
                                 31         01A0
                                            01D2   175 ;-----------------------------------------------------------------------
                                            01D2   176 T_EXECUTE:
72 74 73 6E 69 20 74 73 65 54 20 00'        01D2   177         .ASCIC  " Test instruction execution error.!/"
75 63 65 78 65 20 6E 6F 69 74 63 75         01DE
21 2E 72 6F 72 72 65 20 6E 6F 69 74         01EA
                                 2F         01F6
                                 24         01D2
                                            01F7   178 ;-----------------------------------------------------------------------
                                            01F7   179 T_TIMER:
64 65 74 63 65 70 78 65 6E 55 20 00'        01F7   180         .ASCIC  " Unexpected interval timer interrupt.!/"
69 74 20 6C 61 76 72 65 74 6E 69 20         0203
70 75 72 72 65 74 6E 69 20 72 65 6D         020F
                              2F 21 2E 74   021B
                                 27         01F7
                                            021F   181 ;-----------------------------------------------------------------------
                                            021F   182 T_ISP_ERR:
20 74 70 75 72 72 65 74 6E 49 20 00'        021F   183         .ASCIC  " Interrupt SP is incorrect - EXP = !XL  ACT = !XL!/"
72 72 6F 63 6E 69 20 73 69 20 50 53         022B
20 3D 20 50 58 45 20 2D 20 74 63 65         0237
21 20 3D 20 54 43 41 20 20 4C 58 21         0243
                              2F 21 4C 58   024F
                                 33         021F
                                            0253   184 ;-----------------------------------------------------------------------
                                            0253   185 T_SP_ERR1:
6F 63 6E 69 20 73 69 20 50 53 20 00'        0253   186         .ASCIC  " SP is incorrect - EXP = !XL  ACT = !XL!/"
20 50 58 45 20 2D 20 74 63 65 72 72         025F
3D 20 54 43 41 20 20 4C 58 21 20 3D         026B
                        2F 21 4C 58 21 20   0277
                                 29         0253
                                            027D   187 ;-----------------------------------------------------------------------
                                            027D   188 T_SP_ERR2:
2F 21 4C 58 21 20 3D 20 50 53 20 00'        027D   189         .ASCIC  " SP = !XL!/"
                                 0B         027D
                                            0289   190 ;-----------------------------------------------------------------------
                                            0289   191 T_GPR_ERR:
69 20 61 74 61 64 20 64 61 42 20 00'        0289   192         .ASCIC  " Bad data in general purpose register:!/"
75 70 20 6C 61 72 65 6E 65 67 20 6E         0295
74 73 69 67 65 72 20 65 73 6F 70 72         02A1
                        2F 21 3A 72 65      02AD
                                 28         0289
```

```
                                        02B2    193 ;-----------------------------------------------------------------------
                                        02B2    194 T_BAD_DATA:
58 21 20 3D 20 50 58 45 20 20 20 00'    02B2    195         .ASCIC  "   EXP = !XL    ACT = !XL    XOR= !XL !/"
21 20 3D 20 54 43 41 20 20 20 20 4C     02BE
21 20 3D 52 4F 58 20 20 20 20 4C 58     02CA
                  2F 21 20 4C 58 28     02D6
                                        02B2
                                        02DB    196 ;-----------------------------------------------------------------------
                                        02DB    197 T_GOOD_DATA:
58 21 20 3D 20 50 58 45 20 20 20 00'    02DB    198         .ASCIC  "   EXP = !XL    ACT = !XL !/"
21 20 3D 20 54 43 41 20 20 20 20 4C     02E7
                  2F 21 20 4C 58 1C     02F3
                                        02DB
                                        02F8    199 ;-----------------------------------------------------------------------
                                        02F8    200 T_NOT_ON_KS:
72 65 4B 20 6E 6F 20 74 6F 4E 20 00'    02F8    201         .ASCIC  " Not on Kernel Stack as expected.!/"
73 61 20 6B 63 61 74 53 20 6C 65 6E     0304
2F 21 2E 64 65 74 63 65 70 78 65 20     0310
                              23        02F8
                                        031C    202 ;-----------------------------------------------------------------------
                                        031C    203 T_NOT_ON_IS:
74 6E 49 20 6E 6F 20 74 6F 4E 20 00'    031C    204         .ASCIC  " Not on Interrupt Stack as expected.!/"
6B 63 61 74 53 20 74 70 75 72 72 65     0328
64 65 74 63 65 70 78 65 20 73 61 20     0334
                        2F 21 2E 26     0340
                                        031C
                                        0343    205 ;-----------------------------------------------------------------------
                                        0343    206 T_EXP_PSL:
50 20 64 65 74 63 65 70 78 45 20 00'    0343    207         .ASCIC  " Expected PSL = !_!XL(X) ; !AC!/"
58 28 4C 58 21 5F 21 20 3D 20 4C 53     034F
           2F 21 43 41 21 20 3B 20 29   035B
                                 20     0343
                                        0364    208 ;-----------------------------------------------------------------------
                                        0364    209 T_ACT_PSL:
4C 53 50 20 6C 61 75 74 63 41 20 00'    0364    210         .ASCIC  " Actual PSL = !_!XL(X) ; !AC!/"
20 29 58 28 4C 58 21 5F 21 20 3D 20     0370
              2F 21 43 41 21 20 3B      037C
                                 1E     0364
                                        0383    211 ;-----------------------------------------------------------------------
                                        0383    212 T_PSL:
58 21 5F 21 20 3D 20 4C 53 50 20 00'    0383    213         .ASCIC  " PSL = !_!XL (X) ; !AC!/"
21 43 41 21 20 3B 20 29 58 28 20 4C     038F
                                 2F     039B
                                 18     0383
                                        039C    214 ;-----------------------------------------------------------------------
                                        039C    215 T_GPR:
20 20 72 65 74 73 69 67 65 52 20 00'    039C    216         .ASCIC  " Register  Expected    Actual     XOR!/"
20 20 20 20 64 65 74 63 65 70 78 45     03A8
58 20 20 20 20 20 6C 61 75 74 63 41     03B4
                        2F 21 52 4F     03C0
                                 27     039C
                                        03C4    217 ;-----------------------------------------------------------------------
                                        03C4    218 T_XOR:
21 20 20 5F 21 43 41 21 20 20 20 00'    03C4    219         .ASCIC  "   .AC!_  !XL   !XL   !XL!/"
21 20 20 20 4C 58 21 20 20 20 4C 58     03D0
                        2F 21 4C 58     03DC
                                 1B     03C4
```

```
                                                03E0   220 ;-----------------------------------------------------------------------------
                                                03E0   221 T_NOXOR:
21 20 20 5F 21 43 41 21 20 20 20 00'            03E0   222         .ASCIC  "   !AC!_  !XL   !XL!/"
       2F 21 4C 58 21 20 20 20 4C 58            03EC
                                    15          03E0
                                                03F6   223 ;-----------------------------------------------------------------------------
                                                03F6   224 T_NO_WCS:
57 20 67 6E 69 74 69 78 45 2F 21 00'            03F6   225         .ASCIC  "!/Exiting WCS parity subtest, WCS last address = 0.!/"
75 73 20 79 74 69 72 61 70 20 53 43             0402
6C 20 53 43 57 20 2C 74 73 65 74 62             040E
20 73 73 65 72 64 64 61 20 74 73 61             041A
                  2F 21 2E 30 20 3D             0426
                                    35          03F6
                                                042C   226 ;-----------------------------------------------------------------------------
                                                042C   227
                                                042C   228
                                                042C   229
```

```
                        042C    231 .SBTTL  DATA TABLES
                        042C    232
                        042C    233 ; Data table for test 1
                        042C    234
                00000000 042C   235 DATA_1: .LONG   ^X0              ; expected PSL
                        0430    236
                        0430    237
```

```
                        0430   239
                        0430   240
                        0430   241
                        0430   242
00000007                0430   243 TABLE1: .LONG   7                ; length for printsig
00660088                0434   244         .LONG   DS$_MCHK         ; type of exception = machine check
0000000C                0438   245         .LONG   ^XC              ; count
00000008                043C   246         .LONG   ^X8              ; machine check type = ref. to NXM
00500200                0440   247         .LONG   A_NXM            ; NXM address
00000000                0444   248         .LONG   ^X0              ; error parameter 2
000000BB'               0448   249         .LONG   TAG_1            ; PC
00000000                044C   250         .LONG   ^X0              ; PSL
                        0450   251
                        0450   252
00000007                0450   253 TABLE2: .LONG   7                ; length for printsig
00660088                0454   254         .LONG   DS$_MCHK         ; type of exception = machine check
0000000C                0458   255         .LONG   ^XC              ; count
00000009                045C   256         .LONG   ^X9              ; machine check type = unaligned IO ref.
00F00001                0460   257         .LONG   A_UNAL_IO        ; unaligned IO address
00000000                0464   258         .LONG   ^X0              ; errcr parameter 2
0000019A'               0468   259         .LONG   TAG_2            ; PC
00000000                046C   260         .LONG   ^X0              ; PSL
                        0470   261
                        0470   262
00000007                0470   263 TABLE3: .LONG   7                ; length for printsig
00660088                0474   264         .LONG   DS$_MCHK         ; type of exception = machine check
0000000C                0478   265         .LONG   ^XC              ; count
0000000A                047C   266         .LONG   ^XA              ; machine check type = illegal IO ref.
00F2A000                0480   267         .LONG   A_ILL_IO         ; illegal IO address
00000000                0484   268         .LONG   ^X0              ; error parameter 2
00000279'               0488   269         .LONG   TAG_3            ; PC
00000000                048C   270         .LONG   ^X0              ; PSL
                        0490   271
                        0490   272
00000007                0490   273 TABLE4: .LONG   7                ; length for printsig
00660088                0494   274         .LONG   DS$_MCHK         ; type of exception = machine check
0000000C                0498   275         .LONG   ^XC              ; count
0000000B                049C   276         .LONG   ^XB              ; machine check type = illegal UNIBUS ref.
00FC0000                04A0   277         .LONG   A_ILL_UB         ; illegal unibus address
00000000                04A4   278         .LONG   ^X0              ; error parameter 2
00000358'               04A8   279         .LONG   TAG_4            ; PC
00000000                04AC   280         .LONG   ^X0              ; PSL
                        04B0   281
                        04B0   282
                        04B0   283
                        04B0   284
```

```
                                 04B0   286 .SBTTL ERROR REPORTING ROUTINE
                                 04B0   287
                                 04B0   288 ERR_MESSAGE:
                                 04B0   289          $DS_BGNMESSAGE  REGMASK=<R2,R7> ; Save R2 and R7
                           0084  04B0                       .WORD   ^M<R2,R7>                ; ENTRY MASK
        5E   00000084 8F   C2    04B2   290          SUBL2   #BUFSIZ+4,SP             ; Reserve room on stack for cvtreg
             52   5E   D0    04B9   291          MOVL    SP,R2                    ; Save starting address of buffer
                               04BC   292          $DS_PRINTB_S    FORMAT=T_MC_MESS
             FBA0 CF   9F    04BC                        PUSHAB  T_MC_MESS
        000100E0 9F   01    FB   04C0                   CALLS   #^$$N, @#DS$PRINTB
                               04C7   293          $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)
               14 BC   9F    04C7                        PUSHAB  @ERR$_P1(AP)
        000100E0 9F   01    FB   04CA                   CALLS   #^$$N, @#DS$PRINTB
         32 FB4A CF   0C    E1   04D1   294          BBC     #V_MC_ERR,L_FLAGS,1$     ; br = no machine check error
                               04D7   295          $DS_PRINTB_S    FORMAT=T_MC_ERR
             FC52 CF   9F    04D7                        PUSHAB  T_MC_ERR
        000100E0 9F   01    FB   04DB                   CALLS   #^$$N, @#DS$PRINTB
    57  38   00000000'EF   C1   04E2   296          ADDL3   A_MCHK_BUFFER,#56,R7     ; load address of actual data
            67   FF42 CF   7D   04EA   297          MOVQ    TABLE1,(R7)              ; load size and type
                               04EF   298          $DS_PRINTSIG_G  ARGPTR=(R7)
        00010100 9F   67    FA   04EF                   CALLG   (R7), @#DS$PRINTSIG
                               04F6   299          $DS_PRINTB_S    FORMAT=T_EXPECTED
             FC64 CF   9F    04F6                        PUSHAB  T_EXPECTED
        000100E0 9F   01    FB   04FA                   CALLS   #^$$N, @#DS$PRINTB
                               0501   300          $DS_PRINTSIG_G  ARGPTR=@ERR$_P2(AP)
      00010100 9F   18 BC   FA   0501                   CALLG   @ERR$_P2(AP), @#DS$PRINTSIG
         0B FB12 CF   00    E0   0509   301 1$:      BBS     #V_READY,L_FLAGS,3$      ; br = instruction ready to execute
                               050F   302          $DS_PRINTB_S    FORMAT=T_NOT_READY
             FC8D CF   9F    050F                        PUSHAB  T_NOT_READY
        000100E0 9F   01    FB   0513                   CALLS   #^$$N, @#DS$PRINTB
         0B FB01 CF   02    E1   051A   303 3$:      BBC     #V_EXECUTE_ERR,L_FLAGS,4$ ; br = instruction executed
                               0520   304          $DS_PRINTB_S    FORMAT=T_EXECUTE
             FCAE CF   9F    0520                        PUSHAB  T_EXECUTE
        000100E0 9F   01    FB   0524                   CALLS   #^$$N, @#DS$PRINTB
         17 FAF0 CF   09    E1   052B   305 4$:      BBC     #V_ISP_ERR,L_FLAGS,8$    ; br = no ISP error
                               0531   306          $DS_PRINTX_S    FORMAT=T_ISP_ERR,P0=L_EXP_ISP,P1=L_ACT_ISP
             00000000'EF   DD   0531                        PUSHL   L_ACT_ISP
             00000000'EF   DD   0537                        PUSHL   L_EXP_ISP
                 FCDE CF   9F   053D                        PUSHAB  T_ISP_ERR
        000100E8 9F   03    FB   0541                   CALLS   #^$$N, @#DS$PRINTX
         19 FAD3 CF   0A    E1   0548   307 8$:      BBC     #V_SP_ERR,L_FLAGS,9$     ; br = no SP error
                               054E   308          $DS_PRINTB_S    FORMAT=T_SP_ERR1,P0=L_EXP_SP,P1=L_ACT_SP
             00000000'EF   DD   054E                        PUSHL   L_ACT_SP
             00000000'EF   DD   0554                        PUSHL   L_EXP_SP
                 FCF5 CF   9F   055A                        PUSHAB  T_SP_ERR1
        000100E0 9F   03    FB   055E                   CALLS   #^$$N, @#DS$PRINTB
                     11   11   0565   309          BRB     10$                      ; skip extended error report
                               0567   310 9$:      $DS_PRINTX_S    FORMAT=T_SP_ERR2,P0=L_ACT_SP
             00000000'EF   DD   0567                        PUSHL   L_ACT_SP
                 FD0C CF   9F   056D                        PUSHAB  T_SP_ERR2
        000100E8 9F   02    FB   0571                   CALLS   #^$$N, @#DS$PRINTX
         03 FAA3 CF   05    E0   0578   311 10$:     BBS     #V_PSL_ERR,L_FLAGS,11$   ; br = PSL error
                   008A   31   057E   312          BRW     12$                      ; skip messages
                               0581   313 11$:     $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                               0581   314                          STRBUF=(R2),MAXLEN=#BUFSIZ,-
                               0581   315                          V1=MODELIST,V2=MODELIST
                         00    DD   0581                        PUSHL   #0
```

```
                      00    DD  0583                        PUSHL   #0
                      00    DD  0585                        PUSHL   #0
                      00    DD  0587                        PUSHL   #0
            00000000'EF     DF  0589                        PUSHAL  MODELIST
            00000000'EF     DF  058F                        PUSHAL  MODELIST
            00000080 8F     DD  0595                        PUSHL   #BUFSIZ
                      62    9F  059B                        PUSHAB  (R2)
            00000000'EF     9F  059D                        PUSHAB  A_PSLMNE
            00000000'EF     DD  05A3                        PUSHL   L_EXP_PSL
                      1F    DD  05A9                        PUSHL   #31
          000100B0 9F     0B    FB  05AB                    CALLS   #11, @#DS$CVTREG
                                05B2  316         $DS_PRINTB_S    FORMAT=T_EXP_PSL,P0=L_EXP_PSL,P1=R2
                      52    DD  05B2                        PUSHL   R2
            00000000'EF     DD  05B4                        PUSHL   L_EXP_PSL
                FD85 CF     9F  05BA                        PUSHAB  T_EXP_PSL
          000100E0 9F     03    FB  05BE                    CALLS   #$$N, @#DS$PRINTB
                                05C5  317         $DS_CVTREG_S    MSB=#31,DATA=L_ACT_PSL,MNEADR=A_PSLMNE,-
                                05C5  318                         STRBUF=(R2),MAXLEN=#BUFSIZ,-
                                05C5  319                         V1=MODELIST,V2=MODELIST
                      00    DD  05C5                        PUSHL   #0
                      00    DD  05C7                        PUSHL   #0
                      00    DD  05C9                        PUSHL   #0
                      00    DD  05CB                        PUSHL   #0
            00000000'EF     DF  05CD                        PUSHAL  MODELIST
            00000000'EF     DF  05D3                        PUSHAL  MODELIST
            00000080 8F     DD  05D9                        PUSHL   #BUFSIZ
                      62    9F  05DF                        PUSHAB  (R2)
            00000000'EF     9F  05E1                        PUSHAB  A_PSLMNE
            00000000'EF     DD  05E7                        PUSHL   L_ACT_PSL
                      1F    DD  05ED                        PUSHL   #31
          000100B0 9F     0B    FB  05EF                    CALLS   #11, @#DS$CVTREG
                                05F6  320         $DS_PRINTB_S    FORMAT=T_ACT_PSL,P0=L_ACT_PSL,P1=R2
                      52    DD  05F6                        PUSHL   R2
            00000000'EF     DD  05F8                        PUSHL   L_ACT_PSL
                FD62 CF     9F  05FE                        PUSHAB  T_ACT_PSL
          000100E0 9F     03    FB  0602                    CALLS   #$$N, @#DS$PRINTB
                      44    11  0609  321         BRB     13$
                                060B  322 12$:    $DS_CVTREG_S    MSB=#31,DATA=L_EXP_PSL,MNEADR=A_PSLMNE,-
                                060B  323                         STRBUF=(R2),MAXLEN=#BUFSIZ,-
                                060B  324                         V1=MODELIST,V2=MODELIST
                      00    DD  060B                        PUSHL   #0
                      00    DD  060D                        PUSHL   #0
                      00    DD  060F                        PUSHL   #0
                      00    DD  0611                        PUSHL   #0
            00000000'EF     DF  0613                        PUSHAL  MODELIST
            00000000'EF     DF  0619                        PUSHAL  MODELIST
            00000080 8F     DD  061F                        PUSHL   #BUFSIZ
                      62    9F  0625                        PUSHAB  (R2)
            00000000'EF     9F  0627                        PUSHAB  A_PSLMNE
            00000000'EF     DD  062D                        PUSHL   L_EXP_PSL
                      1F    DD  0633                        PUSHL   #31
          000100B0 9F     0B    FB  0635                    CALLS   #11, @#DS$CVTREG
                                063C  325         $DS_PRINTX_S    FORMAT=T_PSL,P0=L_EXP_PSL,P1=R2
                      52    DD  063C                        PUSHL   R2
            00000000'EF     DD  063E                        PUSHL   L_EXP_PSL
                FD3B CF     9F  0644                        PUSHAB  T_PSL
          000100E8 9F     03    FB  0648                    CALLS   #$$N, @#DS$PRINTX
```

```
                5C F9CC CF   06   E1  064F    326 13$:    BBC     #V_GPR_ERR,L_FLAGS,17$ ; br = no GPR error
                                   0655    327         $DS_PRINTB_S    FORMAT=T_GPR
                         FD43 CF   9F  0655                PUSHAB  T_GPR
                    000100E0 9F    01  FB  0659                CALLS   #$$N, @#DS$PRINTB
                    53   00000000'EF   DE  0660    328         MOVAL   GPR_TABLE,R3            ; load address of expected data
                    54   00000000'EF   D0  0667    329         MOVL    A_MCHK_BUFFER,R4        ; load address of actual data
                                 55   7C  066E    330         CLRQ    R5                      ; init R5 and R6 for counter and data
                              57   0D   D0  0670    331         MOVL    #13,R7                  ; load limit
                58   00000000'EF45   DE  0673    332 14$:    MOVAL   REG_TABLE[R5],R8        ; load address of ascic string
                  56   6445   6345   CD  067B    333         XORL3   (R3)[R5],(R4)[R5],R6    ; get XOR of data if bad
                                17   13  0681    334         BEQL    15$                     ; br = good data
                                   0683    335         $DS_PRINTB_S    FORMAT=T_XOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5],P3=R6
                                56   DD  0683                PUSHL   R6
                              6445   DD  0685                PUSHL   (R4)[R5]
                              6345   DD  0688                PUSHL   (R3)[R5]
                                58   DD  068B                PUSHL   R8
                         FD33 CF   9F  068D                PUSHAB  T_XOR
                    000100E0 9F    05  FB  0691                CALLS   #$$N, @#DS$PRINTB
                                13   11  0698    336         BRB     16$                     ; check if finished
                                   069A    337 15$:    $DS_PRINTX_S    FORMAT=T_NOXOR,P0=R8,P1=(R3)[R5],P2=(R4)[R5]
                              6445   DD  069A                PUSHL   (R4)[R5]
                              6345   DD  069D                PUSHL   (R3)[R5]
                                58   DD  06A0                PUSHL   R8
                         FD3A CF   9F  06A2                PUSHAB  T_NOXOR
                    000100E8 9F    04  FB  06A6                CALLS   #$$N, @#DS$PRINTX
                          C2 55    57  F3  06AD    338 16$:    AOBLEQ  R7,R5,14$
                 0B F96A CF    03   E1  06B1    339 17$:    BBC     #V_TIMER,L_FLAGS,18$    ; br = no timer error
                                   06B7    340         $DS_PRINTB_S    FORMAT=T_TIMER
                         FB3C CF   9F  06B7                PUSHAB  T_TIMER
                    000100E0 9F    01  FB  06BB                CALLS   #$$N, @#DS$PRINTB
                                   06C2    341 18$:    $DS_ENDMESSAGE
                              04   06C2                RET                             ; RETURN TO DIAGNOSTIC SUPERVISOR
                                   06C3    342
                                   06C3    343
                                   06C3    344
```

```
                        06C3   346 .SBTTL  SUBROUTINES
                        06C3   347
                        06C3   348 ; This routine will load R0 through R11 with data patterns and read the
                        06C3   349 ; AP and FP to check after causing a machine check.
                        06C3   350
                        06C3   351 LOAD_GPR:
5B  00000000'EF  DE     06C3   352          MOVAL   GPR_TABLE,R11   ; load address of data table to load GPR's
          50  8B  7D     06CA   353          MOVQ    (R11)+,R0       ; move data into R0 and R1
          52  8B  7D     06CD   354          MOVQ    (R11)+,R2       ; move data into R2 and R3
          54  8B  7D     06D0   355          MOVQ    (R11)+,R4       ; move data into R4 and R5
          56  8B  7D     06D3   356          MOVQ    (R11)+,R6       ; move data into R6 and R7
          58  8B  7D     06D6   357          MOVQ    (R11)+,R8       ; move data into R8 and R9
          5A  8B  7D     06D9   358          MOVQ    (R11)+,R10      ; move data into R10 and R11
00000000'EF   5C  D0     06DC   359          MOVL    AP,L_EXP_AP     ; move contents of AP into expected data table
00000000'EF   5D  D0     06E3   360          MOVL    FP,L_EXP_FP     ; move contents of FP into expected data table
                  05     06EA   361          RSB                     ; return
                        06EB   362
                        06EB   363 ; This routine will read general purpose registers R0 to R11 , the AP, FP, SP
                        06EB   364 ; KSP, ISP and PSL into a buffer area (A_MCHK_BUFFER) to check data patterns
                        06EB   365 ; and data after a machine check exception. It will then retrun to to address
                        06EB   366 ; contained in RENTRY_POINT.
                        06EB   367
                        06EB   368          .ALIGN  LONG
                        06EC   369 READ_GPR:
00000000'8F   00  DA     06EC   370          MTPR    #0,#PR$_MCESR   ; clear pending mach check flag
00000000'FF   50  7D     06F3   371          MOVQ    R0,@A_MCHK_BUFFER ; move contents of R0 and R1 into buffer
50  00000000'EF  D0     06FA   372          MOVL    A_MCHK_BUFFER,R0 ; load R0 with address of buffer
          50  08  C0     0701   373          ADDL2   #8,R0           ; get over first two entries
          80  52  7D     0704   374          MOVQ    R2,(R0)+        ; move contents of R2 and R3 into buffer
          80  54  7D     0707   375          MOVQ    R4,(R0)+        ; move R4 and R5
          80  56  7D     070A   376          MOVQ    R6,(R0)+        ; move R6 and R7
          80  58  7D     070D   377          MOVQ    R8,(R0)+        ; move R8 and R9
          80  5A  7D     0710   378          MOVQ    R10,(R0)+       ; move R10 and R11
          80  5C  D0     0713   379          MOVL    AP,(R0)+        ; move AP
          80  5D  D0     0716   380          MOVL    FP,(R0)+        ; move FP
    00000000'EF   5E  D0     0719   381          MOVL    SP,L_ACT_SP     ; move SP
00000000'EF  00000000'8F  DB     0720   382          MFPR    #PR$_ISP,L_ACT_ISP ; move ISP
       00000000'EF  DC     072B   383          MOVPSL  L_ACT_PSL       ; move PSL
          50  08  C0     0731   384          ADDL2   #8,R0           ; save room for printsig size and type
              57  D4     0734   385          CLRL    R7              ; init counter
          55  05  D0     0736   386          MOVL    #5,R5           ; set limit
          80  8E  D0     0739   387 1$:       MOVL    (SP)+,(R0)+     ; move MC logout into buffer
        F9 57  55  F3     073C   388          AOBLEQ  R5,R7,1$        ; br = not finished
          F8C8 DF  17     0740   389          JMP     @RENTRY_POINT   ; return to subtest
                        0744   390
                        0744   391
                        0744   392
                        0744   393 ; This routire is called after execution of an instruction which should cause
                        0744   394 ; a machine check. The MACHINE CHECK LOGOUT, PSL, SP, KSP, ISP, GENERAL
                        0744   395 ; PURPOSE REGISTERS and FLAG BYTE are examined for expected information. If
                        0744   396 ; an error exists, a flag is set to indicate that it should be reported.
                        0744   397
                        0744   398
                        0744   399 ERROR_CHECK:
              57  D4     0744   400          CLRL    R7              ; initialize counter
          55  0D  D0     0746   401          MOVL    #13,R5          ; set limit
    53  00000000'EF  D0     0749   402          MOVL    A_MCHK_BUFFER,R3 ; load address of actual data to check
```

```
                                    0750   403          MOVAL   GPR_TABLE,R4     ; load address of expected data
                 54   00000000'EF   DE   0750   403          MOVAL   GPR_TABLE,R4     ; load address of expected data
                      84   83       D1   0757   404 1$:      CMPL    (R3)+,(R4)+      ; check data
                           09       13   075A   405          BEQL    2$               ; br = data is good
                F8BB CF   000000C0 8F   C8   075C   406          BISL2   #M_ERROR_FLAG!M_GPR_ERR,L_FLAGS ; set GPR and error flag
                      EE 57   55       F3   0765   407 2$:      AOBLEQ  R5,R7,1$         ; br = not finished
                   1A F8B2 CF   0D       E0   0769   408          BBS     #V_INT,L_FLAGS,5$ ; br = checking an interrupt
                      53   08       C0   076F   409          ADDL2   #8,R3            ; get over size and type to check machine check
                           57       D4   0772   410          CLRL    R7               ; init counter
                      55   05       D0   0774   411          MOVL    #5,R5            ; set limit
                      86   83       D1   0777   412 3$:      CMPL    (R3)+,(R6)+      ; check MC logout data
                           09       13   077A   413          BEQL    4$               ; br = data good
                F89B CF   00001080 8F   C8   077C   414          BISL2   #M_ERROR_FLAG!M_MC_ERR,L_FLAGS ; set error flag
                      EE 57   55       F3   0785   415 4$:      AOBLEQ  R5,R7,3$         ; br = not finished
00000000'EF   00000000'EF   D1   0789   416 5$:      CMPL    L_ACT_SP,L_EXP_SP ; check SP
                           09       13   0794   417          BEQL    7$               ; br = SP is good
                F881 CF   00000480 8F   C8   0796   418          BISL2   #M_SP_ERR!M_ERROR_FLAG,L_FLAGS ; set SP and error flag
00000000'EF   00000000'EF   D1   079F   419 7$:      CMPL    L_ACT_ISP,L_EXP_ISP ; check ISP
                           09       13   07AA   420          BEQL    8$               ; br = ISP is good
                F86B CF   00000280 8F   C8   07AC   421          BISL2   #M_ISP_ERR!M_ERROR_FLAG,L_FLAGS ; set ISP and error flag
00000000'EF   00000000'EF   D1   07B5   422 8$:      CMPL    L_ACT_PSL,L_EXP_PSL ; check PSL
                           3E       13   07C0   423          BEQL    11$              ; br = PSL is good
                      02   18       EF   07C2   424          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,-
                F843 CF   00000000'EF        07C5   425                  L_EXP_PSL,L_EXP_MOD ; extract current mode from expected psl
                      02   18       ED   07CD   426          CMPZV   #PSL$V_CURMOD,#PSL$S_CURMOD,-
                F838 CF   00000000'EF        07D0   427                  L_ACT_PSL,L_EXP_MOD ; check current mode for error
                           18       12   07D8   428          BNEQ    9$               ; br = wrong mode
                      01   1A       EF   07DA   429          EXTZV   #PSL$V_IS,#1,-
                F82F CF   00000000'EF        07DD   430                  L_EXP_PSL,L_EXP_STACK ; extract stack from expected PSL
                      01   1A       ED   07E5   431          CMPZV   #PSL$V_IS,#1,-
                F824 CF   00000000'EF        07E8   432                  L_ACT_PSL,L_EXP_STACK ; check current stack for error
                           05       13   07F0   433          BEQL    10$              ; br = mode is good
                     F829 CF   10       C8   07F2   434 9$:      BISL2   #M_FATAL_ERR,L_FLAGS ; signal fatal error
                F820 CF   000000A0 8F   C8   07F7   435 10$:     BISL2   #M_PSL_ERR!M_ERROR_FLAG,L_FLAGS ; set PSL and error flag
                   09 F81B CF   00       E0   0800   436 11$:     BBS     #V_READY,L_FLAGS,12$ ; br = no ready error
                F811 CF   00000080 8F   C8   0806   437          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
                   09 F80C CF   02       E1   080F   438 12$:     BBC     #V_EXECUTE_ERR,L_FLAGS,13$ ; br = no execute error
                F802 CF   00000080 8F   C8   0815   439          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
                   09 F7FD CF   03       E1   081E   440 13$:     BBC     #V_TIMER,L_FLAGS,14$ ; br = no unexpected timer interrupt
                F7F3 CF   00000080 8F   C8   0824   441          BISL2   #M_ERROR_FLAG,L_FLAGS ; set error flag
                                        05   082D   442 14$:     RSB                      ; return
                                             082E   443
                                             082E   444
                                             082E   445 ; This routine will service any unexpected interval timer interrupts.
                                             082E   446
                                             082E   447          .ALIGN  LONG
                                             0830   448 TIMER_SERVICE:
                     F7EB CF   08       C8   0830   449          BISL2   #M_TIMER,L_FLAGS ; set error flag
                                        02   0835   450          REI                      ; return
                                             0836   451
                                             0836   452
                                             0836   453 ; This routine will clear the buffer area.
                                             0836   454
                                             0836   455 CLEAR_BUFFER:
                 55   00000000'FF   DE   0836   456          MOVAL   @A_MCHK_BUFFER,R5 ; get starting buffer address
                      F7BF CF   D4   083D   457          CLRL    L_COUNT          ; initialize counter
                           85   D4   0841   458 1$:      CLRL    (R5)+            ; clear a buffer location
             F4 F7B4 CF   00000100 8F   F3   0843   459          AOBLEQ  #256,L_COUNT,1$ ; br = not finished
```

```
 05  084D   460          RSB                         ; return
     084E   461
     084E   462
     084E   463
     084E   464
     084E   465          $DS_PAGE
```

```
                        084E                    .SBTTL  TEST 5:  Machine Check Exceptions
                    00000000                    .PSECT  TEST_005, PAGE, NOWRT
                        001C
                        001C    468 ;++
                        001C    469 ; Test description:
                        001C    470 ;
                        001C    471 ;       This test checks that a reference to NXM, unaligned reference to
                        001C    472 ;       I/O space, illegal I/O space address, and illegal UNIBUS reference
                        001C    473 ;       will cause machine checks with the proper logout information.
                        001C    474 ;
                        001C    475 ;--
                        001C    476
                        001C    477             $DS_BGNTEST     <MCHK,ALL,DEFAULT>
                        001C        DATA_005:
                    00000000    001C                    .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                        0020        TEST_005::
                        0000    0020                    .WORD   ^M<>            ; ENTRY MASK
                        0022    478
          00000000'EF   D5      0022    479             TSTL    KA_PTABLE       ; CPU selected?
                    10  18      0028    480             BGEQ    0$              ; br = selected
                        002A    481             $DS_PRINTF_S    FORMAT=T_NO_KA730 ; print exit warning
          00000028'EF   9F      002A                    PUSHAB  T_NO_KA730
      000100F0 9F       01  FB  0030                    CALLS   #$$N, @#DS$PRINTF
                        0037    482             $DS_EXIT        TEST            ; exit test
                    03A8    31  0037                    BRW     TEST_005_X      ; EXIT TEST 5
                        003A    483 0$:
                        003A    484             $DS_SETVEC_S    VECTOR=#SCB$L_TIMER, SRVADR=TIMER_SERVICE
                    00      DD  003A                    PUSHL   #0
          00000830'EF       DF  003C                    PUSHAL  TIMER_SERVICE
          000000C0 8F       DD  0042                    PUSHL   #SCB$L_TIMER
      00010160 9F       03  FB  0048                    CALLS   #3, @#DS$SETVEC
      00000000'EF       5E  D0  004F    485 1$:     MOVL    SP,L_KSP                ; save SP
                        0056    486             $DS_PAGE
```

```
                                0056                    .SBTTL  SUBTEST 5.1:  Reference to NXM
                                0056    489 ;+
                                0056    490 ; Subtest description:
                                0056    491 ;
                                0056    492 ;       This subtest attempts to read a nonexistent memory location,
                                0056    493 ;       which should cause a machine check. The machine check logout
                                0056    494 ;       pushed on the stack is checked for the correct information.
                                0056    495 ;
                                0056    496 ; Subtest steps:
                                0056    497 ;
                                0056    498 ;       SUBTEST Reference to NXM
                                0056    499 ;       : Get Machine Check vector to capture exception
                                0056    500 ;       : IF unsucessful
                                0056    501 ;       : : THEN
                                0056    502 ;       : : Report system error
                                0056    503 ;       : : Abort test
                                0056    504 ;       : ENDIF
                                0056    505 ;       : Setup expected data and load GPR's with data
                                0056    506 ;       : Set ready to execute flag
                                0056    507 ;       : Access nonexistent memory (to cause exception)
                                0056    508 ;       : Set execution error flag
                                0056    509 ;       : IF no exception
                                0056    510 ;       : : THEN
                                0056    511 ;       : : Set execution error flag
                                0056    512 ;       : ENDIF
                                0056    513 ;       : Check machine check logout and register data
                                0056    514 ;       : IF error is detected
                                0056    515 ;       : : THEN
                                0056    516 ;       : : Report data error
                                0056    517 ;       : ENDIF
                                0056    518 ;       : IF fatal error flag is set
                                0056    519 ;       : : THEN
                                0056    520 ;       : : Abort test
                                0056    521 ;       : ENDIF
                                0056    522 ;       : Release Machine Check vector
                                0056    523 ;       : IF unsucessful
                                0056    524 ;       : : THEN
                                0056    525 ;       : : Report system error
                                0056    526 ;       : : Abort test
                                0056    527 ;       : ENDIF
                                0056    528 ;       : Push return PC and PSL on stack
                                0056    529 ;       : Execute an REI
                                0056    530 ;       ENDSUBTEST Reference to NXM
                                0056    531 ;-
                                0056    532
                                0056    533         $DS_BGNSUB
                                0056        T5_S1::
00010030 9F    00000000'EF  FA  0056                CALLG   $S$, @#DS$BGNSUB
         00000000'8F    00  DA  0061    534         MTPR    #0,#PR$_MCESR           ; clear pending mach check
              00000836'EF   16  0068    535         JSB     CLEAR_BUFFER            ; clear buffer area
                            006E    536         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_GPR
                        00  DD  006E                PUSHL   #0
              000006EC'EF   DF  0070                PUSHAL  READ_GPR
                        04  DD  0076                PUSHL   #SCB$L_MACHCHK
         00010160 9F    03  FB  0078                CALLS   #3, @#DS$SETVEC
0000000C'EF   000000CD'8F   D0  007F    537 1$:     MOVL    #RENTRY1,RENTRY_POINT   ; load rentry address
              00000020'EF   D4  008A    538         CLRL    L_FLAGS                 ; init flags
```

```
                00000000'EF   5E   18   C3  0090   539          SUBL3   #^X18,SP,L_EXP_SP       ; load expected SP
           00000000'EF   00000000'8F   DB  0098   540          MFPR    #PR$_ISP,L_EXP_ISP      ; load expected ISP
           00000000'EF   0000042C'EF   D0  00A3   541          MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
                         000006C3'EF   16  00AE   542          JSB     LOAD_GPR                ; load GPR's with data
                00000020'EF   01   C8  00B4   543          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
           00000000'EF   00500200'EF   D0  00BB   544 TAG_1:   MOVL    A_NXM,A_MCHK_BUFFER     ; reference non-existent memory
                                       00C6   545                                          ;       to generate machine check
                00000020'EF   04   C8  00C6   546          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; set execution error flag (should
                                       00CD   547                                          ;       skip over if exception occurs)
                                       00CD   548 RENTRY1:
                                       00CD   549          $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                                              PUSHL   #SCB$L_MACHCHK
                                              CALLS   #1, @#DS$CLRVEC
                00010168 9F   01   FB  00CF
                                 56   D4  00D6   550 3$:      CLRL    R6                      ; init R6
           56   08   00000430'8F   C1  00D8   551          ADDL3   #TABLE1,#8,R6           ; load R6 with address of exp data
                         00000744'EF   16  00E0   552          JSB     ERROR_CHECK             ; go check logout info for errors
             3C 00000020'EF   07   E1  00E6   553          BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no error
                                       00EE   554          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_BUSERR,P2=#TABLE1
                         00000430'8F   DD  00EE
                         00000089'8F   DD  00F4
                         000004B0'EF   DF  00FA
                                 00   DD  0100
                         00000000'EF   DD  0102
                                 01   DD  0108
                                       010A       ;;; TEST 5, SUBTEST 1, ERROR 1
                00010DD0 9F   06   FB  010A
                                       0111   555          $DS_BCOMPLETE   4$
                               07 50   E8  0111
                                       0114   556          $DS_ABORT
                00010020 9F   6C   FA  0114
             07 00000020'EF   04   E1  011B   557 4$:      BBC     #V_FATAL_ERR,L_FLAGS,5$ ; br = no fatal errors
                                       0123   558          $DS_ABORT
                00010020 9F   6C   FA  0123
                                       012A   559 5$:
                                       012A   560          $DS_ENDSUB
                                       012A       T5_S1_X:
           00010038 9F   00000000'EF   FA  012A
                                       0135   561
                                       0135   562          $DS_PAGE
```

```
                                0135                    .SBTTL  SUBTEST 5.2:  Unaligned reference to I/O space
                                0135    565 ;+
                                0135    566 ; Subtest description:
                                0135    567 ;
                                0135    568 ;       This subtest attempts an unaligned reference to I/O space
                                0135    569 ;       and checks that a machine check occurs with the correct
                                0135    570 ;       logout information on the stack.
                                0135    571 ;
                                0135    572 ; Subtest steps:
                                0135    573 ;
                                0135    574 ;       SUBTEST Unaligned reference to I/O space
                                0135    575 ;       :  Get Machine Check vector to capture exception
                                0135    576 ;       :  IF unsucessful
                                0135    577 ;       :  :  THEN
                                0135    578 ;       :  :  Report system error
                                0135    579 ;       :  :  Abort test
                                0135    580 ;       :  ENDIF
                                0135    581 ;       :  Setup expected data and load GPR's with data
                                0135    582 ;       :  Set ready to execute flag
                                0135    583 ;       :  Access unaligned I/O space (to cause exception)
                                0135    584 ;       :  Set execution error flag
                                0135    585 ;       :  IF no exception
                                0135    586 ;       :  :  THEN
                                0135    587 ;       :  :  Set execution error flag
                                0135    588 ;       :  ENDIF
                                0135    589 ;       :  Check machine check logout and register data
                                0135    590 ;       :  IF error is detected
                                0135    591 ;       :  :  THEN
                                0135    592 ;       :  :  Report data error
                                0135    593 ;       :  ENDIF
                                0135    594 ;       :  IF fatal error flag is set
                                0135    595 ;       :  :  THEN
                                0135    596 ;       :  :  Abort test
                                0135    597 ;       :  ENDIF
                                0135    598 ;       :  Release Machine Check vector
                                0135    599 ;       :  IF unsucessful
                                0135    600 ;       :  :  THEN
                                0135    601 ;       :  :  Report system error
                                0135    602 ;       :  :  Abort test
                                0135    603 ;       :  ENDIF
                                0135    604 ;       :  Push return PC and PSL on stack
                                0135    605 ;       :  Execute an REI
                                0135    606 ;       ENDSUBTEST Unaligned reference to I/O space
                                0135    607 ;-
                                0135    608
                                0135    609             $DS_BGNSUB
                                0135            T5_S2::
00010030 9F   0000000C'EF   FA  0135                    CALLG   $S$, @#DS$BGNSUB
         00000000'8F   00   DA  0140    610             MTPR    #0,#PR$_MCESR           ; clear pending mach check
              00000836'EF   16  0147    611             JSB     CLEAR_BUFFER            ; clear buffer area
                            014D    612             $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_GPR
                        00  DD  014D                    PUSHL   #0
              000006EC'EF   DF  014F                    PUSHAL  READ_GPR
                        04  DD  0155                    PUSHL   #SCB$L_MACHCHK
         00010160 9F    03  FB  0157                    CALLS   #3, @#DS$SETVEC
0000000C'EF   000001AC'8F   D0  015E    613 1$:         MOVL    #RENTRY2,RENTRY_POINT   ; load rentry address
              00000020'EF   D4  0169    614             CLRL    L_FLAGS                 ; init flags
```

```
      00000000'EF   5E   18   C3  016F   615          SUBL3   #^X18,SP,L_EXP_SP       ; load expected SP
00000000'EF   00000000'8F   DB  0177   616          MFPR    #PR$_ISP,L_EXP_ISP      ; load expected ISP
00000000'EF   0000042C'EF   D0  0182   617          MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
              000006C3'EF   16  018D   618          JSB     LOAD_GPR                ; load GPR's with data
         00000020'EF   01   C8  0193   619          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
00000000'EF   00F00001'EF   D0  019A   620 TAG_2:    MOVL    A_UNAL_IO,A_MCHK_BUFFER ; reference unaligned I/O address
                            01A5   621                                           ;       to generate machine check
         00000020'EF   04   C8  01A5   622          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; set execution error flag (should
                            01AC   623                                           ;       skip over if exception occurs)
                            01AC   624 RENTRY2:
                            01AC   625          $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                                                PUSHL   #SCB$L_MACHCHK
         00010168 9F   01   FB  01AE                CALLS   #1, @#DS$CLRVEC
                       56   D4  01B5   626 3$:      CLRL    R6                      ; init R6
      56   08   00000450'8F   C1  01B7   627          ADDL3   #TABLE2,#8,R6           ; load R6 with address of exp data
              00000744'EF   16  01BF   628          JSB     ERROR_CHECK             ; go check logout info for errors
        3C 00000020'EF   07   E1  01C5   629          BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no error
                            01CD   630          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_UNALIO_ERR,P2=#TABL
              00000450'8F   DD  01CD                PUSHL   #TABLE2
              000000B6'8F   DD  01D3                PUSHL   #T_UNALIO_ERR
              000004B0'EF   DF  01D9                PUSHAL  ERR_MESSAGE
                       00   DD  01DF                PUSHL   #0
              00000000'EF   DD  01E1                PUSHL   KA_UNIT_NO
                       01   DD  01E7                PUSHL   #$ER
                            01E9       ;;; TEST 5, SUBTEST 2, ERROR 1
         000100D0 9F   06   FB  01E9                CALLS   #$$M, @#DS$ERRHARD
                            01F0   631          $DS_BCOMPLETE   4$
                  07 50   E8  01F0                BLBS    R0, 4$
                            01F3   632          $DS_ABORT
      00010020 9F   6C   FA  01F3                CALLG   (AP), @#DS$ABORT
   07 00000020'EF   04   E1  01FA   633 4$:      BBC     #V_FATAL_ERR,L_FLAGS,5$ ; br = no fatal errors
                            0202   634          $DS_ABORT
      00010020 9F   6C   FA  0202                CALLG   (AP), @#DS$ABORT
                            0209   635 5$:
                            0209   636          $DS_ENDSUB
                            0209       T5_S2_X:
00010038 9F   0000000C'EF   FA  0209                CALLG   $S$, @#DS$ENDSUB
                            0214   637          $DS_PAGE
```

```
                                0214                    .SBTTL  SUBTEST 5.3:  Illegal I/O space address
                                0214    640 ;+
                                0214    641 ; Subtest description:
                                0214    642 ;
                                0214    643 ;       This subtest attempts an illegal reference to I/O space
                                0214    644 ;       and checks that a machine check occurs with the correct
                                0214    645 ;       logout information on the stack.
                                0214    646 ;
                                0214    647 ; Subtest steps:
                                0214    648 ;
                                0214    649 ;       SUBTEST Illegal I/O space address
                                0214    650 ;       :  Get Machine Check vector to capture exception
                                0214    651 ;       :  IF unsucessful
                                0214    652 ;       :  :  THEN
                                0214    653 ;       :  :  Report system error
                                0214    654 ;       :  :  Abort test
                                0214    655 ;       :  ENDIF
                                0214    656 ;       :  Setup expected data and load GPR's with data
                                0214    657 ;       :  Set ready to execute flag
                                0214    658 ;       :  Access illegal I/O space (to cause exception)
                                0214    659 ;       :  Set execution error flag
                                0214    660 ;       :  IF no exception
                                0214    661 ;       :  :  THEN
                                0214    662 ;       :  :  Set execution error flag
                                0214    663 ;       :  ENDIF
                                0214    664 ;       :  Check machine check logout and register data
                                0214    665 ;       :  IF error is detected
                                0214    666 ;       :  :  THEN
                                0214    667 ;       :  :  Report data error
                                0214    668 ;       :  ENDIF
                                0214    669 ;       :  IF fatal error flag is set
                                0214    670 ;       :  :  THEN
                                0214    671 ;       :  :  Abort test
                                0214    672 ;       :  ENDIF
                                0214    673 ;       :  Release Machine Check vector
                                0214    674 ;       :  IF unsucessful
                                0214    675 ;       :  :  THEN
                                0214    676 ;       :  :  Report system error
                                0214    677 ;       :  :  Abort test
                                0214    678 ;       :  ENDIF
                                0214    679 ;       :  Push return PC and PSL on stack
                                0214    680 ;       :  Execute an REI
                                0214    681 ;       ENDSUBTEST Illegal I/O space address
                                0214    682 ;-
                                0214    683
                                0214    684         $DS_BGNSUB
                                0214        T5_S3::
00010030 9F   00000018'EF   FA  0214                CALLG   $S$, @#DS$BGNSUB
         00000000'8F    00  DA  021F    685         MTPR    #0,#PR$_MCESR           ; clear pending mach check
              00000836'EF   16  0226    686         JSB     CLEAR_BUFFER            ; clear buffer area
                            022C    687         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_GPR
                        00  DD  022C                PUSHL   #0
                000006EC'EF  DF  022E                PUSHAL  READ_GPR
                        04  DD  0234                PUSHL   #SCB$L_MACHCHK
         00010160 9F    03  FB  0236                CALLS   #3, @#DS$SETVEC
0000000C'EF   0000028B'8F   D0  023D    688 1$:     MOVL    #RENTRY3,RENTRY_POINT   ; load rentry address
              00000020'EF   D4  0248    689         CLRL    L_FLAGS                 ; init flags
```

```
      00000000'EF    5E    18    C3  024E   690          SUBL3   #^X18,SP,L_EXP_SP         ; load expected SP
   00000000'EF    00000000'8F    DB  0256   691          MFPR    #PR$_ISP,L_EXP_ISP        ; load expected ISP
   00000000'EF    0000042C'EF    D0  0261   692          MOVL    DATA_1,L_EXP_PSL          ; load expected PSL
                  000006C3'EF    16  026C   693          JSB     LOAD_GPR                  ; load GPR's with data
         00000020'EF    01       C8  0272   694          BISL2   #M_READY,L_FLAGS          ; set ready to execute flag
   00000000'EF    00F2A000'EF    D0  0279   695 TAG_3:   MOVL    A_ILL_IO,A_MCHK_BUFFER    ; reference illegal I/O address
                                     0284   696                                            ;         to generate machine check
         00000020'EF    04       C8  0284   697          BISL2   #M_EXECUTE_ERR,L_FLAGS    ; set execution error flag (should
                                     028B   698                                            ;         skip over if exception occurs)
                                     028B   699 RENTRY3:
                                     028B   700          $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                        04       DD  028B                        PUSHL   #SCB$L_MACHCHK
         00010168 9F    01       FB  028D                        CALLS   #1, @#DS$CLRVEC
                        56       D4  0294   701 3$:      CLRL    R6                        ; init R6
     56    08    00000470'8F     C1  0296   702          ADDL3   #TABLE3,#8,R6             ; load R6 with address of exp data
                  00000744'EF    16  029E   703          JSB     ERROR_CHECK               ; go check logout info for errors
      3C 00000020'EF    07       E1  02A4   704          BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no error
                                     02AC   705          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_ILLIO_ERR,P2=#TABLE
                  00000470'8F    DD  02AC                        PUSHL   #TABLE3
                  000000DE'8F    DD  02B2                        PUSHL   #T_ILLIO_ERR
                  000004B0'EF    DF  02B8                        PUSHAL  ERR_MESSAGE
                          00     DD  02BE                        PUSHL   #0
                  00000000'EF    DD  02C0                        PUSHL   KA_UNIT_NO
                          01     DD  02C6                        PUSHL   #$ER
                                     02C8       ;;; TEST 5, SUBTEST 3, ERROR 1
         000100D0 9F    06       FB  02C8                        CALLS   #$$M, @#DS$ERRHARD
                                     02CF   706          $DS_BCOMPLETE   4$
                       07 50     E8  02CF                        BLBS    R0, 4$
                                     02D2   707          $DS_ABORT
         00010020 9F    6C       FA  02D2                        CALLG   (AP), @#DS$ABORT
      07 00000020'EF    04       E1  02D9   708 4$:      BBC     #V_FATAL_ERR,L_FLAGS,5$ ; br = no fatal errors
                                     02E1   709          $DS_ABORT
         00010020 9F    6C       FA  02E1                        CALLG   (AP), @#DS$ABORT
                                     02E8   710 5$:
                                     02E8   711          $DS_ENDSUB
                                     02E8       T5_S3_X:
   00010038 9F    00000018'EF    FA  02E8                        CALLG   $S$, @#DS$ENDSUB
                                     02F3   712          $DS_PAGE
```

```
ZZ-
ENK
Cro

DSA
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ENV
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
ERR
GPR
HAL
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
HP$
IPR
KA$
KA$
KA$
KA$
KA$
KA$
KA$
KA$
KA$
KA$
KA$
```

```
                                02F3                    .SBTTL  SUBTEST 5.4:  Illegal UNIBUS reference
                                02F3     715 ;+
                                02F3     716 ; Subtest description:
                                02F3     717 ;
                                02F3     718 ;       This subtest attempts an illegal UNIBUS reference
                                02F3     719 ;       and checks that a machine check occurs with the correct
                                02F3     720 ;       logout information on the stack.
                                02F3     721 ;
                                02F3     722 ; Subtest steps:
                                02F3     723 ;
                                02F3     724 ;       SUBTEST Illegal UNIBUS reference
                                02F3     725 ;       :  Get Machine Check vector to capture exception
                                02F3     726 ;       :  IF unsucessful
                                02F3     727 ;       :  :  THEN
                                02F3     728 ;       :  :  Report system error
                                02F3     729 ;       :  :  Abort test
                                02F3     730 ;       :  ENDIF
                                02F3     731 ;       :  Setup expected data and load GPR's with data
                                02F3     732 ;       :  Set ready to execute flag
                                02F3     733 ;       :  Access illegal UNIBUS address (to cause exception)
                                02F3     734 ;       :  Set execution error flag
                                02F3     735 ;       :  IF no exception
                                02F3     736 ;       :  :  THEN
                                02F3     737 ;       :  :  Set execution error flag
                                02F3     738 ;       :  ENDIF
                                02F3     739 ;       :  Check machine check logout and register data
                                02F3     740 ;       :  IF error is detected
                                02F3     741 ;       :  :  THEN
                                02F3     742 ;       :  :  Report data error
                                02F3     743 ;       :  ENDIF
                                02F3     744 ;       :  IF fatal error flag is set
                                02F3     745 ;       :  :  THEN
                                02F3     746 ;       :  :  Abort test
                                02F3     747 ;       :  ENDIF
                                02F3     748 ;       :  Release Machine Check vector
                                02F3     749 ;       :  IF unsucessful
                                02F3     750 ;       :  :  THEN
                                02F3     751 ;       :  :  Report system error
                                02F3     752 ;       :  :  Abort test
                                02F3     753 ;       :  ENDIF
                                02F3     754 ;       :  Push return PC and PSL on stack
                                02F3     755 ;       :  Execute an REI
                                02F3     756 ;       ENDSUBTEST Illegal UNIBUS reference
                                02F3     757 ;-
                                02F3     758
                                02F3     759         $DS_BGNSUB
                                02F3             T5_S4::
00010030 9F   00000024'EF   FA  02F3                 CALLG   $S$, a#DS$BGNSUB
         00000000'8F   00   DA  02FE     760         MTPR    #0,#PR$_MCESR           ; clear pending mach check
              00000836'EF   16  0305     761         JSB     CLEAR_BUFFER            ; clear buffer area
                            00  030B     762         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK, SRVADR=READ_GPR
                                030B                 PUSHL   #0
                   000006EC'EF  DF  030D             PUSHAL  READ_GPR
                            04  DD  0313             PUSHL   #SCB$L_MACHCHK
         00010160 9F       03  FB  0315             CALLS   #3, a#DS$SETVEC
00000000'EF   0000036A'8F   D0  031C     763 1$:     MOVL    #RENTRY4,RENTRY_POINT   ; load rentry address
              00000020'EF   D4  0327     764         CLRL    L_FLAGS                 ; init flags
```

```
      00000000'EF   5E   18   C3  032D   765          SUBL3   #^X18,SP,L_EXP_SP       ; load expected SP
00000000'EF   00000000'8F   DB  0335   766          MFPR    #PR$_ISP,L_EXP_ISP      ; load expected ISP
00000000'EF   0000042C'EF   D0  0340   767          MOVL    DATA_1,L_EXP_PSL        ; load expected PSL
              000006C3'EF   16  0348   768          JSB     LOAD_GPR                ; load GPR's with data
           00000020'EF   01   C8  0351   769          BISL2   #M_READY,L_FLAGS        ; set ready to execute flag
00000000'EF   00FC0000'EF   D0  0358   770 TAG_4:    MOVL    A_ILL_UB,A_MCHK_BUFFER  ; reference illegal unibus address
                            0363   771                                              ;       to generate machine check
           00000020'EF   04   C8  0363   772          BISL2   #M_EXECUTE_ERR,L_FLAGS  ; set execution error flag (should
                            036A   773                                              ;       skip over if exception occurs)
                            036A   774 RENTRY4:
                            036A   775          $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                      04    DD  036A                      PUSHL   #SCB$L_MACHCHK
           00010168 9F    01   FB  036C                      CALLS   #1, @#DS$CLRVEC
                      56    D4  0373   776 3$:       CLRL    R6                      ; init R6
      56   08    00000490'8F   C1  0375   777          ADDL3   #TABLE4,#8,R6           ; load R6 with address of exp data
              00000744'EF   16  037D   778          JSB     ERROR_CHECK             ; go check logout info for errors
        3C 00000020'EF   07   E1  0383   779          BBC     #V_ERROR_FLAG,L_FLAGS,5$ ; br = no error
                            038B   780          $DS_ERRHARD_S   UNIT=KA_UNIT_NO,PRLINK=ERR_MESSAGE,P1=#T_ILLUB_ERR,P2=#TABLE
              00000490'8F   DD  038B                      PUSHL   #TABLE4
              00000104'8F   DD  0391                      PUSHL   #T_ILLUB_ERR
              000004B0'EF   DF  0397                      PUSHAL  ERR_MESSAGE
                      00    DD  039D                      PUSHL   #0
              00000000'EF   DD  039F                      PUSHL   KA_UNIT_NO
                      01    DD  03A5                      PUSHL   #$ER
                            03A7       ;;; TEST 5, SUBTEST 4, ERROR 1
           000100D0 9F    06   FB  03A7                      CALLS   #$$M, @#DS$ERRHARD
                            03AE   781          $DS_BCOMPLETE   4$
                   07 50    E8  03AE                      BLBS    R0, 4$
                            03B1   782          $DS_ABORT
        00010020 9F    6C   FA  03B1                      CALLG   (AP), @#DS$ABORT
     07 00000020'EF    04   E1  03B8   783 4$:       BBC     #V_FATAL_ERR,L_FLAGS,5$ ; br = no fatal errors
                            03C0   784          $DS_ABORT
        00010020 9F    6C   FA  03C0                      CALLG   (AP), @#DS$ABORT
                            03C7   785 5$:
                            03C7   786          $DS_ENDSUB
                            03C7       T5_S4_X:
00010038 9F    00000024'EF   FA  03C7                      CALLG   $S$, @#DS$ENDSUB
                            03D2   787
                            03D2   788          $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
           000000C0 8F    DD  03D2                      PUSHL   #SCB$L_TIMER
        00010168 9F    01   FB  03D8                      CALLS   #1, @#DS$CLRVEC
                            03DF   789          $DS_ENDTEST
                   50    01   D0  03DF                      MOVL    #1, R0          ; NORMAL EXIT
                            03E2       TEST_005_X::
        00010058 9F    6E   FA  03E2                      CALLG   (SP), @#DS$BREAK
                      04    03E9                      RET                     ; RETURN TO TEST SEQUENCER
                            03EA   790          $DS_ENDMOD
                            00000000                  .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                   00000005  0000                     .LONG   $TN
                            0004   791          .END
```

```
$$$                             = 00000024 R      06
$$ARGS                          = 0000000A
$$E                             = 00000001
$$M                             = 00000006
$$N                             = 00000001
$$S                             = FFFFFFFF
$$T1                            = 0000002C
$ENV                            = 00000001   G
$ER                             = FFFFFFFF
$MO                             = 00000001   G
$S$                             = 00000024 R      06
$ST                             = 00000000
$TN                             = 00000005
ALL                             = 00000008   G
A_ILL_IO                        = 00F2A000
A_ILL_UB                        = 00FC0000
A_MCHK_BUFFER                     ********     X   02
A_NXM                           = 00500200
A_PSLMNE                          ********     X   02
A_UNAL_IO                       = 00F00001
A_WCS_ADDRESS                     00000024 R      02
BASE_005                          00000000 R      04
BIT...                          = 0000001A
BUFSIZ                          = 00000080
CEP_FUNCTIONAL                  = 00000000
CEP_REPAIR                      = 00000001
CLEAR_BUFFER                      00000836 R      02
CPU$V_COMET                     = 00000002
CPU$V_HS                        = 00000000
CPU$V_STAR                      = 00000001
DATA_005                          0000001C R      04
DATA_1                            0000042C R      02
DEFAULT                         = 00000000   G
DS$ABORT                          00010020
DS$ASKADR                         00010090
DS$ASKDATA                        00010080
DS$ASKLGCL                        00010098
DS$ASKSTR                         000100A0
DS$ASKVLD                         00010088
DS$ATTACH                         000101A8
DS$BGNSUB                         00010030
DS$BRANCH                         000100A8
DS$BREAK                          00010058
DS$CANWAIT                        00010070
DS$CHANNEL                        00010180
DS$CKLOOP                         00010040
DS$CLRVEC                         00010168
DS$CNTRLC                         00010078
DS$CVTREG                         000100B0
DS$ENDPASS                        00010010
DS$ENDSUB                         00010038
DS$ERRDEV                         000100C8
DS$ERRHARD                        000100D0
DS$ERRPREP                        00010108
DS$ERRSOFT                        000100D8
DS$ERRSYS                         000100C0
DS$ESCAPE                         00010050
DS$FREEDBGSYM                     00010188
DS$GETBUF                         00010120
DS$GETMEM                         00010130
DS$GPHARD                         00010018
DS$HELP                           000101B0
DS$INITSCB                        00010170
DS$INLOOP                         00010048
DS$K_ERROR                      = 00000002
DS$K_NORMAL                     = 00000001
DS$K_SEVERE                     = 00000004
DS$K_SUBSYS                     = 00000066
DS$K_WARNING                    = 00000000
DS$LOAD                           00010198
DS$LOADPCS                        000101C0
DS$MAPDBGBLOCK                    00010118
DS$MMOFF                          00010158
DS$MMON                           00010150
DS$MOVPHY                         00010148
DS$MOVVRT                         00010140
DS$PARSE                          000100B8
DS$PRINTB                         000100E0
DS$PRINTF                         000100F0
DS$PRINTS                         000100F8
DS$PRINTSIG                       00010100
DS$PRINTX                         000100E8
DS$PROBE                          000101A0
DS$RELBUF                         00010128
DS$RELMEM                         00010138
DS$SETIPL                         00010178
DS$SETMAP                         00010188
DS$SETPRIEXV                      00010110
DS$SETVEC                         00010160
DS$SHOCHAN                        00010190
DS$SUMMARY                        00010028
DS$WAITMS                         00010060
DS$WAITUS                         00010068
DS$_ARITH                       = 006600D0
DS$_BADLINK                     = 006600F0
DS$_BADTYPE                     = 006600E8
DS$_CHME                        = 006600A8
DS$_CHMK                        = 006600E0
DS$_DEVNAME                     = 00660108
DS$_ERROR                       = 00660002
DS$_FHWE                        = 00660068
DS$_FRAGBUF                     = 00660080
DS$_ICBUSY                      = 006600C8
DS$_ICERR                       = 006600C0
DS$_IHWE                        = 00660060
DS$_ILLCHAR                     = 00660018
DS$_ILLPAGCNT                   = 00660078
DS$_ILLUNIT                     = 00660100
DS$_INSFMEM                     = 00660050
DS$_IPL2HI                      = 006600B8
DS$_IVADDR                      = 00660040
DS$_IVVECT                      = 00660038
DS$_KRNLSTK                     = 00660090
DS$_LOGIC                       = 00660070
```

```
DS$_MCHK                        = 00660088
DS$_MMOFF                       = 00660058
DS$_NEEDUNIT                    = 006600F8
DS$_NOPCS                       = 00660110
DS$_NORMAL                      = 00660001
DS$_NOSUPPORT                   = 006600B1
DS$_NOTDON                      = 00660030
DS$_NOTIMP                      = 006600B0
DS$_NULLSTR                     = 00660010
DS$_OVERFLOW                    = 00660008
DS$_POWER                       = 00660098
DS$_PROGERR                     = 00660020
DS$_SEVERE                      = 00660004
DS$_TRANSL                      = 006600A0
DS$_TRUNCATE                    = 00660028
DS$_UNEXPINT                    = 006600D8
DS$_VASFULL                     = 00660048
DS$_WARNING                     = 00660000
DSA$AL_APTMAIL                    0000FE00
DSA$AT_APTTXT                     0000FA00
DSA$GL_APTCOM                     0000FE04
DSA$GL_DEVLEN                     0000FE58
DSA$GL_ERRNO                      0000FE44
DSA$GL_EVENT                      0000FE48
DSA$GL_FLAGS                      0000FE00
DSA$GL_MSGTYP                     0000FE40
DSA$GL_PASSES                     0000FE08
DSA$GL_PASSNO                     0000FE54
DSA$GL_SECTNO                     0000FE10
DSA$GL_SID                        0000FE14
DSA$GL_SUBTNO                     0000FE4C
DSA$GL_TESTNO                     0000FE50
DSA$GL_UNITS                      0000FE0C
DSA$GQ_MSGPTR                     0000FE68
DSA$GT_DEVNAM                     0000FE5C
DSA$M_APT                       = 80000000
DSA$M_BELL                      = 00000008
DSA$M_BINARY                    = 00000002
DSA$M_CRD_AUTOTEST              = 00000800
DSA$M_CRD_MENUTEST              = 00010000
DSA$M_CRD_TRACE                 = 00020000
DSA$M_DEBUG                     = 04000000
DSA$M_HALT                      = 00000001
DSA$M_IE1                       = 00000010
DSA$M_IE2                       = 00000020
DSA$M_IE3                       = 00000040
DSA$M_IES                       = 00000080
DSA$M_LOAD_DEBUGGER             = 40000000
DSA$M_LOOP                      = 00000004
DSA$M_NORPT                     = 08000000
DSA$M_OPER                      = 00001000
DSA$M_PASS0                     = 20000000
DSA$M_PROMPT                    = 00002000
DSA$M_QA                        = 00008000
DSA$M_QUICK                     = 00000100
DSA$M_SEARCH                    = 00004000
DSA$M_TRACE                     = 00000400
DSA$M_USER                      = 10000000
DSA$M_VERIFY                    = 00000200
DSA$V_APT                       = 0000001F
DSA$V_BELL                      = 00000003
DSA$V_BINARY                    = 00000001
DSA$V_CRD_AUTOTEST              = 0000000B
DSA$V_CRD_MENUTEST              = 00000010
DSA$V_CRD_TRACE                 = 00000011
DSA$V_DEBUG                     = 0000001A
DSA$V_HALT                      = 00000000
DSA$V_IE1                       = 00000004
DSA$V_IE2                       = 00000005
DSA$V_IE3                       = 00000006
DSA$V_IES                       = 00000007
DSA$V_LOAD_DEBUGGER             = 0000001E
DSA$V_LOOP                      = 00000002
DSA$V_NORPT                     = 00000018
DSA$V_OPER                      = 0000000C
DSA$V_PASS0                     = 0000001D
DSA$V_PROMPT                    = 0000000D
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
ERR$_MSGADR                     = 0000000C
ERR$_NARGS                      = 0000000A
ERR$_NUM                        = 00000004
ERR$_P1                         = 00000014
ERR$_P2                         = 00000018
ERR$_P3                         = 0000001C
ERR$_P4                         = 00000020
ERR$_P5                         = 00000024
ERR$_P6                         = 00000028
ERR$_POINTER                    = 00000010
ERR$_UNIT                       = 00000008
ERROR_CHECK                       00000744 R      02
ERR_MESSAGE                       000004B0 R      02
GPR_TABLE                         ******** X    02
HALT                            = 00000004   G
HP$A_DEPENDENT                    00000032
HP$A_DEVICE                       00000018
```

```
HP$A_DVA                  0000001C
HP$A_LINK                 00000020
HP$B_DRIVE                0000000B
HP$B_FLAGS                0000000A
HP$M_ALLOC              = 00000001
HP$M_WASALL             = 00000002
HP$Q_DEVICE               00000000
HP$T_DEVICE               0000000C
HP$T_TYPE                 00000026
HP$V_ALLOC              = 00000000
HP$V_WASALL             = 00000001
HP$W_SIZE                 00000008
HP$W_VECTOR               00000024
IPR                     = 00000006   G
KA$B_ACC_TYPE             00000042
KA$B_RESERVED             00000043
KA$B_SID_TYPE             0000003D
KA$K_LEN                  00000046
KA$L_FLAGS                00000032
KA$L_SID                  0000003A
KA$M_CHAR               = 00000100
KA$M_CRC                = 00000010
KA$M_D_FLOAT            = 00000002
KA$M_EDIT               = 00000080
KA$M_F_FLOAT            = 00000001
KA$M_G_FLOAT            = 00000004
KA$M_H_FLOAT            = 00000008
KA$M_PACKED             = 00000020
KA$M_PACK_MUL           = 00000040
KA$M_SB_ERR             = 02000000
KA$M_TODR               = 00010000
KA$M_WCS_LD             = 01000000
KA$V_CHAR               = 00000008
KA$V_CRC                = 00000004
KA$V_D_FLOAT            = 00000001
KA$V_EDIT               = 00000007
KA$V_F_FLOAT            = 00000000
KA$V_G_FLOAT            = 00000002
KA$V_H_FLOAT            = 00000003
KA$V_PACKED             = 00000005
KA$V_PACK_MUL           = 00000006
KA$V_SB_ERR             = 00000019
KA$V_TODR               = 00000010
KA$V_WCS_LD             = 00000018
KA$W_LATENCY              0000003E
KA$W_MEM_SIZE             00000044
KA$W_PROGRESS             00000040
KA$W_RESERVED             00000038
KA$W_WCS                  00000036
KA_PTABLE                 ********     X    04
KA_UNIT_NO                ********     X    04
LDPCTX                  = 0000000B   G
LOAD_GPR                  000006C3 R        02
L_ACT_ISP                 ********     X    02
L_ACT_PSL                 ********     X    02
L_ACT_SP                  ********     X    02
L_ADDRESS                 00000004 R        02
L_COUNT                   00000000 R        02
L_CUR_ISP                 00000008 R        02
L_DATA                  = 00000000
L_EXP_AP                  ********     X    02
L_EXP_FP                  ********     X    02
L_EXP_ISP                 ********     X    02
L_EXP_MOD                 00000010 R        02
L_EXP_PSL                 ********     X    02
L_EXP_SP                  ********     X    02
L_EXP_STACK               00000014 R        02
L_FLAGS                   00000020 R        02
L_KSP                     ********     X    04
L_TIMER                   0000001C R        02
MANUAL                  = 00000001   G
MCHK                    = 00000005   G
MEM                     = 00000009   G
MODELIST                  ********     X    02
MSG_005                   00000002 R        04
M_BIT19                 = 00080000
M_BIT20                 = 00100000
M_DONE                  = 00000002
M_ERROR_FLAG            = 00000080
M_EXECUTE_ERR           = 00000004
M_FATAL_ERR             = 00000010
M_GPR_ERR               = 00000040
M_INT                   = 00002000
M_ISP_ERR               = 00000200
M_MCINT                 = 00004000
M_MC_ERR                = 00001000
M_PSL_ERR               = 00000020
M_READY                 = 00000001
M_SP_ERR                = 00000400
M_TIMER                 = 00000008
PAR$M_ATDEF             = 00000010
PAR$M_ATHI              = 00000008
PAR$M_ATLO              = 00000004
PAR$M_NODEF             = 00000002
PAR$V_ATDEF             = 00000004
PAR$V_ATHI              = 00000003
PAR$V_ATLO              = 00000002
PAR$V_NODEF             = 00000001
PAR$_BIN                = 00000002
PAR$_DEC                = 0000000A
PAR$_HEX                = 00000010
PAR$_NO                 = 00000000
PAR$_OCT                = 00000008
PAR$_YES                = 00000001
POWER                   = 00000003   G
PR$_ISP                   ********     X    02
PR$_MCESR                 ********     X    02
PSL$C_EXEC              = 00000001
PSL$C_KERNEL            = 00000000
PSL$C_SUPER             = 00000002
PSL$C_USER              = 00000003
PSL$K_EXEC              = 00000001
PSL$K_KERNEL            = 00000000
PSL$K_SUPER             = 00000002
```

```
PSL$K_USER                      = 00000003
PSL$M_C                         = 00000001
PSL$M_CBIT                      = 00000001
PSL$M_CM                        = 80000000
PSL$M_CURMOD                    = 03000000
PSL$M_DV                        = 00000080
PSL$M_FPD                       = 08000000
PSL$M_FU                        = 00000040
PSL$M_IPL                       = 001F0000
PSL$M_IS                        = 04000000
PSL$M_IV                        = 00000020
PSL$M_N                         = 00000008
PSL$M_NBIT                      = 00000008
PSL$M_PRVMOD                    = 00C00000
PSL$M_SAFBITS                   = 000037FF
PSL$M_TBIT                      = 00000010
PSL$M_TP                        = 40000000
PSL$M_V                         = 00000002
PSL$M_VBIT                      = 00000002
PSL$M_Z                         = 00000004
PSL$M_ZBIT                      = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CBIT                      = 00000000
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_NBIT                      = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_VBIT                      = 00000001
PSL$V_Z                         = 00000002
PSL$V_ZBIT                      = 00000002
READ_GPR                        000006EC R      02
REG_TABLE                       ******** X      02
RENTRY1                         000000CD R      04
RENTRY2                         000001AC R      04
RENTRY3                         0000028B R      04
RENTRY4                         0000036A R      04
RENTRY_POINT                    0000000C R      02
SCB$L_ACCESS                    00000020
SCB$L_ARITH                     00000034
SCB$L_BREAK                     0000002C
SCB$L_CHME                      00000044
SCB$L_CHMK                      00000040
SCB$L_CHMS                      00000048
SCB$L_CHMU                      0000004C
SCB$L_COMPAT                    00000030
SCB$L_KNLSTK                    00000008
SCB$L_MACHCHK                   00000004
SCB$L_OPCCUS                    00000014
SCB$L_OPCDEC                    00000010
SCB$L_POWER                     0000000C
SCB$L_RADRMOD                   0000001C
SCB$L_ROPRAND                   00000018
SCB$L_RXDB                      000000F8
SCB$L_SFTLVL1                   00000084
SCB$L_SFTLVL10                  000000A8
SCB$L_SFTLVL11                  000000AC
SCB$L_SFTLVL12                  000000B0
SCB$L_SFTLVL13                  000000B4
SCB$L_SFTLVL14                  000000B8
SCB$L_SFTLVL15                  000000BC
SCB$L_SFTLVL2                   00000088
SCB$L_SFTLVL3                   0000008C
SCB$L_SFTLVL4                   00000090
SCB$L_SFTLVL5                   00000094
SCB$L_SFTLVL6                   00000098
SCB$L_SFTLVL7                   0000009C
SCB$L_SFTLVL8                   000000A0
SCB$L_SFTLVL9                   000000A4
SCB$L_TBIT                      00000028
SCB$L_TIMER                     000000C0
SCB$L_TRANSL                    00000024
SCB$L_TXDB                      000000FC
SCB$L_ZERO                      00000000
SEP_FUNCTIONAL                  = 00000002
SEP_REPAIR                      = 00000003
SIZ...                          = 00000001
SYS$ALLOC                       00010238
SYS$ASCTIM                      00010248
SYS$ASSIGN                      00010250
SYS$BINTIM                      00010258
SYS$CANCEL                      00010260
SYS$CANTIM                      00010268
SYS$CLOSE                       000105B8
SYS$CLREF                       00010298
SYS$CONNECT                     000105C0
SYS$DALLOC                      000102D8
SYS$DASSGN                      000102E0
SYS$DISCONNECT                  000105D0
SYS$FAO                         00010350
SYS$FAOL                        00010358
SYS$GET                         00010580
SYS$GETCHN                      000104C8
SYS$GETTIM                      00010378
SYS$LKWSET                      000103A0
SYS$NUMTIM                      000103B8
SYS$OPEN                        00010608
SYS$QIO                         000103C8
SYS$QIOW                        00010200
SYS$READ                        00010590
SYS$READEF                      000103D0
SYS$SETAST                      000103F8
```

```
SYS$SETEF                       00010400
SYS$SETIMR                      00010420
SYS$SETPRI                      00010428
SYS$SETPRT                      00010430
SYS$SETRWM                      00010438
SYS$SETSWM                      00010448
SYS$ULKPAG                      00010460
SYS$ULWSET                      00010468
SYS$UNWIND                      00010470
SYS$WAITFR                      00010478
SYS$WFLAND                      00010488
SYS$WFLOR                       00010490
T5_S1                           00000056 RG     04
T5_S1_X                         0000012A R      04
T5_S2                           00000135 RG     04
T5_S2_X                         00000209 R      04
T5_S3                           00000214 RG     04
T5_S3_X                         000002E8 R      04
T5_S4                           000002F3 RG     04
T5_S4_X                         000003C7 R      04
TABLE1                          00000430 R      02
TABLE2                          00000450 R      02
TABLE3                          00000470 R      02
TABLE4                          00000490 R      02
TAG_1                           000000BB R      04
TAG_2                           0000019A R      04
TAG_3                           00000279 R      04
TAG_4                           00000358 R      04
TEST_005                        00000020 RG     04
TEST_005_X                      000003E2 RG     04
TIMER_SERVICE                   00000830 R      02
TU58                          = 00000002   G
T_ACT_PSL                       00000364 R      02
T_BAD_DATA                      000002B2 R      02
T_BUSERR                        00000089 R      02
T_EXECUTE                       000001D2 R      02
T_EXPECTED                      0000015E R      02
T_EXP_PSL                       00000343 R      02
T_GOOD_DATA                     000002DB R      02
T_GPR                           0000039C R      02
T_GPR_ERR                       00000289 R      02
T_ILLIO_ERR                     000000DE R      02
T_ILLUB_ERR                     00000104 R      02
T_ISP_ERR                       0000021F R      02
T_MC_ERR                        0000012D R      02
T_MC_MESS                       00000060 R      02
T_NOT_ON_IS                     0000031C R      02
T_NOT_ON_KS                     000002F8 R      02
T_NOT_READY                     000001A0 R      02
T_NOXOR                         000003E0 R      02
T_NO_KA730                      00000028 R      02
T_NO_WCS                        000003F6 R      02
T_PSL                           00000383 R      02
T_SP_ERR1                       00000253 R      02
T_SP_ERR2                       0000027D R      02
T_TIMER                         000001F7 R      02
T_UNALIO_ERR                    000000B6 R      02
T_UNEXP_DATA                    0000016B R      02
T_XOR                           000003C4 R      02
UBI                           = 0000000A   G
V_DONE                        = 00000001
V_ERROR_FLAG                  = 00000007
V_EXECUTE_ERR                 = 00000002
V_FATAL_ERR                   = 00000004
V_GPR_ERR                     = 00000006
V_INT                         = 0000000D
V_ISP_ERR                     = 00000009
V_MCINT                       = 0000000E
V_MC_ERR                      = 0000000C
V_PSL_ERR                     = 00000005
V_READY                       = 00000000
V_SP_ERR                      = 0000000A
V_TIMER                       = 00000003
WCS                           = 00000007   G
```

+----------------+
' Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXF	0000084E (2126.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_005	000003EA (1002.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000018 (24.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	00000030 (48.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
$TSTCNT	00000004 (4.)	07 (7.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION		REFERENCES...							
$$$	=00000024-R	759	(13)	533	(10)	609	(11)	684	(12)	759	(13)
$$ARGS	=0000000A	75	(2)	75	(2)						
$$E	=00000001	759	(13)	560	(10)	636	(11)	711	(12)	786	(13)
$$M	=00000006	780	(13)	477	(9)	554	(10)	630	(11)	705	(12)
				780	(13)						
$$N	=00000001	780	(13)	#-292	(7)	#-293	(7)	#-295	(7)	#-299	(7)
				#-302	(7)	#-304	(7)	306	(7)	308	(7)
				310	(7)	316	(7)	320	(7)	325	(7)
				#-327	(7)	335	(7)	337	(7)	#-340	(7)
				#-481	(9)	554	(10)	630	(11)	705	(12)
				780	(13)	80	(2)				
$$S	=FFFFFFFF	789	(13)	467	(9)	477	(9)	488	(10)	533	(10)
				564	(11)	609	(11)	639	(12)	684	(12)
				714	(13)	759	(13)				
$$T1	=0000002C	75	(2)	75	(2)						
$ENV	=00000001	58	(2)								
$ER	=FFFFFFFF	789	(13)	533	(10)	#-554	(10)	609	(11)	#-630	(11)
				684	(12)	#-705	(12)	759	(13)	#-780	(13)
$MO	=00000001	790	(13)	790	(13)						
S	=00000024-R	759	(13)	533	(10)	560	(10)	609	(11)	636	(11)
				684	(12)	711	(12)	759	(13)	786	(13)
$ST	=00000000	789	(13)	477	(9)	488	(10)	533	(10)	554	(10)
				560	(10)	564	(11)	609	(11)	630	(11)
				636	(11)	639	(12)	684	(12)	705	(12)
				711	(12)	714	(13)	759	(13)	780	(13)
				786	(13)	789	(13)				
$TN	=00000005	790	(13)	467	(9)	477	(9)	482	(9)	488	(10)
				554	(10)	564	(11)	630	(11)	639	(12)
				705	(12)	714	(13)	780	(13)	789	(13)
				790	(13)						
ALL	=00000008	80	(2)	477	(9)						
A_ILL_IO	=00F2A000	130	(2)	267	(6)	#-695	(12)				
A_ILL_UB	=00FC0000	131	(2)	277	(6)	#-770	(13)				
A_MCHK_BUFFER	00000000-XR			#-296	(7)	#-329	(7)	#-371	(8)	#-372	(8)
				#-402	(8)	456	(8)	#-544	(10)	#-620	(11)
				#-695	(12)	#-770	(13)				
A_NXM	=00500200	128	(2)	247	(6)	#-544	(10)				
A_PSLMNE	00000000-XR			315	(7)	319	(7)	324	(7)		
A_UNAL_IO	=00F00001	129	(2)	257	(6)	#-620	(11)				
A_WCS_ADDRESS	00000024-R	135	(2)								
BASE_005	00000000-R	467	(9)	477	(9)						
BIT...	=0000001A	79	(2)	58	(2)	72	(2)	73	(2)	76	(2)
				77	(2)	78	(2)	79	(2)		
BUFSIZ	=00000080	127	(2)	#-290	(7)	#-315	(7)	#-319	(7)	#-324	(7)
CEP_FUNCTIONAL	=00000000	58	(2)								
CEP_REPAIR	=00000001	58	(2)	58	(2)						
CLEAR_BUFFER	00000836-R	455	(8)	535	(10)	611	(11)	686	(12)	761	(13)
CPU$V_COMET	=00000002	73	(2)								
CPU$V_HS	=00000000	73	(2)								
CPU$V_STAR	=00000001	73	(2)								

```
DATA_005                        0000001C-R  477    (9)      477    (9)
DATA_1                          0000042C-R  235    (4)   #-541    (10)   #-617    (11)   #-692    (12)   #-767    (13)
DEFAULT                         =00000000   80     (2)      477    (9)
DS$ABORT                        00010020    74     (2)      556    (10)      558    (10)      632    (11)      634    (11)
                                                            707    (12)      709    (12)      782    (13)      784    (13)
DS$ASKADR                       00010090    74     (2)
DS$ASKDATA                      00010080    74     (2)
DS$ASKLGCL                      00010098    74     (2)
DS$ASKSTR                       000100A0    74     (2)
DS$ASKVLD                       00010088    74     (2)
DS$ATTACH                       000101A8    74     (2)
DS$BGNSUB                       00010030    74     (2)      533    (10)      609    (11)      684    (12)      759    (13)
DS$BRANCH                       000100A8    74     (2)
DS$BREAK                        00010058    74     (2)      789    (13)
DS$CANWAIT                      00010070    74     (2)
DS$CHANNEL                      00010180    74     (2)
DS$CKLOOP                       00010040    74     (2)
DS$CLRVEC                       00010168    74     (2)      549    (10)      625    (11)      700    (12)      775    (13)
                                                            788    (13)
DS$CNTRLC                       00010078    74     (2)
DS$CVTREG                       000100B0    74     (2)      315    (7)       319    (7)       324    (7)
DS$ENDPASS                      00010010    74     (2)
DS$ENDSUB                       00010038    74     (2)      560    (10)      636    (11)      711    (12)      786    (13)
DS$ERRDEV                       000100C8    74     (2)
DS$ERRHARD                      000100D0    74     (2)      554    (10)      630    (11)      705    (12)      780    (13)
DS$ERRPREP                      00010108    74     (2)
DS$ERRSOFT                      000100D8    74     (2)
DS$ERRSYS                       000100C0    74     (2)
DS$ESCAPE                       00010050    74     (2)
DS$FREEDBGSYM                   000101B8    74     (2)
DS$GETBUF                       00010120    74     (2)
DS$GETMEM                       00010130    74     (2)
DS$GPHARD                       00010018    74     (2)
DS$HELP                         000101B0    74     (2)
DS$INITSCB                      00010170    74     (2)
DS$INLOOP                       00010048    74     (2)
DS$K_ERROR                      =00000002   72     (2)
DS$K_NORMAL                     =00000001   72     (2)
DS$K_SEVERE                     =00000004   72     (2)
DS$K_SUBSYS                     =00000066   72     (2)      72     (2)
DS$K_WARNING                    =00000000   72     (2)
DS$LOAD                         00010198    74     (2)
DS$LOADPCS                      000101C0    74     (2)
DS$MAPDBGBLOCK                  00010118    74     (2)
DS$MMOFF                        00010158    74     (2)
DS$MMON                         00010150    74     (2)
DS$MOVPHY                       00010148    74     (2)
DS$MOVVRT                       00010140    74     (2)
DS$PARSE                        000100B8    74     (2)
DS$PRINTB                       000100E0    74     (2)      292    (7)       293    (7)       295    (7)       299    (7)
                                                            302    (7)       304    (7)       308    (7)       316    (7)
                                                            320    (7)       327    (7)       335    (7)       340    (7)
DS$PRINTF                       000100F0    74     (2)      481    (9)
DS$PRINTS                       000100F8    74     (2)
DS$PRINTSIG                     00010100    74     (2)      298    (7)       300    (7)
DS$PRINTX                       000100E8    74     (2)      306    (7)       310    (7)       325    (7)       337    (7)
DS$PROBE                        000101A0    74     (2)
```

```
DS$RELBUF                       00010128      74      (2)
DS$RELMEM                       00010138      74      (2)
DS$SETIPL                       00010178      74      (2)
DS$SETMAP                       00010188      74      (2)
DS$SETPRIEXV                    00010110      74      (2)
DS$SETVEC                       00010160      74      (2)     484     (9)     536     (10)    612     (11)    687     (12)
                                                              762     (13)
DS$SHOCHAN                      00010190      74      (2)
DS$SUMMARY                      00010028      74      (2)
DS$WAITMS                       00010060      74      (2)
DS$WAITUS                       00010068      74      (2)
DS$_ARITH                       =006600D0     72      (2)
DS$_BADLINK                     =006600F0     72      (2)
DS$_BADTYPE                     =006600E8     72      (2)
DS$_CHME                        =006600A8     72      (2)
DS$_CHMK                        =006600E0     72      (2)
DS$_DEVNAME                     =00660108     72      (2)
DS$_ERROR                       =00660002     72      (2)
DS$_FHWE                        =00660068     72      (2)
DS$_FRAGBUF                     =00660080     72      (2)
DS$_ICBUSY                      =006600C8     72      (2)
DS$_ICERR                       =006600C0     72      (2)
DS$_IHWE                        =00660060     72      (2)
DS$_ILLCHAR                     =00660018     72      (2)
DS$_ILLPAGCNT                   =00660078     72      (2)
DS$_ILLUNIT                     =00660100     72      (2)
DS$_INSFMEM                     =00660050     72      (2)
DS$_IPL2HI                      =006600B8     72      (2)
DS$_IVADDR                      =00660040     72      (2)
DS$_IVVECT                      =00660038     72      (2)
DS$_KRNLSTK                     =00660090     72      (2)
DS$_LOGIC                       =00660070     72      (2)
DS$_MCHK                        =00660088     72      (2)     244     (6)     254     (6)     264     (6)     274     (6)
DS$_MMOFF                       =00660058     72      (2)
DS$_NEEDUNIT                    =006600F8     72      (2)
DS$_NOPCS                       =00660110     72      (2)
DS$_NORMAL                      =00660001     72      (2)
DS$_NOSUPPORT                   =006600B1     72      (2)
DS$_NOTDON                      =00660030     72      (2)
DS$_NOTIMP                      =006600B0     72      (2)     72      (2)
DS$_NULLSTR                     =00660010     72      (2)
DS$_OVERFLOW                    =00660008     72      (2)
DS$_POWER                       =00660098     72      (2)
DS$_PROGERR                     =00660020     72      (2)
DS$_SEVERE                      =00660004     72      (2)
DS$_TRANSL                      =006600A0     72      (2)
DS$_TRUNCATE                    =00660028     72      (2)
DS$_UNEXPINT                    =006600D8     72      (2)
DS$_VASFULL                     =00660048     72      (2)
DS$_WARNING                     =00660000     72      (2)     72      (2)
DSA$AL_APTMAIL                   0000FE00     73      (2)
DSA$AT_APTTXT                    0000FA00     73      (2)
DSA$GL_APTCOM                    0000FE04     73      (2)
DSA$GL_DEVLEN                    0000FE58     73      (2)
DSA$GL_ERRNO                     0000FE44     73      (2)
DSA$GL_EVENT                     0000FE48     73      (2)
DSA$GL_FLAGS                     0000FE00     73      (2)
```

```
DSA$GL_MSGTYP                   0000FE40     73     (2)
DSA$GL_PASSES                   0000FE08     73     (2)
DSA$GL_PASSNO                   0000FE54     73     (2)
DSA$GL_SECTNO                   0000FE10     73     (2)
DSA$GL_SID                      0000FE14     73     (2)
DSA$GL_SUBTNO                   0000FE4C     73     (2)
DSA$GL_TESTNO                   0000FE50     73     (2)
DSA$GL_UNITS                    0000FE0C     73     (2)
DSA$GQ_MSGPTR                   0000FE68     73     (2)
DSA$GT_DEVNAM                   0000FE5C     73     (2)
DSA$M_APT                      =80000000     73     (2)
DSA$M_BELL                     =00000008     73     (2)
DSA$M_BINARY                   =00000002     73     (2)
DSA$M_CRD_AUTOTEST             =00000800     73     (2)
DSA$M_CRD_MENUTEST             =00010000     73     (2)
DSA$M_CRD_TRACE                =00020000     73     (2)
DSA$M_DEBUG                    =04000000     73     (2)
DSA$M_HALT                     =00000001     73     (2)
DSA$M_IE1                      =00000010     73     (2)
DSA$M_IE2                      =00000020     73     (2)
DSA$M_IE3                      =00000040     73     (2)
DSA$M_IES                      =00000080     73     (2)
DSA$M_LOAD_DEBUGGER            =40000000     73     (2)
DSA$M_LOOP                     =00000004     73     (2)
DSA$M_NORPT                    =08000000     73     (2)
DSA$M_OPER                     =00001000     73     (2)
DSA$M_PASS0                    =20000000     73     (2)
DSA$M_PROMPT                   =00002000     73     (2)
DSA$M_QA                       =00008000     73     (2)
DSA$M_QUICK                    =00000100     73     (2)
DSA$M_SEARCH                   =00004000     73     (2)
DSA$M_TRACE                    =00000400     73     (2)
DSA$M_USER                     =10000000     73     (2)
DSA$M_VERIFY                   =00000200     73     (2)
DSA$V_APT                      =0000001F     73     (2)
DSA$V_BELL                     =00000003     73     (2)
DSA$V_BINARY                   =00000001     73     (2)
DSA$V_CRD_AUTOTEST             =0000000B     73     (2)
DSA$V_CRD_MENUTEST             =00000010     73     (2)
DSA$V_CRD_TRACE                =00000011     73     (2)
DSA$V_DEBUG                    =0000001A     73     (2)
DSA$V_HALT                     =00000000     73     (2)
DSA$V_IE1                      =00000004     73     (2)
DSA$V_IE2                      =00000005     73     (2)
DSA$V_IE3                      =00000006     73     (2)
DSA$V_IES                      =00000007     73     (2)
DSA$V_LOAD_DEBUGGER            =0000001E     73     (2)
DSA$V_LOOP                     =00000002     73     (2)
DSA$V_NORPT                    =0000001B     73     (2)
DSA$V_OPER                     =0000000C     73     (2)
DSA$V_PASS0                    =0000001D     73     (2)
DSA$V_PROMPT                   =0000000D     73     (2)
DSA$V_QA                       =0000000F     73     (2)
DSA$V_QUICK                    -00000008     73     (2)
DSA$V_SEARCH                   =0000000E     73     (2)
DSA$V_TRACE                    =0000000A     73     (2)
DSA$V_USER                     =0000001C     73     (2)
```

```
DSA$V_VERIFY                    =00000009     73      (2)
ENV$M_DOMAIN                    =00000002     58      (2)
ENV$M_LEVEL                     =00000001     58      (2)
ENV$M_SUPER                     =000003FC     58      (2)
ENV$S_DOMAIN                    =00000001     58      (2)
ENV$S_LEVEL                     =00000001     58      (2)
ENV$S_SUPER                     =00000008     58      (2)
ENV$V_DOMAIN                    =00000001     58      (2)     58      (2)
ENV$V_LEVEL                     =00000000     58      (2)     58      (2)
ENV$V_SUPER                     =00000002     58      (2)
ENV$_CPU                        =00000000     58      (2)     58      (2)
ENV$_FUNCTIONAL                 =00000000     58      (2)     58      (2)
ENV$_REPAIR                     =00000001     58      (2)     58      (2)
ENV$_SUPER                      =00000001     58      (2)
ENV$_SYSTEM                     =00000001     58      (2)     58      (2)
ERR$_MSGADR                     =0000000C     75      (2)
ERR$_NARGS                      =0000000A     75      (2)
ERR$_NUM                        =00000004     75      (2)
ERR$_P1                         =00000014     75      (2)     293     (7)
ERR$_P2                         =00000018     75      (2)     300     (7)
ERR$_P3                         =0000001C     75      (2)
ERR$_P4                         =00000020     75      (2)
ERR$_P5                         =00000024     75      (2)
ERR$_P6                         =00000028     75      (2)
ERR$_POINTER                    =00000010     75      (2)
ERR$_UNIT                       =00000008     75      (2)
ERROR_CHECK                      00000744-R   399     (8)     552     (10)    628     (11)    703     (12)    778     (13)
ERR_MESSAGE                      000004B0-R   288     (7)     554     (10)    630     (11)    705     (12)    780     (13)
GPR_TABLE                        00000000-XR                  328     (7)     352     (8)     403     (8)
HALT                            =00000004     80      (2)
HP$A_DEPENDENT                   00000032     78      (2)     79      (2)
HP$A_DEVICE                      00000018     78      (2)
HP$A_DVA                         0000001C     78      (2)
HP$A_LINK                        00000020     78      (2)
HP$B_DRIVE                       0000000B     78      (2)
HP$B_FLAGS                       0000000A     78      (2)
HP$M_ALLOC                      =00000001     78      (2)
HP$M_WASALL                     =00000002     78      (2)
HP$Q_DEVICE                      00000000     78      (2)
HP$T_DEVICE                      0000000C     78      (2)
HP$T_TYPE                        00000026     78      (2)
HP$V_ALLOC                      =00000000     78      (2)
HP$V_WASALL                     =00000001     78      (2)
HP$W_SIZE                        00000008     78      (2)
HP$W_VECTOR                      00000024     78      (2)
IPR                             =00000006     80      (2)
KA$B_ACC_TYPE                    00000042     79      (2)
KA$B_RESERVED                    00000043     79      (2)
KA$B_SID_TYPE                    0000003D     79      (2)
KA$K_LEN                         00000046     79      (2)
KA$L_FLAGS                       00000032     79      (2)
KA$L_SID                         0000003A     79      (2)
KA$M_CHAR                       =00000100     79      (2)
KA$M_CRC                        =00000010     79      (2)
KA$M_D_FLOAT                    =00000002     79      (2)
KA$M_EDIT                       =00000080     79      (2)
KA$M_F_FLOAT                    =00000001     79      (2)
```

```
KA$M_G_FLOAT                    =00000004      79      (2)
KA$M_H_FLOAT                    =00000008      79      (2)
KA$M_PACKED                     =00000020      79      (2)
KA$M_PACK_MUL                   =00000040      79      (2)
KA$M_SB_ERR                     =02000000      79      (2)
KA$M_TODR                       =00010000      79      (2)
KA$M_WCS_LD                     =01000000      79      (2)
KA$V_CHAR                       =00000008      79      (2)
KA$V_CRC                        =00000004      79      (2)
KA$V_D_FLOAT                    =00000001      79      (2)
KA$V_EDIT                       =00000007      79      (2)
KA$V_F_FLOAT                    =00000000      79      (2)
KA$V_G_FLOAT                    =00000002      79      (2)
KA$V_H_FLOAT                    =00000003      79      (2)
KA$V_PACKED                     =00000005      79      (2)
KA$V_PACK_MUL                   =00000006      79      (2)
KA$V_SB_ERR                     =00000019      79      (2)
KA$V_TODR                       =00000010      79      (2)
KA$V_WCS_LD                     =00000018      79      (2)
KA$W_LATENCY                     0000003E      79      (2)
KA$W_MEM_SIZE                    00000044      79      (2)
KA$W_PROGRESS                    00000040      79      (2)
KA$W_RESERVED                    00000038      79      (2)
KA$W_WCS                         00000036      79      (2)
KA_PTABLE                        00000000-XR                   #-479   (9)
KA_UNIT_NO                       00000000-XR                   #-554   (10)  #-630   (11)  #-705   (12)  #-780   (13)
LDPCTX                          =0000000B      80      (2)
LOAD_GPR                         000006C3-R    351     (8)     542   (10)    618   (11)    693   (12)    768   (13)
L_ACT_ISP                        00000000-XR                   #-306   (7)   #-382   (8)   #-419   (8)
L_ACT_PSL                        00000000-XR                   #-319   (7)   #-320   (7)   #-383   (8)   #-422   (8)
                                                                 427   (8)     432   (8)
L_ACT_SP                         00000000-XR                   #-308   (7)   #-310   (7)   #-381   (8)   #-416   (8)
L_ADDRESS                        00000004-R     88      (2)
L_COUNT                          00000000-R     87      (2)   #-457   (8)   #-459   (8)
L_CUR_ISP                        00000008-R     89      (2)
L_DATA                          =00000000      132     (2)
L_EXP_AP                         00000000-XR                   #-359   (8)
L_EXP_FP                         00000000-XR                   #-360   (8)
L_EXP_ISP                        00000000-XR                   #-306   (7)   #-419   (8)   #-540   (10)  #-616   (11)
                                                               #-691   (12)  #-766   (13)
L_EXP_MOD                        00000010-R     91      (2)   #-425   (8)   #-427   (8)
L_EXP_PSL                        00000000-XR                   #-315   (7)   #-316   (7)   #-324   (7)   #-325   (7)
                                                               #-422   (8)     425   (8)     430   (8)   #-541   (10)
                                                               #-617   (11)  #-692   (12)  #-767   (13)
L_EXP_SP                         00000000-XR                   #-308   (7)   #-416   (8)   #-539   (10)  #-615   (11)
                                                               #-690   (12)  #-765   (13)
L_EXP_STACK                      00000014-R     92      (2)   #-430   (8)   #-432   (8)
L_FLAGS                          00000020-R     96      (2)     294   (7)     301   (7)     303   (7)     305   (7)
                                                                 307   (7)     311   (7)     326   (7)     339   (7)
                                                               #-406   (8)     408   (8)   #-414   (8)   #-418   (8)
                                                               #-421   (8)   #-434   (8)   #-435   (8)     436   (8)
                                                               #-437   (8)     438   (8)   #-439   (8)     440   (8)
                                                               #-441   (8)   #-449   (8)   #-538   (10)  #-543   (10)
                                                               #-546   (10)    553   (10)    557   (10)  #-614   (11)
                                                               #-619   (11)  #-622   (11)    629   (11)    633   (11)
                                                               #-689   (12)  #-694   (12)  #-697   (12)    704   (12)
                                                                 708   (12)  #-764   (13)  #-769   (13)  #-772   (13)
```

```
                                                    779     (13)     783     (13)

L_KSP                            00000000-XR                       #-485     (9)
L_TIMER                          0000001C-R    94      (2)
MANUAL                          =00000001      80      (2)
MCHK                            =00000005      80      (2)       477     (9)
MEM                             =00000009      80      (2)
MODELIST                         00000000-XR                      315     (7)      319     (7)      324     (7)
MSG_005                          00000002-R    467     (9)       477     (9)
M_BIT19                         =00080000      125     (2)
M_BIT20                         =00100000      126     (2)
M_DONE                          =00000002      101     (2)
M_ERROR_FLAG                    =00000080      113     (2)     #-406     (8)     #-414     (8)     #-418     (8)     #-421     (8)
                                                               #-435     (8)     #-437     (8)     #-439     (8)     #-441     (8)
M_EXECUTIVE_ERR                 =00000004      103     (2)     #-546     (10)    #-622     (11)    #-697     (12)    #-772     (13)
M_FATAL_ERR                     =00000010      107     (2)     #-434     (8)
M_GPR_ERR                       =00000040      111     (2)     #-406     (8)
M_INT                           =00002000      121     (2)
M_ISP_ERR                       =00000200      115     (2)     #-421     (8)
M_MCINT                         =00004000      123     (2)
M_MC_ERR                        =00001000      119     (2)     #-414     (8)
M_PSL_ERR                       =00000020      109     (2)     #-435     (8)
M_READY                         =00000001      99      (2)     #-543     (10)    #-619     (11)    #-694     (12)    #-769     (13)
M_SP_ERR                        =00000400      117     (2)     #-418     (8)
M_TIMER                         =00000008      105     (2)     #-449     (8)
PAR$M_ATDEF                     =00000010      76      (2)
PAR$M_ATHI                      =00000008      76      (2)
PAR$M_ATLO                      =00000004      76      (2)
PAR$M_NODEF                     =00000002      76      (2)
PAR$V_ATDEF                     =00000004      76      (2)
PAR$V_ATHI                      =00000003      76      (2)
PAR$V_ATLO                      =00000002      76      (2)
PAR$V_NODEF                     =00000001      76      (2)
PAR$_BIN                        =00000002      76      (2)
PAR$_DEC                        =0000000A      76      (2)
PAR$_HEX                        =00000010      76      (2)
PAR$_NO                         =00000000      76      (2)
PAR$_OCT                        =00000008      76      (2)
PAR$_YES                        =00000001      76      (2)
POWER                           =00000003      80      (2)
PR$_ISP                          00000000-XR                      #-382     (8)     #-540     (10)    #-616     (11)    #-691     (12)
                                                                  #-766     (13)
PR$_MCESR                        00000000-XR                      #-370     (8)     #-534     (10)    #-610     (11)    #-685     (12)
                                                                  #-760     (13)
PSL$C_EXEC                      =00000001      77      (2)
PSL$C_KERNEL                    =00000000      77      (2)
PSL$C_SUPER                     =00000002      77      (2)
PSL$C_USER                      =00000003      77      (2)
PSL$K_EXEC                      =00000001      77      (2)
PSL$K_KERNEL                    =00000000      77      (2)
PSL$K_SUPER                     =00000002      77      (2)
PSL$K_USER                      =00000003      77      (2)
PSL$M_C                         =00000001      77      (2)
PSL$M_CBIT                      =00000001      77      (2)
PSL$M_CM                        =80000000      77      (2)       77      (2)
PSL$M_CURMOD                    =03000000      77      (2)
PSL$M_DV                        =00000080      77      (2)
PSL$M_FPD                       =08000000      77      (2)       77      (2)
```

```
PSL$M_FU                        =00000040      77      (2)
PSL$M_IPL                       =001F0000      77      (2)
PSL$M_IS                        =04000000      77      (2)
PSL$M_IV                        =00000020      77      (2)
PSL$M_N                         =00000008      77      (2)
PSL$M_NBIT                      =00000008      77      (2)
PSL$M_PRVMOD                    =00C00000      77      (2)
PSL$M_SAFBITS                   =000037FF      77      (2)
PSL$M_TBIT                      =00000010      77      (2)
PSL$M_TP                        =40000000      77      (2)       77      (2)
PSL$M_V                         =00000002      77      (2)
PSL$M_VBIT                      =00000002      77      (2)
PSL$M_Z                         =00000004      77      (2)
PSL$M_ZBIT                      =00000004      77      (2)
PSL$S_CURMOD                    =00000002      77      (2)     #-424   (8)     #-426   (8)
PSL$S_IPL                       =00000005      77      (2)
PSL$S_PRVMOD                    =00000002      77      (2)
PSL$V_C                         =00000000      77      (2)
PSL$V_CBIT                      =00000000      77      (2)
PSL$V_CM                        =0000001F      77      (2)
PSL$V_CURMOD                    =00000018      77      (2)     #-424   (8)     #-426   (8)
PSL$V_DV                        =00000007      77      (2)
PSL$V_FPD                       =0000001B      77      (2)
PSL$V_FU                        =00000006      77      (2)
PSL$V_IPL                       =00000010      77      (2)
PSL$V_IS                        =0000001A      77      (2)     #-429   (8)     #-431   (8)
PSL$V_IV                        =00000005      77      (2)
PSL$V_N                         =00000003      77      (2)
PSL$V_NBIT                      =00000003      77      (2)
PSL$V_PRVMOD                    =00000016      77      (2)
PSL$V_TBIT                      =00000004      77      (2)
PSL$V_TP                        =0000001E      77      (2)
PSL$V_V                         =00000001      77      (2)
PSL$V_VBIT                      =00000001      77      (2)
PSL$V_Z                         =00000002      77      (2)
PSL$V_ZBIT                      =00000002      77      (2)
READ_GPR                         000006EC-R    369     (8)       536   (10)      612   (11)      687   (12)      762   (13)
REG_TABLE                        00000000-XR                     332   (7)
RENTRY1                          000000CD-R    548     (10)    #-537   (10)
RENTRY2                          000001AC-R    624     (11)    #-613   (11)
RENTRY3                          0000028B-R    699     (12)    #-688   (12)
RENTRY4                          0000036A-R    774     (13)    #-763   (13)
RENTRY_POINT                     0000000C-R    90      (2)       389   (8)     #-537   (10)    #-613   (11)    #-688   (12)
                                                                #-763   (13)
SCB$L_ACCESS                     00000020      71      (2)
SCB$L_ARITH                      00000034      71      (2)
SCB$L_BREAK                      0000002C      71      (2)
SCB$L_CHME                       00000044      71      (2)
SCB$L_CHMK                       00000040      71      (2)
SCB$L_CHMS                       00000048      71      (2)
SCB$L_CHMU                       0000004C      71      (2)
SCB$L_COMPAT                     00000030      71      (2)
SCB$L_KNLSTK                     00000008      71      (2)
SCB$L_MACHCHK                    00000004      71      (2)     #-536   (10)    #-549   (10)    #-612   (11)    #-625   (11)
                                                                #-687   (12)    #-700   (12)    #-762   (13)    #-775   (13)
SCB$L_OPCCUS                     00000014      71      (2)
SCB$L_OPCDEC                     00000010      71      (2)
```

```
SCB$L_POWER                     0000000C      71      (2)
SCB$L_RADRMOD                   0000001C      71      (2)
SCB$L_ROPRAND                   00000018      71      (2)
SCB$L_RXDB                      000000F8      71      (2)
SCB$L_SFTLVL1                   00000084      71      (2)
SCB$L_SFTLVL10                  000000A8      71      (2)
SCB$L_SFTLVL11                  000000AC      71      (2)
SCB$L_SFTLVL12                  000000B0      71      (2)
SCB$L_SFTLVL13                  000000B4      71      (2)
SCB$L_SFTLVL14                  000000B8      71      (2)
SCB$L_SFTLVL15                  000000BC      71      (2)
SCB$L_SFTLVL2                   00000088      71      (2)
SCB$L_SFTLVL3                   0000008C      71      (2)
SCB$L_SFTLVL4                   00000090      71      (2)
SCB$L_SFTLVL5                   00000094      71      (2)
SCB$L_SFTLVL6                   00000098      71      (2)
SCB$L_SFTLVL7                   0000009C      71      (2)
SCB$L_SFTLVL8                   000000A0      71      (2)
SCB$L_SFTLVL9                   000000A4      71      (2)
SCB$L_TBIT                      00000028      71      (2)
SCB$L_TIMER                     000000C0      71      (2)   #-484     (9)     #-788     (13)
SCB$L_TRANSL                    00000024      71      (2)
SCB$L_TXDB                      000000FC      71      (2)
SCB$L_ZERO                      00000000      71      (2)
SEP_FUNCTIONAL                 =00000002      58      (2)
SEP_REPAIR                     =00000003      58      (2)
SIZ...                         =00000001      79      (2)      58      (2)      73      (2)      76      (2)      77      (2)
                                                                78      (2)      79      (2)
SYS$ALLOC                       00010238      74      (2)
SYS$ASCTIM                      00010248      74      (2)
SYS$ASSIGN                      00010250      74      (2)
SYS$BINTIM                      00010258      74      (2)
SYS$CANCEL                      00010260      74      (2)
SYS$CANTIM                      00010268      74      (2)
SYS$CLOSE                       000105B8      74      (2)
SYS$CLREF                       00010298      74      (2)
SYS$CONNECT                     000105C0      74      (2)
SYS$DALLOC                      000102D8      74      (2)
SYS$DASSGN                      000102E0      74      (2)
SYS$DISCONNECT                  000105D0      74      (2)
SYS$FAO                         00010350      74      (2)
SYS$FAOL                        00010358      74      (2)
SYS$GET                         00010580      74      (2)
SYS$GETCHN                      000104C8      74      (2)
SYS$GETTIM                      00010378      74      (2)
SYS$LKWSET                      000103A0      74      (2)
SYS$NUMTIM                      000103B8      74      (2)
SYS$OPEN                        00010608      74      (2)
SYS$QIO                         000103C8      74      (2)
SYS$QIOW                        00010200      74      (2)
SYS$READ                        00010590      74      (2)
SYS$READEF                      000103D0      74      (2)
SYS$SETAST                      000103F8      74      (2)
SYS$SETEF                       00010400      74      (2)
SYS$SETIMR                      00010420      74      (2)
SYS$SETPRI                      00010428      74      (2)
SYS$SETPRT                      00010430      74      (2)
```

```
SYS$SETRWM                      00010438      74      (2)
SYS$SETSWM                      00010448      74      (2)
SYS$ULKPAG                      00010460      74      (2)
SYS$ULWSET                      00010468      74      (2)
SYS$UNWIND                      00010470      74      (2)
SYS$WAITFR                      00010478      74      (2)
SYS$WFLAND                      00010488      74      (2)
SYS$WFLOR                       00010490      74      (2)
T5_S1                           00000056-R    533     (10)
T5_S1_X                         0000012A-R    560     (10)
T5_S2                           00000135-R    609     (11)
T5_S2_X                         00000209-R    636     (11)
T5_S3                           00000214-R    684     (12)
T5_S3_X                         000002E8-R    711     (12)
T5_S4                           000002F3-R    759     (13)
T5_S4_X                         000003C7-R    786     (13)
TABLE1                          00000430-R    243     (6)     #-297     (7)     #-551     (10)    #-554     (10)
TABLE2                          00000450-R    253     (6)     #-627     (11)    #-630     (11)
TABLE3                          00000470-R    263     (6)     #-702     (12)    #-705     (12)
TABLE4                          00000490-R    273     (6)     #-777     (13)    #-780     (13)
TAG_1                           000000BB-R    544     (10)      249     (6)
TAG_2                           0000019A-R    620     (11)      259     (6)
TAG_3                           00000279-R    695     (12)      269     (6)
TAG_4                           00000358-R    770     (13)      279     (6)
TEST_005                        00000020-R    477     (9)       477     (9)
TEST_005_X                      000003E2-R    789     (13)    #-482     (9)
TIMER_SERVICE                   00000830-R    448     (8)       484     (9)
TU58                            =00000002     80      (2)
T_ACT_PSL                       00000364-R    209     (3)       320     (7)
T_BAD_DATA                      000002B2-R    194     (3)
T_BUSERR                        00000089-R    151     (3)     #-554     (10)
T_EXECUTE                       000001D2-R    176     (3)       304     (7)
T_EXPECTED                      0000015E-R    167     (3)       299     (7)
T_EXP_PSL                       00000343-R    206     (3)       316     (7)
T_GOOD_DATA                     000002DB-R    197     (3)
T_GPR                           0000039C-R    215     (3)       327     (7)
T_GPR_ERR                       00000289-R    191     (3)
T_ILLIO_ERR                     000000DE-R    157     (3)     #-705     (12)
T_ILLUB_ERR                     00000104-R    160     (3)     #-780     (13)
T_ISP_ERR                       0000021F-R    182     (3)       306     (7)
T_MC_ERR                        0000012D-R    163     (3)       295     (7)
T_MC_MESS                       00000060-R    148     (3)       292     (7)
T_NOT_ON_IS                     0000031C-R    203     (3)
T_NOT_ON_KS                     000002F8-R    200     (3)
T_NOT_READY                     000001A0-R    173     (3)       302     (7)
T_NOXOR                         000003E0-R    221     (3)       337     (7)
T_NO_KA730                      00000028-R    145     (3)       481     (9)
T_NO_WCS                        000003F6-R    224     (3)
T_PSL                           00000383-R    212     (3)       325     (7)
T_SP_ERR1                       00000253-R    185     (3)       308     (7)
T_SP_ERR2                       0000027D-R    188     (3)       310     (7)
T_TIMER                         000001F7-R    179     (3)       340     (7)
T_UNALIO_ERR                    000000B6-R    154     (3)     #-630     (11)
T_UNEXP_DATA                    00000168-R    170     (3)
T_XOR                           000003C4-R    218     (3)       335     (7)
UBI                             =0000000A     80      (2)
V_DONE                          =00000001     100     (2)
```

```
V_ERROR_FLAG                    =00000007     112     (2)     #-553     (10)    #-629     (11)    #-704     (12)    #-779     (13)
V_EXECUTE_ERR                   =00000002     102     (2)     #-303     (7)     #-438     (8)
V_FATAL_ERR                     =00000004     106     (2)     #-557     (10)    #-633     (11)    #-708     (12)    #-783     (13)
V_GPR_ERR                       =00000006     110     (2)     #-326     (7)
V_INT                           =0000000D     120     (2)     #-408     (8)
V_ISP_ERR                       =00000009     114     (2)     #-305     (7)
V_MCINT                         =0000000E     122     (2)
V_MC_ERR                        =0000000C     118     (2)     #-294     (7)
V_PSL_ERR                       =00000005     108     (2)     #-311     (7)
V_READY                         =00000000     98      (2)     #-301     (7)     #-436     (8)
V_SP_ERR                        =0000000A     116     (2)     #-307     (7)
V_TIMER                         =00000003     104     (2)     #-339     (7)     #-440     (8)
WCS                             =00000007     80      (2)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_ABPROGRAM	1	556 (10)	556 (10) 558 (10) 632 (11) 634 (11) 707 (12) 709 (12) 782 (13) 784 (13)
$D1_BSUBT	1	533 (10)	533 (10) 609 (11) 684 (12) 759 (13)
$D1_BTEST	1	477 (9)	477 (9)
$D1_ERR	1	554 (10)	554 (10) 630 (11) 705 (12) 780 (13)
$D1_ERR_S	1	554 (10)	554 (10) 630 (11) 705 (12) 780 (13)
$D1_ESUBT	1	560 (10)	560 (10) 636 (11) 711 (12) 786 (13)
$D1_ETEST	1	789 (13)	789 (13)
$D1_EXTEST	1	482 (9)	482 (9)
$D1_PRINT_S	1	292 (7)	292 (7) 293 (7) 295 (7) 299 (7) 302 (7) 304 (7) 306 (7) 308 (7) 310 (7) 316 (7) 320 (7) 325 (7) 327 (7) 335 (7) 337 (7) 340 (7) 481 (9)
$D1_SBTTL	2	467 (9)	467 (9) 488 (10) 564 (11) 639 (12) 714 (13)
$D2_BDATA	1	477 (9)	477 (9)
$D2_BTEST	1	477 (9)	477 (9)
$D2_SBTTL	1	467 (9)	467 (9)
$DEF	1	79 (2)	71 (2) 73 (2) 74 (2) 78 (2) 79 (2)
$DEFEND	1	71 (2)	71 (2) 73 (2) 74 (2) 77 (2) 78 (2) 79 (2)
$DEFINI	1	71 (2)	71 (2) 73 (2) 74 (2) 77 (2) 78 (2) 79 (2)
$DS_ABORT	1	556 (10)	556 (10) 558 (10) 632 (11) 634 (11) 707 (12) 709 (12) 782 (13) 784 (13)
$DS_BCOMPLETE	1	555 (10)	555 (10) 631 (11) 706 (12) 781 (13)
$DS_BGNMESSAGE	1	289 (7)	289 (7)
$DS_BGNMOD	1	58 (2)	58 (2)
$DS_BGNSUB	1	533 (10)	533 (10) 609 (11) 684 (12) 759 (13)
$DS_BGNTEST	1	477 (9)	477 (9)
$DS_BREAK	1	789 (13)	789 (13)
$DS_CLRVEC_S	1	549 (10)	549 (10) 625 (11) 700 (12) 775 (13) 788 (13)
$DS_CVTREG_S	1	313 (7)	313 (7) 317 (7) 322 (7)
$DS_DSADEF	1	73 (2)	73 (2)
$DS_DSBDEF	1	73 (2)	
$DS_DSDEF	2	72 (2)	72 (2)
$DS_DSSDEF	1	74 (2)	74 (2)
$DS_ENDDATA	1	477 (9)	477 (9)
$DS_ENDMESSAGE	1	341 (7)	341 (7)
$DS_ENDMOD	1	790 (13)	790 (13)
$DS_ENDSUB	1	560 (10)	560 (10) 636 (11) 711 (12) 786 (13)
$DS_ENDTEST	1	789 (13)	789 (13)
$DS_ENVDEF	2	58 (2)	58 (2)
$DS_ERRDEF	1	75 (2)	75 (2)
$DS_ERRHARD_S	1	554 (10)	554 (10) 630 (11) 705 (12) 780 (13)
$DS_EXIT	1	482 (9)	482 (9)
$DS_HPODEF	2	78 (2)	78 (2)
$DS_KA_DEF	3	79 (2)	79 (2)
$DS_PAGE	1	465 (8)	465 (8) 486 (9) 562 (10) 637 (11) 712 (12)
$DS_PARDEF	1	76 (2)	76 (2)
$DS_PRINTB_S	1	292 (7)	292 (7) 293 (7) 295 (7) 299 (7) 302 (7)

				304	(7)	308	(7)	316	(7)	320	(7)	327	(7)
				335	(7)	340	(7)						
$DS_PRINTF_S	1	481	(9)	481	(9)								
$DS_PRINTSIG_G	1	298	(7)	298	(7)	300	(7)						
$DS_PRINTX_S	1	306	(7)	306	(7)	310	(7)	325	(7)	337	(7)		
$DS_PSLDEF	1	77	(2)	77	(2)								
$DS_SBTTL	1	467	(9)	467	(9)	488	(10)	564	(11)	639	(12)	714	(13)
$DS_SCBDEF	1	71	(2)	71	(2)								
$DS_SECDEF	1	80	(2)	80	(2)								
$DS_SETVEC_S	1	484	(9)	484	(9)	536	(10)	612	(11)	687	(12)	762	(13)
$EQU	1	79	(2)	58	(2)	72	(2)	76	(2)	77	(2)		
$EQULST	1	77	(2)	58	(2)	72	(2)	77	(2)				
$EQULST	1	58	(2)	58	(2)	72	(2)	77	(2)				
$GBLINI	2	58	(2)	58	(2)	71	(2)	72	(2)	73	(2)	74	(2)
				76	(2)	77	(2)	78	(2)	79	(2)		
$HP_DEFDEF	1	78	(2)										
$KADEF	1	79	(2)										
$OFFDEF	1	75	(2)	75	(2)								
$PSLDEF	1	77	(2)	77	(2)								
$PUSHADR	1	315	(7)	315	(7)	319	(7)	324	(7)	554	(10)	630	(11)
				705	(12)	780	(13)						
$VIELD	1	58	(2)	58	(2)	73	(2)	76	(2)	77	(2)	78	(2)
				79	(2)								
$VIELD1	1	79	(2)	58	(2)	73	(2)	76	(2)	77	(2)	78	(2)
				79	(2)								
ENKAX_SECDEF	1	80	(2)	80	(2)								

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	24	00:00:00.05	00:00:00.22
Command processing	25	00:00:00.23	00:00:00.65
Pass 1	1130	00:00:17.61	00:00:45.16
Symbol table sort	0	00:00:00.81	00:00:02.34
Pass 2	351	00:00:03.79	00:00:07.21
Symbol table output	75	00:00:00.42	00:00:00.69
Psect synopsis output	7	00:00:00.04	00:00:00.05
Cross-reference output	216	00:00:02.59	00:00:06.17
Assembler run totals	1832	00:00:25.55	00:01:02.49

The working set limit was 1000 pages.
142491 bytes (279 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 530 non-local and 47 local symbols.
791 source lines were read in Pass 1, producing 32 object records in Pass 2.
125 pages of virtual memory were used to define 56 macros.

Macro library name	Macro library statistics ! Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	47
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	8
TOTALS (all libraries)	56

866 GETS were required to define 56 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXF

Table of contents

```
0000     1          .SBTTL  ENKAXG: TU58 EXERCISER
0000     2          .TITLE  ENKAXG  VAX-11/730 Specific CPU Cluster Exerciser
0000     3          .IDENT  /1-3/
0000     4
0000     5 ;
0000     6 ; Copyright (C) 1981
0000     7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
0000     8 ;
0000     9 ; This software is furnished  under a license for use only on a single
0000    10 ; computer  system and  may be  copied only with  the inclusion of the
0000    11 ; above copyright notice.  This software, or any other copies thereof,
0000    12 ; may not be provided or  otherwise made available to any other person
0000    13 ; except for use on such system and to one who agrees to these license
0000    14 ; terms.  Title to and  ownership of the  software  shall at all times
0000    15 ; remain in DEC.
0000    16 ;
0000    17 ; The information in this software is subject to change without notice
0000    18 ; and should  not be construed  as a commitment  by Digital  Equipment
0000    19 ; Corporation.
0000    20 ;
0000    21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
0000    22 ; software on equipment which is not supplied by DEC.
0000    23 ;
0000    24
0000    25 ;++
0000    26 ; Facility:     VAX Architecture Diagnostic.
0000    27 ;
0000    28 ; Abstract:     This module exercises the TU58 tape controller to assure
0000    29 ;               basic functionality. A series of subtests, including the
0000    30 ;               controller's self test are executed checking initialize,
0000    31 ;               no operation, write, read, seek, write modified, read
0000    32 ;               modified, head switching and positioning. This test will
0000    33 ;               be modified when a decision is made whether to implement
0000    34 ;               the TU58 control microcode in silo or handshake mode.
0000    35 ;
0000    36 ;
0000    37 ; Environment:  VAX Diagnostic Supervisor.
0000    38 ;
0000    39 ; Author:       Rich Lewis      22-Oct-81       Version 2-X
0000    40 ;
0000    41 ;
0000    42 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
0000    43 ;               Peter Green
0000    44 ;
0000    45 ;               Added quick flag on Test 7(TU58). If quick flag is set
0000    46 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
0000    47 ;               If event flag 3 is set, will override protection,
0000    48 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
0000    49 ;               is clear & tape is not scratch, will report an error.
0000    50 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
0000    51 ;
0000    52 ;
0000    53 ;               Tai-Chun Pan    03-Aug-83       Version 1.3
0000    54 ;
0000    55 ;               fixed bugs on TEST 7 and error summary routine call when
0000    56 ;               running under APT.
0000    57 ;
```

```
                        0000     58 ;--
```

```
                                    0000     60 .SBTTL  DECLARATIONS
                                    0000     61         .LIST   MEB
                                    0000     62         .NLIST  CND
                                00000000     63         .PSECT  ENKAXG, PAGE, NOWRT
                                    0000     64         .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                                    0000     65         $DS_BGNMOD      CEP_REPAIR, IN=6
                                    0000        ;;;     Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                                    0000     66
                                    0000     67 ;
                                    0000     68 ; Include files:
                                    0000     69 ;
                                    0000     70
                                    0000     71         .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
%MACRO-E-MLBOPNERR, Error opening macro library        !
                                    0000
```

```
$$S             = FFFFFFFF
$ENV            = 00000001  G
$ER             = 00000001
$MO             = 00000000  G
$ST             = 00000000
$TN             = 00000006
BIT...          = 00000002
CEP_FUNCTIONAL  = 00000000
CEP_REPAIR      = 00000001
ENV$M_DOMAIN    = 00000002
ENV$M_LEVEL     = 00000001
ENV$M_SUPER     = 000003FC
ENV$S_DOMAIN    = 00000001
ENV$S_LEVEL     = 00000001
ENV$S_SUPER     = 00000008
ENV$V_DOMAIN    = 00000001
ENV$V_LEVEL     = 00000000
ENV$V_SUPER     = 00000002
ENV$_CPU        = 00000000
ENV$_FUNCTIONAL = 00000000
ENV$_REPAIR     = 00000001
ENV$_SUPER      = 00000001
ENV$_SYSTEM     = 00000001
SEP_FUNCTIONAL  = 00000002
SEP_REPAIR      = 00000003
SIZ...          = 00000008
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXG	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$S	=FFFFFFFF	65 (2)	
$ENV	=00000001	65 (2)	
$ER	=00000001	65 (2)	
$MO	=00000000	65 (2)	
$ST	=00000000	65 (2)	
$TN	=00000006	65 (2)	
BIT...	=00000002	65 (2)	65 (2)
CEP_FUNCTIONAL	=00000000	65 (2)	
CEP_REPAIR	=00000001	65 (2)	65 (2)
ENV$M_DOMAIN	=00000002	65 (2)	
ENV$M_LEVEL	=00000001	65 (2)	
ENV$M_SUPER	=000003FC	65 (2)	
ENV$S_DOMAIN	=00000001	65 (2)	
ENV$S_LEVEL	=00000001	65 (2)	
ENV$S_SUPER	=00000008	65 (2)	
ENV$V_DOMAIN	=00000001	65 (2)	65 (2)
ENV$V_LEVEL	=00000000	65 (2)	65 (2)
ENV$V_SUPER	=00000002	65 (2)	
ENV$_CPU	=00000000	65 (2)	65 (2)
ENV$_FUNCTIONAL	=00000000	65 (2)	65 (2)
ENV$_REPAIR	=00000001	65 (2)	65 (2)
ENV$_SUPER	=00000001	65 (2)	
ENV$_SYSTEM	=00000001	65 (2)	65 (2)
SEP_FUNCTIONAL	=00000002	65 (2)	
SEP_REPAIR	=00000003	65 (2)	
SIZ...	=00000008	65 (2)	65 (2)

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$DEF	1	65 (2)	
$DS_BGNMOD	1	65 (2)	65 (2)
$DS_ENVDEF	2	65 (2)	65 (2)
$EQU	1	65 (2)	65 (2)
$EQULS1	1	65 (2)	65 (2)
$EQULST	1	65 (2)	65 (2)
$GBLINI	2	65 (2)	65 (2)
$VIELD	1	65 (2)	65 (2)
$VIELD1	1	65 (2)	65 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.08	00:00:00.61
Command processing	25	00:00:00.23	00:00:00.97
Pass 1	312	00:00:01.61	00:00:06.35
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	78	00:00:00.48	00:00:02.19
Symbol table output	5	00:00:00.02	00:00:00.05
Psect synopsis output	4	00:00:00.03	00:00:00.09
Cross-reference output	25	00:00:00.14	00:00:00.28
Assembler run totals	481	00:00:02.61	00:00:10.60

The working set limit was 900 pages.
9370 bytes (19 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 26 non-local and 0 local symbols.
71 source lines were read in Pass 1, producing 0 object records in Pass 2.
17 pages of virtual memory were used to define 7 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	3
TOTALS (all libraries)	5

100 GETS were required to define 5 macros.

There were 1 error, 0 warnings and 0 information messages, on lines:
   71 (2)

/LIS/CRO ENKAXG

Table of contents

```
                        .SBTTL  ENKAXH: UNIBUS INTERFACE TEST
                        .TITLE  ENKAXH  VAX-11/730 Specific CPU Cluster Exerciser
                        .IDENT  /1-3/

;
; Copyright (C) 1981
; Digital Equipment Corporation, Maynard, Massachusetts 01754
;
; This software is furnished  under a license for use only on a single
; computer  system and  may be  copied only with  the inclusion of the
; above copyright notice.  This software, or any other copies thereof,
; may not be provided or  otherwise made available to any other person
; except for use on such system and to one who agrees to these license
; terms.  Title to and  ownership of the  software  shall at all times
; remain in DEC.
;
; The information in this software is subject to change without notice
; and should  not be construed  as a commitment  by Digital  Equipment
; Corporation.
;
; DEC assumes  no  responsibility  for  the use or  reliability of its
; software on equipment which is not supplied by DEC.
;
;
;++
; Facility:     VAX Architecture Diagnostic.
;
; Abstract:     This module consists of two tests that check the VAX-11/730
;               Unibus interface (UBI). This includes testing of the interface CSR,
;               MAP's, and different types of data transactions and exercising
;               UBE's. This program runs under the Diagnostic Supervisor and
;               is intended for use on a VAX 11/730 system.
;
; Environment:  VAX Diagnostic Supervisor.
;
; Author:       Rich Lewis      14-Oct-81       Version 5X
;
; Modified by:
;               Kevin Martin    9-Nov-1982
;       01      The number of Unibus maps changed from 504 to 496.
;
;               Tai-Chun Pan    01-Apr-83       Version 1.2
;
;               Added quick flag on Test 7(TU58). If quick flag is set
;               will skip subtest 4 through subtest 11.  Added event flag 3.
;               If event flag 3 is set, will override protection,
;               i.e., go ahead and write on tape.  If run APT & event flag 3
;               is clear & tape is not scratch, will report an error.
;               Modified Unibus sizer on Test 8 & 9. make test run faster.
;
;               Tai-Chun Pan    03-Aug-83       Version 1.3
;
;               fixed bugs on TEST 7 and error summary routine call when
;               running under APT.
;
;--
```

```
                     0000     59          .SBTTL  DECLARATIONS
                     0000     60          .LIST   MEB
                     0000     61          .NLIST  CND
                 00000000     62          .PSECT  ENKAXH, PAGE, NOWRT
                     0000     63          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     64          $DS_BGNMOD      CEP_REPAIR, IN=8
                     0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)'
                     0000     65
                     0000     66 ;
                     0000     67 ; Include files:
                     0000     68 ;
                     0000     69
                     0000     70          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                     0000     71
                     0000     75
                     0000     76          $DS_SCBDEF
                     0000     77          $DS_DSADEF
                     0000     78          $DS_DSDEF
                     0000     79          $DS_PARDEF
                     0000     80          $DS_PSLDEF
                     0000     81          $DS_ERRDEF
                     0000     82          $DS_DSSDEF
                     0000     83          $DS_HPODEF
                     0000     84          $DS_KA_DEF
                     0000     85          ENKAX_SECDEF
                     0000     86
                     0000     87 ;
                     0000     88 ; Own storage:
                     0000     89 ;
                     0000     90
                     0000     91 ; Bit positions and masks:
                     0000     92
    82007FFF         0000     93 M_MAP_WRTBL      =^X82007FFF             ; Map Entry writable bit mask
    7DFF8000         0000     94 M_MAP_NOWRT      =^X7DFF8000             ; Map Entry nonwritable bit mask
    02000000         0000     95 M_BYTE_OFF       =1@25                   ; Map Entry Byte-offset mask
    00000019         0000     96 V_BYTE_OFF       =25
    80000000         0000     97 M_MAP_VAL        =1@31                   ; Map Entry Valid bit mask
    0000001F         0000     98 V_MAP_VAL        =31
    02000000         0000     99 M_DSABL_ECC      =1@25                   ; Mem contrlr disable ECC
    04000000         0000    100 M_DIAG_MODE      =1@26                   ; Mem contrlr diagnostic mode
    20000000         0000    101 M_FRCE_TB_ERR    =1@29                   ; Mem contrlr force TB parity err
    00000001         0000    102 M_RUN            =1@0                    ; Interval Timer run bit
    00000000         0000    103 V_RUN            =0
    00000010         0000    104 M_TRANSFER       =1@4                    ; Interval Timer ICCS transfer bit
    00000004         0000    105 V_TRANSFER       =4
    00000040         0000    106 M_ENABLE         =1@6                    ; Interval Timer enable bit
    00000006         0000    107 V_ENABLE         =6
    00000080         0000    108 M_CLK_INTR       =1@7                    ; Interval Timer ICCS interrupt bit
    00000007         0000    109 V_CLK_INTR       =7
    80000000         0000    110 M_CLK_ERR        =1@31                   ; Interval Timer ICCS error bit
    0000001F         0000    111 V_CLK_ERR        =31
                     0000    112
                     0000    113
                     0000    114 ; Error flag positions and masks:
                     0000    115
    00000001         0000    116 M_NO_CSR         =1@0                    ; UBI CSR not accessible
    00000000         0000    117 V_NO_CSR         =0
```

```
00000002  0000   118 M_EXP_BR_INTR   =1@1                    ; expected BR4 interrupt
00000001  0000   119 V_EXP_BR_INTR   =1
00000004  0000   120 M_TIMEOUT       =1@2                    ; indicates Interval Timer expired
00000002  0000   121 V_TIMEOUT       =2
00000008  0000   122 M_BR_INTR       =1@3                    ; interrupt occurred flag
00000003  0000   123 V_BR_INTR       =3
00000010  0000   124 M_INTR_ERR      =1@4                    ; interrupt error flag
00000004  0000   125 V_INTR_ERR      =4
00000040  0000   126 M_UNX_SSYN_ERR  =1@6                    ; unexpected SSYN occurred
00000006  0000   127 V_UNX_SSYN_ERR  =6
00000080  0000   128 M_EXP_WR_ERR    =1@7                    ; expected UBI Write error
00000007  0000   129 V_EXP_WR_ERR    =7
00000200  0000   130 M_EXP_NXM_ERR   =1@9                    ; expected UBI NXM error
00000009  0000   131 V_EXP_NXM_ERR   =9
00000400  0000   132 M_EXP_PB_ERR    =1@10                   ; expected UBI Map Parity Error
0000000A  0000   133 V_EXP_PB_ERR    =10
00001000  0000   134 M_DATA_ERR      =1@12                   ; data error flag
0000000C  0000   135 V_DATA_ERR      =12
          0000   136
          0000   137
          0000   138 ; UBI CSR bit masks:
          0000   139
80000000  0000   140 M_UBI_RDS_ERR   =1@31                   ; Read Data Substitute (uncorr. ECC)
0000001F  0000   141 V_UBI_RDS_ERR   =31
00010000  0000   142 M_UBI_NXM_ERR   =1@16                   ; Non eXistent Memory reference
00000010  0000   143 V_UBI_NXM_ERR   =16
00008000  0000   144 M_UBI_PB_ERR    =1@15                   ; map Parity error
0000000F  0000   145 V_UBI_PB_ERR    =15
00004000  0000   146 M_UBI_WR_ERR    =1@14                   ; WRite error (write w/ invalid map)
0000000E  0000   147 V_UBI_WR_ERR    =14
8001C000  0000   148 M_UBI_STS       =<7@14>!<1@31>          ; all UBI CSR status bits
7FFE3FFF  0000   149 M_UBI_NON_STS   =<^X3FFF@17>!<^X3FFF>   ; UBI Don't-Care bits
          0000   150
          0000   151 ; UBE bit masks:
          0000   152
          0000   153 ; UBE CR 1
          0000   154
00000001  0000   155 M_GO            =1@0                    ; go
00000002  0000   156 M_BR4           =1@1                    ; bus request 4
00000004  0000   157 M_BR5           =1@2                    ; bus request 5
00000008  0000   158 M_BR6           =1@3                    ; bus request 6
00000010  0000   159 M_BR7           =1@4                    ; bus request 7
00000020  0000   160 M_NPR           =1@5                    ; non-processor request
00000040  0000   161 M_INT_ODNE      =1@6                    ; interrupt on done
00000080  0000   162 M_RDY           =1@7                    ; ready
00000007  0000   163 V_RDY           =7
00000000  0000   164 M_DATI          =0@8                    ; DATI
00000100  0000   165 M_DATIP         =1@8                    ; DATIP
00000200  0000   166 M_DATO          =2@8                    ; DATO
00000300  0000   167 M_DATOB         =3@8                    ; DATOB
00000000  0000   168 M_INT_BM        =0@10                   ; interrupt when bus master
00000400  0000   169 M_ONE_XFR       =1@10                   ; exec. one transfer until CCOVF
00000800  0000   170 M_TWO_XFR       =2@10                   ; exec. two transfers until CCOVF
00000C00  0000   171 M_RLS_BUS       =3@10                   ; release bus when master
00001000  0000   172 M_CC_DAT        =1@12                   ; enable transfer from CC reg.
00002000  0000   173 M_INH_ROL       =1@13                   ; inhibit rotate left on DATIP
00004000  0000   174 M_DATOB_IP      =1@14                   ; DATOB on DATIP
```

```
                00008000  0000   175 M_ERR           =1@15                   ; error
                0000000F  0000   176 V_ERR           =15
                          0000   177
                          0000   178 ; UBE CR 2
                          0000   179
                00000000  0000   180 M_EXT_MEM       =00@0                   ; allows access to low 32k
                00000004  0000   181 M_INH_INC       =1@2                    ; inhibit incr. of CC and ADDR
                00000008  0000   182 M_INH_SACK      =1@3                    ; inhibit SACK
                00000010  0000   183 M_PWR_DN        =1@4                    ; begin power down sequence
                00000020  0000   184 M_WNG_GNT       =1@5                    ; wrong grant back error
                00000005  0000   185 V_WNG_GNT       =5
                00000040  0000   186 M_MAX_LAT       =1@6                    ; maximum latency error
                00000006  0000   187 V_MAX_LAT       =6
                00000080  0000   188 M_NO_SACK       =1@7                    ; no CPU timeout on no SACK error
                00000007  0000   189 V_NO_SACK       =7
                00000100  0000   190 M_NO_SSYN       =1@8                    ; no SSYN timeout error
                00000008  0000   191 V_NO_SSYN       =8
                00000200  0000   192 M_WNG_A         =1@9                    ; wrong bus "A" lines error
                00000009  0000   193 V_WNG_A         =9
                00000400  0000   194 M_NO_GNT        =1@10                   ; no grant or not one grant error
                0000000A  0000   195 V_NO_GNT        =10
                00000800  0000   196 M_INT_SSYN      =1@11                   ; interrupt SSYN error
                0000000B  0000   197 V_INT_SSYN      =11
                00001000  0000   198 M_ENB_PB        =1@12                   ; enable parity bit 'B"
                00002000  0000   199 M_CCOVF         =1@13                   ; cycle count overflow
                0000000D  0000   200 V_CCOVF         =13
                00004000  0000   201 M_TM_DLY        =1@14                   ; time delay
                          0000   202
                          0000   203
                          0000   204 ; Error Flag
                          0000   205
                    0000  0000   206 W_FLAGS:        .WORD   0
                          0002   207
                          0002   208
                          0002   209 ; UNIBUS Addresses
                          0002   210
                00F26010  0002   211 A_UBI_CSR       =^XF26010               ; address of CSR
                00F26800  0002   212 A_MAP_BGN       =^XF26800               ; address of beginning of Map
                00F26FFC  0002   213 A_MAP_END       =^XF26FFC               ; address of last Map Entry
                00FC0000  0002   214 A_UB_IO_SP      =^XFC0000               ; base address of UNIBUS I/O space
                00FFF00C  0002   215 A_SIMUL_GO      =^O77770014             ; simultaneous Go address
                          0002   216
                          0002   217
                          0002   218 ; UNIBUS Interrupt vector
                          0002   219
                00000348  0002   220 SCB$L_BE0       =^X348                  ; UBE 0 interrupt vector offset
                0000034C  0002   221 SCB$L_BE1       =^X34C                  ; UBE 1 interrupt vector offset
                00000350  0002   222 SCB$L_BE2       =^X350                  ; UBE 2 interrupt vector offset
                00000354  0002   223 SCB$L_BE3       =^X354                  ; UBE 3 interrupt vector offset
                          0002   224
                          0002   225
                          0002   226 ; Main Memory Storage:
                          0002   227
                FFFFFFFF  0002   228 FRST_VAL_MAP:   .LONG   -1              ; first valid Map Number
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  0006   229 V_VALID_MAPS:   .LONG   -1[16],0       ; bitvector for usable map entries
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  0016
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  0026
```

```
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  0036
                        00000000  0046
                        00000000  004A    230 L_PAGE_NO:       .LONG                    ; page number
                        0000005E  004E    231 L_SAVE_CSR:      .BLKL   4                ; location to save UBI CSR
                        00000086  005E    232 L_EXP_DATA:      .BLKL   10               ; expected data
                        00500200  0086    233 L_NXM_ADR:       .LONG   ^X500200         ; non-existent memory location
                        00000000  008A    234 L_MAP_NUM:       .LONG                    ; map number
                        00000000  008E    235 L_PTRN:          .LONG                    ; holds data pattern for Multi Xfrs
                        000000A2  0092    236 L_PTRNS:         .BLKL   4                ; saved data patterns
                              00  00A2    237 B_DONE_CT:       .BYTE                    ; number of UBEs done
                        00000000  00A3    238 L_MAPS_NEEDED:   .LONG                    ; number of maps needed
                        00000000  00A7    239 L_IPL:           .LONG                    ; IPLs used in Interrupt subtest
                        00000000  00AB    240 L_BR_LVL:        .LONG                    ; BR level used in Interrupt subtest
                        00000000  00AF    241 L_UBI_STATUS:    .LONG                    ; status word from channel services
                            0000  00B3    242 W_UBI_STS_2:     .WORD                    ; interrupt status word from chanl srvc
                            0000  00B5    243 W_VECTOR:        .WORD                    ; vector received
                        00000000  00B7    244 L_XFR_COUNT:     .LONG                    ; transfer count
                        FFF0BDC0  00BB    245 L_MXFR_TIME:     .LONG   -1000000         ; Max time allowed for mult xfer 1 sec
                     04 04 08 08  00BF    246 B_MAPS_REQ_1:    .BYTE   8,8,4,4          ; # of maps req'd in mult transfer 1
                     0C 08 04 04  00C3    247 B_MAPS_REQ_2:    .BYTE   4,4,8,12         ; # of maps req'd in mult transfer 2
                        00000000  00C7    248 L_BUFFER_ADR:    .LONG                    ; starting address of current buffer
                        00000000' 00CB    249 A_BUFRS_DFR_1:   .LONG   A_UBE_BUFFER_0,- ; buffer deferred address table Mxfr 1
                                  00CF    250                         A_UBE_BUFFER_0,-
                        00000000' 00CF    251                         A_UBE_BUFFER_2,-
                00000000'00000000' 00D3   252                         A_UBE_BUFFER_3
                                  00DB    253 A_BUFRS_DFR_2:   .LONG   A_UBE_BUFFER_0,- ; buffer deferred address table Mxfr 2
                        00000000' 00DB    254                         A_UBE_BUFFER_1,-
                        00000000' 00DF    255                         A_UBE_BUFFER_2,-
                00000000'00000000' 00E3   256                         A_UBE_BUFFER_3
                        000000FB  00EB    257 A_BUFFERS_1:     .BLKL   4                ; actual buffer addresses Mxfr 1
                        0000010B  00FB    258 A_BUFFERS_2:     .BLKL   4                ; actual buffer addresses Mxfr 2
                        00000000  010B    259 L_BUFFER_SIZE:   .LONG                    ; length of current buffer
                                  010F    260 L_BUF_SIZES_1:   .LONG   2000,-           ; word lengths of buffers Mxfr 1
                        000007D0  010F    261                         2000,-
                        000007D0  0113    262                         1000,-
                000003E8 000003E8 0117    263                         1000
                                  011F    264 L_BUF_SIZES_2:   .LONG   1000,-           ; word lengths of buffers Mxfr 2
                        000003E8  011F    265                         1000,-
                        000003E8  0123    266                         2000,-
                00000BB8 000007D0 0127    267                         3000
                                  012F    268
                                  012F    269 T_MXFR_1_DESC:   .LONG   T_M1_UBE_0,-     ; adrs of mxfr 1 descriptions
                        0000096B' 012F    270                         T_M1_UBE_1,-
                        000009F4' 0133    271                         T_M1_UBE_2,-
                00000AA5'00000A62' 0137   272                         T_M1_UBE_3
                                  013F    273 T_MXFR_2_DESC:   .LONG   T_M2_UBE_0,-     ; adrs of mxfr 2 descriptions
                        00000AE8' 013F    274                         T_M2_UBE_1,-
                        00000B4C' 0143    275                         T_M2_UBE_2,-
                00000C1E'00000BAD' 0147   276                         T_M2_UBE_3
                        0000015F  014F    277 L_INITL_MAP:     .BLKL   4                ; initial map numbers
                        00000000  015F    278 L_ERROR_CT:      .LONG                    ; error count
                        0000018B  0163    279 L_ERROR_LOG:     .BLKL   10
                                  018B    280
                        00000000  018B    281 L_INFO_BIT:      .LONG                    ; info bit pointer
                        00000000  018F    282 L_PAIR_SEP:      .LONG                    ; separation of redundant relation
                        00000000  0193    283 L_POSITION:      .LONG                    ; bit pointer
                        00000000  0197    284 L_TEMP_A:        .LONG                    ; temporary data
```

```
00000000  019B   285 L_TEMP_B:       .LONG                   ; temporary data
00000000  019F   286 L_POP:          .LONG                   ; population of ones
00000000  01A3   287 L_RETRIEVE:     .LONG                   ; retrieved data
          01A7   288
          01A7   289 ; Misc.
          01A7   290
33550FFF  01A7   291 L_PATTERN       =^X33550FFF             ; initial data pattern for Mult Xfrs
00F20004  01A7   292 A_MEM_CSR_1     =^XF20004               ; Memory CSR 1
00000037  01A7   293 PR$L_UB_INIT    =^X37                   ; unibus init. internal register number
          01A7   294
          01A7   295
          01A7   296 .SBTTL DATA TABLES
          01A7   297
          01A7   298 ; This data is used in the Map Data Bus subtest to float a one through a
          01A7   299 ; field of zeros and a zero through a field of ones in the writable bits
          01A7   300 ; (31,25,<14:0>) of a Map Entry to test that no data lines are stuck or
          01A7   301 ; shorted.  Non-writable bits are ignored.
          01A7   302
00000000  01A7   303 MAP_ENT_TBL:    .LONG   ^X00000000      ; field of zeros
00000001  01AB   304                 .LONG   ^X00000001
00000002  01AF   305                 .LONG   ^X00000002
00000004  01B3   306                 .LONG   ^X00000004
00000008  01B7   307                 .LONG   ^X00000008
00000010  01BB   308                 .LONG   ^X00000010
00000020  01BF   309                 .LONG   ^X00000020
00000040  01C3   310                 .LONG   ^X00000040
00000080  01C7   311                 .LONG   ^X00000080
00000100  01CB   312                 .LONG   ^X00000100
00000200  01CF   313                 .LONG   ^X00000200
00000400  01D3   314                 .LONG   ^X00000400
00000800  01D7   315                 .LONG   ^X00000800
00001000  01DB   316                 .LONG   ^X00001000
00002000  01DF   317                 .LONG   ^X00002000
00004000  01E3   318                 .LONG   ^X00004000
02000000  01E7   319                 .LONG   ^X02000000
80000000  01EB   320                 .LONG   ^X80000000
82007FFF  01EF   321                 .LONG   ^X82007FFF      ; field of ones
82007FFE  01F3   322                 .LONG   ^X82007FFE
82007FFD  01F7   323                 .LONG   ^X82007FFD
82007FFB  01FB   324                 .LONG   ^X82007FFB
82007FF7  01FF   325                 .LONG   ^X82007FF7
82007FEF  0203   326                 .LONG   ^X82007FEF
82007FDF  0207   327                 .LONG   ^X82007FDF
82007FBF  020B   328                 .LONG   ^X82007FBF
82007F7F  020F   329                 .LONG   ^X82007F7F
82007EFF  0213   330                 .LONG   ^X82007EFF
82007DFF  0217   331                 .LONG   ^X82007DFF
82007BFF  021B   332                 .LONG   ^X82007BFF
820077FF  021F   333                 .LONG   ^X820077FF
82006FFF  0223   334                 .LONG   ^X82006FFF
82005FFF  0227   335                 .LONG   ^X82005FFF
82003FFF  022B   336                 .LONG   ^X82003FFF
80007FFF  022F   337                 .LONG   ^X80007FFF
02007FFF  0233   338                 .LONG   ^X02007FFF
          0237   339
          0237   340
          0237   341 ; This data table is generated during execution of the Map Address Bus subtest
```

```
                        0237   342 ; to store the Reed-Muller values generated at the time of execution.
                        0237   343
00000257                0237   344 RD_MLLR_TBL:    .BLKL   8
                        0257   345
                        0257   346
                        0257   347 ; This table is used to store data retrieved from the Map Entries
                        0257   348
00000277                0257   349 RETRIEVE_TBL:   .BLKL   8
                        0277   350
                        0277   351
                        0277   352 ; This table is used to store the values decoded from the retrieved data
                        0277   353
00000297                0277   354 DECODE_TBL:     .BLKL   8
                        0297   355
                        0297   356
                        0297   357 ; This data table is used by the subroutine which generates the Reed-Muller
                        0297   358 ; representations of the numbers 0 - 7.
                        0297   359
                FFFF    0297   360 CODEVECTORS:    .WORD   ^XFFFF
                5555    0299   361                .WORD   ^X5555
                3333    029B   362                .WORD   ^X3333
                0F0F    029D   363                .WORD   ^X0F0F
                        029F   364
                        029F   365
                        029F   366 ; This data is used in the CPU Read/Write subtest to test that no bits
                        029F   367 ; in the UBE address and data registers are stuck or shorted and in the
                        029F   368 ; transfer subtests to test the integrity of the data path.
                        029F   369
                5555    029F   370 W_PTRN_TBL:     .WORD   ^X5555
                AAAA    02A1   371                .WORD   ^XAAAA
                3333    02A3   372                .WORD   ^X3333
                0F0F    02A5   373                .WORD   ^X0F0F
                00FF    02A7   374                .WORD   ^X00FF
                0000    02A9   375                .WORD   ^X0000                  ; terminator
                        02AB   376
                        02AB   377
                        02AB   378 ; The following is a table of addresses of the UBE registers.
                        02AB   379
                        02AB   380 A_UBE_ADR_TBL:
00FFF008                02AB   381 A_BE0_CLR_ERR:  .LONG   ^O77770010              ; clear error address
00FFF000                02AF   382 A_BE0_DB:       .LONG   ^O77770000              ; adr of Data reg
00FFF002                02B3   383 A_BE0_CC:       .LONG   ^O77770002              ; adr of Cycle Count reg
00FFF004                02B7   384 A_BE0_BA:       .LONG   ^O77770004              ; adr of Address reg
00FFF00E                02BB   385 A_BE0_CR2:      .LONG   ^O77770016              ; adr of Control reg 2
00FFF006                02BF   386 A_BE0_CR1:      .LONG   ^O77770006              ; adr of Control reg 1
00FFF032                02C3   387 A_BE3_CC:       .LONG   ^O77770062              ; adr of Cyc Ct UBE 3
00FFF036                02C7   388 A_BF3_CR1:      .LONG   ^O77770066              ; adr of Cont Reg 1 UBE 3
                        02CB   389
                        02CB   390
                        02CB   391
                0000    02CB   392 UBE_DATI_1:     .WORD   0                       ; Clear previous errors
                0000    02CD   393                .WORD   0                       ; Data Register
                FFFF    02CF   394                .WORD   -1                      ; Cycle Count
                0000    02D1   395                .WORD   0                       ; Address Register
                0000    02D3   396                .WORD   0                       ; Control Register 2
                04F1    02D5   397                .WORD   <M_GO!M_BR7!M_NPR!M_INT_ODNE!M_RDY!M_DATI!M_ONE_XFR>
                        02D7   398
```

```
0000  02D7   399 UBE_DATIP_1:    .WORD   0                           ; Clear previous errors
0000  02D9   400                 .WORD   0                           ; Data Register
FFFF  02DB   401                 .WORD   -1                          ; Cycle Count
0000  02DD   402                 .WORD   0                           ; Address Register
0000  02DF   403                 .WORD   0                           ; Control Register 2
05F1  02E1   404                 .WORD   <M_GO!M_BR7!M_NPR!M_INT_ODNE!M_RDY!M_DATIP!M_ONE_XFR>
      02E3   405
0000  02E3   406 UBE_DATOB_1:    .WORD   0                           ; Clear previous errors
0000  02E5   407                 .WORD   0                           ; Data Register
FFFF  02E7   408                 .WORD   -1                          ; Cycle Count
0000  02E9   409                 .WORD   0                           ; Address Register
0000  02EB   410                 .WORD   0                           ; Control Register 2
07F1  02ED   411                 .WORD   <M_GO!M_BR7!M_NPR!M_INT_ODNE!M_RDY!M_DATOB!M_ONE_XFR>
      02EF   412
0000  02EF   413 UBE_DATO_1:     .WORD   0                           ; Clear previous errors
0000  02F1   414                 .WORD   0                           ; Data Register
FFFF  02F3   415                 .WORD   -1                          ; Cycle Count
0000  02F5   416                 .WORD   0                           ; Address Register
0000  02F7   417                 .WORD   0                           ; Control Register 2
06F1  02F9   418                 .WORD   <M_GO!M_BR7!M_NPR!M_INT_ODNE!M_RDY!M_DATO!M_ONE_XFR>
      02FB   419
0000  02FB   420 UBE_PB_ERR:     .WORD   0                           ; Clear previous errors
0000  02FD   421                 .WORD   0                           ; Data Register
FFFF  02FF   422                 .WORD   -1                          ; Cycle Count
0000  0301   423                 .WORD   0                           ; Address Register
1000  0303   424                 .WORD   M_ENB_PB                    ; Control Register 2
02F1  0305   425                 .WORD   <M_GO!M_BR7!M_INT_ODNE!M_NPR!M_DATO!M_RDY>
      0307   426
0000  0307   427 UBE_BR7_INTR:   .WORD   0                           ; Clear previous errors
0000  0309   428                 .WORD   0                           ; Data Register
0000  030B   429                 .WORD   0                           ; Cycle Count
0000  030D   430                 .WORD   0                           ; Address Register
0000  030F   431                 .WORD   0                           ; Control Register 2
0091  0311   432                 .WORD   <M_GO!M_BR7!M_RDY!M_INT_BM> ; Control Reg 1
      0313   433
0000  0313   434 UBE_BR6_INTR:   .WORD   0                           ; Clear previous errors
0000  0315   435                 .WORD   0                           ; Data Register
0000  0317   436                 .WORD   0                           ; Cycle Count
0000  0319   437                 .WORD   0                           ; Address Register
0000  031B   438                 .WORD   0                           ; Control Register 2
0089  031D   439                 .WORD   <M_GO!M_BR6!M_RDY!M_INT_BM> ; Control Reg 1
      031F   440
0000  031F   441 UBE_BR5_INTR:   .WORD   0                           ; Clear previous errors
0000  0321   442                 .WORD   0                           ; Data Register
0000  0323   443                 .WORD   0                           ; Cycle Count
0000  0325   444                 .WORD   0                           ; Address Register
0000  0327   445                 .WORD   0                           ; Control Register 2
0085  0329   446                 .WORD   <M_GO!M_BR5!M_RDY!M_INT_BM> ; Control Reg 1
      032B   447
0000  032B   448 UBE_BR4_INTR:   .WORD   0                           ; Clear previous errors
0000  032D   449                 .WORD   0                           ; Data Register
0000  032F   450                 .WORD   0                           ; Cycle Count
0000  0331   451                 .WORD   0                           ; Address Register
0000  0333   452                 .WORD   0                           ; Control Register 2
0083  0335   453                 .WORD   <M_GO!M_BR4!M_RDY!M_INT_BM> ; Control Reg 1
      0337   454
0000  0337   455 UBE_0_MULT_1:   .WORD   0                           ; Clear previous errors
```

```
0000  0339   456                .WORD   0                               ; Data Register
F830  033B   457                .WORD   -2000                           ; Cycle Count
0000  033D   458                .WORD   0                               ; Address Register
0000  033F   459                .WORD   0                               ; Control Register 2
45C8  0341   460                .WORD   <M_RDY!M_BR6!M_INT_ODNE!M_DATIP!M_ONE_XFR!M_DATOB_IP>
      0343   461
0000  0343   462 UBE_1_MULT_1:   .WORD   0                               ; Clear previous errors
0000  0345   463                .WORD   0                               ; Data Register
F830  0347   464                .WORD   -2000                           ; Cycle Count
0000  0349   465                .WORD   0                               ; Address Register
0000  034B   466                .WORD   0                               ; Control Register 2
25F0  034D   467                .WORD   <M_RDY!M_BR7!M_NPR!M_INT_ODNE!M_DATIP!M_ONE_XFR!M_INH_ROL>
      034F   468
0000  034F   469 UBE_2_MULT_1:   .WORD   0                               ; Clear previous errors
CCCC  0351   470                .WORD   ^XCCCC                          ; Data Register
FC18  0353   471                .WORD   -1000                           ; Cycle Count
0000  0355   472                .WORD   0                               ; Address Register
0000  0357   473                .WORD   0                               ; Control Register 2
06F0  0359   474                .WORD   <M_RDY!M_BR7!M_NPR!M_INT_ODNE!M_DATO!M_ONE_XFR>
      035B   475
0000  035B   476 UBE_3_MULT_1:   .WORD   0                               ; Clear previous errors
5555  035D   477                .WORD   ^X5555                          ; Data Register
FC18  035F   478                .WORD   -1000                           ; Cycle Count
0000  0361   479                .WORD   0                               ; Address Register
0000  0363   480                .WORD   0                               ; Control Register 2
06C4  0365   481                .WORD   <M_RDY!M_BR5!M_INT_ODNE!M_DATO!M_ONE_XFR>
      0367   482
0000  0367   483 UBE_0_MULT_2:   .WORD   0                               ; Clear previous errors
0000  0369   484                .WORD   0                               ; Data Register
F830  036B   485                .WORD   -2000                           ; Cycle Count
0000  036D   486                .WORD   0                               ; Address Register
0000  036F   487                .WORD   0                               ; Control Register 2
1BE8  0371   488                .WORD   <M_RDY!M_BR6!M_NPR!M_INT_ODNE!M_DATOB!M_TWO_XFR!M_CC_DAT>
      0373   489
0000  0373   490 UBE_1_MULT_2:   .WORD   0                               ; Clear previous errors
0000  0375   491                .WORD   0                               ; Data Register
FC18  0377   492                .WORD   -1000                           ; Cycle Count
0000  0379   493                .WORD   0                               ; Address Register
0000  037B   494                .WORD   0                               ; Control Register 2
05F0  037D   495                .WORD   <M_RDY!M_BR7!M_NPR!M_INT_ODNE!M_DATIP!M_ONE_XFR>
      037F   496
0000  037F   497 UBE_2_MULT_2:   .WORD   0                               ; Clear previous errors
0000  0381   498                .WORD   0                               ; Data Register
FC18  0383   499                .WORD   -1000                           ; Cycle Count
0000  0385   500                .WORD   0                               ; Address Register
0000  0387   501                .WORD   0                               ; Control Register 2
08C4  0389   502                .WORD   <M_RDY!M_BR5!M_DATI!M_TWO_XFR!M_INT_ODNE>
      038B   503
0000  038B   504 UBE_3_MULT_2:   .WORD   0                               ; Clear previous errors
0000  038D   505                .WORD   0                               ; Data Register
FFFF  038F   506                .WORD   -1                              ; Cycle Count
0000  0391   507                .WORD   0                               ; Address Register
0000  0393   508                .WORD   0                               ; Control Register 2
04D0  0395   509                .WORD   <M_RDY.M_BR7!M_DATI!M_ONE_XFR!M_INT_ODNE>
      0397   510
      0397   511
```

```
                                        0397    516
                                        0397    517 .SBTTL  ERROR MESSAGES
                                        0397    518 ;-----------------------------------------------------------------------
                                        0397    519 T_NO_BE0:
69 76 65 64 20 45 42 55 20 2F 21 00'    0397    520         .ASCIC  "!/ UBE device 0 not selected. Exiting test."
6C 65 73 20 74 6F 6E 20 30 20 65 63     03A3
69 74 69 78 45 20 2E 64 65 74 63 65     03AF
             2E 74 73 65 74 20 67 6E    03BB
                                  2B    0397
                                        03C3    521 ;-----------------------------------------------------------------------
                                        03C3    522 T_NO_ACCESS:
65 63 63 61 6E 69 20 45 42 55 20 00'    03C3    523         .ASCIC  " UBE inaccessible.  Exiting test."
69 78 45 20 20 2E 65 6C 62 69 73 73     03CF
       2E 74 73 65 74 20 67 6E 69 74    03DB
                                  21    03C3
                                        03E5    524 ;-----------------------------------------------------------------------
                                        03E5    525 T_UBE_ADR:
20 44 45 54 43 45 50 58 45 20 20 00'    03E5    526         .ASCIC  "  EXPECTED UBE at Address = !OL(O) = !XL(X).!/"
65 72 64 64 41 20 74 61 20 45 42 55     03F1
20 29 4F 28 4C 4F 21 20 3D 20 73 73     03FD
   2F 21 2E 29 58 28 4C 58 21 20 3D     0409
                                  2E    03E5
                                        0414    527 ;-----------------------------------------------------------------------
                                        0414    528 T_NO_UBE:
20 73 45 42 55 20 6F 4E 20 2F 21 00'    0414    529         .ASCIC  "!/ No UBEs selected. Exiting test."
78 45 20 2E 64 65 74 63 65 6C 65 73     0420
   2E 74 73 65 74 20 67 6E 69 74 69     042C
                                  22    0414
                                        0437    530 ;-----------------------------------------------------------------------
                                        0437    531 T_NO_MAPS:
6C 62 61 73 75 20 6F 4E 20 2F 21 00'    0437    532         .ASCIC  "!/ No usable MAPS available.  Exiting test."
6C 69 61 76 61 20 53 50 41 4D 20 65     0443
69 74 69 78 45 20 20 2E 65 6C 62 61     044F
             2E 74 73 65 74 20 67 6E    045B
                                  2B    0437
                                        0463    533 ;-----------------------------------------------------------------------
                                        0463    534 T_CSR_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'    0463    535         .ASCIC  " Attempted to test UNIBUS CSR."
42 49 4E 55 20 74 73 65 74 20 6F 74     046F
                2E 52 53 43 20 53 55    047B
                                  1E    0463
                                        0482    536 ;-----------------------------------------------------------------------
                                        0482    537 T_MAP_DBUS_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'    0482    538         .ASCIC  " Attempted to test UNIBUS Map data lines."
42 49 4E 55 20 74 73 65 74 20 6F 74     048E
20 61 74 61 64 20 70 61 4D 20 53 55     049A
                   2E 73 65 6E 69 6C    04A6
                                  29    0482
                                        04AC    539 ;-----------------------------------------------------------------------
                                        04AC    540 T_MAP_ABUS_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'    04AC    541         .ASCIC  " Attempted to test UNIBUS Map address lines."
42 49 4E 55 20 74 73 65 74 20 6F 74     04B8
65 72 64 64 61 20 70 61 4D 20 53 55     04C4
          2E 73 65 6E 69 6C 20 73 73    04D0
                                  2C    04AC
                                        04D9    542 ;-----------------------------------------------------------------------
                                        04D9    543 T_MAP_ENT_ERR:
```

```
20 64 65 74 70 6D 65 74 74 41 20 00'  04D9    544          .ASCIC  " Attempted to test writable bits of UNIBUS Map Entries."
74 69 72 77 20 74 73 65 74 20 6F 74   04E5
66 6F 20 73 74 69 62 20 65 6C 62 61   04F1
20 70 61 4D 20 53 55 42 49 4E 55 20   04FD
            2E 73 65 69 72 74 6E 45   0509
                                 37   04D9
                                      0511    545 ;-----------------------------------------------------------------------
                                      0511    546 T_ADR_REG_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  0511    547          .ASCIC  " Attempted CPU access to UBE Address register."
74 20 73 73 65 63 63 61 20 55 50 43   051D
73 65 72 64 64 41 20 45 42 55 20 6F   0529
   2E 72 65 74 73 69 67 65 72 20 73   0535
                                 2E   0511
                                      0540    548 ;-----------------------------------------------------------------------
                                      0540    549 T_DATA_REG_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  0540    550          .ASCIC  " Attempted CPU access to UBE Data register."
74 20 73 73 65 63 63 61 20 55 50 43   054C
72 20 61 74 61 44 20 45 42 55 20 6F   0558
            2E 72 65 74 73 69 67 65   0564
                                 2B   0540
                                      056C    551 ;-----------------------------------------------------------------------
                                      056C    552 T_DATI_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  056C    553          .ASCIC  " Attempted UNIBUS DATI operation."
20 49 54 41 44 20 53 55 42 49 4E 55   0578
      2E 6E 6F 69 74 61 72 65 70 6F   0584
                                 21   056C
                                      058E    554 ;-----------------------------------------------------------------------
                                      058E    555 T_D_PATH_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  058E    556          .ASCIC  " Attempted UNIBUS Data Path test."
20 61 74 61 44 20 53 55 42 49 4E 55   059A
      2E 74 73 65 74 20 68 74 61 50   05A6
                                 21   058E
                                      05B0    557 ;-----------------------------------------------------------------------
                                      05B0    558 T_MAP_SEL_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  05B0    559          .ASCIC  " Attempted UNIBUS Map Select test."
53 20 70 61 4D 20 53 55 42 49 4E 55   05BC
   2E 74 73 65 74 20 74 63 65 6C 65   05C8
                                 22   05B0
                                      05D3    560 ;-----------------------------------------------------------------------
                                      05D3    561 T_DATIP_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  05D3    562          .ASCIC  " Attempted UNIBUS DATIP/DATO operation."
50 49 54 41 44 20 53 55 42 49 4E 55   05DF
74 61 72 65 70 6F 20 4F 54 41 44 2F   05EB
                        2E 6E 6F 69   05F7
                                 27   05D3
                                      05FB    563 ;-----------------------------------------------------------------------
                                      05FB    564 T_DATOB_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  05FB    565          .ASCIC  " Attempted UNIBUS DATOB operation."
42 4F 54 41 44 20 53 55 42 49 4E 55   0607
   2E 6E 6F 69 74 61 72 65 70 6F 20   0613
                                 22   05FB
                                      061E    566 ;-----------------------------------------------------------------------
                                      061E    567 T_DATO_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00'  061E    568          .ASCIC  " Attempted UNIBUS DATO operation."
20 4F 54 41 44 20 53 55 42 49 4E 55   062A
      2E 6E 6F 69 74 61 72 65 70 6F   0636
                                 21   061E
```

```
                                     0640   569 ;-------------------------------------------------------------------------------
                                     0640   570 T_AO_DATO_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 0640   571         .ASCIC  '' Attempted UNIBUS DATO operation in Address Offset mode.''
20 4F 54 41 44 20 53 55 42 49 4E 55  064C
6E 69 20 6E 6F 69 74 61 72 65 70 6F  0658
66 66 4F 20 73 73 65 72 64 64 41 20  0664
         2E 65 64 6F 6D 20 74 65 73  0670
                                  38 0640
                                     0679   572 ;-------------------------------------------------------------------------------
                                     0679   573 T_AO_DATOB_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 0679   574         .ASCIC  '' Attempted UNIBUS DATOB operation in Address Offset mode.''
42 4F 54 41 44 20 53 55 42 49 4E 55  0685
69 20 6E 6F 69 74 61 72 65 70 6F 20  0691
66 4F 20 73 73 65 72 64 64 41 20 6E  069D
      2E 65 64 6F 6D 20 74 65 73 66  06A9
                                  39 0679
                                     06B3   575 ;-------------------------------------------------------------------------------
                                     06B3   576 T_AO_DATI_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 06B3   577         .ASCIC  '' Attempted UNIBUS DATI operation in Address Offset mode.''
20 49 54 41 44 20 53 55 42 49 4E 55  06BF
6E 69 20 6E 6F 69 74 61 72 65 70 6F  06CB
66 66 4F 20 73 73 65 72 64 64 41 20  06D7
         2E 65 64 6F 6D 20 74 65 73  06E3
                                  38 06B3
                                     06EC   578 ;-------------------------------------------------------------------------------
                                     06EC   579 T_AO_DATIP_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 06EC   580         .ASCIC  '' Attempted UNIBUS DATIP/DATO operation in Address Offset mode.''
50 49 54 41 44 20 53 55 42 49 4E 55  06F8
74 61 72 65 70 6F 20 4F 54 41 44 2F  0704
65 72 64 64 41 20 6E 69 20 6E 6F 69  0710
6F 6D 20 74 65 73 66 66 4F 20 73 73  071C
                           2E 65 64  0728
                                  3E 06EC
                                     072B   581 ;-------------------------------------------------------------------------------
                                     072B   582 T_PG_BND_WR_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 072B   583         .ASCIC  '' Attempted to cause PAGE BOUNDARY WRITE ERROR.''
47 41 50 20 65 73 75 61 63 20 6F 74  0737
57 20 59 52 41 44 4E 55 4F 42 20 45  0743
   2E 52 4F 52 52 45 20 45 54 49 52  074F
                                  2E 072B
                                     075A   584 ;-------------------------------------------------------------------------------
                                     075A   585 T_MAP_FCN_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 075A   586         .ASCIC  '' Attempted testing of UBI mapping function.''
55 20 66 6F 20 67 6E 69 74 73 65 74  0766
66 20 67 6E 69 70 70 61 6D 20 49 42  0772
            2E 6E 6F 69 74 63 6E 75  077E
                                  2B 075A
                                     0786   587 ;-------------------------------------------------------------------------------
                                     0786   588 T_PB_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 0786   589         .ASCIC  '' Attempted to generate Unibus Map Parity Error.''
20 65 74 61 72 65 6E 65 67 20 6F 74  0792
50 20 70 61 4D 20 73 75 62 69 6E 55  079E
2E 72 6F 72 72 45 20 79 74 69 72 61  07AA
                                  2F 0786
                                     07B6   590 ;-------------------------------------------------------------------------------
                                     07B6   591 T_NXM_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 07B6   592         .ASCIC  '' Attempted to generate Unibus NXM Error.''
```

```
20 65 74 61 72 65 6E 65 67 20 6F 74  07C2
45 20 4D 58 4E 20 73 75 62 69 6E 55  07CE
               2E 72 6F 72 72  07DA
                            28  07B6
                               07DF   593 ;------------------------------------------------------------
                               07DF   594 T_INTR_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 07DF   595         .ASCIC  " Attempted to initiate Bus Request Interrupt."
20 65 74 61 69 74 69 6E 69 20 6F 74  07EB
20 74 73 65 75 71 65 52 20 73 75 42  07F7
      2E 74 70 75 72 72 65 74 6E 49  0803
                            2D  07DF
                               080D   596 ;------------------------------------------------------------
                               080D   597 T_MAP_INV_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 080D   598         .ASCIC  " Attempted access through invalid Map Entry."
75 6F 72 68 74 20 73 73 65 63 63 61  0819
4D 20 64 69 6C 61 76 6E 69 20 68 67  0825
         2E 79 72 74 6E 45 20 70 61  0831
                            2C  080D
                               083A   599 ;------------------------------------------------------------
                               083A   600 T_MULT1_ERR:
20 64 65 74 70 6D 65 74 74 41 20 00' 083A   601         .ASCIC  " Attempted to create traffic on the UNIBUS."
72 74 20 65 74 61 65 72 63 20 6F 74  0846
65 68 74 20 6E 6F 20 63 69 66 66 61  0852
            2E 53 55 42 49 4E 55 20  085E
                            2B  083A
                               0866   602 ;------------------------------------------------------------
                               0866   603 T_MXFR_D_ERR:
20 29 44 28 57 55 21 20 20 2F 21 00' 0866   604         .ASCIC  "!/  !UW(D) data word!%S in error."
53 25 21 64 72 6F 77 20 61 74 61 64  0872
      2E 72 6F 72 72 65 20 6E 69 20  087E
                            21  0866
                               0888   605 ;------------------------------------------------------------
                               0888   606 T_NO_CSR_ERR:
52 53 43 20 49 42 55 20 20 2F 21 00' 0888   607         .ASCIC  "!/  UBI CSR failed to respond when accessed."
72 20 6F 74 20 64 65 6C 69 61 66 20  0894
20 6E 65 68 77 20 64 6E 6F 70 73 65  08A0
         2E 64 65 73 73 65 63 63 61  08AC
                            2C  0888
                               08B5   608 ;------------------------------------------------------------
                               08B5   609 T_D_ERR_HDR:
6C 6F 66 20 65 68 54 20 20 2F 21 00' 08B5   610         .ASCIC  "!/  The following data error occurred:!/"-
20 61 74 61 64 20 67 6E 69 77 6F 6C  08C1
72 72 75 63 63 6F 20 72 6F 72 72 65  08CD
               2F 21 3A 64 65  08D9
20 20 20 6E 6F 69 74 61 63 6F 4C 09  08DE   611                 "      Location    Expected      Actual          XOR!/"-
20 20 09 64 65 74 63 65 70 78 45 20  08EA
20 20 20 20 20 20 6C 61 75 74 63 41  08F6
            2F 21 52 4F 58 20 20 20  0902
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 09  090A   612                 "      --------    --------    --------    --------!/"
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 20  0916
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 20  0922
   2F 21 2D 2D 2D 2D 2D 2D 2D 2D 20  092E
                            83  08B5
                               0939   613 ;------------------------------------------------------------
                               0939   614 T_MXFR_TYPE:
65 66 73 6E 61 72 54 20 20 2F 21 00' 0939   615         .ASCIC  "!/  Transfer description:"
6F 69 74 70 69 72 63 73 65 64 20 72  0945
```

```
                                   3A 6E  0951
                                      19  0939
                                          0953    616 ;--------------------------------------------------------------------
                                          0953    617 T_BUFF_ADR:
20 72 65 66 66 75 42 20 20 2F 21 00'      0953    618         .ASCIC  "!/  Buffer address:     !XL"
4C 58 21 09 3A 73 73 65 72 64 64 61       095F
                                      17  0953
                                          096B    619 ;--------------------------------------------------------------------
                                          096B    620 T_M1_UBE_0:
20 29 44 28 30 30 30 32 09 2F 21 00'      096B    621         .ASCIC  "!/     2000(D) BR6 DATIP/DATOB sincle-word transfers with"-
41 44 2F 50 49 54 41 44 20 36 52 42       0977
77 2D 65 6C 67 6E 69 73 20 42 4F 54       0983
72 65 66 73 6E 61 72 74 20 64 72 6F       098F
                  68 74 69 77 20 73       099B
65 6C 20 65 74 61 74 6F 72 09 2F 21       09A1    622                 "!/     rotate left after DATIP, interrupt on done, same"-
54 41 44 20 72 65 74 66 61 20 74 66       09AD
70 75 72 72 65 74 6E 69 20 2C 50 49       09B9
73 20 2C 65 6E 6F 64 20 6E 6F 20 74       09C5
                           65 6D 61       09D1
73 61 20 72 65 66 66 75 62 09 2F 21       09D4    623                 "!/     buffer as UBE 1 (if selected)"
73 20 66 69 28 20 31 20 45 42 55 20       09E0
            29 64 65 74 63 65 6C 65       09EC
                                      88  096B
                                          09F4    624 ;--------------------------------------------------------------------
                                          09F4    625 T_M1_UBE_1:
20 29 44 28 30 30 30 32 09 2F 21 00'      09F4    626         .ASCIC  "!/     2000(D) NPR DATIP/DATO single-word transfers without"-
41 44 2F 50 49 54 41 44 20 52 50 4E       0A00
6F 77 2D 65 6C 67 6E 69 73 20 4F 54       0A0C
73 72 65 66 73 6E 61 72 74 20 64 72       0A18
            74 75 6F 68 74 69 77 20       0A24
69 20 2C 65 74 61 74 6F 72 09 2F 21       0A2C    627                 "!/     rotate, interrupt on done BR7, same buffer as UBE 0"
20 6E 6F 20 74 70 75 72 72 65 74 6E       0A38
61 73 20 2C 37 52 42 20 65 6E 6F 64       0A44
73 61 20 72 65 66 66 75 62 20 65 6D       0A50
                  30 20 45 42 55 20       0A5C
                                      6D  09F4
                                          0A62    628 ;--------------------------------------------------------------------
                                          0A62    629 T_M1_UBE_2:
20 29 44 28 30 30 30 31 09 2F 21 00'      0A62    630         .ASCIC  "!/     1000(D) NPR DATO single-word transfers, interrupt on"-
6E 69 73 20 4F 54 41 44 20 52 50 4E       0A6E
61 72 74 20 64 72 6F 77 2D 65 6C 67       0A7A
65 74 6E 69 20 2C 73 72 65 66 73 6E       0A86
            6E 6F 20 74 70 75 72 72       0A92
   37 52 42 20 65 6E 6F 64 09 2F 21       0A9A    631                 "!/     done BR7"
                                      42  0A62
                                          0AA5    632 ;--------------------------------------------------------------------
                                          0AA5    633 T_M1_UBE_3:
20 29 44 28 30 30 30 31 09 2F 21 00'      0AA5    634         .ASCIC  "!/     1000(D) BR5 DATO single-word transfers, interrupt on"-
6E 69 73 20 4F 54 41 44 20 35 52 42       0AB1
61 72 74 20 64 72 6F 77 2D 65 6C 67       0ABD
65 74 6E 69 20 2C 73 72 65 66 73 6E       0AC9
            6E 6F 20 74 70 75 72 72       0AD5
   35 52 42 20 65 6E 6F 64 09 2F 21       0ADD    635                 "./     done BR5"
                                      42  0AA5
                                          0AE8    636 ;--------------------------------------------------------------------
                                          0AE8    637 T_M2_UBE_0:
20 29 44 28 30 30 30 32 09 2F 21 00'      0AE8    638         .ASCIC  "!/     2000(D) NPR DATOB double-word transfers, data from"-
```

```
6F 64 20 42 4F 54 41 44 20 52 50 4E  0AF4
72 74 20 64 72 6F 77 2D 65 6C 62 75  0B00
74 61 64 20 2C 73 72 65 66 73 6E 61  0B0C
                6D 6F 72 66 20 61  0B18
75 6F 43 20 65 6C 63 79 43 09 2F 21  0B1E    639                 "!/     Cycle Count register, interrupt on done BR6"
2C 72 65 74 73 69 67 65 72 20 74 6E  0B2A
6F 20 74 70 75 72 72 65 74 6E 69 20  0B36
      36 52 42 20 65 6E 6F 64 20 6E  0B42
                                63  0AE8
                                    0B4C    640 ;-------------------------------------------------------------------------
                                    0B4C    641 T_M2_UBE_1:
20 29 44 28 30 30 30 31 09 2F 21 00' 0B4C    642             .ASCIC  "!/     1000(D) NPR DATIP/DATO single-word transfers with"-
41 44 2F 50 49 54 41 44 20 52 50 4E  0B58
6F 77 2D 65 6C 67 6E 69 73 20 4F 54  0B64
73 72 65 66 73 6E 61 72 74 20 64 72  0B70
                  68 74 69 77 20  0B7C
66 61 20 65 74 61 74 6F 72 09 2F 21  0B81    643                 "!/     rotate after DATIP, interrupt on done BR7"
69 20 2C 50 49 54 41 44 20 72 65 74  0B8D
20 6E 6F 20 74 70 75 72 72 65 74 6E  0B99
         37 52 42 20 65 6E 6F 64  0BA5
                                60  0B4C
                                    0BAD    644 ;-------------------------------------------------------------------------
                                    0BAD    645 T_M2_UBE_2:
20 29 44 28 30 30 30 31 09 2F 21 00' 0BAD    646             .ASCIC  "!/     1000(D) BR5 DATI double-word transfers, interrupt on"-
75 6F 64 20 49 54 41 44 20 35 52 42  0BB9
61 72 74 20 64 72 6F 77 2D 65 6C 62  0BC5
65 74 6E 69 20 2C 73 72 65 66 73 6E  0BD1
         6E 6F 20 74 70 75 72 72  0BDD
2C 35 52 42 20 65 6E 6F 64 09 2F 21  0BE5    647                 "!/     done BR5, Data register checked ONLY once (after done)"
74 73 69 67 65 72 20 61 74 61 44 20  0BF1
4F 20 64 65 6B 63 65 68 63 20 72 65  0BFD
66 61 28 20 65 63 6E 6F 20 59 4C 4E  0C09
         29 65 6E 6F 64 20 72 65 74  0C15
                                70  0BAD
                                    0C1E    648 ;-------------------------------------------------------------------------
                                    0C1E    649 T_M2_UBE_3:
20 29 44 28 30 30 30 33 09 2F 21 00' 0C1E    650             .ASCIC  "!/     3000(D) BR7 DATI single-word transfers, interrupt"-
6E 69 73 20 49 54 41 44 20 37 52 42  0C2A
61 72 74 20 64 72 6F 77 2D 65 6C 67  0C36
65 74 6E 69 20 2C 73 72 65 66 73 6E  0C42
                  74 70 75 72 72  0C4E
63 61 65 20 72 65 74 66 61 09 2F 21  0C53    651                 "!/     after each transfer, Data register checked ONLY once"-
20 2C 72 65 66 73 6E 61 72 74 20 68  0C5F
65 74 73 69 67 65 72 20 61 74 61 44  0C6B
4E 4F 20 64 65 6B 63 65 68 63 20 72  0C77
               65 63 6E 6F 20 59 4C  0C83
69 66 20 72 65 74 66 61 28 09 2F 21  0C8A    652                 "!/     (after final done)"
         29 65 6E 6F 64 20 6C 61 6E  0C96
                                80  0C1E
                                    0C9F    653 ;-------------------------------------------------------------------------
                                    0C9F    654 T_MAPS_USED:
73 73 65 72 64 64 41 20 20 2F 21 00' 0C9F    655             .ASCIC  "!/  Address of first Mao used: !XL(X)!/"
61 4D 20 74 73 72 69 66 20 66 6F 20  0CAB
28 4C 58 21 20 3A 64 65 73 75 20 70  0CB7
                     2F 21 29 58  0CC3
                                27  0C9F
                                    0CC7    656 ;-------------------------------------------------------------------------
```

```
                                                OCC7    657 T_MAP_USED:
73 73 65 72 64 64 41 20 20 2F 21 00'            OCC7    658         .ASCIC  "!/  Address of Map used: !XL(X)"
64 65 73 75 20 70 61 4D 20 66 6F 20             OCD3
            29 58 28 4C 58 21 20 3A             OCDF
                                  1F            OCC7
                                                OCE7    659 ;-----------------------------------------------------------------------
                                                OCE7    660 T_MXFR_ERR_HDR:
72 69 66 20 65 68 54 20 20 2F 21 00'            OCE7    661         .ASCIC  "!/  The first eight (or fewer) errors are:!/"-
72 6F 28 20 74 68 67 69 65 20 74 73             OCF3
6F 72 72 65 20 29 72 65 77 65 66 20             OCFF
            2F 21 3A 65 72 61 20 73 72          OD0B
20 20 20 6E 6F 69 74 61 63 6F 4C 09             OD14    662                 "       Location    Expected      Actual          XOR!/"-
20 20 20 64 65 74 63 65 70 78 45 20             OD20
20 20 20 6C 61 75 74 63 41 20 20 20             OD2C
   2F 21 52 4F 58 20 20 20 20 20 20             OD38
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 09             OD43    663                 "       --------    --------    --------    --------!/"
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 20             OD4F
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 20             OD5B
   2F 21 2D 2D 2D 2D 2D 2D 2D 2D 20             OD67
                                  8A            OCE7
                                                OD72    664 ;-----------------------------------------------------------------------
                                                OD72    665 T_BYTE_ERR:
20 20 20 20 20 20 20 4C 58 21 09 00'            OD72    666         .ASCIC  "       !XL         !XB         !XB         !XB!/"
20 20 20 20 20 20 42 58 21 20 20 20             OD7E
20 20 20 20 20 42 58 21 20 20 20 20             OD8A
      2F 21 42 58 21 20 20 20 20 20             OD96
                                  2D            OD72
                                                ODA0    667 ;-----------------------------------------------------------------------
                                                ODA0    668 T_WORD_ERR:
20 20 20 20 20 20 20 4C 58 21 09 00'            ODA0    669         .ASCIC  "       !XL         !XW         !XW         !XW!/"
20 20 20 20 20 20 20 20 57 58 21 20             ODAC
21 20 20 20 20 20 20 20 20 57 58 21             ODB8
                        2F 21 57 58             ODC4
                                  27            ODA0
                                                ODC8    670 ;-----------------------------------------------------------------------
                                                ODC8    671 T_LONG_ERR:
4C 58 21 20 20 20 20 4C 58 21 09 00'            ODC8    672         .ASCIC  "       !XL     !XL     !XL     !XL!/"
21 20 20 20 20 4C 58 21 20 20 20 20             ODD4
                        2F 21 4C 58             ODE0
                                  1B            ODC8
                                                ODE4    673 ;-----------------------------------------------------------------------
                                                ODE4    674 T_UNX_SSYN_ERR:
65 63 65 72 20 4E 59 53 53 20 20 00'            ODE4    675         .ASCIC  "  SSYN received for invalid reference.!/"
76 6E 69 20 72 6F 66 20 64 65 76 69             ODF0
6E 65 72 65 66 65 72 20 64 69 6C 61             ODFC
                  2F 21 2E 65 63                0E08
                                  28            ODE4
                                                0E0D    676 ;-----------------------------------------------------------------------
                                                0E0D    677 T_NO_RDY_ERR:
79 64 61 65 52 20 45 42 55 20 20 00'            0E0D    678         .ASCIC  "  UBE Ready bit failed to set.  Test will abort.!/"
20 64 65 6C 69 61 66 20 74 69 62 20             0E19
73 65 54 20 20 2E 74 65 73 20 6F 74             0E25
74 72 6F 62 61 20 6C 6C 69 77 20 74             0E31
                        2F 21 2E                0E3D
                                  32            0E0D
                                                0E40    679 ;-----------------------------------------------------------------------
                                                0E40    680 T_WNG_GNT_ERR:
```

```
65 72 20 73 75 62 69 6E 55 20 20 00' 0E40   681          .ASCIC  '' Unibus returned wrong grant or no grant.!/''
67 6E 6F 72 77 20 64 65 6E 72 75 74  0E4C
6F 6E 20 72 6F 20 74 6E 61 72 67 20  0E58
         2F 21 2E 74 6E 61 72 67 20  0E64
                                 2C  0E40
                                     0E6D   682 ;---------------------------------------------------------------------------
                                     0E6D   683 T_MAX_LAT_ERR:
20 61 74 61 44 20 45 42 55 20 20 00' 0E6D   684          .ASCIC  '' UBE Data Buffer was not read within allotted time../''
6E 20 73 61 77 20 72 65 66 66 75 42  0E79
68 74 69 77 20 64 61 65 72 20 74 6F  0E85
20 64 65 74 74 6F 6C 6C 61 20 6E 69  0E91
               2F 21 2E 65 6D 69 74  0E9D
                                 36  0E6D
                                     0EA4   685 ;---------------------------------------------------------------------------
                                     0EA4   686 T_NO_SACK_ERR:
65 6C 69 61 66 20 55 50 43 20 20 00' 0EA4   687          .ASCIC  '' CPU failed to timeout on No SACK.!/''
74 75 6F 65 6D 69 74 20 6F 74 20 64  0EB0
2E 4B 43 41 53 20 6F 4E 20 6E 6F 20  0EBC
                           2F 21     0EC8
                                 25  0EA4
                                     0ECA   688 ;---------------------------------------------------------------------------
                                     0ECA   689 T_NO_SSYN_ERR:
20 74 6F 6E 20 4E 59 53 53 20 20 00' 0ECA   690          .ASCIC  '' SSYN not received within 10 usec of MSYN assertion.!/''
74 69 77 20 64 65 76 69 65 63 65 72  0ED6
20 63 65 73 75 20 30 31 20 6E 69 68  0EE2
65 73 73 61 20 4E 59 53 4D 20 66 6F  0EEF
               2F 21 2E 6E 6F 69 74 72  0EF
                                 37  0ECA
                                     0F02   691 ;---------------------------------------------------------------------------
                                     0F02   692 T_WNG_A_ERR:
62 20 34 20 72 65 70 70 55 20 20 00' 0F02   693          .ASCIC  '' Upper 4 bits of Unibus address failed to match those!/''-
75 62 69 6E 55 20 66 6F 20 73 74 69  0F0E
61 66 20 73 73 65 72 64 64 61 20 73  0F1A
63 74 61 6D 20 6F 74 20 64 65 6C 69  0F26
               2F 21 65 73 6F 68 74 20 68  0F32
61 20 45 42 55 20 65 68 74 20 66 6F  0F3B   694                       ''of the UBE address./!''
               21 2F 2E 73 73 65 72 64 64  0F47
                                 4D  0F02
                                     0F50   695 ;---------------------------------------------------------------------------
                                     0F50   696 T_NO_GNT_ERR:
69 65 63 65 72 20 45 42 55 20 20 00' 0F50   697          .ASCIC  '' UBE received Grant of less than 75 nsec duration or!/''-
66 6F 20 74 6E 61 72 47 20 64 65 76  0F5C
37 20 6E 61 68 74 20 73 73 65 6C 20  0F68
74 61 72 75 64 20 63 65 73 6E 20 35  0F74
               2F 21 72 6F 20 6E 6F 69  0F80
6E 6F 20 6E 61 68 74 20 65 72 6F 6D  0F88   698                       ''more than one Grant.!/''
         2F 21 2F 74 6E 61 72 47 20 65  0F94
                                 4D  0F50
                                     0F9E   699 ;---------------------------------------------------------------------------
                                     0F9E   700 T_INT_SSYN_ERR:
65 6C 69 61 66 20 55 50 43 20 20 00' 0F9E   701          .ASCIC  '' CPU failed to return SSYN within 3 usec after bus!/''-
20 6E 72 75 74 65 72 20 6F 74 20 64  0FAA
20 6E 69 68 74 69 77 20 4E 59 53 53  0FB6
72 65 74 66 61 20 63 65 73 75 20 33  0FC2
                  2F 21 73 75 62 20  0FCE
73 61 20 74 70 75 72 72 65 74 6E 69  0FD4   702                       ''interrupt assertion../''
         2F 21 2E 6E 6F 69 74 72 65 73  0FE0
```

```
                                                 4B  0F9E
                                                     0FEA    703 ;-----------------------------------------------------------------------
                                                     0FEA    704 T_UBI_RDS_ERR:
74 63 65 72 72 6F 63 6E 55 20 20 00'                 0FEA    705         .ASCIC  '' Uncorrectable ECC error during Unibus transfer.!/''
72 72 65 20 43 43 45 20 65 6C 62 61                  0FF6
6E 55 20 67 6E 69 72 75 64 20 72 6F                  1002
65 66 73 6E 61 72 74 20 73 75 62 69                  100E
                              2F 21 2E 72            101A
                                 33                  0FEA
                                                     101E    706 ;-----------------------------------------------------------------------
                                                     101E    707 T_UBI_NXM_ERR:
74 61 20 73 75 62 69 6E 55 20 20 00'                 101E    708         .ASCIC  '' Unibus attempted to access Non-existent Memory.!/''
61 20 6F 74 20 64 65 74 70 6D 65 74                  102A
78 65 2D 6E 6F 4E 20 73 73 65 63 63                  1036
72 6F 6D 65 4D 20 74 6E 65 74 73 69                  1042
                              2F 21 2E 79            104E
                                 33                  101E
                                                     1052    709 ;-----------------------------------------------------------------------
                                                     1052    710 T_NO_NXM_ERR:
4E 20 52 53 43 20 49 42 55 20 20 00'                 1052    711         .ASCIC  '' UBI CSR NXM bit did not set when expected.!/''
6E 20 64 69 64 20 74 69 62 20 4D 58                  105E
20 6E 65 68 77 20 74 65 73 20 74 6F                  106A
   2F 21 2E 64 65 74 63 65 70 78 65                  1076
                                 2E                  1052
                                                     1081    712 ;-----------------------------------------------------------------------
                                                     1081    713 T_UBI_PB_ERR:
61 4D 20 73 75 62 69 6E 55 20 20 00'                 1081    714         .ASCIC  '' Unibus Map Parity error occurred.!/''
72 72 65 20 79 74 69 72 61 50 20 70                  108D
2E 64 65 72 72 75 63 63 6F 20 72 6F                  1099
                              2F 21                  10A5
                                 25                  1081
                                                     10A7    715 ;-----------------------------------------------------------------------
                                                     10A7    716 T_NO_PB_ERR:
50 20 52 53 43 20 49 42 55 20 20 00'                 10A7    717         .ASCIC  '' UBI CSR Parity error bit did not set when expected.!/''
20 72 6F 72 72 65 20 79 74 69 72 61                  10B3
20 74 6F 6E 20 64 69 64 20 74 69 62                  10BF
70 78 65 20 6E 65 68 77 20 74 65 73                  10CB
            2F 21 2E 64 65 74 63 65                  10D7
                                 37                  10A7
                                                     10DF    718 ;-----------------------------------------------------------------------
                                                     10DF    719 T_UBI_WR_ERR:
74 61 20 73 75 62 69 6E 55 20 20 00'                 10DF    720         .ASCIC  '' Unibus attempted a write with invalid Map Register.!/''
72 77 20 61 20 64 65 74 70 6D 65 74                  10EB
76 6E 69 20 68 74 69 77 20 65 74 69                  10F7
67 65 52 20 70 61 4D 20 64 69 6C 61                  1103
            2F 21 2E 72 65 74 73 69                  110F
                                 37                  10DF
                                                     1117    721 ;-----------------------------------------------------------------------
                                                     1117    722 T_NO_WR_ERR:
72 57 20 73 75 62 69 6E 55 20 20 00'                 1117    723         .ASCIC  '' Unibus Write error bit did not set when expected.!/''
69 62 20 72 6F 72 72 65 20 65 74 69                  1123
65 73 20 74 6F 6E 20 64 69 64 20 74                  112F
63 65 70 78 65 20 6E 65 68 77 20 74                  113B
               2F 21 2E 64 65 74                     1147
                                 35                  1117
                                                     114D    724 ;-----------------------------------------------------------------------
                                                     114D    725 T_CSR_NCL_ERR:
```

```
66 20 52 53 43 20 49 42 55 20 20 00'  114D    726          .ASCIC  '' UBI CSR failed to clear when read.!/''
65 6C 63 20 6F 74 20 64 65 6C 69 61   1159
64 61 65 72 20 6E 65 68 77 20 72 61   1165
                        2F 21 2E      1171
                              26      114D
                              1174    727 ;-----------------------------------------------------------------------
                              1174    728 T_EXP_INTR:
45 54 43 45 50 58 45 20 20 2F 21 00'  1174    729          .ASCIC  "!/  EXPECTED: BR!XB Interrupt at Vector = 0148(X) IPL = !XB .!/"
74 6E 49 20 42 58 21 52 42 20 3A 44   1180
65 56 20 74 61 20 74 70 75 72 72 65   118C
28 38 34 31 30 20 3D 20 72 6F 74 63   1198
42 58 21 20 3D 20 4C 50 49 20 29 58   11A4
                  2F 21 2E 20         11B0
                              3F      1174
                              11B4    730 ;-----------------------------------------------------------------------
                              11B4    731 T_UNXP_INTR:
45 54 43 45 50 58 45 20 20 2F 21 00'  11B4    732          .ASCIC  "!/  EXPECTED: NO interrupt.  IPL = !XB .!/"
72 72 65 74 6E 69 20 4F 4E 20 3A 44   11C0
20 3D 20 4C 50 49 20 20 2E 74 70 75   11CC
              2F 21 2E 20 42 58 21    11D8
                              2A      11B4
                              11DF    733 ;-----------------------------------------------------------------------
                              11DF    734 T_NO_INTR:
20 20 3A 4C 41 55 54 43 41 20 20 00'  11DF    735          .ASCIC  '' ACTUAL:   No Interrupt occurred.!/''
70 75 72 72 65 74 6E 49 20 6F 4E 20   11EB
21 2E 64 65 72 72 75 63 63 6F 20 74   11F7
                              2F      1203
                              24      11DF
                              1204    736 ;-----------------------------------------------------------------------
                              1204    737 T_ACTL_INTR:
20 20 3A 4C 41 55 54 43 41 20 20 00'  1204    738          .ASCIC  '' ACTUAL:   BR!XB Interrupt at Vector = !XW(X).!/''
72 65 74 6E 49 20 42 58 21 52 42 20   1210
74 63 65 56 20 74 61 20 74 70 75 72   121C
2E 29 58 28 57 58 21 20 3D 20 72 6F   1228
                        2F 21         1234
                              31      1204
                              1236    739 ;-----------------------------------------------------------------------
                              1236    740 T_MAP_ADR_ERR:
64 64 41 20 70 61 4D 20 20 2F 21 00'  1236    741          .ASCIC  "!/  Map Address   Data Encoded  Data Loaded   "-
20 61 74 61 44 20 20 20 73 73 65 72   1242
74 61 44 20 20 64 65 64 6F 63 6E 45   124E
   20 20 20 64 65 64 61 6F 4C 20 61   125A
76 65 69 72 74 65 52 20 61 74 61 44   1265    742                "Data Retrieved  Data Decoded !/  ----------   ----------"-
63 65 44 20 61 74 61 44 20 20 64 65   1271
2D 2D 2D 20 20 2F 21 20 64 65 64 6F   127D
2D 20 20 20 2D 2D 2D 2D 2D 2D 2D 2D   1289
   2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D   1295
2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 20 20   12A0    743                "  ----------   -------------  ----------!/     !XL      "-
2D 2D 2D 2D 2D 2D 2D 2D 20 20 20 2D   12AC
2D 2D 2D 2D 20 20 2D 2D 2D 2D 2D 2D   12B8
20 20 2F 21 2D 2D 2D 2D 2D 2D 2D 2D   12C4
20 20 20 20 20 20 4C 58 21 20 20 20   12D0
4C 58 21 20 20 20 20 20 4C 58 21 20   12DC    744                " .XL      .XL          !XL      !XL!/"
4C 58 21 20 20 20 20 20 20 20 20 20   12E8
   2F 21 4C 58 21 20 20 20 20 20 20   12F4
                              C8      1236
                              12FF    745 ;-----------------------------------------------------------------------
```

```
                                        12FF    746 T_CHANNEL_ERR:
72 75 64 20 72 6F 72 72 45 20 20 00'    12FF    747         .ASCIC  "  Error during Channel Services call.  Test will be aborted."
20 6C 65 6E 6E 61 68 43 20 67 6E 69     130B
6C 61 63 20 73 65 63 69 76 72 65 53     1317
6C 69 77 20 74 73 65 54 20 20 2E 6C     1323
64 65 74 72 6F 62 61 20 65 62 20 6C     132F
                              2E        133B
                              3C        12FF
                                        133C    748 ;-----------------------------------------------------------------------
                                        133C    749 T_IHWE:
74 73 69 78 65 2D 65 72 50 20 20 00'    133C    750         .ASCIC  "  Pre-existing adapter condition.!/"
20 72 65 74 70 61 64 61 20 67 6E 69     1348
2F 21 2E 6E 6F 69 74 69 64 6E 6F 63     1354
                              23        133C
                                        1360    751 ;-----------------------------------------------------------------------
                                        1360    752 T_LOGIC:
63 2F 74 65 73 20 74 69 42 20 20 00'    1360    753         .ASCIC  "  Bit set/clear failure of Interrupt Enable bit.!/"
65 72 75 6C 69 61 66 20 72 61 65 6C     136C
70 75 72 72 65 74 6E 49 20 66 6F 20     1378
74 69 62 20 65 6C 62 61 6E 45 20 74     1384
                        2F 21 2E        1390
                              32        1360
                                        1393    754 ;-----------------------------------------------------------------------
                                        1393    755 T_IVVECT:
76 20 64 69 6C 61 76 6E 49 20 20 00'    1393    756         .ASCIC  "  Invalid vector received.  Vector=!XW(X).!/"
76 69 65 63 65 72 20 72 6F 74 63 65     139F
3D 72 6F 74 63 65 56 20 20 2E 64 65     13AB
         2F 21 2E 29 58 28 57 58 21     13B7
                              2C        1393
                                        13C0    757 ;-----------------------------------------------------------------------
                                        13C0    758
                                        13C0    759
```

```
                                13C0   761
                                13C0   762 .SBTTL  ERROR REPORTING ROUTINES
                                13C0   763
                                13C0   764 ;-------------------------------------------------------------------------------
                                13C0   765
                                13C0   766 ; This extended error-reporting routine prints a header message and the follow-
                                13C0   767 ; ing: location, expected data, actual data, and exclusive-or of expected and
                                13C0   768 ; actual data for each data error found.
                                13C0   769
                                13C0   770 DATA_ERR:
                                13C0   771
                                13C0   772          $DS_BGNMESSAGE
                          0000  13C0                .WORD   ^M<>                ; ENTRY MASK
                                13C2   773
                                13C2   774 1$:      $DS_PRINTB_S    FORMAT=T_D_ERR_HDR      ; print header
                   F4EF CF   9F 13C2                PUSHAB  T_D_ERR_HDR
          000100E0 9F    01   FB 13C6                CALLS   #$$N, @#DS$PRINTB
                                13CD   775
     5A   18 AC    1C AC    CD 13CD   776 3$:      XORL3   ERR$_P3(AP),ERR$_P2(AP),R10      ; XOR actual and expected
                                13D3   777
                                13D3   778          $DS_PRINTB_S    FORMAT=@ERR$_P4(AP),-   ; print error description
                                13D3   779                          P1=ERR$_P1(AP),-        ; location
                                13D3   780                          P2=ERR$_P2(AP),-        ; expected
                                13D3   781                          P3=ERR$_P3(AP),-        ; actual
                                13D3   782                          P4=R10                  ; XOR
                      5A    DD 13D3                   PUSHL   R10
                   1C AC    DD 13D5                   PUSHL   ERR$_P3(AP)
                   18 AC    DD 13D8                   PUSHL   ERR$_P2(AP)
                   14 AC    DD 13DB                   PUSHL   ERR$_P1(AP)
                   20 BC    9F 13DE                   PUSHAB  @ERR$_P4(AP)
          000100E0 9F    05   FB 13E1                   CALLS   #$$N, @#DS$PRINTB
                                13E8   783 9$:
                                13E8   784          $DS_ENDMESSAGE
                          04    13E8                   RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                                13E9   785
                                13E9   786
                                13E9   787 ; This extended error-reporting routine is called whenever a map address bus
                                13E9   788 ; error is encountered using the Reed-Muller error correcting code.  It prints
                                13E9   789 ; the Map Address, value loaded, value retrieved, value encoded, and value
                                13E9   790 ; decoded.
                                13E9   791
                                13E9   792 MAP_ADR_ERR:
                                13E9   793
                                13E9   794          $DS_BGNMESSAGE
                          0000  13E9                .WORD   ^M<>                ; ENTRY MASK
                                13EB   795
54   80000000 8F    55   9C 13EB   796          ROTL    R5,#1@31,R4                     ; calc map offset
     54   80000000 8F    CA 13F3   797          BICL2   #1@31,R4                        ; clear bit 31
     54   00F26800 8F    C0 13FA   798          ADDL2   #A_MAP_BGN,R4                   ; add map base
                                1401   799
                                1401   800          $DS_PRINTB_S    FORMAT=T_MAP_ADR_ERR,-  ; print error header
                                1401   801                          P1=R4,-                 ; map address
                                1401   802                          P2=R5,-                 ; value encoded
                                1401   803                          P3=RD_MLLR_TBL[R5],-    ; value loaded
                                1401   804                          P4=RETRIEVE_TBL[R5],-   ; value retrieved
                                1401   805                          P5=DECODE_TBL[R5]       ; decoded value
                EE71 CF45    DD 1401                   PUSHL   DECODE_TB[[R5]
```

```
                EE4C CF45   DD  1406                    PUSHL   RETRIEVE_TBL[R5]
                EE27 CF45   DD  140B                    PUSHL   RD_MLLR_TBL[R5]
                     55     DD  1410                    PUSHL   R5
                     54     DD  1412                    PUSHL   R4
                FE1E CF     9F  1414                    PUSHAB  T_MAP_ADR_ERR
          000100E0 9F   06   FB  1418                    CALLS   #$$N, @#DS$PRINTB
                            141F   806
                            141F   807          $DS_ENDMESSAGE
                           04  141F                    RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                            1420   808
                            1420   809
                            1420   810 ; This extended error-reporting routine is called when any execution error
                            1420   811 ; occurs during a Unibus data transaction.  The status bits in the UBE CR2 are
                            1420   812 ; checked and the appropriate errors are reported.
                            1420   813
                            1420   814 XFER_ERR:
                            1420   815
                            1420   816          $DS_BGNMESSAGE
                         0000  1420                    .WORD   ^M<>            ; ENTRY MASK
                            1422   817
             42 EBD9 CF    0C   E1  1422   818          BBC     #V_DATA_ERR,W_FLAGS,10$         ; br = no data error
          5A   1C AC    18 AC   CD  1428   819          XORL3   ERR$_P2(AP),ERR$_P3(AP),R10      ; load XOR
                            142E   820
                            142E   821 1$:       $DS_PRINTB_S    FORMAT=T_D_ERR_HDR      ; print header
                F483 CF     9F  142E                    PUSHAB  T_D_ERR_HDR
          000100E0 9F   01   FB  1432                    CALLS   #$$N, @#DS$PRINTB
                            1439   822
                            1439   823 2$:       $DS_PRINTB_S    FORMAT=@ERR$_P4(AP),-   ; print error description
                            1439   824                     P1=ERR$_P1(AP),-        ; location
                            1439   825                     P2=ERR$_P2(AP),-        ; expected
                            1439   826                     P3=ERR$_P3(AP),-        ; actual
                            1439   827                     P4=R10                  ; XOR
                     5A     DD  1439                    PUSHL   R10
                  1C AC     DD  143B                    PUSHL   ERR$_P3(AP)
                  18 AC     DD  143E                    PUSHL   ERR$_P2(AP)
                  14 AC     DD  1441                    PUSHL   ERR$_P1(AP)
                  20 BC     9F  1444                    PUSHAB  @ERR$_P4(AP)
          000100E0 9F   05   FB  1447                    CALLS   #$$N, @#DS$PRINTB
                            144E   828
               57   24 AC    D0  144E   829          MOVL    ERR$_P5(AP),R7                  ; load map number
            57   57   02    78  1452   830          ASHL    #2,R7,R7                        ; calc Map offset
          57   00F26800 8F    C0  1456   831          ADDL2   #A_MAP_BGN,R7                   ; add map base
                            145D   832 3$:       $DS_PRINTX_S    FORMAT=T_MAP_USED,-     ; report map used
                            145D   833                     P1=R7
                     57     DD  145D                    PUSHL   R7
                F864 CF     9F  145F                    PUSHAB  T_MAP_USED
          000100E8 9F   02   FB  1463                    CALLS   #$$N, @#DS$PRINTX
                            146A   834
               00001471'EF    16  146A   835 10$:      JSB     CSR_CHK                 ; call CSR check routine
                            1470   836          $DS_ENDMESSAGE
                           04  1470                    RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                            1471   837
                            1471   838
                            1471   839 ; This subroutine is called by prlinks XFER_ERR and MXFR_ERR via a JSB.
                            1471   840 ; It checks Control Register 2 of the appropriate UBE (R8 = device number)
                            1471   841 ; and the UBI CSR and prints error messages accordingly.
                            1471   842
```

```
                                1471   843 CSR_CHK:                                    ; entry point for Multi-Xfrs
      57   58   04   9C        1471   844          ROTL    #4,R8,R7                   ; calc index from dev number
                                1475   845
 0B EE44 DF47   07   E0        1475   846          BBS     #V_RDY,@A_BE0_CR1[R7],4$   ; br = RDY bit set
                                147C   847          $DS_PRINTB_S    FORMAT=T_NO_RDY_ERR
          F98D CF   9F          147C                 PUSHAB  T_NO_RDY_ERR
  000100E0 9F   01   FB        1480                 CALLS   #$$N, @#DS$PRINTB
                                1487   848
 0B EE2E DF47   05   E1        1487   849 4$:      BBC     #V_WNG_GNT,@A_BE0_CR2[R7],6$ ; br = no wrong grant error
                                148E   850          $DS_PRINTB_S    FORMAT=T_WNG_GNT_ERR
          F9AE CF   9F          148E                 PUSHAB  T_WNG_GNT_ERR
  000100E0 9F   01   FB        1492                 CALLS   #$$N, @#DS$PRINTB
                                1499   851
 0B EE1C DF47   06   E1        1499   852 6$:      BBC     #V_MAX_LAT,@A_BE0_CR2[R7],8$ ; br = no max. latency error
                                14A0   853          $DS_PRINTB_S    FORMAT=T_MAX_LAT_ERR
          F9C9 CF   9F          14A0                 PUSHAB  T_MAX_LAT_ERR
  000100E0 9F   01   FB        14A4                 CALLS   #$$N, @#DS$PRINTB
                                14AB   854
 0B EE0A DF47   07   E1        14AB   855 8$:      BBC     #V_NO_SACK,@A_BE0_CR2[R7],10$ ; br = no no-SACK error
                                14B2   856          $DS_PRINTB_S    FORMAT=T_NO_SACK_ERR
          F9EE CF   9F          14B2                 PUSHAB  T_NO_SACK_ERR
  000100E0 9F   01   FB        14B6                 CALLS   #$$N, @#DS$PRINTB
                                14BD   857
 0B EDF8 DF47   08   E1        14BD   858 10$:     BBC     #V_NO_SSYN,@A_BE0_CR2[R7],12$ ; br = no no-SSYN error
                                14C4   859          $DS_PRINTB_S    FORMAT=T_NO_SSYN_ERR
          FA02 CF   9F          14C4                 PUSHAB  T_NO_SSYN_ERR
  000100E0 9F   01   FB        14C8                 CALLS   #$$N, @#DS$PRINTB
                                14CF   860
 0B EDE6 DF47   09   E1        14CF   861 12$:     BBC     #V_WNG_A,@A_BE0_CR2[R7],14$  ; br = no wrong A lines error
                                14D6   862          $DS_PRINTB_S    FORMAT=T_WNG_A_ERR
          FA28 CF   9F          14D6                 PUSHAB  T_WNG_A_ERR
  000100E0 9F   01   FB        14DA                 CALLS   #$$N, @#DS$PRINTB
                                14E1   863
 0B EDD4 DF47   0A   E1        14E1   864 14$:     BBC     #V_NO_GNT,@A_BE0_CR2[R7],16$ ; br = no no-grant error
                                14E8   865          $DS_PRINTB_S    FORMAT=T_NO_GNT_ERR
          FA64 CF   9F          14E8                 PUSHAB  T_NO_GNT_ERR
  000100E0 9F   01   FB        14EC                 CALLS   #$$N, @#DS$PRINTB
                                14F3   866
 0B EDC2 DF47   0B   E1        14F3   867 16$:     BBC     #V_INT_SSYN,@A_BE0_CR2[R7],18$ ; br = no interrupt SSYN error
                                14FA   868          $DS_PRINTB_S    FORMAT=T_INT_SSYN_ERR
          FAA0 CF   9F          14FA                 PUSHAB  T_INT_SSYN_ERR
  000100E0 9F   01   FB        14FE                 CALLS   #$$N, @#DS$PRINTB
                                1505   869
  0B EB44 CF   10   E1        1505   870 18$:     BBC     #V_UBI_NXM_ERR,L_SAVE_CSR,20$ ; br = no UBI NXM error
                                1508   871          $DS_PRINTB_S    FORMAT=T_UBI_NXM_ERR
          FB0F CF   9F          1508                 PUSHAB  T_UBI_NXM_ERR
  000100E0 9F   01   FB        150F                 CALLS   #$$N, @#DS$PRINTB
                                1516   872
  0B EB33 CF   0F   E1        1516   873 20$:     BBC     #V_UBI_PB_ERR,L_SAVE_CSR,24$ ; br = no UBI Map Parity error
                                151C   874          $DS_PRINTB_S    FORMAT=T_UBI_PB_ERR
          FB61 CF   9F          151C                 PUSHAB  T_UBI_PB_ERR
  000100E0 9F   01   FB        1520                 CALLS   #$$N, @#DS$PRINTB
                                1527   875
  0B EB22 CF   0E   E1        1527   876 24$:     BBC     #V_UBI_WR_ERR,L_SAVE_CSR,26$ ; br = no UBI write error
                                152D   877          $DS_PRINTB_S    FORMAT=T_UBI_WR_ERR
          FBAE CF   9F          152D                 PUSHAB  T_UBI_WR_ERR
  000100E0 9F   01   FB        1531                 CALLS   #$$N, @#DS$PRINTB
```

```
                                1538   878
                 0B EB11 CF   1F E1 1538   879 26$:    BBC     #V_UBI_RDS_ERR,L_SAVE_CSR,28$   ; br = no RDS err
                                153E   880          $DS_PRINTB_S    FORMAT=T_UBI_RDS_ERR
                          FAA8 CF   9F 153E                   PUSHAB  T_UBI_RDS_ERR
                     000100E0 9F   01 FB 1542                 CALLS   #$$N, @#DS$PRINTB
                                1549   881
                                05 1549   882 28$:    RSB                                     ; return
                                154A   883
                                154A   884
                                154A   885 ; This error reporting routine is called whenever a data error is
                                154A   886 ; found in any buffer used for the Multitransfer Test.  It prints the number
                                154A   887 ; of words in error, and optionally (as an extended message) a description
                                154A   888 ; of up to eight locations found to be in error.  This routine makes a nested
                                154A   889 ; call to CSR_CHK, a routine within prlink XFER_ERR which reports any errors
                                154A   890 ; that are detected through the UBE (CR2) or the UBI CSR.
                                154A   891
                                154A   892 MXFR_ERR:
                                154A   893
                                154A   894          $DS_BGNMESSAGE
                              0000 154A                   .WORD   ^M<>                    ; ENTRY MASK
                                154C   895
                                154C   896          $DS_PRINTB_S    FORMAT=T_MXFR_TYPE      ; description message
                          F3E9 CF   9F 154C                   PUSHAB  T_MXFR_TYPE
                     000100E0 9F   01 FB 1550                 CALLS   #$$N, @#DS$PRINTB
                                1557   897
                                1557   898          $DS_PRINTB_S    FORMAT=@ERR$_P1(AP)     ; print description
                            14 BC   9F 1557                 PUSHAB  @ERR$_P1(AP)
                     000100E0 9F   01 FB 155A                 CALLS   #$$N, @#DS$PRINTB
                                1561   899
                                1561   900          $DS_PRINTB_S    FORMAT=T_BUFF_ADR,-     ; print buffer start address
                                1561   901                          P1=ERR$_P2(AP)
                              18 AC   DD 1561                   PUSHL   ERR$_P2(AP)
                          F3EB CF   9F 1564                   PUSHAB  T_BUFF_ADR
                     000100E0 9F   02 FB 1568                 CALLS   #$$N, @#DS$PRINTB
                                156F   902
                                156F   903          $DS_PRINTB_S    FORMAT=T_MXFR_D_ERR,-   ; report # of words in error
                                156F   904                          P1=L_ERROR_CT           ; number of errors
                          EBEC CF   DD 156F                   PUSHL   L_ERROR_CT
                          F2EF CF   9F 1573                   PUSHAB  T_MXFR_D_ERR
                     000100E0 9F   02 FB 1577                 CALLS   #$$N, @#DS$PRINTB
                                157E   905          $DS_PRINTX_S    FORMAT=T_MXFR_ERR_HDR   ; print header
                          F765 CF   9F 157E                   PUSHAB  T_MXFR_ERR_HDR
                     000100E8 9F   01 FB 1582                 CALLS   #$$N, @#DS$PRINTX
                                52 D4 1589   906          CLRL    R2                      ; clear index
                  53    EBD3 CF42   D0 158B   907 5$:     MOVL    L_ERROR_LOG[R2],R3      ; load location of error
                           53   63 B0 1591   908          MOVW    (R3),R3                 ; fetch actual data
              5A   53    EAC5 CF42   CD 1594   909          XORL3   L_EXP_DATA[R2],R3,R10   ; load XOR
                                1598   910          $DS_PRINTX_S    FORMAT=T_WORD_ERR,-     ; print error description
                                1598   911                          P1=L_ERROR_LOG[R2],-    ; location
                                1598   912                          P2=L_EXP_DATA[R2],-     ; expected
                                1598   913                          P3=R3,-                 ; actual
                                1598   914                          P4=R10                  ; XOR
                                  5A   DD 1598                   PUSHL   R10
                                  53   DD 159D                   PUSHL   R3
                          EABA CF42   DD 159F                   PUSHL   L_EXP_DATA[R2]
                          EBBA CF42   DD 15A4                   PUSHL   L_ERROR_LOG[R2]
                            F7F3 CF   9F 15A9                   PUSHAB  T_WORD_ERR
```

```
                000100E8 9F   05    FB  15AD                        CALLS   #$$N, @#DS$PRINTX
                         07   52    D1  15B4   915                  CMPL    R2,#7                       ; 8 lines of data printed?
                              06    13  15B7   916                  BEQL    10$                         ; br = 8 lines printed
                 CC 52   EBA2 CF    F2  15B9   917                  AOBLSS  L_ERROR_CT,R2,5$            ; incr index, br = not done
                                    15BF   918
             57   EB8A CF48   02    78  15BF   919 10$:             ASHL    #2,L_INITL_MAP[R8],R7       ; calc Map offset
                57   00F26800 8F    C0  15C6   920                  ADDL2   #A_MAP_BGN,R7               ; add map base
                                    15CD   921 3$:                  $DS_PRINTX_S    FORMAT=T_MAPS_USED,-    ; report map used
                                    15CD   922                                      P1=R7
                              57    DD  15CD                        PUSHL   R7
                         F6CC CF    9F  15CF                        PUSHAB  T_MAPS_USED
                000100E8 9F   02    FB  15D3                        CALLS   #$$N, @#DS$PRINTX
                                    15DA   923
                         FE93 CF    16  15DA   924                  JSB     CSR_CHK                     ; report errors from CSRs
                                    15DE   925
                                    15DE   926                  $DS_ENDMESSAGE
                               - 04 15DE                            RET                         ; RETURN TO DIAGNOSTIC SUPERVISOR
                                    15DF   927
                                    15DF   928
                                    15DF   929 ; This extended error reporting routine is called when any error bit in the
                                    15DF   930 ; UBI CSR either does not set when expected to or does not clear when read.
                                    15DF   931 ; An error message is printed if SSYN was issued when not expected, i.e.,
                                    15DF   932 ; during any attempt to set UBI CSR error bits.
                                    15DF   933
                                    15DF   934 CSR_ERR:
                                    15DF   935
                                    15DF   936                  $DS_BGNMESSAGE
                               0000 15DF                            .WORD   ^M<>                ; ENTRY MASK
                                    15E1   937
                 0E EA1A CF    00    E1  15E1   938                  BBC     #V_NO_CSR,W_FLAGS,5$        ; br = CSR accessible
                                    15E7   939                  $DS_PRINTB_S    FORMAT=T_NO_CSR_ERR     ; report no-access error
                         F29D CF    9F  15E7                        PUSHAB  T_NO_CSR_ERR
                000100E0 9F   01    FB  15EB                        CALLS   #$$N, @#DS$PRINTB
                              0060  31  15F2   940                  BRW     50$                         ; end routine
                                    15F5   941
                 14 EA06 CF    07    E1  15F5   942 5$:             BBC     #V_EXP_WR_ERR,W_FLAGS,10$   ; br = WR ERR not expected
                 0B EA4E CF    0E    E0  15FB   943                  BBS     #V_UBI_WR_ERR,L_SAVE_CSR,7$ ; br = CSR WR ERR bit set
                                    1601   944                  $DS_PRINTB_S    FORMAT=T_NO_WR_ERR      ; print "never set" message
                         FB12 CF    9F  1601                        PUSHAB  T_NO_WR_ERR
                000100E0 9F   01    FB  1605                        CALLS   #$$N, @#DS$PRINTB
                              0031  31  160C   945 7$:             BRW     40$                         ; end routine
                                    160F   946
                 14 E9EC CF    09    E1  160F   947 10$:            BBC     #V_EXP_NXM_ERR,W_FLAGS,15$  ; br = NXM not expected
                 0B EA34 CF    10    E0  1615   948                  BBS     #V_UBI_NXM_ERR,L_SAVE_CSR,13$ ; br = CSR NXM bit set
                                    161B   949                  $DS_PRINTB_S    FORMAT=T_NO_NXM_ERR     ; print "never set" message
                         FA33 CF    9F  161B                        PUSHAB  T_NO_NXM_ERR
                000100E0 9F   01    FB  161F                        CALLS   #$$N, @#DS$PRINTB
                              0017  31  1626   950 13$:            BRW     40$                         ; end routine
                                    1629   951
                 11 E9D2 CF    0A    E1  1629   952 15$:            BBC     #V_EXP_PB_ERR,W_FLAGS,40$   ; br = PB ERR not expected
                 0B EA1A CF    0F    E0  162F   953                  BBS     #V_UBI_PB_ERR,L_SAVE_CSR,40$ ; br = CSR PB ERR bit set
                                    1635   954                  $DS_PRINTB_S    FORMAT=T_NO_PB_ERR      ; print "never set" message
                         FA6E CF    9F  1635                        PUSHAB  T_NO_PB_ERR
                000100E0 9F   01    FB  1639                        CALLS   #$$N, @#DS$PRINTB
                                    1640   955
             0B EC72 DF   00000008'EF    E0  1640   956 40$:        BBS     V_NO_SSYN,@A_BEO_CR2,50$    ; br = no SSYN issued
                                    164A   957                  $DS_PRINTB_S    FORMAT=T_UNX_SSYN_ERR   ; print unxp SSYN message
```

```
                F796 CF    9F   164A                        PUSHAB  T_UNX_SSYN_ERR
          000100E0 9F      01   FB  164E                    CALLS   #$$N, @#DS$PRINTB
                                1655   958
     00F26010 9F    8001C000 8F   D3  1655   959 50$:      BITL    #M_UBI_STS,@#A_UBI_CSR          ; did CSR clear when read?
                             0B   13  1660   960            BEQL    60$                             ; br = CSR did clear
                                  1662   961            $DS_PRINTB_S    FORMAT=T_CSR_NCL_ERR    ; print "didnt clear" message
                FAE7 CF    9F   1662                        PUSHAB  T_CSR_NCL_ERR
          000100E0 9F      01   FB  1666                    CALLS   #$$N, @#DS$PRINTB
                                166D   962 60$:
                                166D   963            $DS_ENDMESSAGE
                             04   166D                        RET                             ; RETURN TO DIAGNOSTIC SUPERVISOR
                                166E   964
                                166E   965
                                166E   966 ; This extended error-reporting routine is called to report any errors
                                166E   967 ; pertaining to interrupts, i.e., when an unexpected interrupt occurs or
                                166E   968 ; an expected interrupt occurs with wrong vector or at wrong IPL.
                                166E   969
                                166E   970 INTR_ERR:
                                166E   971
                                166E   972            $DS_BGNMESSAGE
                           0000 166E                        .WORD   ^M<>                    ; ENTRY MASK
                                1670   973
           16 E98B CF    01   E1  1670   974            BBC             #V_EXP_BR_INTR,W_FLAGS,3$ ; br = no interrupt expected
                                1676   975            $DS_PRINTB_S    FORMAT=T_EXP_INTR,-     ; print expected message
                                1676   976                            P1=L_BR_LVL,-
                                1676   977                            P2=L_IPL
                EA2D CF    DD   1676                        PUSHL   L_IPL
                EA2D CF    DD   167A                        PUSHL   L_BR_LVL
                FAF2 CF    9F   167E                        PUSHAB  T_EXP_INTR
          000100E0 9F      03   FB  1682                    CALLS   #$$N, @#DS$PRINTB
                   000F    31   1689   978            BRW             4$                              ; br around unxp intr msg
                                168C   979
                                168C   980 3$:       $DS_PRINTB_S    FORMAT=T_UNXP_INTR,-    ; print unxp intr msg
                                168C   981                            P1=L_IPL
                EA17 CF    DD   168C                        PUSHL   L_IPL
                FB20 CF    9F   1690                        PUSHAB  T_UNXP_INTR
          000100E0 9F      02   FB  1694                    CALLS   #$$N, @#DS$PRINTB
                                169B   982
            0E E960 CF    03   E0  169B   983 4$:       BBS             #V_BR_INTR,W_FLAGS,5$   ; br = any interrupt occurred
                                16A1   984
                                16A1   985            $DS_PRINTB_S    FORMAT=T_NO_INTR        ; print no-interrupt message
                FB3A CF    9F   16A1                        PUSHAB  T_NO_INTR
          000100E0 9F      01   FB  16A5                    CALLS   #$$N, @#DS$PRINTB
                   001C    31   16AC   986            BRW             10$                             ; br to end of report
                                16AF   987
    E9FA CF    03   00000000'8F   EF  16AF   988 5$:       EXTZV           #CHI$V_IPL,#3,W_UBI_STS_2,R2 ; extract actual BR level
                             52   16B9
                                16BA   989            $DS_PRINTB_S    FORMAT=T_ACTL_INTR,-    ; print actual intrpt message
                                16BA   990                            P1=R2,-
                                16BA   991                            P2=W_VECTOR
                E9F7 CF    DD   16BA                        PUSHL   W_VECTOR
                             52   DD  16BE                    PUSHL   R2
                FB40 CF    9F   16C0                        PUSHAB  T_ACTL_INTR
          000100E0 9F      03   FB  16C4                    CALLS   #$$N, @#DS$PRINTB
                                16CB   992 10$:
                                16CB   993            $DS_ENDMESSAGE
                             04   16CB                        RET                             ; RETURN TO DIAGNOSTIC SUPERVISOR
```

```
                                16CC   994
                                16CC   995
                                16CC   996 ; This extended error reporting routine is used to report any error encountered
                                16CC   997 ; during execution of Channel Services functions.
                                16CC   998
                                16CC   999 CHANNEL_ERR:
                                16CC  1000
                                16CC  1001          $DS_BGNMESSAGE
                          0000  16CC                        .WORD   ^M<>                    ; ENTRY MASK
                                16CE  1002
00660060 8F    E9DD CF    D1   16CE  1003          CMPL    L_UBI_STATUS,#DS$_IHWE          ; initial hware error?
                        0E    12   16D7  1004          BNEQ    3$                              ; br = not initial hware error
                                16D9  1005          $DS_PRINTB_S    FORMAT=T_IHWE           ; print error description
                   FC5F CF    9F   16D9                       PUSHAB  T_IHWE
        000100E0 9F    01    FB   16DD                       CALLS   #3$N, @#DS$PRINTB
                     0033    31   16E4  1006          BRW     7$                              ; br to return
                                16E7  1007 3$:
00660070 8F    E9C4 CF    D1   16E7  1008          CMPL    L_UBI_STATUS,#DS$_LOGIC         ; logic error?
                        0E    12   16F0  1009          BNEQ    5$                              ; br = not logic error
                                16F2  1010          $DS_PRINTB_S    FORMAT=T_LOGIC          ; print error description
                   FC6A CF    9F   16F2                       PUSHAB  T_LOGIC
        000100E0 9F    01    FB   16F6                       CALLS   #3$N, @#DS$PRINTB
                     001A    31   16FD  1011          BRW     7$                              ; br to return
                                1700  1012 5$:
00660038 8F    E9AB CF    D1   1700  1013          CMPL    L_UBI_STATUS,#DS$_IVVECT        ; invalid vector error?
                        0F    12   1709  1014          BNEQ    7$                              ; br = not invalid vector error
                                170B  1015          $DS_PRINTB_S    FORMAT=T_IVVECT,-       ; print error description
                                170B  1016                          P1=W_VECTOR             ; vector received
                   E9A6 CF    DD   170B                       PUSHL   W_VECTOR
                   FC80 CF    9F   170F                       PUSHAB  T_IVVECT
        000100E0 9F    02    FB   1713                       CALLS   #3$N, @#DS$PRINTB
                                171A  1017 7$:
                                171A  1018          $DS_ENDMESSAGE
                          04    171A                        RET                             ; RETURN TO DIAGNOSTIC SUPERVISOR
                                171B  1019
                                171B  1020 ; This extended error-reporting routine is called when access to a selected
                                171B  1021 ; UBE fails during the first multitransfer subtest to report the address
                                171B  1022 ; of the attempted access.
                                171B  1023
                                171B  1024 NO_ACCESS:
                                171B  1025
                                171B  1026          $DS_BGNMESSAGE
                          0000  171B                        .WORD   ^M<>                    ; ENTRY MASK
                                171D  1027
                                171D  1028          $DS_PRINTB_S    FORMAT=T_UBE_ADR,-      ; report address
                                171D  1029                          P1=ERR$_P1(AP),-
                                171D  1030                          P2=ERR$_P1(AP)
                      14 AC    DD   171D                       PUSHL   ERR$_P1(AP)
                      14 AC    DD   1720                       PUSHL   ERR$_P1(AP)
                   ECBE CF    9F   1723                       PUSHAB  T_UBE_ADR
        000100E0 9F    03    FB   1727                       CALLS   #3$N, @#DS$PRINTB
                                172E  1031
                                172E  1032          $DS_ENDMESSAGE
                          04    172E                        RET                             ; RETURN TO DIAGNOSTIC SUPERVISOR
                                172F  1033
```

```
                                 172F  1035 .SBTTL  SUBROUTINES
                                 172F  1036
                                 172F  1037 ; This subroutine is called to generate the Reed-Muller encoded data which
                                 172F  1038 ; will be used in the Map Address Bus subtest.  The data table will be created
                                 172F  1039 ; by multiplying CODEVECTORS[I] by bit I of the number to be encoded and sum-
                                 172F  1040 ; ming (binarily) the products.  The words thus created will be expanded into
                                 172F  1041 ; longwords by inserting zeroes in the non-writable bit positions of the Map
                                 172F  1042 ; Entries.  The numbers 0 - 7 will be encoded.  A more complete explanation of
                                 172F  1043 ; this encoding scheme appears in the Map Address Bus subtest description.
                                 172F  1044
                                 172F  1045 RD_MLLR_ENCODE:
                                 172F  1046
                      54   D4    172F  1047          CLRL    R4                          ; clear input number
                      52   D4    1731  1048 1$:      CLRL    R2                          ; clear temp location
                      51   D4    1733  1049          CLRL    R1                          ; clear vector index (I)
          53   54   01   51   EF  1735  1050 2$:      EXTZV   R1,#1,R4,R3                 ; extract bit I from input
             53   EB58 CF41   A4  173A  1051          MULW2   CODEVECTORS[R1],R3          ; multiply bit by vector
               52   52   53   CD  1740  1052          XORL3   R3,R2,R2                    ; binary sum
                 ED 51   03   F3  1744  1053          AOBLEQ  #3,R1,2$                    ; br = not last vector
                   07 52   0F   E5  1748  1054          BBCC    #15,R2,3$                   ; br = bit 15 clear and clear
           52   02000000 8F   C8  174C  1055          BISL2   #M_BYTE_OFF,R2              ; set bit 25 (byte-offset bit)
             EADE CF44   52   D0  1753  1056 3$:      MOVL    R2,RD_MLLR_TBL[R4]          ; load into table
                 D4 54   07   F3  1759  1057          AOBLEQ  #7,R4,1$                    ; do until 0-7 are encoded
                             05  175D  1058          RSB                                 ; return to mainline code
                                 175E  1059
                                 175E  1060
                                 175E  1061 ; This subroutine is called to decode the data retrieved from the Map Entries
                                 175E  1062 ; during the Map Address Bus subtest and loads it into DECODE_TBL.  A brief
                                 175E  1063 ; explanation of this decoding scheme appears in the subtest description.
                                 175E  1064
                                 175E  1065 RD_MLLR_DECODE:
                                 175E  1066
                                 175E  1067 ; Decode 3 MSBs  of the information nibble first.
                                 175E  1068
                      52   D4    175E  1069          CLRL    R2                          ; clear index
                   EB12 CF42   D4  1760  1070 0$:      CLRL    DECODE_TBL[R2]              ; clear decoded word
          EA36 CF    EAED CF42   D0  1765  1071          MOVL    RETRIEVE_TBL[R2],L_RETRIEVE ; get retrieved word
               09 EA31 CF    19   E5  176D  1072          BBCC    #V_BYTE_OFF,L_RETRIEVE,1$ ; br = byte-offset bit clr, clr
        EA27 CF    00008000 8F   C8  1773  1073          BISL2   #<1@15>,L_RETRIEVE          ; set bit 15 of retrieved word
                EA0E CF    01   D0  177C  1074 1$:      MOVL    #1,L_PAIR_SEP               ; pair separation = 1
                EA05 CF    01   D0  1781  1075          MOVL    #1,L_INFO_BIT               ; info bit position = 1
                     EA15 CF    D4  1786  1076 2$:      CLRL    L_POP                       ; clear ones population
                01   E9FD CF    D1  178A  1077          CMPL    L_INFO_BIT,#1               ; first iteration ?
                             08   13  178F  1078          BEQL    3$                          ; br = first iteration
E9F6 CF    E9F9 CF    01   78  1791  1079          ASHL    #1,L_PAIR_SEP,L_PAIR_SEP    ; double separation
           E9F5 CF    0F   D0  1799  1080 3$:      MOVL    #15,L_POSITION              ; load bit position
E9FD CF    01   E9F1 CF    EF  179E  1081 4$:      EXTZV   L_POSITION,#1,L_RETRIEVE,L_TEMP_A ; first bit of relation
                     E9EE CF        17A6
     E9E3 CF    E9E2 CF    C2  17A9  1082          SUBL2   L_PAIR_SEP,L_POSITION       ; point to other bit of relation
E9EB CF    01   E9DF CF    EF  17B0  1083          EXTZV   L_POSITION,#1,L_RETRIEVE,L_TEMP_B ; second bit of relation
                     E9E0 CF        17B8
     E9D5 CF    E9DC CF    CC  17B8  1084          XORL2   L_TEMP_B,L_TEMP_A           ; XOR redundant relation bits
     E9D6 CF    E9D1 CF    C0  17C2  1085          ADDL2   L_TEMP_A,L_POP              ; incr ones population if XOR = 1
     59   01    E9BE CF    78  17C9  1086          ASHL    L_INFO_BIT,#1,R9            ; 2**L_INFO_BIT
7E   00   E9BF CF    01   7A  17CF  1087          EMUL    #1,L_POSITION,#0,-(SP)      ; extend position to quadword
     5A   5A   8E   59   7B  17D6  1088          EDIV    R9,(SP)+,R10,R10            ; R10 = L_POSITION mod (2**L_INFO_BIT)
                          5A   D5  17DB  1089          TSTL    R10                         ; R10 clear?
```

```
                     07   13  17DD  1090          BEQL    10$                             ; br = R10 clear
          E9AD CF    E9AC CF   C0  17DF  1091          ADDL2   L_PAIR_SEP,L_POSITION       ; restore position
               B3 E9A9 CF      F4  17E6  1092 10$:      SOBGEQ  L_POSITION,4$               ; decr position, br >= 0
            04    E9B0 CF      D1  17EB  1093          CMPL    L_POP,#4                    ; is decoded bit 0?
                     06   19  17F0  1094          BLSS    1T$                             ; br = decoded bit 0?
             EA7F CF42    59   C8  17F2  1095          BISL2   R9,DECODE_TBL[R2]           ; set info bit in decoded word
          88 E98E CF      03   F3  17F8  1096 11$:     AOBLEQ  #3,L_INFO_BIT,2$            ; incr info bit, br = not done
                                   17FE  1097
                                   17FE  1098 ; Decode LSB, using the previously-decoded 3 MSBs of the information nibble.
                                   17FE  1099
                        53   01   D0  17FE  1100 12$:     MOVL    #1,R3                       ; info bit pointer = 1
          E992 CF    EA71 CF42    D0  1801  1101          MOVL    DECODE_TBL[R2],L_TEMP_B     ; load partially-decoded word
     E985 CF    E98C CF    01   53   EF  1809  1102 13$:     EXTZV   R3,#1,L_TEMP_B,L_TEMP_A     ; extr info bit from decoded word
          E97D CF    EA80 CF43    A4  1812  1103          MULW2   CODEVECTORS[R3],L_TEMP_A    ; multiply info bit by codevector
            E982 CF    E979 CF    AC  181A  1104          XORW2   L_TEMP_A,L_RETRIEVE         ; XOR result with retrieved word
                  E4 53    03   F3  1821  1105          AOBLEQ  #3,R3,T3$                   ; incr info bit pointer, br = not done
                     E976 CF      D4  1825  1106          CLRL    L_POP                       ; clear ones population
                        53        D4  1829  1107          CLRL    R3                          ; clear bit pointer
                  04 E973 CF      53  E1  182B  1108 14$:     BBC     R3,L_RETRIEVE,15$           ; br = XOR bit clear
                     E96A CF      D6  1831  1109          INCL    L_POP                       ; incr ones population
                        F2 53     0F  F3  1835  1110 15$:     AOBLEQ  #T5,R3,14$              ; incr pointer, br = not done
                     08   E962 CF      D1  1839  1111          CMPL    L_POP,#8            ; determine info bit 0
                              05   19  183E  1112          BLSS    16$                 ; br = info bit 0 will be 0
                        EA32 CF42      D6  1840  1113          INCL    DECODE_TBL[R2]      ; set bit 0
                              52       D6  1845  1114 16$:     INCL    R2                  ; incr index
                           07   52      D1  1847  1115          CMPL    R2,#7               ; done?
                                03   14  184A  1116          BGTR    17$                 ; br = done
                              FF11   31  184C  1117          BRW     0$                  ; br = beginning of loop
                                     184F  1118 17$:
                                05   184F  1119          RSB
                                     1850  1120
                                     1850  1121
                                     1850  1122 ; This subroutine is used to set up a UBE operation.  The five UBE registers
                                     1850  1123 ; of the specified UBE are loaded with the command specified.  This routine
                                     1850  1124 ; contains a rather unconventional use of index mode in which an index is
                                     1850  1125 ; used to add the appropriate offset in order to load into the desired device.
                                     1850  1126 ; This index is not incremented during this subroutine.  The following
                                     1850  1127 ; parameters are expected to be loaded before calling this routine:
                                     1850  1128 ;
                                     1850  1129 ;       R8      UBE device number (0-3)
                                     1850  1130 ;       R9      address of command table for desired function
                                     1850  1131
                                     1850  1132
                                     1850  1133 UBE_SET_UP:
                                     1850  1134
                     5A    EA57 CF    DE  1850  1135          MOVAL   A_UBE_ADR_TBL,R10       ; load adr of UBE Reg adr table
                     51    58   03    78  1855  1136          ASHL    #3,R8,R1                ; calc word index from device number
                        9A41    89    B0  1859  1137 1$:      MOVW    (R9)+,@(R10)+[R1]       ; load UBE register
               000002BF'8F    5A    D1  185D  1138          CMPL    R10,#A_BE0_CR1          ; all registers loaded ?
                              F3    15  1864  1139          BLEQ    1$                      ; br = not all registers loaded
                                    05  1866  1140          RSB                             ; return to mainline code
                                        1867  1141
                                        1867  1142
                                        1867  1143 ; This routine sizes Unibus Memory space from Unibus address 0 through
                                        1867  1144 ; 777000 (octal) in increments of 512 byte addresses.  A bitvector is gener-
                                        1867  1145 ; ated (V_VALID_MAPS) with a bit set for each valid map entry (those for which
                                        1867  1146 ; Unibus memory has not been implemented).  The bitvector is initially all
```

```
                                     1867  1147 ; ones and bits are cleared in this routine each time a Unibus address is suc-
                                     1867  1148 ; cessfully accessed.  The Valid bit is set in each Map Entry which is found to
                                     1867  1149 ; be usable and cleared otherwise.  The Number of the first valid Map Entry is
                                     1867  1150 ; returned in the location FRST_VAL_MAP.  If there are no valid maps this loca-
                                     1867  1151 ; tion will retain its initial value of -1.
                                     1867  1152
                                     1867  1153 UNIBUS_SIZER:
                                     1867  1154
                                     1867  1155
                                     1867  1156
                                     1867  1157 ;
                                     1867  1158 ; Clear vector bits representing invalid Map Entries due to the presence of
                                     1867  1159 ; any Unibus devices.
                                     1867  1160 ;
                                     1867  1161
                                     1867  1162         $DS_SETVEC_S    #SCB$L_MACHCHK,200$     ;set machine check handler
                          00   DD    1867                        PUSHL   #0
                 000018E8'EF   DF    1869                        PUSHAL  200$
                          04   DD    186F                        PUSHL   #SCB$L_MACHCHK
               00010160 9F    03   FB 1871                        CALLS   #3, @#DS$SETVEC
                          52   D4    1878  1163         CLRL    R2                      ; initialize counter
                          54   7C    187A  1164         CLRQ    R4                      ;  clear indices
               53   00FC0000 8F    D0    187C  1165         MOVL    #A_UB_IO_SP,R3          ; load adress of I/O space into R3
                    50   6345    B0    1883  1166 1$:      MOVW    (R3)[R5],R0             ; attempt access of memory
                                     1887  1167
          E778 CF   01   54   00    F0    1887  1168         INSV    #0,R4,#1,V_VALID_MAPS   ; clear vector bit if device response
                          0F   11    188E  1169         BRB     6$                      ; br out of loop
          E76F CF   01   54   01    F0    1890  1170 5$:      INSV    #1,R4,#1,V_VALID_MAPS   ; set nector bit if no device respone
             E4 55   0000007F 8F    F3    1897  1171         AOBLEQ  #127,R5,1$              ; incr index, br = not end of page
                          55   D4    189F  1172 6$:      CLRL    R5                      ; clear index
                    09 52     32    F3    18A1  1173         AOBLEQ  #50,R2,30$              ; limit calls $DS_BREAK
                          52   D4    18A5  1174         CLRL    R2                      ; reset counter
                               18A7  1175         $DS_BREAK                       ; check for ^C
                00010058 9F    6E   FA 18A7                        CALLG   (SP), @#DS$BREAK
                53   00000200 8F    C0    18AE  1176 30$:     ADDL2   #512,R3                 ; adjust R3 to next page boundary
             C6 54   00000200 8F    F2    18B5  1177         AOBLSS  #512,R4,1$              ; incr page no and br = not done
                               18BD  1178 ;
                               18BD  1179 ; Find first valid Map Entry.
                               18BD  1180 ;
          E73C CF   FFFFFFFF 8F    D0    18BD  1181         MOVL    #-1,FRST_VAL_MAP        ; load no-valid-map value
                          54   D4    18C6  1182         CLRL    R4                      ; clear position
                07 E739 CF   54   E1    18C8  1183 7$:      BBC     R4,V_VALID_MAPS,8$      ; br = valid bit found
                   E72F CF   54   D0    18CE  1184         MOVL    R4,FRST_VAL_MAP         ; load number of first valid Map Entry
                          08   11    18D3  1185         BRB     9$                      ; br out of loop
             EB 54   00000200 8F    F2    18D5  1186 8$:      AOBLSS  #512,R4,7$              ; incr pos, br = not done
                               18DD  1187 9$:
                               18DD  1188
                               18DD  1189         $DS_CLRVEC_S    #SCB$L_MACHCHK  ; restore machine check handler
                          04   DD    18DD                        PUSHL   #SCB$L_MACHCHK
                00010168 9F    01   FB 18DF                        CALLS   #1, @#DS$CLRVEC
                               05    18E6  1190         RSB                             ; return to mainline code
                               18E7  1191
                               18E7  1192 ; This routine services machine check
                               18E7  1193
                               18E7  1194         .ALIGN  LONG                    ; machine check handler
                               18E8  1195
                               18E8  1196 200$:
```

```
   00000000'8F    FFFFFFFF 8F   DA  18E8            MTPR    #-1,#PR$_MCESR           ; clear any error
                        5E  8E   C0  18F3            ADDL    (SP)+,SP                 ; reset SP
                   6E   97 AF   9E  18F6            MOVAB   5$,(SP)                  ; set return address
                                02  18FA            REI                              ; return
                                    18FB
                                    18FB
                                    18FB  ; This subroutine is used to invalidate all Map Entries.
                                    18FB
                                    18FB  INVAL_MAPS:
                                    18FB
                             54  D4  18FB            CLRL    R4                       ; clear index
                 00F26800 9F44  D4  18FD  1$:       CLRL    @#A_MAP_BGN[R4]          ; clear Map Entry
            F1 54    00000200 8F F2  1904            AOBLSS  #512,R4,1$               ; incr index, br = not done
                                05  190C            RSB                              ; return to mainline code
                                    190D
                                    190D
                                    190D  ; This subroutine sets the valid bit in each usable Map Entry.
                                    190D
                                    190D  VAL_MAPS:
                                    190D
                             54  D4  190D            CLRL    R4                       ; clear index
              0C E6F2 CF     54  E1  190F  1$:       BBC     R4,V_VALID_MAPS,3$       ; br = Map Invalid
  00F26800 9F44  80000000 8F    C8  1915            BISL2   #M_MAP_VAL,@#A_MAP_BGN[R4] ; validate Map Entry
            E6 54    00000200 8F F2  1921  3$:       AOBLSS  #512,R4,1$               ; incr index, br = not done
                                05  1929            RSB                              ; return to mainline code
                                    192A
                                    192A
                                    192A  ; This subroutine sets up all valid Map Entries to point to the page number
                                    192A  ; passed in the location L_PAGE_NO.
                                    192A
                                    192A  ALL_MAPS:
                                    192A
   59    E717 CF    80000000 8F    C9  192A            BISL3   #M_MAP_VAL,L_PAGE_NO,R9 ; load data to put in Map Entries
                             54  D4  1934            CLRL    R4                       ; clear index
              08 E6CB CF     54  E1  1936  1$:       BBC     R4,V_VALID_MAPS,3$       ; br = Map Invalid
           00F26800 9F44     59  D0  193C            MOVL    R9,@#A_MAP_BGN[R4]       ; load Map Entry
            EA 54    00000200 8F F2  1944  3$:       AOBLSS  #512,R4,1$               ; incr index, br = not done
                                05  194C            RSB                              ; return to mainline code
                                    194D
                                    194D
                                    194D  ; This subroutine is called during the Multi-transfer test to set up a
                                    194D  ; sequence of contiguous maps as described by the following parameters:
                                    194D  ;
                                    194D  ;         L_BUFFER_ADR     starting address of buffer
                                    194D  ;         L_MAPS_NEEDED    number of contiguous maps needed
                                    194D  ;         L_MAP_NUM        next map number to try
                                    194D  ;         W_FLAGS          bit (M_BO) set to indicate byte-offset mode
                                    194D  ;
                                    194D  ;         R0               lsb set if successful
                                    194D
                                    194D  SET_UP_MAPS:
                                    194D
                             50  D4  194D            CLRL    R0                               ; clear success code
                             52  D4  194F            CLRL    R2                               ; clear index
                  51    E735 CF  D0  1951            MOVL    L_MAP_NUM,R1                     ; load next Map Number
  E6EB CF    E76B CF    0F   09  EF  1956            EXTZV   #9,#15,L_BUFFER_ADR,L_PAGE_NO    ; extract page number
                                    195F  ;
```

```
                                    195F  1254 ; Find next block of usable maps, set up Map Entries
                                    195F  1255 ;
      E69F CF    E73F CF    51    EC  195F  1256 1$:     CMPV    R1,L_MAPS_NEEDED,V_VALID_MAPS,#-1 ; block usable?
                   FFFFFFFF 8F        1967
                            0B    13  196C  1257         BEQL    3$                                ; br = not usable
          E9 51     00000200 8F    F2  196E  1258         AOBLSS  #512,R1,1$                        ; incr index br = not done
                             003B  31  1976  1259         BRW     10$                               ; br = maps used up
                               50  D6  1979  1260 3$:     INCL    R0                                ; set success code
                E7CE CF48      51  D0  197B  1261         MOVL    R1,L_INITL_MAP[R8]                ; save initial map number
        00F26800 9F41    E6C5 CF  D0  1981  1262 4$:     MOVL    L_PAGE_NO,@#A_MAP_BGN[R1]         ; load PFN into Map Entry
                         E6BB CF  B6  1988  1263         INCW    L_PAGE_NO                         ; increment page number
      00F26800 9F41      80000000 8F  C8  198F  1264         BISL2   #M_MAP_VAL,@#A_MAP_BGN[R1]        ; set valid bit
      00F26800 9F41      02000000 8F  C8  199B  1265         BISL2   #M_BYTE_OFF,@#A_MAP_BGN[R1]       ; set byte-offset bit
                               51  D6  19A7  1266 5$:     INCL    R1                                ; incr index
                   D2 52   E6F6 CF  F2  19A9  1267         AOBLSS  L_MAPS_NEEDED,R2,4$               ; incr count, br = not done
                      E6D6 CF    51  D0  19AF  1268         MOVL    R1,L_MAP_NUM                      ; update map number
                                     19B4  1269 10$:
                                  05 19B4  1270         RSB                                       ; return to mainline code
                                     19B5  1271
                                     19B5  1272
                                     19B5  1273 ; This subroutine is called during the Multi-transfer test to load up a
                                     19B5  1274 ; buffer as described by the following parameters:
                                     19B5  1275 ;
                                     19B5  1276 ;       L_BUFFER_ADR    starting address of buffer
                                     19B5  1277 ;       L_BUFFER_SIZE   size of buffer (in words)
                                     19B5  1278 ;       L_PTRNS[R8]     pattern to be loaded in each location
                                     19B5  1279
                                     19B5  1280 LOAD_BUFFER:
                                     19B5  1281
                                  52 D4  19B5  1282         CLRL    R2                                ; clear index
                    53   E6D6 CF48  D0  19B7  1283         MOVL    L_PTRNS[R8],R3                    ; load pattern
                     E704 DF42   53  B0  19BD  1284 1$:     MOVW    R3,@L_BUFFER_ADR[R2]              ; load pattern into buffer
                   F4 52   E744 CF   F3  19C3  1285         AOBLEQ  L_BUFFER_SIZE,R2,1$               ; incr index, br = not done
                                  05 19C9  1286         RSB
                                     19CA  1287
                                     19CA  1288
                                     19CA  1289 ; This subroutine compares the contents of the buffer as described below to
                                     19CA  1290 ; the expected data pattern.  LSB of R0 is set if no errors are found.  The
                                     19CA  1291 ; first ten errors found are logged in the error buffer.
                                     19CA  1292 ;
                                     19CA  1293 ;       L_EXP_DATA      expected pattern
                                     19CA  1294 ;       L_BUFFER_ADR    starting address of buffer
                                     19CA  1295 ;       L_BUFFER_SIZE   size (in words) of buffer
                                     19CA  1296
                                     19CA  1297 CHECK_BUFFER:
                                     19CA  1298
                                  50 7C  19CA  1299         CLRQ    R0                                ; clear success code, index
                                  54 D4  19CC  1300         CLRL    R4                                ; clear index
                  E6F1 DF44    E68C CF  B1  19CE  1301 1$:     CMPW    L_EXP_DATA,@L_BUFFER_ADR[R4]      ; check data
                                  18 13  19D6  1302         BEQL    5$                                ; br = ok
                               0A 51  D1  19D8  1303         CMPL    R1,#10                            ; 10 errors logged?
                                  11 18  19DB  1304         BGEQ    3$                                ; br = 10 errors logged
                E77D CF41    E6E5 DF44  3E  19DD  1305         MOVAW   @L_BUFFER_ADR[R4],L_ERROR_LOG[R1] ; save location of error
                  E670 CF41    E674 CF  D0  19E6  1306         MOVL    L_EXP_DATA,L_EXP_DATA[R1]         ; save expected data
                                  51 D6  19EE  1307 3$:     INCL    R1                                ; incr error count
                       D8 54   E717 CF  F2  19F0  1308 5$:     AOBLSS  L_BUFFER_SIZE,R4,1$               ; incr index, br = not done
                          E764 CF   51  D0  19F6  1309         MOVL    R1,L_ERROR_CT                     ; save error count
```

```
                    02   12  19FB  1310          BNEQ    7$                                  ; br = any errors
                    50   D6  19FD  1311          INCL    R0                                  ; set success code
                         05  19FF  1312 7$:      RSB                                         ; return to mainline code
                             1A00  1313
                             1A00  1314
                             1A00  1315 ; This subroutine compares the contents of the buffer as described below to
                             1A00  1316 ; the data expected for Multitransfer 1, UBE 0 (data from Cycle Count register).
                             1A00  1317 ; The first eight errors found are logged in the error buffers.
                             1A00  1318 ;
                             1A00  1319 ;       L_BUFFER_ADR    starting address of buffer
                             1A00  1320 ;       L_BUFFER_SIZE   size (in words) of buffer
                             1A00  1321
                             1A00  1322 CHECK_BUF_0:
                             1A00  1323
                    50   7C  1A00  1324          CLRQ    R0                                  ; clear success code, index
                    54   D4  1A02  1325          CLRL    R4                                  ; clear index
              53    F830 8F  B0  1A04  1326          MOVW    #-2000,R3                           ; load expected data
         E6B8 DF44    53   B1  1A09  1327 1$:      CMPW    R3,@L_BUFFER_ADR[R4]                ; check data
                    16   13  1A0F  1328          BEQL    5$                                  ; br = ok
                 08 51   D1  1A11  1329          CMPL    R1,#8                               ; 8 errors logged?
                    0F   18  1A14  1330          BGEQ    3$                                  ; br = 8 errors logged
    E744 CF41    E6AC DF44   3E  1A16  1331          MOVAW   @L_BUFFER_ADR[R4],L_ERROR_LOG[R1] ; save location of error
         E639 CF41    53   D0  1A1F  1332          MOVL    R3,L_EXP_DATA[R1]                   ; save expected data
                    51   D6  1A25  1333 3$:      INCL    R1                                  ; incr error count
                 53 02   C0  1A27  1334 5$:      ADDL2   #2,R3                               ; incr expected data
          D9 54    E6DD CF   F2  1A2A  1335          AOBLSS  L_BUFFER_SIZE,R4,1$                 ; incr index, br = not done
             E72A CF    51   D0  1A30  1336          MOVL    R1,L_ERROR_CT                       ; save error count
                    02   12  1A35  1337          BNEQ    7$                                  ; br = any errors
                    50   D6  1A37  1338          INCL    R0                                  ; set success code
                         05  1A39  1339 7$:      RSB                                         ; return to mainline code
                             1A3A  1340
                             1A3A  1341
                             1A3A  1342 ;
                             1A3A  1343 ; This routine services interrupts generated by UBE 0 in the Interrupt subtest
                             1A3A  1344 ; only.
                             1A3A  1345 ;
                             1A3A  1346          .ALIGN  LONG
                             1A3C  1347
                             1A3C  1348 BR_SERVICE:
                             1A3C  1349
                             1A3C  1350          $DS_CANWAIT_S                               ; cancel wait
 00010070 9F    00   FB  1A3C                       CALLS   #0, @#DS$CANWAIT
             E5B8 CF    08   A8  1A43  1351          BISW2   #M_BR_INTR,W_FLAGS                  ; set interrupt flag
          E666 CF    0148 8F   B1  1A48  1352          CMPW    #^X148,W_VECTOR                     ; check vector
                    05   13  1A4F  1353          BEQL    3$                                  ; br = correct vector
             E5AA CF    10   A8  1A51  1354          BISW2   #M_INTR_ERR,W_FLAGS                 ; set interrupt error flag
E653 CF    03    00000000'8F   ED  1A56  1355 3$:      CMPZV   #CHI$V_IPL,#3,W_UBI_STS_2,L_BR_LVL ; check BR level
                  E648 CF        1A60
                    05   13  1A63  1356          BEQL    5$                                  ; br = correct level
             E596 CF    10   A8  1A65  1357          BISW2   #M_INTR_ERR,W_FLAGS                 ; set interrupt error flag
                         02  1A6A  1358 5$:
                             1A6A  1359          REI                                         ; return
                             1A6B  1360
                             1A6B  1361
                             1A6B  1362 ;
                             1A6B  1363 ; This interrupt service routine services any BR interrupts generated during
                             1A6B  1364 ; Functional Test except those generated in the Interrupt subtest.
```

```
                                1A6B  1365 ;
                                1A6B  1366          .ALIGN  LONG
                                1A6C  1367
                                1A6C  1368 INTR_SERVICE:
                                1A6C  1369
                                1A6C  1370          $DS_CANWAIT_S                           ; cancel wait
00010070 9F   00    FB          1A6C                        CALLS   #0, @#DS$CANWAIT
                                02  1A73  1371          REI                                 ; return
                                1A74  1372
                                1A74  1373
                                1A74  1374 ; This interrupt-service routine is entered during the Multi-transfer test
                                1A74  1375 ; when UBE 0 has stopped transferring data.
                                1A74  1376
                                1A74  1377          .ALIGN  LONG
                                1A74  1378
                                1A74  1379 BE0_SERVICE:
                                1A74  1380
                    E62A CF     96  1A74  1381          INCB    B_DONE_CT                   ; incr done count
                                02  1A78  1382          REI                                 ; return to mainline code
                                1A79  1383
                                1A79  1384
                                1A79  1385 ; This interrupt-service routine is entered during the Multi-transfer test
                                1A79  1386 ; when UBE 3 has stopped transferring data.  This routine sets up UBE 3
                                1A79  1387 ; to execute another data transfer until it has transferred 3000 words.
                                1A79  1388
                                1A79  1389          .ALIGN  LONG
                                1A7C  1390
                                1A7C  1391 BE3_SERVICE:
                                1A7C  1392
                    E637 CF     D6  1A7C  1393          INCL    L_XFR_COUNT                 ; increment count
          00000BB8 8F   E633 CF     D1  1A80  1394          CMPL    L_XFR_COUNT,#3000           ; 3000 transfers?
                              11    13  1A89  1395          BEQL    5$                          ; br = 3000 transfers
              E831 DF   FFFF 8F     B0  1A8B  1396          MOVW    #-1,@A_BE3_CC               ; load cycle count
              E82E DF   04D1 8F     B0  1A92  1397          MOVW    #<M_RDY!M_BR7!M_DATI!M_ONE_XFR!M_INT_ODNE!M_GO>,@A_BE3_CR1 ; CR1
                           0004     31  1A99  1398          BRW     10$                         ; br to return
                    E602 CF     96  1A9C  1399 5$:      INCB    B_DONE_CT                   ; incr done count
                                02  1AA0  1400 10$:     REI                                 ; return to mainline code
                                1AA1  1401
                                1AA1  1402
                                1AA1  1403 ; This interrupt-service routine is used to set TIMEOUT flag in the event of
                                1AA1  1404 ; of an interval timer timeout.
                                1AA1  1405
                                1AA1  1406          .ALIGN  LONG
                                1AA4  1407
                                1AA4  1408 TIMER_SERVICE:
                                1AA4  1409
00000000'8F     80000080 8F     DA  1AA4  1410          MTPR    #<M_CLK_ERR+M_CLK_INTR>,#PR$_ICCS ; stop clock
                    E54C CF     04  1AAF  1411          BISW2   #M_TIMEOUT,W_FLAGS          ; set timeout flag
                                02  1AB4  1412          REI                                 ; return to mainline code
                                1AB5  1413
                                1AB5  1414
                                1AB5  1415          $DS_PAGE
```

```
                                1AB5                .SBTTL  TEST 8:  UNIBUS Functional Test
                              00000000                        .PSECT  TEST_008, PAGE, NOWRT
                                  001C
                                  001C  1418 ;++
                                  001C  1419 ; Test description:
                                  001C  1420 ;
                                  001C  1421 ;       This test will be implemented to run using a Unibus Exerciser (UBE).
                                  001C  1422 ;       This test checks the UBI CSR, MAP's and various data transactions
                                  001C  1423 ;       using a Unibus Exerciser. The UBE must be set up at address 770000 (Oc-
                                  001C  1424 ;       tal) with vector 510 (Octal).
                                  001C  1425 ;
                                  001C  1426 ;--
                                  001C  1427
                                  001C  1428          $DS_BGNTEST     <DEFAULT,UBI,ALL>
                                  001C        DATA_008:
                         00000000 001C                .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                  0020        TEST_008::
                             0000 0020                .WORD   ^M<>            ; ENTRY MASK
                                  0022  1429
                00000000'EF   D5  0022  1430          TSTL              BE0_UNIT_NO                       ; BE0 selected?
                           10   18  0028  1431          BGEQ              0$                                ; br = BE0 selected
                                  002A  1432          $DS_PRINTF_S      FORMAT=T_NO_BE0                   ; print exit warning
                00000397'EF   9F  002A                        PUSHAB    T_NO_BE0
            000100F0 9F    01   FB  0030                        CALLS     #$$N, @#DS$PRINTF
                                  0037  1433          $DS_EXIT          TEST                              ; exit test
                       195E   31  0037                        BRW       TEST_008_X      ; EXIT TEST 8
                                  003A  1434
                         58   D4  003A  1435 0$:      CLRL              R8                                ; clear device number
                                  003C  1436
                                  003C  1437          $DS_PAGE
```

```
                                003C                    .SBTTL  SUBTEST 8.1:  CSR
                                003C  1440 ;+
                                003C  1441 ; Subtest description:
                                003C  1442 ;
                                003C  1443 ;       This subtest checks that the UBI CSR register is accessible and that
                                003C  1444 ;       writing a one to the status bits results in the bits remaining cleared.
                                003C  1445 ;       In error reporting the UNUSED bits will be masked out since they
                                003C  1446 ;       are unpredictable.  The bits are defined as follows:
                                003C  1447 ;
                                003C  1448 ;       CSR
                                003C  1449 ;           31 30    17    16            15       14 13     0
                                003C  1450 ;       +-----+--------+-----+------------+--------+--------+
                                003C  1451 ;       ! RDS ' Unused ! NXM ! Map Parity ! Wr Err ! Unused !
                                003C  1452 ;       +-----+--------+-----+------------+--------+--------+
                                003C  1453 ;
                                003C  1454 ;
                                003C  1455 ; Subtest steps:
                                003C  1456 ;
                                003C  1457 ;       SUBTEST CSR
                                003C  1458 ;       :  Attempt access to CSR
                                003C  1459 ;       :  IF unsuccessful
                                003C  1460 ;       :  :  THEN REPORT access error
                                003C  1461 ;       :  ENDIF
                                003C  1462 ;       :  CLEAR Status bits (with read)
                                003C  1463 ;       :  IF Status bits not clear
                                003C  1464 ;       :  :  THEN REPORT error
                                003C  1465 ;       :  ENDIF
                                003C  1466 ;       :  WRITE ones to CSR Status Bits
                                003C  1467 ;       :  READ CSR
                                003C  1468 ;       :  IF Status bits not clear
                                003C  1469 ;       :  :  THEN REPORT error
                                003C  1470 ;       :  ENDIF
                                003C  1471 ;       ENDSUBTEST CSR
                                003C  1472 ;
                                003C  1473 ; Errors:
                                003C  1474 ;
                                003C  1475 ;
                                003C  1476 ;-
                                003C  1477
                                003C  1478
                                003C  1479          $DS_BGNSUB
                                003C       T8_S1::
00010030 9F   00000000'EF   FA  003C                CALLG   $S$, a#DS$BGNSUB
                                0047  1480
              00000000'EF   B4  0047  1481          CLRW    W_FLAGS                         ; clear flags word
                                004D  1482 ;
                                004D  1483 ; Attempt access to UBI CSR, report error if access fails
                                004D  1484 ;
                                004D  1485 1$:      $DS_PROBE_S     UNIT=BE0_UNIT_NO,-
                                004D  1486                          LENGTH=#4,-
                                004D  1487                          ADDRESS=A_UBI_CSR
              00000000'EF   DD  004D                PUSHL   BE0_UNIT_NO
                         04   DD  0053                PUSHL   #4
              00F26010'EF   9F  0055                PUSHAB  A_UBI_CSR
         000101A0 9F    03   FB  005B                CALLS   #3, a#DS$PROBE
                                0062  1488
                                0062  1489          $DS_BNERROR     3$                      ; br = access successful
```

```
                          50   DD  0062                    PUSHL   R0
                          01   DD  0064                    PUSHL   #$ER
                00000000'8F    DD  0066                    PUSHL   #DS$K_BNERROR
            000100A8 9F   03   FB  006C                    CALLS   #3, @#DS$BRANCH
                       22 50   E8  0073                    BLBS    R0, 3$
                               0076  1490
            00000000'EF   01   A8  0076  1491        BISW2   #M_NO_CSR,W_FLAGS              ; set no-CSR flag
                               007D  1492        $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-       ; report no-access error
                               007D  1493                        MSGADR=T_CSR_ERR,-
                               007D  1494                        PRLINK=CSR_ERR
                000015DF'EF    DF  007D                    PUSHAL  CSR_ERR
                00000463'EF    DF  0083                    PUSHAL  T_CSR_ERR
                00000000'EF    DD  0089                    PUSHL   BEO_UNIT_NO
                          01   DD  008F                    PUSHL   #$ER
                               0091        ;;; TEST 8, SUBTEST 1, ERROR 1
            000100D0 9F   04   FB  0091                    CALLS   #$$M, @#DS$ERRHARD
                               0098  1495 ;
                               0098  1496 ; Attempt to clear status bits by reading CSR
                               0098  1497 ;
                00F26010 9F    D5  0098  1498 3$:      TSTL    @#A_UBI_CSR                 ; read CSR
57  00F26010 9F   7FFE3FFF 8F    CB  009E  1499          BICL3   #M_UBI_NON_STS,@#A_UBI_CSR,R7 ; read again, save status bits
                          57   D5  00AA  1500          TSTL    R7                          ; status bits clear?
                          2B   13  00AC  1501          BEQL    4$                          ; br = status bits clear
                               00AE  1502
                               00AE  1503          $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-       ; report UBI CSR data error
                               00AE  1504                          MSGADR=T_CSR_ERR,-
                               00AE  1505                          PRLINK=DATA_ERR,-
                               00AE  1506                          P1=#A_UBI_CSR,-          ; location
                               00AE  1507                          P2=#0,-                  ; expected value
                               00AE  1508                          P3=R7,-                  ; actual value
                               00AE  1509                          P4=#T_LONG_ERR           ; data type
                00000DC8'8F    DD  00AE                    PUSHL   #T_LONG_ERR
                          57   DD  00B4                    PUSHL   R7
                          00   DD  00B6                    PUSHL   #0
                00F26010 8F    DD  00B8                    PUSHL   #A_UBI_CSR
                000013C0'EF    DF  00BE                    PUSHAL  DATA_ERR
                00000463'EF    DF  00C4                    PUSHAL  T_CSR_ERR
                00000000'EF    DD  00CA                    PUSHL   BEO_UNIT_NO
                          02   DD  00D0                    PUSHL   #$ER
                               00D2        ;;; TEST 8, SUBTEST 1, ERROR 2
            000100D0 9F   08   FB  00D2                    CALLS   #$$M, @#DS$ERRHARD
                               00D9  1510
                               00D9  1511 4$:      $DS_CKLOOP      3$                      ; scope loop at 3$
        00010040 9F    BC AF   FA  00D9                    CALLG   3$, @#DS$CKLOOP
                               00E1  1512 ;
                               00E1  1513 ; Attempt to set Status bits, they should remain clear, report error otherwise
                               00E1  1514 ;
    00F26010 9F    8001C000 8F    C8  00E1  1515 5$:      BISL2   #M_UBI_STS,@#A_UBI_CSR      ; attempt to set Status bits
57  00F26010 9F    7FFE3FFF 8F    CB  00EC  1516          BICL3   #M_UBI_NON_STS,@#A_UBI_CSR,R7 ; read, save status bits
                          57   D5  00F8  1517          TSTL    R7                          ; status bits clear?
                          2B   13  00FA  1518          BEQL    7$                          ; br = status bits clear
                               00FC  1519
                               00FC  1520          $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-       ; report UBI CSR data error
                               00FC  1521                          MSGADR=T_CSR_ERR,-
                               00FC  1522                          PRLINK=DATA_ERR,-
                               00FC  1523                          P1=#A_UBI_CSR,-          ; location
                               00FC  1524                          P2=#0,-                  ; expected value
```

```
                                00FC  1525                        P3=R7,-                         ; actual value
                                00FC  1526                        P4=#T_LONG_ERR                  ; data type
                00000DC8'8F   DD  00FC                    PUSHL   #T_LONG_ERR
                          57  DD  0102                    PUSHL   R7
                          00  DD  0104                    PUSHL   #0
                00F26010 8F   DD  0106                    PUSHL   #A_UBI_CSR
                000013C0'EF   DF  010C                    PUSHAL  DATA_ERR
                00000463'EF   DF  0112                    PUSHAL  T_CSR_ERR
                00000000'EF   DD  0118                    PUSHL   BE0_UNIT_NO
                          03  DD  011E                    PUSHL   #$ER
                              0120          ;;; TEST 8, SUBTEST 1, ERROR 3
          00010DD0 9F   08    FB  0120                    CALLS   #$$M, @#DS$ERRHARD
                              0127  1527
                              0127  1528 7$:      $DS_CKLOOP      5$                              ; scope loop at 5$
          00010040 9F   B7 AF   FA  0127                    CALLG   5$, @#DS$CKLOOP
                              012F  1529
                              012F  1530          $DS_ENDSUB
                              012F          T8_S1_X:
     00010038 9F   00000000'EF    FA  012F                    CALLG   $S$, @#DS$ENDSUB
                              013A  1531
                              013A  1532          $DS_PAGE
```

```
                                013A                    .SBTTL  SUBTEST 8.2:  Map Data Bus
                                013A  1535 ;+
                                013A  1536 ; Subtest description:
                                013A  1537 ;
                                013A  1538 ;       This subtest checks the integrity of the map data bus by floating a one
                                013A  1539 ;       through a field of zeros and a zero through a field of ones. If all
                                013A  1540 ;       bits can toggle from a one to a zero then the data bus is considered
                                013A  1541 ;       to be good.  Only writable bits (31,25,<14:0>) are tested.  The remain-
                                013A  1542 ;       ing bits are ignored.  In error reporting the unused bits will be
                                013A  1543 ;       masked out . nce they are unpredictable.
                                013A  1544 ;
                                013A  1545 ; Subtest steps:
                                013A  1546 ;
                                013A  1547 ;       SUBTEST Map Data Bus
                                013A  1548 ;       :  REPEAT
                                013A  1549 ;       :  :  WRITE data pattern to first Map Entry
                                013A  1550 ;       :  :  READ pattern back
                                013A  1551 ;       :  :  IF error
                                013A  1552 ;       :  :  :  THEN REPORT error
                                013A  1553 ;       :  :  Update pattern
                                013A  1554 ;       :  : UNTIL all patterns written
                                013A  1555 ;       :  ENDREPEAT
                                013A  1556 ;       ENDSUBTEST Map Data Bus
                                013A  1557 ;
                                013A  1558 ; Errors:
                                013A  1559 ;
                                013A  1560 ;-
                                013A  1561          $DS_BGNSUB
                                013A            T8_S2::
00010030 9F   0000000C'EF   FA  013A                    CALLG   $S$, @#DS$BGNSUB
                                0145  1562
              00000000'EF   B4  0145  1563          CLRW    W_FLAGS                         ; clear flags word
                                014B  1564 ;
                                014B  1565 ; Write patterns, read back, report errors if any
                                014B  1566 ;
         55   000001A7'EF   DE  014B  1567          MOVAL   MAP_ENT_TBL,R5                  ; load address of data table
      00F26800 9F     65   D0  0152  1568 1$:       MOVL    (R5),@#A_MAP_BGN                ; write pattern to map entry
57 00F26800 9F   7DFF8000 8F   CB  0159  1569          BICL3   #M_MAP_NOWRT,@#A_MAP_BGN,R7      ; load actual data into R7
                      65   57   D1  0165  1570          CMPL    R7,(R5)                         ; check data
                           2B   13  0168  1571          BEQL    4$                              ; br = ok
                                016A  1572
                                016A  1573          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report Map Entry data error
                                016A  1574                          MSGADR=T_MAP_DBUS_ERR,-
                                016A  1575                          PRLINK=DATA_ERR,-
                                016A  1576                          P1=#A_MAP_BGN,-          ; location
                                016A  1577                          P2=(R5),-                ; expected value
                                016A  1578                          P3=R7,-                  ; actual value
                                016A  1579                          P4=#T_LONG_ERR           ; data type
              00000DC8'8F   DD  016A                        PUSHL   #T_LONG_ERR
                         57   DD  0170                        PUSHL   R7
                         65   DD  0172                        PUSHL   (R5)
              00F26800 8F     DD  0174                        PUSHL   #A_MAP_BGN
              000013C0'EF     DF  017A                        PUSHAL  DATA_ERR
              00000482'EF     DF  0180                        PUSHAL  T_MAP_DBUS_ERR
              00000000'EF     DD  0186                        PUSHL   BE0_UNIT_NO
                         01   DD  018C                        PUSHL   #$ER
                              018E          ;;; TEST 8, SUBTEST 2, ERROR 1
```

```
                000100D0 9F    08    FB  018E                        CALLS   #$$M, @#DS$ERRHARD
                                         0195  1580
                                         0195  1581 4$:    $DS_CKLOOP      1$                          ; scope loop at 1$
              00010040 9F    BA AF    FA  0195                        CALLG   1$, @#DS$CKLOOP
                                         019D  1582
                 02007FFF 8F    85    D1  019D  1583        CMPL    (R5)+,#^X02007FFF                   ; patterns exhausted ?
                                AC    12  01A4  1584        BNEQ    1$                                  ; br = patterns not exhausted
                                         01A6  1585
                                         01A6  1586        $DS_ENDSUB
                                         01A6          T8_S2_X:
     00010038 9F    0000000C'EF    FA  01A6                        CALLG   $S$, @#DS$ENDSUB
                                         01B1  1587
                                         01B1  1588        $DS_PAGE
```

```
01B1  1590
01B1                    .SBTTL  SUBTEST 8.3:  Map Address Bus
01B1  1592 ;+
01B1  1593 ; Subtest description:
01B1  1594 ;
01B1  1595 ;       This subtest checks that there are no map address bits stuck or
01B1  1596 ;       shorted. A unique pattern is written into each map entry at maps num-
01B1  1597 ;       bered 0, 1, 2, 4, 8, 16, 32, 64, and 128, (so that each address
01B1  1598 ;       bit assumes both a zero and a one). Reading each of these map
01B1  1599 ;       entries after all patterns are written will reveal any stuck or short-
01B1  1600 ;       ed address bus bits because the last pattern written to a doubly ad-
01B1  1601 ;       dressed map will overwrite the earlier pattern written. Unfortunately
01B1  1602 ;       a data bit fault in one of the RAMs in a particular map entry may point
01B1  1603 ;       erroneously to an address bus bit fault. Therefore, to make this test
01B1  1604 ;       insensitive to a limited number of RAM data bit failures, it is
01B1  1605 ;       necessary to encode each value prior to storing. The Reed-Muller (16,4)
01B1  1606 ;       error-correcting code is used.  In error reporting the unused bits are
01B1  1607 ;       masked out since they are unpredictable.
01B1  1608 ;
01B1  1609 ;       EXPLANATION OF REED-MULLER ERROR-CORRECTING CODE
01B1  1610 ;
01B1  1611 ;       The basis of the Reed-Muller (16,4) coding space comprises the vectors:
01B1  1612 ;
01B1  1613 ;               I       1111111111111111
01B1  1614 ;               V1      0101010101010101
01B1  1615 ;               V2      0011001100110011
01B1  1616 ;               V3      0000111100001111
01B1  1617 ;
01B1  1618 ;       A word is encoded using the following expression:
01B1  1619 ;
01B1  1620 ;               CODEWORD = G0*I XOR G1*V1 XOR G2*V2 XOR G3*V3
01B1  1621 ;
01B1  1622 ;       where G0 is the least significant bit and G3 the most significant bit.
01B1  1623 ;
01B1  1624 ;       Each codeword has 8 disjoint redundant relations due to the nature of
01B1  1625 ;       the encoding vectors.  In V1 the redundant relations are the bit pairs
01B1  1626 ;       (31,30), (29,28), (27,26), etc.  In V2, they are the bit pairs (31,29),
01B1  1627 ;       (30,28), (27,25), etc.  In V3, the relations are (31,27), (30,26),
01B1  1628 ;       (29,25), etc.  Thus, there are three sets of redundant relations in
01B1  1629 ;       each codeword, and therefore, each bit is represented by 8 disjoint
01B1  1630 ;       relations.
01B1  1631 ;
01B1  1632 ;       To decode a particular bit, the value of each of the 8 redundant rela-
01B1  1633 ;       tions is calculated.  The position of the bit in the decoded word de-
01B1  1634 ;       termines which bit pairs will be used.  The original word is represent-
01B1  1635 ;       as:
01B1  1636 ;
01B1  1637 ;               +--+--+--+--+
01B1  1638 ;               |G3|G2|G1|G0|
01B1  1639 ;               +--+--+--+--+
01B1  1640 ;
01B1  1641 ;       The bits G0 - G3 are referred to a  information bits.  The LSB has no
01B1  1642 ;       inherent redundant relations, since it is represented by the product
01B1  1643 ;       of itself and the identity vector.  Here it is convenient to introduce
01B1  1644 ;       the notion of pair separation.  The pair separation for the redundant
01B1  1645 ;       relations representing G1 is 1; for G2, 2, and for G3, 4.
01B1  1646 ;
```

```
                                01B1  1647 ;       After decoding the value of each redundant relation for a particular
                                01B1  1648 ;       bit, these values are summed.  If there was no corruption of the code-
                                01B1  1649 ;       word, this sum will be 0 or 8, corresponding to an original value of
                                01B1  1650 ;       0 or 1.  If there was any corruption, a majority decision must be made,
                                01B1  1651 ;       i.e., if the sum is 4 or greater, the bit Gn is assumed to have been a
                                01B1  1652 ;       1, otherwise, it is assumed 0.
                                01B1  1653 ;
                                01B1  1654 ;       After decoding the three MSBs, the LSB is decoded by eliminating each
                                01B1  1655 ;       of the terms G1*V1, G2*V2, and G3*V3, leaving simply G0*I.  If there
                                01B1  1656 ;       had been no corruption, this term would be 16 0s or 16 1s, representing
                                01B1  1657 ;       0 or 1 respectively.  Otherwise, another majority decision must be
                                01B1  1658 ;       made, i.e., if 8 or more bits are set, G0 is a 1, else, it is 0.
                                01B1  1659 ;
                                01B1  1660 ;
                                01B1  1661 ; Subtest steps:
                                01B1  1662 ;
                                01B1  1663 ;       SUBTEST Map Address Bus
                                01B1  1664 ;       :  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
                                01B1  1665 ;       :  :  WRITE pattern to Map location
                                01B1  1666 ;       :  ENDDO
                                01B1  1667 ;       :  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
                                01B1  1668 ;       :  :  READ pattern from Map location
                                01B1  1669 ;       :  ENDDO
                                01B1  1670 ;       :  DECODE Data retrieved
                                01B1  1671 ;       :  IF error
                                01B1  1672 ;       :  :  THEN REPORT error
                                01B1  1673 ;       :  ENDIF
                                01B1  1674 ;       ENDSUBTEST Map Address Bus
                                01B1  1675 ;
                                01B1  1676 ; Errors:
                                01B1  1677 ;
                                01B1  1678 ;-
                                01B1  1679         $DS_BGNSUB
                                01B1       T8_S3::
00010030 9F   00000018'EF   FA  01B1                CALLG   $S$, @#DS$BGNSUB
                                01BC  1680 ;
                                01BC  1681 ; Generate Reed-Muller code and load into Map Entry PFN and Byte-offset bits
                                01BC  1682 ;
            0000172F'EF   16    01BC  1683         JSB     RD_MLLR_ENCODE                      ; encode data
            00000000'EF   B4    01C2  1684         CLRW    W_FLAGS                             ; clear flags word
                  54      7C    01C8  1685         CLRQ    R4                                  ; clear indices
00F26800 9F44 00000237'EF 45    D0    01CA  1686 1$:   MOVL    RD_MLLR_TBL[R5],@#A_MAP_BGN[R4] ; load pattern into Map Entry
                  55      D6    01D7  1687         INCL    R5                                  ; incr index
            54    54      01    78    01D9  1688         ASHL    #1,R4,R4                            ; float 1 to next Map No bit
                  02      12    01DD  1689         BNEQ    2$                                  ; br = index not 0
                  54      D6    01DF  1690         INCL    R4                                  ; load 1 into index
         00000200 8F      54    D1    01E1  1691 2$:   CMPL    R4,#512                             ; Map Addresses exhausted ?
                  E0      19    01E8  1692         BLSS    1$                                  ; br = Map Nos not exhausted
                                01EA  1693 ;
                                01EA  1694 ; Retrieve data from Map Entries, load into table
                                01EA  1695 ;
                  54      7C    01EA  1696         CLRQ    R4                                  ; clear indices
00F26800 9F44   7DFF8000 8F     CB    01EC  1697 3$:   BICL3   #M_MAP_NOWRT,@#A_MAP_BGN[R4],RETRIEVE_TBL[R5] ; load act'l data
            00000257'EF 45            01F8
                  55      D6    01FE  1698         INCL    R5                                  ; incr Reed-Muller table index
            54    54      01    78    0200  1699         ASHL    #1,R4,R4                            ; float 1 to next Map No bit
                  02      12    0204  1700         BNEQ    4$                                  ; br = index not 0
```

```
                      54   D6  0206  1701          INCL    R4                              ; load 1 into index
            00000200 8F    54   D1  0208  1702 4$:     CMPL    R4,#512                         ; Map Nos exhausted ?
                           DB   19  020F  1703          BLSS    3$                              ; br = Map Nos not exhausted
                                0211  1704 ;
                                0211  1705 ; Decode retrieved data, report errors if any
                                0211  1706 ;
                0000175E'EF    16   0211  1707          JSB     RD_MLLR_DECODE                  ; decode data retrieved
                           54   7C  0217  1708          CLRQ    R4                              ; clear indices
                00000277'EF45  55   D1  0219  1709 5$:     CMPL    R5,DECODE_TBL[R5]               ; check for data error
                           1D   13  0221  1710          BEQL    6$                              ; br = no error
                                0223  1711
                                0223  1712          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-              ; report Map Address bus err
                                0223  1713                          MSGADR=T_MAP_ABUS_ERR,-
                                0223  1714                          PRLINK=MAP_ADR_ERR,-
                                0223  1715                          P1=R5                       ; bit in error
                           55   DD  0223                          PUSHL   R5
                    000013E9'EF   DF  0225                          PUSHAL  MAP_ADR_ERR
                    000004AC'EF   DF  022B                          PUSHAL  T_MAP_ABUS_ERR
                    00000000'EF   DD  0231                          PUSHL   BE0_UNIT_NO
                           01   DD  0237                          PUSHL   #$ER
                                0239      ;;; TEST 8, SUBTEST 3, ERROR 1
            000100D0 9F    05   FB  0239                          CALLS   #$$M, @#DS$ERRHARD
                                0240  1716
                     D5 55     07   F3  0240  1717 6$:     AOBLEQ  #7,R5,5$                        ; Map Nos exhausted ?
                                0244  1718
                                0244  1719          $DS_ENDSUB
                                0244      T8_S3_X:
    00010038 9F    00000018'EF    FA  0244                          CALLG   $S$, @#DS$ENDSUB
                                024F  1720
                                024F  1721          $DS_PAGE
```

```
                                        .SBTTL  SUBTEST 8.4:  Map Entry
                                ;+
                                ; Subtest description:
                                ;
                                ;       This subtest checks that all readable/writeable bits of each map entry
                                ;       are not stuck at zero or one. This is accomplished by writing all zeros
                                ;       to a map entry, then reading it to make certain the bits are not stuck
                                ;       at 1.  Then a pattern of all ones is written to the entry and read
                                ;       again to verify that it contains all ones to check that there are no
                                ;       bits stuck at zero. This is repeated for all map entries.  Again, in
                                ;       error reporting the unused bits will be masked out since they are
                                ;       unpredictable.
                                ;
                                ; Subtest steps:
                                ;
                                ;       SUBTEST Map Entry
                                ;       :  REPEAT
                                ;       :  :  CLEAR Map Entry writable bits
                                ;       :  :  IF Map Entry not clear
                                ;       :  :  :  THEN REPORT error
                                ;       :  :  ENDIF
                                ;       :  :  WRITE Mask to Map Entry
                                ;       :  :  IF Map Entry NEQ Mask
                                ;       :  :  :  THEN REPORT error
                                ;       :  :  ENDIF
                                ;       :  :  CLEAR Map Entry writable bits
                                ;       :  :  IF Map Entry not clear
                                ;       :  :  :  THEN REPORT error
                                ;       :  :  ENDIF
                                ;       :  : UNTIL all Map Entries tested
                                ;       :  ENDREPEAT
                                ;       ENDSUBTEST Map Entry
                                ;
                                ; Errors:
                                ;
                                ;-
                                        $DS_BGNSUB
                                T8_S4::
00010030 9F   00000024'EF   FA          CALLG   $S$, @#DS$BGNSUB
                                
              00000000'EF   B4          CLRW    W_FLAGS                         ; clear flags word
                                ;
                                ; Test the ability to clear map entry, report errors if any
                                ;
        54    00F26800 8F   D0          MOVL    #A_MAP_BGN,R4                   ; load map starting address
                       64   D4  10$:     CLRL    (R4)                            ; clear Map Entry
    57  64    7DFF8000 8F   CB          BICL3   #M_MAP_NOWRT,(R4),R7            ; load actual data into R7
                       27   13          BEQL    30$                             ; br = Map Entry clear
                                
                                        $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report Map Entry data error
                                                        MSGADR=T_MAP_ENT_ERR,-
                                                        PRLINK=DATA_ERR,-
                                                        P1=R4,-                 ; location
                                                        P2=#0,-                 ; expected value
                                                        P3=R7,-                 ; actual value
                                                        P4=#T_LONG_ERR          ; data type
              00000DC8'8F   DD          PUSHL   #T_LONG_ERR
```

```
                        57   DD  0279                        PUSHL   R7
                        00   DD  027B                        PUSHL   #0
                        54   DD  027D                        PUSHL   R4
              000013C0'EF    DF  027F                        PUSHAL  DATA_ERR
              000004D9'EF    DF  0285                        PUSHAL  T_MAP_ENT_ERR
              00000000'EF    DD  028B                        PUSHL   BE0_UNIT_NO
                        01   DD  0291                        PUSHL   #$ER
                             0293            ;;; TEST 8, SUBTEST 4, ERROR 1
         000100D0 9F    08   FB  0293                        CALLS   #$$M, @#DS$ERRHARD
                             029A  1777
                             029A  1778 30$:    $DS_CKLOOP      10$                     ; scope loop at 10$
       00010040 9F    CA AF    FA  029A                        CALLG   10$, @#DS$CKLOOP
                             02A2  1779 ;
                             02A2  1780 ; Test ability to set writable bits, report errors if any
                             02A2  1781 ;
         64    82007FFF 8F    D0  02A2  1782 40$:    MOVL    #M_MAP_WRTBL,(R4)       ; set Map Entry writable bits
   57    64    7DFF8000 8F    CB  02A9  1783         BICL3   #M_MAP_NOWRT,(R4),R7    ; load actual data into R7
         82007FFF 8F    57    D1  02B1  1784         CMPL    R7,#M_MAP_WRTBL         ; Map Entry = mask ?
                        2B   13  02B8  1785         BEQL    45$                     ; br = Map Entry = mask
                             02BA  1786
                             02BA  1787         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-   ; report Map Entry data error
                             02BA  1788                         MSGADR=T_MAP_ENT_ERR,-
                             02BA  1789                         PRLINK=DATA_ERR,-
                             02BA  1790                         P1=R4,-              ; location
                             02BA  1791                         P2=#M_MAP_WRTBL,-    ; expected value
                             02BA  1792                         P3=R7,-              ; actual value
                             02BA  1793                         P4=#T_LONG_ERR       ; data type
              00000DC8'8F    DD  02BA                        PUSHL   #T_LONG_ERR
                        57   DD  02C0                        PUSHL   R7
              82007FFF 8F    DD  02C2                        PUSHL   #M_MAP_WRTBL
                        54   DD  02C8                        PUSHL   R4
              000013C0'EF    DF  02CA                        PUSHAL  DATA_ERR
              000004D9'EF    DF  02D0                        PUSHAL  T_MAP_ENT_ERR
              00000000'EF    DD  02D6                        PUSHL   BE0_UNIT_NO
                        02   DD  02DC                        PUSHL   #$ER
                             02DE            ;;; TEST 8, SUBTEST 4, ERROR 2
         000100D0 9F    08   FB  02DE                        CALLS   #$$M, @#DS$ERRHARD
                             02E5  1794
                             02E5  1795 45$:    $DS_CKLOOP      40$                     ; scope loop at 40$
       00010040 9F    BA AF    FA  02E5                        CALLG   40$, @#DS$CKLOOP
                             02ED  1796 ;
                             02ED  1797 ; Test ability to clear writable bits, report errors if any
                             02ED  1798 ;
                        64   D4  02ED  1799 50$:    CLRL    (R4)                    ; clear Map Entry
   57    64    7DFF8000 8F    CB  02EF  1800         BICL3   #M_MAP_NOWRT,(R4),R7    ; load actual data into R7
                        27   13  02F7  1801         BEQL    60$                     ; br = Map Entry clear
                             02F9  1802
                             02F9  1803         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-   ; report Map Entry data error
                             02F9  1804                         MSGADR=T_MAP_ENT_ERR,-
                             02F9  1805                         PRLINK=DATA_ERR,-
                             02F9  1806                         P1=R4,-              ; location
                             02F9  1807                         P2=#0,-              ; expected value
                             02F9  1808                         P3=R7,-              ; actual value
                             02F9  1809                         P4=#T_LONG_ERR       ; data type
              00000DC8'8F    DD  02F9                        PUSHL   #T_LONG_ERR
                        57   DD  02FF                        PUSHL   R7
                        00   DD  0301                        PUSHL   #0
```

```
                      54   DD  0303                        PUSHL   R4
              000013C0'EF   DF  0305                        PUSHAL  DATA_ERR
              000004D9'EF   DF  030B                        PUSHAL  T_MAP_ENT_ERR
              00000000'EF   DD  0311                        PUSHL   BE0_UNIT_NO
                      03    DD  0317                        PUSHL   #SER
                            0319        ;;; TEST 8, SUBTEST 4, ERROR 3
          000100D0 9F   08  FB  0319                        CALLS   #S$M, @#DS$ERRHARD
                            0320  1810
                            0320  1811 60$:    $DS_CKLOOP      50$                     ; scope loop at 50$
          00010040 9F   CA AF FA  0320                      CALLG   50$, @#DS$CKLOOP
                            0328  1812
    FF35 54   04   00F26FFC 8F  F1  0328  1813              ACBL    #A_MAP_END,#4,R4,10$            ; incr index and br = not done
                            0332  1814
                            0332  1815      $DS_ENDSUB
                            0332        T8_S4_X:
     00010038 9F   00000024'EF  FA  0332                    CALLG   $S$, @#DS$ENDSUB
                            033D  1816
                            033D  1817      $DS_PAGE
```

```
                        .SBTTL  SUBTEST 8.5:  CPU Read/Write
                ;+
                ; Subtest description:
                ;
                ;       This subtest checks the ability of the UBI to handle CPU reads
                ;       and writes to the unibus. The UBE data and address registers are writ-
                ;       ten and read, checking for stuck at one or zero faults. The following
                ;       data patterns are used to perform this test:
                ;
                ;               0101010101010101
                ;               1010101010101010
                ;               0011001100110011
                ;               0000111100001111
                ;               0000000011111111
                ;
                ;       to check for word writes, and
                ;
                ;               01010101
                ;               10101010
                ;               00110011
                ;               00001111
                ;
                ;       to check for byte writes.
                ;
                ; Subtest steps:
                ;
                ;       SUBTEST CPU Read/Write
                ;       :  PROBE UBE Data Register
                ;       :  IF not present
                ;       :  :  THEN REPORT error
                ;       :  ENDIF
                ;       :  REPEAT
                ;       :  :  WRITE word pattern to Address register
                ;       :  :  READ Address register
                ;       :  :  IF error
                ;       :  :  :  THEN REPORT error
                ;       :  :  ENDIF
                ;       :  : UNTIL patterns exhausted
                ;       :  ENDREPEAT
                ;       :  REPEAT
                ;       :  :  WRITE word pattern to Data register
                ;       :  :  READ Data register
                ;       :  :  IF error
                ;       :  :  :  THEN DO
                ;       :  :  :  :  REPORT error
                ;       :  :  :  :  EXIT Test
                ;       :  :  :  ENDDO
                ;       :  :  ENDIF
                ;       :  : UNTIL patterns exhausted
                ;       :  ENDREPEAT
                ;       :  REPEAT
                ;       :  :  WRITE byte pattern to Data register high byte
                ;       :  :  WRITE byte pattern to Data register low byte
                ;       :  :  READ Data register high byte
                ;       :  :  IF error
                ;       :  :  :  THEN REPORT error
                ;       :  :  ENDIF
```

```
                                033D  1876 ;       :  :  READ Data register low byte
                                033D  1877 ;       :  :  IF error
                                033D  1878 ;       :  :  :  THEN REPORT error
                                033D  1879 ;       :  :  ENDIF
                                033D  1880 ;       :  : UNTIL patterns exhausted
                                033D  1881 ;       :  ENDREPEAT
                                033D  1882 ;       ENDSUBTEST CPU Read/Write
                                033D  1883 ;
                                033D  1884 ; Errors:
                                033D  1885 ;
                                033D  1886 ;-
                                033D  1887         $DS_BGNSUB
                                033D       T8_S5::
00010030 9F   00000030'EF   FA  033D                 CALLG   $S$, @#DS$BGNSUB
                                0348  1888
              00000000'EF   B4  0348  1889         CLRW    W_FLAGS                         ; clear flags word
                                034E  1890 ;
                                034E  1891 ; Test UBE accessibility.  Report error and exit if not accessible.
                                034E  1892 ;
                                034E  1893         $DS_PROBE_S-                        ; probe Data Register
                                034E  1894                 UNIT=BE0_UNIT_NO,-
                                034E  1895                 LENGTH=#2,-                 ; word access
                                034E  1896                 ADDRESS=@A_BE0_DB
              00000000'EF   DD  034E                 PUSHL   BE0_UNIT_NO
                         02   DD  0354                 PUSHL   #2
              000002AF'FF   9F  0356                 PUSHAB  @A_BE0_DB
         000101A0 9F    03   FB  035C                 CALLS   #3, @#DS$PROBE
                                0363  1897
                                0363  1898         $DS_BNERROR     0$                  ; br = no access error
                         50   DD  0363                 PUSHL   R0
                         01   DD  0365                 PUSHL   #$ER
              00000000'8F   DD  0367                 PUSHL   #DS$K_BNERROR
    000100A8 9F    03   FB  036D                 CALLS   #3, @#DS$BRANCH
                    24 50   E8  0374                 BLBS    R0, 0$
                                0377  1899
                                0377  1900         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-  ; report error
                                0377  1901                 MSGADR=T_NO_ACCESS,-
                                0377  1902                 PRLINK=NO_ACCESS,-
                                0377  1903                 P1=#^XFFF000                ; device address
              00FFF000 8F   DD  0377                 PUSHL   #^XFFF000
              0000171B'EF   DF  037D                 PUSHAL  NO_ACCESS
              000003C3'EF   DF  0383                 PUSHAL  T_NO_ACCESS
              00000000'EF   DD  0389                 PUSHL   BE0_UNIT_NO
                         01   DD  038F                 PUSHL   #$ER
                                0391       ;;; TEST 8, SUBTEST 5, ERROR 1
    000100D0 9F    05   FB  0391                 CALLS   #$$M, @#DS$ERRHARD
                                0398  1904
                                0398  1905         $DS_EXIT        TEST                ; exit test
                    15FD   31   0398                 BRW     TEST_008_X      ; EXIT TEST 8
                                039B  1906 ;
                                039B  1907 ; Test UBE Address Register with word writes
                                039B  1908 ;
55    0000029F'EF   3E  039B  1909 0$:    MOVAW   W_PTRN_TBL,R5                       ; load address of pattern table
000002B7'FF   65    B0  03A2  1910 1$:    MOVW    (R5),@A_BE0_BA                      ; write pattern to address reg.
000002B7'FF   65    B1  03A9  1911        CMPW    (R5),@A_BE0_BA                      ; check address register
              32    13  03B0  1912        BEQL    2$                                  ; br = ok
                                03B2  1913
```

```
                57   000002B7'FF    B0  03B2  1914          MOVW    @A_BE0_BA,R7                    ; read data register
                                        03B9  1915          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                        03B9  1916                          MSGADR=T_ADR_REG_ERR,-
                                        03B9  1917                          PRLINK=DATA_ERR,-
                                        03B9  1918                          P1=A_BE0_BA,-            ; location
                                        03B9  1919                          P2=(R5),-                ; expected value
                                        03B9  1920                          P3=R7,-                  ; actual error
                                        03B9  1921                          P4=#T_WORD_ERR           ; data type
            00000DA0'8F    DD  03B9                          PUSHL   #T_WORD_ERR
                       57  DD  03BF                          PUSHL   R7
                       65  DD  03C1                          PUSHL   (R5)
            000002B7'EF    DD  03C3                          PUSHL   A_BE0_BA
            000013C0'EF    DF  03C9                          PUSHAL  DATA_ERR
            00000511'EF    DF  03CF                          PUSHAL  T_ADR_REG_ERR
            00000000'EF    DD  03D5                          PUSHL   BE0_UNIT_NO
                       02  DD  03DB                          PUSHL   #SER
                           03DD          ;;; TEST 8, SUBTEST 5, ERROR 2
        000100D0 9F    08  FB  03DD                          CALLS   #$$M, @#DS$ERRHARD
                           03E4  1922
                           03E4  1923 2$:      $DS_CKLOOP      1$                              ; scope loop at 1$
    00010040 9F    BB AF   FA  03E4                          CALLG   1$, @#DS$CKLOOP
                           03EC  1924
            0F0F 8F    85  B1  03EC  1925 3$:      CMPW    (R5)+,#^X0F0F                   ; patterns exhausted ?
                       AF  12  03F1  1926          BNEQ    1$                              ; br = patterns not exhausted
                           03F3  1927
                           03F3  1928 ;
                           03F3  1929 ; Test UBE Data Register with word writes
                           03F3  1930 ;
        55   0000029F'EF    3E  03F3  1931          MOVAW   W_PTRN_TBL,R5                   ; load address of pattern table
    000002AF'FF    65  B0  03FA  1932 4$:      MOVW    (R5),@A_BE0_DB                  ; write pattern to data register
    000002AF'FF    65  B1  0401  1933          CMPW    (R5),@A_BE0_DB                  ; check data register
                       32  13  0408  1934          BEQL    5$                              ; br = ok
                           040A  1935
        57   000002AF'FF    B0  040A  1936          MOVW    @A_BE0_DB,R7                    ; read data register
                           0411  1937          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                           0411  1938                          MSGADR=T_DATA_REG_ERR,-
                           0411  1939                          PRLINK=DATA_ERR,-
                           0411  1940                          P1=A_BE0_DB,-            ; location
                           0411  1941                          P2=(R5),-                ; expected value
                           0411  1942                          P3=R7,-                  ; actual value
                           0411  1943                          P4=#T_WORD_ERR           ; data type
            00000DA0'8F    DD  0411                          PUSHL   #T_WORD_ERR
                       57  DD  0417                          PUSHL   R7
                       65  DD  0419                          PUSHL   (R5)
            000002AF'EF    DD  041B                          PUSHL   A_BE0_DB
            000013C0'EF    DF  0421                          PUSHAL  DATA_ERR
            00000540'EF    DF  0427                          PUSHAL  T_DATA_REG_ERR
            00000000'EF    DD  042D                          PUSHL   BE0_UNIT_NO
                       03  DD  0433                          PUSHL   #SER
                           0435          ;;; TEST 8, SUBTEST 5, ERROR 3
        000100D0 9F    08  FB  0435                          CALLS   #$$M, @#DS$ERRHARD
                           043C  1944
                           043C  1945 5$:      $DS_CKLOOP      4$                              ; scope loop at 4$
    00010040 9F    BB AF   FA  043C                          CALLG   4$, @#DS$CKLOOP
                           0444  1946
            00FF 8F    85  B1  0444  1947          CMPW    (R5)+,#^X00FF                   ; patterns exhausted ?
                       AF  12  0449  1948          BNEQ    4$                              ; br = patterns not exhausted
```

```
                                044B  1949
                                044B  1950 ;
                                044B  1951 ; Test UBE Data Register high and low bytes with byte writes
                                044B  1952 ;
          54   000002AF'EF   D0   044B  1953          MOVL    A_BE0_DB,R4                     ; load adr of UBE data reg
          55   0000029F'EF   3E   0452  1954 6$:       MOVAW   W_PTRN_TBL,R5                   ; load adr of pattern table
               64   65       90   0459  1955 7$:       MOVB    (R5),(R4)                       ; write pattern to data byte
               64   65       91   045C  1956          CMPB    (R5),(R4)                       ; check data byte
                    2A       13   045F  1957          BEQL    8$                              ; br = ok
                                0461  1958
               57   64       90   0461  1959          MOVB    (R4),R7                         ; read data register
                                0464  1960          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                0464  1961                          MSGADR=T_DATA_REG_ERR,-
                                0464  1962                          PRLINK=DATA_ERR,-
                                0464  1963                          P1=R4,-                   ; location
                                0464  1964                          P2=(R5),-                 ; expected value
                                0464  1965                          P3=R7,-                   ; actual value
                                0464  1966                          P4=#T_BYTE_ERR            ; data type
          00000D72'8F   DD   0464                       PUSHL   #T_BYTE_ERR
                   57   DD   046A                       PUSHL   R7
                   65   DD   046C                       PUSHL   (R5)
                   54   DD   046E                       PUSHL   R4
          000013C0'EF   DF   0470                       PUSHAL  DATA_ERR
          00000540'EF   DF   0476                       PUSHAL  T_DATA_REG_ERR
          00000000'EF   DD   047C                       PUSHL   BE0_UNIT_NO
                   04   DD   0482                       PUSHL   #$ER
                                0484      ;;; TEST 8, SUBTEST 5, ERROR 4
     000100D0 9F   08   FB   0484                       CALLS   #$$M, @#DS$ERRHARD
                                048B  1967
                                048B  1968 8$:       $DS_CKLOOP      7$                      ; scope loop at 7$
     00010040 9F   CB AF   FA   048B                   CALLG   7$, @#DS$CKLOOP
                                0493  1969
               0F0F 8F   85   B1   0493  1970          CMPW    (R5)+,#^X0F0F                   ; patterns exhausted ?
                         BF   12   0498  1971          BNEQ    7$                              ; br = patterns not exhausted
                  B4 54    00   E3   049A  1972          BBCS    #0,R4,6$                        ; br = low & high bytes tested
                                049E  1973 9$:
                                049E  1974          $DS_ENDSUB
                                049E      T8_S5_X:
00010038 9F   00000030'EF   FA   049E                   CALLG   $S$, @#DS$ENDSUB
                                04A9  1975
                                04A9  1976          $DS_PAGE
```

```
                                04A9                    .SBTTL  SUBTEST 8.6:  Interrupts
                                04A9  1979 ;+
                                04A9  1980 ; Subtest description:
                                04A9  1981 ;
                                04A9  1982 ;       This subtest checks the ability of the UBI to handle interrupts at BR4-
                                04A9  1983 ;       BR7, using the UBE to initiate interrupts at each BR level.  The
                                04A9  1984 ;       service routine is set up to set flags if the interrupt occurs or if
                                04A9  1985 ;       error conditions (at wrong IPL, wrong vector) exist.  Any errors are
                                04A9  1986 ;       reported.
                                04A9  1987 ;
                                04A9  1988 ; Subtest steps:
                                04A9  1989 ;
                                04A9  1990 ;       SUBTEST INTERRUPTS
                                04A9  1991 ;       :  Set IPL at BR7 (17(X))
                                04A9  1992 ;       :  Set up UBE to interrupt upon becoming bus master at all 4 levels
                                04A9  1993 ;       :  DO FOR BR level = {4,5,6,7}
                                04A9  1994 ;       :  :  Set IPL to prohibit BR interrupt
                                04A9  1995 ;       :  :  IF interrupt occurs
                                04A9  1996 ;       :  :  :  THEN REPORT error
                                04A9  1997 ;       :  :  ENDIF
                                04A9  1998 ;       :  :  Decrement IPL
                                04A9  1999 ;       :  :  If interrupt does not occur OR at wrong IPL OR wrong vector
                                04A9  2000 ;       :  :  :  THEN REPORT error
                                04A9  2001 ;       :  :  ENDIF
                                04A9  2002 ;       :  ENDDO
                                04A9  2003 ;       ENDSUBTEST INTERRUPTS
                                04A9  2004 ;
                                04A9  2005 ; Errors:
                                04A9  2006 ;
                                04A9  2007 ;-
                                04A9  2008          $DS_BGNSUB
                                04A9            T8_S6::
00010030 9F   0000003C'EF   FA  04A9                        CALLG   $S$, @#DS$BGNSUB
                                04B4  2009
                                04B4  2010          $DS_CHANNEL_S   UNIT=BE0_UNIT_NO,-       ; clear channel status
                                04B4  2011                          FUNC=#CHC$_CLEAR,-
                                04B4  2012                          VECADR=BR_SERVICE,-
                                04B4  2013                          STSADR=L_UBI_STATUS
          000000AF'EF   7F      04B4                          PUSHAQ  L_UBI_STATUS
          00001A3C'EF   DF      04BA                          PUSHAL  BR_SERVICE
          00000000'8F   DD      04C0                          PUSHL   #CHC$_CLEAR
          00000000'EF   DD      04C6                          PUSHL   BE0_UNIT_NO
       00010180 9F      04   FB  04CC                          CALLS   #4, @#DS$CHANNEL
                                04D3  2014          $DS_BNERROR     0$                       ; br = no error
                     50   DD      04D3                          PUSHL   R0
                     01   DD      04D5                          PUSHL   #$ER
          00000000'8F   DD      04D7                          PUSHL   #DS$K_BNERROR
       000100A8 9F      03   FB  04DD                          CALLS   #3, @#DS$BRANCH
                  1E 50   E8      04E4                          BLBS    R0, 0$
                                04E7  2015          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report channel error
                                04E7  2016                          MSGADR=T_CHANNEL_ERR,-
                                04E7  2017                          PRLINK=CHANNEL_ERR
          000016CC'EF   DF      04E7                          PUSHAL  CHANNEL_ERR
          000012FF'EF   DF      04ED                          PUSHAL  T_CHANNEL_ERR
          00000000'EF   DD      04F3                          PUSHL   BE0_UNIT_NO
                     01   DD      04F9                          PUSHL   #$ER
                                04FB                ;;; TEST 8, SUBTEST 6, ERROR 1
```

```
000100D0 9F   04    FB  04FB                      CALLS   #$$M, @#DS$ERRHARD
                        0502  2018          $DS_ABORT       TEST                    ; abort test
              50    D4  0502                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                    04  0504                      RET                     ; TERMINATE TEST
                        0505  2019
                        0505  2020 0$:      $DS_CHANNEL_S   UNIT=BE0_UNIT_NO,-      ; enable interrupts
                        0505  2021                          FUNC=#CHC$_ENINT,-
                        0505  2022                          VECADR=BR_SERVICE,-
                        0505  2023                          STSADR=L_UBI_STATUS
     000000AF'EF    7F  0505                      PUSHAQ  L_UBI_STATUS
     00001A3C'EF    DF  050B                      PUSHAL  BR_SERVICE
     00000000'8F    DD  0511                      PUSHL   #CHC$_ENINT
     00000000'EF    DD  0517                      PUSHL   BE0_UNIT_NO
00010180 9F   04    FB  051D                      CALLS   #4, @#DS$CHANNEL
                        0524  2024          $DS_BNERROR     5$                      ; br = no error
              50    DD  0524                      PUSHL   R0
              02    DD  0526                      PUSHL   #$ER
     00000000'8F    DD  0528                      PUSHL   #DS$K_BNERROR
000100A8 9F   03    FB  052E                      CALLS   #3, @#DS$BRANCH
           1E 50    E8  0535                      BLBS    R0, 5$
                        0538  2025          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report channel error
                        0538  2026                          MSGADR=T_CHANNEL_ERR,-
                        0538  2027                          PRLINK=CHANNEL_ERR
     000016CC'EF    DF  0538                      PUSHAL  CHANNEL_ERR
     000012FF'EF    DF  053E                      PUSHAL  T_CHANNEL_ERR
     00000000'EF    DD  0544                      PUSHL   BE0_UNIT_NO
              02    DD  054A                      PUSHL   #$ER
                        054C        ;;; TEST 8, SUBTEST 6, ERROR 2
   000100D0 9F   04    FB  054C                      CALLS   #$$M, @#DS$ERRHARD
                        0553  2028          $DS_ABORT       TEST                    ; abort test
              50    D4  0553                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                    04  0555                      RET                     ; TERMINATE TEST
                        0556  2029
        37  00    DA  0556  2030 5$:      MTPR    #0,#PR$_UB_INIT                 ; initialize Unibus devices
59 00000307'EF    3E  0559  2031          MOVAW   UBE_BR7_INTR,R9                 ; load command table address
   000000AB'EF    07  D0  0560  2032          MOVL    #7,L_BR_LVL                 ; load initial BR intr level
   000000A7'EF    17  D0  0567  2033          MOVL    #^X17,L_IPL                 ; load initial prio level
                      056E  2034          $DS_SETIPL_S    LEVEL=L_IPL             ; set IPL at 17
     000000A7'EF    DD  056E                      PUSHL   L_IPL
   00010178 9F    01  FB  0574                      CALLS   #1, @#DS$SETIPL
                      057B  2035 ;
                      057B  2036 ; If Ready bit not set then report error and abort test
                      057B  2037 ;
     00000000'EF    B4  057B  2038 10$:     CLRW    W_FLAGS                 ; clear flags word
1E 000002BF'FF    07  E0  0581  2039          BBS     #V_RDY,@A_BE0_CR1,20$           ; br = ready bit set
                      0589  2040          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-
                      0589  2041                          MSGADR=T_INTR_ERR,-
                      0589  2042                          PRLINK=XFER_ERR
     00001420'EF    DF  0589                      PUSHAL  XFER_ERR
     000007DF'EF    DF  058F                      PUSHAL  T_INTR_ERR
     00000000'EF    DD  0595                      PUSHL   BE0_UNIT_NO
              03    DD  059B                      PUSHL   #$ER
                        059D        ;;; TEST 8, SUBTEST 6, ERROR 3
   000100D0 9F   04    FB  059D                      CALLS   #$$M, @#DS$ERRHARD
                        05A4  2043
                        05A4  2044          $DS_ABORT       TEST                    ; abort test
              50    D4  05A4                      CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
```

```
                          04  05A6                        RET                         ; TERMINATE TEST
                              05A7  2045 ;
                              05A7  2046 ; Check that interrupt does not occur when IPL should prohibit
                              05A7  2047 ;
            00001850'EF   16  05A7  2048 20$:    JSB     UBE_SET_UP                  ; load up UBE registers
                              05AD  2049         $DS_WAITMS_S    TIME=#1             ; wait 1 msec
                        00    DD  05AD                    PUSHL   #0
                        01    DD  05AF                    PUSHL   #1
                00010060 9F   02  FB  05B1                CALLS   #2, @#DS$WAITMS
          21 00000000'EF   03   E1  05B8  2050         BBC     #V_BR_INTR,W_FLAGS,40$  ; br = no unexpected interrupt
                              05C0  2051
                              05C0  2052         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-   ; report unexpected interrupt
                              05C0  2053                         MSGADR=T_INTR_ERR,-
                              05C0  2054                         PRLINK=INTR_ERR
            0000166E'EF       DF  05C0                    PUSHAL  INTR_ERR
            000007DF'EF       DF  05C6                    PUSHAL  T_INTR_ERR
            00000000'EF       DD  05CC                    PUSHL   BE0_UNIT_NO
                        04    DD  05D2                    PUSHL   #$ER
                              05D4        ;;; TEST 8, SUBTEST 6, ERROR 4
                00100D0 9F    04  FB  05D4                CALLS   #$$M, @#DS$ERRHARD
                              05DB  2055
            00000000'EF       B4  05DB  2056         CLRW            W_FLAGS         ; clear flags word
                              05E1  2057 ;
                              05E1  2058 ; Decrement IPL to allow the interrupt and check that it occurs with the correct
                              05E1  2059 ; vector and at the correct IPL
                              05E1  2060 ;
            000000A7'EF       D7  05E1  2061 40$:    DECL    L_IPL                   ; decrement IPL
                              05E7  2062         $DS_SETIPL_S    LEVEL=L_IPL         ; set IPL to allow interrupt
            000000A7'EF       DD  05E7                    PUSHL   L_IPL
                00010178 9F   01  FB  05ED                CALLS   #1, @#DS$SETIPL
                              05F4  2063
            00000000'EF       02  C8  05F4  2064 50$:    BISL2   #M_EXP_BR_INTR,W_FLAGS  ; set expect interrupt flag
                              05FB  2065         $DS_WAITMS_S    TIME=#1             ; wait 1 msec
                        00    DD  05FB                    PUSHL   #0
                        01    DD  05FD                    PUSHL   #1
                00010060 9F   02  FB  05FF                CALLS   #2, @#DS$WAITMS
          08 00000000'EF   03   E1  0606  2066         BBC     #V_BR_INTR,W_FLAGS,60$  ; br = no interrupt occurred
          21 00000000'EF   04   E1  060E  2067         BBC     #V_INTR_ERR,W_FLAGS,70$ ; br = vector & BR level okay
                              0616  2068
                              0616  2069 60$:    $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-
                              0616  2070                         MSGADR=T_INTR_ERR,-
                              0616  2071                         PRLINK=INTR_ERR
                  0000166E'EF   DF  0616                    PUSHAL  INTR_ERR
                  000007DF'EF   DF  061C                    PUSHAL  T_INTR_ERR
                  00000000'EF   DD  0622                    PUSHL   BE0_UNIT_NO
                        05    DD  0628                    PUSHL   #$ER
                              062A        ;;; TEST 8, SUBTEST 6, ERROR 5
                00100D0 9F    04  FB  062A                CALLS   #$$M, @#DS$ERRHARD
                              0631  2072
                  00000000'EF   B4  0631  2073         CLRW            W_FLAGS         ; clear flags word
                              0637  2074 ;
                              0637  2075 ; If all levels tested then branch out of loop
                              0637  2076 ;
                  000000AB'EF   D7  0637  2077 70$:    DECL    L_BR_LVL                ; decr BR level
          03      000000AB'EF   D1  063D  2078         CMPL    L_BR_LVL,#3             ; all 4 intr levels tested?
                        03    13  0644  2079         BEQL    90$                     ; br = all 4 levels tested
                      FF32    31  0646  2080         BRW     10$                     ; go to beginning of loop
```

```
                                0649  2081 90$:    $DS_SETIPL_S    LEVEL=#0                ; restore IPL to 0
                          00    DD  0649                 PUSHL   #0
              00010178 9F 01    FB  064B                 CALLS   #1, @#DS$SETIPL
                                0652  2082
                                0652  2083          $DS_ENDSUB
                                0652        T8_S6_X:
     00010038 9F   0000003C'EF    FA  0652                 CALLG   $S$, @#DS$ENDSUB
                                065D  2084
                                065D  2085          $DS_PAGE
```

```
                                065D                    .SBTTL  SUBTEST 8.7:  DATI 1
                                065D  2088 ;+
                                065D  2089 ; Subtest description:
                                065D  2090 ;
                                065D  2091 ;       This subtest checks the ability to execute a Unibus read (DATI)
                                065D  2092 ;       transaction.  The UBE Data register is cleared with a CPU write
                                065D  2093 ;       and a unique data pattern is written to a main memory location.
                                065D  2094 ;       All usable Map Registers are validated and set up to access the main
                                065D  2095 ;       memory location.  A DATI is executed and after a nominal wait, the
                                065D  2096 ;       UBE Data register is read.  Transfers are made from longword- and
                                065D  2097 ;       word-aligned locations.
                                065D  2098 ;
                                065D  2099 ;
                                065D  2100 ; Subtest steps:
                                065D  2101 ;
                                065D  2102 ;       SUBTEST DATI 1
                                065D  2103 ;       :  Size UNIBUS
                                065D  2104 ;       :  IF no usable map entries
                                065D  2105 ;       :  :  THEN DO
                                065D  2106 ;       :  :  :  PRINT exit warning
                                065D  2107 ;       :  :  :  EXIT test
                                065D  2108 ;       :  :  ENDDO
                                065D  2109 ;       :  ENDIF
                                065D  2110 ;       :  Validate and point all usable Map Entries to main memory address
                                065D  2111 ;       :  DO FOR Alignment = (word, longword)
                                065D  2112 ;       :  :  Load UBE address registers
                                065D  2113 ;       :  :  Execute DATI
                                065D  2114 ;       :  :  Wait
                                065D  2115 ;       :  :  IF Error
                                065D  2116 ;       :  :  :  THEN REPORT Error
                                065D  2117 ;       :  :  ENDIF
                                065D  2118 ;       :  ENDDO
                                065D  2119 ;       ENDSUBTEST DATI 1
                                065D  2120 ;
                                065D  2121 ; Errors:
                                065D  2122 ;
                                065D  2123 ;-
                                065D  2124          $DS_BGNSUB
                                065D        T8_S7::
00010030 9F   00000048'EF   FA  065D                    CALLG   $S$, a#DS$BGNSUB
                                0668  2125
                                0668  2126          $DS_CHANNEL_S   UNIT=BE0_UNIT_NO,-       ; enable interrupts
                                0668  2127                          FUNC=#CHC$_ENINT,-
                                0668  2128                          VECADR=INTR_SERVICE,-
                                0668  2129                          STSADR=L_UBI_STATUS
          000000AF'EF   7F      0668                      PUSHAQ  L_UBI_STATUS
          00001A6C'EF   DF      066E                      PUSHAL  INTR_SERVICE
          00000000'8F   DD      0674                      PUSHL   #CHC$_ENINT
          00000000'EF   DD      067A                      PUSHL   BE0_UNIT_NO
      00010180 9F   04  FB      0680                      CALLS   #4, a#DS$CHANNEL
                                0687  2130          $DS_BNERROR     0$                         ; br = no error
                    50  DD      0687                      PUSHL   R0
                    01  DD      0689                      PUSHL   #$ER
          00000000'8F   DD      068B                      PUSHL   #DS$K_BNERROR
      000100A8 9F   03  FB      0691                      CALLS   #3, a#DS$BRANCH
                1E  50  E8      0698                      BLBS    R0, 0$
                                069B  2131          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report channel error
```

```
                                 069B  2132                    MSGADR=T_CHANNEL_ERR,-
                                 069B  2133                    PRLINK=CHANNEL_ERR
                000016CC'EF      DF   069B                    PUSHAL  CHANNEL_ERR
                000012FF'EF      DF   06A1                    PUSHAL  T_CHANNEL_ERR
                00000000'EF      DD   06A7                    PUSHL   BE0_UNIT_NO
                          01     DD   06AD                    PUSHL   #$ER
                                 06AF        ;;; TEST 8, SUBTEST 7, ERROR 1
            000100D0 9F   04     FB   06AF                    CALLS   #$$M, @#DS$ERRHARD
                                 06B6  2134          $DS_ABORT       TEST                        ; abort test
                          50     D4   06B6                    CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                                 04   06B8                    RET                         ; TERMINATE TEST
                                 06B9  2135
                00000000'EF      B4   06B9  2136 0$:      CLRW    W_FLAGS                         ; clear flags word
                      37   00    DA   06BF  2137          MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
                00001867'EF      16   06C2  2138          JSB     UNIBUS_SIZER                    ; size Unibus address space
                00000000'FF      D4   06C8  2139          CLRL    @A_UBE_BUFFER_0                 ; clear buffer longword
            56  00000000'E      D0   06CE  2140          MOVL    A_UBE_BUFFER_0,R6               ; load adr of buffer
                  6F     1       D0   06D5  2141          MOVL    #1,(R6)                         ; load data pattern
    0000004A'EF   56   CF   C9   EF   06D8  2142          EXTZV   #9,#15,R6,L_PAGE_NO             ; extract Page Frame Number
                0000192A'E      16   06E1  2143          JSB     ALL_MAPS
                                 06E7  2144 ;
                                 06E7  2145 ; Exit test if no Map Entries are usable since all the remaining subtests
                                 06E7  2146 ; depend upon the availability of usable map registers.
                                 06E7  2147 ;
                00000002'EF      D5   06E7  2148          TSTL    FRST_VAL_MAP                    ; was valid map found ?
                          10     18   06ED  2149          BGEQ    1$                              ; br = valid map found
                                 06EF  2150          $DS_PRINTB_S    FORMAT=T_NO_MAPS            ; print exit warning
                00000437'EF      9F   06EF                    PUSHAB  T_NO_MAPS
            000100E0 9F   01     FB   06F5                    CALLS   #$$N, @#DS$PRINTB
                                 06FC  2151          $DS_EXIT        TEST                        ; exit test
                        1299     31   06FC                    BRW     TEST_008_X      ; EXIT TEST 8
                                 06FF  2152 ;
                                 06FF  2153 ; Load address into DATI command table.
                                 06FF  2154 ;
                          58     D4   06FF  2155 1$:      CLRL    R8                              ; clear device number
                59  000002CB'EF      DE   0701  2156          MOVAL   UBE_DATI_1,R9                   ; load command table address
                     06 A9   56     B0   0708  2157          MOVW    R6,6(R9)                        ; load offset within page
06 A9   09   09  00000002'EF      F0   070C  2158          INSV    FRST_VAL_MAP,#9,#9,6(R9)        ; load Map Number
                                 0716  2159 ;
                                 0716  2160 ; If ready bit not set then report error and abort test
                                 0716  2161 ;
            25 000002BF'FF       07   E0   0716  2162 3$:      BBS     #V_RDY,@A_BE0_CR1,5$            ; br = ready bit set
                                 071E  2163
    0000004E'EF   00F26010 9F    D0   071E  2164          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                 0729  2165          $DS_ERRHARD_S   MSGADR=T_DATI_ERR,-      ; report no-RDY error
                                 0729  2166                          PRLINK=XFER_ERR
                00001420'EF      DF   0729                    PUSHAL  XFER_ERR
                0000056C'EF      DF   072F                    PUSHAL  T_DATI_ERR
                          00     DD   0735                    PUSHL   #0
                          02     DD   0737                    PUSHL   #$ER
                                 0739        ;;; TEST 8, SUBTEST 7, ERROR 2
            000100D0 9F   04     FB   0739                    CALLS   #$$M, @#DS$ERRHARD
                                 0740  2167
                                 0740  2168          $DS_ABORT       TEST                        ; abort test
                          50     D4   0740                    CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                                 04   0742                    RET                         ; TERMINATE TEST
                                 0743  2169 ;
```

```
                                0743  2170 ; Execute DATI
                                0743  2171 ;
                00001850'EF   16  0743  2172 5$:     JSB     UBE_SET_UP
                                0749  2173         $DS_WAITMS_S    TIME=#1                 ; wait 1 msec
                          00    DD  0749                 PUSHL   #0
                          01    DD  074B                 PUSHL   #1
              00010060 9F     02  FB  074D                 CALLS   #2, @#DS$WAITMS
                                0754  2174 ;
                                0754  2175 ; Check for data or execution errors and report if any
                                0754  2176 ;
     0000004E'EF    00F26010 9F  D0  0754  2177         MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
              01    000002AF'FF  B1  075F  2178         CMPW    @A_BE0_DB,#1                    ; check UBE data register
                          12    13  0766  2179         BEQL    6$                              ; br = no data error
              52    000002AF'FF  3C  0768  2180         MOVZWL  @A_BE0_DB,R2                    ; extend actual data
        00000000'EF     1000 8F  A8  076F  2181         BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
                          15    11  0778  2182         BRB     7$                              ; br to error report
          0D 000002BF'FF    0F  E0  077A  2183 6$:     BBS     #V_ERR,@A_BE0_CR1,7$            ; br = UBE shows error
     0000004E'EF    8001C000 8F  D3  0782  2184         BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI CSR status bits
                          37    13  078D  2185         BEQL    9$                              ; br = no UBI error
                                078F  2186
                                078F  2187 7$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                078F  2188                         MSGADR=T_DATI_ERR,-
                                078F  2189                         PRLINK=XFER_ERR,-
                                078F  2190                         P1=A_BE0_DB,-            ; location
                                078F  2191                         P2=#T,-                  ; expected value
                                078F  2192                         P3=R2,-                  ; actual value
                                078F  2193                         P4=#T_WORD_ERR,-         ; data type
                                078F  2194                         P5=FRST_VAL_MAP          ; map number used
                00000002'EF     DD  078F                 PUSHL   FRST_VAL_MAP
                00000DA0'8F     DD  0795                 PUSHL   #T_WORD_ERR
                          52    DD  079B                 PUSHL   R2
                          01    DD  079D                 PUSHL   #1
                000002AF'EF     DD  079F                 PUSHL   A_BE0_DB
                00001420'EF     DF  07A5                 PUSHAL  XFER_ERR
                0000056C'EF     DF  07AB                 PUSHAL  T_DATI_ERR
                00000000'EF     DD  07B1                 PUSHL   BE0_UNIT_NO
                          03    DD  07B7                 PUSHL   #$ER
                                07B9         ;;; TEST 8, SUBTEST 7, ERROR 3
            00010DD0 9F     09  FB  07B9                 CALLS   #$$M, @#DS$ERRHARD
                                07C0  2195
                00000000'EF     B4  07C0  2196         CLRW    W_FLAGS                         ; clear flags word
                                07C6  2197 9$:     $DS_CKLOOP      1$                      ; scope loop at 1$
     00010040 9F    FF35 CF     FA  07C6                 CALLG   1$, @#DS$CKLOOP
                                07CF  2198
                 06 56    01    E2  07CF  2199         BBSS    #1,R6,10$                       ; br = done for longword & word
                    66    01    D0  07D3  2200         MOVL    #1,(R6)                         ; load data in buffer (wd-algnd)
                       FF26    31  07D6  2201         BRW     1$                              ; br, do for word alignment
                                07D9  2202 10$:
                                07D9  2203         $DS_ENDSUB
                                07D9         T8_S7_X:
 00010038 9F    00000048'EF     FA  07D9                 CALLG   $S$, @#DS$ENDSUB
                                07E4  2204
                                07E4  2205         $DS_PAGE
```

```
                                07E4                    .SBTTL  SUBTEST 8.8:  Map Select
                                07E4  2208 ;+
                                07E4  2209 ; Subtest description:
                                07E4  2210 ;
                                07E4  2211 ;       This subtest checks the ability of the UBI to correctly select
                                07E4  2212 ;       a map from the page frame of the unibus address. The type of fault that
                                07E4  2213 ;       could occur here is a stuck or shorted Unibus address bit, which would
                                07E4  2214 ;       cause the wrong map entry to be selected. In order to detect this type
                                07E4  2215 ;       of fault, the following steps are executed:
                                07E4  2216 ;
                                07E4  2217 ; Subtest steps:
                                07E4  2218 ;
                                07E4  2219 ;       SUBTEST MAP SELECT
                                07E4  2220 ;       :  Invalidate all Maps
                                07E4  2221 ;       :  DO FOR Map Number = {0,1,2,4,8,16,32,64,128,256}
                                07E4  2222 ;       :  :  IF Map Entry usable
                                07E4  2223 ;       :  :  :  THEN DO
                                07E4  2224 ;       :  :  :  :  VALIDATE Map Entry
                                07E4  2225 ;       :  :  :  :  CLEAR UBE Data register
                                07E4  2226 ;       :  :  :  :  EXECUTE DATI
                                07E4  2227 ;       :  :  :  :  WAIT
                                07E4  2228 ;       :  :  :  :  IF error
                                07E4  2229 ;       :  :  :  :  :  THEN REPORT error
                                07E4  2230 ;       :  :  :  :  ENDIF
                                07E4  2231 ;       :  :  :  :  INVALIDATE Map Entry
                                07E4  2232 ;       :  :  :  ENDDO
                                07E4  2233 ;       :  :  ENDIF
                                07E4  2234 ;       :  ENDDO
                                07E4  2235 ;       ENDSUBTEST MAP SELECT
                                07E4  2236 ;
                                07E4  2237 ; Errors:
                                07E4  2238 ;
                                07E4  2239 ;-
                                07E4  2240              $DS_BGNSUB
                                07E4            T8_S8::
00010030 9F   00000054'EF   FA  07E4                    CALLG   $S$, @#DS$BGNSUB
                                07EF  2241
              00000000'EF   B4  07EF  2242              CLRW    W_FLAGS                         ; clear flags word
                  37   00   DA  07F5  2243              MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
              000018FB'EF   16  07F8  2244              JSB     INVAL_MAPS                      ; invalidate all Map Entries
          00000000'FF   01   D0  07FE  2245              MOVL    #1,@A_UBE_BUFFER_0              ; load data into buffer
          56  00000000'EF   D0  0805  2246              MOVL    A_UBE_BUFFER_0,R6               ; load memory loc address
0000004A'EF   56   0F   09   EF  080C  2247              EXTZV   #9,#15,R6,L_PAGE_NO             ; extract page frame number
                                0815  2248 ;
                                0815  2249 ; If map is usable then: load address into DATI command table, load Map Entry
                                0815  2250 ; PFN, validate Map Entry
                                0815  2251 ;
                    54   D4  0815  2252              CLRL    R4                              ; clear index
       03 00000006'EF    54   E0  0817  2253 1$:      BBS     R4,V_VALID_MAPS,2$              ; br = usable Map Entry
                 00D9   31  081F  2254              BRW     7$                              ; br = not usable Map Entry
          59   000002CB'EF   DE  0822  2255 2$:      MOVAL   UBE_DATI_1,R9                   ; load command table address
               06 A9   56   B0  0829  2256              MOVW    R6,6(R9)                        ; load offset within page
       06 A9   09   09   54   F0  082D  2257              INSV    R4,#9,#9,6(R9)                  ; load Map Number
00F26800 9F44   0000004A'EF   D0  0833  2258              MOVL    L_PAGE_NO,@#A_MAP_BGN[R4]       ; load Map Entry PFN
00F26800 9F44   80000000 8F   C8  083F  2259              BISL2   #M_MAP_VAL,@#A_MAP_BGN[R4]       ; set Map Entry Valid bit
                                084B  2260 ;
                                084B  2261 ; If ready bit not set then report error and abort test
```

```
                                 084B  2262 ;
          29 000002BF'FF    07   E0  084B  2263 3$:     BBS     #V_RDY,@A_BE0_CR1,4$            ; br = ready bit set
                                 0853  2264
     0000004E'EF    00F26010 9F   D0  0853  2265         MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                 085E  2266         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report no-RDY error
                                 085E  2267                         MSGADR=T_MAP_SEL_ERR,-
                                 085E  2268                         PRLINK=XFER_ERR
                00001420'EF      DF  085E                        PUSHAL  XFER_ERR
                000005B0'EF      DF  0864                        PUSHAL  T_MAP_SEL_ERR
                00000000'EF      DD  086A                        PUSHL   BE0_UNIT_NO
                          01     DD  0870                        PUSHL   #$ER
                                 0872       ;;; TEST 8, SUBTEST 8, ERROR 1
           000100D0 9F    04    FB  0872                        CALLS   #$$M, @#DS$ERRHARD
                                 0879  2269
                                 0879  2270         $DS_ABORT       TEST                        ; abort test
                          50     D4  0879                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                 04  087B                        RET                     ; TERMINATE TEST
                                 087C  2271 ;
                                 087C  2272 ; Initiate DATI transaction, wait
                                 087C  2273 ;
                00001850'EF      16  087C  2274 4$:     JSB     UBE_SET_UP                      ; execute DATI
                                 0882  2275         $DS_WAITMS_S    TIME=#1                     ; wait
                          00     DD  0882                        PUSHL   #0
                          01     DD  0884                        PUSHL   #1
           00010060 9F    02    FB  0886                        CALLS   #2, @#DS$WAITMS
                                 088D  2276 ;
                                 088D  2277 ; Check for execution or data errors, report if any
                                 088D  2278 ;
     0000004E'EF    00F26010 9F   D0  088D  2279         MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
          01    000002AF'FF    B1  0898  2280         CMPW    @A_BE0_DB,#1                    ; check UBE Data register
                          12     13  089F  2281         BEQL    5$                              ; br = no data error
        00000000'EF    1000 8F   A8  08A1  2282         BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
          52    000002AF'FF    3C  08AA  2283         MOVZWL  @A_BE0_DB,R2                    ; extend actual data
                          15     11  08B1  2284         BRB     6$                              ; br to error report
     0000004E'EF    8001C000 8F   D3  08B3  2285 5$:     BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status
                          08     12  08BE  2286         BNEQ    6$                              ; br = UBI error
          33 000002BF'FF    0F   E1  08C0  2287         BBC     #V_ERR,@A_BE0_CR1,7$            ; br = no UBE error
                                 08C8  2288
                                 08C8  2289 6$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                 08C8  2290                         MSGADR=T_MAP_SEL_ERR,-
                                 08C8  2291                         PRLINK=XFER_ERR,-
                                 08C8  2292                         P1=A_BE0_DB,-                ; location
                                 08C8  2293                         P2=#T,-                      ; expected value
                                 08C8  2294                         P3=R2,-                      ; actual value
                                 08C8  2295                         P4=#T_WORD_ERR,-             ; data type
                                 08C8  2296                         P5=R4                        ; map number used
                          54     DD  08C8                        PUSHL   R4
                00000DA0'8F      DD  08CA                        PUSHL   #T_WORD_ERR
                          52     DD  08D0                        PUSHL   R2
                          01     DD  08D2                        PUSHL   #1
                000002AF'EF      DD  08D4                        PUSHL   A_BE0_DB
                00001420'EF      DF  08DA                        PUSHAL  XFER_ERR
                000005B0'EF      DF  08E0                        PUSHAL  T_MAP_SEL_ERR
                00000000'EF      DD  08E6                        PUSHL   BE0_UNIT_NO
                          02     DD  08EC                        PUSHL   #$ER
                                 08EE       ;;; TEST 8, SUBTEST 8, ERROR 2
           000100D0 9F    09    FB  08EE                        CALLS   #$$M, @#DS$ERRHARD
```

```
                                08F5  2297
                00000000'EF   B4  08F5  2298          CLRW    W_FLAGS                         ; clear flags word
                                08FB  2299 7$:        $DS_CKLOOP      2$                      ; scope loop at 2$
        00010040 9F    FF23 CF   FA  08FB                  CALLG   2$, @#DS$CKLOOP
                                0904  2300
              00F26800 9F44   D4  0904  2301          CLRL    @#A_MAP_BGN[R4]                 ; invalidate Map Entry
               54   54    01   78  090B  2302          ASHL    #1,R4,R4                        ; float one to next address bit
                          02   12  090F  2303          BNEQ    9$                              ; br = index not 0
                          54   D6  0911  2304          INCL    R4                              ; load one into index
             00000200 8F   54   D1  0913  2305 9$:     CMPL    R4,#512                         ; done ?
                          03   13  091A  2306          BEQL    10$                             ; br = done
                       FEF8   31  091C  2307          BRW     1$                              ; br = not done
                                091F  2308 10$:
                                091F  2309          $DS_ENDSUB
                                091F        T8_S8_X:
    00010038 9F    00000054'EF   FA  091F                  CALLG   $S$, @#DS$ENDSUB
                                092A  2310
                                092A  2311          $DS_PAGE
                                092A  2312

                                092A                .SBTTL  SUBTEST 8.9:  DATI 2
                                092A  2314 ;+
                                092A  2315 ; Subtest description:
                                092A  2316 ;
                                092A  2317 ;        This subtest checks the ability of the UBI to perform a DATI
                                092A  2318 ;        transaction and the integrity of the data path.  The transactions are
                                092A  2319 ;        made from longword and word aligned addresses.  The following data
                                092A  2320 ;        patterns are used:
                                092A  2321 ;
                                092A  2322 ;                0101010101010101
                                092A  2323 ;                1010101010101010
                                092A  2324 ;                0011001100110011
                                092A  2325 ;                0000111100001111
                                092A  2326 ;                0000000011111111
                                092A  2327 ;
                                092A  2328 ; Subtest steps:
                                092A  2329 ;
                                092A  2330 ;        SUBTEST DATI 2
                                092A  2331 ;        :  Invalidate all maps but one
                                092A  2332 ;        :  DO for Alignment = {longword,word}
                                092A  2333 ;        :  :  REPEAT
                                092A  2334 ;        :  :  :  EXECUTE DATI
                                092A  2335 ;        :  :  :  WAIT
                                092A  2336 ;        :  :  :  IF error
                                092A  2337 ;        :  :  :  :  THEN REPORT error
                                092A  2338 ;        :  :  :  ENDIF
                                092A  2339 ;        :  :  : UNTIL patterns exhausted
                                092A  2340 ;        :  :  ENDREPEAT
                                092A  2341 ;        :  ENDDO
                                092A  2342 ;        ENDSUBTEST DATI 2
                                092A  2343 ;
                                092A  2344 ; Errors:
                                092A  2345 ;
                                092A  2346 ;-
                                092A  2347          $DS_BGNSUB
                                092A        T8_S9::
    00010030 9F    00000060'EF   FA  092A                  CALLG   $S$, @#DS$BGNSUB
```

ZZ-ENKAX-1.3 SUBTEST 8.8: Map Select 0935 2348 I 2 3-AUG-1983 Fiche 3 Frame 12 Sequence 433

```
                    37   00    DA  0935  2349          MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
              00000000'EF    B4  0938  2350          CLRW    W_FLAGS                         ; clear flags word
              00000000'FF    D4  093E  2351          CLRL    @#A_UBE_BUFFER_0                ; clear buffer longword
                                 0944  2352 ;
                                 0944  2353 ; Invalidate all Map Entries except the first usable one.
                                 0944  2354 ;
              000018FB'EF    16  0944  2355          JSB     INVAL_MAPS                      ; invalidate all maps
           52 00000002'EF    D0  094A  2356          MOVL    FRST_VAL_MAP,R2                 ; load No of first valid Map
 00F26800 9F42 0000004A'EF    D0  0951  2357          MOVL    L_PAGE_NO,@#A_MAP_BGN[R2]       ; load PFN into Map Entry
 00F26800 9F42 80000000 8F    C8  095D  2358          BISL2   #M_MAP_VAL,@#A_MAP_BGN[R2]      ; validate Map Entry
                                 0969  2359
           55 0000029F'EF    3E  0969  2360 10$:     MOVAW   W_PTRN_TBL,R5                   ; load pattern table address
                                 0970  2361 ;
                                 0970  2362 ; Load address into DATI command table
                                 0970  2363 ;
           59 000002CB'EF    3E  0970  2364 30$:     MOVAW   UBE_DATI_1,R9                   ; load command table address
                 06 A9   56  B0  0977  2365          MOVW    R6,6(R9)                        ; load offset within page
 06 A9   09   09 00000002'EF    F0  097B  2366          INSV    FRST_VAL_MAP,#9,#9,6(R9)        ; load Map Number
                                 0985  2367
                                 0985  2368 ;
                                 0985  2369 ; If ready bit not set then report error and abort test
                                 0985  2370 ;
        29 000002BF'FF   07      E0  0985  2371 40$:     BBS     #V_RDY,@A_BE0_CR1,50$           ; br = ready bit set
                                 098D  2372
    0000004E'EF 00F26010 9F      D0  098D  2373          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                 0998  2374          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report no-RDY error
                                 0998  2375                          MSGADR=T_D_PATH_ERR,-
                                 0998  2376                          PRLINK=XFER_ERR
              00001420'EF    DF  0998                PUSHAL  XFER_ERR
              0000058E'EF    DF  099E                PUSHAL  T_D_PATH_ERR
              00000000'EF    DD  09A4                PUSHL   BE0_UNIT_NO
                        01   DD  09AA                PUSHL   #$ER
                                 09AC        ;;; TEST 8, SUBTEST 9, ERROR 1
        00010DD0 9F     04   FB  09AC                CALLS   #$$M, @#DS$ERRHARD
                                 09B3  2377
                                 09B3  2378          $DS_ABORT       TEST                    ; abort test
                        50   D4  09B3                CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                        04   09B5                RET                             ; TERMINATE TEST
                                 09B6  2379 ;
                                 09B6  2380 ; Execute DATI, wait
                                 09B6  2381 ;
                     66  65  B0  09B6  2382 50$:     MOVW    (R5),(R6)                       ; load pattern
              00001850'EF    16  09B9  2383          JSB     UBE_SET_UP                      ; execute DATI
                                 09BF  2384          $DS_WAITMS_S    TIME=#1                 ; wait 1 msec
                        00   DD  09BF                PUSHL   #0
                        01   DD  09C1                PUSHL   #1
        00010060 9F     02   FB  09C3                CALLS   #2, @#DS$WAITMS
                                 09CA  2385 ;
                                 09CA  2386 ; Check for execution errors, report if any
                                 09CA  2387 ;
0000004E'EF 00F26010 9F      D0  09CA  2388          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
        65 000002AF'FF       B1  09D5  2389          CMPW    @A_BE0_DB,(R5)                  ; check UBE data register
                        12   13  09DC  2390          BEQL    60$                             ; br = no data error
     00000000'EF   1000 8F   A8  09DE  2391          BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
           52 000002AF'FF    3C  09E7  2392          MOVZWL  @A_BE0_DB,R2                    ; extend actual data
                        15   11  09EE  2393          BRB     70$                             ; br to error report
0000004E'EF 8001C000 8F      D3  09F0  2394 60$:     BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status bits
```

```
                      08   12  09FB  2395          BNEQ    70$                                ; br = UBI error
          37 000002BF'FF   0F   E1  09FD  2396          BBC     #V_ERR,@A_BE0_CR1,75$              ; br = no UBE errors
                               0A05  2397
                               0A05  2398 70$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-          ; report data error
                               0A05  2399                          MSGADR=T_D_PATH_ERR,-
                               0A05  2400                          PRLINK=XFER_ERR,-
                               0A05  2401                          P1=A_BE0_DB,-              ; location
                               0A05  2402                          P2=(R5),-                  ; expected value
                               0A05  2403                          P3=R2,-                    ; actual value
                               0A05  2404                          P4=#T_WORD_ERR,-           ; data type
                               0A05  2405                          P5=FRST_VAL_MAP            ; map number used
               00000002'EF     DD  0A05                          PUSHL   FRST_VAL_MAP
               00000DA0'8F     DD  0A08                          PUSHL   #T_WORD_ERR
                         52    DD  0A11                          PUSHL   R2
                         65    DD  0A13                          PUSHL   (R5)
               000002AF'EF     DD  0A15                          PUSHL   A_BE0_DB
               00001420'EF     DF  0A18                          PUSHAL  XFER_ERR
               0000058E'EF     DF  0A21                          PUSHAL  T_D_PATH_ERR
               00000000'EF     DD  0A27                          PUSHL   BE0_UNIT_NO
                         02    DD  0A2D                          PUSHL   #$ER
                               0A2F          ;;; TEST 8, SUBTEST 9, ERROR 2
                               0A2F                          CALLS   #$$M, @#DS$ERRHARD
        000100D0 9F    09   FB  0A2F
                               0A36  2406
               00000000'EF     B4  0A36  2407          CLRW    W_FLAGS                            ; clear flags word
                               0A3C  2408 75$:     $DS_CKLOOP      30$                        ; scope loop at 30$
      00010040 9F    FF30 CF   FA  0A3C                          CALLG   30$, @#DS$CKLOOP
                               0A45  2409 ;
                               0A45  2410 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                               0A45  2411 ; word-aligned cases have been tested.
                               0A45  2412 ;
               00FF 8F   85    B1  0A45  2413 80$:     CMPW    (R5)+,#^X00FF                  ; patterns exhausted?
                         03    13  0A4A  2414          BEQL    90$                            ; br = patterns exhausted
                      FF21     31  0A4C  2415          BRW     30$                            ; br = patterns not exhausted
                 03 56   01    E2  0A4F  2416 90$:     BBSS    #1,R6,100$                     ; br = done for word alignment
                      FF13     31  0A53  2417          BRW     10$                            ; br, repeat for word alignment
                               0A56  2418 100$:
                               0A56  2419          $DS_ENDSUB
                               0A56          T8_S9_X:
  00010038 9F    00000060'EF   FA  0A56                          CALLG   $S$, @#DS$ENDSUB
                               0A61  2420
                               0A61  2421          $DS_PAGE
```

```
                                0A61                    .SBTTL  SUBTEST 8.10:  DATO
                                0A61  2424 ;+
                                0A61  2425 ; Subtest description:
                                0A61  2426 ;
                                0A61  2427 ;       This subtest checks the ability of the UBI to perform a DATO
                                0A61  2428 ;       transaction. This test is similar to the DATI 2 subtest, with
                                0A61  2429 ;       the data transfer in the opposite direction.
                                0A61  2430 ;
                                0A61  2431 ; Subtest steps:
                                0A61  2432 ;
                                0A61  2433 ;       SUBTEST DATO
                                0A61  2434 ;       :  Load UBE address registers
                                0A61  2435 ;       :  DO FOR Alignment = {word, longword}
                                0A61  2436 ;       :  :  REPEAT
                                0A61  2437 ;       :  :  :  EXECUTE DATO
                                0A61  2438 ;       :  :  :  WAIT
                                0A61  2439 ;       :  :  :  IF error
                                0A61  2440 ;       :  :  :  :  THEN REPORT error
                                0A61  2441 ;       :  :  :  ENDIF
                                0A61  2442 ;       :  :  : UNTIL patterns exhausted
                                0A61  2443 ;       :  :  ENDREPEAT
                                0A61  2444 ;       :  ENDDO
                                0A61  2445 ;       ENDSUBTEST DATO
                                0A61  2446 ;
                                0A61  2447 ; Errors:
                                0A61  2448 ;
                                0A61  2449 ;-
                                0A61  2450          $DS_BGNSUB
                                0A61        T8_S10::
 00010030 9F   0000006C'EF   FA 0A61                CALLG   $S$, @#DS$BGNSUB
                                0A6C  2451
                    37   00   DA 0A6C  2452          MTPR    #0,#PR$L_UB_INIT        ; initialize Unibus devices
              00000000'EF   B4 0A6F  2453          CLRW    W_FLAGS                 ; clear flags word
           56 00000000'EF   D0 0A75  2454          MOVL    A_UBE_BUFFER_0,R6       ; load memory loc address
              00000000'FF   D4 0A7C  2455          CLRL    @A_UBE_BUFFER_0         ; clear buffer longword
           55 0000029F'EF   3E 0A82  2456 0$:      MOVAW   W_PTRN_TBL,R5           ; load pattern table address
                                0A89  2457 ;
                                0A89  2458 ; Load address and data into command table
                                0A89  2459 ;
           0000005E'EF   65   B0 0A89  2460 10$:    MOVW    (R5),L_EXP_DATA         ; load pattern
           59 000002EF'EF   DE 0A90  2461 15$:     MOVAL   UBE_DATO_1,R9           ; load command table address
        02 A9 0000005E'EF   B0 0A97  2462          MOVW    L_EXP_DATA,2(R9)        ; load data
              06 A9   56   B0 0A9F  2463          MOVW    R6,6(R9)                ; load offset within page
06 A9   09   09 00000002'EF   F0 0AA3  2464          INSV    FRST_VAL_MAP,#9,#9,6(R9) ; load Map Number
                        66   B4 0AAD  2465          CLRW    (R6)                    ; clear buffer
                                0AAF  2466
                                0AAF  2467 ;
                                0AAF  2468 ; If ready bit not set then report error and abort test
                                0AAF  2469 ;
        29 000002BF'FF   07   E0 0AAF  2470 20$:    BBS     #V_RDY,@A_BE0_CR1,30$   ; br = ready bit set
                                0AB7  2471
 0000004E'EF   00F26010 9F   D0 0AB7  2472          MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
                                0AC2  2473          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-  ; report no-RDY error
                                0AC2  2474                          MSGADR=T_DATO_ERR,-
                                0AC2  2475                          PRLINK=XFER_ERR
              00001420'EF   DF 0AC2                PUSHAL  XFER_ERR
              0000061E'EF   DF 0AC8                PUSHAL  T_DATO_ERR
```

```
                00000000'EF   DD  0ACE                        PUSHL   BE0_UNIT_NO
                          01  DD  0AD4                        PUSHL   #$ER
                              0AD6        ;;; TEST 8, SUBTEST 10, ERROR 1
              000100D0 9F     04  FB  0AD6                    CALLS   #$$M, @#DS$ERRHARD
                              0ADD  2476
                              0ADD  2477          $DS_ABORT       TEST                        ; abort
                          50  D4  0ADD                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04  0ADF                        RET                             ; TERMINATE TEST
                              0AE0  2478 ;
                              0AE0  2479 ; Execute DATO, wait
                              0AE0  2480 ;
                00001850'EF   16  0AE0  2481 30$:       JSB     UBE_SET_UP                  ; execute DATO
                              0AE6  2482          $DS_WAITMS_S    TIME=#1                     ; wait
                          00  DD  0AE6                        PUSHL   #0
                          01  DD  0AE8                        PUSHL   #1
              00010060 9F     02  FB  0AEA                    CALLS   #2, @#DS$WAITMS
                              0AF1  2483 ;
                              0AF1  2484 ; Check for data or execution errors, report if any
                              0AF1  2485 ;
     0000004E'EF    00F26010 9F     D0  0AF1  2486          MOVL    @#A_UBI_CSR,L_SAVE_CSR      ; save UBI CSR contents
              66    0000005E'EF     B1  0AFC  2487          CMPW    L_EXP_DATA,(R6)             ; check UBE data register
                              0B  13  0B03  2488          BEQL    40$                         ; br = no data error
          00000000'EF    1000 8F     A8  0B05  2489          BISW2   #M_DATA_ERR,W_FLAGS         ; set data error flag
                          15  11  0B0E  2490          BRB     50$                         ; br to error report
     0000004E'EF    8001C000 8F     D3  0B10  2491 40$:     BITL    #M_UBI_STS,L_SAVE_CSR       ; check UBI status bits
                          08  12  0B1B  2492          BNEQ    50$                         ; br = UBI error
           37 000002BF'FF     0F  E1  0B1D  2493          BBC     #V_ERR,@A_BE0_CR1,55$       ; br = no UBE errors
                              0B25  2494
                              0B25  2495 50$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-          ; report data error
                              0B25  2496                          MSGADR=T_DATO_ERR,-
                              0B25  2497                          PRLINK=XFER_ERR,-
                              0B25  2498                          P1=R6,-                     ; location
                              0B25  2499                          P2=L_EXP_DATA,-             ; expected value
                              0B25  2500                          P3=(R6),-                   ; actual value
                              0B25  2501                          P4=#T_WORD_ERR,-            ; data type
                              0B25  2502                          P5=FRST_VAL_MAP             ; map number used
            00000002'EF   DD  0B25                          PUSHL   FRST_VAL_MAP
            00000DA0'8F   DD  0B2B                          PUSHL   #T_WORD_ERR
                      66  DD  0B31                          PUSHL   (R6)
            0000005E'EF   DD  0B33                          PUSHL   L_EXP_DATA
                      56  DD  0B39                          PUSHL   R6
            00001420'EF   DF  0B3B                          PUSHAL  XFER_ERR
            0000061E'EF   DF  0B41                          PUSHAL  T_DATO_ERR
            00000000'EF   DD  0B47                          PUSHL   BE0_UNIT_NO
                      02  DD  0B4D                          PUSHL   #$ER
                              0B4F        ;;; TEST 8, SUBTEST 10, ERROR 2
          000100D0 9F     09  FB  0B4F                      CALLS   #$$M, @#DS$ERRHARD
                              0B56  2503
            00000000'EF   B4  0B56  2504          CLRW    W_FLAGS                             ; clear flags word
                              0B5C  2505 55$:     $DS_CKLOOP      15$                         ; scope loop at 15$
    00010040 9F     FF30 CF     FA  0B5C                      CALLG   15$, @#DS$CKLOOP
                              0B65  2506 ;
                              0B65  2507 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                              0B65  2508 ; word-aligned cases have been tested.
                              0B65  2509 ;
              0F0F 8F     85  B1  0B65  2510 60$:     CMPW    (R5)+,#^X0F0F                   ; patterns exhausted?
                          03  13  0B6A  2511          BEQL    80$                             ; br = patterns exhausted
```

```
                FF1A   31  0B6C  2512          BRW     10$                             ; br = patterns not exhausted
          03 56   01   E2  0B6F  2513 80$:     BBSS    #1,R6,100$                      ; br = done for word alignment
                FF0C   31  0B73  2514          BRW     0$                              ; br, repeat for word alignment
                           0B76  2515 100$:
                           0B76  2516
                           0B76  2517          $DS_ENDSUB
                           0B76        T8_S10_X:
00010038 9F   0000006C'EF   FA  0B76                      CALLG   $S$, @#DS$ENDSUB
                           0B81  2518
                           0B81  2519          $DS_PAGE
```

```
                                                .SBTTL  SUBTEST 8.11:  DATOB
                                        ;+
                                        ; Subtest description:
                                        ;
                                        ;       This subtest checks the ability of the UBI to perform a DATOB transac-
                                        ;       tion.  The transactions are made from longword- and word-aligned main
                                        ;       memory addresses.  The following data patterns are used:
                                        ;
                                        ;               01010101
                                        ;               10101010
                                        ;               00110011
                                        ;               00001111
                                        ;
                                        ; Subtest steps:
                                        ;
                                        ;       SUBTEST DATOB
                                        ;       :  DO FOR Alignment = (word, longword)
                                        ;       :  CLEAR Main Memory location
                                        ;       :  :  REPEAT
                                        ;       :  :  :  Load pattern into UBE Data Register
                                        ;       :  :  :  Initiate DATOB
                                        ;       :  :  :  WAIT
                                        ;       :  :  :  IF error
                                        ;       :  :  :  :  THEN REPORT error
                                        ;       :  :  :  ENDIF
                                        ;       :  :  : UNTIL patterns exhausted
                                        ;       :  :  ENDREPEAT
                                        ;       :  ENDDO
                                        ;       ENDSUBTEST DATOB
                                        ;
                                        ; Errors:
                                        ;
                                        ;-
                                                $DS_BGNSUB
                                        T8_S11::
00010030 9F   00000078'EF    FA  0B81           CALLG   $S$, @#DS$BGNSUB
                                    0B8C 2555
                37   00      DA  0B8C 2556      MTPR    #0,#PR$L_UB_INIT            ; initialize Unibus devices
          00000000'EF        B4  0B8F 2557      CLRW    W_FLAGS                     ; clear flags word
          00000000'FF        D4  0B95 2558      CLRL    @A_UBE_BUFFER_0             ; clear buffer longword
       56 00000000'EF        D0  0B98 2559      MOVL    A_UBE_BUFFER_0,R6           ; load memory loc address
                66           B4  0BA2 2560      CLRW    (R6)                        ; clear main memory location
          0000005E'EF        D4  0BA4 2561      CLRL    L_EXP_DATA                  ; clear expected data
       55 0000029F'EF        3E  0BAA 2562 10$: MOVAW   W_PTRN_TBL,R5               ; load pattern table address
                                    0BB1 2563 ;
                                    0BB1 2564 ; Load data and address into DATOB command table
                                    0BB1 2565 ;
          0000005E'EF   65      90  0BB1 2566 30$: MOVB    (R5),L_EXP_DATA             ; load expected data
                        66      B4  0BB8 2567      CLRW    (R6)                        ; clear destination
       59 000002E3'EF           DE  0BBA 2568      MOVAL   UBE_DATOB_1,R9              ; load command table address
             02 A9   65          B0  0BC1 2569      MOVW    (R5),2(R9)                  ; load data pattern
             06 A9   56          B0  0BC5 2570      MOVW    R6,6(R9)                    ; load offset within page
06 A9    09   09   00000002'EF   F0  0BC9 2571      INSV    FRST_VAL_MAP,#9,#9,6(R9)    ; load Map Number
                                    0BD3 2572 ;
                                    0BD3 2573 ; If ready bit not set then report error and abort test
                                    0BD3 2574 ;
         29 000002BF'FF    07    E0  0BD3 2575 40$: BBS     #V_RDY,@A_BE0_CR1,50$       ; br = ready bit set
```

```
                                        0BDB  2576
0000004E'EF    00F26010 9F    D0        0BDB  2577          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                        0BE6  2578          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report no-RDY error
                                        0BE6  2579                          MSGADR=T_DATOB_ERR,-
                                        0BE6  2580                          PRLINK=XFER_ERR
                00001420'EF    DF       0BE6                        PUSHAL  XFER_ERR
                000005FB'EF    DF       0BEC                        PUSHAL  T_DATOB_ERR
                00000000'EF    DD       0BF2                        PUSHL   BE0_UNIT_NO
                          01   DD       0BF8                        PUSHL   #$ER
                                        0BFA              ;;; TEST 8, SUBTEST 11, ERROR 1
             000100D0 9F       04    FB 0BFA                        CALLS   #$$M, @#DS$ERRHARD
                                        0C01  2581
                                        0C01  2582          $DS_ABORT       TEST                    ; abort test
                          50   D4       0C01                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                          04   0C03                        RET                                 ; TERMINATE TEST
                                        0C04  2583 ;
                                        0C04  2584 ; Execute DATOB, wait
                                        0C04  2585 ;
                00001850'EF    16       0C04  2586 50$:     JSB     UBE_SET_UP                      ; execute DATOB
                                        0C0A  2587          $DS_WAITMS_S    TIME=#1                 ; wait
                          00   DD       0C0A                        PUSHL   #0
                          01   DD       0C0C                        PUSHL   #1
             00010060 9F       02    FB 0C0E                        CALLS   #2, @#DS$WAITMS
                                        0C15  2588 ;
                                        0C15  2589 ; Check for data or execution errors, report if any.
                                        0C15  2590 ;
0000004E'EF    00F26010 9F    D0        0C15  2591          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
          66   0000005E'EF    91       0C20  2592          CMPB    L_EXP_DATA,(R6)                 ; check data sent
                          0B   13       0C27  2593          BEQL    60$                             ; br = no data error
      00000000'EF    1000 8F    A8     0C29  2594          BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
                          15   11       0C32  2595          BRB     70$                             ; br to error report
0000004E'EF    8001C000 8F    D3        0C34  2596 60$:     BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status bits
                          08   12       0C3F  2597          BNEQ    70$                             ; br = UBI error
       37 000002BF'FF    0F    E1      0C41  2598          BBC     #V_ERR,@A_BE0_CR1,75$           ; br = no UBE errors
                                        0C49  2599
                                        0C49  2600 70$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report data error
                                        0C49  2601                          MSGADR=T_DATOB_ERR,-
                                        0C49  2602                          PRLINK=XFER_ERR,-
                                        0C49  2603                          P1=R6,-                 ; location
                                        0C49  2604                          P2=L_EXP_DATA,-         ; expected value
                                        0C49  2605                          P3=(R6),-               ; actual value
                                        0C49  2606                          P4=#T_BYTE_ERR,-        ; data type
                                        0C49  2607                          P5=FRST_VAL_MAP         ; map number used
                00000002'EF    DD       0C49                        PUSHL   FRST_VAL_MAP
                00000D72'8F    DD       0C4F                        PUSHL   #T_BYTE_ERR
                          66   DD       0C55                        PUSHL   (R6)
                0000005E'EF    DD       0C57                        PUSHL   L_EXP_DATA
                          56   DD       0C5D                        PUSHL   R6
                00001420'EF    DF       0C5F                        PUSHAL  XFER_ERR
                000005FB'EF    DF       0C65                        PUSHAL  T_DATOB_ERR
                00000000'EF    DD       0C6B                        PUSHL   BE0_UNIT_NO
                          02   DD       0C71                        PUSHL   #$ER
                                        0C73              ;;; TEST 8, SUBTEST 11, ERROR 2
             000100D0 9F       09    FB 0C73                        CALLS   #$$M, @#DS$ERRHARD
                                        0C7A  2608
                00000000'EF    B4       0C7A  2609          CLRW    W_FLAGS                         ; clear flags word
                                        0C80  2610 75$:     $DS_CKLOOP      30$                     ; scope loop at 30$
```

```
00010040 9F   FF2D CF    FA  0C80                        CALLG   30$, @#DS$CKLOOP
                          0C89  2611 ;
                          0C89  2612 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                          0C89  2613 ; word-aligned cases have been tested.
                          0C89  2614 ;
              0F0F 8F    85   B1  0C89  2615 80$:    CMPW    (R5)+,#^X0F0F                   ; patterns exhausted ?
                           03   13  0C8E  2616        BEQL    90$                             ; br = patterns exhausted
                       FF1E   31  0C90  2617        BRW     30$                             ; br = patterns not exhausted
                 03 56   01   E2  0C93  2618 90$:    BBSS    #1,R6,100$                      ; br = done for word alignment
                       FF10   31  0C97  2619        BRW     10$                             ; br, repeat for word alignment
                          0C9A  2620 100$:
                          0C9A  2621        $DS_ENDSUB
                          0C9A       T8_S11_X:
00010038 9F   00000078'EF    FA  0C9A                        CALLG   $S$, @#DS$ENDSUB
                          0CA5  2622
                          0CA5  2623        $DS_PAGE
```

```
                                                        0CA5                    .SBTTL  SUBTEST 8.12:  DATIP/DATO
                                                        0CA5  2626 ;+
                                                        0CA5  2627 ; Subtest description:
                                                        0CA5  2628 ;
                                                        0CA5  2629 ;       This subtest checks the ability of the UBI to perform a DATIP/DATO
                                                        0CA5  2630 ;       transaction. This test is similar to the DATI 2 subtest, with
                                                        0CA5  2631 ;       the addition of checking that the DATO portion of the transaction also
                                                        0CA5  2632 ;       works.
                                                        0CA5  2633 ;
                                                        0CA5  2634 ; Subtest steps:
                                                        0CA5  2635 ;
                                                        0CA5  2636 ;       SUBTEST DATIP/DATO
                                                        0CA5  2637 ;       :  Invalidate all maps but one
                                                        0CA5  2638 ;       :  Load UBE address registers
                                                        0CA5  2639 ;       :  DO FOR Alignment = {word, longword}
                                                        0CA5  2640 ;       :  :  REPEAT
                                                        0CA5  2641 ;       :  :  :  EXECUTE DATIP/DATO
                                                        0CA5  2642 ;       :  :  :  WAIT
                                                        0CA5  2643 ;       :  :  :  IF error
                                                        0CA5  2644 ;       :  :  :  :  THEN REPORT error
                                                        0CA5  2645 ;       :  :  :  ENDIF
                                                        0CA5  2646 ;       :  :  : UNTIL patterns exhausted
                                                        0CA5  2647 ;       :  :  ENDREPEAT
                                                        0CA5  2648 ;       :  ENDDO
                                                        0CA5  2649 ;       ENDSUBTEST DATIP/DATO
                                                        0CA5  2650 ;
                                                        0CA5  2651 ; Errors:
                                                        0CA5  2652 ;
                                                        0CA5  2653 ;-
                                                        0CA5  2654          $DS_BGNSUB
                                                        0CA5        T8_S12::
 00010030 9F   00000084'EF    FA                        0CA5                    CALLG   $S$, @#DS$BGNSUB
                                                        0CB0  2655
                    37   00   DA                        0CB0  2656          MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
              00000000'EF   B4                        0CB3  2657          CLRW    W_FLAGS                         ; clear flags word
              00000000'FF   7C                        0CB9  2658          CLRQ    @#A_UBE_BUFFER_0                ; clear buffer quadword
           56 00000000'EF   D0                        0CBF  2659          MOVL    A_UBE_BUFFER_0,R6               ; load memory loc address
                                                        0CC6  2660 ;
                                                        0CC6  2661 ; Set up first valid Map Entry.
                                                        0CC6  2662 ;
           52 00000002'EF   D0                        0CC6  2663          MOVL    FRST_VAL_MAP,R2                 ; load No of first valid Map
 00F26800 9F42 0000004A'EF   D0                        0CCD  2664          MOVL    L_PAGE_NO,@#A_MAP_BGN[R2]       ; load PFN into Map Entry
 00F26800 9F42 80000000 8F   C8                        0CD9  2665          BISL2   #M_MAP_VAL,@#A_MAP_BGN[R2]      ; validate Map Entry
           55 0000029F'EF   3E                        0CE5  2666 0$:      MOVAW   W_PTRN_TBL,R5                   ; load pattern table address
                                                        0CEC  2667 ;
                                                        0CEC  2668 ; Load address into command table and load expected value
                                                        0CEC  2669 ;
                 66   65   B0                        0CEC  2670 10$:     MOVW    (R5),(R6)                       ; load pattern
      0000005E'EF   66   01   9C                        0CEF  2671          ROTL    #1,(R6),L_EXP_DATA              ; load expected data
   54 0000005E'EF   01   10   EF                        0CF7  2672          EXTZV   #16,#1,L_EXP_DATA,R4            ; extract end-around carry bit
           0000005E'EF   54   C8                        0D00  2673          BISL2   R4,L_EXP_DATA                   ; insert carry bit
           59 000002D7'EF   DE                        0D07  2674          MOVAL   UBE_DATIP_1,R9                  ; load command table address
              06 A9   56   B0                        0D0E  2675          MOVW    R6,6(R9)                        ; load offset within page
06 A9   09   09 00000002'EF   F0                        0D12  2676          INSV    FRST_VAL_MAP,#9,#9,6(R9)        ; load Map Number
                                                        0D1C  2677
                                                        0D1C  2678 ;
                                                        0D1C  2679 ; If ready bit not set then report error and abort test
```

```
                                 0D1C  2680 ;
         29 000002BF'FF    07    E0  0D1C  2681 20$:    BBS     #V_RDY,@A_BE0_CR1,30$            ; br = ready bit set
                                 0D24  2682
     0000004E'EF    00F26010 9F    D0  0D24  2683         MOVL    @#A_UBI_CSR,L_SAVE_CSR           ; save UBI CSR
                                 0D2F  2684         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report no-RDY error
                                 0D2F  2685                         MSGADR=T_DATIP_ERR,-
                                 0D2F  2686                         PRLINK=XFER_ERR
                00001420'EF    DF  0D2F                        PUSHAL  XFER_ERR
                000005D3'EF    DF  0D35                        PUSHAL  T_DATIP_ERR
                00000000'EF    DD  0D3B                        PUSHL   BE0_UNIT_NO
                          01    DD  0D41                        PUSHL   #$ER
                                 0D43      ;;; TEST 8, SUBTEST 12, ERROR 1
            000100D0 9F    04    FB  0D43                        CALLS   #$$M, @#DS$ERRHARD
                                 0D4A  2687
                                 0D4A  2688         $DS_ABORT       TEST                     ; abort
                          50    D4  0D4A                        CLRL    R0               ; SEND A WARNING TO THE SUPERVISOR
                          04    0D4C                            RET                      ; TERMINATE TEST
                                 0D4D  2689 ;
                                 0D4D  2690 ; Execute DATIP/DATO, wait
                                 0D4D  2691 ;
                00001850'EF    16  0D4D  2692 30$:    JSB     UBE_SET_UP                       ; execute DATIP/DATO
                                 0D53  2693         $DS_WAITMS_S    TIME=#1                  ; wait
                          00    DD  0D53                        PUSHL   #0
                          01    DD  0D55                        PUSHL   #1
            00010060 9F    02    FB  0D57                        CALLS   #2, @#DS$WAITMS
                                 0D5E  2694 ;
                                 0D5E  2695 ; Check for DATIP data or execution errors, report if any
                                 0D5E  2696 ;
0000004E'EF    00F26010 9F    D0  0D5E  2697         MOVL    @#A_UBI_CSR,L_SAVE_CSR           ; save UBI CSR contents
000002AF'FF    0000005E'EF    B1  0D69  2698         CMPW    L_EXP_DATA,@A_BE0_DB             ; check UBE data register
                          12    13  0D74  2699         BEQL    40$                              ; br = no data error
     00000000'EF    1000 8F    A8  0D76  2700         BISW2   #M_DATA_ERR,W_FLAGS              ; set data error flag
          53    000002AF'FF    3C  0D7F  2701         MOVZWL  @A_BE0_DB,R3                     ; extend actual data
                          15    11  0D86  2702         BRB     50$                              ; br to error report
0000004E'EF    8001C000 8F    D3  0D88  2703 40$:    BITL    #M_UBI_STS,L_SAVE_CSR            ; check UBI status bits
                          08    12  0D93  2704         BNEQ    50$                              ; br = UBI error
        3B 000002BF'FF    0F    E1  0D95  2705         BBC     #V_ERR,@A_BE0_CR1,60$            ; br = no UBE errors
                                 0D9D  2706
                                 0D9D  2707 50$:    $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                 0D9D  2708                         MSGADR=T_DATIP_ERR,-
                                 0D9D  2709                         PRLINK=XFER_ERR,-
                                 0D9D  2710                         P1=A_BE0_DB,-             ; location
                                 0D9D  2711                         P2=L_EXP_DATA,-           ; expected value
                                 0D9D  2712                         P3=R3,-                   ; actual value
                                 0D9D  2713                         P4=#T_WORD_ERR,-          ; data type
                                 0D9D  2714                         P5=FRST_VAL_MAP           ; map number used
                00000002'EF    DD  0D9D                        PUSHL   FRST_VAL_MAP
                00000DA0'8F    DD  0DA3                        PUSHL   #T_WORD_ERR
                          53    DD  0DA9                        PUSHL   R3
                0000005E'EF    DD  0DAB                        PUSHL   L_EXP_DATA
                000002AF'EF    DD  0DB1                        PUSHL   A_BE0_DB
                00001420'EF    DF  0DB7                        PUSHAL  XFER_ERR
                000005D3'EF    DF  0DBD                        PUSHAL  T_DATIP_ERR
                00000000'EF    DD  0DC3                        PUSHL   BE0_UNIT_NO
                          02    DD  0DC9                        PUSHL   #$ER
                                 0DCB      ;;; TEST 8, SUBTEST 12, ERROR 2
            000100D0 9F    09    FB  0DCB                        CALLS   #$$M, @#DS$ERRHARD
```

```
                                 0DD2  2715
                00000000'EF   B4 0DD2  2716          CLRW    W_FLAGS                         ; clear flags word
                                 0DD8  2717 ;
                                 0DD8  2718 ; Check DATO data, if error then report
                                 0DD8  2719 ;
            66  0000005E'EF   B1 0DD8  2720 60$:     CMPW    L_EXP_DATA,(R6)                 ; check data returned
                         40   13 0DDF  2721          BEQL    65$                             ; br = ok
   00000000'EF       1000 8F   A8 0DE1  2722          BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
                                 0DEA  2723
                                 0DEA  2724          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report data error
                                 0DEA  2725                          MSGADR=T_DATIP_ERR,-
                                 0DEA  2726                          PRLINK=XFER_ERR,-
                                 0DEA  2727                          P1=R6,-                 ; location
                                 0DEA  2728                          P2=L_EXP_DATA,-         ; expected value
                                 0DEA  2729                          P3=(R6),-               ; actual value
                                 0DEA  2730                          P4=#T_WORD_ERR,-        ; data type
                                 0DEA  2731                          P5=FRST_VAL_MAP         ; map number used
                00000002'EF   DD 0DEA                        PUSHL   FRST_VAL_MAP
                00000DA0'8F   DD 0DF0                        PUSHL   #T_WORD_ERR
                         66   DD 0DF6                        PUSHL   (R6)
                0000005E'EF   DD 0DF8                        PUSHL   L_EXP_DATA
                         56   DD 0DFE                        PUSHL   R6
                00001420'EF   DF 0E00                        PUSHAL  XFER_ERR
                000005D3'EF   DF 0E06                        PUSHAL  T_DATIP_ERR
                00000000'EF   DD 0E0C                        PUSHL   BE0_UNIT_NO
                         03   DD 0E12                        PUSHL   #$ER
                                 0E14          ;;; TEST 8, SUBTEST 12, ERROR 3
      000100D0 9F        09   FB 0E14                        CALLS   #$$M, @#DS$ERRHARD
                                 0E1B  2732
                00000000'EF   B4 0E1B  2733          CLRW    W_FLAGS                         ; clear flags word
                                 0E21  2734 65$:     $DS_CKLOOP      10$                     ; scope loop at 10$
   00010040 9F      FEC7 CF   FA 0E21                        CALLG   10$, @#DS$CKLOOP
                                 0E2A  2735 ;
                                 0E2A  2736 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                                 0E2A  2737 ; word-aligned cases have been tested.
                                 0E2A  2738 ;
                00FF 8F      85   B1 0E2A  2739 70$:     CMPW    (R5)+,#^X00FF           ; patterns exhausted?
                             03   13 0E2F  2740          BEQL    80$                     ; br = patterns exhausted
                           FEB8   31 0E31  2741          BRW     10$                     ; br = patterns not exhausted
                      03 56   01   E2 0E34  2742 80$:     BBSS    #1,R6,100$              ; br = done for word alignment
                           FEAA   31 0E38  2743          BRW     0$                      ; br, repeat for word alignment
                                 0E3B  2744 100$:
                                 0E3B  2745
                                 0E3B  2746          $DS_ENDSUB
                                 0E3B          T8_S12_X:
   00010038 9F   00000084'EF   FA 0E3B                        CALLG   $S$, @#DS$ENDSUB
                                 0E46  2747
                                 0E46  2748          $DS_PAGE
```

```
                                        0E46                    .SBTTL  SUBTEST 8.13:  Address Offset DATI
                                        0E46  2751 ;+
                                        0E46  2752 ; Subtest description:
                                        0E46  2753 ;
                                        0E46  2754 ;       This subtest checks the ability of the UBI to perform a DATI
                                        0E46  2755 ;       transaction in address offset mode.
                                        0E46  2756 ;
                                        0E46  2757 ; Subtest steps:
                                        0E46  2758 ;
                                        0E46  2759 ;       SUBTEST Address Offset DATI
                                        0E46  2760 ;       :  Set Byte offset bit in first valid Map Entry
                                        0E46  2761 ;       :  DO for Alignment = {longword,word}
                                        0E46  2762 ;       :  :  REPEAT
                                        0E46  2763 ;       :  :  :  EXECUTE DATI
                                        0E46  2764 ;       :  :  :  WAIT
                                        0E46  2765 ;       :  :  :  IF Execution error
                                        0E46  2766 ;       :  :  :  :  THEN REPORT error
                                        0E46  2767 ;       :  :  :  ENDIF
                                        0E46  2768 ;       :  :  :  IF Data error
                                        0E46  2769 ;       :  :  :  :  THEN REPORT error
                                        0E46  2770 ;       :  :  :  ENDIF
                                        0E46  2771 ;       :  :  : UNTIL patterns exhausted
                                        0E46  2772 ;       :  :  ENDREPEAT
                                        0E46  2773 ;       :  ENDDO
                                        0E46  2774 ;       ENDSUBTEST Address Offset DATI
                                        0E46  2775 ;
                                        0E46  2776 ; Errors:
                                        0E46  2777 ;
                                        0E46  2778 ;-
                                        0E46  2779          $DS_BGNSUB
                                        0E46        T8_S13::
    00010030 9F   00000090'EF    FA     0E46                CALLG   $S$, @#DS$BGNSUB
                                        0E51  2780
                      37   00    DA     0E51  2781          MTPR    #0,#PR$L_UB_INIT            ; initialize Unibus devices
                00000000'EF    B4     0E54  2782          CLRW    W_FLAGS                     ; clear flags word
                00000000'FF    D4     0E5A  2783          CLRL    @A_UBE_BUFFER_0             ; clear buffer longword
          56    00000000'EF    01     C1    0E60  2784          ADDL3   #1,A_UBE_BUFFER_0,R6        ; load memory loc address
                                        0E68  2785 ;
                                        0E68  2786 ; Set Byte Offset bit in first valid Map Entry
                                        0E68  2787 ;
             54   00000002'EF    D0     0E68  2788          MOVL    FRST_VAL_MAP,R4             ; load No of first valid Map
   00F26800 9F   44   02000000 8F    C8    0E6F  2789          BISL2   #M_BYTE_OFF,@#A_MAP_BGN[R4]  ; set byte offset bit
             55   0000029F'EF    3E     0E7B  2790 10$:     MOVAW   W_PTRN_TBL,R5               ; load pattern table address
                                        0E82  2791 ;
                                        0E82  2792 ; Load address into DATI command table
                                        0E82  2793 ;
             59   000002CB'EF    3E     0E82  2794 30$:     MOVAW   UBE_DATI_1,R9               ; load command table address
               06 A9    56    01    A3    0E89  2795          SUBW3   #1,R6,6(R9)                 ; load offset within page
06 A9    09    09   00000002'EF    F0     0E8E  2796          INSV    FRST_VAL_MAP,#9,#9,6(R9)    ; load Map Number
                                        0E98  2797 ;
                                        0E98  2798 ; If ready bit not set then report error and abort test
                                        0E98  2799 ;
          29 000002BF'FF    07    E0     0E98  2800 40$:     BBS     #V_RDY,@A_BE0_CR1,50$       ; br = ready bit set
                                        0EA0  2801
    0000004E'EF   00F26010 9F    D0     0EA0  2802          MOVL    @#A_UBI_CSR,L_SAVE_CSR      ; save UBI CSR
                                        0EAB  2803          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-  ; report no-RDY error
                                        0EAB  2804                          MSGADR=T_A0_DATI_ERR,-
```

```
                                 0EAB  2805                          PRLINK=XFER_ERR
          00001420'EF    DF      0EAB                        PUSHAL  XFER_ERR
          000006B3'EF    DF      0EB1                        PUSHAL  T_AO_DATI_ERR
          00000000'EF    DD      0EB7                        PUSHL   BE0_UNIT_NO
                   01    DD      0EBD                        PUSHL   #$ER
                                 0EBF        ;;; TEST 8, SUBTEST 13, ERROR 1
       000100D0 9F       04    FB  0EBF                        CALLS   #$$M, @#DS$ERRHARD
                                 0EC6  2806
                                 0EC6  2807          $DS_ABORT       TEST                    ; abort test
                   50    D4      0EC6                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                         04      0EC8                        RET                     ; TERMINATE TEST
                                 0EC9  2808 ;
                                 0EC9  2809 ; Execute DATI, wait
                                 0EC9  2810 ;
                66       65    B0  0EC9  2811 50$:     MOVW    (R5),(R6)                       ; load pattern
          00001850'EF    16      0ECC  2812          JSB     UBE_SET_UP                      ; execute DATI
                                 0ED2  2813          $DS_WAITMS_S    TIME=#1                 ; wait
                   00    DD      0ED2                        PUSHL   #0
                   01    DD      0ED4                        PUSHL   #1
       00010060 9F       02    FB  0ED6                        CALLS   #2, @#DS$WAITMS
                                 0EDD  2814 ;
                                 0EDD  2815 ; Check for data or execution errors, report if any
                                 0EDD  2816 ;
    0000004E'EF    00F26010 9F    D0  0EDD  2817          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
             65    000002AF'FF    B1  0EE8  2818          CMPW    @A_BE0_DB,(R5)                  ; check UBE data register
                      12    13    0EEF  2819          BEQL    60$                             ; br = no data error
       00000000'EF    1000 8F    A8  0EF1  2820          BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
             52    000002AF'FF    3C  0EFA  2821          MOVZWL  @A_BE0_DB,R2                    ; extend actual data
                      15    11    0F01  2822          BRB     70$                             ; br to error report
    0000004E'EF    8001C000 8F    D3  0F03  2823 60$:     BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status bits
                      08    12    0F0E  2824          BNEQ    70$                             ; br = UBI error
          37 000002BF'FF    0F    E1  0F10  2825          BBC     #V_ERR,@A_BE0_CR1,75$           ; br = no UBE errors
                                 0F18  2826
                                 0F18  2827 70$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                 0F18  2828                          MSGADR=T_AO_DATI_ERR,-
                                 0F18  2829                          PRLINK=XFER_ERR,-
                                 0F18  2830                          P1=A_BE0_DB,-           ; location
                                 0F18  2831                          P2=(R5),-               ; expected value
                                 0F18  2832                          P3=R2,-                 ; actual value
                                 0F18  2833                          P4=#T_WORD_ERR,-        ; data type
                                 0F18  2834                          P5=FRST_VAL_MAP         ; map number used
          00000002'EF    DD      0F18                        PUSHL   FRST_VAL_MAP
          00000DA0'8F    DD      0F1E                        PUSHL   #T_WORD_ERR
                   52    DD      0F24                        PUSHL   R2
                   65    DD      0F26                        PUSHL   (R5)
          000002AF'EF    DD      0F28                        PUSHL   A_BE0_DB
          00001420'EF    DF      0F2E                        PUSHAL  XFER_ERR
          000006B3'EF    DF      0F34                        PUSHAL  T_AO_DATI_ERR
          00000000'EF    DD      0F3A                        PUSHL   BE0_UNIT_NO
                   02    DD      0F40                        PUSHL   #$ER
                                 0F42        ;;; TEST 8, SUBTEST 13, ERROR 2
       000100D0 9F       09    FB  0F42                        CALLS   #$$M, @#DS$ERRHARD
                                 0F49  2835
          00000000'EF    B4      0F49  2836          CLRW    W_FLAGS                         ; clear flags word
                                 0F4F  2837 75$:     $DS_CKLOOP      30$                     ; scope loop at 30$
    0001004C 9F    FF2F CF    FA  0F4F                        CALLG   30$, @#DS$CKLOOP
                                 0F58  2838 ;
```

```
                                0F58  2839 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                                0F58  2840 ; word-aligned cases have been tested.
                                0F58  2841 ;
          00FF 8F    85    B1   0F58  2842 80$:    CMPW    (R5)+,#^X00FF                   ; patterns exhausted ?
                       03    13   0F5D  2843         BEQL    90$                             ; br = patterns exhausted
                     FF20    31   0F5F  2844         BRW     30$                             ; br = patterns not exhausted
            03 56     01    E2   0F62  2845 90$:    BBSS    #1,R6,100$                      ; br = done for word alignment
                     FF12    31   0F66  2846         BRW     10$                             ; br, repeat for word alignment
                                0F69  2847 100$:
                                0F69  2848         $DS_ENDSUB
                                0F69       T8_S13_X:
00010038 9F    00000090'EF    FA   0F69                 CALLG   $S$, @#DS$ENDSUB
                                0F74  2849
                                0F74  2850         $DS_PAGE
```

```
                                0F74                    .SBTTL  SUBTEST 8.14:  Address Offset DATO
                                0F74  2853 ;+
                                0F74  2854 ; Subtest description:
                                0F74  2855 ;
                                0F74  2856 ;       This subtest checks the ability of the UBI to  perform a
                                0F74  2857 ;       DATO transaction in address offset mode.
                                0F74  2858 ;
                                0F74  2859 ; Subtest steps:
                                0F74  2860 ;
                                0F74  2861 ;       SUBTEST Address Offset DATO
                                0F74  2862 ;       :  Invalidate all maps but one
                                0F74  2863 ;       :  DO FOR Alignment = {word, longword}
                                0F74  2864 ;       :  :  REPEAT
                                0F74  2865 ;       :  :  :  Initiate DATO
                                0F74  2866 ;       :  :  :  WAIT
                                0F74  2867 ;       :  :  :  IF Execution error
                                0F74  2868 ;       :  :  :  :  THEN REPORT error
                                0F74  2869 ;       :  :  :  ENDIF
                                0F74  2870 ;       :  :  :  IF Data error
                                0F74  2871 ;       :  :  :  :  THEN REPORT error
                                0F74  2872 ;       :  :  :  ENDIF
                                0F74  2873 ;       :  :  :  UNTIL patterns exhausted
                                0F74  2874 ;       :  :  ENDREPEAT
                                0F74  2875 ;       :  ENDDO
                                0F74  2876 ;       ENDSUBTEST Address Offset DATO
                                0F74  2877 ;
                                0F74  2878 ; Errors:
                                0F74  2879 ;
                                0F74  2880 ;-
                                0F74  2881          $DS_BGNSUB
                                0F74       T8_S14::
00010030 9F   0000009C'EF   FA  0F74                CALLG   $S$, @#DS$BGNSUB
                                0F7F  2882
              37   00       DA  0F7F  2883          MTPR    #0,#PR$L_UB_INIT        ; initialize Unibus devices
          00000000'EF       B4  0F82  2884          CLRW    W_FLAGS                 ; clear flags word
     56   00000000'EF       C1  0F88  2885          ADDL3   #T,A_UBE_BUFFER_0,R6    ; load memory loc address
     55   0000029F'EF       3E  0F90  2886 0$:      MOVAW   W_PTRN_TBL,R5           ; load pattern table address
                                0F97  2887 ;
                                0F97  2888 ; Load address and data into DATO command table
                                0F97  2889 ;
          0000005E'EF       65  B0  0F97  2890 10$:     MOVW    (R5),L_EXP_DATA         ; load pattern
     59   000002EF'EF       3E  0F9E  2891          MOVAW   UBE_DATO_1,R9           ; load command table address
   02 A9  0000005E'EF       B0  0FA5  2892          MOVW    L_EXP_DATA,2(R9)        ; load data
    06 A9   56               01  A3  0FAD  2893          SUBW3   #T,R6,6(R9)             ; load offset within page
06 A9   09   09   00000002'EF   F0  0FB2  2894          INSV    FRST_VAL_MAP,#9,#9,6(R9)    ; load Map Number
                         66     B4  0FBC  2895          CLRW    (R6)                    ; clear buffer location
                                0FBE  2896 ;
                                0FBE  2897 ; If ready bit not set then report error and abort test
                                0FBE  2898 ;
   29 000002BF'FF       07      E0  0FBE  2899 20$:     BBS     #V_RDY,@A_BE0_CR1,30$    ; br = ready bit set
                                0FC6  2900
0000004E'EF   00F26010 9F       D0  0FC6  2901          MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
                                0FD1  2902          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-  ; report no-RDY error
                                0FD1  2903                          MSGADR=T_AO_DATO_ERR,-
                                0FD1  2904                          PRLINK=XFER_ERR
              00001420'EF       DF  0FD1                    PUSHAL  XFER_ERR
              00000640'EF       DF  0FD7                    PUSHAL  T_AO_DATO_ERR
```

```
                00000000'EF   DD  0FDD                        PUSHL   BE0_UNIT_NO
                          01  DD  0FE3                        PUSHL   #$ER
                              0FE5        ;;; TEST 8, SUBTEST 14, ERROR 1
                000100D0 9F   04  FB  0FE5                    CALLS   #$$M, @#DS$ERRHARD
                              0FEC  2905
                              0FEC  2906        $DS_ABORT     TEST                    ; abort test
                          50  D4  0FEC                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04  0FEE                            RET                     ; TERMINATE TEST
                              0FEF  2907 ;
                              0FEF  2908 ; Execute DATO, wait
                              0FEF  2909 ;
                00001850'EF   16  0FEF  2910 30$:     JSB     UBE_SET_UP              ; execute DATO
                              0FF5  2911        $DS_WAITMS_S  TIME=#1                 ; wait
                          00  DD  0FF5                        PUSHL   #0
                          01  DD  0FF7                        PUSHL   #1
                000010060 9F  02  FB  0FF9                    CALLS   #2, @#DS$WAITMS
                              1000  2912 ;
                              1000  2913 ; Check for data or execution errors, report if any
                              1000  2914 ;
    0000004E'EF   00F26010 9F   D0  1000  2915        MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
            66    0000005E'EF   B1  100B  2916        CMPW    L_EXP_DATA,(R6)         ; check UBE data register
                          0B  13  1012  2917        BEQL    40$                     ; br = ok
        00000000'EF     1000  8F  A8  1014  2918        BISW2   #M_DATA_ERR,W_FLAGS     ; set data error flag
                          15  11  101D  2919        BRB     50$                     ; br to error report
    0000004E'EF   8001C000  8F  D3  101F  2920 40$:     BITL    #M_UBI_STS,L_SAVE_CSR   ; check UBI status bits
                          08  12  102A  2921        BNEQ    50$                     ; br = UBI error
           37 000002BF'FF   0F  E1  102C  2922        BBC     #V_ERR,@A_BE0_CR1,55$   ; br = no UBE errors
                              1034  2923
                              1034  2924 50$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-  ; report data error
                              1034  2925                        MSGADR=T_AO_DATO_ERR,-
                              1034  2926                        PRLINK=XFER_ERR,-
                              1034  2927                        P1=R6,-               ; location
                              1034  2928                        P2=L_EXP_DATA,-       ; expected value
                              1034  2929                        P3=(R6),-             ; actual value
                              1034  2930                        P4=#T_WORD_ERR,-      ; data type
                              1034  2931                        P5=FRST_VAC_MAP       ; map number used
            00000002'EF   DD  1034                        PUSHL   FRST_VAC_MAP
            00000DA0'8F   DD  103A                        PUSHL   #T_WORD_ERR
                      66  DD  1040                        PUSHL   (R6)
            0000005E'EF   DD  1042                        PUSHL   L_EXP_DATA
                      56  DD  1048                        PUSHL   R6
            00001420'EF   DF  104A                        PUSHAL  XFER_ERR
            00000640'EF   DF  1050                        PUSHAL  T_AO_DATO_ERR
            00000000'EF   DD  1056                        PUSHL   BE0_UNIT_NO
                      02  DD  105C                        PUSHL   #$ER
                              105E        ;;; TEST 8, SUBTEST 14, ERROR 2
        000100D0 9F   09  FB  105E                        CALLS   #$$M, @#DS$ERRHARD
                              1065  2932
            00000000'EF   B4  1065  2933        CLRW    W_FLAGS                 ; clear flags word
                              106B  2934 55$:     $DS_CKLOOP    10$               ; scope loop at 10$
    00010040 9F   FF28 CF   FA  106B                        CALLG   10$, @#DS$CKLOOP
                              1074  2935 ;
                              1074  2936 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                              1074  2937 ; word-aligned cases have been tested.
                              1074  2938 ;
            00FF 8F   85  B1  1074  2939 60$:     CMPW    (R5)+,#^X00FF           ; patterns exhausted?
                      03  13  1079  2940        BEQL    90$                     ; br = patterns exhausted
```

```
                      FF19   31  107B  2941          BRW     10$                             ; br = patterns not exhausted
                  03 56  01   E2  107E  2942 90$:     BBSS    #1,R6,100$                      ; br = done for word alignment
                      FF0B   31  1082  2943          BRW     0$                              ; br, repeat for word alignment
                             1085  2944 100$:
                             1085  2945
                             1085  2946          $DS_ENDSUB
                             1085        T8_S14_X:
00010038 9F   0000009C'EF    FA  1085                        CALLG   $S$, @#DS$ENDSUB
                             1090  2947
                             1090  2948          $DS_PAGE
```

```
                                1090                    .SBTTL  SUBTEST 8.15:  Address Offset DATOB
                                1090  2951 ;+
                                1090  2952 ; Subtest description:
                                1090  2953 ;
                                1090  2954 ;       This subtest checks the ability of the UBI to  perform a DATOB
                                1090  2955 ;       transaction in address offset mode.
                                1090  2956 ;
                                1090  2957 ; Subtest steps:
                                1090  2958 ;
                                1090  2959 ;       SUBTEST Address Offset DATOB
                                1090  2960 ;       :  Invalidate all maps but one
                                1090  2961 ;       :  DO FOR Alignment = {word, longword}
                                1090  2962 ;       :  :  REPEAT
                                1090  2963 ;       :  :  :  Initiate DATOB
                                1090  2964 ;       :  :  :  WAIT
                                1090  2965 ;       :  :  :  IF Execution error
                                1090  2966 ;       :  :  :  :  THEN REPORT error
                                1090  2967 ;       :  :  :  ENDIF
                                1090  2968 ;       :  :  :  IF Data error
                                1090  2969 ;       :  :  :  :  THEN REPORT error
                                1090  2970 ;       :  :  :  ENDIF
                                1090  2971 ;       :  :  :  UNTIL patterns exhausted
                                1090  2972 ;       :  :  ENDREPEAT
                                1090  2973 ;       :  ENDDO
                                1090  2974 ;       ENDSUBTEST Address Offset DATOB
                                1090  2975 ;
                                1090  2976 ; Errors:
                                1090  2977 ;
                                1090  2978 ;-
                                1090  2979          $DS_BGNSUB
                                1090          T8_S15::
00010030 9F   000000A8'EF   FA  1090                  CALLG   $S$, @#DS$BGNSUB
                                109B  2980
                   37   00  DA  109B  2981          MTPR    #0,#PR$L_UB_INIT        ; initialize Unibus devices
              00000000'EF   B4  109E  2982          CLRW    W_FLAGS                 ; clear flags word
              00000000'FF   D4  10A4  2983          CLRL    @A_UBE_BUFFER_0         ; clear buffer longword
        56    00000000'EF   01  C1  10AA  2984          ADDL3   #1,A_UBE_BUFFER_0,R6    ; load memory loc address
        55    0000029F'EF   3E  10B2  2985 0$:      MOVAW   W_PTRN_TBL,R5           ; load pattern table address
                                10B9  2986 ;
                                10B9  2987 ; Load address and data into DATOB command table
                                10B9  2988 ;
                   66   B4  10B9  2989 10$:      CLRW    (R6)                    ; clear destination
              0000005E'EF   B0  10BB  2990          MOVW    (R5),L_EXP_DATA         ; load expected pattern
        59    000002E3'EF   3E  10C2  2991          MOVAW   UBE_DATOB_T,R9          ; load command table address
     02 A9    0000005E'EF   90  10C9  2992          MOVB    L_EXP_DATA,2(R9)        ; load data
  06 A9   56   01   A3  10D1  2993          SUBW3   #T,R6,6(R9)             ; load offset within page
06 A9   09   09   00000002'EF   F0  10D6  2994          INSV    FRST_VAL_MAP,#9,#9,6(R9)    ; load Map Number
                                10E0  2995 ;
                                10E0  2996 ; If ready bit not set then report error and abort test
                                10E0  2997 ;
     29 000002BF'FF   07   E0  10E0  2998 20$:      BBS     #V_RDY,@A_BE0_CR1,30$           ; br = ready bit set
                                10E8  2999
0000004E'EF   00F26010 9F   D0  10E8  3000          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                10F3  3001          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report no-RDY error
                                10F3  3002                          MSGADR=T_A0_DATOB_ERR,-
                                10F3  3003                          PRLINK=XFER_ERR
              00001420'EF   DF  10F3                  PUSHAL  XFER_ERR
```

```
                00000679'EF   DF  10F9                        PUSHAL  T_AO_DATOB_ERR
                00000000'EF   DD  10FF                        PUSHL   BE0_UNIT_NO
                          01  DD  1105                        PUSHL   #$ER
                                  1107        ;;; TEST 8, SUBTEST 15, ERROR 1
              0100D0 9F   04  FB  1107                        CALLS   #$$M, @#DS$ERRHARD
                                  110E  3004
                                  110E  3005          $DS_ABORT       TEST                    ; abort test
                          50  D4  110E                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04  1110                            RET                     ; TERMINATE TEST
                                  1111  3006 ;
                                  1111  3007 ; Execute DATOB, wait
                                  1111  3008 ;
                00001850'EF   16  1111  3009 30$:       JSB     UBE_SET_UP                      ; execute DATOB
                                  1117  3010           $DS_WAITMS_S    TIME=#1                 ; wait
                          00  DD  1117                        PUSHL   #0
                          01  DD  1119                        PUSHL   #1
              00010060 9F   02  FB  111B                        CALLS   #2, @#DS$WAITMS
                                  1122  3011 ;
                                  1122  3012 ; Check for data or execution errors, report if any
                                  1122  3013 ;
    0000004E'EF   00F26010 9F   D0  1122  3014                  MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
            66    0000005E'EF   91  112D  3015                  CMPB    L_EXP_DATA,(R6)         ; check UBE data register
                          0B  13  1134  3016                    BEQL    40$                     ; br = ok
       00000000'EF    1000 8F   A8  1136  3017                  BISW2   #M_DATA_ERR,W_FLAGS     ; set data error flag
                          15  11  113F  3018                    BRB     50$                     ; br to error report
    0000004E'EF    8001C000 8F   D3  1141  3019 40$:       BITL    #M_UBI_STS,L_SAVE_CSR   ; check UBI status bits
                          08  12  114C  3020                    BNEQ    50$                     ; br = UBI error
          37 000002BF'FF    0F  E1  114E  3021                  BBC     #V_ERR,@A_BE0_CR1,55$   ; br = no UBE errors
                                  1156  3022
                                  1156  3023 50$:       $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report data error
                                  1156  3024                          MSGADR=T_AO_DATOB_ERR,-
                                  1156  3025                          PRLINK=XFER_ERR,-
                                  1156  3026                          P1=R6,-                 ; location
                                  1156  3027                          P2=L_EXP_DATA,-         ; expected value
                                  1156  3028                          P3=(R6),-               ; actual value
                                  1156  3029                          P4=#T_BYTE_ERR,-        ; data type
                                  1156  3030                          P5=FRST_VAL_MAP         ; map number used
              00000002'EF   DD  1156                            PUSHL   FRST_VAL_MAP
              00000D72'8F   DD  115C                            PUSHL   #T_BYTE_ERR
                          66  DD  1162                        PUSHL   (R6)
              0000005E'EF   DD  1164                            PUSHL   L_EXP_DATA
                          56  DD  116A                        PUSHL   R6
              00001420'EF   DF  116C                            PUSHAL  XFER_ERR
              00000679'EF   DF  1172                            PUSHAL  T_AO_DATOB_ERR
              00000000'EF   DD  1178                            PUSHL   BE0_UNIT_NO
                          02  DD  117E                        PUSHL   #$ER
                                  1180        ;;; TEST 8, SUBTEST 15, ERROR 2
            000100D0 9F   09  FB  1180                        CALLS   #$$M, @#DS$ERRHARD
                                  1187  3031
              00000000'EF   B4  1187  3032          CLRW    W_FLAGS                         ; clear flags word
                                  118D  3033 55$:       $DS_CKLOOP      10$                     ; scope loop at 10$
    00010040 9F    FF28 CF   FA  118D                        CALLG   10$, @#DS$CKLOOP
                                  1196  3034 ;
                                  1196  3035 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                                  1196  3036 ; word-aligned cases have been tested.
                                  1196  3037 ;
              0F0F 8F   85  B1  1196  3038 60$:       CMPW    (R5)+,#^X0F0F                   ; patterns exhausted?
```

```
                      03   13  119B  3039          BEQL    90$                             ; br = patterns exhausted
                    FF19   31  119D  3040          BRW     10$                             ; br = patterns not exhausted
                 03 56 01   E2  11A0  3041 90$:     BBSS    #1,R6,100$                      ; br = done for word alignment
                    FF08   31  11A4  3042          BRW     0$                              ; br, repeat for word alignment
                           11A7  3043 100$:
                           11A7  3044
                           11A7  3045          $DS_ENDSUB
                           11A7        T8_S15_X:
00010038 9F   000000A8'EF   FA  11A7                      CALLG   $S$, @#DS$ENDSUB
                           11B2  3046
                           11B2  3047          $DS_PAGE
```

```
                                11B2                    .SBTTL  SUBTEST 8.16:  Address Offset DATIP/DATO
                                11B2  3050 ;+
                                11B2  3051 ; Subtest description:
                                11B2  3052 ;
                                11B2  3053 ;       This subtest checks the ability of the UBI to  perform a
                                11B2  3054 ;       DATIP/DATO transaction in address offset mode.
                                11B2  3055 ;
                                11B2  3056 ; Subtest steps:
                                11B2  3057 ;
                                11B2  3058 ;       SUBTEST Address Offset DATIP/DATO
                                11B2  3059 ;       :  Invalidate all maps but one
                                11B2  3060 ;       :  DO FOR Alignment = {word, longword}
                                11B2  3061 ;       :  :  REPEAT
                                11B2  3062 ;       :  :  :  Initiate DATIP/DATO
                                11B2  3063 ;       :  :  :  WAIT
                                11B2  3064 ;       :  :  :  IF Execution error
                                11B2  3065 ;       :  :  :  :  THEN REPORT error
                                11B2  3066 ;       :  :  :  ENDIF
                                11B2  3067 ;       :  :  :  IF Data error
                                11B2  3068 ;       :  :  :  :  THEN REPORT error
                                11B2  3069 ;       :  :  :  ENDIF
                                11B2  3070 ;       :  :  : UNTIL patterns exhausted
                                11B2  3071 ;       :  :  ENDREPEAT
                                11B2  3072 ;       :  ENDDO
                                11B2  3073 ;       ENDSUBTEST Address Offset DATIP/DATO
                                11B2  3074 ;
                                11B2  3075 ; Errors:
                                11B2  3076 ;
                                11B2  3077 ;-
                                11B2  3078          $DS_BGNSUB
                                11B2       T8_S16::
    00010030 9F   000000B4'EF   FA  11B2                CALLG   $S$, @#DS$BGNSUB
                                11BD  3079
                     37   00    DA  11BD  3080          MTPR    #0,#PR$L_UB_INIT            ; initialize Unibus devices
                00000000'EF    B4  11C0  3081          CLRW    W_FLAGS                     ; clear flags word
                00000000'FF    7C  11C6  3082          CLRQ    @A_UBE_BUFFER_0             ; clear buffer quadword
          56    00000000'EF    01  11CC  3083          ADDL3   #1,A_UBE_BUFFER_0,R6        ; load memory loc address
          55    0000029F'EF    3E  11D4  3084 0$:      MOVAW   W_PTRN_TBL,R5               ; load pattern table address
                                11DB  3085 ;
                                11DB  3086 ; Set up DATIP/DATO from/to main memory location and load expected value
                                11DB  3087 ;
                     66   65    B0  11DB  3088 10$:     MOVW    (R5),(R6)                   ; load pattern
          0000005E'EF   66    01  11DE  3089          ROTL    #1,(R6),L_EXP_DATA          ; load expected data into R4
    51    0000005E'EF   01    10  11E6  3090          EXTZV   #16,#1,L_EXP_DATA,R1        ; extract end-around carry bit
               0000005E'EF   51  C8  11EF  3091          BISL2   R1,L_EXP_DATA               ; insert carry bit
               59   000002D7'EF  3E  11F6  3092          MOVAW   UBE_DATIP_1,R9              ; load command table address
                06 A9   56   01  A3  11FD  3093          SUBW3   #1,R6,6(R9)                 ; load offset within page
06 A9    09   09   00000002'EF   F0  1202  3094          INSV    FRST_VAL_MAP,#9,#9,6(R9)    ; load Map Number
                                1210  3095 ;
                                1210  3096 ; If ready bit not set then report error and abort test
                                1210  3097 ;
            29 000002BF'FF   07  E0  1210  3098 20$:     BBS     #V_RDY,@A_BE0_CR1,30$       ; br = ready bit set
                                1218  3099
    0000004E'EF    00F26010 9F   D0  1218  3100          MOVL    @#A_UBI_CSR,L_SAVE_CSR      ; save UBI CSR
                                1223  3101          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report no-RDY error
                                1223  3102                          MSGADR=T_A0_DATIP_ERR,-
                                1223  3103                          PRLINK=XFER_ERR
```

```
                00001420'EF   DF  121F                        PUSHAL  XFER_ERR
                000006EC'EF   DF  1225                        PUSHAL  T_AO_DATIP_ERR
                00000000'EF   DD  122B                        PUSHL   BE0_UNIT_NO
                          01  DD  1231                        PUSHL   #$ER
                              1233        ;;; TEST 8, SUBTEST 16, ERROR 1
            000100D0 9F   04  FB  1233                        CALLS   #$$M, @#DS$ERRHARD
                              123A  3104
                              123A  3105        $DS_ABORT       TEST                        ; abort test
                          50  D4  123A                        CLRL    R0                  ; SEND A WARNING TO THE SUPERVISOR
                          04  123C                            RET                         ; TERMINATE TEST
                              123D  3106 ;
                              123D  3107 ; Execute DATIP/DATO, wait
                              123D  3108 ;
                00001850'EF   16  123D  3109 30$:     JSB     UBE_SET_UP                  ; execute DATIP/DATO
                              1243  3110        $DS_WAITMS_S    TIME=#1                     ; wait
                          00  DD  1243                        PUSHL   #0
                          01  DD  1245                        PUSHL   #1
            00010060 9F   02  FB  1247                        CALLS   #2, @#DS$WAITMS
                              124E  3111 ;
                              124E  3112 ; Check for DATIP data or execution errors, report if any
                              124E  3113 ;
    0000004E'EF   00F26010 9F   D0  124E  3114        MOVL    @#A_UBI_CSR,L_SAVE_CSR      ; save UBI CSR
    000002AF'FF   0000005E'EF   B1  1259  3115        CMPW    L_EXP_DATA,@A_BE0_DB        ; check UBE data register
                          12    13  1264  3116        BEQL    40$                         ; br = ok
        00000000'EF    1000 8F   A8  1266  3117        BISW2   #M_DATA_ERR,W_FLAGS         ; set data error flag
              52  000002AF'FF   3C  126F  3118        MOVZWL  @A_BE0_DB,R2                ; load actual data
                          15    11  1276  3119        BRB     50$                         ; br to error report
    0000004E'EF   8001C000 8F   D3  1278  3120 40$:     BITL    #M_UBI_STS,L_SAVE_CSR       ; check UBI status bits
                          08    12  1283  3121        BNEQ    50$                         ; br = UBI error
           3B 000002BF'FF   0F   E1  1285  3122        BBC     #V_ERR,@A_BE0_CR1,60$       ; br = no UBE errors
                              128D  3123
                              128D  3124 50$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                              128D  3125                          MSGADR=T_AO_DATIP_ERR,-
                              128D  3126                          PRLINK=XFER_ERR,-
                              128D  3127                          P1=A_BE0_DB,-            ; location
                              128D  3128                          P2=L_EXP_DATA,-          ; expected value
                              128D  3129                          P3=R2,-                  ; actual value
                              128D  3130                          P4=#T_WORD_ERR,-         ; data type
                              128D  3131                          P5=FRST_VAL_MAP          ; map number used
            00000002'EF   DD  128D                            PUSHL   FRST_VAL_MAP
            00000DA0'8F   DD  1293                            PUSHL   #T_WORD_ERR
                      52  DD  1299                            PUSHL   R2
            0000005E'EF   DD  1298                            PUSHL   L_EXP_DATA
            000002AF'EF   DD  12A1                            PUSHL   A_BE0_DB
            00001420'EF   DF  12A7                            PUSHAL  XFER_ERR
            000006EC'EF   DF  12AD                            PUSHAL  T_AO_DATIP_ERR
            00000000'EF   DD  12B3                            PUSHL   BE0_UNIT_NO
                      02  DD  12B9                            PUSHL   #$ER
                              12BB        ;;; TEST 8, SUBTEST 16, ERROR 2
        000100D0 9F   09  FB  12BB                            CALLS   #$$M, @#DS$ERRHARD
                              12C2  3132
            00000000'EF   B4  12C2  3133        CLRW    W_FLAGS                         ; clear flags word
                              12C8  3134 ;
                              12C8  3135 ; Check DATO data, if error then report
                              12C8  3136 ;
    66      0000005E'EF   B1  12C8  3137 60$:     CMPW    L_EXP_DATA,(R6)             ; check data returned
                      40  13  12CF  3138        BEQL    65$                         ; br = ok
```

```
                                12D1  3139
        00000000'EF    1000 8F   A8  12D1  3140          BISW2   #M_DATA_ERR,W_FLAGS              ; set data error flag
                                12DA  3141          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                12DA  3142                          MSGADR=T_AO_DATIP_ERR,-
                                12DA  3143                          PRLINK=XFER_ERR,-
                                12DA  3144                          P1=R6,-                     ; location
                                12DA  3145                          P2=L_EXP_DATA,-             ; expected value
                                12DA  3146                          P3=(R6),-                   ; actual value
                                12DA  3147                          P4=#T_WORD_ERR,-            ; data type
                                12DA  3148                          P5=FRST_VAC_MAP             ; map number used
                00000002'EF     DD  12DA                      PUSHL   FRST_VAC_MAP
                00000DA0'8F     DD  12E0                      PUSHL   #T_WORD_ERR
                          66    DD  12E6                      PUSHL   (R6)
                0000005E'EF     DD  12E8                      PUSHL   L_EXP_DATA
                          56    DD  12EE                      PUSHL   R6
                00001420'EF     DF  12F0                      PUSHAL  XFER_ERR
                000006EC'EF     DF  12F6                      PUSHAL  T_AO_DATIP_ERR
                00000000'EF     DD  12FC                      PUSHL   BE0_UNIT_NO
                          03    DD  1302                      PUSHL   -#$ER
                                1304          ;;; TEST 8, SUBTEST 16, ERROR 3
             000100D0 9F    09   FB  1304                      CALLS   #$$M, @#DS$ERRHARD
                                130B  3149
                00000000'EF     B4  130B  3150          CLRW    W_FLAGS                         ; clear flags word
                                1311  3151 65$:        $DS_CKLOOP      10$                     ; scope loop at 10$
        00010040 9F    FEC6 CF   FA  1311                      CALLG   10$, @#DS$CKLOOP
                                131A  3152 ;
                                131A  3153 ; Adjust pointer and end subtest if patterns exhausted and both longword- and
                                131A  3154 ; word-aligned cases have been tested.
                                131A  3155 ;
                  00FF 8F    85   B1  131A  3156 70$:        CMPW    (R5)+,#^X00FF           ; patterns exhausted?
                          03    13  131F  3157          BEQL    90$                             ; br = patterns exhausted
                        FEB7    31  1321  3158          BRW     10$                             ; br = patterns not exhausted
                     03 56   01   E2  1324  3159 90$:        BBSS    #1,R6,100$              ; br = done for word alignment
                        FEA9    31  1328  3160          BRW     0$                              ; br, repeat for word alignment
                                1328  3161 100$:
                                1328  3162
                                1328  3163          $DS_ENDSUB
                                1328        T8_S16_X:
        00010038 9F    000000B4'EF   FA  1328                      CALLG   $S$, @#DS$ENDSUB
                                1336  3164
                                1336  3165          $DS_PAGE
```

```
                                1336                    .SBTTL  SUBTEST 8.17:  Map Function
                                1336  3168 ;+
                                1336  3169 ; Subtest description:
                                1336  3170 ;
                                1336  3171 ;       This subtest checks the ability of the UBI to correctly handle
                                1336  3172 ;       transactions using all useable map registers. 512 map registers are
                                1336  3173 ;       potentially useable, but of that set some may not be useable due to the
                                1336  3174 ;       presence of Unibus Memory. Therefore, before the testing of the maps is
                                1336  3175 ;       started, the Unibus is sized for memory to ascertain which of the maps
                                1336  3176 ;       are useable. A bitvector is generated by the Unibus sizer routine. If a
                                1336  3177 ;       map is useable, its bit in the bitvector is set. For each map that is
                                1336  3178 ;       available for testing, a DATI with a unique pattern is set up, executed
                                1336  3179 ;       and verified.
                                1336  3180 ;
                                1336  3181 ; Subtest steps:
                                1336  3182 ;
                                1336  3183 ;       SUBTEST MAP FUNCTION
                                1336  3184 ;       :  Invalidate all Maps
                                1336  3185 ;       :  REPEAT
                                1336  3186 ;       :  :  IF Map Entry usable
                                1336  3187 ;       :  :  :  THEN DO
                                1336  3188 ;       :  :  :  :  VALIDATE Map Entry
                                1336  3189 ;       :  :  :  :  CLEAR UBE Data register
                                1336  3190 ;       :  :  :  :  EXECUTE DATI
                                1336  3191 ;       :  :  :  :  WAIT
                                1336  3192 ;       :  :  :  :  IF error
                                1336  3193 ;       :  :  :  :  :  THEN REPORT error
                                1336  3194 ;       :  :  :  :  ENDIF
                                1336  3195 ;       :  :  :  :  INVALIDATE Map Entry
                                1336  3196 ;       :  :  :  ENDDO
                                1336  3197 ;       :  :  ENDIF
                                1336  3198 ;       :  : UNTIL All Usable Map Entries exhausted
                                1336  3199 ;       :  ENDREPEAT
                                1336  3200 ;       ENDSUBTEST MAP FUNCTION
                                1336  3201 ;
                                1336  3202 ; Errors:
                                1336  3203 ;
                                1336  3204 ;-
                                1336  3205           $DS_BGNSUB
                                1336        T8_S17::
00010030 9F   000000C0'EF   FA  1336                   CALLG   $S$, @#DS$BGNSUB
                                1341  3206
            00000000'EF     B4  1341  3207           CLRW    W_FLAGS                         ; clear flags word
                37   00     DA  1347  3208           MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
            000018FB'EF     16  134A  3209           JSB     INVAL_MAPS                      ; invalidate all Map Entries
      00000000'FF   01      D0  1350  3210           MOVL    #1,@A_UBE_BUFFER_0              ; load data into buffer
 00000000'EF   0F   09      EF  1357  3211           EXTZV   #9,#15,A_UBE_BUFFER_0,L_PAGE_NO ; extract page frame number
            0000004A'EF         135F
                                1364  3212 ;
                                1364  3213 ; If map is usable then: load address into DATI command table,load Map Entry
                                1364  3214 ; PFN, validate Map Entry
                                1364  3215 ;
                     54     D4  1364  3216           CLRL    R4                              ; clear index
00000000'FF   0000029F'EF   B0  1366  3217 10$:     MOVW    W_PTRN_TBL,@A_UBE_BUFFER_0      ; load data pattern
      03 00000006'EF   54   E0  1371  3218           BBS     R4,V_VALID_MAPS,20$             ; br = usable Map Entry
                  00EE    31  1379  3219           BRW     70$                             ; br = not usable Map Entry
         59   000002CB'EF   DE  137C  3220 20$:     MOVAL   UBE_DATI_1,R9                   ; load command table address
```

```
                06 A9   00000000'EF   B0  1383  3221          MOVW    A_UBE_BUFFER[0],6(R9)           ; load offset within page
              06 A9   09   09   54     F0  138B  3222          INSV    R4,#9,#9,6(R9)                  ; load Map Number
      00F26800 9F44    0000004A'EF     D0  1391  3223          MOVL    L_PAGE_NO,@#A_MAP_BGN[R4]       ; load Map Entry PFN
      00F26800 9F44    80000000 8F     C8  139D  3224          BISL2   #M_MAP_VAL,@#A_MAP_BGN[R4]      ; set Map Entry Valid bit
                                       13A9  3225 ;
                                       13A9  3226 ; If ready bit not set then report error and abort test
                                       13A9  3227 ;
            29 000002BF'FF     07      E0  13A9  3228 30$:     BBS     #V_RDY,@A_BE0_CR1,40$           ; br = ready bit set
                                       13B1  3229
       0000004E'EF    00F26010 9F      D0  13B1  3230          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                       13BC  3231          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report no-RDY error
                                       13BC  3232                          MSGADR=T_MAP_FCN_ERR,-
                                       13BC  3233                          PRLINK=XFER_ERR
                    00001420'EF        DF  13BC                        PUSHAL  XFER_ERR
                    0000075A'EF        DF  13C2                        PUSHAL  T_MAP_FCN_ERR
                    00000000'EF        DD  13C8                        PUSHL   BE0_UNIT_NO
                               01      DD  13CE                        PUSHL   #$ER
                                       13D0        ;;; TEST 8, SUBTEST 17, ERROR 1
                00010DD0 9F    04      FB  13D0                        CALLS   #$$M, @#DS$ERRHARD
                                       13D7  3234
                                       13D7  3235          $DS_ABORT       TEST                        ; abort test
                               50      D4  13D7                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                                       04  13D9                        RET                     ; TERMINATE TEST
                                       13DA  3236 ;
                                       13DA  3237 ; Initiate DATI transaction, wait
                                       13DA  3238 ;
                    00001850'EF        16  13DA  3239 40$:     JSB     UBE_SET_UP                      ; execute DATI
                                       13E0  3240          $DS_WAITMS_S    TIME=#1                     ; wait
                               00      DD  13E0                        PUSHL   #0
                               01      DD  13E2                        PUSHL   #1
                00010060 9F    02      FB  13E4                        CALLS   #2, @#DS$WAITMS
                                       13EB  3241 ;
                                       13EB  3242 ; Check for execution or data errors, report if any
                                       13EB  3243 ;
       0000004E'EF    00F26010 9F      D0  13EB  3244          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
       0000029F'EF    000002AF'FF      B1  13F6  3245          CMPW    @A_BE0_DB,W_PTRN_TBL            ; check UBE Data register
                                       13  13  1401  3246          BEQL    50$                         ; br = no data error
            00000000'EF    1000 8F     A8  1403  3247          BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
                52   000002AF'FF       3C  140C  3248          MOVZWL  @A_BE0_DB,R2                    ; extend actual data
                                       15  11  1413  3249          BRB     60$                         ; br to error report
       00C0004E'EF    8001C000 8F      D3  1415  3250 50$:     BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status
                                       08  12  1420  3251          BNEQ    60$                         ; br = UBI error
            37 000002BF'FF     0F      E1  1422  3252          BBC     #V_ERR,@A_BE0_CR1,65$           ; br = no UBE error
                                       142A  3253
                                       142A  3254 60$:     $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report data error
                                       142A  3255                          MSGADR=T_MAP_FCN_ERR,-
                                       142A  3256                          PRLINK=XFER_ERR,-
                                       142A  3257                          P1=A_BE0_DB,-           ; location
                                       142A  3258                          P2=W_PTRN_TBL,-         ; expected value
                                       142A  3259                          P3=R2,-                 ; actual value
                                       142A  3260                          P4=#T_WORD_ERR,-        ; data type
                                       142A  3261                          P5=R4                   ; map number used
                               54      DD  142A                        PUSHL   R4
                    00000DA0'8F        DD  142C                        PUSHL   #T_WORD_ERR
                               52      DD  1432                        PUSHL   R2
                    000002?F'EF        DD  1434                        PUSHL   W_PTRN_TBL
                    000002AF'EF        DD  143A                        PUSHL   A_BE0_DB
```

```
                00001420'EF   DF  1440                        PUSHAL  XFER_ERR
                0000075A'EF   DF  1446                        PUSHAL  T_MAP_FCN_ERR
                00000000'EF   DD  144C                        PUSHL   BE0_UNIT_NO
                          02  DD  1452                        PUSHL   #SER
                              1454        ;;; TEST 8, SUBTEST 17, ERROR 2
          00010DD0 9F   09    FB  1454                        CALLS   #S$M, @#DS$ERRHARD
                              145B  3262
                00000000'EF   B4  145B  3263        CLRW    W_FLAGS                         ; clear flags word
                              1461  3264 65$:     $DS_CKLOOP      10$                       ; scope loop at 10$
        00010040 9F   FF01 CF   FA  1461                        CALLG   10$, @#DS$CKLOOP
                              146A  3265
             00F26800 9F44    D4  146A  3266 70$:     CLRL    @#A_MAP_BGN[R4]             ; invalidate Map Entry
                          54   D6  1471  3267        INCL    R4                          ; increment into index
          00000200 8F     54   D1  1473  3268 90$:     CMPL    R4,#512                     ; done ?
                          03   13  147A  3269        BEQL    100$                    -   ; br = done
                        FEE7   31  147C  3270        BRW     10$                         ; br = not done
                              147F  3271 100$:
                              147F  3272        $DS_ENDSUB
                              147F        T8_S17_X:
    00010038 9F   000000C0'EF   FA  147F                        CALLG   $S$, @#DS$ENDSUB
                              148A  3273
                              148A  3274        $DS_PAGE
```

```
                                148A                    .SBTTL  SUBTEST 8.18:  Address Offset Map Function
                                148A  3277 ;+
                                148A  3278 ; Subtest description:
                                148A  3279 ;
                                148A  3280 ;       This subtest checks the ability of the UBI to correctly handle byte-off-
                                148A  3281 ;       set transactions using all useable map registers. 512 map registers are
                                148A  3282 ;       potentially useable, but of that set some may not be useable due to the
                                148A  3283 ;       presence of Unibus Memory. Therefore, before the testing of the maps is
                                148A  3284 ;       started, the Unibus is sized for memory to ascertain which of the maps
                                148A  3285 ;       are useable. A bitvector is generated by the Unibus sizer routine. If a
                                148A  3286 ;       map is useable, its bit in the bitvector is set. For each map that is
                                148A  3287 ;       available for testing, a DATI with a unique pattern is set up, executed
                                148A  3288 ;       and verified.
                                148A  3289 ;
                                148A  3290 ; Subtest steps:
                                148A  3291 ;
                                148A  3292 ;       SUBTEST ADDRESS OFFSET MAP FUNCTION
                                148A  3293 ;       :  Invalidate all Maps
                                148A  3294 ;       :  REPEAT
                                148A  3295 ;       :  :  IF Map Entry usable
                                148A  3296 ;       :  :  :  THEN DO
                                148A  3297 ;       :  :  :  :  VALIDATE Map Entry
                                148A  3298 ;       :  :  :  :  Set up Map Entry in Byte-offset mode
                                148A  3299 ;       :  :  :  :  CLEAR UBE Data register
                                148A  3300 ;       :  :  :  :  EXECUTE DATI
                                148A  3301 ;       :  :  :  :  WAIT
                                148A  3302 ;       :  :  :  :  IF error
                                148A  3303 ;       :  :  :  :  :  THEN REPORT error
                                148A  3304 ;       :  :  :  :  ENDIF
                                148A  3305 ;       :  :  :  :  INVALIDATE Map Entry
                                148A  3306 ;       :  :  :  ENDDO
                                148A  3307 ;       :  :  ENDIF
                                148A  3308 ;       :  : UNTIL All Usable Map Entries exhausted
                                148A  3309 ;       :  ENDREPEAT
                                148A  3310 ;       ENDSUBTEST ADDRESS OFFSET MAP FUNCTION
                                148A  3311 ;
                                148A  3312 ; Errors:
                                148A  3313 ;
                                148A  3314 ;-
                                148A  3315          $DS_BGNSUB
                                148A          T8_S18::
00010030 9F    000000CC'EF   FA  148A                  CALLG   $S$, @#DS$BGNSUB
                                1495  3316
             00000000'EF      B4  1495  3317          CLRW    W_FLAGS                         ; clear flags word
                 37   00      DA  149B  3318          MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
             000018FB'EF      16  149E  3319          JSB     INVAL_MAPS                      ; invalidate all Map Entries
  00000000'EF    0F   09      EF  14A4  3320          EXTZV   #9,#15,A_UBE_BUFFER_0,L_PAGE_NO  ; extract page frame number
             0000004A'EF          14AC
  56   00000000'EF    01      C1  14B1  3321          ADDL3   #1,A_UBE_BUFFER_0,R6            ; load address
                                14B9  3322 ;
                                14B9  3323 ; If map is usable then: load address into DATI command table,load Map Entry
                                14B9  3324 ; PFN, validate Map Entry in Byte-offset mode
                                14B9  3325 ;
                   54   D4      14B9  3326          CLRL    R4                              ; clear index
      66   0000029F'EF   D0      14BB  3327 10$:     MOVL    W_PTRN_TBL,(R6)                 ; load data into buffer
   03 00000006'EF     54   E0    14C2  3328          BBS     R4,V_VALID_MAPS,20$             ; br = usable Map Entry
                 00EE   31      14CA  3329          BRW     70$                             ; br = not usable Map Entry
```

```
                59   000002CB'EF    DE  14CD  3330 20$:    MOVAL   UBE_DATI_1,R9                    ; load command table address
              06 A9   00000000'EF    B0  14D4  3331         MOVW    A_UBE_BUFFER_0,6(R9)            ; load offset within page
           06 A9   09   09   54     F0  14DC  3332         INSV    R4,#9,#9,6(R9)                  ; load Map Number
   00F26800 9F44    0000004A'EF    D0  14E2  3333         MOVL    L_PAGE_NO,@#A_MAP_BGN[R4]       ; load Map Entry PFN
   00F26800 9F44    82000000 8F    C8  14EE  3334         BISL2   #<M_MAP_VAL!M_BYTE_OFF>,@#A_MAP_BGN[R4] ; set Map Valid & Offset
                                       14FA  3335 ;
                                       14FA  3336 ; If ready bit not set then report error and abort test
                                       14FA  3337 ;
           29 000002BF'FF    07    E0  14FA  3338 30$:    BBS     #V_RDY,@A_BE0_CR1,40$            ; br = ready bit set
                                       1502  3339
    0000004E'EF    00F26010 9F    D0  1502  3340         MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                       150D  3341         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report no-RDY error
                                       150D  3342                         MSGADR=T_MAP_FCN_ERR,-
                                       150D  3343                         PRLINK=XFER_ERR
                 00001420'EF    DF  150D                        PUSHAL  XFER_ERR
                 0000075A'EF    DF  1513                        PUSHAL  T_MAP_FCN_ERR
                 00000000'EF    DD  1519                        PUSHL   BE0_UNIT_NO
                          01    DD  151F                        PUSHL   #$ER
                                    1521        ;;; TEST 8, SUBTEST 18, ERROR 1
              000100D0 9F    04    FB  1521                        CALLS   #$$M, @#DS$ERRHARD
                                       1528  3344
                                       1528  3345         $DS_ABORT       TEST                    ; abort test
                          50    D4  1528                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04    04  152A                        RET                     ; TERMINATE TEST
                                       152B  3346 ;
                                       152B  3347 ; Initiate DATI transaction, wait
                                       152B  3348 ;
                 00001850'EF    16  152B  3349 40$:    JSB     UBE_SET_UP                      ; execute DATI
                                    1531  3350         $DS_WAITMS_S    TIME=#1                 ; wait
                          00    DD  1531                        PUSHL   #0
                          01    DD  1533                        PUSHL   #1
              00010060 9F    02    FB  1535                        CALLS   #2, @#DS$WAITMS
                                       153C  3351 ;
                                       153C  3352 ; Check for execution or data errors, report if any
                                       153C  3353 ;
    0000004E'EF    00F26010 9F    D0  153C  3354         MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
    0000029F'EF    000002AF'FF    B1  1547  3355         CMPW    @A_BE0_DB,W_PTRN_TBL            ; check UBE Data register
                          12    13  1552  3356         BEQL    50$                             ; br = no data error
         00000000'EF    1000 8F    A8  1554  3357         BISW2   #M_DATA_ERR,W_FLAGS             ; set data error flag
               52   000002AF'FF    3C  155D  3358         MOVZWL  @A_BE0_DB,R2                    ; extend actual data
                          15    11  1564  3359         BRB     60$                             ; br to error report
    0000004E'EF    8001C000 8F    D3  1566  3360 50$:    BITL    #M_UBI_STS,L_SAVE_CSR           ; check UBI status
                          08    12  1571  3361         BNEQ    60$                             ; br = UBI error
           37 000002BF'FF    0F    E1  1573  3362         BBC     #V_ERR,@A_BE0_CR1,65$           ; br = no UBE error
                                       157B  3363
                                       157B  3364 60$:    $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report data error
                                       157B  3365                         MSGADR=T_MAP_FCN_ERR,-
                                       157B  3366                         PRLINK=XFER_ERR,-
                                       157B  3367                         P1=A_BE0_DB,-            ; location
                                       157B  3368                         P2=W_PTRN_TBL,-          ; expected value
                                       157B  3369                         P3=R2,-                  ; actual value
                                       157B  3370                         P4=#T_WORD_ERR,-         ; data type
                                       157B  3371                         P5=R4                    ; map number used
                          54    DD  157B                        PUSHL   R4
                 00000DA0'8F    DD  157D                        PUSHL   #T_WORD_ERR
                          52    DD  1583                        PUSHL   R2
                 0000029F'EF    DD  1585                        PUSHL   W_PTRN_TBL
```

```
                000002AF'EF    DD  158B                        PUSHL   A_BE0_DB
                00001420'EF    DF  1591                        PUSHAL  XFER_ERR
                0000075A'EF    DF  1597                        PUSHAL  T_MAP_FCN_ERR
                00000000'EF    DD  159D                        PUSHL   BE0_UNIT_NO
                          02   DD  15A3                        PUSHL   #$ER
                               15A5            ;;; TEST 8, SUBTEST 18, ERROR 2
             00010DD0 9F   09  FB  15A5                        CALLS   #$$M, @#DS$ERRHARD
                               15AC  3372
                00000000'EF    B4  15AC  3373          CLRW    W_FLAGS                         ; clear flags word
                               15B2  3374 65$:         $DS_CKLOOP      10$                     ; scope loop at 10$
          00010040 9F   FF05 CF  FA  15B2                        CALLG   10$, @#DS$CKLOOP
                               15BB  3375
              00F26800 9F44    D4  15BB  3376 70$:     CLRL    @#A_MAP_BGN[R4]                 ; invalidate Map Entry
                          54   D6  15C2  3377          INCL    R4                              ; increment index
             00000200 8F    54   D1  15C4  3378 90$:     CMPL    R4,#512                         ; done ?
                          03   13  15CB  3379          BEQL    100$                            ; br = done
                        FEEB   31  15CD  3380          BRW     10$                             ; br = not done
                               15D0  3381 100$:
                               15D0  3382          $DS_ENDSUB
                               15D0        T8_S18_X:
      00010038 9F   000000CC'EF    FA  15D0                        CALLG   $S$, @#DS$ENDSUB
                               15DB  3383
                               15DB  3384          $DS_PAGE
```

```
                                15DB                    .SBTTL  SUBTEST 8.19:  Page Boundary Write Error
                                15DB  3387 ;+
                                15DB  3388 ; Subtest description:
                                15DB  3389 ;
                                15DB  3390 ;       This subtest checks that a byte offset write across a page boundary
                                15DB  3391 ;       will cause an error if the second page is invalidated.
                                15DB  3392 ;
                                15DB  3393 ; Subtest steps:
                                15DB  3394 ;
                                15DB  3395 ;       SUBTEST Page Boundary Write Error
                                15DB  3396 ;       :  Set up first valid Map Entry to point to valid page
                                15DB  3397 ;       :  Set up DATO to last byte in this page in byte-offset mode
                                15DB  3398 ;       :  Invalidate the page immediately following
                                15DB  3399 ;       :  Initiate DATO
                                15DB  3400 ;       :  IF no error condition generated
                                15DB  3401 ;       :  :  THEN REPORT error
                                15DB  3402 ;       :  ENDIF
                                15DB  3403 ;       :  IF UBI CSR not cleared when read
                                15DB  3404 ;       :  :  THEN REPORT error
                                15DB  3405 ;       :  ENDIF
                                15DB  3406 ;       :  IF unexpected SSYN asserted
                                15DB  3407 ;       :  :  THEN REPORT error
                                15DB  3408 ;       :  ENDIF
                                15DB  3409 ;       ENDSUBTEST Page Boundary Write Error
                                15DB  3410 ;
                                15DB  3411 ; Errors:
                                15DB  3412 ;
                                15DB  3413 ;-
                                15DB  3414          $DS_BGNSUB
                                15DB              T8_S19::
 00010030 9F   000000D8'EF   FA 15DB                       CALLG   $S$, @#DS$BGNSUB
                                15E6  3415
               00000000'EF   B4 15E6  3416          CLRW    W_FLAGS                         ; clear flags word
     00000000'EF    0080 8F   A8 15EC  3417          BISW2   #M_EXP_WR_ERR,W_FLAGS           ; set expect write error flag
                      37   00   DA 15F5  3418          MTPR    #0,#PR$_UB_INIT                 ; initialize Unibus devices
               00000000'FF   D4 15F8  3419          CLRL    @A_UBE_BUFFER_0                 ; clear buffer longword
                                15FE  3420 ;
                                15FE  3421 ; Set up Map Entry in Byte-offset mode, invalidate next entry
                                15FE  3422 ;
         54    00000002'EF   D0 15FE  3423          MOVL    FRST_VAL_MAP,R4                 ; load Map Number
 9F44    00000000'EF   0F   09   EF 1605  3424          EXTZV   #9,#15,A_UBE_BUFFER_0,@#A_MAP_BGN[R4] ; load PFN into Map Entry
                              00F26800 160F
  00F26800 9F44    80000000 8F   C8 1613  3425          BISL2   #M_MAP_VAL,@#A_MAP_BGN[R4]      ; validate Map Entry
  00F26800 9F44    02000000 8F   C8 161F  3426          BISL2   #M_BYTE_OFF,@#A_MAP_BGN[R4]     ; put in byte-offset mode
                          54   D6 162B  3427          INCL    R4                              ; incr index
              00F26800 9F44    D4 162D  3428          CLRL    @#A_MAP_BGN[R4]                 ; invalidate next Map Entry
                                1634  3429 ;
                                1634  3430 ; Load address into DATO Command table
                                1634  3431 ;
         59    00000EF'EF   DE 1634  3432          MOVAL   UBE_DATO_1,R9                   ; load command table address
      55    00000000'EF   02   C3 163B  3433          SUBL3   #2,A_UBE_BUFFER_1,R5            ; adr of last word buffer 1
                 06 A9   55   B0 1643  3434          MOVW    R5,6(R9)                        ; load offset within page
06 A9    09   09    00000002'EF   F0 1647  3435          INSV    FRST_VAL_MAP,#9,#9,6(R9)        ; load Map Number
                                1651  3436 ;
                                1651  3437 ; If ready bit not set then report error and abort test
                                1651  3438 ;
         29 000002BF'FF   07   E0 1651  3439 2$:     BBS     #V_RDY,@A_BE0_CR1,3$            ; br = ready bit set
```

```
                                        1659  3440
0000004E'EF    00F26010 9F    D0        1659  3441          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                        1664  3442          $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-      ; report no-RDY error
                                        1664  3443                          MSGADR=T_PG_BND_WR_ERR,-
                                        1664  3444                          PRLINK=XFER_ERR
                00001420'EF    DF       1664                          PUSHAL  XFER_ERR
                0000072B'EF    DF       166A                          PUSHAL  T_PG_BND_WR_ERR
                00000000'EF    DD       1670                          PUSHL   BEO_UNIT_NO
                          01   DD       1676                          PUSHL   #$ER
                                        1678      ;;; TEST 8, SUBTEST 19, ERROR 1
              00010DD0 9F      04   FB  1678                          CALLS   #$$M, @#DS$ERRHARD
                                        167F  3445
                                        167F  3446          $DS_ABORT       TEST                    ; abort test
                          50   D4       167F                          CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                               04       1681                          RET                     ; TERMINATE TEST
                                        1682  3447 ;
                                        1682  3448 ; Execute DATO, wait
                                        1682  3449 ;
                00001850'EF    16       1682  3450 3$:       JSB     UBE_SET_UP                      ; execute DATIP/DATO
                                        1688  3451          $DS_WAITMS_S    TIME=#1                 ; wait
                          00   DD       1688                          PUSHL   #0
                          01   DD       168A                          PUSHL   #1
              00010060 9F      02   FB  168C                          CALLS   #2, @#DS$WAITMS
                                        1693  3452 ;
                                        1693  3453 ; Check that Write Error bit in UBI CSR set and cleared when read
                                        1693  3454 ;
0000004E'EF    00F26010 9F    D0        1693  3455 7$:       MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
       15 0000004E'EF         0E   E1  169E  3456          BBC     #V_UBI_WR_ERR,L_SAVE_CSR,8$     ; br = Write err bit never set
       0D 000002BB'FF         08   E1  16A6  3457          BBC     #V_NO_SSYN,@A_BEO_CR2,8$        ; br = unxp SSYN received
00F26010 9F    8001C000 8F    D3       16AE  3458          BITL    #M_UBI_STS,@#A_UBI_CSR          ; did CSR clear when read?
                          1B   13       16B9  3459          BEQL    9$                              ; br = CSR clear
                                        16BB  3460
                                        16BB  3461 8$:       $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-      ; report CSR bit error
                                        16BB  3462                          MSGADR=T_PG_BND_WR_ERR,-
                                        16BB  3463                          PRLINK=CSR_ERR
                000015DF'EF    DF       16BB                          PUSHAL  CSR_ERR
                0000072B'EF    DF       16C1                          PUSHAL  T_PG_BND_WR_ERR
                00000000'EF    DD       16C7                          PUSHL   BEO_UNIT_NO
                          02   DD       16CD                          PUSHL   #$ER
                                        16CF      ;;; TEST 8, SUBTEST 19, ERROR 2
              00010DD0 9F      04   FB  16CF                          CALLS   #$$M, @#DS$ERRHARD
                                        16D6  3464 9$:
                                        16D6  3465          $DS_ENDSUB
                                        16D6      T8_S19_X:
00010038 9F    000000D8'EF      FA       16D6                          CALLG   $S$, @#DS$ENDSUB
                                        16E1  3466
                                        16F1  3467          $DS_PAGE
```

```
                                16E1                    .SBTTL  SUBTEST 8.20:  Map Parity Error
                                16E1  3470 ;+
                                16E1  3471 ; Subtest description:
                                16E1  3472 ;
                                16E1  3473 ;       This subtest checks that when the Unibus PB is asserted (simulating
                                16E1  3474 ;       a Map Parity Error) and a transfer is attempted, the Map Parity Error
                                16E1  3475 ;       bit in the UBI CSR will set and that it will clear when read.
                                16E1  3476 ;
                                16E1  3477 ; Subtest steps:
                                16E1  3478 ;
                                16E1  3479 ;       SUBTEST Map Parity Error
                                16E1  3480 ;       :  Set FORCE TB PARITY in Memory CSR
                                16E1  3481 ;       :  LOAD Map Entry
                                16E1  3482 ;       :  CLEAR TB Parity bit
                                16E1  3483 ;       :  EXECUTE DATI
                                16E1  3484 ;       :  WAIT
                                16E1  3485 ;       :  IF Map Parity Error bit not set
                                16E1  3486 ;       :  :  THEN REPORT error
                                16E1  3487 ;       :  ENDIF
                                16E1  3488 ;       :  IF Map Parity Error did not clear when read
                                16E1  3489 ;       :  :  THEN REPORT error
                                16E1  3490 ;       :  ENDIF
                                16E1  3491 ;       :  IF unexpected SSYN asserted
                                16E1  3492 ;       :  :  THEN REPORT error
                                16E1  3493 ;       :  ENDIF
                                16E1  3494 ;       ENDSUBTEST Map Parity Error
                                16E1  3495 ;
                                16E1  3496 ; Errors:
                                16E1  3497 ;
                                16E1  3498 ;-
                                16E1  3499          $DS_BGNSUB
                                16E1            T8_S20::
00010030 9F   000000E4'EF   FA  16E1                        CALLG   $S$, @#DS$BGNSUB
                                16EC  3500
              37   00       DA  16EC  3501          MTPR    #0,#PR$L_UB_INIT                ; initialize Unibus devices
          00000000'EF      B4  16EF  3502          CLRW    W_FLAGS                         ; clear flags word
          00000000'FF      D4  16F5  3503          CLRL    @A_UBE_BUFFER_0                 ; clear buffer longword
    00000000'EF   0400 8F  A8  16FB  3504          BISW2   #M_EXP_PB_ERR,W_FLAGS           ; set expect PB error flag
                                1704  3505 ;
                                1704  3506 ; If Ready bit not set then report error and abort test
                                1704  3507 ;
     29 000002BF'FF   07   E0  1704  3508          BBS     #V_RDY,@A_BE0_CR1,4$            ; br = ready bit set
                                170C  3509
0000004E'EF   00F26010 9F   D0  170C  3510          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                1717  3511          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report no-RDY error
                                1717  3512                          MSGADR=T_PB_ERR,-
                                1717  3513                          PRLINK=XFER_ERR
          00001420'EF      DF  1717                        PUSHAL  XFER_ERR
          00000786'EF      DF  171D                        PUSHAL  T_PB_ERR
          00000000'EF      DD  1723                        PUSHL   BE0_UNIT_NO
                   01      DD  1729                        PUSHL   #$ER
                               172B              ;;; TEST 8, SUBTEST 20, ERROR 1
      000100D0 9F   04   FB  172B                        CALLS   #$$M, @#DS$ERRHARD
                                1732  3514
                                1732  3515          $DS_ABORT       TEST                    ; abort test
                   50   D4  1732                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                        04  1734                        RET                     ; TERMINATE TEST
```

```
                                        1735  3516 ;
                                        1735  3517 ; Simulate Map Parity conditions
                                        1735  3518 ;
    00F20004 9F    20000000 8F    C8    1735  3519 4$:     BISL2   #M_FRCE_TB_ERR,@#A_MEM_CSR_1    ; set FORCE TB PARITY bit
        00000000'EF    0F    09   EF    1740  3520         EXTZV   #9,#15,A_UBE_BUFFER_0,L_PAGE_NO ; extract Page Frame Number
                    0000004A'EF          1748
    0000004A'EF    80000000 8F    C8    174D  3521         BISL2   #M_MAP_VAL,L_PAGE_NO           ; set map valid bit
              54   00000002'EF    D0    1758  3522         MOVL    FRST_VAL_MAP,R4                ; load first valid map number
  00F26800 9F44    0000004A'EF    D0    175F  3523         MOVL    L_PAGE_NO,@#A_MAP_BGN[R4]      ; load Map Entry
    00F20004 9F    20000000 8F    CA    176B  3524         BICL2   #M_FRCE_TB_ERR,@#A_MEM_CSR_1    ; clear FORCE TB PARITY bit
                                        1776  3525 ;
                                        1776  3526 ; Load address into DATI command table.
                                        1776  3527 ;
                          58    D4    1776  3528         CLRL    R8                             ; clear device number
              59   000002CB'EF    DE    1778  3529         MOVAL   UBE_DATI_1,R9                  ; load command table address
 06 A9    09    09   00000002'EF    F0    177F  3530         INSV    FRST_VAL_MAP,#9,#9,6(R9)       ; load Map Number
                                        1789  3531 ;
                                        1789  3532 ; Execute DATI
                                        1789  3533 ;
                    00001850'EF    16    1789  3534 5$:     JSB     UBE_SET_UP
                                        178F  3535         $DS_WAITMS_S    TIME=#1                ; wait
                          00    DD    178F                   PUSHL   #0
                          01    DD    1791                   PUSHL   #1
                00010060 9F    02    FB    1793                   CALLS   #2, @#DS$WAITMS
                                        179A  3536 ;
                                        179A  3537 ; Check for execution errors, report if any
                                        179A  3538 ;
    0000004E'EF    00F26010 9F    D0    179A  3539 7$:     MOVL    @#A_UBI_CSR,L_SAVE_CSR         ; save UBI CSR
          15 0000004E'EF    0F    E1    17A5  3540         BBC     #V_UBI_PB_ERR,L_SAVE_CSR,8$    ; br = Par bit never set
          0D 000002BB'FF    08    E1    17AD  3541         BBC     #V_NO_SSYN,@A_BEO_CR2,8$       ; br = unxp SSYN received
00F26010 9F    8001C000 8F    D3    17B5  3542         BITL    #M_UBI_STS,@#A_UBI_CSR         ; did CSR clear when read?
                          1B    13    17C0  3543         BEQL    9$                             ; br = CSR clear
                                        17C2  3544
                                        17C2  3545 8$:     $DS_ERRHARD_S   UNIT=BEO_UNIT_NO,-     ; report CSR bit error
                                        17C2  3546                         MSGADR=T_PB_ERR,-
                                        17C2  3547                         PRLINK=CSR_ERR
                    000015DF'EF    DF    17C2                   PUSHAL  CSR_ERR
                    00000786'EF    DF    17C8                   PUSHAL  T_PB_ERR
                    00000000'EF    DD    17CE                   PUSHL   BEO_UNIT_NO
                          02    DD    17D4                   PUSHL   #$ER
                                        17D6       ;;; TEST 8, SUBTEST 20, ERROR 2
                00010DD0 9F    04    FB    17D6                   CALLS   #$$M, @#DS$ERRHARD
                                        17DD  3548
                                        17DD  3549 9$:     $DS_ENDSUB
                                        17DD       T8_S20_X:
00010038 9F    000000E4'EF    FA    17DD                   CALLG   $S$, @#DS$ENDSUB
                                        17E8  3550
                                        17E8  3551         $DS_PAGE
```

```
                                17E8                    .SBTTL  SUBTEST 8.21:  NXM Error
                                17E8  3554 ;+
                                17E8  3555 ; Subtest description:
                                17E8  3556 ;
                                17E8  3557 ;       This subtest checks that when the UBE attempts to access a non-
                                17E8  3558 ;       existent memory location, the NXM Error bit in the UBI CSR will set
                                17E8  3559 ;       and that it will clear when read.
                                17E8  3560 ;
                                17E8  3561 ; Subtest steps:
                                17E8  3562 ;
                                17E8  3563 ;       SUBTEST NXM Error
                                17E8  3564 ;       :  Set up DATI to access NXM address
                                17E8  3565 ;       :  Initiate DATI
                                17E8  3566 ;       :  WAIT
                                17E8  3567 ;       :  IF NXM bit did not set
                                17E8  3568 ;       :  :  THEN REPORT error
                                17E8  3569 ;       :  ENDIF
                                17E8  3570 ;       :  IF NXM bit did not clear when read
                                17E8  3571 ;       :  :  THEN REPORT error
                                17E8  3572 ;       :  ENDIF
                                17E8  3573 ;       :  IF unexpected SSYN asserted
                                17E8  3574 ;       :  :  THEN REPORT error
                                17E8  3575 ;       :  ENDIF
                                17E8  3576 ;       ENDSUBTEST NXM Error
                                17E8  3577 ;
                                17E8  3578 ; Errors:
                                17E8  3579 ;
                                17E8  3580 ;-
                                17E8  3581          $DS_BGNSUB
                                17E8        T8_S21::
00010030 9F   000000F0'EF   FA  17E8                CALLG   $S$, @#DS$BGNSUB
                                17F3  3582
                     37   00  DA  17F3  3583          MTPR    #0,#PR$L_UB_INIT        ; initialize Unibus devices
                00000000'FF  D4  17F6  3584          CLRL    @A_UBE_BUFFER_0         ; clear buffer longword
                00000000'EF  B4  17FC  3585          CLRW    W_FLAGS                 ; clear flags word
     00000000'EF    0200 8F  A8  1802  3586          BISW2   #M_EXP_NXM_ERR,W_FLAGS  ; set expect NXM error flag
          55   00000002'EF  D0  180B  3587          MOVL    FRST_VAL_MAP,R5         ; load first valid Map Number
                                1812  3588 ;
                                1812  3589 ; Set up DATI from non-existent memory location
                                1812  3590 ;
          59   000002CB'EF  3E  1812  3591          MOVAW   UBE_DATI_1,R9           ; load adr of DATI command tbl
        06 A9  00000086'EF  B0  1819  3592          MOVW    L_NXM_ADR,6(R9)         ; load offset within page
06 A9   09   09  00000002'EF  F0  1821  3593          INSV    FRST_VAL_MAP,#9,#9,6(R9) ; load Map Number
     00000086'EF   0F   09  EF  182B  3594          EXTZV   #9,#15,L_NXM_ADR,L_PAGE_NO ; extract page frame number
                0000004A'EF       1833
  00F26800 9F45    0000004A'EF  D0  1838  3595          MOVL    L_PAGE_NO,@#A_MAP_BGN[R5] ; load Map Entry PFN
  00F26800 9F45    80000000 8F  C8  1844  3596          BISL2   #M_MAP_VAL,@#A_MAP_BGN[R5] ; set Map Valid bit
                                1850  3597 ;
                                1850  3598 ; If ready bit not set then report error and abort test
                                1850  3599 ;
        29 000002BF'FF   07  E0  1850  3600 4$:       BBS     #V_RDY,@A_BE0_CR1,5$    ; br = ready bit set
                                1858  3601
    0000004E'EF   00F26010 9F  D0  1858  3602          MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
                                1863  3603          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,- ; report no-RDY error
                                1863  3604                          MSGADR=T_NXM_ERR,-
                                1863  3605                          PRLINK=XFER_ERR
                00001420'EF   DF  1863                      PUSHAL  XFER_ERR
```

```
                000007B6'EF   DF  1869                        PUSHAL  T_NXM_ERR
                00000000'EF   DD  186F                        PUSHL   BE0_UNIT_NO
                          01  DD  1875                        PUSHL   #$ER
                              1877        ;;; TEST 8, SUBTEST 21, ERROR 1
            000100D0 9F   04  FB  1877                        CALLS   #$$M, @#DS$ERRHARD
                              187E  3606
                              187E  3607        $DS_ABORT       TEST                    ; abort test
                          50  D4  187E                        CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                          04  1880                            RET                     ; TERMINATE TEST
                              1881  3608 ;
                              1881  3609 ; Execute DATI, wait
                              1881  3610 ;
                00001850'EF   16  1881  3611 5$:      JSB     UBE_SET_UP              ; execute DATI
                              1887  3612        $DS_WAITMS_S    TIME=#1                 ; wait
                          00  DD  1887                        PUSHL   #0
                          01  DD  1889                        PUSHL   #1
            00010060 9F   02  FB  188B                        CALLS   #2, @#DS$WAITMS
                              1892  3613 ;
                              1892  3614 ; Check that NXM bit set and cleared when read
                              1892  3615 ;
    0000004E'EF   00F26010 9F   D0  1892  3616 7$:      MOVL    @#A_UBI_CSR,L_SAVE_CSR  ; save UBI CSR
           15 0000004E'EF   10  E1  189D  3617          BBC     #V_UBI_NXM_ERR,L_SAVE_CSR,8$ ; br = NXM bit never set
           0D 00000288'FF   08  E1  18A5  3618          BBC     #V_NO_SSYN,@A_BE0_CR2,8$ ; br = unxp SSYN received
    00F26010 9F   8001C000 8F   D3  18AD  3619          BITL    #M_UBI_STS,@#A_UBI_CSR  ; did CSR clear when read?
                          1B  13  18B8  3620          BEQL    9$                      ; br = CSR clear
                              18BA  3621
                              18BA  3622 8$:      $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-   ; report execution error
                              18BA  3623                          MSGADR=T_NXM_ERR,-
                              18BA  3624                          PRLINK=CSR_ERR
                000015DF'EF   DF  18BA                            PUSHAL  CSR_ERR
                000007B6'EF   DF  18C0                            PUSHAL  T_NXM_ERR
                00000000'EF   DD  18C6                            PUSHL   BE0_UNIT_NO
                          02  DD  18CC                            PUSHL   #$ER
                              18CE        ;;; TEST 8, SUBTEST 21, ERROR 2
            000100D0 9F   04  FB  18CE                            CALLS   #$$M, @#DS$ERRHARD
                              18D5  3625
                              18D5  3626 9$:      $DS_ENDSUB
                              18D5        T8_S21_X:
    00010038 9F   000000F0'EF   FA  18D5                            CALLG   $S$, @#DS$ENDSUB
                              18E0  3627
                              18E0  3628        $DS_PAGE
```

```
                                18E0                    .SBTTL  SUBTEST 8.22:  Map Invalid
                                18E0  3631 ;+
                                18E0  3632 ; Subtest description:
                                18E0  3633 ;
                                18E0  3634 ;       This subtest checks that the UBI refuses to issue SSYN if the UBE at-
                                18E0  3635 ;       tempts to access a map entry which has the valid bit turned off.
                                18E0  3636 ;
                                18E0  3637 ; Subtest steps:
                                18E0  3638 ;
                                18E0  3639 ;       SUBTEST Map Invalid
                                18E0  3640 ;       :  Invalidate first valid Map Entry
                                18E0  3641 ;       :  Set up DATI to reference first valid Map Entry
                                18E0  3642 ;       :  CLEAR UBE Data register
                                18E0  3643 ;       :  Load pattern into memory location
                                18E0  3644 ;       :  Execute DATI
                                18E0  3645 ;       :  WAIT
                                18E0  3646 ;       :  IF SSYN received
                                18E0  3647 ;       :  :  THEN REPORT no-timeout error
                                18E0  3648 ;       :  ENDIF
                                18E0  3649 ;       ENDSUBTEST Map Invalid
                                18E0  3650 ;
                                18E0  3651 ; Errors:
                                18E0  3652 ;
                                18E0  3653 ;-
                                18E0  3654              $DS_BGNSUB
                                18E0                T8_S22::
00010030 9F   000000FC'EF   FA  18E0                        CALLG   $S$, @#DS$BGNSUB
                                18EB  3655
            00000000'FF     D4  18EB  3656              CLRL    @A_UBE_BUFFER_0                 ; clear buffer longword
            00000000'EF     B4  18F1  3657              CLRW    W_FLAGS                         ; clear flags word
                                18F7  3658 ;
                                18F7  3659 ; Set up DATI to reference invalid Map Entry
                                18F7  3660 ;
      54    00000002'EF     D0  18F7  3661              MOVL    FRST_VAL_MAP,R4                 ; load first valid Map Number
         00F26800 9F44     D4  18FE  3662              CLRL    @#A_MAP_BGN[R4]                 ; invalidate Map Entry
      59    000002CB'EF     3E  1905  3663              MOVAW   UBE_DATI_1,R9                   ; load command table address
   06 A9    00000000'EF     B0  190C  3664              MOVW    A_UBE_BUFFER_0,6(R9)            ; load offset within page
06 A9   09    09    54      F0  1914  3665              INSV    R4,#9,#9,6(R9)                  ; load Map Number
                                191A  3666 ;
                                191A  3667 ; If ready bit not set then report error and abort test
                                191A  3668 ;
   29 000002BF'FF     07      E0  191A  3669 4$:         BBS     #V_RDY,@A_BE0_CR1,5$            ; br = ready bit set
                                1922  3670
0000004E'EF   00F26010 9F      DO  1922  3671              MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
                                192D  3672              $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-      ; report no-RDY error
                                192D  3673                              MSGADR=T_MAP_INV_ERR,-
                                192D  3674                              PRLINK=XFER_ERR
            00001420'EF     DF  192D                            PUSHAL  XFER_ERR
            0000080D'EF     DF  1933                            PUSHAL  T_MAP_INV_ERR
            00000000'EF     DD  1939                            PUSHL   BE0_UNIT_NO
                       01   DD  193F                            PUSHL   #$ER
                                1941            ;;; TEST 8, SUBTEST 22, ERROR 1
      000100D0 9F      04  FB  1941                            CALLS   #$$M, @#DS$ERRHARD
                                1948  3675
                                1948  3676              $DS_ABORT       TEST                    ; abort test
                       50   D4  1948                            CLRL    R0              ; SEND A WARNING TO THE SUPERVISOR
                            04  194A                            RET                     ; TERMINATE TEST
```

```
                                    194B  3677 ;
                                    194B  3678 ; Execute DATI, wait
                                    194B  3679 ;
                 00001850'EF   16   194B  3680 5$:     JSB     UBE_SET_UP                      ; Execute DATI
                                    1951  3681          $DS_WAITMS_S    TIME=#1                 ; wait
                         00    DD   1951                PUSHL   #0
                         01    DD   1953                PUSHL   #1
               00010060 9F     02   FB   1955                CALLS   #2, @#DS$WAITMS
                                    195C  3682 ;
                                    195C  3683 ; If SSYN asserted then report error
                                    195C  3684 ;
     0000004E'EF   00F26010 9F     D0   195C  3685          MOVL    @#A_UBI_CSR,L_SAVE_CSR          ; save UBI CSR
           1B 000002BB'FF     08   E0   1967  3686          BBS     #V_NO_SSYN,@A_BE0_CR2,7$        ; br = SSYN not asserted
                                    196F  3687
                                    196F  3688          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-       ; report execution error
                                    196F  3689                          MSGADR=T_MAP_INV_ERR,-
                                    196F  3690                          PRLINK=CSR_ERR
                 000015DF'EF       DF   196F                PUSHAL  CSR_ERR
                 0000080D'EF       DF   1975                PUSHAL  T_MAP_INV_ERR
                 00000000'EF       DD   197B                PUSHL   BE0_UNIT_NO
                         02        DD   1981                PUSHL   #$ER
                                    1983          ;;; TEST 8, SUBTEST 22, ERROR 2
               00010DD0 9F     04   FB   1983                CALLS   #$$M, @#DS$ERRHARD
                                    198A  3691
                                    198A  3692 7$:
                                    198A  3693          $DS_ENDSUB
                                    198A          T8_S22_X:
     00010038 9F     000000FC'EF   FA   198A                CALLG   $S$, @#DS$ENDSUB
                                    1995  3694
                                    1995  3695          $DS_ENDTEST
                         50    01   D0   1995                MOVL    #1, R0                  ; NORMAL EXIT
                                    1998          TEST_008_X::
               00010058 9F     6E   FA   1998                CALLG   (SP), @#DS$BREAK
                               04   199F                    RET                             ; RETURN TO TEST SEQUENCER
                                    19A0  3696
                                    19A0  3697          $DS_CHANNEL_S   UNIT=BE0_UNIT_NO,-       ; disable interrupts
                                    19A0  3698                          FUNC=#CHC$_DSINT,-
                                    19A0  3699                          VECADR=BR_SERVICE,-
                                    19A0  3700                          STSADR=L_UBI_STATUS
                 000000AF'EF       7F   19A0                PUSHAQ  L_UBI_STATUS
                 00001A3C'EF       DF   19A6                PUSHAL  BR_SERVICE
                 00000000'8F       DD   19AC                PUSHL   #CHC$_DSINT
                 00000000'EF       DD   19B2                PUSHL   BE0_UNIT_NO
               000'0180 9F     04   FB   19B8                CALLS   #4, @#DS$CHANNEL
                                    19BF  3701
                                    19BF  3702          $DS_PAGE
```

```
                     19BF                       .SBTTL  TEST 9:  UBE Multi-transfer Test
                 00000000                               .PSECT  TEST_009, PAGE, NOWRT
                     001C
                     001C  3705 ;++
                     001C  3706 ; Test description:
                     001C  3707 ;
                     001C  3708 ;       This test supports up to four UBE's if they are attached and selected
                     001C  3709 ;       for testing. It consists of two subtests, the first is a multi-transfer
                     001C  3710 ;       subtest that starts traffic on the bus simultaneously and the second
                     001C  3711 ;       is a multi-transfer subtest that will have UBE's doing various trans-
                     001C  3712 ;       fers and/or interrupts at different request levels. Both subtests are
                     001C  3713 ;       designed to check CPU handling capabilities. This test is modeled after
                     001C  3714 ;       the existing PDP-11 Unibus System Exerciser, CZKUAE0.  Note that the
                     001C  3715 ;       UBEs MUST be addressed and vectored as specified in the BUS EXERCISER
                     001C  3716 ;       ENGINEERING SPECIFICATION, i.e., the first device must be at address
                     001C  3717 ;       777000 (octal) with vector 510 (octal), the second at address 777020
                     001C  3718 ;       (octal) with vector 514 (octal), etc.
                     001C  3719 ;
                     001C  3720 ;--
                     001C  3721
                     001C  3722         $DS_BGNTEST     <DEFAULT,UBI,ALL>
                     001C           DATA_009:
                 00000000  001C                 .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                     0020           TEST_009::
                     0000  0020                 .WORD   ^M<>            ; ENTRY MASK
                     0022  3723
                     0022  3724 ;
                     0022  3725 ; Report if no devices selected and exit test
                     0022  3726 ;
          00000000'EF  D5  0022  3727         TSTL    L_LAST_UBE                  ; 4 or more UBEs selected?
                       10  18  0028  3728         BGEQ    10$                         ; br = selected
                           002A  3729         $DS_PRINTF_S    FORMAT=T_NO_UBE         ; print warning
          00000414'EF  9F  002A                 PUSHAB  T_NO_UBE
     000100F0 9F   01  FB  0030                 CALLS   #$$N, @#DS$PRINTF
                     0037  3730         $DS_EXIT        TEST                        ; exit test if no UBE
                     0635  31  0037                 BRW     TEST_009_X      ; EXIT TEST 9
                     003A  3731 ;
                     003A  3732 ; Size Unibus space
                     003A  3733 ;
          00001867'EF  16  003A  3734 10$:    JSB     UNIBUS_SIZER
                     0040  3735 ;
                     0040  3736 ; Set up UBE interrupt vectors
                     0040  3737 ;
                     0040  3738         $DS_SETVEC_S    VECTOR=#SCB$L_BE0,SRVADR=BE0_SERVICE
                       00  DD  0040                 PUSHL   #0
          00001A74'EF  DF  0042                 PUSHAL  BE0_SERVICE
          00000348 8F  DD  0048                 PUSHL   #SCB$L_BE0
     00010160 9F   03  FB  004E                 CALLS   #3, @#DS$SETVEC
                     0055  3739         $DS_SETVEC_S    VECTOR=#SCB$L_BE1,SRVADR=BE0_SERVICE
                       00  DD  0055                 PUSHL   #0
          00001A74'EF  DF  0057                 PUSHAL  BE0_SERVICE
          0000034C 8F  DD  005D                 PUSHL   #SCB$L_BE1
     00010160 9F   03  FB  0063                 CALLS   #3, @#DS$SETVEC
                     006A  3740         $DS_SETVEC_S    VECTOR=#SCB$L_BE2,SRVADR=BE0_SERVICE
                       00  DD  006A                 PUSHL   #0
          00001A74'EF  DF  006C                 PUSHAL  BE0_SERVICE
          00000350 8F  DD  0072                 PUSHL   #SCB$L_BE2
```

```
00010160 9F   03   FB  0078                        CALLS   #3, @#DS$SETVEC
                      007F  3741          $DS_SETVEC_S    VECTOR=#SCB$L_BE3,SRVADR=BE0_SERVICE
                   00   DD  007F                        PUSHL   #0
         00001A74'EF    DF  0081                        PUSHAL  BE0_SERVICE
         00000354 8F    DD  0087                        PUSHL   #SCB$L_BE3
00010160 9F   03   FB  008D                        CALLS   #3, @#DS$SETVEC
                      0094  3742 ;
                      0094  3743 ; Set up Interval Timer vector
                      0094  3744 ;
                      0094  3745          $DS_SETVEC_S    VECTOR=#SCB$L_TIMER,SRVADR=TIMER_SERVICE
                   00   DD  0094                        PUSHL   #0
         00001AA4'EF    DF  0096                        PUSHAL  TIMER_SERVICE
         000000C0 8F    DD  009C                        PUSHL   #SCB$L_TIMER
00010160 9F   03   FB  00A2                        CALLS   #3, @#DS$SETVEC
                      00A9  3746 ;
                      00A9  3747 ; Load actual byte-offset buffer addresses into buffer address table
                      00A9  3748 ;
                   58   D4  00A9  3749          CLRL    R8                      ; clear index
      51   000000CB'EF48   D0  00AB  3750 20$:    MOVL    A_BUFRS_DFR_1[R8],R1    ; load deferred address
000000EB'EF48   61   01   C1  00B3  3751          ADDL3   #1,(R1),A_BUFFERS_1[R8] ; load actual byte-offset address
            EB 58   08   F2  00BC  3752          AOBLSS  #8,R8,20$               ; incr index, br = not done
                      00C0  3753
                      00C0  3754
                      00C0  3755          $DS_PAGE
```

```
                                00C0                    .SBTTL  SUBTEST 9.1:  UBE Multi-transfer 1
                                00C0  3758 ;+
                                00C0  3759 ; Subtest description:
                                00C0  3760 ;
                                00C0  3761 ;       This subtest will attempt to create traffic on the Unibus and check
                                00C0  3762 ;       that the CPU can handle the data transactions without errors. All se-
                                00C0  3763 ;       lected UBE's are set up and started simultaneously to cause the bus
                                00C0  3764 ;       traffic. The functions cycle through the map entries using byte offset
                                00C0  3765 ;       mode and are set up as follows:
                                00C0  3766 ;
                                00C0  3767 ;       UBE 0 - DATIP/DATOB single word BR6 transfer with a rotate data
                                00C0  3768 ;                       left after DATIP, interrupt on done, 2000 cycles.
                                00C0  3769 ;
                                00C0  3770 ;       UBE 1 - DATIP/DATO single word NPR transfer without rotate,
                                00C0  3771 ;                       interrupt on done BR7, 2000 cycles.
                                00C0  3772 ;
                                00C0  3773 ;       UBE 2 - DATO single word NPR transfer, interrupt on done BR7,
                                00C0  3774 ;                       1000 cycles.
                                00C0  3775 ;
                                00C0  3776 ;       UBE 3 - DATO single word BR5 transfer, interrupt on done BR5,
                                00C0  3777 ;                       1000 cycles.
                                00C0  3778 ;
                                00C0  3779 ; Subtest steps:
                                00C0  3780 ;
                                00C0  3781 ;       BGNSUBTEST Multitransfer 1
                                00C0  3782 ;       :  DO FOR UBE Number = {0,1,2,3}
                                00C0  3783 ;       :  :  PROBE Data Register
                                00C0  3784 ;       :  :  IF NO ACCESS
                                00C0  3785 ;       :  :  :  THEN DO
                                00C0  3786 ;       :  :  :  :  REPORT ERROR
                                00C0  3787 ;       :  :  :  :  EXIT Test
                                00C0  3788 ;       :  :  :  ENDDO
                                00C0  3789 ;       :  :  ENDIF
                                00C0  3790 ;       :  ENDDO
                                00C0  3791 ;       :  DO FOR UBE Number = {0,1,2,3}
                                00C0  3792 ;       :  :  Set up UBE transfer
                                00C0  3793 ;       :  ENDDO
                                00C0  3794 ;       :  START Timer
                                00C0  3795 ;       :  SET Simultaneous GO bit
                                00C0  3796 ;       :  WAIT UNTIL All transfers stopped
                                00C0  3797 ;       :  DO FOR UBE Number = {0,1,2,3}
                                00C0  3798 ;       :  :  IF Data or Execution error
                                00C0  3799 ;       :  :  :  THEN REPORT error
                                00C0  3800 ;       :  :  ENDIF
                                00C0  3801 ;       :  ENDDO
                                00C0  3802 ;       ENDSUBTEST Multitransfer 1
                                00C0  3803 ;
                                00C0  3804 ; Errors:
                                00C0  3805 ;
                                00C0  3806 ;-
                                00C0  3807              $DS_BGNSUB
                                00C0            T9_S1::
00010030 9F   00000108'EF   FA  00C0                    CALLG   $S$, @#DS$BGNSUB
                                00CB  3808
                     37   01  DA  00CB  3809          MTPR    #1,#PR$L_UB_INIT        ; initialize Unibus devices
                0000008A'EF  D4  00CE  3810          CLRL    L_MAP_NUM               ; clear initial map number
0000008E'EF   33550FFF  8F  D0  00D4  3811          MOVL    #L_PATTERN,L_PTRN       ; load initial pattern
```

```
                                00DF  3812 ;
                                00DF  3813 ; Attempt access to Data Register of each UBE.  Report error and exit if
                                00DF  3814 ; not accessible.
                                00DF  3815 ;
                          58    D4  00DF  3816          CLRL    R8                              ; clear device number
                57  000002AF'EF    D0  00E1  3817          MOVL    A_BE0_DB,R7                     ; load first UBE address
                                00E8  3818 1$:      $DS_PROBE_S     UNIT=BE0_UNIT_NO[R8],-  ; probe data register
                                00E8  3819                      LENGTH=#2,-              ; word access
                                00E8  3820                      ADDRESS=(R7)
             00000000'EF48    DD  00E8                        PUSHL   BE0_UNIT_NO[R8]
                          02    DD  00EF                        PUSHL   #2
                          67    9F  00F1                        PUSHAB  (R7)
          000101A0 9F     03    FB  00F3                        CALLS   #3, @#DS$PROBE
                                00FA  3821
                                00FA  3822          $DS_BNERROR     3$
                          50    DD  00FA                        PUSHL   R0
                          01    DD  00FC                        PUSHL   #$ER
               00000000'8F      DD  00FE                        PUSHL   #DS$K_BNERROR
          000100A8 9F     03    FB  0104                        CALLS   #3, @#DS$BRANCH
                       21 50    E8  010B                        BLBS    R0, 3$
                                010E  3823
                                010E  3824          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO[R8],-  ; report no-access error
                                010E  3825                      MSGADR=T_NO_ACCESS,-
                                010E  3826                      PRLINK=NO_ACCESS,-
                                010E  3827                      P1=R7
                          57    DD  010E                        PUSHL   R7
                0000171B'EF     DF  0110                        PUSHAL  NO_ACCESS
                000003C3'EF     DF  0116                        PUSHAL  T_NO_ACCESS
             00000000'EF48    DD  011C                        PUSHL   BE0_UNIT_NO[R8]
                          01    DD  0123                        PUSHL   #$ER
                                0125        ;;; TEST 9, SUBTEST 1, ERROR 1
                000100D0 9F     05    FB  0125                        CALLS   #$$M, @#DS$ERRHARD
                                012C  3828          $DS_EXIT        TEST                    ; exit test
                             0540    31  012C                        BRW     TEST_009_X      ; EXIT TEST 9
                          57   10    C0  012F  3829 3$:      ADDL2   #^X10,R7                ; incr to next UBE
                   AE 58    00000000'EF    F3  0132  3830          AOBLEQ  L_LAST_UBE,R8,1$        ; incr dev num, br = not done
                                013A  3831 ;
                                013A  3832 ; Load buffer, set up for next transfer.
                                013A  3833 ;
                                013A  3834 9$:      $DS_BREAK                               ; look for control-c
                000100058 9F    6E    FA  013A                        CALLG   (SP), @#DS$BREAK
                          58    D4  0141  3835          CLRL    R8                          ; clear device number
                     000000A2'EF    94  0143  3836          CLRB    B_DONE_CT               ; clear done count
                     00000000'EF    B4  0149  3837          CLRW    W_FLAGS                 ; clear flags word
          0000010B'EF    0000010F'EF    D0  014F  3838          MOVL    L_BUF_SIZES_1,L_BUFFER_SIZE     ; load current buffer size
          000000C7'EF    000000EB'EF    D0  015A  3839          MOVL    A_BUFFERS_1,L_BUFFER_ADR        ; load buffer start address
0000008E'EF    0000008E'EF    01    9C  0165  3840          ROTL    #1,L_PTRN,L_PTRN        ; rotate data pattern
     00000092'EF48    0000008E'EF    D0  0171  3841          MOVL    L_PTRN,L_PTRNS[R8]      ; save data pattern
                     000019B5'EF    16  017D  3842          JSB     LOAD_BUFFER             ; load data into buffer
               00000000'EF    0F    09  EF  0183  3843          EXTZV   #9,#15,A_UBE_BUFFER_0,L_PAGE_NO ; extract page frame number
                     0000004A'EF        018B
                                0190  3844 ;
                                0190  3845 ; Set up all UBEs and maps as needed
                                0190  3846 ;
                59   00000337'EF    DE  0190  3847 10$:     MOVAL   UBE_0_MULT_1,R9         ; load command table address
     000000C7'EF    000000EB'EF48    D0  0197  3848 13$:     MOVL    A_BUFFERS_1[R8],L_BUFFER_ADR    ; load buffer start address
     000000A3'EF    000000BF'EF48    9A  01A3  3849          MOVZBL  B_MAPS_REQ_1[R8],L_MAPS_NEEDED  ; load number of maps needed
```

```
                  0000194D'EF   16  01AF  3850          JSB     SET_UP_MAPS                     ; set up maps
                        03 50   E8  01B5  3851          BLBS    R0,T5$                          ; br = maps set up
                         01E8   31  01B8  3852          BRW     100$                            ; br = no more usable maps
                     02   58    D1  01BB  3853 15$:     CMPL    R8,#2                           ; which device?
                          24    19  01BE  3854          BLSS    17$                             ; br = 0 or 1
0000008E'EF   0000008E'EF   01   9C  01C0  3855          ROTL    #1,L_PTRN,L_PTRN                ; rotate data pattern
   00000092'EF48   0000008E'EF   D0  01CC  3856          MOVL    L_PTRN,L_PTRNS[R8]              ; save data pattern
           51   00000092'EF48   D0  01D8  3857          MOVL    L_PTRNS[R8],R1                  ; load data UBE 2 or 3
              02 A9    51      B0  01E0  3858          MOVW    R1,2(R9)                        ; load data UBE 2 or 3
      09   09   0000014F'EF48   F0  01E4  3859 17$:     INSV    L_INITL_MAP[R8],#9,#9,6(R9)     ; load initial map number
                        06 A9       01ED
                  00001850'EF   16  01EF  3860          JSB     UBE_SET_UP                      ; load UBE registers
         9A 58    00000000'EF   F3  01F5  3861 20$:     AOBLEQ  L_LAST_UBE,R8,13$               ; incr dev num, br = not done
                                    01FD  3862 ;
                                    01FD  3863 ; Set timer, start transfers simultaneously, wait until all transfers complete
                                    01FD  3864 ; or Interval Timer expires
                                    01FD  3865 ;
    00000000'8F   80000080 8F   DA  01FD  3866          MTPR    #<M_CLK_ERR+M_CLK_INTR>,#PR$_ICCS ; stop clock
    00000000'8F   000000BB'EF   DA  0208  3867          MTPR    L_MXFR_TIME,#PR$_NICR           ; load time
            00000000'8F   10    DA  0213  3868          MTPR    #M_TRANSFER,#PR$_ICCS           ; transfer time to ICR
            00000000'8F   01    DA  021A  3869          MTPR    #1,#PR$_NICR                    ; load long next interval
            00000000'EF   04    AA  0221  3870          BICW2   #M_TIMEOUT,W_FLAGS              ; clear timeout flag
    00000000'8F   80000041 8F   DA  0228  3871          MTPR    #<M_CLK_ERR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
                                    0233  3872
                  00FFF00C 9F   B6  0233  3873          INCW    @#A_SIMUL_GO                    ; start all UBEs
                                    0239  3874
         18 00000000'EF   02   E0  0239  3875 30$:     BBS     #V_TIMEOUT,W_FLAGS,40$          ; br = timer expired
   00000000'EF   000000A2'EF   91  0241  3876          CMPB    B_DONE_CT,L_LAST_UBE            ; all transfers stopped ?
                         EB    15  024C  3877          BLEQ    30$                             ; br = transfers not stopped
   00000000'8F   80000080 8F   DA  024E  3878          MTPR    #<M_CLK_ERR+M_CLK_INTR>,#PR$_ICCS ; stop clock
                                    0259  3879 ;
                                    0259  3880 ; Check buffer 0 for data errors, report if any.  Note that 1/2 of buffer will
                                    0259  3881 ; have rotated data and the second 1/2 will be the original data due to the use
                                    0259  3882 ; of DATOB after DATIP.
                                    0259  3883 ;
                      58    D4  0259  3884 40$:     CLRL    R8                              ; initialize index
          00000092'EF48    01   9C  025B  3885          ROTL    #1,L_PTRNS[R8],L_EXP_DATA       ; first rotation
                  0000005E'EF       0263
 52   0000005E'EF    01    10   EF  0268  3886          EXTZV   #16,#1,L_EXP_DATA,R2            ; extract end-around carry
0000005E'EF    01    00    52   F0  0271  3887          INSV    R2,#0,#1,L_EXP_DATA             ; insert end-around carry
 52   0000005E'EF    01    07   EF  027A  3888          EXTZV   #7,#1,L_EXP_DATA,R2             ; extract low-byte carry
0000005E'EF    01    08    52   F0  0283  3889          INSV    R2,#8,#1,L_EXP_DATA             ; insert low-byte carry
        0000010F'EF    FF 8F   9C  028C  3890          ROTL    #-1,L_BUF_SIZES_1,L_BUFFER_SIZE ; load 1/2 buffer 0 size
                  0000010B'EF       0294
000000C7'EF   000000EB'EF48   D0  0299  3891          MOVL    A_BUFFERS_1[R8],L_BUFFER_ADR    ; load buffer address
                  000019CA'EF   16  02A5  3892          JSB     CHECK_BUFFER                    ; check buffer contents
                        29 50   E8  02AB  3893          BLBS    R0,42$                          ; br = no errors found
                                    02AE  3894
                                    02AE  3895          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-
                                    02AE  3896                          MSGADR=T_MULTI_ERR,-
                                    02AE  3897                          PRLINK=MXFR_ERR,-
                                    02AE  3898                          P1=T_MXFR_1_DESC[R8],-
                                    02AE  3899                          P2=A_BUFFERS_1[R8]
              000000EB'EF48   DD  02AE                    PUSHL   A_BUFFERS_1[R8]
              0000012F'EF48   DD  02B5                    PUSHL   T_MXFR_1_DESC[R8]
                0000154A'EF   DF  02BC                    PUSHAL  MXFR_ERR
                0000083A'EF   DF  02C2                    PUSHAL  T_MULTI_ERR
```

```
                00000000'EF   DD  02C8                    PUSHL   BE0_UNIT_NO
                          02   DD  02CE                    PUSHL   #$ER
                               02D0        ;;; TEST 9, SUBTEST 1, ERROR 2
            000100D0 9F   06   FB  02D0                    CALLS   #$$M, @#DS$ERRHARD
                               02D7  3900
  000000C7'EF   0000010F'EF   C0  02D7  3901 42$:    ADDL2   L_BUF_SIZES_1,L_BUFFER_ADR      ; load adr of second half
0000005E'EF   00000092'EF48   D0  02E2  3902         MOVL    L_PTRNS[R8],L_EXP_DATA          ; load expected data
              000019CA'EF     16  02EE  3903         JSB     CHECK_BUFFER                    ; check buffer contents
                    29 50     E8  02F4  3904         BLBS    R0,44$                          ; br = no errors found
                               02F7  3905
                               02F7  3906         $DS_ERRHARD_S   UNIT=BE1_UNIT_NO,-
                               02F7  3907                         MSGADR=T_MULTI_ERR,-
                               02F7  3908                         PRLINK=MXFR_ERR,-
                               02F7  3909                         P1=T_MXFR_1_DESC[R8],-
                               02F7  3910                         P2=A_BUFFERS_1[R8]
            000000EB'EF48   DD  02F7                    PUSHL   A_BUFFERS_1[R8]
            0000012F'EF48   DD  02FE                    PUSHL   T_MXFR_1_DESC[R8]
              0000154A'EF   DF  0305                    PUSHAL  MXFR_ERR
              0000083A'EF   DF  030B                    PUSHAL  T_MULT1_ERR
              00000000'EF   DD  0311                    PUSHL   BE1_UNIT_NO
                          03  DD  0317                    PUSHL   #$ER
                               0319        ;;; TEST 9, SUBTEST 1, ERROR 3
            000100D0 9F   06   FB  0319                    CALLS   #$$M, @#DS$ERRHARD
                               0320  3911
                               0320  3912 ;
                               0320  3913 ; Check buffers 2, 3 for data errors, report if any
                               0320  3914 ;
                    58   02    D0  0320  3915 44$:    MOVL    #2,R8                           ; load dev num
          00000000'EF    58    D1  0323  3916         CMPL    R8,L_LAST_UBE                   ; beyond last ube?
                       5F   14   032A  3917         BGTR    80$                             ; br = beyond last UBE
                               032C  3918
0000010B'EF   0000010F'EF48   D0  032C  3919 45$:    MOVL    L_BUF_SIZES_1[R8],L_BUFFER_SIZE ; load buffer size
0000005E'EF   00000092'EF48   D0  0338  3920         MOVL    L_PTRNS[R8],L_EXP_DATA          ; load expected data
000000C7'EF   000000EB'EF48   D0  0344  3921 47$:    MOVL    A_BUFFERS_1[R8],L_BUFFER_ADR    ; load buffer address
              000019CA'EF     16  0350  3922         JSB     CHECK_BUFFER                    ; check buffer contents
                    2A 50     E8  0356  3923         BLBS    R0,50$                          ; br = no errors found
                               0359  3924
                               0359  3925         $DS_ERRHARD_S   UNIT=BE0_UNIT_NO[R8],-
                               0359  3926                         MSGADR=T_MULTI_ERR,-
                               0359  3927                         PRLINK=MXFR_ERR,-
                               0359  3928                         P1=T_MXFR_1_DESC[R8],-
                               0359  3929                         P2=A_BUFFERS_1[R8]
            000000EB'EF48   DD  0359                    PUSHL   A_BUFFERS_1[R8]
            0000012F'EF48   DD  0360                    PUSHL   T_MXFR_1_DESC[R8]
              0000154A'EF   DF  0367                    PUSHAL  MXFR_ERR
              0000083A'EF   DF  036D                    PUSHAL  T_MULT1_ERR
            00000000'EF48   DD  0373                    PUSHL   BE0_UNIT_NO[R8]
                          04  DD  037A                    PUSHL   #$ER
                               037C        ;;; TEST 9, SUBTEST 1, ERROR 4
            000100D0 9F   06   FB  037C                    CALLS   #$$M, @#DS$ERRHARD
                               0383  3930
        A1 58   00000000'EF    F3  0383  3931 50$:    AOBLEQ  L_LAST_UBE,R8,45$               ; incr dev num, br = not done
                               038B  3932 ;
                               038B  3933 ; End test if usable maps exhausted
                               038B  3934 ;
                               038B  3935 80$:    $DS_BQUICK      100$
                          08   E0  038B                    BBS     #DSA$V_QUICK,-  ; BR IF QUICK
```

```
                10 0000FE00 9F        038D                                  @#DSA$GL_FLAGS, 100$
    000001F0 8F   0000008A'EF   D1    0393  3936          CMPL    L_MAP_NUM,#496                  ; usable maps exhausted? [01]
                          03    18    039E  3937          BGEQ    100$                            ; br = usable maps exhausted
                        FD97    31    03A0  3938          BRW     9$                              ; br to beginning of loop
                                      03A3  3939 100$:
                                      03A3  3940          $DS_ENDSUB
                                      03A3       T9_S1_X:
    00010038 9F   00000108'EF   FA    03A3                      CALLG   $S$, @#DS$ENDSUB
                                      03AE  3941
                                      03AE  3942          $DS_PAGE
```

```
                                03AE                    .SBTTL  SUBTEST 9.2:  UBE Multi-transfer 2
                                03AE  3945 ;+
                                03AE  3946 ; Subtest description:
                                03AE  3947 ;
                                03AE  3948 ;       This subtest will have all selected UBE's doing various transfer and/or
                                03AE  3949 ;       interrupts at different request levels to further check CPU handling
                                03AE  3950 ;       ablity.  The functions cycle through the map entries using byte offset
                                03AE  3951 ;       mode with each data pattern being rotated after each block transfer.
                                03AE  3952 ;       The following functions are performed:
                                03AE  3953 ;
                                03AE  3954 ;       UBE 0 - DATOB double word NPR transfers, data from cycle count
                                03AE  3955 ;                     register, interrupt on done BR6, 2000 cycles.
                                03AE  3956 ;
                                03AE  3957 ;       UBE 1 - DATIP/DATO single word NPR transfers, with rotate after
                                03AE  3958 ;                     DATIP, interrupt on done BR7, 1000 cycles.
                                03AE  3959 ;
                                03AE  3960 ;       UBE 2 - DATI double word BR5 transfers, 1000 cycles, interrupt
                                03AE  3961 ;                     on done BR5.
                                03AE  3962 ;
                                03AE  3963 ;       UBE 3 - DATI single word BR7 transfers with an interrupt after
                                03AE  3964 ;                     each transfer, 3000 cycles.
                                03AE  3965 ;
                                03AE  3966 ; Subtest steps:
                                03AE  3967 ;
                                03AE  3968 ;       BGNSUBTEST Multitransfer 2
                                03AE  3969 ;       :  DO FOR UBE Number = {0,1,2,3}
                                03AE  3970 ;       :  :  Set up UBE transfer
                                03AE  3971 ;       :  ENDDO
                                03AE  3972 ;       :  START Timer
                                03AE  3973 ;       :  SET Simultaneous GO bit
                                03AE  3974 ;       :  WAIT UNTIL All transfers stopped
                                03AE  3975 ;       :  DO FOR UBE Number = {0,1,2,3}
                                03AE  3976 ;       :  :  IF Data or Execution error
                                03AE  3977 ;       :  :  :  THEN REPORT error
                                03AE  3978 ;       :  :  ENDIF
                                03AE  3979 ;       :  ENDDO
                                03AE  3980 ;       ENDSUBTEST Multitransfer 2
                                03AE  3981 ;
                                03AE  3982 ; Errors:
                                03AE  3983 ;
                                03AE  3984 ;-
                                03AE  3985          $DS_BGNSUB
                                03AE              T9_S2::
00010030 9F   00000114'EF   FA  03AE                    CALLG   $S$, @#DS$BGNSUB
                                03B9  3986
                                03B9  3987          $DS_SETVEC_S    VECTOR=#SCB$L_BE3,SRVADR=BE3_SERVICE
                     00     DD  03B9                    PUSHL   #0
              00001A7C'EF   DF  03BB                    PUSHAL  BE3_SERVICE
              00000354 8F   DD  03C1                    PUSHL   #SCB$L_BE3
          00010160 9F   03  FB  03C7                    CALLS   #3, @#DS$SETVEC
                                03CE  3988
                     37   01  DA  03CE  3989          MTPR    #1,#PR$L_UB_INIT                ; initialize Unibus devices
              0000008A'EF   D4  03D1  3990          CLRL    L_MAP_NUM                       ; clear map number
0000008E'EF   33550FFF 8F   D0  03D7  3991          MOVL    #L_PATTERN,L_PTRN               ; load initial pattern
              000000B7'EF   D4  03E2  3992 1$:      CLRL    L_XFR_COUNT                     ; clear transfer count
                        58  D4  03E8  3993          CLRL    R8                              ; clear device number
              000000A2'EF   94  03EA  3994          CLRB    B_DONE_CT                       ; clear done count
```

```
                00000000'EF   B4  03F0  3995          CLRW    W_FLAGS                          ; clear flags word
             59 00000367'EF   DE  03F6  3996          MOVAL   UBE_0_MULT_2,R9                  ; load command table address
                              03FD  3997 ;
                              03FD  3998 ; Set up all UBEs and maps as needed, load buffers for devs 1,2,3
                              03FD  3999 ;
                              03FD  4000 10$:     $DS_BREAK                                    ; look for control-c
             00010058 9F   6E   FA  03FD                      CALLG   (SP), @#DS$BREAK
   000000C7'EF 000000FB'EF48   D0  0404  4001          MOVL    A_BUFFERS_2[R8],L_BUFFER_ADR     ; load buffer start address
   000000A3'EF 000000C3'EF48   9A  0410  4002          MOVZBL  B_MAPS_REQ_2[R8],L_MAPS_NEEDED   ; load number of maps needed
               0000194D'EF     16  041C  4003          JSB     SET_UP_MAPS                      ; set up maps
      09   09 0000014F'EF48   F0  0422  4004          INSV    L_INIT[_MAP[R8],#9,#9,6(R9)      ; load initial map number
                        06 A9      042B
                        03 50   E8  042D  4005          BLBS    R0,15$                           ; br = maps set up
                         01ED   31  0430  4006          BRW     100$                             ; br = no more usable maps
0000008E'EF 0000008E'EF    01   9C  0433  4007 15$:     ROTL    #1,L_PTRN,L_PTRN                 ; rotate data pattern
   00000092'EF48 0000008E'EF    D0  043F  4008          MOVL    L_PTRN,L_PTRNS[R8]               ; save data pattern
                           58   D5  044B  4009          TSTL    R8                               ; which device?
                           0F   14  044D  4010          BGTR    17$                              ; br = not dev 0
             51 00000092'EF48   D0  044F  4011          MOVL    L_PTRNS[R8],R1                   ; load data in BE0 Data Reg
                02 A9    51   B0  0457  4012          MOVW    R1,2(R9)                         ; load data in BE0 Data Reg
                         0012   31  045B  4013          BRW     19$                              ; br = dev 0
   0000010B'EF 0000011F'EF48   D0  045E  4014 17$:     MOVL    L_BUF_SIZES_2[R8],L_BUFFER_SIZE  ; load current buffer size
               000019B5'EF     16  046A  4015          JSB     LOAD_BUFFER                      ; load data into buffer
               00001850'EF     16  0470  4016 19$:     JSB     UBE_SET_UP                       ; load actual UBE registers
                           58   D6  0476  4017 20$:     INCL    R8                               ; incr dev num
             00000000'EF   58   D1  0478  4018          CMPL    R8,L_LAST_UBE                    ; last UBE set up
                           03   14  047F  4019          BGTR    25$                              ; br = last UBE set up
                         FF79   31  0481  4020          BRW     10$
                                0484  4021 ;
                                0484  4022 ; Set up timer, start transfers simultaneously, wait until all transfers
                                0484  4023 ; complete or Interval Timer expires
                                0484  4024 ;
00000000'8F 80000080 8F    DA  0484  4025 25$:     MTPR    #<M_CLK_ERR+M_CLK_INTR>,#PR$_ICCS ; stop clock
00000000'8F 000000BB'EF    DA  048F  4026          MTPR    L_MXFR_TIME,#PR$_NICR            ; load time
          00000000'8F   10   DA  049A  4027          MTPR    #M_TRANSFER,#PR$_ICCS            ; transfer time to ICR
          00000000'8F   01   DA  04A1  4028          MTPR    #1,#PR$_NICR                     ; load long next interval
          00000000'EF   04   AA  04A8  4029          BICW2   #M_TIMEOUT,W_FLAGS               ; clear timeout flag
00000000'8F 80000041 8F    DA  04AF  4030          MTPR    #<M_CLK_ERR+M_ENABLE+M_RUN>,#PR$_ICCS ; start timer
                               04BA  4031
             00FFF00C 9F   B6  04BA  4032          INCW    @#A_SIMUL_GO                     ; start all UBEs
                               04C0  4033
      18 00000000'EF   02   E0  04C0  4034 30$:     BBS     #V_TIMEOUT,W_FLAGS,40$           ; br = timer expired
00000000'EF 000000A2'EF    91  04C8  4035          CMPB    B_DONE_CT,L_LAST_UBE             ; all transfers stopped ?
                        EB   15  04D3  4036          BLEQ    30$                              ; br = transfers not stopped
00000000'8F 80000080 8F    DA  04D5  4037          MTPR    #<M_CLK_ERR+M_CLK_INTR>,#PR$_ICCS ; stop clock
                               04E0  4038 ;
                               04E0  4039 ; Check buffer 0 (data from CC) for data errors, report if any
                               04E0  4040 ;
000000C7'EF 000000FB'EF    D0  04E0  4041 40$:     MOVL    A_BUFFERS_2,L_BUFFER_ADR         ; load buffer address
0000010B'EF 0000011F'EF    D0  04EB  4042          MOVL    L_BUF_SIZES_2,L_BUFFER_SIZE      ; load buffer size
                        58   D4  04F6  4043          CLRL    R8                               ; initialize index
             00001A00'EF   16  04F8  4044          JSB     CHECK_BUF_0                      ; check buffer 0
                     29 50   E8  04FE  4045          BLBS    R0,42$                           ; br = no errors found
                               0501  4046
                               0501  4047          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO,-
                               0501  4048                          MSGADR=T_MULTI_ERR,-
                               0501  4049                          PRLINK=MXFR_ERR,-
```

```
                                0501  4050                          P1=T_MXFR_2_DESC[R8],-
                                0501  4051                          P2=A_BUFFERS_2[R8]
          000000FB'EF48   DD    0501                        PUSHL   A_BUFFERS_2[R8]
          0000013F'EF48   DD    0508                        PUSHL   T_MXFR_2_DESC[R8]
            0000154A'EF   DF    050F                        PUSHAL  MXFR_ERR
            0000083A'EF   DF    0515                        PUSHAL  T_MULT1_ERR
            00000000'EF   DD    051B                        PUSHL   BE0_UNIT_NO
                     01   DD    0521                        PUSHL   #$ER
                                0523        ;;; TEST 9, SUBTEST 2, ERROR 1
      000100D0 9F    06   FB    0523                        CALLS   #$$M, @#DS$ERRHARD
                                052A  4052
                                052A  4053 ;
                                052A  4054 ; Check buffer 1 for data errors, report if any
                                052A  4055 ;
                     58   D6    052A  4056 42$:     INCL    R8                              ; incr dev num
          00000000'EF    58   D1    052C  4057          CMPL    R8,L_LAST_UBE                   ; beyond last UBE?
                     03   15    0533  4058          BLEQ    43$                             ; br = not beyond last UBE
                   00D0   31    0535  4059          BRW     80$                             ; br = beyond last UBE
        00000092'EF48    01   9C    0538  4060 43$:     ROTL    #1,L_PTRNS[R8],L_EXP_DATA       ; load expected data
                0000005E'EF         0540
  52    0000005E'EF    01    10    EF    0545  4061          EXTZV   #16,#1,L_EXP_DATA,R2            ; extract end-around carry
  0000005E'EF    01    00    52    F0    054E  4062          INSV    R2,#0,#1,L_EXP_DATA             ; insert end-around carry
0000010B'EF    0000011F'EF48    D0    0557  4063          MOVL    L_BUF_SIZES_2[R8],L_BUFFER_SIZE ; load buffer size
000000C7'EF    000000FB'EF48    D0    0563  4064          MOVL    A_BUFFERS_2[R8],L_BUFFER_ADR    ; load buffer address
                000019CA'EF    16    056F  4065          JSB     CHECK_BUFFER                    ; check buffer contents
                       2A 50   E8    0575  4066          BLBS    R0,45$                          ; br = no errors found
                                0578  4067
                                0578  4068          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO[R8],-
                                0578  4069                          MSGADR=T_MULTT_ERR,-
                                0578  4070                          PRLINK=MXFR_ERR,-
                                0578  4071                          P1=T_MXFR_2_DESC[R8],-
                                0578  4072                          P2=A_BUFFERS_2[R8]
          000000FB'EF48    DD   0578                        PUSHL   A_BUFFERS_2[R8]
          0000013F'EF48    DD   057F                        PUSHL   T_MXFR_2_DESC[R8]
            0000154A'EF    DF   0586                        PUSHAL  MXFR_ERR
            0000083A'EF    DF   058C                        PUSHAL  T_MULT1_ERR
          00000000'EF48    DD   0592                        PUSHL   BE0_UNIT_NO[R8]
                     02    DD   0599                        PUSHL   #$ER
                                059B        ;;; TEST 9, SUBTEST 2, ERROR 2
      000100D0 9F    06   FB    059B                        CALLS   #$$M, @#DS$ERRHARD
                                05A2  4073
                                05A2  4074 ;
                                05A2  4075 ; Check UBE 2,3 data buffers
                                05A2  4076 ;
                     58   D6    05A2  4077 45$:     INCL    R8                              ; incr dev num
          00000000'EF    58   D1    05A4  4078          CMPL    R8,L_LAST_UBE                   ; beyond last UBE?
                     5B   14    05AB  4079          BGTR    80$                             ; br = beyond last UBE
             57    58    03   78    05AD  4080          ASHL    #3,R8,R7                        ; calc index from dev num
0000005E'EF    00000092'EF48    D0    05B1  4081          MOVL    L_PTRNS[R8],L_EXP_DATA          ; log expected data
0000005E'EF    000002AF'FF47    B1    05BD  4082          CMPW    @A_BE0_DB[R7],L_EXP_DATA        ; check UBE data bufer
                     D7   13    05C9  4083          BEQL    45$                             ; br = no error
00000163'EF    000002AF'FF47    3E    05CB  4084          MOVAW   @A_BE0_DB[R7],L_ERROR_LOG       ; log address of error
          0000015F'EF    01    D0    05D7  4085          MOVL    #1,L_ERROR_CT                   ; load error count
                                05DE  4086
                                05DE  4087          $DS_ERRHARD_S   UNIT=BE0_UNIT_NO[R8],-  ; report error
                                05DE  4088                          MSGADR=T_MULTT_ERR,-
                                05DE  4089                          PRLINK=MXFR_ERR,-
```

```
                                05DE  4090                        P1=T_MXFR_2_DESC[R8],-
                                05DE  4091                        P2=A_BUFFERS_2[R8]
          000000FB'EF48   DD    05DE                      PUSHL   A_BUFFERS_2[R8]
          0000013F'EF48   DD    05E5                      PUSHL   T_MXFR_2_DESC[R8]
            0000154A'EF   DF    05EC                      PUSHAL  MXFR_ERR
            0000083A'EF   DF    05F2                      PUSHAL  T_MULT1_ERR
          00000000'EF48   DD    05F8                      PUSHL   BE0_UNIT_NO[R8]
                    03    DD    05FF                      PUSHL   #$ER
                          0601        ;;; TEST 9, SUBTEST 2, ERROR 3
        000100D0 9F   06    FB  0601                      CALLS   #$$M, @#DS$ERRHARD
                                0608  4092
                                0608  4093 ;
                                0608  4094 ; End test if usable maps exhausted
                                0608  4095 ;
                                0608  4096 80$:    $DS_BQUICK      100$
                    08    E0    0608                      BBS     #DSA$V_QUICK,-  ; BR IF QUICK
                                060A                      @#DSA$GL_FLAGS, 100$
      10 0000FE00 9F            0610  4097        CMPL    L_MAP_NUM,#496          ; usable maps exhausted? [01]
000001F0 8F   0000008A'EF   D1  0618  4098        BGEQ    100$                    ; br = usable maps exhausted
                    03    18    061B  4099        BRW     1$                      ; br to beginning of loop
                  FDC2    31    061D  4100 100$:
                                0620  4101
                                0620  4102        $DS_ENDSUB
                                0620        T9_S2_X:
00010038 9F   00000114'EF   FA  0620                      CALLG   $S$, @#DS$ENDSUB
                                062B  4103
                                062B  4104        $DS_CLRVEC_S    VECTOR=#SCB$L_BE0
          00000348 8F   DD    062B                      PUSHL   #SCB$L_BE0
        00010168 9F   01    FB  0631                      CALLS   #1, @#DS$CLRVEC
                                0638  4105        $DS_CLRVEC_S    VECTOR=#SCB$L_BE1
          0000034C 8F   DD    0638                      PUSHL   #SCB$L_BE1
        00010168 9F   01    FB  063E                      CALLS   #1, @#DS$CLRVEC
                                0645  4106        $DS_CLRVEC_S    VECTOR=#SCB$L_BE2
          00000350 8F   DD    0645                      PUSHL   #SCB$L_BE2
        00010168 9F   01    FB  064B                      CALLS   #1, @#DS$CLRVEC
                                0652  4107        $DS_CLRVEC_S    VECTOR=#SCB$L_BE3
          00000354 8F   DD    0652                      PUSHL   #SCB$L_BE3
        00010168 9F   01    FB  0658                      CALLS   #1, @#DS$CLRVEC
                                065F  4108        $DS_CLRVEC_S    VECTOR=#SCB$L_TIMER
          000000C0 8F   DD    065F                      PUSHL   #SCB$L_TIMER
        00010168 9F   01    FB  0665                      CALLS   #1, @#DS$CLRVEC
                                066C  4109
                                066C  4110        $DS_ENDTEST
                    50    01    D0  066C                      MOVL    #1, R0          ; NORMAL EXIT
                                066F        TEST_009_X::
        00010058 9F   6E    FA  066F                      CALLG   (SP), @#DS$BREAK
                                04  0676                      RET                     ; RETURN TO TEST SEQUENCER
                                0677  4111
                                0677  4112        $DS_ENDMOD
                          00000000                        .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                      00000009  0000                      .LONG   $TN
                                0004  4113        .END
```

```
$$$                     = 00000114 R      06
$$ARGS                  = 0000000A
$$E                     = 00000001
$$M                     = 00000006
$$N                     = 00000001
$$S                     = FFFFFFFF
$$T1                    = 0000002C
$ENV                    = 00000001  G
$ER                     = FFFFFFFF
$MO                     = 00000001  G
$S$                     = 00000114 R      06
$ST                     = 00000000
$TN                     = 00000009
ALL                     = 00000008  G
ALL_MAPS                  0000192A R      02
A_BE0_BA                  000002B7 R      02
A_BE0_CC                  000002B3 R      02
A_BE0_CLR_ERR             000002AB R      02
A_BE0_CR1                 000002BF R      02
A_BE0_CR2                 000002BB R      02
A_BE0_DB                  000002AF R      02
A_BE3_CC                  000002C3 R      02
A_BE3_CR1                 000002C7 R      02
A_BUFFERS_1               000000EB R      02
A_BUFFERS_2               000000FB R      02
A_BUFRS_DFR_1             000000CB R      02
A_BUFRS_DFR_2             000000DB R      02
A_MAP_BGN               = 00F26800
A_MAP_END               = 00F26FFC
A_MEM_CSR_1             = 00F20004
A_SIMUL_GO              = 00FFF00C
A_UBE_ADR_TBL             000002AB R      02
A_UBE_BUFFER_0            ********   X    02
A_UBE_BUFFER_1            ********   X    02
A_UBE_BUFFER_2            ********   X    02
A_UBE_BUFFER_3            ********   X    02
A_UBI_CSR               = 00F26010
A_UB_IO_SP              = 00FC0000
BASE_008                  00000000 R      04
BASE_009                  00000000 R      07
BEO_SERVICE               00001A74 R      02
BEO_UNIT_NO               ********   X    04
BE1_UNIT_NO               ********   X    07
BE3_SERVICE               00001A7C R      02
BIT...                  = 0000001A
BR_SERVICE                00001A3C R      02
B_DONE_CT                 000000A2 R      02
B_MAPS_REQ_1              000000BF R      02
B_MAPS_REQ_2              000000C3 R      02
CEP_FUNCTIONAL          = 00000000
CEP_REPAIR              = 00000001
CHANNEL_ERR               000016CC R      02
CHC$_CLEAR                ********   X    04
CHC$_DSINT                ********   X    04
CHC$_ENINT                ********   X    04
CHECK_BUFFER              000019CA R      02
CHECK_BUF_0               00001A00 R      02
CHI$V_IPL                 ********   X    02
CODEVECTORS               00000297 R      02
CPU$V_COMET             = 00000002
CPU$V_HS                = 00000000
CPU$V_STAR              = 00000001
CSR_CRK                   00001471 R      02
CSR_ERR                   000015DF R      02
DATA_008                  0000001C R      04
DATA_009                  0000001C R      07
DATA_ERR                  000013C0 R      02
DECODE_TBL                00000277 R      02
DEFAULT                 = 00000000  G
DS$ABORT                  00010020
DS$ASKADR                 00010090
DS$ASKDATA                00010080
DS$ASKLGCL                00010098
DS$ASKSTR                 000100A0
DS$ASKVLD                 00010088
DS$ATTACH                 000101A8
DS$BGNSUB                 00010030
DS$BRANCH                 000100A8
DS$BREAK                  00010058
DS$CANWAIT                00010070
DS$CHANNEL                00010180
DS$CKLOOP                 00010040
DS$CLRVEC                 00010168
DS$CNTRLC                 00010078
DS$CVTREG                 000100B0
DS$ENDPASS                00010010
DS$ENDSUB                 00010038
DS$ERRDEV                 000100C8
DS$ERRHARD                000100D0
DS$ERRPREP                00010108
DS$ERRSOFT                000100D8
DS$ERRSYS                 000100C0
DS$ESCAPE                 00010050
DS$FREEDBGSYM             000101B8
DS$GETBUF                 00010120
DS$GETMEM                 00010130
DS$GPHARD                 00010018
DS$HELP                   000101B0
DS$INITSCB                00010170
DS$INLOOP                 00010048
DS$K_BNERROR              ********   X    04
DS$K_ERROR              = 00000002
DS$K_NORMAL             = 00000001
DS$K_SEVERE             = 00000004
DS$K_SUBSYS             = 00000066
DS$K_WARNING            = 00000000
DS$LOAD                   00010198
DS$LOADPCS                000101C0
DS$MAPDBGBLOCK            00010118
DS$MMOFF                  00010158
DS$MMON                   00010150
DS$MOVPHY                 00010148
DS$MOVVRT                 00010140
DS$PARSE                  000100B8
```

```
DS$PRINTB                       000100E0
DS$PRINTF                       000100F0
DS$PRINTS                       000100F8
DS$PRINTSIG                     00010100
DS$PRINTX                       000100E8
DS$PROBE                        000101A0
DS$RELBUF                       00010128
DS$RELMEM                       00010138
DS$SETIPL                       00010178
DS$SETMAP                       00010188
DS$SETPRIEXV                    00010110
DS$SETVEC                       00010160
DS$SHOCHAN                      00010190
DS$SUMMARY                      00010028
DS$WAITMS                       00010060
DS$WAITUS                       00010068
DS$_ARITH                   =   006600D0
DS$_BADLINK                 =   006600F0
DS$_BADTYPE                 =   006600E8
DS$_CHME                    =   006600A8
DS$_CHMK                    =   006600E0
DS$_DEVNAME                 =   00660108
DS$_ERROR                   =   00660002
DS$_FHWE                    =   00660068
DS$_FRAGBUF                 =   00660080
DS$_ICBUSY                  =   006600C8
DS$_ICERR                   =   006600C0
DS$_IHWE                    =   00660060
DS$_ILLCHAR                 =   00660018
DS$_ILLPAGCNT               =   00660078
DS$_ILLUNIT                 =   00660100
DS$_INSFMEM                 =   00660050
DS$_IPL2HI                  =   006600B8
DS$_IVADDR                  =   00660040
DS$_IVVECT                  =   00660038
DS$_KRNLSTK                 =   00660090
DS$_LOGIC                   =   00660070
DS$_MCHK                    =   00660088
DS$_MMOFF                   =   00660058
DS$_NEEDUNIT                =   006600F8
DS$_NOPCS                   =   00660110
DS$_NORMAL                  =   00660001
DS$_NOSUPPORT               =   006600B1
DS$_NOTDON                  =   00660030
DS$_NOTIMP                  =   006600B0
DS$_NULLSTR                 =   00660010
DS$_OVERFLOW                =   00660008
DS$_POWER                   =   00660098
DS$_PROGERR                 =   00660020
DS$_SEVERE                  =   00660004
DS$_TRANSL                  =   006600A0
DS$_TRUNCATE                =   00660028
DS$_UNEXPINT                =   006600D8
DS$_VASFULL                 =   00660048
DS$_WARNING                 =   00660000
DSA$AL_APTMAIL                  0000FE00
DSA$AT_APTTXT                   0000FA00
DSA$GL_APTCOM                   0000FE04
DSA$GL_DEVLEN                   0000FE58
DSA$GL_ERRNO                    0000FE44
DSA$GL_EVENT                    0000FE48
DSA$GL_FLAGS                    0000FE00
DSA$GL_MSGTYP                   0000FE40
DSA$GL_PASSES                   0000FE08
DSA$GL_PASSNO                   0000FE54
DSA$GL_SECTNO                   0000FE10
DSA$GL_SID                      0000FE14
DSA$GL_SUBTNO                   0000FE4C
DSA$GL_TESTNO                   0000FE50
DSA$GL_UNITS                    0000FE0C
DSA$GQ_MSGPTR                   0000FE68
DSA$GT_DEVNAM                   0000FE5C
DSA$M_APT                   =   80000000
DSA$M_BELL                  =   00000008
DSA$M_BINARY                =   00000002
DSA$M_CRD_AUTOTEST          =   00000800
DSA$M_CRD_MENUTEST          =   00010000
DSA$M_CRD_TRACE             =   00020000
DSA$M_DEBUG                 =   04000000
DSA$M_HALT                  =   00000001
DSA$M_IE1                   =   00000010
DSA$M_IE2                   =   00000020
DSA$M_IE3                   =   00000040
DSA$M_IES                   =   00000080
DSA$M_LOAD_DEBUGGER         =   40000000
DSA$M_LOOP                  =   00000004
DSA$M_NORPT                 =   08000000
DSA$M_OPER                  =   00001000
DSA$M_PASS0                 =   20000000
DSA$M_PROMPT                =   00002000
DSA$M_QA                    =   00008000
DSA$M_QUICK                 =   00000100
DSA$M_SEARCH                =   00004000
DSA$M_TRACE                 =   00000400
DSA$M_USER                  =   10000000
DSA$M_VERIFY                =   00000200
DSA$V_APT                   =   0000001F
DSA$V_BELL                  =   00000003
DSA$V_BINARY                =   00000001
DSA$V_CRD_AUTOTEST          =   0000000B
DSA$V_CRD_MENUTEST          =   00000010
DSA$V_CRD_TRACE             =   00000011
DSA$V_DEBUG                 =   0000001A
DSA$V_HALT                  =   00000000
DSA$V_IE1                   =   00000004
DSA$V_IE2                   =   00000005
DSA$V_IE3                   =   00000006
DSA$V_IES                   =   00000007
DSA$V_LOAD_DEBUGGER         =   0000001E
DSA$V_LOOP                  =   00000002
DSA$V_NORPT                 =   0000001B
DSA$V_OPER                  =   0000000C
DSA$V_PASS0                 =   0000001D
DSA$V_PROMPT                =   0000000D
```

```
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
ERR$_MSGADR                     = 0000000C
ERR$_NARGS                      = 0000000A
ERR$_NUM                        = 00000004
ERR$_P1                         = 00000014
ERR$_P2                         = 00000018
ERR$_P3                         = 0000001C
ERR$_P4                         = 00000020
ERR$_P5                         = 00000024
ERR$_P6                         = 00000028
ERR$_POINTER                    = 00000010
ERR$_UNIT                       = 00000008
FRST_VAL_MAP                      00000002 R      02
HALT                            = 00000004   G
HP$A_DEPENDENT                    00000032
HP$A_DEVICE                       00000018
HP$A_DVA                          0000001C
HP$A_LINK                         00000020
HP$B_DRIVE                        0000000B
HP$B_FLAGS                        0000000A
HP$M_ALLOC                      = 00000001
HP$M_WASALL                     = 00000002
HP$Q_DEVICE                       00000000
HP$T_DEVICE                       0000000C
HP$T_TYPE                         00000026
HP$V_ALLOC                      = 00000000
HP$V_WASALL                     = 00000001
HP$W_SIZE                         00000008
HP$W_VECTOR                       00000024
INTR_ERR                          0000166E R      02
INTR_SERVICE                      00001A6C R      02
INVAL_MAPS                        000018FB R      02
IPR                             = 00000006   G
KA$B_ACC_TYPE                     00000042
KA$B_RESERVED                     00000043
KA$B_SID_TYPE                     0000003D
KA$K_LEN                          00000046
KA$L_FLAGS                        00000032
KA$L_SID                          0000003A
KA$M_CHAR                       = 00000100
KA$M_CRC                        = 00000010
KA$M_D_FLOAT                    = 00000002
KA$M_EDIT                       = 00000080
KA$M_F_FLOAT                    = 00000001
KA$M_G_FLOAT                    = 00000004
KA$M_H_FLOAT                    = 00000008
KA$M_PACKED                     = 00000020
KA$M_PACK_MUL                   = 00000040
KA$M_SB_ERR                     = 02000000
KA$M_TODR                       = 00010000
KA$M_WCS_LD                     = 01000000
KA$V_CHAR                       = 00000008
KA$V_CRC                        = 00000004
KA$V_D_FLOAT                    = 00000001
KA$V_EDIT                       = 00000007
KA$V_F_FLOAT                    = 00000000
KA$V_G_FLOAT                    = 00000002
KA$V_H_FLOAT                    = 00000003
KA$V_PACKED                     = 00000005
KA$V_PACK_MUL                   = 00000006
KA$V_SB_ERR                     = 00000019
KA$V_TODR                       = 00000010
KA$V_WCS_LD                     = 00000018
KA$W_LATENCY                      0000003E
KA$W_MEM_SIZE                     00000044
KA$W_PROGRESS                     00000040
KA$W_RESERVED                     00000038
KA$W_WCS                          00000036
LDPCTX                          = 0000000B   G
LOAD_BUFFER                       000019B5 R      02
L_BR_LVL                          000000AB R      02
L_BUFFER_ADR                      000000C7 R      02
L_BUFFER_SIZE                     0000010B R      02
L_BUF_SIZES_1                     0000010F R      02
L_BUF_SIZES_2                     0000011F R      02
L_ERROR_CT                        0000015F R      02
L_ERROR_LOG                       00000163 R      02
L_EXP_DATA                        0000005E R      02
L_INFO_BIT                        0000018B R      02
L_INIT_MAP                        0000014F R      02
L_IPL                             000000A7 R      02
L_LAST_UBE                        ********     X  07
L_MAPS_NEEDED                     000000A3 R      02
L_MAP_NUM                         0000008A R      02
L_MXFR_TIME                       000000BB R      02
L_NXM_ADR                         00000086 R      02
L_PAGE_NO                         0000004A R      02
L_PAIR_SEP                        0000018F R      02
L_PATTERN                       = 33550FFF
L_POP                             0000019F R      02
L_POSITION                        00000193 R      02
L_PTRN                            0000008E R      02
L_PTRNS                           00000092 R      02
L_RETRIEVE                        000001A3 R      02
L_SAVE_CSR                        0000004E R      02
```

```
L_TEMP_A                        00000197 R      02
L_TEMP_B                        0000019B R      02
L_UBI_STATUS                    000000AF R      02
L_XFR_COUNT                     000000B7 R      02
MANUAL                        = 00000001   G
MAP_ADR_ERR                     000013E9 R      02
MAP_ENT_TBL                     000001A7 R      02
MCHK                          = 00000005   G
MEM                           = 00000009   G
MSG_008                         00000002 R      04
MSG_009                         00000002 R      07
MXFR_ERR                        0000154A R      02
M_BR4                         = 00000002
M_BR5                         = 00000004
M_BR6                         = 00000008
M_BR7                         = 00000010
M_BR_INTR                     = 00000008
M_BYTE_OFF                    = 02000000
M_CCOVF                       = 00002000
M_CC_DAT                      = 00001000
M_CLK_ERR                     = 80000000
M_CLK_INTR                    = 00000080
M_DATA_ERR                    = 00001000
M_DATI                        = 00000000
M_DATIP                       = 00000100
M_DATO                        = 00000200
M_DATOB                       = 00000300
M_DATOB_IP                    = 00004000
M_DIAG_MODE                   = 04000000
M_DSABL_ECC                   = 02000000
M_ENABLE                      = 00000040
M_ENB_PB                      = 00001000
M_ERR                         = 00008000
M_EXP_BR_INTR                 = 00000002
M_EXP_NXM_ERR                 = 00000200
M_EXP_PB_ERR                  = 00000400
M_EXP_WR_ERR                  = 00000080
M_EXT_MEM                     = 00000000
M_FRCE_TB_ERR                 = 20000000
M_GO                          = 00000001
M_INH_INC                     = 00000004
M_INH_ROL                     = 00002000
M_INH_SACK                    = 00000008
M_INTR_ERR                    = 00000010
M_INT_BM                      = 00000000
M_INT_ODNE                    = 00000040
M_INT_SSYN                    = 00000800
M_MAP_NOWRT                   = 7DFF8000
M_MAP_VAL                     = 80000000
M_MAP_WRTBL                   = 82007FFF
M_MAX_LAT                     = 00000040
M_NO_CSR                      = 00000001
M_NO_GNT                      = 00000400
M_NO_SACK                     = 00000080
M_NO_SSYN                     = 00000100
M_NPR                         - 00000020
M_ONE_XFR                     - 00000400
M_PWR_DN                      = 00000010
M_RDY                         = 00000080
M_RLS_BUS                     = 00000C00
M_RUN                         = 00000001
M_TIMEOUT                     = 00000004
M_TM_DLY                      = 00004000
M_TRANSFER                    = 00000010
M_TWO_XFR                     = 00000800
M_UBI_NON_STS                 = 7FFE3FFF
M_UBI_NXM_ERR                 = 00010000
M_UBI_PB_ERR                  = 00008000
M_UBI_RDS_ERR                 = 80000000
M_UBI_STS                     = 8001C000
M_UBI_WR_ERR                  = 00004000
M_UNX_SSYN_ERR                = 00000040
M_WNG_A                       = 00000200
M_WNG_GNT                     = 00000020
NO_ACCESS                       00001718 R      02
PAR$M_ATDEF                   = 00000010
PAR$M_ATHI                    = 00000008
PAR$M_ATLO                    = 00000004
PAR$M_NODEF                   = 00000002
PAR$V_ATDEF                   = 00000004
PAR$V_ATHI                    = 00000003
PAR$V_ATLO                    = 00000002
PAR$V_NODEF                   = 00000001
PAR$_BIN                      = 00000002
PAR$_DEC                      = 0000000A
PAR$_HEX                      = 00000010
PAR$_NO                       = 00000000
PAR$_OCT                      = 00000008
PAR$_YES                      = 00000001
POWER                         = 00000003   G
PR$L_UB_INIT                  = 00000037
PR$_ICCS                        ********   X    02
PR$_MCESR                       ********   X    02
PR$_NICR                        ********   X    07
PSL$C_EXEC                    = 00000001
PSL$C_KERNEL                  = 00000000
PSL$C_SUPER                   = 00000002
PSL$C_USER                    = 00000003
PSL$K_EXEC                    = 00000001
PSL$K_KERNEL                  = 00000000
PSL$K_SUPER                   = 00000002
PSL$K_USER                    = 00000003
PSL$M_C                       = 00000001
PSL$M_CBIT                    = 00000001
PSL$M_CM                      = 80000000
PSL$M_CURMOD                  = 03000000
PSL$M_DV                      = 00000080
PSL$M_FPD                     = 08000000
PSL$M_FU                      = 00000040
PSL$M_IPL                     = 001F0000
PSL$M_IS                      = 04000000
PSL$M_IV                      = 00000020
PSL$M_N                       = 00000008
PSL$M_NBIT                    = 00000008
```

```
PSL$M_PRVMOD                    = 00C00000
PSL$M_SAFBITS                   = 000037FF
PSL$M_TBIT                      = 00000010
PSL$M_TP                        = 40000000
PSL$M_V                         = 00000002
PSL$M_VBIT                      = 00000002
PSL$M_Z                         = 00000004
PSL$M_ZBIT                      = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CBIT                      = 00000000
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_NBIT                      = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_VBIT                      = 00000001
PSL$V_Z                         = 00000002
PSL$V_ZBIT                      = 00000002
RD_MLLR_DECODE                    0000175E R      02
RD_MLLR_ENCODE                    0000172F R      02
RD_MLLR_TBL                       00000237 R      02
RETRIEVE_TBL                      00000257 R      02
SCB$L_ACCESS                      00000020
SCB$L_ARITH                       00000034
SCB$L_BE0                       = 00000348
SCB$L_BE1                       = 0000034C
SCB$L_BE2                       = 00000350
SCB$L_BE3                       = 00000354
SCB$L_BREAK                       0000002C
SCB$L_CHME                        00000044
SCB$L_CHMK                        00000040
SCB$L_CHMS                        00000048
SCB$L_CHMU                        0000004C
SCB$L_COMPAT                      00000030
SCB$L_KNLSTK                      00000008
SCB$L_MACHCHK                     00000004
SCB$L_OPCCUS                      00000014
SCB$L_OPCDEC                      00000010
SCB$L_POWER                       0000000C
SCB$L_RADRMOD                     0000001C
SCB$L_ROPRAND                     00000018
SCB$L_RXDB                        000000F8
SCB$L_SFTLVL1                     00000084
SCB$L_SFTLVL10                    000000A8
SCB$L_SFTLVL11                    000000AC
SCB$L_SFTLVL12                    000000B0
SCB$L_SFTLVL13                    000000B4
SCB$L_SFTLVL14                    000000B8
SCB$L_SFTLVL15                    000000BC
SCB$L_SFTLVL2                     00000088
SCB$L_SFTLVL3                     0000008C
SCB$L_SFTLVL4                     00000090
SCB$L_SFTLVL5                     00000094
SCB$L_SFTLVL6                     00000098
SCB$L_SFTLVL7                     0000009C
SCB$L_SFTLVL8                     000000A0
SCB$L_SFTLVL9                     000000A4
SCB$L_TBIT                        00000028
SCB$L_TIMER                       000000C0
SCB$L_TRANSL                      00000024
SCB$L_TXDB                        000000FC
SCB$L_ZERO                        00000000
SEP_FUNCTIONAL                  = 00000002
SEP_REPAIR                      = 00000003
SET_UP_MAPS                       0000194D R      02
SIZ...                          = 00000001
SYS$ALLOC                         00010238
SYS$ASCTIM                        00010248
SYS$ASSIGN                        00010250
SYS$BINTIM                        00010258
SYS$CANCEL                        00010260
SYS$CANTIM                        00010268
SYS$CLOSE                         000105B8
SYS$CLREF                         00010298
SYS$CONNECT                       000105C0
SYS$DALLOC                        000102D8
SYS$DASSGN                        000102E0
SYS$DISCONNECT                    000105D0
SYS$FAO                           00010350
SYS$FAOL                          00010358
SYS$GET                           00010580
SYS$GETCHN                        000104C8
SYS$GETTIM                        00010378
SYS$LKWSET                        000103A0
SYS$NUMTIM                        000103B8
SYS$OPEN                          00010608
SYS$QIO                           000103C8
SYS$QIOW                          00010200
SYS$READ                          00010590
SYS$READEF                        000103D0
SYS$SETAST                        000103F8
SYS$SETEF                         00010400
SYS$SETIMR                        00010420
SYS$SETPRI                        00010428
SYS$SETPRT                        00010430
SYS$SETRWM                        00010438
SYS$SETSWM                        00010448
SYS$ULKPAG                        00010460
SYS$ULWSET                        00010468
SYS$UNWIND                        00010470
SYS$WAITFR                        00010478
SYS$WFLAND                        00010488
```

```
SYS$WFLOR                       00010490
T8_S1                           0000003C RG    04
T8_S10                          00000A61 RG    04
T8_S10_X                        00000B76 R     04
T8_S11                          00000B81 RG    04
T8_S11_X                        00000C9A R     04
T8_S12                          00000CA5 RG    04
T8_S12_X                        00000E3B R     04
T8_S13                          00000E46 RG    04
T8_S13_X                        00000F69 R     04
T8_S14                          00000F74 RG    04
T8_S14_X                        00001085 R     04
T8_S15                          00001090 RG    04
T8_S15_X                        000011A7 R     04
T8_S16                          000011B2 RG    04
T8_S16_X                        0000132B R     04
T8_S17                          00001336 RG    04
T8_S17_X                        0000147F R     04
T8_S18                          0000148A RG    04
T8_S18_X                        000015D0 R     04
T8_S19                          000015DB RG    04
T8_S19_X                        000016D6 R     04
T8_S1_X                         0000012F R     04
T8_S2                           0000013A RG    04
T8_S20                          000016E1 RG    04
T8_S20_X                        000017DD R     04
T8_S21                          000017E8 RG    04
T8_S21_X                        000018D5 R     04
T8_S22                          000018E0 RG    04
T8_S22_X                        0000198A R     04
T8_S2_X                         000001A6 R     04
T8_S3                           000001B1 RG    04
T8_S3_X                         00000244 R     04
T8_S4                           0000024F RG    04
T8_S4_X                         00000332 R     04
T8_S5                           0000033D RG    04
T8_S5_X                         0000049E R     04
T8_S6                           000004A9 RG    04
T8_S6_X                         00000652 R     04
T8_S7                           0000065D RG    04
T8_S7_X                         000007D9 R     04
T8_S8                           000007E4 RG    04
T8_S8_X                         0000091F R     04
T8_S9                           0000092A RG    04
T8_S9_X                         00000A56 R     04
T9_S1                           000000C0 RG    07
T9_S1_X                         000003A3 R     07
T9_S2                           000003AE RG    07
T9_S2_X                         00000620 R     07
TEST_008                        00000020 RG    04
TEST_008_X                      00001998 RG    04
TEST_009                        00000020 RG    07
TEST_009_X                      0000066F RG    07
TIMER_SERVICE                   00001AA4 R     02
TU58                        =   00000002   G
T_ACTL_INTR                     00001204 R     02
T_ADR_REG_ERR                   00000511 R     02
T_AO_DATIP_ERR                  000006EC R     02
T_AO_DATI_ERR                   000006B3 R     02
T_AO_DATOB_ERR                  00000679 R     02
T_AO_DATO_ERR                   00000640 R     02
T_BUFF_ADR                      00000953 R     02
T_BYTE_ERR                      00000D72 R     02
T_CHANNEL_ERR                   000012FF R     02
T_CSR_ERR                       00000463 R     02
T_CSR_NCL_ERR                   0000114D R     02
T_DATA_REG_ERR                  00000540 R     02
T_DATIP_ERR                     000005D3 R     02
T_DATI_ERR                      0000056C R     02
T_DATOB_ERR                     000005FB R     02
T_DATO_ERR                      0000061E R     02
T_D_ERR_HDR                     000008B5 R     02
T_D_PATH_ERR                    0000058E R     02
T_EXP_INTR                      00001174 R     02
T_IHWE                          0000133C R     02
T_INTR_ERR                      000007DF R     02
T_INT_SSYN_ERR                  00000F9E R     02
T_IVVECT                        00001393 R     02
T_LOGIC                         00001360 R     02
T_LONG_ERR                      00000DC8 R     02
T_M1_UBE_0                      0000096B R     02
T_M1_UBE_1                      000009F4 R     02
T_M1_UBE_2                      00000A62 R     02
T_M1_UBE_3                      00000AA5 R     02
T_M2_UBE_0                      00000AE8 R     02
T_M2_UBE_1                      00000B4C R     02
T_M2_UBE_2                      00000BAD R     02
T_M2_UBE_3                      00000C1E R     02
T_MAPS_USED                     00000C9F R     02
T_MAP_ABUS_ERR                  000004AC R     02
T_MAP_ADR_ERR                   00001236 R     02
T_MAP_DBUS_ERR                  00000482 R     02
T_MAP_ENT_ERR                   000004D9 R     02
T_MAP_FCN_ERR                   0000075A R     02
T_MAP_INV_ERR                   0000080D R     02
T_MAP_SEL_ERR                   000005B0 R     02
T_MAP_USED                      00000CC7 R     02
T_MAX_LAT_ERR                   00000E6D R     02
T_MULTI_ERR                     0000083A R     02
T_MXFR_1_DESC                   0000012F R     02
T_MXFR_2_DESC                   0000013F R     02
T_MXFR_D_ERR                    00000866 R     02
T_MXFR_ERR_HDR                  00000CE7 R     02
T_MXFR_TYPE                     00000939 R     02
T_NO_ACCESS                     000003C3 R     02
T_NO_BEO                        00000397 R     02
T_NO_CSR_ERR                    00000888 R     02
T_NO_GNT_ERR                    00000F50 R     02
T_NO_INTR                       000011DF R     02
T_NO_MAPS                       00000437 R     02
T_NO_NXM_ERR                    00001052 R     02
T_NO_PB_ERR                     000010A7 R     02
T_NO_RDY_ERR                    00000E0D R     02
T_NO_SACK_ERR                   00000EA4 R     02
```

```
T_NO_SSYN_ERR                   00000ECA R      02
T_NO_UBE                        00000414 R      02
T_NO_WR_ERR                     00001117 R      02
T_NXM_ERR                       000007B6 R      02
T_PB_ERR                        00000786 R      02
T_PG_BND_WR_ERR                 0000072B R      02
T_UBE_ADR                       000003C5 R      02
T_UBI_NXM_ERR                   0000101E R      02
T_UBI_PB_ERR                    00001081 R      02
T_UBI_RDS_ERR                   00000FEA R      02
T_UBI_WR_ERR                    000010DF R      02
T_UNXP_INTR                     000011B4 R      02
T_UNX_SSYN_ERR                  00000DE4 R      02
T_WNG_A_ERR                     00000F02 R      02
T_WNG_GNT_ERR                   00000E40 R      02
T_WORD_ERR                      00000DA0 R      02
UBE_0_MULT_1                    00000337 R      02
UBE_0_MULT_2                    00000367 R      02
UBE_1_MULT_1                    00000343 R      02
UBE_1_MULT_2                    00000373 R      02
UBE_2_MULT_1                    0000034F R      02
UBE_2_MULT_2                    0000037F R      02
UBE_3_MULT_1                    0000035B R      02
UBE_3_MULT_2                    0000038B R      02
UBE_BR4_INTR                    0000032B R      02
UBE_BR5_INTR                    0000031F R      02
UBE_BR6_INTR                    00000313 R      02
UBE_BR7_INTR                    00000307 R      02
UBE_DATIP_1                     000002D7 R      02
UBE_DATI_T                      000002CB R      02
UBE_DATOB_1                     000002E3 R      02
UBE_DATO_T                      000002EF R      02
UBE_PB_ERR                      000002FB R      02
UBE_SET_UP                      00001850 R      02
UBI                           = 0000000A  G
UNIBUS_SIZER                    00001867 R      02
VAL_MAPS                        0000190D R      02
V_BR_INTR                     = 00000003
V_BYTE_OFF                    = 00000019
V_CCOVF                       = 0000000D
V_CLK_ERR                     = 0000001F
V_CLK_INTR                    = 00000007
V_DATA_ERR                    = 0000000C
V_ENABLE                      = 00000006
V_ERR                         = 0000000F
V_EXP_BR_INTR                 = 00000001
V_EXP_NXM_ERR                 = 00000009
V_EXP_PB_ERR                  = 0000000A
V_EXP_WR_ERR                  = 00000007
V_INTR_ERR                    = 00000004
V_INT_SSYN                    = 0000000B
V_MAP_VAL                     = 0000001F
V_MAX_LAT                     = 00000006
V_NO_CSR                      = 00000000
V_NO_GNT                      = 0000000A
V_NO_SACK                     = 00000007
V_NO_SSYN                     = 00000008
V_RDY                         = 00000007
V_RUN                         = 00000000
V_TIMEOUT                     = 00000002
V_TRANSFER                    = 00000004
V_UBI_NXM_ERR                 = 00000010
V_UBI_PB_ERR                  = 0000000F
V_UBI_RDS_ERR                 = 0000001F
V_UBI_WR_ERR                  = 0000000E
V_UNX_SSYN_ERR                = 00000006
V_VALID_MAPS                    00000006 R      02
V_WNG_A                       = 00000009
V_WNG_GNT                     = 00000005
WCS                           = 00000007  G
W_FLAGS                         00000000 R      02
W_PTRN_TBL                      0000029F R      02
W_UBI_STS_2                     000000B3 R      02
W_VECTOR                        000000B5 R      02
XFER_ERR                        00001420 R      02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXH	00001AB5 (6837.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_008	000019BF (6591.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000030 (48.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	00000120 (288.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
TEST_009	00000677 (1655.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
$TSTCNT	00000004 (4.)	08 (8.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$$	=00000114-R	3985 (34)	1479 (9) 1561 (10) 1679 (12) 1759 (13)
			1887 (14) 2008 (15) 2124 (16) 2240 (17)
			2347 (18) 2450 (19) 2554 (20) 2654 (21)
			2779 (22) 2881 (23) 2979 (24) 3078 (25)
			3205 (26) 3315 (27) 3414 (28) 3499 (29)
			3581 (30) 3654 (31) 3807 (33) 3985 (34)
$$ARGS	=0000000A	81 (2)	81 (2)
$$E	=00000001	3985 (34)	1530 (9) 1586 (10) 1719 (12) 1815 (13)
			1974 (14) 2083 (15) 2203 (16) 2309 (17)
			2419 (18) 2517 (19) 2621 (20) 2746 (21)
			2848 (22) 2946 (23) 3045 (24) 3163 (25)
			3272 (26) 3382 (27) 3465 (28) 3549 (29)
			3626 (30) 3693 (31) 3940 (33) 4102 (34)
$$M	=00000006	4091 (34)	1428 (8) 1494 (9) 1509 (9) 1526 (9)
			1579 (10) 1715 (12) 1776 (13) 1793 (13)
			1809 (13) 1903 (14) 1921 (14) 1943 (14)
			1966 (14) 2017 (15) 2027 (15) 2042 (15)
			2054 (15) 2071 (15) 2133 (16) 2166 (16)
			2194 (16) 2268 (17) 2296 (17) 2376 (18)
			2405 (18) 2475 (19) 2502 (19) 2580 (20)
			2607 (20) 2686 (21) 2714 (21) 2731 (21)
			2805 (22) 2834 (22) 2904 (23) 2931 (23)
			3003 (24) 3030 (24) 3103 (25) 3131 (25)
			3148 (25) 3233 (26) 3261 (26) 3343 (27)
			3371 (27) 3444 (28) 3463 (28) 3513 (29)
			3547 (29) 3605 (30) 3624 (30) 3674 (31)
			3690 (31) 3722 (32) 3827 (33) 3899 (33)
			3910 (33) 3929 (33) 4051 (34) 4072 (34)
			4091 (34)
$$N	=00000001	4091 (34)	#-1005 (6) #-1010 (6) 1016 (6) 1030 (6)
			#-1432 (8) 1494 (9) 1509 (9) 1526 (9)
			1579 (10) 1715 (12) 1776 (13) 1793 (13)
			1809 (13) 1903 (14) 1921 (14) 1943 (14)
			1966 (14) 2017 (15) 2027 (15) 2042 (15)
			2054 (15) 2071 (15) 2133 (16) #-2150 (16)
			2166 (16) 2194 (16) 2268 (17) 2296 (17)
			2376 (18) 2405 (18) 2475 (19) 2502 (19)
			2580 (20) 2607 (20) 2686 (21) 2714 (21)
			2731 (21) 2805 (22) 2834 (22) 2904 (23)
			2931 (23) 3003 (24) 3030 (24) 3103 (25)
			3131 (25) 3148 (25) 3233 (26) 3261 (26)
			3343 (27) 3371 (27) 3444 (28) 3463 (28)
			3513 (29) 3547 (29) 3605 (30) 3624 (30)
			3674 (31) 3690 (31) #-3729 (32) 3827 (33)
			3899 (33) 3910 (33) 3929 (33) 4051 (34)
			4072 (34) 4091 (34) #-774 (6) 782 (6)
			805 (6) #-821 (6) 827 (6) 833 (6)
			#-847 (6) 85 (2) #-850 (6) #-853 (6)
			#-856 (6) #-859 (6) #-862 (6) #-865 (6)
			#-868 (6) #-871 (6) #-874 (6) #-877 (6)

$TN =00000009 4112 (34)
1509 (9) 1511 (9) 1526 (9) 1528 (9)
1530 (9) 1534 (10) 1561 (10) 1579 (10)
1581 (10) 1586 (10) 1591 (12) 1679 (12)
1715 (12) 1719 (12) 1723 (13) 1759 (13)
1776 (13) 1778 (13) 1793 (13) 1795 (13)
1809 (13) 1811 (13) 1815 (13) 1819 (14)
1887 (14) 1903 (14) 1921 (14) 1923 (14)
1943 (14) 1945 (14) 1966 (14) 1968 (14)
1974 (14) 1978 (15) 2008 (15) 2017 (15)
2027 (15) 2042 (15) 2054 (15) 2071 (15)
2083 (15) 2087 (16) 2124 (16) 2133 (16)
2166 (16) 2194 (16) 2197 (16) 2203 (16)
2207 (17) 2240 (17) 2268 (17) 2296 (17)
2299 (17) 2309 (17) 2313 (18) 2347 (18)
2376 (18) 2405 (18) 2408 (18) 2419 (18)
2423 (19) 2450 (19) 2475 (19) 2502 (19)
2505 (19) 2517 (19) 2521 (20) 2554 (20)
2580 (20) 2607 (20) 2610 (20) 2621 (20)
2625 (21) 2654 (21) 2686 (21) 2714 (21)
2731 (21) 2734 (21) 2746 (21) 2750 (22)
2779 (22) 2805 (22) 2834 (22) 2837 (22)
2848 (22) 2852 (23) 2881 (23) 2904 (23)
2931 (23) 2934 (23) 2946 (23) 2950 (24)
2979 (24) 3003 (24) 3030 (24) 3033 (24)
3045 (24) 3049 (25) 3078 (25) 3103 (25)
3131 (25) 3148 (25) 3151 (25) 3163 (25)
3167 (26) 3205 (26) 3233 (26) 3261 (26)
3264 (26) 3272 (26) 3276 (27) 3315 (27)
3343 (27) 3371 (27) 3374 (27) 3382 (27)
3386 (28) 3414 (28) 3444 (28) 3463 (28)
3465 (28) 3469 (29) 3499 (29) 3513 (29)
3547 (29) 3549 (29) 3553 (30) 3581 (30)
3605 (30) 3624 (30) 3626 (30) 3630 (31)
3654 (31) 3674 (31) 3690 (31) 3693 (31)
3695 (31) 3722 (32) 3757 (33) 3807 (33)
3827 (33) 3899 (33) 3910 (33) 3929 (33)
3940 (33) 3944 (34) 3985 (34) 4051 (34)
4072 (34) 4091 (34) 4102 (34) 4110 (34)
1417 (8) 1428 (8) 1433 (8) 1439 (9)
1494 (9) 1509 (9) 1526 (9) 1534 (10)
1579 (10) 1591 (12) 1715 (12) 1723 (13)
1776 (13) 1793 (13) 1809 (13) 1819 (14)
1903 (14) 1905 (14) 1921 (14) 1943 (14)
1966 (14) 1978 (15) 2017 (15) 2027 (15)
2042 (15) 2054 (15) 2071 (15) 2087 (16)
2133 (16) 2151 (16) 2166 (16) 2194 (16)
2207 (17) 2268 (17) 2296 (17) 2313 (18)
2376 (18) 2405 (18) 2423 (19) 2475 (19)
2502 (19) 2521 (20) 2580 (20) 2607 (20)
2625 (21) 2686 (21) 2714 (21) 2731 (21)
2750 (22) 2805 (22) 2834 (22) 2852 (23)
2904 (23) 2931 (23) 2950 (24) 3003 (24)
3030 (24) 3049 (25) 3103 (25) 3131 (25)
3148 (25) 3167 (26) 3233 (26) 3261 (26)
3276 (27) 3343 (27) 3371 (27) 3386 (28)
3444 (28) 3463 (28) 3469 (29) 3513 (29)
3547 (29) 3553 (30) 3605 (30) 3624 (30)

```
                                                3630  (31)    3674  (31)    3690  (31)    3695  (31)
                                                3704  (32)    3722  (32)    3730  (32)    3757  (33)
                                                3827  (33)    3828  (33)    3899  (33)    3910  (33)
                                                3929  (33)    3944  (34)    4051  (34)    4072  (34)
                                                4091  (34)    4110  (34)    4112  (34)
ALL                     =00000008     85  (2)     1428  (8)     3722  (32)
ALL_MAPS                 0000192A-R   1227  (7)     2143  (16)
A_BE0_BA                 000002B7-R    384  (2)  #-1910  (14)  #-1911  (14)  #-1914  (14)  #-1921  (14)
A_BE0_CC                 000002B3-R    383  (2)
A_BE0_CLR_ERR            000002AB-R    381  (2)
A_BE0_CR1                000002BF-R    386  (2)  #-1138  (7)     2039  (15)    2162  (16)    2183  (16)
                                                2263  (17)    2287  (17)    2371  (18)    2396  (18)
                                                2470  (19)    2493  (19)    2575  (20)    2598  (20)
                                                2681  (21)    2705  (21)    2800  (22)    2825  (22)
                                                2899  (23)    2922  (23)    2998  (24)    3021  (24)
                                                3098  (25)    3122  (25)    3228  (26)    3252  (26)
                                                3338  (27)    3362  (27)    3439  (28)    3508  (29)
                                                3600  (30)    3669  (31)    846   (6)
A_BE0_CR2                000002BB-R    385  (2)     3457  (28)    3541  (29)    3618  (30)    3686  (31)
                                                849   (6)     852   (6)     855   (6)     858   (6)
                                                861   (6)     864   (6)     867   (6)     956   (6)
A_BE0_DB                 000002AF-R    382  (2)     1896  (14)  #-1932  (14)  #-1933  (14)  #-1936  (14)
                                             #-1943  (14)  #-1953  (14)  #-2178  (16)  #-2180  (16)
                                             #-2194  (16)  #-2280  (17)  #-2283  (17)  #-2296  (17)
                                             #-2389  (18)  #-2392  (18)  #-2405  (18)  #-2698  (21)
                                             #-2701  (21)  #-2714  (21)  #-2818  (22)  #-2821  (22)
                                             #-2834  (22)  #-3115  (25)  #-3118  (25)  #-3131  (25)
                                             #-3245  (26)  #-3248  (26)  #-3261  (26)  #-3355  (27)
                                             #-3358  (27)  #-3371  (27)  #-3817  (33)  #-4082  (34)
                                                4084  (34)
A_BE3_CC                 000002C3-R    387  (2)  #-1396  (7)
A_BE3_CR1                000002C7-R    388  (2)  #-1397  (7)
A_BUFFERS_1              000000EB-R    257  (2)  #-3751  (32)  #-3839  (33)  #-3848  (33)  #-3891  (33)
                                             #-3899  (33)  #-3910  (33)  #-3921  (33)  #-3929  (33)
A_BUFFERS_2              000000FB-R    258  (2)  #-4001  (34)  #-4041  (34)  #-4051  (34)  #-4064  (34)
                                             #-4072  (34)  #-4091  (34)
A_BUFRS_DFR_1            000000CB-R    249  (2)  #-3750  (32)
A_BUFRS_DFR_2            000000DB-R    253  (2)
A_MAP_BGN               =00F26800      212  (2)  #-1208  (7)   #-1219  (7)   #-1232  (7)   #-1262  (7)
                                             #-1264  (7)   #-1265  (7)   #-1568  (10)  #-1569  (10)
                                             #-1579  (10)  #-1686  (12)  #-1697  (12)  #-1765  (13)
                                             #-2258  (17)  #-2259  (17)  #-2301  (17)  #-2357  (18)
                                             #-2358  (18)  #-2664  (21)  #-2665  (21)  #-2789  (22)
                                             #-3223  (26)  #-3224  (26)  #-3266  (26)  #-3333  (27)
                                             #-3334  (27)  #-3376  (27)  #-3424  (28)  #-3425  (28)
                                             #-3426  (28)  #-3428  (28)  #-3523  (29)  #-3595  (30)
                                             #-3596  (30)  #-3662  (31)  #-798   (6)   #-831   (6)
                                             #-920   (6)
A_MAP_END               =00F26FFC      213  (2)  #-1813  (13)
A_MEM_CSR_1             =00F20004      292  (2)  #-3519  (29)  #-3524  (29)
A_SIMUL_GO              =00FFF00C      215  (2)  #-3873  (33)  #-4032  (34)
A_UBE_ADR_TBL            000002AB-R    380  (2)     1135  (7)
A_UBE_BUFFER_0           00000000-XR           #-2139  (16)  #-2140  (16)  #-2245  (17)  #-2246  (17)
                                             #-2351  (18)  #-2454  (19)  #-2455  (19)    249  (2)
                                                250  (2)     253  (2)   #-2558  (20)  #-2559  (20)
                                             #-2658  (21)  #-2659  (21)  #-2783  (22)  #-2784  (22)
                                             #-2885  (23)  #-2983  (24)  #-2984  (24)  #-3082  (25)
```

```
                                                    #-3083  (25)    #-3210  (26)      3211  (26)    #-3217  (26)
                                                    #-3221  (26)      3320  (27)    #-3321  (27)    #-3331  (27)
                                                    #-3419  (28)      3424  (28)    #-3503  (29)      3520  (29)
                                                    #-3584  (30)    #-3656  (31)    #-3664  (31)      3843  (33)
A_UBE_BUFFER_1          00000000-XR                   254   (2)     #-3433  (28)
A_UBE_BUFFER_2          00000000-XR                   251   (2)       255   (2)
A_UBE_BUFFER_3          00000000-XR                   252   (2)       256   (2)
A_UBI_CSR               =00F26010       211   (2)    1487   (9)    #-1498  (9)    #-1499  (9)    #-1509  (9)
                                                    #-1515  (9)    #-1516  (9)    #-1526  (9)    #-2164  (16)
                                                    #-2177  (16)   #-2265  (17)   #-2279  (17)   #-2373  (18)
                                                    #-2388  (18)   #-2472  (19)   #-2486  (19)   #-2577  (20)
                                                    #-2591  (20)   #-2683  (21)   #-2697  (21)   #-2802  (22)
                                                    #-2817  (22)   #-2901  (23)   #-2915  (23)   #-3000  (24)
                                                    #-3014  (24)   #-3100  (25)   #-3114  (25)   #-3230  (26)
                                                    #-3244  (26)   #-3340  (27)   #-3354  (27)   #-3441  (28)
                                                    #-3455  (28)   #-3458  (28)   #-3510  (29)   #-3539  (29)
                                                    #-3542  (29)   #-3602  (30)   #-3616  (30)   #-3619  (30)
                                                    #-3671  (31)   #-3685  (31)   #-959   (6)
A_UB_IO_SP              =00FC0000       214   (2)    #-1165  (7)
BASE_008                 00000000-R     1417  (8)      1428  (8)
BASE_009                 00000000-R     3704  (32)     3722  (32)
BEO_SERVICE              00001A74-R     1379  (7)      3738  (32)      3739  (32)      3740  (32)      3741  (32)
BEO_UNIT_NO              00000000-XR                   #-1430  (8)     #-1487  (9)     #-1494  (9)    #-1509  (9)
                                                    #-1526  (9)     #-1579  (10)    #-1715  (12)    #-1776  (13)
                                                    #-1793  (13)    #-1809  (13)    #-1896  (14)    #-1903  (14)
                                                    #-1921  (14)    #-1943  (14)    #-1966  (14)    #-2013  (15)
                                                    #-2017  (15)    #-2023  (15)    #-2027  (15)    #-2042  (15)
                                                    #-2054  (15)    #-2071  (15)    #-2129  (16)    #-2133  (16)
                                                    #-2194  (16)    #-2268  (17)    #-2296  (17)    #-2376  (18)
                                                    #-2405  (18)    #-2475  (19)    #-2502  (19)    #-2580  (20)
                                                    #-2607  (20)    #-2686  (21)    #-2714  (21)    #-2731  (21)
                                                    #-2805  (22)    #-2834  (22)    #-2904  (23)    #-2931  (23)
                                                    #-3003  (24)    #-3030  (24)    #-3103  (25)    #-3131  (25)
                                                    #-3148  (25)    #-3233  (26)    #-3261  (26)    #-3343  (27)
                                                    #-3371  (27)    #-3444  (28)    #-3463  (28)    #-3513  (29)
                                                    #-3547  (29)    #-3605  (30)    #-3624  (30)    #-3674  (31)
                                                    #-3690  (31)    #-3700  (31)    #-3820  (33)    #-3827  (33)
                                                    #-3899  (33)    #-3929  (33)    #-4051  (34)    #-4072  (34)
                                                    #-4091  (34)
BE1_UNIT_NO              00000000-XR                   #-3910  (33)
BE3_SERVICE              00001A7C-R     1391  (7)      3987  (34)
BIT...                  =0000001A       84    (2)      64    (2)       77    (2)       78    (2)       79    (2)
                                                      80    (2)       83    (2)       84    (2)
BR_SERVICE               00001A3C-R     1348  (7)      2013  (15)      2023  (15)      3700  (31)
B_DONE_CT                000000A2-R     237   (2)    #-1381  (7)     #-1399  (7)     #-3836  (33)    #-3876  (33)
                                                    #-3994  (34)    #-4035  (34)
B_MAPS_REQ_1             000000BF-R     246   (2)    #-3849  (33)
B_MAPS_REQ_2             000000C3-R     247   (2)    #-4002  (34)
CEP_FUNCTIONAL          =00000000       64    (2)
CEP_REPAIR              =00000001       64    (2)      64    (2)
CHANNEL_ERR              000016CC-R     999   (6)      2017  (15)      2027  (15)      2133  (16)
CHC$_CLEAR               00000000-XR                 #-2013  (15)
CHC$_DSINT               00000000-XR                 #-3700  (31)
CHC$_ENINT               00000000-XR                 #-2023  (15)    #-2129  (16)
CHECK_BUFFER             000019CA-R     1297  (7)      3892  (33)      3903  (33)      3922  (33)      4065  (34)
CHECK_BUF_0              00001A00-R     1322  (7)      4044  (34)
CHI$V_IPL                00000000-XR                 #-1355  (7)     #-988   (6)
```

```
CODEVECTORS                     00000297-R  360   (2)    #-1051  (7)    #-1103  (7)
CPU$V_COMET                     =00000002   77    (2)
CPU$V_HS                        =00000000   77    (2)
CPU$V_STAR                      =00000001   77    (2)
CSR_CHK                         00001471-R  843   (6)      835   (6)      924   (6)
CSR_ERR                         000015DF-R  934   (6)     1494   (9)     3463  (28)    3547  (29)    3624  (30)
                                                         3690  (31)
DATA_008                        0000001C-R  1428  (8)     1428   (8)
DATA_009                        0000001C-R  3722  (32)    3722  (32)
DATA_ERR                        000013C0-R  770   (6)     1509   (9)     1526   (9)    1579  (10)    1776  (13)
                                                         1793  (13)     1809  (13)    1921  (14)    1943  (14)
                                                         1966  (14)
DECODE_TBL                      00000277-R  354   (2)    #-1070  (7)    #-1095  (7)  #-1101  (7)  #-1113  (7)
                                                        #-1709  (12)   #-805   (6)
DEFAULT                         =00000000   85    (2)     1428   (8)     3722  (32)
DS$ABORT                        00010020    82    (2)
DS$ASKADR                       00010090    82    (2)
DS$ASKDATA                      00010080    82    (2)
DS$ASKLGCL                      00010098    82    (2)
DS$ASKSTR                       000100A0    82    (2)
DS$ASKVLD                       00010088    82    (2)
DS$ATTACH                       000101A8    82    (2)
DS$BGNSUB                       00010030    82    (2)     1479   (9)     1561  (10)    1679  (12)    1759  (13)
                                                         1887  (14)     2008  (15)    2124  (16)    2240  (17)
                                                         2347  (18)     2450  (19)    2554  (20)    2654  (21)
                                                         2779  (22)     2881  (23)    2979  (24)    3078  (25)
                                                         3205  (26)     3315  (27)    3414  (28)    3499  (29)
                                                         3581  (30)     3654  (31)    3807  (33)    3985  (34)
DS$BRANCH                       000100A8    82    (2)     1489   (9)     1898  (14)    2014  (15)    2024  (15)
                                                         2130  (16)     3822  (33)
DS$BREAK                        00010058    82    (2)     1175   (7)     3695  (31)    3834  (33)    4000  (34)
                                                         4110  (34)
DS$CANWAIT                      00010070    82    (2)     1350   (7)     1370   (7)
DS$CHANNEL                      00010180    82    (2)     2013  (15)     2023  (15)    2129  (16)    3700  (31)
DS$CKLOOP                       00010040    82    (2)     1511   (9)     1528   (9)    1581  (10)    1778  (13)
                                                         1795  (13)     1811  (13)    1923  (14)    1945  (14)
                                                         1968  (14)     2197  (16)    2299  (17)    2408  (18)
                                                         2505  (19)     2610  (20)    2734  (21)    2837  (22)
                                                         2934  (23)     3033  (24)    3151  (25)    3264  (26)
                                                         3374  (27)
DS$CLRVEC                       00010168    82    (2)     1189   (7)     4104  (34)    4105  (34)    4106  (34)
                                                         4107  (34)     4108  (34)
DS$CNTRLC                       00010078    82    (2)
DS$CVTREG                       000100B0    82    (2)
DS$ENDPASS                      00010010    82    (2)
DS$ENDSUB                       00010038    82    (2)     1530   (9)     1586  (10)    1719  (12)    1815  (13)
                                                         1974  (14)     2083  (15)    2203  (16)    2309  (17)
                                                         2419  (18)     2517  (19)    2621  (20)    2746  (21)
                                                         2848  (22)     2946  (23)    3045  (24)    3163  (25)
                                                         3272  (26)     3382  (27)    3465  (28)    3549  (29)
                                                         3626  (30)     3693  (31)    3940  (33)    4102  (34)
DS$ERRDEV                       000100C8    82    (2)
DS$ERRHARD                      000100D0    82    (2)     1494   (9)     1509   (9)    1526   (9)    1579  (10)
                                                         1715  (12)     1776  (13)    1793  (13)    1809  (13)
                                                         1903  (14)     1921  (14)    1943  (14)    1966  (14)
                                                         2017  (15)     2027  (15)    2042  (15)    2054  (15)
                                                         2071  (15)     2133  (16)    2166  (16)    2194  (16)
```

```
                                                    2268  (17)    2296  (17)    2376  (18)    2405  (18)
                                                    2475  (19)    2502  (19)    2580  (20)    2607  (20)
                                                    2686  (21)    2714  (21)    2731  (21)    2805  (22)
                                                    2834  (22)    2904  (23)    2931  (23)    3003  (24)
                                                    3030  (24)    3103  (25)    3131  (25)    3148  (25)
                                                    3233  (26)    3261  (26)    3343  (27)    3371  (27)
                                                    3444  (28)    3463  (28)    3513  (29)    3547  (29)
                                                    3605  (30)    3624  (30)    3674  (31)    3690  (31)
                                                    3827  (33)    3899  (33)    3910  (33)    3929  (33)
                                                    4051  (34)    4072  (34)    4091  (34)

DS$ERRPREP                  00010108      82  (2)
DS$ERRSOFT                  000100D8      82  (2)
DS$ERRSYS                   000100C0      82  (2)
DS$ESCAPE                   00010050      82  (2)
DS$FREEDBGSYM               000101B8      82  (2)
DS$GETBUF                   00010120      82  (2)
DS$GETMEM                   00010130      82  (2)
DS$GPHARD                   00010018      82  (2)
DS$HELP                     000101B0      82  (2)
DS$INITSCB                  00010170      82  (2)
DS$INLOOP                   00010048      82  (2)
DS$K_BNERROR                00000000-XR             #-1489  (9)    #-1898  (14)   #-2014  (15)   #-2024  (15)
                                                    #-2130  (16)   #-3822  (33)
DS$K_ERROR                  =00000002     78  (2)
DS$K_NORMAL                 =00000001     78  (2)
DS$K_SEVERE                 =00000004     78  (2)
DS$K_SUBSYS                 =00000066     78  (2)    78  (2)
DS$K_WARNING                =00000000     78  (2)
DS$LOAD                     00010198      82  (2)
DS$LOADPCS                  000101C0      82  (2)
DS$MAPDBGBLOCK              00010118      82  (2)
DS$MMOFF                    00010158      82  (2)
DS$MMON                     00010150      82  (2)
DS$MOVPHY                   00010148      82  (2)
DS$MOVVRT                   00010140      82  (2)
DS$PARSE                    000100B8      82  (2)
DS$PRINTB                   000100E0      82  (2)  1005  (6)    1010  (6)    1016  (6)    1030  (6)
                                                    2150  (16)   774   (6)    782   (6)    805   (6)
                                                    821   (6)    827   (6)    847   (6)    850   (6)
                                                    853   (6)    856   (6)    859   (6)    862   (6)
                                                    865   (6)    868   (6)    871   (6)    874   (6)
                                                    877   (6)    880   (6)    896   (6)    898   (6)
                                                    901   (6)    904   (6)    939   (6)    944   (6)
                                                    949   (6)    954   (6)    957   (6)    961   (6)
                                                    977   (6)    981   (6)    985   (6)    991   (6)
DS$PRINTF                   000100F0      82  (2)  1432  (8)    3729  (32)
DS$PRINTS                   000100F8      82  (2)
DS$PRINTSIG                 00010100      82  (2)
DS$PRINTX                   000100E8      82  (2)  833   (6)    905   (6)    914   (6)    922   (6)
DS$PROBE                    000101A0      82  (2)  1487  (9)    1896  (14)   3820  (33)
DS$RELBUF                   00010128      82  (2)
DS$RELMEM                   00010138      82  (2)
DS$SETIPL                   00010178      82  (2)  2034  (15)   2062  (15)   2081  (15)
DS$SETMAP                   00010188      82  (2)
DS$SETPRIEXV                00010110      82  (2)
DS$SETVEC                   00010160      82  (2)  1162  (7)    3738  (32)   3739  (32)   3740  (32)
                                                    3741  (32)   3745  (32)   3987  (34)
```

```
DS$SHOCHAN                      00010190      82      (2)
DS$SUMMARY                      00010028      82      (2)
DS$WAITMS                       00010060      82      (2)     2049    (15)    2065    (15)    2173    (16)    2275    (17)
                                                              2384    (18)    2482    (19)    2587    (20)    2693    (21)
                                                              2813    (22)    2911    (23)    3010    (24)    3110    (25)
                                                              3240    (26)    3350    (27)    3451    (28)    3535    (29)
                                                              3612    (30)    3681    (31)
DS$WAITUS                       00010068      82      (2)
DS$_ARITH                     =006600D0       78      (2)
DS$_BADLINK                   =006600F0       78      (2)
DS$_BADTYPE                   =006600E8       78      (2)
DS$_CHME                      =006600A8       78      (2)
DS$_CHMK                      =006600E0       78      (2)
DS$_DEVNAME                   =00660108       78      (2)
DS$_ERROR                     =00660002       78      (2)
DS$_FHWE                      =00660068       78      (2)
DS$_FRAGBUF                   =00660080       78      (2)
DS$_ICBUSY                    =006600C8       78      (2)
DS$_ICERR                     =006600C0       78      (2)
DS$_IHWE                      =00660060       78      (2)     #-1003  (6)
DS$_ILLCHAR                   =00660018       78      (2)
DS$_ILLPAGCNT                 =00660078       78      (2)
DS$_ILLUNIT                   =00660100       78      (2)
DS$_INSFMEM                   =00660050       78      (2)
DS$_IPL2HI                    =006600B8       78      (2)
DS$_IVADDR                    =00660040       78      (2)
DS$_IVVECT                    =00660038       78      (2)     #-1013  (6)
DS$_KRNLSTK                   =00660090       78      (2)
DS$_LOGIC                     =00660070       78      (2)     #-1008  (6)
DS$_MCHK                      =00660088       78      (2)
DS$_MMOFF                     =00660058       78      (2)
DS$_NEEDUNIT                  =006600F8       78      (2)
DS$_NOPCS                     =00660110       78      (2)
DS$_NORMAL                    =00660001       78      (2)
DS$_NOSUPPORT                 =006600B1       78      (2)
DS$_NOTDON                    =00660030       78      (2)
DS$_NOTIMP                    =006600B0       78      (2)       78    (2)
DS$_NULLSTR                   =00660010       78      (2)
DS$_OVERFLOW                  =00660008       78      (2)
DS$_POWER                     =00660098       78      (2)
DS$_PROGERR                   =00660020       78      (2)
DS$_SEVERE                    =00660004       78      (2)
DS$_TRANSL                    =006600A0       78      (2)
DS$_TRUNCATE                  =00660028       78      (2)
DS$_UNEXPINT                  =006600D8       78      (2)
DS$_VASFULL                   =00660048       78      (2)
DS$_WARNING                   =00660000       78      (2)       78    (2)
DSA$AL_APTMAIL                  0000FE00      77      (2)
DSA$AT_APTTXT                   0000FA00      77      (2)
DSA$GL_APTCOM                   0000FE04      77      (2)
DSA$GL_DEVLEN                   0000FE58      77      (2)
DSA$GL_ERRNO                    0000FE44      77      (2)
DSA$GL_EVENT                    0000FE48      77      (2)
DSA$GL_FLAGS                    0000FE00      77      (2)     3935    (33)    4096    (34)
DSA$GL_MSGTYP                   0000FE40      77      (2)
DSA$GL_PASSES                   0000FE08      77      (2)
DSA$GL_PASSNO                   0000FE54      77      (2)
```

```
DSA$GL_SECTNO                   0000FE10     77     (2)
DSA$GL_SID                      0000FE14     77     (2)
DSA$GL_SUBTNO                   0000FE4C     77     (2)
DSA$GL_TESTNO                   0000FE50     77     (2)
DSA$GL_UNITS                    0000FE0C     77     (2)
DSA$GQ_MSGPTR                   0000FE68     77     (2)
DSA$GT_DEVNAM                   0000FE5C     77     (2)
DSA$M_APT                      =80000000     77     (2)
DSA$M_BELL                     =00000008     77     (2)
DSA$M_BINARY                   =00000002     77     (2)
DSA$M_CRD_AUTOTEST             =00000800     77     (2)
DSA$M_CRD_MENUTEST             =00010000     77     (2)
DSA$M_CRD_TRACE                =00020000     77     (2)
DSA$M_DEBUG                    =04000000     77     (2)
DSA$M_HALT                     =00000001     77     (2)
DSA$M_IE1                      =00000010     77     (2)
DSA$M_IE2                      =00000020     77     (2)
DSA$M_IE3                      =00000040     77     (2)
DSA$M_IES                      =00000080     77     (2)
DSA$M_LOAD_DEBUGGER            =40000000     77     (2)
DSA$M_LOOP                     =00000004     77     (2)
DSA$M_NORPT                    =08000000     77     (2)
DSA$M_OPER                     =00001000     77     (2)
DSA$M_PASSO                    =20000000     77     (2)
DSA$M_PROMPT                   =00002000     77     (2)
DSA$M_QA                       =00008000     77     (2)
DSA$M_QUICK                    =00000100     77     (2)
DSA$M_SEARCH                   =00004000     77     (2)
DSA$M_TRACE                    =00000400     77     (2)
DSA$M_USER                     =10000000     77     (2)
DSA$M_VERIFY                   =00000200     77     (2)
DSA$V_APT                      =0000001F     77     (2)
DSA$V_BELL                     =00000003     77     (2)
DSA$V_BINARY                   =00000001     77     (2)
DSA$V_CRD_AUTOTEST             =0000000B     77     (2)
DSA$V_CRD_MENUTEST             =00000010     77     (2)
DSA$V_CRD_TRACE                =00000011     77     (2)
DSA$V_DEBUG                    =0000001A     77     (2)
DSA$V_HALT                     =00000000     77     (2)
DSA$V_IE1                      =00000004     77     (2)
DSA$V_IE2                      =00000005     77     (2)
DSA$V_IE3                      =00000006     77     (2)
DSA$V_IES                      =00000007     77     (2)
DSA$V_LOAD_DEBUGGER            =0000001E     77     (2)
DSA$V_LOOP                     =00000002     77     (2)
DSA$V_NORPT                    =0000001B     77     (2)
DSA$V_OPER                     =0000000C     77     (2)
DSA$V_PASSO                    =0000001D     77     (2)
DSA$V_PROMPT                   =0000000D     77     (2)
DSA$V_QA                       =0000000F     77     (2)
DSA$V_QUICK                    =00000008     77     (2)    #-3935    (33)    #-4096    (34)
DSA$V_SEARCH                   =0000000E     77     (2)
DSA$V_TRACE                    =0000000A     77     (2)
DSA$V_USER                     =0000001C     77     (2)
DSA$V_VERIFY                   =00000009     77     (2)
ENV$M_DOMAIN                   =00000002     64     (2)
ENV$M_LEVEL                    =00000001     64     (2)
```

```
ENV$M_SUPER                     =000003FC      64      (2)
ENV$S_DOMAIN                    =00000001      64      (2)
ENV$S_LEVEL                     =00000001      64      (2)
ENV$S_SUPER                     =00000008      64      (2)
ENV$V_DOMAIN                    =00000001      64      (2)     64      (2)
ENV$V_LEVEL                     =00000000      64      (2)     64      (2)
ENV$V_SUPER                     =00000002      64      (2)
ENV$_CPU                        =00000000      64      (2)     64      (2)
ENV$_FUNCTIONAL                 =00000000      64      (2)     64      (2)
ENV$_REPAIR                     =00000001      64      (2)     64      (2)
ENV$_SUPER                      =00000001      64      (2)
ENV$_SYSTEM                     =00000001      64      (2)     64      (2)
ERR$_MSGADR                     =0000000C      81      (2)
ERR$_NARGS                      =0000000A      81      (2)
ERR$_NUM                        =00000004      81      (2)
ERR$_P1                         =00000014      81      (2)     #-1030  (6)     #-782   (6)     #-827   (6)       898   (6)
ERR$_P2                         =00000018      81      (2)     #-776   (6)     #-782   (6)     #-819   (6)     #-827   (6)
                                                               #-901   (6)
ERR$_P3                         =0000001C      81      (2)     #-776   (6)     #-782   (6)     #-819   (6)     #-827   (6)
ERR$_P4                         =00000020      81      (2)       782   (6)       827   (6)
ERR$_P5                         =00000024      81      (2)     #-829   (6)
ERR$_P6                         =00000028      81      (2)
ERR$_POINTER                    =00000010      81      (2)
ERR$_UNIT                       =00000008      81      (2)
FRST_VAL_MAP                     00000002-R    228     (2)     #-1181  (7)     #-1184  (7)     #-2148  (16)    #-2158  (16)
                                                               #-2194  (16)    #-2356  (18)    #-2366  (18)    #-2405  (18)
                                                               #-2464  (19)    #-2502  (19)    #-2571  (20)    #-2607  (20)
                                                               #-2663  (21)    #-2676  (21)    #-2714  (21)    #-2731  (21)
                                                               #-2788  (22)    #-2796  (22)    #-2834  (22)    #-2894  (23)
                                                               #-2931  (23)    #-2994  (24)    #-3030  (24)    #-3094  (25)
                                                               #-3131  (25)    #-3148  (25)    #-3423  (28)    #-3435  (28)
                                                               #-3522  (29)    #-3530  (29)    #-3587  (30)    #-3593  (30)
                                                               #-3661  (31)
HALT                            =00000004      85      (2)
HP$A_DEPENDENT                   00000032      83      (2)       84    (2)
HP$A_DEVICE                      00000018      83      (2)
HP$A_DVA                         0000001C      83      (2)
HP$A_LINK                        00000020      83      (2)
HP$B_DRIVE                       0000000B      83      (2)
HP$B_FLAGS                       0000000A      83      (2)
HP$M_ALLOC                      =00000001      83      (2)
HP$M_WASALL                     =00000002      83      (2)
HP$Q_DEVICE                      00000000      83      (2)
HP$T_DEVICE                      0000000C      83      (2)
HP$T_TYPE                        00000026      83      (2)
HP$V_ALLOC                      =00000000      83      (2)
HP$V_WASALL                     =00000001      83      (2)
HP$W_SIZE                        00000008      83      (2)
HP$W_VECTOR                      00000024      83      (2)
INTR_ERR                         0000166E-R    970     (6)     2054    (15)    2071    (15)
INTR_SERVICE                     00001A6C-R    1368    (7)     2129    (16)
INVAL_MAPS                       000018FB-R    1205    (7)     2244    (17)    2355    (18)    3209    (26)    3319    (27)
IPR                             =00000006      85      (2)
KA$B_ACC_TYPE                    00000042      84      (2)
KA$B_RESERVED                    00000043      84      (2)
KA$B_SID_TYPE                    0000003D      84      (2)
KA$K_LEN                         00000046      84      (2)
```

```
KA$L_FLAGS                      00000032      84      (2)
KA$L_SID                        0000003A      84      (2)
KA$M_CHAR                      =00000100      84      (2)
KA$M_CRC                       =00000010      84      (2)
KA$M_D_FLOAT                   =00000002      84      (2)
KA$M_EDIT                      =00000080      84      (2)
KA$M_F_FLOAT                   =00000001      84      (2)
KA$M_G_FLOAT                   =00000004      84      (2)
KA$M_H_FLOAT                   =00000008      84      (2)
KA$M_PACKED                    =00000020      84      (2)
KA$M_PACK_MUL                  =00000040      84      (2)
KA$M_SB_ERR                    =02000000      84      (2)
KA$M_TODR                      =00010000      84      (2)
KA$M_WCS_LD                    =01000000      84      (2)
KA$V_CHAR                      =00000008      84      (2)
KA$V_CRC                       =00000004      84      (2)
KA$V_D_FLOAT                   =00000001      84      (2)
KA$V_EDIT                      =00000007      84      (2)
KA$V_F_FLOAT                   =00000000      84      (2)
KA$V_G_FLOAT                   =00000002      84      (2)
KA$V_H_FLOAT                   =00000003      84      (2)
KA$V_PACKED                    =00000005      84      (2)
KA$V_PACK_MUL                  =00000006      84      (2)
KA$V_SB_ERR                    =00000019      84      (2)
KA$V_TODR                      =00000010      84      (2)
KA$V_WCS_LD                    =00000018      84      (2)
KA$W_LATENCY                    0000003E      84      (2)
KA$W_MEM_SIZE                   00000044      84      (2)
KA$W_PROGRESS                   00000040      84      (2)
KA$W_RESERVED                   00000038      84      (2)
KA$W_WCS                        00000036      84      (2)
LDPCTX                         =0000000B      85      (2)
LOAD_BUFFER                     000019B5-R    1280    (7)       3842  (33)      4015  (34)
L_BR_LVL                        000000AB-R    240     (2)   #-1355  (7)   #-2032  (15)  #-2077  (15)  #-2078  (15)
                                              #-977   (6)
L_BUFFER_ADR                    000000C7-R    248     (2)     1252  (7)   #-1284  (7)   #-1301  (7)     1305  (7)
                                              #-1327  (7)     1331  (7)   #-3839  (33)  #-3848  (33)
                                              #-3891  (33)  #-3901  (33)  #-3921  (33)  #-4001  (34)
                                              #-4041  (34)  #-4064  (34)
L_BUFFER_SIZE                   0000010B-R    259     (2)   #-1285  (7)   #-1308  (7)   #-1335  (7)   #-3838  (33)
                                              #-3890  (33)  #-3919  (33)  #-4014  (34)  #-4042  (34)
                                              #-4063  (34)
L_BUF_SIZES_1                   0000010F-R    260     (2)   #-3838  (33)  #-3890  (33)  #-3901  (33)  #-3919  (33)
L_BUF_SIZES_2                   0000011F-R    264     (2)   #-4014  (34)  #-4042  (34)  #-4063  (34)
L_ERROR_CT                      0000015F-R    278     (2)   #-1309  (7)   #-1336  (7)   #-4085  (34)  #-904   (6)
                                              #-917   (6)
L_ERROR_LOG                     00000163-R    279     (2)   #-1305  (7)   #-1331  (7)   #-4084  (34)  #-907   (6)
                                              #-914   (6)
L_EXP_DATA                      0000005E-R    232     (2)   #-1301  (7)   #-1306  (7)   #-1332  (7)   #-2460  (19)
                                              #-2462  (19)  #-2487  (19)  #-2502  (19)  #-2561  (20)
                                              #-2566  (20)  #-2592  (20)  #-2607  (20)  #-2671  (21)
                                                2672  (21)  #-2673  (21)  #-2698  (21)  #-2714  (21)
                                              #-2720  (21)  #-2731  (21)  #-2890  (23)  #-2892  (23)
                                              #-2916  (23)  #-2931  (23)  #-2990  (24)  #-2992  (24)
                                              #-3015  (24)  #-3030  (24)  #-3089  (25)    3090  (25)
                                              #-3091  (25)  #-3115  (25)  #-3131  (25)  #-3137  (25)
                                              #-3148  (25)  #-3885  (33)    3886  (33)    3887  (33)
```

```
                                                    3888  (33)     3889  (33)   #-3902  (33)   #-3920  (33)
                                                #-4060  (34)     4061  (34)     4062  (34)   #-4081  (34)
                                                #-4082  (34)   #-909   (6)    #-914   (6)
L_INFO_BIT                      0000018B-R  281 (2)     #-1075  (7)    #-1077  (7)    #-1086  (7)    #-1096  (7)
L_INIT_MAP                      0000014F-R  277 (2)     #-1261  (7)    #-3859  (33)   #-4004  (34)   #-919   (6)
L_IPL                           000000A7-R  239 (2)     #-2033  (15)   #-2034  (15)   #-2061  (15)   #-2062  (15)
                                                #-977   (6)    #-981   (6)
L_LAST_UBE                      00000000-XR             #-3727  (32)   #-3830  (33)   #-3861  (33)   #-3876  (33)
                                                #-3916  (33)   #-3931  (33)   #-4018  (34)   #-4035  (34)
                                                #-4057  (34)   #-4078  (34)
L_MAPS_NEEDED                   000000A3-R  238 (2)     #-1256  (7)    #-1267  (7)    #-3849  (33)   #-4002  (34)
L_MAP_NUM                       0000008A-R  234 (2)     #-1251  (7)    #-1268  (7)    #-3810  (33)   #-3936  (33)
                                                #-3990  (34)   #-4097  (34)
L_MXFR_TIME                     000000BB-R  245 (2)     #-3867  (33)   #-4026  (34)
L_NXM_ADR                       00000086-R  233 (2)     #-3592  (30)     3594  (30)
L_PAGE_NO                       0000004A-R  230 (2)     #-1229  (7)    #-1252  (7)    #-1262  (7)    #-1263  (7)
                                                #-2142  (16)   #-2247  (17)   #-2258  (17)   #-2357  (18)
                                                #-2664  (21)   #-3211  (26)   #-3223  (26)   #-3320  (27)
                                                #-3333  (27)   #-3520  (29)   #-3521  (29)   #-3523  (29)
                                                #-3594  (30)   #-3595  (30)   #-3843  (33)
L_PAIR_SEP                      0000018F-R  282 (2)     #-1074  (7)    #-1079  (7)    #-1082  (7)    #-1091  (7)
L_PATTERN                      =33550FFF    291 (2)     #-3811  (33)   #-3991  (34)
L_POP                           0000019F-R  286 (2)     #-1076  (7)    #-1085  (7)    #-1093  (7)    #-1106  (7)
                                                #-1109  (7)    #-1111  (7)
L_POSITION                      00000193-R  283 (2)     #-1080  (7)    #-1081  (7)    #-1082  (7)    #-1083  (7)
                                                #-1087  (7)    #-1091  (7)    #-1092  (7)
L_PTRN                          0000008E-R  235 (2)     #-3811  (33)   #-3840  (33)   #-3841  (33)   #-3855  (33)
                                                #-3856  (33)   #-3991  (34)   #-4007  (34)   #-4008  (34)
L_PTRNS                         00000092-R  236 (2)     #-1283  (7)    #-3841  (33)   #-3856  (33)   #-3857  (33)
                                                #-3885  (33)   #-3902  (33)   #-3920  (33)   #-4008  (34)
                                                #-4011  (34)   #-4060  (34)   #-4081  (34)
L_RETRIEVE                      000001A3-R  287 (2)     #-1071  (7)      1072  (7)    #-1073  (7)      1081  (7)
                                                  1083  (7)    #-1104  (7)      1108  (7)
L_SAVE_CSR                      0000004E-R  231 (2)     #-2164  (16)   #-2177  (16)   #-2184  (16)   #-2265  (17)
                                                #-2279  (17)   #-2285  (17)   #-2373  (18)   #-2388  (18)
                                                #-2394  (18)   #-2472  (19)   #-2486  (19)   #-2491  (19)
                                                #-2577  (20)   #-2591  (20)   #-2596  (20)   #-2683  (21)
                                                #-2697  (21)   #-2703  (21)   #-2802  (22)   #-2817  (22)
                                                #-2823  (22)   #-2901  (23)   #-2915  (23)   #-2920  (23)
                                                #-3000  (24)   #-3014  (24)   #-3019  (24)   #-3100  (25)
                                                #-3114  (25)   #-3120  (25)   #-3230  (26)   #-3244  (26)
                                                #-3250  (26)   #-3340  (27)   #-3354  (27)   #-3360  (27)
                                                #-3441  (28)   #-3455  (28)     3456  (28)   #-3510  (29)
                                                #-3539  (29)     3540  (29)   #-3602  (30)   #-3616  (30)
                                                  3617  (30)   #-3671  (31)   #-3685  (31)     870  (6)
                                                   873  (6)      876  (6)      879  (6)      943  (6)
                                                   948  (6)      953  (6)
L_TEMP_A                        00000197-R  284 (2)     #-1081  (7)    #-1084  (7)    #-1085  (7)    #-1102  (7)
                                                #-1103  (7)    #-1104  (7)
L_TEMP_B                        0000019B-R  285 (2)     #-1083  (7)    #-1084  (7)    #-1101  (7)      1102  (7)
L_UBI_STATUS                    000000AF-R  241 (2)     #-1003  (6)    #-1008  (6)    #-1013  (6)      2013  (15)
                                                  2023  (15)     2129  (16)     3700  (31)
L_XFR_COUNT                     000000B7-R  244 (2)     #-1393  (7)    #-1394  (7)    #-3992  (34)
MANUAL                         =00000001    85  (2)
MAP_ADR_ERR                     000013E9-R  792 (6)       1715  (12)
MAP_ENT_TBL                     000001A7-R  303 (2)       1567  (10)
MCHK                           =00000005    85  (2)
```

```
MEM                             =00000009     85    (2)
MSG_008                          00000002-R   1417  (8)     1428  (8)
MSG_009                          00000002-R   3704  (32)    3722  (32)
MXFR_ERR                         0000154A-R   892   (6)     3899  (33)    3910  (33)    3929  (33)    4051  (34)
                                                            4072  (34)    4091  (34)
M_BR4                           =00000002     156   (2)     453   (2)
M_BR5                           =00000004     157   (2)     446   (2)     481   (2)     502   (2)
M_BR6                           =00000008     158   (2)     439   (2)     460   (2)     488   (2)
M_BR7                           =00000010     159   (2)   #-1397  (7)     397   (2)     404   (2)     411   (2)
                                                            418   (2)     425   (2)     432   (2)     467   (2)
                                                            474   (2)     495   (2)     509   (2)
M_BR_INTR                       =00000008     122   (2)   #-1351  (7)
M_BYTE_OFF                      =02000000     95    (2)   #-1055  (7)   #-1265  (7)   #-2789  (22)  #-3334  (27)
                                                          #-3426  (28)
M_CCOVF                         =00002000     199   (2)
M_CC_DAT                        =00001000     172   (2)     488   (2)
M_CLK_ERR                       =80000000     110   (2)   #-1410  (7)   #-3866  (33)  #-3871  (33)  #-3878  (33)
                                                          #-4025  (34)  #-4030  (34)  #-4037  (34)
M_CLK_INTR                      =00000080     108   (2)   #-1410  (7)   #-3866  (33)  #-3878  (33)  #-4025  (34)
                                                          #-4037  (34)
M_DATA_ERR                      =00001000     134   (2)   #-2181  (16)  #-2282  (17)  #-2391  (18)  #-2489  (19)
                                                          #-2594  (20)  #-2700  (21)  #-2722  (21)  #-2820  (22)
                                                          #-2918  (23)  #-3017  (24)  #-3117  (25)  #-3140  (25)
                                                          #-3247  (26)  #-3357  (27)
M_DATI                          =00000000     164   (2)   #-1397  (7)     397   (2)     502   (2)     509   (2)
M_DATIP                         =00000100     165   (2)     404   (2)     460   (2)     467   (2)     495   (2)
M_DATO                          =00000200     166   (2)     418   (2)     425   (2)     474   (2)     481   (2)
M_DATOB                         =00000300     167   (2)     411   (2)     488   (2)
M_DATOB_IP                      =00004000     174   (2)     460   (2)
M_DIAG_MODE                     =04000000     100   (2)
M_DSABL_ECC                     =02000000     99    (2)
M_ENABLE                        =00000040     106   (2)   #-3871  (33)  #-4030  (34)
M_ENB_PB                        =00001000     198   (2)     424   (2)
M_ERR                           =00008000     175   (2)
M_EXP_BR_INTR                   =00000002     118   (2)   #-2064  (15)
M_EXP_NXM_ERR                   =00000200     130   (2)   #-3586  (30)
M_EXP_PB_ERR                    =00000400     132   (2)   #-3504  (29)
M_EXP_WR_ERR                    =00000080     128   (2)   #-3417  (28)
M_EXT_MEM                       =00000000     180   (2)
M_FRCE_TB_ERR                   =20000000     101   (2)   #-3519  (29)  #-3524  (29)
M_GO                            =00000001     155   (2)   #-1397  (7)     397   (2)     404   (2)     411   (2)
                                                            418   (2)     425   (2)     432   (2)     439   (2)
                                                            446   (2)     453   (2)
M_INH_INC                       =00000004     181   (2)
M_INH_ROL                       =00002000     173   (2)     467   (2)
M_INH_SACK                      =00000008     182   (2)
M_INTR_ERR                      =00000010     124   (2)   #-1354  (7)   #-1357  (7)
M_INT_BM                        =00000000     168   (2)     432   (2)     439   (2)     446   (2)     453   (2)
M_INT_ODNE                      =00000040     161   (2)   #-1397  (7)     397   (2)     404   (2)     411   (2)
                                                            418   (2)     425   (2)     460   (2)     467   (2)
                                                            474   (2)     481   (2)     488   (2)     495   (2)
                                                            502   (2)     509   (2)
M_INT_SSYN                      =00000800     196   (2)
M_MAP_NOWRT                     =7DFF8000     94    (2)   #-1569  (10)  #-1697  (12)  #-1767  (13)  #-1783  (13)
                                                          #-1800  (13)
M_MAP_VAL                       =80000000     97    (2)   #-1219  (7)   #-1229  (7)   #-1264  (7)   #-2259  (17)
                                                          #-2358  (18)  #-2665  (21)  #-3224  (26)  #-3334  (27)
```

```
                                                #-3425  (28)  #-3521  (29)  #-3596  (30)
M_MAP_WRTBL     =82007FFF   93   (2)            #-1782  (13)  #-1784  (13)  #-1793  (13)
M_MAX_LAT       =00000040   186  (2)
M_NO_CSR        =00000001   116  (2)            #-1491  (9)
M_NO_GNT        =00000400   194  (2)
M_NO_SACK       =00000080   188  (2)
M_NO_SSYN       =00000100   190  (2)
M_NPR           =00000020   160  (2)              397   (2)     404   (2)     411   (2)     418   (2)
                                                 425   (2)     467   (2)     474   (2)     488   (2)
                                                 495   (2)
M_ONE_XFR       =00000400   169  (2)            #-1397  (7)     397   (2)     404   (2)     411   (2)
                                                 418   (2)     460   (2)     467   (2)     474   (2)
                                                 481   (2)     495   (2)     509   (2)
M_PWR_DN        =00000010   183  (2)
M_RDY           =00000080   162  (2)            #-1397  (7)     397   (2)     404   (2)     411   (2)
                                                 418   (2)     425   (2)     432   (2)     439   (2)
                                                 446   (2)     453   (2)     460   (2)     467   (2)
                                                 474   (2)     481   (2)     488   (2)     495   (2)
                                                 502   (2)     509   (2)
M_RLS_BUS       =00000C00   171  (2)
M_RUN           =00000001   102  (2)            #-3871  (33)  #-4030  (34)
M_TIMEOUT       =00000004   120  (2)            #-1411  (7)   #-3870  (33)  #-4029  (34)
M_TM_DLY        =00004000   201  (2)
M_TRANSFER      =00000010   104  (2)            #-3868  (33)  #-4027  (34)
M_TWO_XFR       =00000800   170  (2)              488   (2)     502   (2)
M_UBI_NON_STS   =7FFE3FFF   149  (2)            #-1499  (9)   #-1516  (9)
M_UBI_NXM_ERR   =00010000   142  (2)
M_UBI_PB_ERR    =00008000   144  (2)
M_UBI_RDS_ERR   =80000000   140  (2)
M_UBI_STS       =8001C000   148  (2)            #-1515  (9)   #-2184  (16)  #-2285  (17)  #-2394  (18)
                                                #-2491  (19)  #-2596  (20)  #-2703  (21)  #-2823  (22)
                                                #-2920  (23)  #-3019  (24)  #-3120  (25)  #-3250  (26)
                                                #-3360  (27)  #-3458  (28)  #-3542  (29)  #-3619  (30)
                                                #-959   (6)
M_UBI_WR_ERR    =00004000   146  (2)
M_UNX_SSYN_ERR  =00000040   126  (2)
M_WNG_A         =00000200   192  (2)
M_WNG_GNT       =00000020   184  (2)
NO_ACCESS        0000171B-R 1024 (6)              1903  (14)    3827  (33)
PAR$M_ATDEF     =00000010   79   (2)
PAR$M_ATHI      =00000008   79   (2)
PAR$M_ATLO      =00000004   79   (2)
PAR$M_NODEF     =00000002   79   (2)
PAR$V_ATDEF     =00000004   79   (2)
PAR$V_ATHI      =00000003   79   (2)
PAR$V_ATLO      =00000002   79   (2)
PAR$V_NODEF     =00000001   79   (2)
PAR$_BIN        =00000002   79   (2)
PAR$_DEC        =0000000A   79   (2)
PAR$_HEX        =00000010   79   (2)
PAR$_NO         =00000000   79   (2)
PAR$_OCT        =00000008   79   (2)
PAR$_YES        =00000001   79   (2)
POWER           =00000003   85   (2)
PR$L_UB_INIT    =00000037   293  (2)            #-2030  (15)  #-2137  (16)  #-2243  (17)  #-2349  (18)
                                                #-2452  (19)  #-2556  (20)  #-2656  (21)  #-2781  (22)
                                                #-2883  (23)  #-2981  (24)  #-3080  (25)  #-3208  (26)
```

				#-3318	(27)	#-3418	(28)	#-3501	(29)	#-3583	(30)	
				#-3809	(33)	#-3989	(34)					
PR$_ICCS	00000000-XR			#-1410	(7)	#-3866	(33)	#-3868	(33)	#-3871	(33)	
				#-3878	(33)	#-4025	(34)	#-4027	(34)	#-4030	(34)	
				#-4037	(34)							
PR$_MCESR	00000000-XR			#-1197	(7)							
PR$_NICR	00000000-XR			#-3867	(33)	#-3869	(33)	#-4026	(34)	#-4028	(34)	
PSL$C_EXEC	=00000001	80	(2)									
PSL$C_KERNEL	=00000000	80	(2)									
PSL$C_SUPER	=00000002	80	(2)									
PSL$C_USER	=00000003	80	(2)									
PSL$K_EXEC	=00000001	80	(2)									
PSL$K_KERNEL	=00000000	80	(2)									
PSL$K_SUPER	=00000002	80	(2)									
PSL$K_USER	=00000003	80	(2)									
PSL$M_C	=00000001	80	(2)									
PSL$M_CBIT	=00000001	80	(2)									
PSL$M_CM	=80000000	80	(2)	80	(2)							
PSL$M_CURMOD	=03000000	80	(2)									
PSL$M_DV	=00000080	80	(2)									
PSL$M_FPD	=08000000	80	(2)	80	(2)							
PSL$M_FU	=00000040	80	(2)									
PSL$M_IPL	=001F0000	80	(2)									
PSL$M_IS	=04000000	80	(2)									
PSL$M_IV	=00000020	80	(2)									
PSL$M_N	=00000008	80	(2)									
PSL$M_NBIT	=00000008	80	(2)									
PSL$M_PRVMOD	=00C00000	80	(2)									
PSL$M_SAFBITS	=000037FF	80	(2)									
PSL$M_TBIT	=00000010	80	(2)									
PSL$M_TP	=40000000	80	(2)	80	(2)							
PSL$M_V	=00000002	80	(2)									
PSL$M_VBIT	=00000002	80	(2)									
PSL$M_Z	=00000004	80	(2)									
PSL$M_ZBIT	=00000004	80	(2)									
PSL$S_CURMOD	=00000002	80	(2)									
PSL$S_IPL	=00000005	80	(2)									
PSL$S_PRVMOD	=00000002	80	(2)									
PSL$V_C	=00000000	80	(2)									
PSL$V_CBIT	=00000000	80	(2)									
PSL$V_CM	=0000001F	80	(2)									
PSL$V_CURMOD	=00000018	80	(2)									
PSL$V_DV	=00000007	80	(2)									
PSL$V_FPD	=0000001B	80	(2)									
PSL$V_FU	=00000006	80	(2)									
PSL$V_IPL	=00000010	80	(2)									
PSL$V_IS	=0000001A	80	(2)									
PSL$V_IV	=00000005	80	(2)									
PSL$V_N	=00000003	80	(2)									
PSL$V_NBIT	=00000003	80	(2)									
PSL$V_PRVMOD	=00000016	80	(2)									
PSL$V_TBIT	=00000004	80	(2)									
PSL$V_TP	=0000001E	80	(2)									
PSL$V_V	=00000001	80	(2)									
PSL$V_VBIT	=00000001	80	(2)									
PSL$V_Z	=00000002	80	(2)									
PSL$V_ZBIT	=00000002	80	(2)									

```
RD_MLLR_DECODE                  0000175E-R    1065    (7)       1707    (12)
RD_MLLR_ENCODE                  0000172F-R    1045    (7)       1683    (12)
RD_MLLR_TBL                     00000237-R    344     (2)    #-1056     (7)    #-1686     (12)    #-805     (6)
RETRIEVE_TBL                    00000257-R    349     (2)    #-1071     (7)    #-1697     (12)    #-805     (6)
SCB$L_ACCESS                     00000020     76      (2)
SCB$L_ARITH                      00000034     76      (2)
SCB$L_BE0                       =00000348     220     (2)    #-3738     (32)   #-4104     (34)
SCB$L_BE1                       =0000034C     221     (2)    #-3739     (32)   #-4105     (34)
SCB$L_BE2                       =00000350     222     (2)    #-3740     (32)   #-4106     (34)
SCB$L_BE3                       =00000354     223     (2)    #-3741     (32)   #-3987     (34)    #-4107    (34)
SCB$L_BREAK                      0000002C     76      (2)
SCB$L_CHME                       00000044     76      (2)
SCB$L_CHMK                       00000040     76      (2)
SCB$L_CHMS                       00000048     76      (2)
SCB$L_CHMU                       0000004C     76      (2)
SCB$L_COMPAT                     00000030     76      (2)
SCB$L_KNLSTK                     00000008     76      (2)
SCB$L_MACHCHK                    00000004     76      (2)    #-1162     (7)    #-1189     (7)
SCB$L_OPCCUS                     00000014     76      (2)
SCB$L_OPCDEC                     00000010     76      (2)
SCB$L_POWER                      0000000C     76      (2)
SCB$L_RADRMOD                    0000001C     76      (2)
SCB$L_ROPRAND                    00000018     76      (2)
SCB$L_RXDB                       000000F8     76      (2)
SCB$L_SFTLVL1                    00000084     76      (2)
SCB$L_SFTLVL10                   000000A8     76      (2)
SCB$L_SFTLVL11                   000000AC     76      (2)
SCB$L_SFTLVL12                   000000B0     76      (2)
SCB$L_SFTLVL13                   000000B4     76      (2)
SCB$L_SFTLVL14                   000000B8     76      (2)
SCB$L_SFTLVL15                   000000BC     76      (2)
SCB$L_SFTLVL2                    00000088     76      (2)
SCB$L_SFTLVL3                    0000008C     76      (2)
SCB$L_SFTLVL4                    00000090     76      (2)
SCB$L_SFTLVL5                    00000094     76      (2)
SCB$L_SFTLVL6                    00000098     76      (2)
SCB$L_SFTLVL7                    0000009C     76      (2)
SCB$L_SFTLVL8                    000000A0     76      (2)
SCB$L_SFTLVL9                    000000A4     76      (2)
SCB$L_TBIT                       00000028     76      (2)
SCB$L_TIMER                      000000C0     76      (2)    #-3745     (32)   #-4108     (34)
SCB$L_TRANSL                     00000024     76      (2)
SCB$L_TXDB                       000000FC     76      (2)
SCB$L_ZERO                       00000000     76      (2)
SEP_FUNCTIONAL                  =00000002     64      (2)
SEP_REPAIR                      =00000003     64      (2)
SET_UP_MAPS                      0000194D-R    1247    (7)       3850    (33)      4003    (34)
SIZ...                          =00000001     84      (2)       64      (2)       77      (2)       79    (2)    80    (2)
                                                                 83      (2)       84      (2)
SYS$ALLOC                        00010238     82      (2)
SYS$ASCTIM                       00010248     82      (2)
SYS$ASSIGN                       00010250     82      (2)
SYS$BINTIM                       00010258     82      (2)
SYS$CANCEL                       00010260     82      (2)
SYS$CANTIM                       00010268     82      (2)
SYS$CLOSE                        000105B8     82      (2)
SYS$CLREF                        00010298     82      (2)
```

```
SYS$CONNECT                     000105C0      82      (2)
SYS$DALLOC                      000102D8      82      (2)
SYS$DASSGN                      000102E0      82      (2)
SYS$DISCONNECT                  000105D0      82      (2)
SYS$FAO                         00010350      82      (2)
SYS$FAOL                        00010358      82      (2)
SYS$GET                         00010580      82      (2)
SYS$GETCHN                      000104C8      82      (2)
SYS$GETTIM                      00010378      82      (2)
SYS$LKWSET                      000103A0      82      (2)
SYS$NUMTIM                      000103B8      82      (2)
SYS$OPEN                        00010608      82      (2)
SYS$QIO                         000103C8      82      (2)
SYS$QIOW                        00010200      82      (2)
SYS$READ                        00010590      82      (2)
SYS$READEF                      000103D0      82      (2)
SYS$SETAST                      000103F8      82      (2)
SYS$SETEF                       00010400      82      (2)
SYS$SETIMR                      00010420      82      (2)
SYS$SETPRI                      00010428      82      (2)
SYS$SETPRT                      00010430      82      (2)
SYS$SETRWM                      00010438      82      (2)
SYS$SETSWM                      00010448      82      (2)
SYS$ULKPAG                      00010460      82      (2)
SYS$ULWSET                      00010468      82      (2)
SYS$UNWIND                      00010470      82      (2)
SYS$WAITFR                      00010478      82      (2)
SYS$WFLAND                      00010488      82      (2)
SYS$WFLOR                       00010490      82      (2)
T8_S1                           0000003C-R    1479    (9)
T8_S10                          00000A61-R    2450    (19)
T8_S10_X                        00000B76-R    2517    (19)
T8_S11                          00000B81-R    2554    (20)
T8_S11_X                        00000C9A-R    2621    (20)
T8_S12                          00000CA5-R    2654    (21)
T8_S12_X                        00000E3B-R    2746    (21)
T8_S13                          00000E46-R    2779    (22)
T8_S13_X                        00000F69-R    2848    (22)
T8_S14                          00000F74-R    2881    (23)
T8_S14_X                        00001085-R    2946    (23)
T8_S15                          00001090-R    2979    (24)
T8_S15_X                        000011A7-R    3045    (24)
T8_S16                          000011B2-R    3078    (25)
T8_S16_X                        0000132B-R    3163    (25)
T8_S17                          00001336-R    3205    (26)
T8_S17_X                        0000147F-R    3272    (26)
T8_S18                          0000148A-R    3315    (27)
T8_S18_X                        000015D0-R    3382    (27)
T8_S19                          000015DB-R    3414    (28)
T8_S19_X                        000016D6-R    3465    (28)
T8_S1_X                         0000012F-R    1530    (9)
T8_S2                           0000013A-R    1561    (10)
T8_S20                          000016E1-R    3499    (29)
T8_S20_X                        000017DD-R    3549    (29)
T8_S21                          000017E8-R    3581    (30)
T8_S21_X                        000018D5-R    3626    (30)
T8_S22                          000018E0-R    3654    (31)
```

```
T8_S22_X                        0000198A-R  3693  (31)
T8_S2_X                         000001A6-R  1586  (10)
T8_S3                           000001B1-R  1679  (12)
T8_S3_X                         00000244-R  1719  (12)
T8_S4                           0000024F-R  1759  (13)
T8_S4_X                         00000332-R  1815  (13)
T8_S5                           0000033D-R  1887  (14)
T8_S5_X                         0000049E-R  1974  (14)
T8_S6                           000004A9-R  2008  (15)
T8_S6_X                         00000652-R  2083  (15)
T8_S7                           0000065D-R  2124  (16)
T8_S7_X                         000007D9-R  2203  (16)
T8_S8                           000007E4-R  2240  (17)
T8_S8_X                         0000091F-R  2309  (17)
T8_S9                           0000092A-R  2347  (18)
T8_S9_X                         00000A56-R  2419  (18)
T9_S1                           000000C0-R  3807  (33)
T9_S1_X                         000003A3-R  3940  (33)
T9_S2                           000003AE-R  3985  (34)
T9_S2_X                         00000620-R  4102  (34)
TEST_008                        00000020-R  1428  (8)      1428  (8)
TEST_008_X                      00001998-R  3695  (31)   #-1433  (8)    #-1905  (14)   #-2151  (16)
TEST_009                        00000020-R  3722  (32)     3722  (32)
TEST_009_X                      0000066F-R  4110  (34)   #-3730  (32)   #-3828  (33)
TIMER_SERVICE                   00001AA4-R  1408  (7)      3745  (32)
TU58                           =00000002    85  (2)
T_ACTL_INTR                     00001204-R  737  (4)       991  (6)
T_ADR_REG_ERR                   00000511-R  546  (4)      1921  (14)
T_AO_DATIP_ERR                  000006EC-R  579  (4)      3103  (25)     3131  (25)     3148  (25)
T_AO_DATI_ERR                   000006B3-R  576  (4)      2805  (22)     2834  (22)
T_AO_DATOB_ERR                  00000679-R  573  (4)      3003  (24)     3030  (24)
T_AO_DATO_ERR                   00000640-R  570  (4)      2904  (23)     2931  (23)
T_BUFF_ADR                      00000953-R  617  (4)       901  (6)
T_BYTE_ERR                      00000D72-R  665  (4)   #-1966  (14)   #-2607  (20)   #-3030  (24)
T_CHANNEL_ERR                   000012FF-R  746  (4)      2017  (15)     2027  (15)     2133  (16)
T_CSR_ERR                       00000463-R  534  (4)      1494  (9)      1509  (9)      1526  (9)
T_CSR_NCL_ERR                   0000114D-R  725  (4)       961  (6)
T_DATA_REG_ERR                  00000540-R  549  (4)      1943  (14)     1966  (14)
T_DATIP_ERR                     000005D3-R  561  (4)      2686  (21)     2714  (21)     2731  (21)
T_DATI_ERR                      0000056C-R  552  (4)      2166  (16)     2194  (16)
T_DATOB_ERR                     000005FB-R  564  (4)      2580  (20)     2607  (20)
T_DATO_ERR                      0000061E-R  567  (4)      2475  (19)     2502  (19)
T_D_ERR_HDR                     000008B5-R  609  (4)       774  (6)       821  (6)
T_D_PATH_ERR                    0000058E-R  555  (4)      2376  (18)     2405  (18)
T_EXP_INTR                      00001174-R  728  (4)       977  (6)
T_IHWE                          0000133C-R  749  (4)      1005  (6)
T_INTR_ERR                      000007DF-R  594  (4)      2042  (15)     2054  (15)     2071  (15)
T_INT_SSYN_ERR                  00000F9E-R  700  (4)       868  (6)
T_IVVECT                        00001393-R  755  (4)      1016  (6)
T_LOGIC                         00001360-R  752  (4)      1010  (6)
T_LONG_ERR                      00000DC8-R  671  (4)   #-1509  (9)    #-1526  (9)    #-1579  (10)   #-1776  (13)
                                                       #-1793  (13)   #-1809  (13)
T_M1_UBE_0                      0000096B-R  620  (4)       269  (2)
T_M1_UBE_1                      000009F4-R  625  (4)       270  (2)
T_M1_UBE_2                      00000A62-R  629  (4)       271  (2)
T_M1_UBE_3                      00000AA5-R  633  (4)       272  (2)
T_M2_UBE_0                      00000AE8-R  637  (4)       273  (2)
```

```
T_M2_UBE_1                      00000B4C-R  641    (4)      274    (2)
T_M2_UBE_2                      00000BAD-R  645    (4)      275    (2)
T_M2_UBE_3                      00000C1E-R  649    (4)      276    (2)
T_MAPS_USED                     00000C9F-R  654    (4)      922    (6)
T_MAP_ABUS_ERR                  000004AC-R  540    (4)     1715    (12)
T_MAP_ADR_ERR                   00001236-R  740    (4)      805    (6)
T_MAP_DBUS_ERR                  00000482-R  537    (4)     1579    (10)
T_MAP_ENT_ERR                   000004D9-R  543    (4)     1776    (13)    1793    (13)    1809    (13)
T_MAP_FCN_ERR                   0000075A-R  585    (4)     3233    (26)    3261    (26)    3343    (27)    3371    (27)
T_MAP_INV_ERR                   0000080D-R  597    (4)     3674    (31)    3690    (31)
T_MAP_SEL_ERR                   000005B0-R  558    (4)     2268    (17)    2296    (17)
T_MAP_USED                      00000CC7-R  657    (4)      833    (6)
T_MAX_LAT_ERR                   00000E6D-R  683    (4)      853    (6)
T_MULTI_ERR                     0000083A-R  600    (4)     3899    (33)    3910    (33)    3929    (33)    4051    (34)
                                                        4072    (34)    4091    (34)
T_MXFR_1_DESC                   0000012F-R  269    (2)   #-3899    (33)  #-3910    (33)  #-3929    (33)
T_MXFR_2_DESC                   0000013F-R  273    (2)   #-4051    (34)  #-4072    (34)  #-4091    (34)
T_MXFR_D_ERR                    00000866-R  603    (4)      904    (6)
T_MXFR_ERR_HDR                  00000CE7-R  660    (4)      905    (6)
T_MXFR_TYPE                     00000939-R  614    (4)      896    (6)
T_NO_ACCESS                     000003C3-R  522    (4)     1903    (14)    3827    (33)
T_NO_BEO                        00000397-R  519    (4)     1432    (8)
T_NO_CSR_ERR                    00000888-R  606    (4)      939    (6)
T_NO_GNT_ERR                    00000F50-R  696    (4)      865    (6)
T_NO_INTR                       000011DF-R  734    (4)      985    (6)
T_NO_MAPS                       00000437-R  531    (4)     2150    (16)
T_NO_NXM_ERR                    00001052-R  710    (4)      949    (6)
T_NO_PB_ERR                     000010A7-R  716    (4)      954    (6)
T_NO_RDY_ERR                    00000E0D-R  677    (4)      847    (6)
T_NO_SACK_ERR                   00000EA4-R  686    (4)      856    (6)
T_NO_SSYN_ERR                   00000ECA-R  689    (4)      859    (6)
T_NO_UBE                        00000414-R  528    (4)     3729    (32)
T_NO_WR_ERR                     00001117-R  722    (4)      944    (6)
T_NXM_ERR                       000007B6-R  591    (4)     3605    (30)    3624    (30)
T_PB_ERR                        00000786-R  588    (4)     3513    (29)    3547    (29)
T_PG_BND_WR_ERR                 0000072B-R  582    (4)     3444    (28)    3463    (28)
T_UBE_ADR                       000003E5-R  525    (4)     1030    (6)
T_UBI_NXM_ERR                   0000101E-R  707    (4)      871    (6)
T_UBI_PB_ERR                    00001081-R  713    (4)      874    (6)
T_UBI_RDS_ERR                   00000FEA-R  704    (4)      880    (6)
T_UBI_WR_ERR                    000010DF-R  719    (4)      877    (6)
T_UNXP_INTR                     000011B4-R  731    (4)      981    (6)
T_UNX_SSYN_ERR                  00000DE4-R  674    (4)      957    (6)
T_WNG_A_ERR                     00000F02-R  692    (4)      862    (6)
T_WNG_GNT_ERR                   00000E40-R  680    (4)      850    (6)
T_WORD_ERR                      00000DA0-R  668    (4)   #-1921    (14)  #-1943    (14)  #-2194    (16)  #-2296    (17)
                                                      #-2405    (18)  #-2502    (19)  #-2714    (21)  #-2731    (21)
                                                      #-2834    (22)  #-2931    (23)  #-3131    (25)  #-3148    (25)
                                                      #-3261    (26)  #-3371    (27)    914    (6)

UBE_0_MULT_1                    00000337-R  455    (2)     3847    (33)
UBE_0_MULT_2                    00000367-R  483    (2)     3996    (34)
UBE_1_MULT_1                    00000343-R  462    (2)
UBE_1_MULT_2                    00000373-R  490    (2)
UBE_2_MULT_1                    0000034F-R  469    (2)
UBE_2_MULT_2                    0000037F-R  497    (2)
UBE_3_MULT_1                    0000035B-R  476    (2)
UBE_3_MULT_2                    0000038B-R  504    (2)
```

```
UBE_BR4_INTR                    0000032B-R  448   (2)
UBE_BR5_INTR                    0000031F-R  441   (2)
UBE_BR6_INTR                    00000313-R  434   (2)
UBE_BR7_INTR                    00000307-R  427   (2)     2031  (15)
UBE_DATIP_1                     000002D7-R  399   (2)     2674  (21)    3092  (25)
UBE_DATI_T                      000002CB-R  392   (2)     2156  (16)    2255  (17)    2364  (18)    2794  (22)
                                                          3220  (26)    3330  (27)    3529  (29)    3591  (30)
                                                          3663  (31)
UBE_DATOB_1                     000002E3-R  406   (2)     2568  (20)    2991  (24)
UBE_DATO_T                      000002EF-R  413   (2)     2461  (19)    2891  (23)    3432  (28)
UBE_PB_ERR                      000002FB-R  420   (2)
UBE_SET_UP                      00001850-R  1133  (7)     2048  (15)    2172  (16)    2274  (17)    2383  (18)
                                                          2481  (19)    2586  (20)    2692  (21)    2812  (22)
                                                          2910  (23)    3009  (24)    3109  (25)    3239  (26)
                                                          3349  (27)    3450  (28)    3534  (29)    3611  (30)
                                                          3680  (31)    3860  (33)    4016  (34)
UBI                             =0000000A   85    (2)     1428  (8)     3722  (32)
UNIBUS_SIZER                     00001867-R 1153  (7)     2138  (16)    3734  (32)
VAL_MAPS                         0000190D-R 1215  (7)
V_BR_INTR                       =00000003   123   (2)   #-2050  (15)  #-2066  (15)  #-983   (6)
V_BYTE_OFF                      =00000019   96    (2)   #-1072  (7)
V_CCOVF                         =0000000D   200   (2)
V_CLK_ERR                       =0000001F   111   (2)
V_CLK_INTR                      =00000007   109   (2)
V_DATA_ERR                      =0000000C   135   (2)   #-818   (6)
V_ENABLE                        =00000006   107   (2)
V_ERR                           =0000000F   176   (2)   #-2183  (16)  #-2287  (17)  #-2396  (18)  #-2493  (19)
                                                        #-2598  (20)  #-2705  (21)  #-2825  (22)  #-2922  (23)
                                                        #-3021  (24)  #-3122  (25)  #-3252  (26)  #-3362  (27)
V_EXP_BR_INTR                   =00000001   119   (2)   #-974   (6)
V_EXP_NXM_ERR                   =00000009   131   (2)   #-947   (6)
V_EXP_PB_ERR                    =0000000A   133   (2)   #-952   (6)
V_EXP_WR_ERR                    =00000007   129   (2)   #-942   (6)
V_INTR_ERR                      =00000004   125   (2)   #-2067  (15)
V_INT_SSYN                      =0000000B   197   (2)   #-867   (6)
V_MAP_VAL                       =0000001F   98    (2)
V_MAX_LAT                       =00000006   187   (2)   #-852   (6)
V_NO_CSR                        =00000000   117   (2)   #-938   (6)
V_NO_GNT                        =0000000A   195   (2)   #-864   (6)
V_NO_SACK                       =00000007   189   (2)   #-855   (6)
V_NO_SSYN                       =00000008   191   (2)   #-3457  (28)  #-3541  (29)  #-3618  (30)  #-3686  (31)
                                                        #-858   (6)   #-956   (6)
V_RDY                           =00000007   163   (2)   #-2039  (15)  #-2162  (16)  #-2263  (17)  #-2371  (18)
                                                        #-2470  (19)  #-2575  (20)  #-2681  (21)  #-2800  (22)
                                                        #-2899  (23)  #-2998  (24)  #-3098  (25)  #-3228  (26)
                                                        #-3338  (27)  #-3439  (28)  #-3508  (29)  #-3600  (30)
                                                        #-3669  (31)  #-846   (6)
V_RUN                           =00000000   103   (2)
V_TIMEOUT                       =00000002   121   (2)   #-3875  (33)  #-4034  (34)
V_TRANSFER                      =00000004   105   (2)
V_UBI_NXM_ERR                   =00000010   143   (2)   #-3617  (30)  #-870   (6)   #-948   (6)
V_UBI_PB_ERR                    =0000000F   145   (2)   #-3540  (29)  #-873   (6)   #-953   (6)
V_UBI_RDS_ERR                   =0000001F   141   (2)   #-879   (6)
V_UBI_WR_ERR                    =0000000E   147   (2)   #-3456  (28)  #-876   (6)   #-943   (6)
V_UNX_SSYN_ERR                  =00000006   127   (2)
V_VALID_MAPS                     00000006-R 229   (2)     1168  (7)     1170  (7)     1183  (7)     1218  (7)
                                                          1231  (7)     1256  (7)     2253  (17)    3218  (26)
```

Macros Cross Reference

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_ABTEST	1	2018 (15)	2018 (15) 2028 (15) 2044 (15) 2134 (16) 2168 (16) 2270 (17) 2378 (18) 2477 (19) 2582 (20) 2688 (21) 2807 (22) 2906 (23) 3005 (24) 3105 (25) 3235 (26) 3345 (27) 3446 (28) 3515 (29) 3607 (30) 3676 (31)
$D1_BSUBT	1	1479 (9)	1479 (9) 1561 (10) 1679 (12) 1759 (13) 1887 (14) 2008 (15) 2124 (16) 2240 (17) 2347 (18) 2450 (19) 2554 (20) 2654 (21) 2779 (22) 2881 (23) 2979 (24) 3078 (25) 3205 (26) 3315 (27) 3414 (28) 3499 (29) 3581 (30) 3654 (31) 3807 (33) 3985 (34)
$D1_BTEST	1	1428 (8)	1428 (8) 3722 (32)
$D1_ERR	1	1494 (9)	1494 (9) 1509 (9) 1526 (9) 1579 (10) 1715 (12) 1776 (13) 1793 (13) 1809 (13) 1903 (14) 1921 (14) 1943 (14) 1966 (14) 2017 (15) 2027 (15) 2042 (15) 2054 (15) 2071 (15) 2133 (16) 2166 (16) 2194 (16) 2268 (17) 2296 (17) 2376 (18) 2405 (18) 2475 (19) 2502 (19) 2580 (20) 2607 (20) 2686 (21) 2714 (21) 2731 (21) 2805 (22) 2834 (22) 2904 (23) 2931 (23) 3003 (24) 3030 (24) 3103 (25) 3131 (25) 3148 (25) 3233 (26) 3261 (26) 3343 (27) 3371 (27) 3444 (28) 3463 (28) 3513 (29) 3547 (29) 3605 (30) 3624 (30) 3674 (31) 3690 (31) 3827 (33) 3899 (33) 3910 (33) 3929 (33) 4051 (34) 4072 (34) 4091 (34)
$D1_ERR_S	1	1494 (9)	1494 (9) 1509 (9) 1526 (9) 1579 (10) 1715 (12) 1776 (13) 1793 (13) 1809 (13) 1903 (14) 1921 (14) 1943 (14) 1966 (14) 2017 (15) 2027 (15) 2042 (15) 2054 (15) 2071 (15) 2133 (16) 2166 (16) 2194 (16) 2268 (17) 2296 (17) 2376 (18) 2405 (18) 2475 (19) 2502 (19) 2580 (20) 2607 (20) 2686 (21) 2714 (21) 2731 (21) 2805 (22) 2834 (22) 2904 (23) 2931 (23) 3003 (24) 3030 (24) 3103 (25) 3131 (25) 3148 (25) 3233 (26) 3261 (26) 3343 (27) 3371 (27) 3444 (28) 3463 (28) 3513 (29) 3547 (29) 3605 (30) 3624 (30) 3674 (31) 3690 (31) 3827 (33) 3899 (33) 3910 (33) 3929 (33) 4051 (34) 4072 (34) 4091 (34)
$D1_ESUBT	1	1530 (9)	1530 (9) 1586 (10) 1719 (12) 1815 (13) 1974 (14) 2083 (15) 2203 (16) 2309 (17) 2419 (18) 2517 (19) 2621 (20) 2746 (21) 2848 (22) 2946 (23) 3045 (24) 3163 (25) 3272 (26) 3382 (27) 3465 (28) 3549 (29) 3626 (30) 3693 (31) 3940 (33) 4102 (34)
$D1_ETEST	1	3695 (31)	3695 (31) 4110 (34)
$D1_EXTEST	1	1433 (8)	1433 (8) 1905 (14) 2151 (16) 3730 (32) 3828 (33)
$D1_PRINT_S	1	774 (6)	1005 (6) 1010 (6) 1016 (6) 1030 (6) 1432 (8) 2150 (16) 3729 (32) 774 (6) 782 (6) 805 (6) 821 (6) 827 (6) 833 (6) 847 (6) 850 (6) 853 (6) 856 (6) 859 (6) 862 (6) 865 (6) 868 (6) 871 (6) 874 (6) 877 (6) 880 (6) 896 (6) 898 (6) 901 (6) 904 (6) 905 (6) 914 (6) 922 (6) 939 (6) 944 (6) 949 (6) 954 (6) 957 (6) 961 (6) 977 (6) 981 (6) 985 (6) 991 (6)

```
$D1_SBTTL           2       1417    (8)     1417    (8)     1439    (9)     1534    (10)    1591    (12)    1723    (13)
                                            1819    (14)    1978    (15)    2087    (16)    2207    (17)    2313    (18)
                                            2423    (19)    2521    (20)    2625    (21)    2750    (22)    2852    (23)
                                            2950    (24)    3049    (25)    3167    (26)    3276    (27)    3386    (28)
                                            3469    (29)    3553    (30)    3630    (31)    3704    (32)    3757    (33)
                                            3944    (34)
$D2_BDATA           1       1428    (8)     1428    (8)     3722    (32)
$D2_BTEST           1       1428    (8)     1428    (8)     3722    (32)
$D2_SBTTL           1       1417    (8)     1417    (8)     3704    (32)
$DEF                1       84      (2)     76      (2)     77      (2)     82      (2)     83      (2)     84      (2)
$DEFEND             1       76      (2)     76      (2)     77      (2)     80      (2)     82      (2)     83      (2)
                                            84      (2)
$DEFINI             1       76      (2)     76      (2)     77      (2)     80      (2)     82      (2)     83      (2)
                                            84      (2)
$DS_ABORT           1       2018    (15)    2018    (15)    2028    (15)    2044    (15)    2134    (16)    2168    (16)
                                            2270    (17)    2378    (18)    2477    (19)    2582    (20)    2688    (21)
                                            2807    (22)    2906    (23)    3005    (24)    3105    (25)    3235    (26)
                                            3345    (27)    3446    (28)    3515    (29)    3607    (30)    3676    (31)
$DS_BGNMESSAGE      1       772     (6)     1001    (6)     1026    (6)     772     (6)     794     (6)     816     (6)
                                            894     (6)     936     (6)     972     (6)
$DS_BGNMOD          1       64      (2)     64      (2)
$DS_BGNSUB          1       1479    (9)     1479    (9)     1561    (10)    1679    (12)    1759    (13)    1887    (14)
                                            2008    (15)    2124    (16)    2240    (17)    2347    (18)    2450    (19)
                                            2554    (20)    2654    (21)    2779    (22)    2881    (23)    2979    (24)
                                            3078    (25)    3205    (26)    3315    (27)    3414    (28)    3499    (29)
                                            3581    (30)    3654    (31)    3807    (33)    3985    (34)
$DS_BGNTEST         1       1428    (8)     1428    (8)     3722    (32)
$DS_BNERROR         1       1489    (9)     1489    (9)     1898    (14)    2014    (15)    2024    (15)    2130    (16)
                                            3822    (33)
$DS_BQUICK          1       3935    (33)    3935    (33)    4096    (34)
$DS_BREAK           1       1175    (7)     1175    (7)     3695    (31)    3834    (33)    4000    (34)    4110    (34)
$DS_CANWAIT_S       1       1350    (7)     1350    (7)     1370    (7)
$DS_CHANNEL_S       1       2010    (15)    2010    (15)    2020    (15)    2126    (16)    3697    (31)
$DS_CKLOOP          1       1511    (9)     1511    (9)     1528    (9)     1581    (10)    1778    (13)    1795    (13)
                                            1811    (13)    1923    (14)    1945    (14)    1968    (14)    2197    (16)
                                            2299    (17)    2408    (18)    2505    (19)    2610    (20)    2734    (21)
                                            2837    (22)    2934    (23)    3033    (24)    3151    (25)    3264    (26)
                                            3374    (27)
$DS_CLRVEC_S        1       1189    (7)     1189    (7)     4104    (34)    4105    (34)    4106    (34)    4107    (34)
                                            4108    (34)
$DS_DSADEF          1       77      (2)     77      (2)
$DS_DSBDEF          1       77      (2)
$DS_DSDEF           2       78      (2)     78      (2)
$DS_DSSDEF          1       82      (2)     82      (2)
$DS_ENDDATA         1       1428    (8)     1428    (8)     3722    (32)
$DS_ENDMESSAGE      1       784     (6)     1018    (6)     1032    (6)     784     (6)     807     (6)     836     (6)
                                            926     (6)     963     (6)     993     (6)
$DS_ENDMOD          1       4112    (34)    4112    (34)
$DS_ENDSUB          1       1530    (9)     1530    (9)     1586    (10)    1719    (12)    1815    (13)    1974    (14)
                                            2083    (15)    2203    (16)    2309    (17)    2419    (18)    2517    (19)
                                            2621    (20)    2746    (21)    2848    (22)    2946    (23)    3045    (24)
                                            3163    (25)    3272    (26)    3382    (27)    3465    (28)    3549    (29)
                                            3626    (30)    3693    (31)    3940    (33)    4102    (34)
$DS_ENDTEST         1       3695    (31)    3695    (31)    4110    (34)
$DS_ENVDEF          2       64      (2)     64      (2)
$DS_ERRDEF          1       81      (2)     81      (2)
$DS_ERRHARD_S       1       1492    (9)     1492    (9)     1503    (9)     1520    (9)     1573    (10)    1712    (12)
```

				1770	(13)	1787	(13)	1803	(13)	1900	(14)	1915	(14)
				1937	(14)	1960	(14)	2015	(15)	2025	(15)	2040	(15)
				2052	(15)	2069	(15)	2131	(16)	2165	(16)	2187	(16)
				2266	(17)	2289	(17)	2374	(18)	2398	(18)	2473	(19)
				2495	(19)	2578	(20)	2600	(20)	2684	(21)	2707	(21)
				2724	(21)	2803	(22)	2827	(22)	2902	(23)	2924	(23)
				3001	(24)	3023	(24)	3101	(25)	3124	(25)	3141	(25)
				3231	(26)	3254	(26)	3341	(27)	3364	(27)	3442	(28)
				3461	(28)	3511	(29)	3545	(29)	3603	(30)	3622	(30)
				3672	(31)	3688	(31)	3824	(33)	3895	(33)	3906	(33)
				3925	(33)	4047	(34)	4068	(34)	4087	(34)		
$DS_EXIT	1	1433	(8)	1433	(8)	1905	(14)	2151	(16)	3730	(32)	3828	(33)
$DS_HPODEF	2	83	(2)	83	(2)								
$DS_KA_DEF	3	84	(2)	84	(2)								
$DS_PAGE	1	1415	(7)	1415	(7)	1437	(8)	1532	(9)	1588	(10)	1721	(12)
				1817	(13)	1976	(14)	2085	(15)	2205	(16)	2311	(17)
				2421	(18)	2519	(19)	2623	(20)	2748	(21)	2850	(22)
				2948	(23)	3047	(24)	3165	(25)	3274	(26)	3384	(27)
				3467	(28)	3551	(29)	3628	(30)	3702	(31)	3755	(32)
				3942	(33)								
$DS_PARDEF	1	79	(2)	79	(2)								
$DS_PRINTB_S	1	774	(6)	1005	(6)	1010	(6)	1015	(6)	1028	(6)	2150	(16)
				774	(6)	778	(6)	800	(6)	821	(6)	823	(6)
				847	(6)	850	(6)	853	(6)	856	(6)	859	(6)
				862	(6)	865	(6)	868	(6)	871	(6)	874	(6)
				877	(6)	880	(6)	896	(6)	898	(6)	900	(6)
				903	(6)	939	(6)	944	(6)	949	(6)	954	(6)
				957	(6)	961	(6)	975	(6)	980	(6)	985	(6)
				989	(6)								
$DS_PRINTF_S	1	1432	(8)	1432	(8)	3729	(32)						
$DS_PRINTX_S	1	832	(6)	832	(6)	905	(6)	910	(6)	921	(6)		
$DS_PROBE_S	1	1485	(9)	1485	(9)	1894	(14)	3818	(33)				
$DS_PSLDEF	1	80	(2)	80	(2)								
$DS_SBTTL	1	1417	(8)	1417	(8)	1439	(9)	1534	(10)	1591	(12)	1723	(13)
				1819	(14)	1978	(15)	2087	(16)	2207	(17)	2313	(18)
				2423	(19)	2521	(20)	2625	(21)	2750	(22)	2852	(23)
				2950	(24)	3049	(25)	3167	(26)	3276	(27)	3386	(28)
				3469	(29)	3553	(30)	3630	(31)	3704	(32)	3757	(33)
				3944	(34)								
$DS_SCBDEF	1	76	(2)	76	(2)								
$DS_SECDEF	1	85	(2)	85	(2)								
$DS_SETIPL_S	1	2034	(15)	2034	(15)	2062	(15)	2081	(15)				
$DS_SETVEC_S	1	1162	(7)	1162	(7)	3738	(32)	3739	(32)	3740	(32)	3741	(32)
				3745	(32)	3987	(34)						
$DS_WAITMS_S	1	2049	(15)	2049	(15)	2065	(15)	2173	(16)	2275	(17)	2384	(18)
				2482	(19)	2587	(20)	2693	(21)	2813	(22)	2911	(23)
				3010	(24)	3110	(25)	3240	(26)	3350	(27)	3451	(28)
				3535	(29)	3612	(30)	3681	(31)				
$EQU	1	84	(2)	64	(2)	78	(2)	79	(2)	80	(2)		
$EQULS1	1	80	(2)	64	(2)	78	(2)	80	(2)				
$EQULST	1	64	(2)	64	(2)	78	(2)	80	(2)				
$GBLINI	2	64	(2)	64	(2)	76	(2)	77	(2)	78	(2)	79	(2)
				80	(2)	82	(2)	83	(2)	84	(2)		
$HP_DEFDEF	1	83	(2)										
$KADEF	1	84	(2)										
$OFFDEF	1	81	(2)	81	(2)								
$PSLDEF	1	80	(2)	80	(2)								

```
$PUSHADR        1       1494    (9)     1494    (9)     1509    (9)     1526    (9)     1579    (10)    1715    (12)
                                        1776    (13)    1793    (13)    1809    (13)    1903    (14)    1921    (14)
                                        1943    (14)    1966    (14)    2013    (15)    2017    (15)    2023    (15)
                                        2027    (15)    2042    (15)    2049    (15)    2054    (15)    2065    (15)
                                        2071    (15)    2129    (16)    2133    (16)    2166    (16)    2173    (16)
                                        2194    (16)    2268    (17)    2275    (17)    2296    (17)    2376    (18)
                                        2384    (18)    2405    (18)    2475    (19)    2482    (19)    2502    (19)
                                        2580    (20)    2587    (20)    2607    (20)    2686    (21)    2693    (21)
                                        2714    (21)    2731    (21)    2805    (22)    2813    (22)    2834    (22)
                                        2904    (23)    2911    (23)    2931    (23)    3003    (24)    3010    (24)
                                        3030    (24)    3103    (25)    3110    (25)    3131    (25)    3148    (25)
                                        3233    (26)    3240    (26)    3261    (26)    3343    (27)    3350    (27)
                                        3371    (27)    3444    (28)    3451    (28)    3463    (28)    3513    (29)
                                        3535    (29)    3547    (29)    3605    (30)    3612    (30)    3624    (30)
                                        3674    (31)    3681    (31)    3690    (31)    3700    (31)    3827    (33)
                                        3899    (33)    3910    (33)    3929    (33)    4051    (34)    4072    (34)
                                        4091    (34)
$VIELD          1       64      (2)     64      (2)     77      (2)     79      (2)     80      (2)     83      (2)
                                        84      (2)
$VIELD1         1       84      (2)     64      (2)     77      (2)     79      (2)     80      (2)     83      (2)
                                        84      (2)
ENKAX_SECDEF    1       85      (2)     85      (2)
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	21	00:00:00.05	00:00:00.18
Command processing	28	00:00:00.19	00:00:00.46
Pass 1	2661	00:00:45.81	00:01:11.43
Symbol table sort	3	00:00:01.46	00:00:02.99
Pass 2	1918	00:00:15.11	00:00:22.66
Symbol table output	101	00:00:00.55	00:00:00.88
Psect synopsis output	7	00:00:00.04	00:00:00.08
Cross-reference output	372	00:00:07.15	00:00:09.05
Assembler run totals	5115	00:01:10.36	00:01:48.08

The working set limit was 1000 pages.
511770 bytes (1000 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 759 non-local and 295 local symbols.
4113 source lines were read in Pass 1, producing 68 object records in Pass 2.
130 pages of virtual memory were used to define 61 macros.

Macro library statistics

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	52
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	8
TOTALS (all libraries)	61

891 GETS were required to define 61 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXH

Table of contents

```
                     1          .SBTTL  ENKAXI: MEMORY TEST
                     2          .TITLE  ENKAXI  VAX-11/730 Specific CPU Cluster Exerciser
                     3          .IDENT  /1-3/
                     4
                     5 ;
                     6 ; Copyright (C) 1981
                     7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
                     8 ;
                     9 ; This software is furnished  under a license for use only on a single
                    10 ; computer  system and  may be  copied only with  the inclusion of the
                    11 ; above copyright notice.  This software, or any other copies thereof,
                    12 ; may not be provided or  otherwise made available to any other person
                    13 ; except for use on such system and to one who agrees to these license
                    14 ; terms.  Title to and  ownership of the  software  shall at all times
                    15 ; remain in DEC.
                    16 ;
                    17 ; The information in this software is subject to change without notice
                    18 ; and should  not be construed  as a commitment  by Digital  Equipment
                    19 ; Corporation.
                    20 ;
                    21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
                    22 ; software on equipment which is not supplied by DEC.
                    23 ;
                    24
                    25 ;++
                    26 ; Facility:     VAX Architecture Diagnostic.
                    27 ;
                    28 ; Abstract:     This module does a quick test of main memory for the VAX 11/730
                    29 ;               CPU. First memory is sized and compared against the memory
                    30 ;               size stored in CSR2 by the micro code and then a march test is
                    31 ;               done on all existing memory above that which is allocated to
                    32 ;               the Diagnostic Supervisor or this diagnostic.  Any gaps found
                    33 ;               in memory will be reported. Two parameters will be implemented
                    34 ;               in the P_table: one to enable or disable single-bit error re-
                    35 ;               porting and the other to enter the expected memory size. An en-
                    36 ;               try of zero in the latter invokes self-sizing.
                    37 ;
                    38 ; Environment:  VAX Diagnostic Supervisor.
                    39 ;
                    40 ; Author:       Rich Lewis      7-Jly-81        Version 1.0
                    41 ;
                    42 ; Modified by:  Kevin Martin    9-Nov-82        Version 1.1
                    43 ;       01      Added $Ds_Break in test 1 for APT.
                    44 ;
                    45 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
                    46 ;
                    47 ;               Added quick flag on Test 7(TU58). If quick flag is set
                    48 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
                    49 ;               If event flag 3 is set, will override protection,
                    50 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
                    51 ;               is clear & tape is not scratch, will report an error.
                    52 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
                    53 ;
                    54 ;
                    55 ;               Tai-Chun Pan    03-Aug-83       Version 1.3
                    56 ;
                    57 ;               fixed bugs on TEST 7 and error summary routine call when
```

```
                        0000     58 ;                  running under APT.
                        0000     59 ;
                        0000     60 ;--
```

```
                        62          .SBTTL  DECLARATIONS
                        63          .LIST   MEB
                        64          .NLIST  CND
                        65          .PSECT  ENKAXI, PAGE, NOWRT
                        66          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                        67          $DS_BGNMOD      CEP_REPAIR, IN=10
                            ;;;     Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)'
                        68
                        69 ;
                        70 ; Include files:
                        71 ;
                        72
                        73          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
                        74
                        78
                        79          $DS_SCBDEF
                        80          $DS_DSADEF
                        81          $DS_DSDEF
                        82          $DS_PARDEF
                        83          $DS_PSLDEF
                        84          $DS_ERRDEF
                        85          $DS_DSSDEF
                        86          $DS_HPODEF
                        87          KA_DEF
                        88          ENKAX_SECDEF
                        89
                        90 ;
                        91 ; Own storage:
                        92 ;
                        93
00000000  0000          94 L_SIZE_CSR:     .LONG                   ; size found in CSR 3
00000000  0004          95 L_SAVE_CSR3:    .LONG                   ; location to save contents of CSR3
00000000  0008          96 L_SIZE_ACTL:    .LONG                   ; size found by this program
00000000  000C          97 L_SIZE_PTBL:    .LONG                   ; size in bytes from p table
00000000  0010          98 M_MAP_BIT:      .LONG                   ; memory map bit mask
00000000  0014          99 V_MAP_BIT:      .LONG                   ; memory map bit field
00080000' 0018         100 L_INCR:         .LONG   L_512K          ; memory size increment
00000013  001C         101 L_V_SIZE:       .LONG   19              ; field size of L_INCR
00000000  0020         102 L_MAP_ACTL:     .LONG                   ; actual memory map
00000000  0024         103 L_RTRN_ADR:     .LONG                   ; return address for mach check service
       00 0028         104 B_FLAGS:        .BYTE                   ; flag byte
                0029   105
00000001  0029         106 V_UNXP_FLAG     =1                      ; set after unexpected machine check
00000002  0029         107 M_UNXP_FLAG     =1@1
00000003  0029         108 V_MEM_ERR_FLAG  =3                      ; set to indicate error in memory
00000008  0029         109 M_MEM_ERR_FLAG  =1@3
00000018  0029         110 V_SB_ERR        =24                     ; report single-bit errors flag
01000000  0029         111 M_SB_ERR        =1@24
00000018  0029         112 V_CHIP_SIZE     =24                     ; memory size bit in CSR2
01000000  0029         113 M_CHIP_SIZE     =1@24
02000000  0029         114 M_DSABL_ECC     =1@25                   ; disable ECC bit in Memory CSR 1
                0029   115
00500000  0029         116 A_MEM_LIM       =^X500000               ; limit of physical address space
00F20008  0029         117 A_MEM_CSR3      =^XF20008               ; location of CSR3
00F20004  0029         118 A_MEM_CSR1      =^XF20004               ; location af CSR1
00020000  0029         119 L_128K          =^X20000                ; 128k
00080000  0029         120 L_512K          =^X80000                ; 512k
```

```
                0029   121
                0029   122
```

```
                                        0029   127 .SBTTL  ERROR MESSAGES
                                        0029   128
                                        0029   129 ;-----------------------------------------------------------------------
                                        0029   130 T_NO_KA730:
69 76 65 64 20 6F 4E 20 20 2F 21 00'    0029   131         .ASCIC  "!/  No devices of type KA730 selected.  Exiting test."
20 65 70 79 74 20 66 6F 20 73 65 63     0035
74 63 65 6C 65 73 20 30 33 37 41 4B     0041
67 6E 69 74 69 78 45 20 20 2E 64 65     004D
                  2E 74 73 65 74 20     0059
                                 35     0029
                                        005F   132 ;-----------------------------------------------------------------------
                                        005F   133 T_SIZ_ERR:
20 67 6E 69 74 70 6D 65 74 74 41 00'    005F   134         .ASCIC  "Attempting to SIZE memory."
6F 6D 65 6D 20 45 5A 49 53 20 6F 74     006B
                        2E 79 72 1A     005F
                                        007A   135 ;-----------------------------------------------------------------------
                                        007A   136 T_DPATH_ERR:
20 67 6E 69 74 70 6D 65 74 74 41 00'    007A   137         .ASCIC  "Attempting to test DATA PATH."
41 54 41 44 20 74 73 65 74 20 6F 74     0086
                  2E 48 54 41 50 2C     0092
                                 1D     007A
                                        0098   138 ;-----------------------------------------------------------------------
                                        0098   139 T_MARCH_ERR:
20 67 6E 69 74 70 6D 65 74 74 41 00'    0098   140         .ASCIC  "Attempting a MEMORY MARCH test."
52 41 4D 20 59 52 4F 4D 45 4D 20 61     00A4
              2E 74 73 65 74 20 48 43   00B0
                                 1F     0098
                                        00B8   141 ;-----------------------------------------------------------------------
                                        00B8   142 T_NO_MEM_TST:
20 79 72 6F 6D 65 6D 20 6F 4E 20 00'    00B8   143         .ASCIC  " No memory found above supervisor.  Exiting Test.!/"
20 65 76 6F 62 61 20 64 6E 75 6F 66     00C4
20 2E 72 6F 73 69 76 72 65 70 75 73     00D0
73 65 54 20 67 6E 69 74 69 78 45 20     00DC
                        2F 21 2E 74     00E8
                                 33     00B8
                                        00EC   144 ;-----------------------------------------------------------------------
                                        00EC   145 T_UNXP_MC_ERR:
63 65 70 78 65 6E 75 20 6E 41 20 00'    00EC   146         .ASCIC  " An unexpected machine check occurred with this logout:!/"
20 65 6E 69 68 63 61 6D 20 64 65 74     00F8
72 72 75 63 63 6F 20 6B 63 65 68 63     0104
73 69 68 74 20 68 74 69 77 20 64 65     0110
      2F 21 3A 74 75 6F 67 6F 6C 20     011C
                                 39     00EC
                                        0126   147 ;-----------------------------------------------------------------------
                                        0126   148 T_SIZE_1_ERR:
6E 75 6F 66 20 72 6F 72 72 45 20 00'    0126   149         .ASCIC  " Error found in comparing maps and/or sizes: !/"-
69 72 61 70 6D 6F 63 20 6E 69 20 64     0132
2F 64 6E 61 20 73 70 61 6D 20 67 6E     013E
2F 21 20 3A 73 65 7A 69 73 20 72 6F     014A
65 4D 20 6D 6F 72 66 20 70 61 4D 09     0156
21 09 3A 33 52 53 43 20 79 72 6F 6D     0162
                     2F 21 20 57 58     016E
                                        0173   150                 "       Map from Memory CSR3:   !XW !/"-
3A 70 61 6D 20 6C 61 75 74 63 41 09     0173   151                 "       Actual map:             !XW !/"-
              2F 21 20 57 58 21 09 09   017F
4D 20 6D 6F 72 66 20 65 7A 69 53 09     0187   152                 "       Size from Memory CSR3:  !UL  kbytes!/"-
```

```
09 3A 33 52 53 43 20 79 72 6F 6D 65  0193
21 73 65 74 79 62 6B 20 20 4C 55 21  019F
                                 2F  01AB
70 20 6D 6F 72 66 20 65 7A 69 53 09  01AC   153                 ''      Size from p table:      !UL  kbytes!/''-
20 4C 55 21 09 3A 65 6C 62 61 74 20  01B8
          2F 21 73 65 74 79 62 6B 20  01C4
65 7A 69 73 20 6C 61 75 74 63 41 09  01CD   154                 ''      Actual size:            !UL  kbytes!/''
74 79 62 6B 20 20 4C 55 21 09 09 3A  01D9
                        2F 21 73 65  01E5
                                 C2  0126
                                     01E9   155 ;-------------------------------------------------------------------------
                                     01E9   156 T_SIZE_2_ERR:
6E 75 6F 66 20 72 6F 72 72 45 20 00' 01E9   157         .ASCIC  '' Error found in comparing maps and/or sizes: !/''-
69 72 61 70 6D 6F 63 20 6E 69 20 64  01F5
2F 64 6E 61 20 73 70 61 6D 20 67 6E  0201
2F 21 20 3A 73 65 7A 69 73 20 72 6F  020D
65 4D 20 6D 6F 72 66 20 70 61 4D 09  0219   158                 ''      Map from Memory CSR3:   !XW !/''-
21 09 3A 33 52 53 43 20 79 72 6F 6D  0225
                     2F 21 20 57 58  0231
3A 70 61 6D 20 6C 61 75 74 63 41 09  0236   159                 ''      Actual map:             !XW !/''-
          2F 21 20 57 58 21 09 09  0242
4D 20 6D 6F 72 66 20 65 7A 69 53 09  024A   160                 ''      Size from Memory CSR3:  !UL  kbytes!/''-
09 3A 33 52 53 43 20 79 72 6F 6D 65  0256
21 73 65 74 79 62 6B 20 20 4C 55 21  0262
                                 2F  026E
65 7A 69 73 20 6C 61 75 74 63 41 09  026F   161                 ''      Actual size:            !UL  kbytes!/''
74 79 62 6B 20 20 4C 55 21 09 09 3A  027B
                        2F 21 73 65  0287
                                 A1  01E9
                                     028B   162 ;-------------------------------------------------------------------------
                                     028B   163 T_MEM_ERR:
6E 75 6F 66 20 72 6F 72 72 45 20 00' 028B   164         .ASCIC  '' Error found in data at location:!XL.!/''-
74 61 20 61 74 61 64 20 6E 69 20 64  0297
58 21 3A 6E 6F 69 74 61 63 6F 6C 20  02A3
                        2F 21 2E 4C  02AF
58 21 3A 64 65 74 63 65 70 78 45 20  02B3   165                 '' Expected:!XL  Actual:!XL             XOR:!XL!/''
4C 58 21 3A 6C 61 75 74 63 41 09 4C  02BF
   2F 21 4C 58 21 3A 52 4F 58 09 09  02CB
                                 4A  028B
                                     02D6   166 ;-------------------------------------------------------------------------
                                     02D6   167 T_MEM_MAP:
4D 20 6C 61 75 74 63 41 20 2F 21 00' 02D6   168         .ASCIC  ''!/ Actual MEMORY MAP, each bit set represents !UL kbytes:''-
65 20 2C 50 41 4D 20 59 52 4F 4D 45  02E2
20 74 65 73 20 74 69 62 20 68 63 61  02EE
21 20 73 74 6E 65 73 65 72 70 65 72  02FA
      3A 73 65 74 79 62 6B 20 4C 55  0306
75 74 63 41 20 2F 21 20 4C 58 21 09  0310   169                 ''      !XL !/ Actual size, in kbytes:  !UL !/''
20 6E 69 20 2C 65 7A 69 73 20 6C 61  031C
20 4C 55 21 09 3A 73 65 74 79 62 6B  0328
                              2F 21  0334
                                 5F  02D6
                                     0336   170 ;-------------------------------------------------------------------------
                                     0336   171
```

```
                        0336   173 .SBTTL  DATA TABLES
                        0336   174
                        0336   175 ; Data table for DATA PATH subtest (2)
                        0336   176
00000000                0336   177 TABLE_1:        .LONG   ^X0             ; float one in field in zeroes
00000001                033A   178                 .LONG   ^X1
00000002                033E   179                 .LONG   ^X2
00000004                0342   180                 .LONG   ^X4
00000008                0346   181                 .LONG   ^X8
00000010                034A   182                 .LONG   ^X10
00000020                034E   183                 .LONG   ^X20
00000040                0352   184                 .LONG   ^X40
00000080                0356   185                 .LONG   ^X80
00000100                035A   186                 .LONG   ^X100
00000200                035E   187                 .LONG   ^X200
00000400                0362   188                 .LONG   ^X400
00000800                0366   189                 .LONG   ^X800
00001000                036A   190                 .LONG   ^X1000
00002000                036E   191                 .LONG   ^X2000
00004000                0372   192                 .LONG   ^X4000
00008000                0376   193                 .LONG   ^X8000
00010000                037A   194                 .LONG   ^X10000
00020000                037E   195                 .LONG   ^X20000
00040000                0382   196                 .LONG   ^X40000
00080000                0386   197                 .LONG   ^X80000
00100000                038A   198                 .LONG   ^X100000
00200000                038E   199                 .LONG   ^X200000
00400000                0392   200                 .LONG   ^X400000
00800000                0396   201                 .LONG   ^X800000
01000000                039A   202                 .LONG   ^X1000000
02000000                039E   203                 .LONG   ^X2000000
04000000                03A2   204                 .LONG   ^X4000000
08000000                03A6   205                 .LONG   ^X8000000
10000000                03AA   206                 .LONG   ^X10000000
20000000                03AE   207                 .LONG   ^X20000000
40000000                03B2   208                 .LONG   ^X40000000
80000000                03B6   209                 .LONG   ^X80000000
FFFFFFFF                03BA   210                 .LONG   ^XFFFFFFFF      ; float zero in field of ones
FFFFFFFE                03BE   211                 .LONG   ^XFFFFFFFE
FFFFFFFD                03C2   212                 .LONG   ^XFFFFFFFD
FFFFFFFB                03C6   213                 .LONG   ^XFFFFFFFB
FFFFFFF7                03CA   214                 .LONG   ^XFFFFFFF7
FFFFFFEF                03CE   215                 .LONG   ^XFFFFFFEF
FFFFFFDF                03D2   216                 .LONG   ^XFFFFFFDF
FFFFFFBF                03D6   217                 .LONG   ^XFFFFFFBF
FFFFFF7F                03DA   218                 .LONG   ^XFFFFFF7F
FFFFFEFF                03DE   219                 .LONG   ^XFFFFFEFF
FFFFFDFF                03E2   220                 .LONG   ^XFFFFFDFF
FFFFFBFF                03E6   221                 .LONG   ^XFFFFFBFF
FFFFF7FF                03EA   222                 .LONG   ^XFFFFF7FF
FFFFEFFF                03EE   223                 .LONG   ^XFFFFEFFF
FFFFDFFF                03F2   224                 .LONG   ^XFFFFDFFF
FFFFBFFF                03F6   225                 .LONG   ^XFFFFBFFF
FFFF7FFF                03FA   226                 .LONG   ^XFFFF7FFF
FFFEFFFF                03FE   227                 .LONG   ^XFFFEFFFF
FFFDFFFF                0402   228                 .LONG   ^XFFFDFFFF
FFFBFFFF                0406   229                 .LONG   ^XFFFBFFFF
```

```
                    FFF7FFFF  040A   230                .LONG   ^XFFF7FFFF
                    FFEFFFFF  040E   231                .LONG   ^XFFEFFFFF
                    FFDFFFFF  0412   232                .LONG   ^XFFDFFFFF
                    FFBFFFFF  0416   233                .LONG   ^XFFBFFFFF
                    FF7FFFFF  041A   234                .LONG   ^XFF7FFFFF
                    FEFFFFFF  041E   235                .LONG   ^XFEFFFFFF
                    FDFFFFFF  0422   236                .LONG   ^XFDFFFFFF
                    FBFFFFFF  0426   237                .LONG   ^XFBFFFFFF
                    F7FFFFFF  042A   238                .LONG   ^XF7FFFFFF
                    EFFFFFFF  042E   239                .LONG   ^XEFFFFFFF
                    DFFFFFFF  0432   240                .LONG   ^XDFFFFFFF
                    BFFFFFFF  0436   241                .LONG   ^XBFFFFFFF
                    7FFFFFFF  043A   242                .LONG   ^X7FFFFFFF
                    00000000  043E   243                .LONG   ^X0
                              0442   244
                              0442   245
                              0442   246
                              0442   247
.SBTTL  ERROR REPORTING ROUTINES
                              0442   248
                              0442   249
                              0442   250 ; This entended error reporting routine is called to report data errors found
                              0442   251 ; in memory or an unexpected machine check occurrence.
                              0442   252
                              0442   253 ERROR_MESSAGE:
                              0442   254
                              0442   255         $DS_BGNMESSAGE
                         0000 0442                      .WORD   ^M<>                    ; ENTRY MASK
          1C FBDF CF    01   E0 0444   256 5$:     BBS     #V_UNXP_FLAG,B_FLAGS,10$        ; br = unexpected mach ck
          59   14 AC    66   CD 044A   257         XORL3   (R6),ERR$_P1(AP),R9             ; load XOR
                              044F   258         $DS_PRINTX_S    FORMAT=T_MEM_ERR,-      ; report data error
                              044F   259                         P1=R6,-                 ; location
                              044F   260                         P2=ERR$_P1(AP),-        ; expected value
                              044F   261                         P3=(R6),-               ; actual value
                              044F   262                         P4=R9                   ; XOR
                                 59   DD 044F                      PUSHL   R9
                                 66   DD 0451                      PUSHL   (R6)
                              14 AC   DD 0453                      PUSHL   ERR$_P1(AP)
                                 56   DD 0456                      PUSHL   R6
                           FE2F CF    9F 0458                      PUSHAB  T_MEM_ERR
          000100E8 9F         05    FB 045C                      CALLS   #5$N, @#DS$PRINTX
                            0016    31 0463   263         BRW     20$                     ; done
                                       0466   264 10$:     $DS_PRINTX_S    FORMAT=T_UNXP_MC_ERR    ; report unexp. mach. check
                           FC82 CF    9F 0466                      PUSHAB  T_UNXP_MC_ERR
          000100E8 9F         01    FB 046A                      CALLS   #5$N, @#DS$PRINTX
                                       0471   265         $DS_PRINTSIG_G  ARGPTR=@A_MCHK_BUFFER   ; print logout stack
00010100  9F   00000000'FF    FA 0471                      CALLG   @A_MCHK_BUFFER, @#DS$PRINTSIG
                                       047C   266 20$.
                                       047C   267         $DS_ENDMESSAGE
                                    04 047C                      RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                       047D   268
                                       047D   269
                                       047D   270 ; This extended error reporting routine is called to report any discrepancy
                                       047D   271 ; found between the map in Memory CSR 3 and that generated in the sizing rou-
                                       047D   272 ; tine or between the size entered in the hardware p table and that generated
                                       047D   273 ; in the sizing routine.  If the size was not entered in the p table, it will,
                                       047D   274 ; of course, not be printed.  Sizes are reported in kbytes.
```

```
                                 047D   276 SIZ_ERR:
                                 047D   277
                                 047D   278          $DS_BGNMESSAGE
                          0000   047D                .WORD   ^M<>            ; ENTRY MASK
                                 047F   279 ;
                                 047F   280 ; Convert byte sizes to kbytes.
                                 047F   281 ;
    52   FB78 CF   00000400 8F   C7   047F   282          DIVL3           #^X400,L_SIZE_CSR,R2    ; convert from CSR
    54   FB76 CF   00000400 8F   C7   0489   283          DIVL3           #^X400,L_SIZE_ACTL,R4   ; convert actual
                                 0493   284 ;
                                 0493   285 ; Determine which error message dependent upon whether size entered in P table.
                                 0493   286 ;
                  00000000'EF    D5   0493   287          TSTL            L_KSIZE_PTBL            ; size entered in p table?
                          20     13   0499   288          BEQL            5$                      ; br = size not entered
                                 049B   289 ;
                                 049B   290 ; This message if size entered in P table
                                 049B   291 ;
                                 049B   292          $DS_PRINTB_S    FORMAT=T_SIZE_1_ERR,-
                                 049B   293                          P1=L_SAVE_CSR3,-        ; map from CSR 3
                                 049B   294                          P2=L_MAP_ACTL,-         ; actual map
                                 049B   295                          P3=R2,-                 ; size from CSR
                                 049B   296                          P4=L_KSIZE_PTBL,-       ; size from P table
                                 049B   297                          P5=R4                   ; actual size
                             54  DD     049B                        PUSHL   R4
                  00000000'EF    DD     049D                        PUSHL   L_KSIZE_PTBL
                             52  DD     04A3                        PUSHL   R2
                      FB77 CF    DD     04A5                        PUSHL   L_MAP_ACTL
                      FB57 CF    DD     04A9                        PUSHL   L_SAVE_CSR3
                      FC75 CF    9F     04AD                        PUSHAB  T_SIZE_1_ERR
000100E0 9F             06      FB     04B1                        CALLS   #$$N, a#DS$PRINTB
                        0017    31     04B8   298          BRW             10$
                                       04BB   299 ;
                                       04BB   300 ; This message if size not entered in P table
                                       04BB   301 ;
                                       04BB   302 5$:      $DS_PRINTB_S    FORMAT=T_SIZE_2_ERR,-
                                       04BB   303                          P1=L_SAVE_CSR3,-        ; map from CSR 3
                                       04BB   304                          P2=L_MAP_ACTL,-         ; actual map
                                       04BB   305                          P3=R2,-                 ; size from CSR 3
                                       04BB   306                          P4=R4                   ; actual size
                                   54  DD     04BB                        PUSHL   R4
                                   52  DD     04BD                        PUSHL   R2
                              FB5D CF  DD     04BF                        PUSHL   L_MAP_ACTL
                              FB3D CF  DD     04C3                        PUSHL   L_SAVE_CSR3
                              FD1E CF  9F     04C7                        PUSHAB  T_SIZE_2_ERR
000100E0 9F                     05    FB     04CB                        CALLS   #$$N, a#DS$PRINTB
                                       04D2   307 10$:
                                       04D2   308          $DS_ENDMESSAGE
                                   04  04D2                        RET                     ; RETURN TO DIAGNOSTIC SUPERVISOR
                                       04D3   309
```

```
                                04D3   311 .SBTTL  SUBROUTINES
                                04D3   312
                                04D3   313 ; This subroutine sizes existing memory by attempting to read locations
                                04D3   314 ; on successive 128- or 512 kbyte boundaries, creating a memory map with
                                04D3   315 ; a bit set for each segment present.  The size in bytes is also calculated.
                                04D3   316
                                04D3   317 MEM_SIZE:
                                04D3   318
                FB38 CF    01   D0  04D3   319          MOVL    #1,M_MAP_BIT                    ; initialize map mask
                     FB2C CF    D4  04D8   320          CLRL    L_SIZE_ACTL                     ; initialize actual size
                     FB40 CF    D4  04DC   321          CLRL    L_MAP_ACTL                      ; initialize actual map
                          56    D4  04E0   322          CLRL    R6                              ; initialize index
                                04E2   323
                                04E2   324 1$:      $DS_PROBE_S     UNIT=KA_UNIT_NO,-       ; probe 1st byte in segment
                                04E2   325                          LENGTH=#1,-
                                04E2   326                          ADDRESS=(R6)
                 00000000'EF    DD  04E2                        PUSHL   KA_UNIT_NO
                          01    DD  04E8                        PUSHL   #1
                          66    9F  04EA                        PUSHAB  (R6)
                000101A0 9F    03  FB  04EC                    CALLS   #3, @#DS$PROBE
                                04F3   327          $DS_BERROR      5$                          ; br = NXM probed
                          50    DD  04F3                        PUSHL   R0
                          01    DD  04F5                        PUSHL   #$ER
                 00000000'8F    DD  04F7                        PUSHL   #DS$K_BERROR
                000100A8 9F    03  FB  04FD                    CALLS   #3, @#DS$BRANCH
                       0E 50    E9  0504                    BLBC    R0, 5$
              FB12 CF    FB05 CF    C8  0507   328          BISL2           M_MAP_BIT,L_MAP_ACTL    ; set bit in actual map
              FAF3 CF    FB06 CF    C0  050E   329          ADDL2           L_INCR,L_SIZE_ACTL      ; increment actual size
                                0515   330
        FAF3 CF    FAF6 CF    01   78  0515   331 5$:      ASHL    #1,M_MAP_BIT,M_MAP_BIT          ; shift mask
B9 56    FAF2 CF    00500000 8F   F1  051D   332          ACBL    #A_MEM_LIM,L_INCR,R6,1$         ; incr index, br = not done
                                FF  0527
                                0529   333
                                05  0529   334          RSB                                     ; return
                                052A   335
                                052A   336
                                052A   337 ; This interrupt-service routine sets NXM flag to indicate that a NXM machine
                                052A   338 ; check has occurred or UNXP flag if any unexpected machine check occurred.
                                052A   339 ; The machine check logout stack is also saved in the machine check buffer.
                                052A   340
                                052A   341          .ALIGN  LONG
                                052C   342 MCHK_SERVICE:
                                052C   343
          00000000'8F    00   DA  052C   344          MTPR    #0,#PR$_MCESR                   ; clear pending mc exception
          53    00000000'EF    D0  0533   345          MOVL    A_MCHK_BUFFER,R3                ; load buffer address
                      83    06  D0  053A   346          MOVL    #6,(R3)+                        ; load count for printsig
          83    00660088'EF    DE  053D   347          MOVAL   DS$_MCHK,(R3)+                  ; load type of exception
                      54    6E  D0  0544   348          MOVL    (SP),R4                         ; load argument byte count
           54    54    FE 8F   78  0547   349          ASHL    #-2,R4,R4                       ; convert to longword count
                      54    03  C0  054C   350          ADDL2   #3,R4                           ; include PC,PSL,TYPE in count
                      83    8E  D0  054F   351 1$:      MOVL    (SP)+,(R3)+                     ; pop longword into buffer
                            54  D7  0552   352          DECL    R4                              ; count longword
                            F9  12  0554   353          BNEQ    1$                              ; br = not done
          52    00000000'EF    D0  0556   354          MOVL    A_MCHK_BUFFER,R2                ; load buffer address
              FAC6 CF    02   C8  055D   355 5$:      BISL2   #M_UNXP_FLAG,B_FLAGS            ; set UNXP flag
                   FABE DF    17  0562   356 10$:     JMP     @L_RTRN_ADR
                                0566   357
```

```
                     0566   358          $DS_PAGE
                     0566   359
```

```
                                0566                .SBTTL  TEST 10:  Memory
                             00000000                       .PSECT  TEST_010, PAGE, NOWRT
                                000C
                                000C    362 ; ++
                                000C    363 ; Test description:
                                000C    364 ;
                                000C    365 ;                   This test does a quick verify of main memory for the VAX 11/730
                                000C    366 ;                   CPU. First memory is sized and compared against the memory
                                000C    367 ;                   size stored in CSR2 and the P_table entry. The data path is
                                000C    368 ;                   checked and a march test is performed on all sized memory above
                                000C    369 ;                   the first 128K bytes excluding that which may be allocated to
                                000C    370 ;                   the diagnostic supervisor. Single bit errors are reported if
                                000C    371 ;                   requested in the P_table entry.
                                000C    372 ;
                                000C    373 ;--
                                000C    374
                                000C    375             $DS_BGNTEST     <MEM,DEFAULT,ALL>
                                000C        DATA_010:
                         00000000  000C                .LONG   0               ; TEST ARGUMENT TABLE TERMINATOR
                                0010        TEST_010::
                             0000  0010                .WORD   ^M<>            ; ENTRY MASK
                                0012    376
                 00000000'EF    D5  0012    377             TSTL    KA_PTABLE               ; CPU selected?
                               10   18  0018    378             BGEQ    5$                      ; br = CPU selected
                                001A    379             $DS_PRINTF_S    FORMAT=T_NO_KA730 ; print exit warning
                 00000029'EF    9F  001A                        PUSHAB  T_NO_KA730
             000100F0 9F        01  FB  0020                    CALLS   #$$N, @#DS$PRINTF
                                0027    380             $DS_EXIT        TEST            ; exit test
                             0388   31  0027                    BRW     TEST_010_X      ; EXIT TEST 10
                                002A    381 5$:
                                002A    382             $DS_PAGE
```

```
                                002A                    .SBTTL  SUBTEST 10.1:  Memory Size
                                002A   385 ;+
                                002A   386 ; Subtest description:
                                002A   387 ;
                                002A   388 ;       This subtest sizes memory and then compares memory found
                                002A   389 ;       to the size stored in memory CSR2.
                                002A   390 ;
                                002A   391 ; Subtest steps:
                                002A   392 ;
                                002A   393 ;       SUBTEST Memory Size
                                002A   394 ;       :  Get expected size from ka p table
                                002A   395 ;       :  Read memory CSR3 (memory found at power up by micro code)
                                002A   396 ;       :  Size memory
                                002A   397 ;       :  Compare two sizes
                                002A   398 ;       :  IF size is not the same
                                002A   399 ;       :  :  THEN
                                002A   400 ;       :  :  Report Memory size error
                                002A   401 ;       :  ENDIF
                                002A   402 ;       ENDSUBTEST Memory Size
                                002A   403 ;
                                002A   404 ;-
                                002A   405
                                002A   406
                                002A   407               $DS_BGNSUB
                                002A       T10_S1::
     00010030 9F    00000000'EF    FA  002A                       CALLG   $S$, @#DS$BGNSUB
              00000028'EF          94  0035   408               CLRB    B_FLAGS                         ; init flag byte
     00000004'EF    00F20008 9F    D0  003B   409               MOVL    @#A_MEM_CSR3,L_SAVE_CSR3        ; save CSR3 contents
0000000C'EF    00000000'EF    0A    78  0046   410               ASHL    #10,L_KSIZE_PTBL,L_SIZE_PTBL    ; convert kbytes to bytes
                                0052   411 ;
                                0052   412 ; Calculate expected size from Memory CSR3
                                0052   413 ;
                00000000'EF          D4  0052   414               CLRL    L_SIZE_CSR                      ; init memory size
                        55      00F20008 9F    D0  0058   415               MOVL    @#A_MEM_CSR3,R5                 ; read CSR3
                                  12 55    18    E0  005F   416               BBS     #V_CHIP_SIZE,R5,1$              ; br = 64k chips
     00000018'EF                00020000 8F    D0  0063   417               MOVL    #L_128K,L_INCR                  ; adjust incr for 16k chips
                0000001C'EF          11    D0  006E   418               MOVL    #17,L_V_SIZE                    ; load field size for 16k chips
                        55      FFFF0000 8F    CA  0075   419 1$:           BICL2   #^XFFFF0000,R5                  ; Clear don't care bits
                                         55    D5  007C   420 2$:           TSTL    R5                              ; any bits set?
                                         12    13  007E   421               BEQL    3$                              ; br = no bits set
     00000000'EF                00000018'EF    C0  0080   422               ADDL2   L_INCR,L_SIZE_CSR               ; incr size
                  55      55          FF 8F    78  0088   423               ASHL    #-1,R5,R5                       ; shift map right
                                           EA    11  0090   424               BRB     2$                              ; br to beginning of loop
                                                  0092   425 ;
                                                  0092   426 ; Size memory and print memory map
                                                  0092   427 ;
                                000004D3'EF          16  0092   428 3$:           JSB             MEM_SIZE                        ; go size memory
52   00000008'EF                00000400 8F    C7  0098   429               DIVL3           #^X400,L_SIZE_ACTL,R2           ; convert to kbytes
53   00000018'EF                00000400 8F    C7  00A4   430               DIVL3           #^X400,L_INCR,R3                ; convert to kbytes
                                                  00B0   431               $DS_PRINTF_S    FORMAT=T_MEM_MAP,-              ; print memory map
                                                  00B0   432                               P1=R3,-                         ; memory increment
                                                  00B0   433                               P2=L_MAP_ACTL,-                 ; actual map
                                                  00B0   434                               P3=R2                           ; size in kbytes
                                           52    DD  00B0                       PUSHL   R2
                                00000020'EF          DD  00B2                       PUSHL   L_MAP_ACTL
                                           53    DD  00B8                       PUSHL   R3
                                000002D6'EF          9F  00BA                       PUSHAB  T_MEM_MAP
```

```
                000100F0 9F    04    FB  00C0                        CALLS   #$$N, @#DS$PRINTF
                                         00C7   435 ;
                                         00C7   436 ; Check for size or map errors, report if any
                                         00C7   437 ;
     00000004'EF     00000020'EF    B1  00C7   438          CMPW    L_MAP_ACTL,L_SAVE_CSR3          ; compare maps
                                 15    12  00D2   439          BNEQ    4$                              ; br = no error
                     0000000C'EF    D5  00D4   440          TSTL    L_SIZE_PTBL                     ; size entered in ka p table?
                                 28    13  00DA   441          BEQL    10$                             ; br = size not entered
     00000008'EF     0000000C'EF    D1  00DC   442          CMPL    L_SIZE_PTBL,L_SIZE_ACTL         ; compare sizes
                                 1B    13  00E7   443          BEQL    10$                             ; br = no error
                                         00E9   444
                                         00E9   445 4$:      $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-
                                         00E9   446                          MSGADR=T_SIZ_ERR,-
                                         00E9   447                          PRLINK=SIZ_ERR
                     0000047D'EF    DF  00E9                        PUSHAL  SIZ_ERR
                     0000005F'EF    DF  00EF                        PUSHAL  T_SIZ_ERR
                     00000000'EF    DD  00F5                        PUSHL   KA_UNIT_NO
                                 01    DD  00FB                        PUSHL   #$ER
                                         00FD       ;;; TEST 10, SUBTEST 1, ERROR 1
                000100D0 9F    04    FB  00FD                        CALLS   #$$M, @#DS$ERRHARD
                                         0104   448 10$:
                                         0104   449          $DS_ENDSUB
                                         0104       T10_S1_X:
     00010038 9F     00000000'EF    FA  0104                        CALLG   $S$, @#DS$ENDSUB
                                         010F   450          $DS_PAGE
```

```
                                010F                    .SBTTL  SUBTEST 10.2:  Data Path
                                010F    453 ;+
                                010F    454 ; Subtest description:
                                010F    455 ;
                                010F    456 ;       This subtest checks the data path for stuck at one, stuck at zero
                                010F    457 ;       and shorted data lines. This is accomplished by floating a one and
                                010F    458 ;       a zero through the data. If an error is detected it is reported with
                                010F    459 ;       information describing the nature of the error.
                                010F    460 ;
                                010F    461 ;
                                010F    462 ; Subtest steps:
                                010F    463 ;
                                010F    464 ;       SUBTEST Data Path
                                010F    465 ;       :  IF REPORT SB ERRORS
                                010F    466 ;       :  :  THEN SET Memory CSR1 Disable ECC
                                010F    467 ;       :  ENDIF
                                010F    468 ;       :  Initialize data pattern
                                010F    469 ;       :  REPEAT
                                010F    470 ;       :  :  Write pattern
                                010F    471 ;       :  :  Read pattern
                                010F    472 ;       :  :  IF pattern incorrect
                                010F    473 ;       :  :  :  THEN
                                010F    474 ;       :  :  :  Report error
                                010F    475 ;       :  :  ENDIF
                                010F    476 ;       :  :  Update pattern
                                010F    477 ;       :  : UNTIL all patterns done
                                010F    478 ;       :  ENDREPEAT
                                010F    479 ;       ENDSUBTEST Data Path
                                010F    480 ;
                                010F    481 ;-
                                010F    482
                                010F    483
                                010F    484         $DS_BGNSUB
                                010F        T10_S2::
00010030 9F   0000000C'EF   FA  010F                CALLG   $S$, @#DS$BGNSUB
      0B 00000000'EF    18  E1  011A    485       BBC     #V_SB_ERR,L_KA_FLAGS,0$         ; br = no SB error report
00F20004 9F   02000000    8F  C8  0122    486       BISL2   #M_DSABL_ECC,@#A_MEM_CSR1       ; disable ECC
                                012D    487 0$:     $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=MCHK_SERVICE
                          00    DD  012D                PUSHL   #0
                0000052C'EF     DF  012F                PUSHAL  MCHK_SERVICE
                          04    DD  0135                PUSHL   #SCB$L_MACHCHK
          00010160 9F     03    FB  0137                CALLS   #3, @#DS$SETVEC
00000024'EF   0000017D'8F   D0  013E    488       MOVL    #5$,L_RTRN_ADR                  ; load return address
              00000028'EF   94  0149    489       CLRB    B_FLAGS                         ; init flag byte
              00000000'EF   D5  014F    490       TSTL    A_FREE_MEM                      ; memory found above DS?
                        10    12  0155    491       BNEQ    1$                              ; br = memory above DS
                                0157    492       $DS_PRINTF_S    T_NO_MEM_TST            ; print exit warning
              000000B8'EF   9F  0157                PUSHAB  T_NO_MEM_TST
          000100F0 9F     01    FB  015D                CALLS   #$$N, @#DS$PRINTF
                                0164    493       $DS_EXIT        TEST                    ; exit test
                      024B      31  0164                BRW     TEST_010_X      ; EXIT TEST 10
          56  00000000'EF      D0  0167    494 1$:     MOVL    A_FREE_MEM,R6                   ; load address to check
          57  00000336'EF      DE  016E    495       MOVAL   TABLE_T,R7                      ; load table address
                  66    67    D0  0175    496 3$:     MOVL    (R7),(R6)                       ; load pattern into memory
                  66    67    D1  0178    497       CMPL    (R7),(R6)                       ; check data
                        1D    13  017B    498       BEQL    6$                              ; br = error
                                017D    499
```

```
                                 017D   500 5$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-          ; report error
                                 017D   501                         MSGADR=T_DPATH_ERR,-
                                 017D   502                         PRLINK=ERROR_MESSAGE,-
                                 017D   503                         P1=(R7)                    ; expected data
                        67   DD  017D                         PUSHL   (R7)
              00000442'EF    DF  017F                         PUSHAL  ERROR_MESSAGE
              0000007A'EF    DF  0185                         PUSHAL  T_DPATH_ERR
              00000000'EF    DD  018B                         PUSHL   KA_UNIT_NO
                        01   DD  0191                         PUSHL   #$ER
                                 0193       ;;; TEST 10, SUBTEST 2, ERROR 1
          000100D0 9F   05   FB  0193                         CALLS   #$$M, @#DS$ERRHARD
                                 019A   504
                                 019A   505 6$:     $DS_CKLOOP      3$                         ; scope loop at 3$
        00010040 9F   D8 AF   FA  019A                         CALLG   3$, @#DS$CKLOOP
                                 01A2   506
                     57   04   C0  01A2   507         ADDL2   #4,R7                          ; incr index
                          67   D5  01A5   508         TSTL    (R7)                           ; done ?
                          CC   12  01A7   509         BNEQ    3$                             ; br = not done
                                 01A9   510
                                 01A9   511         $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                        04   DD  01A9                         PUSHL   #SCB$L_MACHCHK
          00010168 9F   01   FB  01AB                         CALLS   #1, @#DS$CLRVEC
                                 01B2   512
                                 01B2   513         $DS_ENDSUB
                                 01B2       T10_S2_X:
    00010038 9F   0000000C'EF    FA  01B2                         CALLG   $S$, @#DS$ENDSUB
                                 01BD   514
                                 01BD   515         $DS_PAGE
```

```
                                01BD                    .SBTTL  SUBTEST 10.3:  Memory March
                                01BD    518 ;+
                                01BD    519 ; Subtest description:
                                01BD    520 ;
                                01BD    521 ;       This subtest does a march thru all of memory above that which is
                                01BD    522 ;       allocated to the supervisor.  A background pattern of all zeros is
                                01BD    523 ;       written into memory.  Starting with location @A_FREE_MEM thru all
                                01BD    524 ;       available memory, the following is done.  First, the background pattern
                                01BD    525 ;       is checked (i.e., checked for stuck or shorted address lines), then a
                                01BD    526 ;       pattern of all ones is written and checked.  A reverse march is then
                                01BD    527 ;       executed, starting at the highest usable memory location, checking that
                                01BD    528 ;       the old pattern of all ones is still present (again, checking for stuck
                                01BD    529 ;       or shorted address lines) and loading a new pattern of all zeroes.  The
                                01BD    530 ;       new pattern is also checked and the address is then decremented.  This
                                01BD    531 ;       process assures also that there are no stuck data bits at any location.
                                01BD    532 ;       If a NXM machine check occurs while accessing expected good memory an
                                01BD    533 ;       error is reported.  This test skips over "holes" in memory, using the
                                01BD    534 ;       map generated in the sizing routine.
                                01BD    535 ;
                                01BD    536 ;
                                01BD    537 ; Subtest steps:
                                01BD    538 ;
                                01BD    539 ;       SUBTEST Memory March
                                01BD    540 ;       :  DO FOR each physical memory location beyond supervisor
                                01BD    541 ;       :  :  Clear locations under test
                                01BD    542 ;       :  ENDDO
                                01BD    543 ;       :  DO FOR each physical memory location beyond supervisor
                                01BD    544 ;       :  :  IF data error
                                01BD    545 ;       :  :  :  THEN REPORT Error
                                01BD    546 ;       :  :  ENDIF
                                01BD    547 ;       :  :  SET all bits at location
                                01BD    548 ;       :  ENDDO
                                01BD    549 ;       :  REVERSE march direction
                                01BD    550 ;       :  DO FOR each physical memory location beyond supervisor
                                01BD    551 ;       :  :  IF data error
                                01BD    552 ;       :  :  :  THEN REPORT error
                                01BD    553 ;       :  :  ENDIF
                                01BD    554 ;       :  :  CLEAR location
                                01BD    555 ;       :  :  IF location not clear
                                01BD    556 ;       :  :  :  THEN REPORT Error
                                01BD    557 ;       :  :  ENDIF
                                01BD    558 ;       :  ENDDO
                                01BD    559 ;       ENDSUBTEST Memory March
                                01BD    560 ;
                                01BD    561 ;-
                                01BD    562         $DS_BGNSUB
                                01BD        T10_S3::
 00010030 9F   00000018'EF   FA 01BD                CALLG   $S$, @#DS$BGNSUB
                                01C8    563
                                01C8    564         $DS_SETVEC_S    VECTOR=#SCB$L_MACHCHK,SRVADR=MCHK_SERVICE
                          00    DD 01C8                PUSHL   #0
               0000052C'EF      DF 01CA                PUSHAL  MCHK_SERVICE
                          04    DD 01D0                PUSHL   #SCB$L_MACHCHK
           00010160 9F    03    FB 01D2                CALLS   #3, @#DS$SETVEC
               00000028'EF      94 01D9    565         CLRB    B_FLAGS                         ; init flag byte
          56   00000000'EF      D0 01DF    566         MOVL    A_FREE_MEM,R6                   ; load starting address
00000014'EF    56   00000018'EF  C7 01E6    567         DIVL3   L_INCR,R6,V_MAP_BIT             ; load map bit pointer
```

```
     00000024'EF   00000231'8F   D0  01F2   568          MOVL    #12$,L_RTRN_ADR                 ; load return address for m.c.
                           54   D4  01FD   569          CLRL    R4                              ; clear R4
                                    01FF   570 ;
                                    01FF   571 ; Clear each location (background pattern)
                                    01FF   572 ;
                           86   D4  01FF   573 0$:      CLRL    (R6)+                           ; clear location
    54   56   0000001C'EF   00   EC  0201   574 5$:      CMPV    #0,L_V_SIZE,R6,R4               ; on segment boundary?
                           F3   12  020A   575          BNEQ    0$                              ; br = not on segment boundary
                                    020C   576 10$:     $DS_BREAK                               ; check for ^C
                00010058 9F   6E   FA  020C                      CALLG   (SP), @#DS$BREAK
                     00000014'EF   D6  0213   577          INCL    V_MAP_BIT                       ; incr map bit pointer
  DA 00000020'EF   00000014'EF   E0  0219   578          BBS     V_MAP_BIT,L_MAP_ACTL,0$         ; br = no hole
                00000008'EF   56   D1  0225   579          CMPL    R6,L_SIZE_ACTL                  ; end of physical memory?
                           DE   19  022C   580          BLSS    10$                             ; br = not end of memory
                         001E   31  022E   581          BRW     15$                             ; br around error report
                                    0231   582
                                    0231   583 12$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-       ; returns here from machine ck
                                    0231   584                         MSGADR=T_MARCH_ERR,-
                                    0231   585                         PRLINK=ERROR_MESSAGE
                     00000442'EF   DF  0231                      PUSHAL  ERROR_MESSAGE
                     00000098'EF   DF  0237                      PUSHAL  T_MARCH_ERR
                     00000000'EF   DD  023D                      PUSHL   KA_UNIT_NO
                           01   DD  0243                      PUSHL   #$ER
                                    0245       ;;; TEST 10, SUBTEST 3, ERROR 1
                00010DD0 9F   04   FB  0245                      CALLS   #$$M, @#DS$ERRHARD
                         FFB0   31  024C   586          BRW     0$                              ; continue clearing memory
                                    024F   587 ;
                                    024F   588 ; Forward march, check old data (0s), load new pattern of all 1s, check new
                                    024F   589 ; pattern
                                    024F   590 ;
                00000000'EF   D0  024F   591 15$:     MOVL    A_FREE_MEM,R6                   ; load beginning address
                FFFFFFFF 8F   D0  0256   592          MOVL    #-1,R7                          ; load data pattern of all 1s
00000014'EF     00000018'EF   C7  025D   593          DIVL3   L_INCR,R6,V_MAP_BIT             ; load map bit pointer
                                    0269   594
     00000024'EF   00000278'8F   D0  0269   595 30$:     MOVL    #35$,L_RTRN_ADR                 ; load return address
                           66   D5  0274   596          TSTL    (R6)                            ; check data
                           1D   13  0276   597          BEQL    37$                             ; br = clear
                                    0278   598
                                    0278   599 35$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-       ; report error
                                    0278   600                         MSGADR=T_MARCH_ERR,-
                                    0278   601                         PRLINK=ERROR_MESSAGE,-
                                    0278   602                         P1=#0                   ; expected data
                           00   DD  0278                      PUSHL   #0
                     00000442'EF   DF  027A                      PUSHAL  ERROR_MESSAGE
                     00000098'EF   DF  0280                      PUSHAL  T_MARCH_ERR
                     00000000'EF   DD  0286                      PUSHL   KA_UNIT_NO
                           02   DD  028C                      PUSHL   #$ER
                                    028E       ;;; TEST 10, SUBTEST 3, ERROR 2
                00010DD0 9F   05   FB  028E                      CALLS   #$$M, @#DS$ERRHARD
                                    0295   603
     00000024'EF   000002A8'8F   D0  0295   604 37$:     MOVL    #38$,L_RTRN_ADR                 ; load return address
                        66   57   D0  02A0   605          MOVL    R7,(R6)                         ; load pattern into location
                        66   57   D1  02A3   606          CMPL    R7,(R6)                         ; check new pattern
                           21   13  02A6   607          BEQL    39$                             ; br = no error
                                    02A8   608
                                    02A8   609 38$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-       ; report error
                                    02A8   610                         MSGADR=T_MARCH_ERR,-
```

```
                                    02A8   611                          PRLINK=ERROR_MESSAGE,-
                                    02A8   612                          P1=#-1                          ; expected data
                FFFFFFFF 8F   DD    02A8                         PUSHL   #-1
                00000442'EF   DF    02AE                         PUSHAL  ERROR_MESSAGE
                00000098'EF   DF    02B4                         PUSHAL  T_MARCH_ERR
                00000000'EF   DD    02BA                         PUSHL   KA_UNIT_NO
                          03  DD    02C0                         PUSHL   #$ER
                                    02C2       ;;; TEST 10, SUBTEST 3, ERROR 3
          000100D0 9F   05    FB    02C2                         CALLS   #$$M, @#DS$ERRHARD
                                    02C9   613
                       56   04    C0  02C9   614 39$:     ADDL2   #4,R6                           ; incr index
    54    56   0000001C'EF   00    EC  02CC   615          CMPV    #0,L_V_SIZE,R6,R4               ; on segment boundary?
                          03   13    02D5   616          BEQL    40$                             ; br = on segment boundary
                       FF8F   31    02D7   617          BRW     30$                             ; br = not on boundary
                                    02DA   618 40$:     $DS_BREAK                               ; check for ^C
          00010058 9F   6E    FA    02DA                         CALLG   (SP), @#DS$BREAK
                00000014'EF   D6    02E1   619          INCL    V_MAP_BIT                       ; incl map bit pointer
  03 00000020'EF   00000014'EF   E1  02E7   620          BBC     V_MAP_BIT,L_MAP_ACTL,45$        ; br = hole
                       FF73   31    02F3   621          BRW     30$                             ; br = no hole
          00000008'EF   56   D1    02F6   622 45$:     CMPL    R6,L_SIZE_ACTL                  ; end of physical memory?
                          DB   19    02FD   623          BLSS    40$                             ; br = not end of memory
                                    02FF   624 ;
                                    02FF   625 ; Reverse march, check old data, load new pattern of 0s, check new pattern
                                    02FF   626 ;
    00000024'EF   00000312'8F   D0  02FF   627 47$:     MOVL    #53$,L_RTRN_ADR                 ; load return address
                       56   04    C2  030A   628          SUBL2   #4,R6                           ; decrement index
                       57   66    D1  030D   629 50$:     CMPL    (R6),R7                         ; check old pattern
                          1D   13    0310   630          BEQL    55$                             ; br = no error
                                    0312   631
                                    0312   632 53$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-       ; report error
                                    0312   633                          MSGADR=T_MARCH_ERR,-
                                    0312   634                          PRLINK=ERROR_MESSAGE,-
                                    0312   635                          P1=R7                           ; expected data
                          57   DD    0312                         PUSHL   R7
                00000442'EF   DF    0314                         PUSHAL  ERROR_MESSAGE
                00000098'EF   DF    031A                         PUSHAL  T_MARCH_ERR
                00000000'EF   DD    0320                         PUSHL   KA_UNIT_NO
                          04  DD    0326                         PUSHL   #$ER
                                    0328       ;;; TEST 10, SUBTEST 3, ERROR 4
          000100D0 9F   05    FB    0328                         CALLS   #$$M, @#DS$ERRHARD
                                    032F   636
  00000024'EF   00000340'8F   D0  032F   637 55$:     MOVL    #59$,L_RTRN_ADR                 ; load return address
                       66   D4    033A   638 57$:     CLRL    (R6)                            ; load new pattern
                       66   D5    033C   639          TSTL    (R6)                            ; check new pattern
                          1D   13    033E   640          BEQL    60$                             ; br = no error
                                    0340   641
                                    0340   642 59$:     $DS_ERRHARD_S   UNIT=KA_UNIT_NO,-       ; report error
                                    0340   643                          MSGADR=T_MARCH_ERR,-
                                    0340   644                          PRLINK=ERROR_MESSAGE,-
                                    0340   645                          P1=#0                           ; expected data
                          00   DD    0340                         PUSHL   #0
                00000442'EF   DF    0342                         PUSHAL  ERROR_MESSAGE
                00000098'EF   DF    0348                         PUSHAL  T_MARCH_ERR
                00000000'EF   DD    034E                         PUSHL   KA_UNIT_NO
                          05  DD    0354                         PUSHL   #$ER
                                    0356       ;;; TEST 10, SUBTEST 3, ERROR 5
          000100D0 9F   05    FB    0356                         CALLS   #$$M, @#DS$ERRHARD
```

```
                                        035D   646
            00000000'EF    56    D1   035D   647 60$:    CMPL    R6,A_FREE_MEM                   ; lower limit reached?
                           2A    13   0364   648         BEQL    100$                            ; br = lower limit reached
    54    56   0000001C'EF   00    EC   0366   649         CMPV    #0,L_V_SIZE,R6,R4               ; on segment boundary?
                           03    13   036F   650         BEQL    65$                             ; br = on boundary
                       FF8B    31   0371   651         BRW     47$                             ; br = not on boundary
                                        0374   652 65$:    $DS_BREAK                           ; Check for ^C [01]
            00010058 9F    6E    FA   0374                       CALLG   (SP), @#DS$BREAK
                 00000014'EF    D7   037B   653         DECL    V_MAP_BIT                       ; decr map bit pointer
  E7 00000020'EF   00000014'EF    E1   0381   654         BBC     V_MAP_BIT,L_MAP_ACTL,65$        ; br = hole
                       FF6F    31   0385   655         BRW     47$                             ; br = no hole
                                        0390   656
                                        0390   657 100$:   $DS_CLRVEC_S    VECTOR=#SCB$L_MACHCHK
                           04    DD   0390                  PUSHL   #SCB$L_MACHCHK
            00010168 9F    01    FB   0392                  CALLS   #1, @#DS$CLRVEC
    00F20004 9F    02000000 8F    CA   0399   658         BICL2   #M_DSABL_ECC,@#A_MEM_CSR1   ; re-enable ECC
                                        03A4   659         $DS_ENDSUB
                                        03A4       T10_S3_X:
    00010038 9F    00000018'EF    FA   03A4                  CALLG   $S$, @#DS$ENDSUB
                                        03AF   660
                                        03AF   661         $DS_ENDTEST
                           50    01   D0   03AF                  MOVL    #1, R0          ; NORMAL EXIT
                                        03B2       TEST_010_X::
            00010058 9F    6E    FA   03B2                  CALLG   (SP), @#DS$BREAK
                                 04    03B9                  RET                     ; RETURN TO TEST SEQUENCER
                                        03BA   662         $DS_ENDMOD
                                 00000000                  .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                            0000000A    0000               .LONG   $TN
                                        0004   663         .END
```

```
$$$                     = 00000018 R      06
$$ARGS                  = 0000000A
$$E                     = 00000001
$$M                     = 00000005
$$N                     = 00000001
$$S                     = FFFFFFFF
$$T1                    = 0000002C
$ENV                    = 00000001   G
$ER                     = FFFFFFFF
$MO                     = 00000001   G
$S$                     = 00000018 R      06
$ST                     = 00000000
$TN                     = 0000000A
ALL                     = 00000008   G
A_FREE_MEM                ********     X    04
A_MCHK_BUFFER             ********     X    02
A_MEM_CSR1              = 00F20004
A_MEM_CSR3              = 00F20008
A_MEM_LIM               = 00500000
BASE_010                  00000000 R      04
BIT...                  = 00000011
B_FLAGS                   00000028 R      02
CEP_FUNCTIONAL          = 00000000
CEP_REPAIR              = 00000001
CPU$V_COMET             = 00000002
CPU$V_HS                = 00000000
CPU$V_STAR              = 00000001
DATA_010                  0000000C R      04
DEFAULT                 = 00000000   G
DS$ABORT                  00010020
DS$ASKADR                 00010090
DS$ASKDATA                00010080
DS$ASKLGCL                00010098
DS$ASKSTR                 000100A0
DS$ASKVLD                 00010088
DS$ATTACH                 000101A8
DS$BGNSUB                 00010030
DS$BRANCH                 000100A8
DS$BREAK                  00010058
DS$CANWAIT                00010070
DS$CHANNEL                00010180
DS$CKLOOP                 00010040
DS$CLRVEC                 00010168
DS$CNTRLC                 00010078
DS$CVTREG                 000100B0
DS$ENDPASS                00010010
DS$ENDSUB                 00010038
DS$ERRDEV                 000100C8
DS$ERRHARD                000100D0
DS$ERRPREP                00010108
DS$ERRSOFT                000100D8
DS$ERRSYS                 000100C0
DS$ESCAPE                 00010050
DS$FREEDBGSYM             000101B8
DS$GETBUF                 00010120
DS$GETMEM                 00010130
DS$GPHARD                 00010018
DS$HELP                   000101B0
DS$INITSCB                00010170
DS$INLOOP                 00010048
DS$K_BERROR               ********     X    02
DS$K_ERROR              = 00000002
DS$K_NORMAL             = 00000001
DS$K_SEVERE             = 00000004
DS$K_SUBSYS             = 00000066
DS$K_WARNING            = 00000000
DS$LOAD                   00010198
DS$LOADPCS                000101C0
DS$MAPDBGBLOCK            00010118
DS$MMOFF                  00010158
DS$MMON                   00010150
DS$MOVPHY                 00010148
DS$MOVVRT                 00010140
DS$PARSE                  000100B8
DS$PRINTB                 000100E0
DS$PRINTF                 000100F0
DS$PRINTS                 000100F8
DS$PRINTSIG               00010100
DS$PRINTX                 000100E8
DS$PROBE                  000101A0
DS$RELBUF                 00010128
DS$RELMEM                 00010138
DS$SETIPL                 00010178
DS$SETMAP                 00010188
DS$SETPRIEXV              00010110
DS$SETVEC                 00010160
DS$SHOCHAN                00010190
DS$SUMMARY                00010028
DS$WAITMS                 00010060
DS$WAITUS                 00010068
DS$_ARITH               = 006600D0
DS$_BADLINK             = 006600F0
DS$_BADTYPE             = 006600E8
DS$_CHME                = 006600A8
DS$_CHMK                = 006600E0
DS$_DEVNAME             = 00660108
DS$_ERROR               = 00660002
DS$_FHWE                = 00660068
DS$_FRAGBUF             = 00660080
DS$_ICBUSY              = 006600C8
DS$_ICERR               = 006600C0
DS$_IHWE                = 00660060
DS$_ILLCHAR             = 00660018
DS$_ILLPAGCNT           = 00660078
DS$_ILLUNIT             = 00660100
DS$_INSFMEM             = 00660050
DS$_IPL2HI              = 006600B8
DS$_IVADDR              = 00660040
DS$_IVVECT              = 00660038
DS$_KRNLSTK             = 00660090
DS$_LOGIC               = 00660070
DS$_MCHK                = 00660088
DS$_MMOFF               = 00660058
DS$_NEEDUNIT            = 006600F8
```

```
DS$_NOPCS                       = 00660110
DS$_NORMAL                      = 00660001
DS$_NOSUPPORT                   = 006600B1
DS$_NOTDON                      = 00660030
DS$_NOTIMP                      = 006600B0
DS$_NULLSTR                     = 00660010
DS$_OVERFLOW                    = 00660008
DS$_POWER                       = 00660098
DS$_PROGERR                     = 00660020
DS$_SEVERE                      = 00660004
DS$_TRANSL                      = 006600A0
DS$_TRUNCATE                    = 00660028
DS$_UNEXPINT                    = 006600D8
DS$_VASFULL                     = 00660048
DS$_WARNING                     = 00660000
DSA$AL_APTMAIL                    0000FE00
DSA$AT_APTTXT                     0000FA00
DSA$GL_APTCOM                     0000FE04
DSA$GL_DEVLEN                     0000FE58
DSA$GL_ERRNO                      0000FE44
DSA$GL_EVENT                      0000FE48
DSA$GL_FLAGS                      0000FE00
DSA$GL_MSGTYP                     0000FE40
DSA$GL_PASSES                     0000FE08
DSA$GL_PASSNO                     0000FE54
DSA$GL_SECTNO                     0000FE10
DSA$GL_SID                        0000FE14
DSA$GL_SUBTNO                     0000FE4C
DSA$GL_TESTNO                     0000FE50
DSA$GL_UNITS                      0000FE0C
DSA$GQ_MSGPTR                     0000FE68
DSA$GT_DEVNAM                     0000FE5C
DSA$M_APT                       = 80000000
DSA$M_BELL                      = 00000008
DSA$M_BINARY                    = 00000002
DSA$M_CRD_AUTOTEST              = 00000800
DSA$M_CRD_MENUTEST              = 00010000
DSA$M_CRD_TRACE                 = 00020000
DSA$M_DEBUG                     = 04000000
DSA$M_HALT                      = 00000001
DSA$M_IE1                       = 00000010
DSA$M_IE2                       = 00000020
DSA$M_IE3                       = 00000040
DSA$M_IES                       = 00000080
DSA$M_LOAD_DEBUGGER             = 40000000
DSA$M_LOOP                      = 00000004
DSA$M_NORPT                     = 08000000
DSA$M_OPER                      = 00001000
DSA$M_PASSO                     = 20000000
DSA$M_PROMPT                    = 00002000
DSA$M_QA                        = 00008000
DSA$M_QUICK                     = 00000100
DSA$M_SEARCH                    = 00004000
DSA$M_TRACE                     = 00000400
DSA$M_USER                      = 10000000
DSA$M_VERIFY                    - 00000200
DSA$V_APT                       - 0000001F
DSA$V_BELL                      = 00000003
DSA$V_BINARY                    = 00000001
DSA$V_CRD_AUTOTEST              = 0000000B
DSA$V_CRD_MENUTEST              = 00000010
DSA$V_CRD_TRACE                 = 00000011
DSA$V_DEBUG                     = 0000001A
DSA$V_HALT                      = 00000000
DSA$V_IE1                       = 00000004
DSA$V_IE2                       = 00000005
DSA$V_IE3                       = 00000006
DSA$V_IES                       = 00000007
DSA$V_LOAD_DEBUGGER             = 0000001E
DSA$V_LOOP                      = 00000002
DSA$V_NORPT                     = 0000001B
DSA$V_OPER                      = 0000000C
DSA$V_PASSO                     = 0000001D
DSA$V_PROMPT                    = 0000000D
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
ERR$_MSGADR                     = 0000000C
ERR$_NARGS                      = 0000000A
ERR$_NUM                        = 00000004
ERR$_P1                         = 00000014
ERR$_P2                         = 00000018
ERR$_P3                         = 0000001C
ERR$_P4                         = 00000020
ERR$_P5                         = 00000024
ERR$_P6                         = 00000028
ERR$_POINTER                    = 00000010
ERR$_UNIT                       = 00000008
ERROR_MESSAGE                     00000442 R      02
HALT                            = 00000004   G
HP$A_DEPENDENT                    00000032
HP$A_DEVICE                       00000018
HP$A_DVA                          0000001C
HP$A_LINK                         00000020
HP$B_DRIVE                        0000000B
HP$B_FLAGS                        0000000A
HP$M_ALLOC                      = 00000001
```

```
HP$M_WASALL                     = 00000002
HP$Q_DEVICE                       00000000
HP$T_DEVICE                       0000000C
HP$T_TYPE                         00000026
HP$V_ALLOC                      = 00000000
HP$V_WASALL                     = 00000001
HP$W_SIZE                         00000008
HP$W_VECTOR                       00000024
IPR                             = 00000006  G
KA$B_ACC_TYPE                     00000042
KA$B_SID_TYPE                     0000003D
KA$K_LEN                          00000043
KA$L_FLAGS                        00000032
KA$L_SID                          0000003A
KA$M_CHAR                       = 00000100
KA$M_CRC                        = 00000010
KA$M_D_FLOAT                    = 00000002
KA$M_EDIT                       = 00000080
KA$M_F_FLOAT                    = 00000001
KA$M_G_FLOAT                    = 00000004
KA$M_H_FLOAT                    = 00000008
KA$M_PACKED                     = 00000020
KA$M_PACK_MUL                   = 00000040
KA$M_TODR                       = 00010000
KA$V_CHAR                       = 00000008
KA$V_CRC                        = 00000004
KA$V_D_FLOAT                    = 00000001
KA$V_EDIT                       = 00000007
KA$V_F_FLOAT                    = 00000000
KA$V_G_FLOAT                    = 00000002
KA$V_H_FLOAT                    = 00000003
KA$V_PACKED                     = 00000005
KA$V_PACK_MUL                   = 00000006
KA$V_TODR                       = 00000010
KA$W_LATENCY                      0000003E
KA$W_PROGRESS                     00000040
KA$W_RESERVED                     00000038
KA$W_WCS                          00000036
KA_PTABLE                         ********    X    04
KA_UNIT_NO                        ********    X    02
LDPCTX                          = 0000000B  G
L_128K                          = 00020000
L_512K                          = 00080000
L_INCR                            00000018  R       02
L_KA_FLAGS                        ********    X    04
L_KSIZE_PTBL                      ********    X    02
L_MAP_ACTL                        00000020  R       02
L_RTRN_ADR                        00000024  R       02
L_SAVE_CSR3                       00000004  R       02
L_SIZE_ACTL                       00000008  R       02
L_SIZE_CSR                        00000000  R       02
L_SIZE_PTBL                       0000000C  R       02
L_V_SIZE                          0000001C  R       02
MANUAL                          = 00000001  G
MCHK                            = 00000005  G
MCHK_SERVICE                      0000052C  R       02
MEM                             - 00000009  G
MEM_SIZE                          000004D3  R       02
MSG_010                           00000002  R       04
M_CHIP_SIZE                     = 01000000
M_DSABL_ECC                     = 02000000
M_MAP_BIT                         00000010  R       02
M_MEM_ERR_FLAG                  = 00000008
M_SB_ERR                        = 01000000
M_UNXP_FLAG                     = 00000002
PAR$M_ATDEF                     = 00000010
PAR$M_ATHI                      = 00000008
PAR$M_ATLO                      = 00000004
PAR$M_NODEF                     = 00000002
PAR$V_ATDEF                     = 00000004
PAR$V_ATHI                      = 00000003
PAR$V_ATLO                      = 00000002
PAR$V_NODEF                     = 00000001
PAR$_BIN                        = 00000002
PAR$_DEC                        = 0000000A
PAR$_HEX                        = 00000010
PAR$_NO                         = 00000000
PAR$_OCT                        = 00000008
PAR$_YES                        = 00000001
POWER                           = 00000003  G
PR$_MCESR                         ********    X    02
PSL$C_EXEC                      = 00000001
PSL$C_KERNEL                    = 00000000
PSL$C_SUPER                     = 00000002
PSL$C_USER                      = 00000003
PSL$K_EXEC                      = 00000001
PSL$K_KERNEL                    = 00000000
PSL$K_SUPER                     = 00000002
PSL$K_USER                      = 00000003
PSL$M_C                         = 00000001
PSL$M_CBIT                      = 00000001
PSL$M_CM                        = 80000000
PSL$M_CURMOD                    = 03000000
PSL$M_DV                        = 00000080
PSL$M_FPD                       = 08000000
PSL$M_FU                        = 00000040
PSL$M_IPL                       = 001F0000
PSL$M_IS                        = 04000000
PSL$M_IV                        = 00000020
PSL$M_N                         = 00000008
PSL$M_NBIT                      = 00000008
PSL$M_PRVMOD                    = 00C00000
PSL$M_SAFBITS                   = 000037FF
PSL$M_TBIT                      = 00000010
PSL$M_TP                        = 40000000
PSL$M_V                         = 00000002
PSL$M_VBIT                      = 00000002
PSL$M_Z                         = 00000004
PSL$M_ZBIT                      = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CBIT                      = 00000000
```

```
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_NBIT                      = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_VBIT                      = 00000001
PSL$V_Z                         = 00000002
PSL$V_ZBIT                      = 00000002
SCB$L_ACCESS                      00000020
SCB$L_ARITH                       00000034
SCB$L_BREAK                       0000002C
SCB$L_CHME                        00000044
SCB$L_CHMK                        00000040
SCB$L_CHMS                        00000048
SCB$L_CHMU                        0000004C
SCB$L_COMPAT                      00000030
SCB$L_KNLSTK                      00000008
SCB$L_MACHCHK                     00000004
SCB$L_OPCCUS                      00000014
SCB$L_OPCDEC                      00000010
SCB$L_POWER                       0000000C
SCB$L_RADRMOD                     0000001C
SCB$L_ROPRAND                     00000018
SCB$L_RXDB                        000000F8
SCB$L_SFTLVL1                     00000084
SCB$L_SFTLVL10                    000000A8
SCB$L_SFTLVL11                    000000AC
SCB$L_SFTLVL12                    000000B0
SCB$L_SFTLVL13                    000000B4
SCB$L_SFTLVL14                    000000B8
SCB$L_SFTLVL15                    000000BC
SCB$L_SFTLVL2                     00000088
SCB$L_SFTLVL3                     0000008C
SCB$L_SFTLVL4                     00000090
SCB$L_SFTLVL5                     00000094
SCB$L_SFTLVL6                     00000098
SCB$L_SFTLVL7                     0000009C
SCB$L_SFTLVL8                     000000A0
SCB$L_SFTLVL9                     000000A4
SCB$L_TBIT                        00000028
SCB$L_TIMER                       000000C0
SCB$L_TRANSL                      00000024
SCB$L_TXDB                        000000FC
SCB$L_ZERO                        00000000
SEP_FUNCTIONAL                  = 00000002
SEP_REPAIR                      = 00000003
SIZ...                          = 00000001
SIZ_ERR                           0000047D R      02
SYS$ALLOC                         00010238
SYS$ASCTIM                        00010248
SYS$ASSIGN                        00010250
SYS$BINTIM                        00010258
SYS$CANCEL                        00010260
SYS$CANTIM                        00010268
SYS$CLOSE                         000105B8
SYS$CLREF                         00010298
SYS$CONNECT                       000105C0
SYS$DALLOC                        000102D8
SYS$DASSGN                        000102E0
SYS$DISCONNECT                    000105D0
SYS$FAO                           00010350
SYS$FAOL                          00010358
SYS$GET                           00010580
SYS$GETCHN                        000104C8
SYS$GETTIM                        00010378
SYS$LKWSET                        000103A0
SYS$NUMTIM                        000103B8
SYS$OPEN                          00010608
SYS$QIO                           000103C8
SYS$QIOW                          00010200
SYS$READ                          00010590
SYS$READEF                        000103D0
SYS$SETAST                        000103F8
SYS$SETEF                         00010400
SYS$SETIMR                        00010420
SYS$SETPRI                        00010428
SYS$SETPRT                        00010430
SYS$SETRWM                        00010438
SYS$SETSWM                        00010448
SYS$ULKPAG                        00010460
SYS$ULWSET                        00010468
SYS$UNWIND                        00010470
SYS$W[illegible]FR                00010478
SYS$WFL[illegible]ND              00010488
SYS$WFLOR                         00010490
T10_S1                            0000002A RG     04
T10_S1_X                          00000104 R      04
T10_S2                            0000010F RG     04
T10_S2_X                          000001B2 R      04
T10_S3                            000001BD RG     04
T10_S3_X                          000003A4 R      04
TABLE_T                           00000336 R      02
TEST_010                          00000010 RG     04
TEST_010_X                        000003B2 RG     04
TU58                            = 00000002  G
T_DPATH_ERR                       0000007A R      02
T_MARCH_ERR                       00000098 R      02
T_MEM_ERR                         0000028B R      02
T_MEM_MAP                         000002D6 R      02
T_NO_KA730                        00000029 R      02
T_NO_MEM_TST                      000000B8 R      02
T_SIZE_1_ERR                      00000126 R      02
T_SIZE_2_ERR                      000001E9 R      02
T_SIZ_ERR                         0000005F R      02
T_UNXP_MC_ERR                     000000EC R      02
```

```
UBI                             = 0000000A  G
V_CHIP_SIZE                     = 00000018
V_MAP_BIT                       = 00000014 R     02
V_MEM_ERR_FLAG                  = 00000003
V_SB_ERR                        = 00000018
V_UNXP_FLAG                     = 00000001
WCS                             = 00000007  G
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXI	00000566 (1382.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
ABS	00010610 (67088.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_010	000003BA (954.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000018 (24.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
ARGLIST	00000024 (36.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
$TSTCNT	00000004 (4.)	07 (7.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION		REFERENCES...							
$$$	=00000018-R	562	(10)	407	(8)	484	(9)	562	(10)		
$$ARGS	=0000000A	84	(2)	84	(2)						
$$E	=00000001	562	(10)	449	(8)	513	(9)	659	(10)		
$$M	=00000005	645	(10)	375	(7)	447	(8)	503	(9)	585	(10)
				602	(10)	612	(10)	635	(10)	645	(10)
$$N	=00000001	645	(10)	262	(5)	#-264	(5)	297	(5)	306	(5)
				#-379	(7)	434	(8)	447	(8)	#-492	(9)
				503	(9)	585	(10)	602	(10)	612	(10)
				635	(10)	645	(10)	88	(2)		
$$S	=FFFFFFFF	661	(10)	361	(7)	375	(7)	384	(8)	407	(8)
				452	(9)	484	(9)	517	(10)	562	(10)
$$T1	=0000002C	84	(2)	84	(2)						
$ENV	=00000001	67	(2)								
$ER	=FFFFFFFF	661	(10)	#-327	(6)	407	(8)	#-447	(8)	484	(9)
				#-503	(9)	562	(10)	#-585	(10)	#-602	(10)
				#-612	(10)	#-635	(10)	#-645	(10)		
$MO	=00000001	662	(10)	662	(10)						
S	=00000018-R	562	(10)	407	(8)	449	(8)	484	(9)	513	(9)
				562	(10)	659	(10)				
$ST	=00000000	661	(10)	375	(7)	384	(8)	407	(8)	447	(8)
				449	(8)	452	(9)	484	(9)	503	(9)
				505	(9)	513	(9)	517	(10)	562	(10)
				585	(10)	602	(10)	612	(10)	635	(10)
				645	(10)	659	(10)	661	(10)		
$TN	=0000000A	662	(10)	361	(7)	375	(7)	380	(7)	384	(8)
				447	(8)	452	(9)	493	(9)	503	(9)
				517	(10)	585	(10)	602	(10)	612	(10)
				635	(10)	645	(10)	661	(10)	662	(10)
ALL	=00000008	88	(2)	375	(7)						
A_FREE_MEM	00000000-XR			#-490	(9)	#-494	(9)	#-566	(10)	#-591	(10)
				#-647	(10)						
A_MCHK_BUFFER	00000000-XR			265	(5)	#-345	(6)	#-354	(6)		
A_MEM_CSR1	=00F20004	118	(2)	#-486	(9)	#-658	(10)				
A_MEM_CSR3	=00F20008	117	(2)	#-409	(8)	#-415	(8)				
A_MEM_LIM	=00500000	116	(2)	#-332	(6)						
BASE_010	00000000-R	361	(7)	375	(7)						
BIT...	=00000011	87	(2)	67	(2)	80	(2)	81	(2)	82	(2)
				83	(2)	86	(2)	87	(2)		
B_FLAGS	00000028-R	104	(2)	256	(5)	#-355	(6)	#-408	(8)	#-489	(9)
				#-565	(10)						
CEP_FUNCTIONAL	=00000000	67	(2)								
CEP_REPAIR	=00000001	67	(2)	67	(2)						
CPU$V_COMET	=00000002	80	(2)								
CPU$V_HS	=00000000	80	(2)								
CPU$V_STAR	=00000001	80	(2)								
DATA_010	0000000C-R	375	(7)	375	(7)						
DEFAULT	=00000000	88	(2)	375	(7)						
DS$ABORT	00010020	85	(2)								
DS$ASKADR	00010090	85	(2)								
DS$ASKDATA	00010080	85	(2)								

```
DS$ASKLGCL                      00010098      85      (2)
DS$ASKSTR                       000100A0      85      (2)
DS$ASKVLD                       00010088      85      (2)
DS$ATTACH                       000101A8      85      (2)
DS$BGNSUB                       00010030      85      (2)     407     (8)     484     (9)     562     (10)
DS$BRANCH                       000100A8      85      (2)     327     (6)
DS$BREAK                        00010058      85      (2)     576     (10)    618     (10)    652     (10)    661     (10)
DS$CANWAIT                      00010070      85      (2)
DS$CHANNEL                      00010180      85      (2)
DS$CKLOOP                       00010040      85      (2)     505     (9)
DS$CLRVEC                       00010168      85      (2)     511     (9)     657     (10)
DS$CNTRLC                       00010078      85      (2)
DS$CVTREG                       000100B0      85      (2)
DS$ENDPASS                      00010010      85      (2)
DS$ENDSUB                       00010038      85      (2)     449     (8)     513     (9)     659     (10)
DS$ERRDEV                       000100C8      85      (2)
DS$ERRHARD                      000100D0      85      (2)     447     (8)     503     (9)     585     (10)    602     (10)
                                                              612     (10)    635     (10)    645     (10)
DS$ERRPREP                      00010108      85      (2)
DS$ERRSOFT                      000100D8      85      (2)
DS$ERRSYS                       000100C0      85      (2)
DS$ESCAPE                       00010050      85      (2)
DS$FREEDBGSYM                   000101B8      85      (2)
DS$GETBUF                       00010120      85      (2)
DS$GETMEM                       00010130      85      (2)
DS$GPHARD                       00010018      85      (2)
DS$HELP                         000101B0      85      (2)
DS$INITSCB                      00010170      85      (2)
DS$INLOOP                       00010048      85      (2)
DS$K_BERROR                     00000000-XR             #-327   (6)
DS$K_ERROR                      =00000002     81      (2)
DS$K_NORMAL                     =00000001     81      (2)
DS$K_SEVERE                     =00000004     81      (2)
DS$K_SUBSYS                     =00000066     81      (2)     81      (2)
DS$K_WARNING                    =00000000     81      (2)
DS$LOAD                         00010198      85      (2)
DS$LOADPCS                      000101C0      85      (2)
DS$MAPDBGBLOCK                  00010118      85      (2)
DS$MMOFF                        00010158      85      (2)
DS$MMON                         00010150      85      (2)
DS$MOVPHY                       00010148      85      (2)
DS$MOVVRT                       00010140      85      (2)
DS$PARSE                        000100B8      85      (2)
DS$PRINTB                       000100E0      85      (2)     297     (5)     306     (5)
DS$PRINTF                       000100F0      85      (2)     379     (7)     434     (8)     492     (9)
DS$PRINTS                       000100F8      85      (2)
DS$PRINTSIG                     00010100      85      (2)     265     (5)
DS$PRINTX                       000100E8      85      (2)     262     (5)     264     (5)
DS$PROBE                        000101A0      85      (2)     326     (6)
DS$RELBUF                       00010128      85      (2)
DS$RELMEM                       00010138      85      (2)
DS$SETIPL                       00010178      85      (2)
DS$SETMAP                       00010188      85      (2)
DS$SETPRIEXV                    00010110      85      (2)
DS$SETVEC                       00010160      85      (2)     487     (9)     564     (10)
DS$SHOCHAN                      00010190      85      (2)
DS$SUMMARY                      00010028      85      (2)
```

DS$WAITMS	00010060	85	(2)		
DS$WAITUS	00010068	85	(2)		
DS$_ARITH	=006600D0	81	(2)		
DS$_BADLINK	=006600F0	81	(2)		
DS$_BADTYPE	=006600E8	81	(2)		
DS$_CHME	=006600A8	81	(2)		
DS$_CHMK	=006600E0	81	(2)		
DS$_DEVNAME	=00660108	81	(2)		
DS$_ERROR	=00660002	81	(2)		
DS$_FHWE	=00660068	81	(2)		
DS$_FRAGBUF	=00660080	81	(2)		
DS$_ICBUSY	=006600C8	81	(2)		
DS$_ICERR	=006600C0	81	(2)		
DS$_IHWE	=00660060	81	(2)		
DS$_ILLCHAR	=00660018	81	(2)		
DS$_ILLPAGCNT	=00660078	81	(2)		
DS$_ILLUNIT	=00660100	81	(2)		
DS$_INSFMEM	=00660050	81	(2)		
DS$_IPL2HI	=006600B8	81	(2)		
DS$_IVADDR	=00660040	81	(2)		
DS$_IVVECT	=00660038	81	(2)		
DS$_KRNLSTK	=00660090	81	(2)		
DS$_LOGIC	=00660070	81	(2)		
DS$_MCHK	=00660088	81	(2)	347	(6)
DS$_MMOFF	=00660058	81	(2)		
DS$_NEEDUNIT	=006600F8	81	(2)		
DS$_NOPCS	=00660110	81	(2)		
DS$_NORMAL	=00660001	81	(2)		
DS$_NOSUPPORT	=006600B1	81	(2)		
DS$_NOTDON	=00660030	81	(2)		
DS$_NOTIMP	=006600B0	81	(2)	81	(2)
DS$_NULLSTR	=00660010	81	(2)		
DS$_OVERFLOW	=00660008	81	(2)		
DS$_POWER	=00660098	81	(2)		
DS$_PROGERR	=00660020	81	(2)		
DS$_SEVERE	=00660004	81	(2)		
DS$_TRANSL	=006600A0	81	(2)		
DS$_TRUNCATE	=00660028	81	(2)		
DS$_UNEXPINT	=006600D8	81	(2)		
DS$_VASFULL	=00660048	81	(2)		
DS$_WARNING	=00660000	81	(2)	81	(2)
DSA$AL_APTMAIL	0000FE00	80	(2)		
DSA$AT_APTTXT	0000FA00	80	(2)		
DSA$GL_APTCOM	0000FE04	80	(2)		
DSA$GL_DEVLEN	0000FE58	80	(2)		
DSA$GL_ERRNO	0000FE44	80	(2)		
DSA$GL_EVENT	0000FE48	80	(2)		
DSA$GL_FLAGS	0000FE00	80	(2)		
DSA$GL_MSGTYP	0000FE40	80	(2)		
DSA$GL_PASSES	0000FE08	80	(2)		
DSA$GL_PASSNO	0000FE54	80	(2)		
DSA$GL_SECTNO	0000FE10	80	(2)		
DSA$GL_SID	0000FE14	80	(2)		
DSA$GL_SUBTNO	0000FE4C	80	(2)		
DSA$GL_TESTNO	0000FE50	80	(2)		
DSA$GL_UNITS	0000FE0C	80	(2)		
DSA$GQ_MSGPTR	0000FE68	80	(2)		

```
DSA$GT_DEVNAM                    0000FE5C     80      (2)
DSA$M_APT                       =80000000     80      (2)
DSA$M_BELL                      =00000008     80      (2)
DSA$M_BINARY                    =00000002     80      (2)
DSA$M_CRD_AUTOTEST              =00000800     80      (2)
DSA$M_CRD_MENUTEST              =00010000     80      (2)
DSA$M_CRD_TRACE                 =00020000     80      (2)
DSA$M_DEBUG                     =04000000     80      (2)
DSA$M_HALT                      =00000001     80      (2)
DSA$M_IE1                       =00000010     80      (2)
DSA$M_IE2                       =00000020     80      (2)
DSA$M_IE3                       =00000040     80      (2)
DSA$M_IES                       =00000080     80      (2)
DSA$M_LOAD_DEBUGGER             =40000000     80      (2)
DSA$M_LOOP                      =00000004     80      (2)
DSA$M_NORPT                     =08000000     80      (2)
DSA$M_OPER                      =00001000     80      (2)
DSA$M_PASSO                     =20000000     80      (2)
DSA$M_PROMPT                    =00002000     80      (2)
DSA$M_QA                        =00008000     80      (2)
DSA$M_QUICK                     =00000100     80      (2)
DSA$M_SEARCH                    =00004000     80      (2)
DSA$M_TRACE                     =00000400     80      (2)
DSA$M_USER                      =10000000     80      (2)
DSA$M_VERIFY                    =00000200     80      (2)
DSA$V_APT                       =0000001F     80      (2)
DSA$V_BELL                      =00000003     80      (2)
DSA$V_BINARY                    =00000001     80      (2)
DSA$V_CRD_AUTOTEST              =0000000B     80      (2)
DSA$V_CRD_MENUTEST              =00000010     80      (2)
DSA$V_CRD_TRACE                 =00000011     80      (2)
DSA$V_DEBUG                     =0000001A     80      (2)
DSA$V_HALT                      =00000000     80      (2)
DSA$V_IE1                       =00000004     80      (2)
DSA$V_IE2                       =00000005     80      (2)
DSA$V_IE3                       =00000006     80      (2)
DSA$V_IES                       =00000007     80      (2)
DSA$V_LOAD_DEBUGGER             =0000001E     80      (2)
DSA$V_LOOP                      =00000002     80      (2)
DSA$V_NORPT                     =0000001B     80      (2)
DSA$V_OPER                      =0000000C     80      (2)
DSA$V_PASSO                     =0000001D     80      (2)
DSA$V_PROMPT                    =0000000D     80      (2)
DSA$V_QA                        =0000000F     80      (2)
DSA$V_QUICK                     =00000008     80      (2)
DSA$V_SEARCH                    =0000000E     80      (2)
DSA$V_TRACE                     =0000000A     80      (2)
DSA$V_USER                      =0000001C     80      (2)
DSA$V_VERIFY                    =00000009     80      (2)
ENV$M_DOMAIN                    =00000002     67      (2)
ENV$M_LEVEL                     =00000001     67      (2)
ENV$M_SUPER                     =000003FC     67      (2)
ENV$S_DOMAIN                    =00000001     67      (2)
ENV$S_LEVEL                     =00000001     67      (2)
ENV$S_SUPER                     =00000008     67      (2)
ENV$V_DOMAIN                    =00000001     67      (2)     67      (2)
ENV$V_LEVEL                     =00000000     67      (2)     67      (2)
```

```
ENV$V_SUPER                     =00000002     67     (2)
ENV$_CPU                        =00000000     67     (2)      67     (2)
ENV$_FUNCTIONAL                 =00000000     67     (2)      67     (2)
ENV$_REPAIR                     =00000001     67     (2)      67     (2)
ENV$_SUPER                      =00000001     67     (2)
ENV$_SYSTEM                     =00000001     67     (2)      67     (2)
ERR$_MSGADR                     =0000000C     84     (2)
ERR$_NARGS                      =0000000A     84     (2)
ERR$_NUM                        =00000004     84     (2)
ERR$_P1                         =00000014     84     (2)   #-257    (5)   #-262    (5)
ERR$_P2                         =00000018     84     (2)
ERR$_P3                         =0000001C     84     (2)
ERR$_P4                         =00000020     84     (2)
ERR$_P5                         =00000024     84     (2)
ERR$_P6                         =00000028     84     (2)
ERR$_POINTER                    =00000010     84     (2)
ERR$_UNIT                       =00000008     84     (2)
ERROR_MESSAGE                    00000442-R   253    (5)    503    (9)    585    (10)   602    (10)   612    (10)
                                                           635    (10)   645    (10)
HALT                            =00000004     88     (2)
HP$A_DEPENDENT                   00000032     86     (2)      87     (2)
HP$A_DEVICE                      00000018     86     (2)
HP$A_DVA                         0000001C     86     (2)
HP$A_LINK                        00000020     86     (2)
HP$B_DRIVE                       0000000B     86     (2)
HP$B_FLAGS                       0000000A     86     (2)
HP$M_ALLOC                      =00000001     86     (2)
HP$M_WASALL                     =00000002     86     (2)
HP$Q_DEVICE                      00000000     86     (2)
HP$T_DEVICE                      0000000C     86     (2)
HP$T_TYPE                        00000026     86     (2)
HP$V_ALLOC                      =00000000     86     (2)
HP$V_WASALL                     =00000001     86     (2)
HP$W_SIZE                        00000008     86     (2)
HP$W_VECTOR                      00000024     86     (2)
IPR                             =00000006     88     (2)
KA$B_ACC_TYPE                    00000042     87     (2)
KA$B_SID_TYPE                    0000003D     87     (2)
KA$K_LEN                         00000043     87     (2)
KA$L_FLAGS                       00000032     87     (2)
KA$L_SID                         0000003A     87     (2)
KA$M_CHAR                       =00000100     87     (2)
KA$M_CRC                        =00000010     87     (2)
KA$M_D_FLOAT                    =00000002     87     (2)
KA$M_EDIT                       =00000080     87     (2)
KA$M_F_FLOAT                    =00000001     87     (2)
KA$M_G_FLOAT                    =00000004     87     (2)
KA$M_H_FLOAT                    =00000008     87     (2)
KA$M_PACKED                     =00000020     87     (2)
KA$M_PACK_MUL                   =00000040     87     (2)
KA$M_TODR                       =00010000     87     (2)
KA$V_CHAR                       =00000008     87     (2)
KA$V_CRC                        =00000004     87     (2)
KA$V_D_FLOAT                    =00000001     87     (2)
KA$V_EDIT                       =00000007     87     (2)
KA$V_F_FLOAT                    =00000000     87     (2)
KA$V_G_FLOAT                    =00000002     87     (2)
```

```
KA$V_H_FLOAT                    =00000003      87      (2)
KA$V_PACKED                     =00000005      87      (2)
KA$V_PACK_MUL                   =00000006      87      (2)
KA$V_TODR                       =00000010      87      (2)
KA$W_LATENCY                     0000003E      87      (2)
KA$W_PROGRESS                    00000040      87      (2)
KA$W_RESERVED                    00000038      87      (2)
KA$W_WCS                         00000036      87      (2)
KA_PTABLE                        00000000-XR                   #-377    (7)
KA_UNIT_NO                       00000000-XR                   #-326    (6)   #-447    (8)   #-503    (9)   #-585    (10)
                                                               #-602    (10)  #-612    (10)  #-635    (10)  #-645    (10)
LDPCTX                          =0000000B      88      (2)
L_128K                          =00020000      119     (2)   #-417    (8)
L_512K                          =00080000      120     (2)     100    (2)
L_INCR                           00000018-R    100     (2)   #-329    (6)   #-332    (6)   #-417    (8)   #-422    (8)
                                                               #-430    (8)   #-567    (10)  #-593    (10)
L_KA_FLAGS                       00000000-XR                     485    (9)
L_KSIZE_PTBL                     00000000-XR                   #-287    (5)   #-297    (5)   #-410    (8)
L_MAP_ACTL                       00000020-R    102     (2)   #-297    (5)   #-306    (5)   #-321    (6)   #-328    (6)
                                                               #-434    (8)   #-438    (8)     578    (10)    620    (10)
                                                                 654    (10)
L_RTRN_ADR                       00000024-R    103     (2)     356    (6)   #-488    (9)   #-568    (10)  #-595    (10)
                                                               #-604    (10)  #-627    (10)  #-637    (10)
L_SAVE_CSR3                      00000004-R    95      (2)   #-297    (5)   #-306    (5)   #-409    (8)   #-438    (8)
L_SIZE_ACTL                      00000008-R    96      (2)   #-283    (5)   #-320    (6)   #-329    (6)   #-429    (8)
                                                               #-442    (8)   #-579    (10)  #-622    (10)
L_SIZE_CSR                       00000000-R    94      (2)   #-282    (5)   #-414    (8)   #-422    (8)
L_SIZE_PTBL                      0000000C-R    97      (2)   #-410    (8)   #-440    (8)   #-442    (8)
L_V_SIZE                         0000001C-R    101     (2)   #-418    (8)   #-574    (10)  #-615    (10)  #-649    (10)
MANUAL                          =00000001      88      (2)
MCHK                            =00000005      88      (2)
MCHK_SERVICE                     0000052C-R    342     (6)     487    (9)     564    (10)
MEM                             =00000009      88      (2)     375    (7)
MEM_SIZE                         000004D3-R    317     (6)     428    (8)
MSG_010                          00000002-R    361     (7)     375    (7)
M_CHIP_SIZE                     =01000000      113     (2)
M_DSABL_ECC                     =02000000      114     (2)   #-486    (9)   #-658    (10)
M_MAP_BIT                        00000010-R    98      (2)   #-319    (6)   #-328    (6)   #-331    (6)
M_MEM_ERR_FLAG                  =00000008      109     (2)
M_SB_ERR                        =01000000      111     (2)
M_UNXP_FLAG                     =00000002      107     (2)   #-355    (6)
PAR$M_ATDEF                     =00000010      82      (2)
PAR$M_ATHI                      =00000008      82      (2)
PAR$M_ATLO                      =00000004      82      (2)
PAR$M_NODEF                     =00000002      82      (2)
PAR$V_ATDEF                     =00000004      82      (2)
PAR$V_ATHI                      =00000003      82      (2)
PAR$V_ATLO                      =00000002      82      (2)
PAR$V_NODEF                     =00000001      82      (2)
PAR$_BIN                        =00000002      82      (2)
PAR$_DEC                        =0000000A      82      (2)
PAR$_HEX                        =00000010      82      (2)
PAR$_NO                         =00000000      82      (2)
PAR$_OCT                        =00000008      82      (2)
PAR$_YES                        =00000001      82      (2)
POWER                           =00000003      88      (2)
PR$_MCESR                        00000000-XR                   #-344    (6)
```

```
PSL$C_EXEC                      =00000001      83      (2)
PSL$C_KERNEL                    =00000000      83      (2)
PSL$C_SUPER                     =00000002      83      (2)
PSL$C_USER                      =00000003      83      (2)
PSL$K_EXEC                      =00000001      83      (2)
PSL$K_KERNEL                    =00000000      83      (2)
PSL$K_SUPER                     =00000002      83      (2)
PSL$K_USER                      =00000003      83      (2)
PSL$M_C                         =00000001      83      (2)
PSL$M_CBIT                      =00000001      83      (2)
PSL$M_CM                        =80000000      83      (2)       83      (2)
PSL$M_CURMOD                    =03000000      83      (2)
PSL$M_DV                        =00000080      83      (2)
PSL$M_FPD                       =08000000      83      (2)       83      (2)
PSL$M_FU                        =00000040      83      (2)
PSL$M_IPL                       =001F0000      83      (2)
PSL$M_IS                        =04000000      83      (2)
PSL$M_IV                        =00000020      83      (2)
PSL$M_N                         =00000008      83      (2)
PSL$M_NBIT                      =00000008      83      (2)
PSL$M_PRVMOD                    =00C00000      83      (2)
PSL$M_SAFBITS                   =000037FF      83      (2)
PSL$M_TBIT                      =00000010      83      (2)
PSL$M_TP                        =40000000      83      (2)       83      (2)
PSL$M_V                         =00000002      83      (2)
PSL$M_VBIT                      =00000002      83      (2)
PSL$M_Z                         =00000004      83      (2)
PSL$M_ZBIT                      =00000004      83      (2)
PSL$S_CURMOD                    =00000002      83      (2)
PSL$S_IPL                       =00000005      83      (2)
PSL$S_PRVMOD                    =00000002      83      (2)
PSL$V_C                         =00000000      83      (2)
PSL$V_CBIT                      =00000000      83      (2)
PSL$V_CM                        =0000001F      83      (2)
PSL$V_CURMOD                    =00000018      83      (2)
PSL$V_DV                        =00000007      83      (2)
PSL$V_FPD                       =00000018      83      (2)
PSL$V_FU                        =00000006      83      (2)
PSL$V_IPL                       =00000010      83      (2)
PSL$V_IS                        =0000001A      83      (2)
PSL$V_IV                        =00000005      83      (2)
PSL$V_N                         =00000003      83      (2)
PSL$V_NBIT                      =00000003      83      (2)
PSL$V_PRVMOD                    =00000016      83      (2)
PSL$V_TBIT                      =00000004      83      (2)
PSL$V_TP                        =0000001E      83      (2)
PSL$V_V                         =00000001      83      (2)
PSL$V_VBIT                      =00000001      83      (2)
PSL$V_Z                         =00000002      83      (2)
PSL$V_ZBIT                      =00000002      83      (2)
SCB$L_ACCESS                     00000020      79      (2)
SCB$L_ARITH                      00000034      79      (2)
SCB$L_BREAK                      0000002C      79      (2)
SCB$L_CHME                       00000044      79      (2)
SCB$L_CHMK                       00000040      79      (2)
SCB$L_CHMS                       00000048      79      (2)
SCB$L_CHMU                       0000004C      79      (2)
```

```
SCB$L_COMPAT                    00000030      79      (2)
SCB$L_KNLSTK                    00000008      79      (2)
SCB$L_MACHCHK                   00000004      79      (2)    #-487    (9)    #-511    (9)    #-564    (10)    #-657    (10)
SCB$L_OPCCUS                    00000014      79      (2)
SCB$L_OPCDEC                    00000010      79      (2)
SCB$L_POWER                     0000000C      79      (2)
SCB$L_RADRMOD                   0000001C      79      (2)
SCB$L_ROPRAND                   00000018      79      (2)
SCB$L_RXDB                      000000F8      79      (2)
SCB$L_SFTLVL1                   00000084      79      (2)
SCB$L_SFTLVL10                  000000A8      79      (2)
SCB$L_SFTLVL11                  000000AC      79      (2)
SCB$L_SFTLVL12                  000000B0      79      (2)
SCB$L_SFTLVL13                  000000B4      79      (2)
SCB$L_SFTLVL14                  000000B8      79      (2)
SCB$L_SFTLVL15                  000000BC      79      (2)
SCB$L_SFTLVL2                   00000088      79      (2)
SCB$L_SFTLVL3                   0000008C      79      (2)
SCB$L_SFTLVL4                   00000090      79      (2)
SCB$L_SFTLVL5                   00000094      79      (2)
SCB$L_SFTLVL6                   00000098      79      (2)
SCB$L_SFTLVL7                   0000009C      79      (2)
SCB$L_SFTLVL8                   000000A0      79      (2)
SCB$L_SFTLVL9                   000000A4      79      (2)
SCB$L_TBIT                      00000028      79      (2)
SCB$L_TIMER                     000000C0      79      (2)
SCB$L_TRANSL                    00000024      79      (2)
SCB$L_TXDB                      000000FC      79      (2)
SCB$L_ZERO                      00000000      79      (2)
SEP_FUNCTIONAL                  =00000002     67      (2)
SEP_REPAIR                      =00000003     67      (2)
SIZ...                          =00000001     87      (2)    67       (2)    80       (2)    82       (2)    83       (2)
                                                              86       (2)    87       (2)
SIZ_ERR                         0000047D-R    276     (5)    447      (8)
SYS$ALLOC                       00010238      85      (2)
SYS$ASCTIM                      00010248      85      (2)
SYS$ASSIGN                      00010250      85      (2)
SYS$BINTIM                      00010258      85      (2)
SYS$CANCEL                      00010260      85      (2)
SYS$CANTIM                      00010268      85      (2)
SYS$CLOSE                       000105B8      85      (2)
SYS$CLREF                       00010298      85      (2)
SYS$CONNECT                     000105C0      85      (2)
SYS$DALLOC                      00J102D8      85      (2)
SYS$DASSGN                      000102E0      85      (2)
SYS$DISCONNECT                  000105D0      85      (2)
SYS$FAO                         00010350      85      (2)
SYS$FAOL                        00010358      85      (2)
SYS$GET                         00010580      85      (2)
SYS$GETCHN                      000104C8      85      (2)
SYS$GETTIM                      00010378      85      (2)
SYS$LKWSET                      000103A0      85      (2)
SYS$NUMTIM                      00010388      85      (2)
SYS$OPEN                        00010608      85      (2)
SYS$QIO                         000103C8      85      (2)
SYS$QIOW                        00010200      85      (2)
SYS$READ                        00010590      85      (2)
```

```
SYS$READEF                      000103D0      85      (2)
SYS$SETAST                      000103F8      85      (2)
SYS$SETEF                       00010400      85      (2)
SYS$SETIMR                      00010420      85      (2)
SYS$SETPRI                      00010428      85      (2)
SYS$SETPRT                      00010430      85      (2)
SYS$SETRWM                      00010438      85      (2)
SYS$SETSWM                      00010448      85      (2)
SYS$ULKPAG                      00010460      85      (2)
SYS$ULWSET                      00010468      85      (2)
SYS$UNWIND                      00010470      85      (2)
SYS$WAITFR                      00010478      85      (2)
SYS$WFLAND                      00010488      85      (2)
SYS$WFLOR                       00010490      85      (2)
T10_S1                          0000002A-R    407     (8)
T10_S1_X                        00000104-R    449     (8)
T10_S2                          0000010F-R    484     (9)
T10_S2_X                        000001B2-R    513     (9)
T10_S3                          000001BD-R    562     (10)
T10_S3_X                        000003A4-R    659     (10)
TABLE_T                         00000336-R    177     (4)       495     (9)
TEST_010                        00000010-R    375     (7)       375     (7)
TEST_010_X                      000003B2-R    661     (10)    #-380     (7)     #-493     (9)
TU58                            =00000002     88      (2)
T_DPATH_ERR                     0000007A-R    136     (3)       503     (9)
T_MARCH_ERR                     00000098-R    139     (3)       585     (10)      602     (10)      612     (10)      635     (10)
                                                                645     (10)
T_MEM_ERR                       0000028B-R    163     (3)       262     (5)
T_MEM_MAP                       000002D6-R    167     (3)       434     (8)
T_NO_KA730                      00000029-R    130     (3)       379     (7)
T_NO_MEM_TST                    000000B8-R    142     (3)       492     (9)
T_SIZE_1_ERR                    00000126-R    148     (3)       297     (5)
T_SIZE_2_ERR                    000001E9-R    156     (3)       306     (5)
T_SIZ_ERR                       0000005F-R    133     (3)       447     (8)
T_UNXP_MC_ERR                   000000EC-R    145     (3)       264     (5)
UBI                             =0000000A     88      (2)
V_CHIP_SIZE                     =00000018     112     (2)     #-416     (8)
V_MAP_BIT                       00000014-R    99      (2)     #-567     (10)      #-577     (10)      #-578     (10)      #-593     (10)
                                                              #-619     (10)      #-620     (10)      #-653     (10)      #-654     (10)
V_MEM_ERR_FLAG                  =00000003     108     (2)
V_SB_ERR                        =00000018     110     (2)     #-485     (9)
V_UNXP_FLAG                     =00000001     106     (2)     #-256     (5)
WCS                             =00000007     88      (2)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_BSUBT	1	407 (8)	407 (8) 484 (9) 562 (10)
$D1_BTEST	1	375 (7)	375 (7)
$D1_ERR	1	447 (8)	447 (8) 503 (9) 585 (10) 602 (10) 612 (10)
			635 (10) 645 (10)
$D1_ERR_S	1	447 (8)	447 (8) 503 (9) 585 (10) 602 (10) 612 (10)
			635 (10) 645 (10)
$D1_ESUBT	1	449 (8)	449 (8) 513 (9) 659 (10)
$D1_ETEST	1	661 (10)	661 (10)
$D1_EXTEST	1	380 (7)	380 (7) 493 (9)
$D1_PRINT_S	1	262 (5)	262 (5) 264 (5) 297 (5) 306 (5) 379 (7)
			434 (8) 492 (9)
$D1_SBTTL	2	361 (7)	361 (7) 384 (8) 452 (9) 517 (10)
$D2_BDATA	1	375 (7)	375 (7)
$D2_BTEST	1	375 (7)	375 (7)
$D2_SBTTL	1	361 (7)	361 (7)
$DEF	1	87 (2)	79 (2) 80 (2) 85 (2) 86 (2) 87 (2)
$DEFEND	1	79 (2)	79 (2) 80 (2) 83 (2) 85 (2) 86 (2)
			87 (2)
$DEFINI	1	79 (2)	79 (2) 80 (2) 83 (2) 85 (2) 86 (2)
			87 (2)
$DS_BERROR	1	327 (6)	327 (6)
$DS_BGNMESSAGE	1	255 (5)	255 (5) 278 (5)
$DS_BGNMOD	1	67 (2)	67 (2)
$DS_BGNSUB	1	407 (8)	407 (8) 484 (9) 562 (10)
$DS_BGNTEST	1	375 (7)	375 (7)
$DS_BREAK	1	576 (10)	576 (10) 618 (10) 652 (10) 661 (10)
$DS_CKLOOP	1	505 (9)	505 (9)
$DS_CLRVEC_S	1	511 (9)	511 (9) 657 (10)
$DS_DSADEF	1	80 (2)	80 (2)
$DS_DSBDEF	1	80 (2)	
$DS_DSDEF	2	81 (2)	81 (2)
$DS_DSSDEF	1	85 (2)	85 (2)
$DS_ENDDATA	1	375 (7)	375 (7)
$DS_ENDMESSAGE	1	267 (5)	267 (5) 308 (5)
$DS_ENDMOD	1	662 (10)	662 (10)
$DS_ENDSUB	1	449 (8)	449 (8) 513 (9) 659 (10)
$DS_ENDTEST	1	661 (10)	661 (10)
$DS_ENVDEF	2	67 (2)	67 (2)
$DS_ERRDEF	1	84 (2)	84 (2)
$DS_ERRHARD_S	1	445 (8)	445 (8) 500 (9) 583 (10) 599 (10) 609 (10)
			632 (10) 642 (10)
$DS_EXIT	1	380 (7)	380 (7) 493 (9)
$DS_HPODEF	2	86 (2)	86 (2)
$DS_PAGE	1	358 (6)	358 (6) 382 (7) 450 (8) 515 (9)
$DS_PARDEF	1	82 (2)	82 (2)
$DS_PRINTB_S	1	292 (5)	292 (5) 302 (5)
$DS_PRINTF_S	1	379 (7)	379 (7) 431 (8) 492 (9)
$DS_PRINTSIG_G	1	265 (5)	265 (5)
$DS_PRINTX_S	1	258 (5)	258 (5) 264 (5)
$DS_PROBE_S	1	324 (6)	324 (6)

```
$DS_PSLDEF          1          83      (2)       83      (2)
$DS_SBTTL           1         361      (7)      361      (7)      384      (8)      452      (9)      517      (10)
$DS_SCBDEF          1          79      (2)       79      (2)
$DS_SECDEF          1          88      (2)       88      (2)
$DS_SETVEC_S        1         487      (9)      487      (9)      564      (10)
$EQU                1          87      (2)       67      (2)       81      (2)       82      (2)       83      (2)
$EQULST1            1          83      (2)       67      (2)       81      (2)       83      (2)
$EQULST             1          67      (2)       67      (2)       81      (2)       83      (2)
$GBLINI             2          67      (2)       67      (2)       79      (2)       80      (2)       81      (2)      82      (2)
                                                 83      (2)       85      (2)       86      (2)       87      (2)
$HP_DEFDEF          1          86      (2)
$KADEF              1          87      (2)
$OFFDEF             1          84      (2)       84      (2)
$PSLDEF             1          83      (2)       83      (2)
$PUSHADR            1         447      (8)      447      (8)      503      (9)      585      (10)     602      (10)     612      (10)
                                                635      (10)     645      (10)
$VIELD              1          67      (2)       67      (2)       80      (2)       82      (2)       83      (2)      86      (2)
                                                 87      (2)
$VIELD1             1          87      (2)       67      (2)       80      (2)       82      (2)       83      (2)      86      (2)
                                                 87      (2)
ENKAX_SECDEF        1          88      (2)       88      (2)
KA_DEF              2          87      (2)       87      (2)
```

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.05	00:00:00.15
Command processing	22	00:00:00.19	00:00:00.51
Pass 1	1098	00:00:14.88	00:00:19.93
Symbol table sort	0	00:00:00.66	00:00:00.66
Pass 2	340	00:00:02.96	00:00:06.26
Symbol table output	60	00:00:00.37	00:00:00.79
Psect synopsis output	5	00:00:00.04	00:00:00.04
Cross-reference output	243	00:00:02.11	00:00:05.21
Assembler run totals	1797	00:00:21.26	00:00:33.56

The working set limit was 900 pages.
115755 bytes (227 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 463 non-local and 42 local symbols.
663 source lines were read in Pass 1, producing 27 object records in Pass 2.
123 pages of virtual memory were used to define 55 macros.

! Macro library statistics !

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	2
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	45
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	8
TOTALS (all libraries)	55

858 GETS were required to define 55 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXI

Table of contents

```
                        1          .SBTTL  ENKAXJ: FAST CONTROL STORE INTERRUPT
                        2          .TITLE  ENKAXJ  VAX-11/730 Specific CPU Cluster Exerciser
                        3          .IDENT  /1-3/
                        4
                        5 ;
                        6 ; Copyright (C) 1980
                        7 ; Digital Equipment Corporation, Maynard, Massachusetts 01754
                        8 ;
                        9 ; This software is furnished  under a license for use only on a single
                       10 ; computer  system and  may be  copied only with  the inclusion of the
                       11 ; above copyright notice.  This software, or any other copies thereof,
                       12 ; may not be provided or  otherwise made available to any other person
                       13 ; except for use on such system and to one who agrees to these license
                       14 ; terms.  Title to and  ownership of the  software  shall at all times
                       15 ; remain in DEC.
                       16 ;
                       17 ; The information in this software is subject to change without notice
                       18 ; and should  not be construed  as a commitment  by Digital  Equipment
                       19 ; Corporation.
                       20 ;
                       21 ; DEC assumes  no  responsibility  for  the use or  reliability of its
                       22 ; software on equipment which is not supplied by DEC.
                       23 ;
                       24
                       25 ;++
                       26 ; Facility:     VAX Architecture Diagnostic.
                       27 ;
                       28 ; Abstract:     It was agreed at the Functional Specification Review that this
                       29 ;               module will not be implemented. The microcode requirements
                       30 ;               of the Fast Interrupt feature make testing this function not
                       31 ;               feasible. Possibly this issue could be addressed by IDC macro
                       32 ;               or micro diagnostics.
                       33 ;
                       34 ;
                       35 ; Environment:  VAX Diagnostic Supervisor.
                       36 ;
                       37 ; Author:       Fred Roemer     22-Sep-80       Version 1X
                       38 ;
                       39 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
                       40 ;
                       41 ;               Added quick flag on Test 7(TU58). If quick flag is set
                       42 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
                       43 ;               If event flag 3 is set, will override protection,
                       44 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
                       45 ;               is clear & tape is not scratch, will report an error.
                       46 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
                       47 ;
                       48 ;
                       49 ;
                       50 ;               Tai-Chun Pan    03-Aug-83       Version 1.3
                       51 ;
                       52 ;               fixed bugs on TEST 7 and error summary routine call when
                       53 ;               running under APT.
                       54 ;
                       55 ;--
```

```
                     0000     57          .SBTTL  DECLARATIONS
                     0000     58          .LIST   MEB
                     0000     59          .NLIST  CND
                 00000000     60          .PSECT  ENKAXJ, PAGE, NOWRT
                     0000     61          .LIBRARY        "SYS$LIBRARY:DIAG"      ; VAX family diagnostic library.
                     0000     62          $DS_BGNMOD      CEP_REPAIR, IN=1
                     0000        ;;;      Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                     0000     63
                     0000     64 ;
                     0000     65 ; Include files:
                     0000     66 ;
                     0000     67
                     0000     68          .LIBRARY        "ENKAX"         ; CPU Cluster Exerciser library.
%MACRO-E-MLBOPNERR, Error opening macro library
                     0000
```

```
$$S                = FFFFFFFF
$ENV               = 00000001  G
$I-R               = 00000001
$MO                = 00000000  G
$ST                = 00000000
$TN                = 00000001
BIT...             = 00000002
CEP_FUNCTIONAL     = 00000000
CEP_REPAIR         = 00000001
ENV$M_DOMAIN       = 00000002
ENV$M_LEVEL        = 00000001
ENV$M_SUPER        = 000003FC
ENV$S_DOMAIN       = 00000001
ENV$S_LEVEL        = 00000001
ENV$S_SUPER        = 00000008
ENV$V_DOMAIN       = 00000001
ENV$V_LEVEL        = 00000000
ENV$V_SUPER        = 00000002
ENV$_CPU           = 00000000
ENV$_FUNCTIONAL    = 00000000
ENV$_REPAIR        = 00000001
ENV$_SUPER         = 00000001
ENV$_SYSTEM        = 00000001
SEP_FUNCTIONAL     = 00000002
SEP_REPAIR         = 00000003
SIZ...             = 00000008
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ENKAXJ	00000000 (0.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Symbol Cross Reference

SYMBOL	VALUE	DEFINITION	REFERENCES...
$$S	=FFFFFFFF	62 (2)	
$ENV	=00000001	62 (2)	
$ER	=00000001	62 (2)	
$MO	=00000000	62 (2)	
$ST	=00000000	62 (2)	
$TN	=00000001	62 (2)	
BIT...	=00000002	62 (2)	62 (2)
CEP_FUNCTIONAL	=00000000	62 (2)	
CEP_REPAIR	=00000001	62 (2)	62 (2)
ENV$M_DOMAIN	=00000002	62 (2)	
ENV$M_LEVEL	=00000001	62 (2)	
ENV$M_SUPER	=000003FC	62 (2)	
ENV$S_DOMAIN	=00000001	62 (2)	
ENV$S_LEVEL	=00000001	62 (2)	
ENV$S_SUPER	=00000008	62 (2)	
ENV$V_DOMAIN	=00000001	62 (2)	62 (2)
ENV$V_LEVEL	=00000000	62 (2)	62 (2)
ENV$V_SUPER	=00000002	62 (2)	
ENV$_CPU	=00000000	62 (2)	62 (2)
ENV$_FUNCTIONAL	=00000000	62 (2)	62 (2)
ENV$_REPAIR	=00000001	62 (2)	62 (2)
ENV$_SUPER	=00000001	62 (2)	
ENV$_SYSTEM	=00000001	62 (2)	62 (2)
SEP_FUNCTIONAL	=00000002	62 (2)	
SEP_REPAIR	=00000003	62 (2)	
SIZ...	=00000008	62 (2)	62 (2)

! Macros Cross Reference !

MACRO	SIZE	DEFINITION		REFERENCES...	
$DEF	1	62	(2)		
$DS_BGNMOD	1	62	(2)	62	(2)
$DS_ENVDEF	2	62	(2)	62	(2)
$EQU	1	62	(2)	62	(2)
$EQULS1	1	62	(2)	62	(2)
$EQULST	1	62	(2)	62	(2)
$GBLINI	2	62	(2)	62	(2)
$VIELD	1	62	(2)	62	(2)
$VIELD1	1	62	(2)	62	(2)

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	23	00:00:00.05	00:00:00.18
Command processing	21	00:00:00.23	00:00:00.48
Pass 1	331	00:00:01.30	00:00:02.06
Symbol table sort	0	00:00:00.01	00:00:00.00
Pass 2	66	00:00:00.27	00:00:00.49
Symbol table output	5	00:00:00.03	00:00:00.03
Psect synopsis output	5	00:00:00.01	00:00:00.01
Cross-reference output	19	00:00:00.15	00:00:00.18
Assembler run totals	474	00:00:02.06	00:00:03.46

The working set limit was 450 pages.
9360 bytes (19 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 26 non-local and 0 local symbols.
68 source lines were read in Pass 1, producing 0 object records in Pass 2.
17 pages of virtual memory were used to define 7 macros.

! Macro library statistics !

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	2
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	3
TOTALS (all libraries)	5

100 GETS were required to define 5 macros.

There were 1 error, 0 warnings and 0 information messages, on lines:
   68 (2)

$ LIS/CRO ENKAXJ

```
;          0001
;          0002  MODULE ENKAXK (ident = '1.3') =
;          0003  BEGIN
;          0004  %TITLE 'ENKAX Privileged Instruction Ldpctx'            ! LDPCTX
;          0005  ! ++
;          0006  !
;          0007  ! FACILITY :    VAX Architecture Diagnostic
;          0008  !
;          0009  ! ABSTRACT :    This module tests the process context switching instruction
;          0010  !               (LDPCTX ,SRM chapter 7).
;          0011  !
;          0012  ! ENVIRONMENT : Stand-alone with VAX Diagnostic Supervisor
;          0013  !
;          0014  ! AUTHOR :      Kevin Martin            9-NOV-1982 Version 1
;          0015  !
;          0016  !               Tai-Chun Pan    01-Apr-83       Version 1.2
;          0017  !
;          0018  !               Added quick flag on Test 7(TU58). If quick flag is set
;          0019  !               will skip subtest 4 through subtest 11.  Added event flag 3.
;          0020  !               If event flag 3 is set, will override protection,
;          0021  !               i.e., go ahead and write on tape.  If run APT & event flag 3
;          0022  !               is clear & tape is not scratch, will report an error.
;          0023  !
;          0024  !               Modified Unibus sizer on Test 8 & 9. make test run faster.
;          0025  !
;          0026  !
;          0027  !
;          0028  !
;          0029  ! --
;          0030
;          0031  ! ++
;          0032  !       Module comments
;          0033  !       This test may be run in Stand-alone mode ONLY, any attempt to
;          0034  !       run in User mode will result in a test abort.
;          0035  !       The ENKAXK and ENKAXL modules must be linked with other modules
;          0036  !       for a successful run.  This test is not dependent on any other
;          0037  !       test in ENKAX
;          0038  ! --
;          0039
;          0040  LIBRARY
;          0041      'SYS$LIBRARY:LIB';
;          0042  LIBRARY
;          0043      'SYS$LIBRARY:DIAG';
;          0044
;          0045  ! ENKAX library is for MACRO and is relatively useless here
;          0046
;          0047  LINKAGE
;          0048      EX_LINK = JSB;
;          0049
;          0050  EXTERNAL ROUTINE
;          0051      EXECUTE :  EX_LINK ADDRESSING_MODE (LONG_RELATIVE);
;          0052
;          0053  EXTERNAL
;          0054      SAVE_RESULTS : ADDRESSING_MODE (LONG_RELATIVE);      ! Address where
;          0055                                                           ! exception handler
;          0056                                                           ! returns to in Execute
;          0057
```

```
;          0058  ! Macro defs for DSA$V_USER
;          0059
;          0060      $DS_DSADEF
;          0061
;        L 0062  $DS_BGNMOD (ENV=CEP_REPAIR, TEST = 11);
; %PRINT:          Bliss-32 Diagnostic Macro Library version 6.10 "DIAG.L32 (45)"
;          0063
;          0064  %ASSIGN ($$$TN, $$$TN + 1)      ! Fake the effect of a BGNTEST in this module
;          0065      COMPILETIME
;          0066          $$$ERR = 1,             ! These are defined by BGNTEST, by defining
;          0067          $$$ST = 0,              ! them in this module we are able to use
;          0068          $$$ERROK = 1;           ! the DS error calls and subtest macros
;          0069
;          0070  LITERAL
;          0071      PTX$K_LENGTH = 96;
;          0072  MACRO
;          0073
;          0074  !   Define the PTX$L_x and MBZ PCB offsets as macro expansions.
;          0075
;          0076      PTX$L_KSP = 0, 0, 32, 0 %,          ! Define stack register offsets
;          0077      PTX$L_ESP = 4, 0, 32, 0 %,
;          0078      PTX$L_SSP = 8, 0, 32, 0 %,
;          0079      PTX$L_USP = 12, 0, 32, 0 %,
;          0080
;          0081      PTX$L_R0 = 16, 0, 32, 0 %,          ! Define general register offsets
;          0082      PTX$L_R1 = 20, 0, 32, 0 %,
;          0083      PTX$L_R2 = 24, 0, 32, 0 %,
;          0084      PTX$L_R3 = 28, 0, 32, 0 %,
;          0085      PTX$L_R4 = 32, 0, 32, 0 %,
;          0086      PTX$L_R5 = 36, 0, 32, 0 %,
;          0087      PTX$L_R6 = 40, 0, 32, 0 %,
;          0088      PTX$L_R7 = 44, 0, 32, 0 %,
;          0089      PTX$L_R8 = 48, 0, 32, 0 %,
;          0090      PTX$L_R9 = 52, 0, 32, 0 %,
;          0091      PTX$L_R10 = 56, 0, 32, 0 %,
;          0092      PTX$L_R11 = 60, 0, 32, 0 %,
;          0093      PTX$L_AP = 64, 0, 32, 0 %,
;          0094      PTX$L_FP = 68, 0, 32, 0 %,
;          0095
;          0096      PTX$L_P0BR = 80, 0, 32, 0 %,        ! Define mem management offsets
;          0097      PTX$V_P0LR = 84, 0, 22, 0 %,
;          0098      PTX$V_MBZ1 = 84, 22, 2, 0 %,        ! MBZ's are must be zero fields in
;          0099      PTX$V_MBZ2 = 84, 27, 5, 0 %,        ! in the PCB
;          0100      PTX$V_ASTLVL = 84, 24, 3, 0 %,      !
;          0101      PTX$L_P1BR = 88, 0, 32, 0 %,
;          0102      PTX$V_P1LR = 92, 0, 22, 0 %,
;          0103      PTX$V_MBZ3 = 92, 22, 9, 0 %,
;          0104      PTX$V_PME = 92, 31, 1, 0 %,
;          0105      PCB = BLOCK [96, BYTE] %;           ! The standard size PCB
;          0106
;          0107
;          0108  ! Declare variables
;          0109
;          0110  BIND
;          0111      NONE = $ASCIC ('None.'),                          ! Values of expected
;          0112      RESV_OPR - $ASCIC ('Reserved operand abort.'),    ! and actual exception
;          0113                                                        . printed out in the
```

```
;          0114                                                    ! Print_err routine
;          0115  !  DS Error messages
;          0116
;          0117      FAIL_LD_M = $ASCIC ('Fail on LDPCTX.') ,
;          0118
;          0119  !  Subtest fail error messages
;          0120
;          0121      SUB_MSG_1 = $ASCIC ('P0 Base Register high byte is not two.!/'),
;          0122      SUB_MSG_2 = $ASCIC ('P0 Base Register low bit is not zero.!/'),
;          0123      SUB_MSG_3 = $ASCIC ('Must Be zero field one, has all ones.!/'),
;          0124      SUB_MSG_4 = $ASCIC ('ASTLVL = 5, reserved to Digital.!/'),
;          0125      SUB_MSG_5 = $ASCIC ('P1 Base Register high byte not two.!/'),
;          0126      SUB_MSG_6 = $ASCIC ('P1 Base Register low bit not zero.!/');
;          0127
;          0128  GLOBAL
;          0129      ACTUAL_EXCEPT,              ! For passing the exception received, if any
;          0130                                  ! back from the handler and EXECUTE modules
;          0131      EXPECTED_EXCEPT,            ! Compared in main routine with ACTUAL_EXCEPT
;          0132
;          0133      CHECK_STACK : BYTE,         ! Boolean, tells EXECUTE if resulting
;          0134                                  ! stacks should be checked for error.
;          0135      MODE : BYTE,                ! Used in Execute, what mode is the
;          0136                                  ! instruction to be executed from
;          0137      PCB_ERR_FLAG : BITVECTOR [26],! There are 26 bits which correspond to
;          0138                                  ! the 26 data fields in a PCB, if there
;          0139                                  ! there is a discrepency between Actual_pcb
;          0140                                  ! and Expect_Pcb the appropriate bit is set.
;          0141      REG_ERR_FLAG : BITVECTOR [26],! Same as the PCB_Err_Flag, but applies to
;          0142                                  ! registers
;          0143      ORIG_PCBB,          ! Save the place of the Pcb the test disrupts
;          0144      REG_DATA : PCB,     ! Error free data used in testing a normal run
;          0145      INIT_PCB : PCB,     ! Used to initialize return PCBs with background.
;          0146      EXPECT_PCB : PCB,   ! What is expected in ACTUAL_PCB after execution.
;          0147      ACTUAL_PCB : PCB,   ! What the Pcb contains after execution
;          0148
;          0149      INIT_REG : PCB,     ! Used to initialize the registers
;          0150      EXPECT_REG : PCB,   ! What the registers should look like after the
;          0151                          ! instruction
;          0152      ACTUAL_REG : PCB;   ! The contents of the registers after the instruction
;          0153
;          0154
;          0155
;          0156  ! End variable declarations
;          0157
;          0158  ! Subroutines
;          0159
;          0160  ! Error Print Out Routine
;          0161
;          0162  $DS_BGNMESSAGE (ROUTINE_NAME = PRINT_ERR)
;          0163  BIND
;          0164      BASIC_ERR_MSG = .P1,        ! Message to be printed in Basic Error Info
;          0165      EXPECTED_EXCEPT = P2,       ! Exception codes passed in
;          0166      ACTUAL_EXCEPT = P3;
;          0167
;          0168  MAP
;          0169      ACTUAL_PCB : VECTOR,        ! Make PCB's and Regs easier to index
;          0170      ACTUAL_REG : VECTOR,
```

```
;          0171      EXPECT_PCB : VECTOR,
;          0172      EXPECT_REG : VECTOR;
;          0173
;          0174      $DS_PRINTB (BASIC_ERR_MSG); ! Print basic error info
;          0175  !
;          0176  !   Print out longwords which do not match EXPECT_PCB
;          0177  !
;          0178      IF .PCB_ERR_FLAG NEQ 0 THEN
;          0179          BEGIN
;          0180  !
;          0181  !       Give PCB base address as Actual_Pcb's address
;          0182  !
;          0183          $DS_PRINTB ($ASCIC ('PCB starting at !XL.!/'),ACTUAL_PCB);
;          0184  !
;          0185  !       Print table header for error info
;          0186  !
;          0187          $DS_PRINTB ($ASCIC ('!_!_INITIAL!4* EXPECTED!3* ACTUAL!5* XOR!/'));
;          0188  !
;          0189  !
;          0190  !       Print all fields where there is a data discrepency, and if in
;          0191  !       extended error info mode, print all other fields as well.
;          0192  !
;          0193          INCR I FROM 0 TO 23 DO
;          0194              BEGIN
;          0195              MAP
;          0196                  ACTUAL_PCB : VECTOR,    ! Make it easier to index
;          0197                  EXPECT_PCB : VECTOR,
;          0198                  INIT_PCB : VECTOR;
;          0199  !
;          0200
;          0201              IF .PCB_ERR_FLAG[.I] EQL 1 THEN
;        P 0202                  $DS_PRINTB ($ASCIC ('PCB + !ZB !_!XL!3* !XL!3* !XL!3* !XL!/'),
;        P 0203                  .I, .INIT_PCB [.I], .EXPECT_PCB [.I],
;          0204                  .ACTUAL_PCB [.I], (.EXPECT_PCB[.I] XOR .ACTUAL_PCB[.I]))
;          0205              ELSE
;          0206  !
;          0207  !           If no error on this line print only when extended info is on
;          0208  !
;        P 0209                  $DS_PRINTX ($ASCIC ('PCB + !ZB !_!XL!3* !XL!3* !XL!/'),
;        P 0210                  .I, .INIT_PCB [.I], .EXPECT_PCB [.I],
;          0211                  .ACTUAL_PCB [.I]);
;          0212              END
;          0213          END
;          0214      ELSE
;          0215          $DS_PRINTB ($ASCIC ('No Pcb data error encountered!/'));
;          0216  !
;          0217  !   Print register errors, if any
;          0218  !
;          0219      IF .REG_ERR_FLAG NEQ 0 THEN
;          0220          BEGIN
;          0221  !
;          0222  !       Tell user where to find copy of resultant registers
;          0223  !
;        P 0224          $DS_PRINTB ($ASCIC ('A copy of register contents  begins at !XL.!/'),
;          0225                  ACTUAL_REG);
;          0226  !
;          0227  .       Print table header for error info
```

```
;          0228  !
;          0229          $DS_PRINTB ($ASCIC ('!_!_INITIAL!4* EXPECTED!3* ACTUAL!5* XOR!/'));
;          0230          INCR I FROM 0 TO 23 DO
;          0231              BEGIN
;          0232              MAP
;          0233                  ACTUAL_REG : VECTOR,    ! Make the block easier to index
;          0234                  EXPECT_REG : VECTOR,
;          0235                  INIT_REG:VECTOR;
;          0236              IF .REG_ERR_FLAG[.I] EQL 1 THEN
;        P 0237                  $DS_PRINTB ($ASCIC ('REG + !ZB !_!XL!3* !XL!3* !XL!3* !XL!/'),
;        P 0238                  .I, .INIT_REG [.I], .EXPECT_REG [.I],
;          0239                  .ACTUAL_REG [.I], (.EXPECT_REG [.I] XOR .ACTUAL_REG [.I]))
;          0240              ELSE
;        P 0241                  $DS_PRINTX ($ASCIC ('REG + !ZB !_!XL.3* !XL!3* !XL!/'),
;        P 0242                  .I, .INIT_REG [.I],
;          0243                  .EXPECT_REG [.I], .ACTUAL_REG [.I])
;          0244              END;
;          0245          END                         ! End if Reg_Flag set
;          0246      ELSE
;          0247          $DS_PRINTB ($ASCIC ('No Register data error was encountered.!/'));
;          0248  !
;          0249  !
;          0250  !   Print exception info
;          0251  !
;        P 0252      $DS_PRINTB ($ASCIC ('! Expected exception was !AC!/'),
;          0253                  .EXPECTED_EXCEPT);
;          0254
;        P 0255      $DS_PRINTB ($ASCIC ('! Exception received was !AC!/'),
;          0256                  .ACTUAL_EXCEPT);
;          0257
;          0258  $DS_ENDMESSAGE;
```

```
                        .TITLE  ENKAXK ENKAX Privileged Instruction Ldpctx
                        .IDENT  \1.3\

                        .PSECT  GLOBALS,NOEXE,2

00000 ACTUAL_EXCEPT::
                        .BLKB   4
00004 EXPECTED_EXCEPT::
                        .BLKB   4
00008 CHECK_STACK::
                        .BLKB   1
00009 MODE::    .BLKB   1
0000A           .BLKB   2
0000C PCB_ERR_FLAG::
                        .BLKB   4
00010 REG_ERR_FLAG::
                        .BLKB   4
00014 ORIG_PCBB::
                        .BLKB   4
00018 REG_DATA::
                        .BLKB   96
00078 INIT_PCB::
                        .BLKB   96
000D8 EXPECT_PCB::
```

```
                                                                                00138 ACTUAL_PCB::
                                                                                                .BLKB   96
                                                                                00198 INIT_REG::
                                                                                                .BLKB   96
                                                                                001F8 EXPECT_REG::
                                                                                                .BLKB   96
                                                                                00258 ACTUAL_REG::
                                                                                                .BLKB   96

                                                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                                2E 65 6E 6F 4E 05 00000 P.AAA:  .ASCII  <5>\None.\
61 72 65 70 6F 20 64 65 76 72 65 73 65 52 17 00006 P.AAB:  .ASCII  <23>\Reserved operand abort.\
                  2E 74 72 6F 62 61 20 64 6E 00015
58 54 43 50 44 4C 20 6E 6F 20 6C 69 61 46 0F 0001E P.AAC:  .ASCII  <15>\Fail on LDPCTX.\
                                                 2E 0002D
74 73 69 67 65 52 20 65 73 61 42 20 30 50 28 0002E P.AAD:  .ASCII  \(P0 Base Register high byte is not two.!\
73 69 20 65 74 79 62 20 68 67 69 68 20 72 65 0003D
                  21 2E 6F 77 74 20 74 6F 6E 20 0004C
                                                 2F 00056          .ASCII  \/\
74 73 69 67 65 52 20 65 73 61 42 20 30 50 27 00057 P.AAE:  .ASCII  \'P0 Base Register low bit is not zero.!/\
6E 20 73 69 20 74 69 62 20 77 6F 6C 20 72 65 00066
                  2F 21 2E 6F 72 65 7A 20 74 6F 00075
66 20 6F 72 65 7A 20 65 42 20 74 73 75 4D 27 0007F P.AAF:  .ASCII  \'Must Be zero field one, has all ones.!/\
61 20 73 61 68 20 2C 65 6E 6F 20 64 6C 65 69 0008E
                  2F 21 2E 73 65 6E 6F 20 6C 6C 0009D
65 72 20 2C 35 20 3D 20 4C 56 4C 54 53 41 22 000A7 P.AAG:  .ASCII  \"ASTLVL = 5, reserved to Digital.!/\
74 69 67 69 44 20 6F 74 20 64 65 76 72 65 73 000B6
                                 2F 21 2E 6C 61 000C5
74 73 69 67 65 52 20 65 73 61 42 20 31 50 25 000CA P.AAH:  .ASCII  \%P1 Base Register high byte not two.!/\
6F 6E 20 65 74 79 62 20 68 67 69 68 20 72 65 000D9
                        2F 21 2E 6F 77 74 20 74 000E8
74 73 69 67 65 52 20 65 73 61 42 20 31 50 24 000F0 P.AAI:  .ASCII  \$P1 Base Register low bit not zero.!/\
20 74 6F 6E 20 74 69 62 20 77 6F 6C 20 72 65 000FF
                        2F 21 2E 6F 72 65 7A 0010E
61 20 67 6E 69 74 72 61 74 73 20 42 43 50 16 00115 P.AAJ:  .ASCII  <22>\PCB starting at !XL.!/\
                  2F 21 2E 4C 58 21 20 74 00124
2A 34 21 4C 41 49 54 49 4E 49 5F 21 5F 21 2A 0012C P.AAK:  .ASCII  \*!_!_INITIAL!4* EXPECTED!3* ACTUAL!5* XO\
43 41 20 2A 33 21 44 45 54 43 45 50 58 45 20 0013B
                  4F 58 20 2A 35 21 4C 41 55 54 0014A
                                 2F 21 52 00154          .ASCII  \R!/\
58 21 5F 21 20 42 5A 21 20 2B 20 42 43 50 26 00157 P.AAL:  .ASCII  \&PCB + !ZB !_!XL!3* !XL!3* !XL!3* !XL!/\
4C 58 21 20 2A 33 21 4C 58 21 20 2A 33 21 4C 00166
                  2F 21 4C 58 21 20 2A 33 21 00175
58 21 5F 21 20 42 5A 21 20 2B 20 42 43 50 1F 0017E P.AAM:  .ASCII  <31>\PCB + !ZB !_!XL!3* !XL!3* !XL!/\
4C 58 21 20 2A 33 21 4C 58 21 20 2A 33 21 4C 0018D
                                 2F 21 0019C
72 65 20 61 74 61 64 20 62 63 50 20 6F 4E 1F 0019E P.AAN:  .ASCII  <31>\No Pcb data error encountered!/\
64 65 72 65 74 6E 75 6F 63 6E 65 20 72 6F 72 001AD
                                 2F 21 001BC
69 67 65 72 20 66 6F 20 79 70 6F 63 20 41 2D 001BE P.AAO:  .ASCII  \-A copy of register contents  begins at \
20 20 73 74 6E 65 74 6E 6F 63 20 72 65 74 73 001CD
                  20 74 61 20 73 6E 69 67 65 62 001DC
                                 2F 21 2E 4C 58 21 001E6          .ASCII  \!XL.!/\
2A 34 21 4C 41 49 54 49 4E 49 5F 21 5F 21 2A 001EC P.AAP:  .ASCII  \*!_!_INITIAL!4* EXPECTED!3* ACTUAL!5* XO\
43 41 20 2A 33 21 44 45 54 43 45 50 58 45 20 001FB
```

```
                        4F 58 20 2A 35 21 4C 41 55 54 0020A
                                                 2F 21 52 00214          .ASCII  \R!/\
58 21 5F 21 20 42 5A 21 20 2B 20 47 45 52 26 00217 P.AAQ:  .ASCII  \&REG + !ZB !_!XL!3* !XL!3* !XL!3* !XL!/\
4C 58 21 20 2A 33 21 4C 58 21 20 2A 33 21 4C 00226
                  2F 21 4C 58 21 20 2A 33 21 00235
58 21 5F 21 20 42 5A 21 20 2B 20 47 45 52 1F 0023E P.AAR:  .ASCII  <31>\  G + !ZB !_!XL!3* !XL!3* !XL!/\
4C 58 21 20 2A 33 21 4C 58 21 20 2A 33 21 4C 0024D
                                        2F 21 0025C
61 64 20 72 65 74 73 69 67 65 52 20 6F 4E 29 0025E P.AAS:  .ASCII  \)No Register data error was encountered.\
6E 65 20 73 61 77 20 72 6F 72 72 65 20 61 74 0026D
                  2E 64 65 72 65 74 6E 75 6F 63 0027C
                                        2F 21 00286          .ASCII  \!/\
63 78 65 20 64 65 74 63 65 70 78 45 5F 21 1E 00288 P.AAT:  .ASCII  <30>\!_Expected exception was !AC!/\
21 43 41 21 20 73 61 77 20 6E 6F 69 74 70 65 00297
                                        2F 002A6
65 72 20 6E 6F 69 74 70 65 63 78 45 5F 21 1E 002A7 P.AAU:  .ASCII  <30>\!_Exception received was !AC!/\
21 43 41 21 20 73 61 77 20 64 65 76 69 65 63 002B6
                                        2F 002C5
```

```
DSA$AL_APTMAIL=      65024
DSA$GL_FLAGS=        65024
DSA$GL_APTCOM=       65028
DSA$GL_PASSES=       65032
DSA$GL_UNITS=        65036
DSA$GL_SECTNO=       65040
DSA$GL_SID=          65044
DSA$GL_MSGTYP=       65088
DSA$GL_ERRNO=        65092
DSA$GL_EVENT=        65096
DSA$GL_SUBTNO=       65100
DSA$GL_TESTNO=       65104
DSA$GL_PASSNO=       65108
DSA$GL_DEVLEN=       65112
DSA$GL_DEVNAM=       65116
DSA$GL_MSGPTR=       65128
NONE=                P.AAA
RESV_OPR=            P.AAB
FAIL_LD_M=           P.AAC
SUB_MSG_1=           P.AAD
SUB_MSG_2=           P.AAE
SUB_MSG_3=           P.AAF
SUB_MSG_4=           P.AAG
SUB_MSG_5=           P.AAH
SUB_MSG_6=           P.AAI
                .EXTRN  EXECUTE, SAVE_RESULTS
                .EXTRN  DS$PRINTB, DS$PRINTX

                .PSECT  CODE,NOWRT,2

                007C 00000      .ENTRY  PRINT_ERR, Save R2,R3,R4,R5,R6          ; 0162
56 00000000G    9F   9E 00002   MOVAB   @#DS$PRINTX, R6                        ;
55      0000'   CF   9E 00009   MOVAB   PCB_ERR_FLAG, R5                       ;
54 00000000G    9F   9E 0000E   MOVAB   @#DS$PRINTB, R4                        ;
          14    AC   DD 00015   PUSHL   P1                                     ; 0174
64              01   FB 00018   CALLS   #1, DS$PRINTB                         ;
                65   D5 0001B   TSTL    PCB_ERR_FLAG                         ; 0178
                58   13 0001D   BEQL    4$                                   ;
```

```
                                 012C   C5   9F 0001F          PUSHAB  ACTUAL_PCB                              ; 0183
                                 0000'  CF   9F 00023          PUSHAB  P.AAJ                                   ;
                           64           02   FB 00027          CALLS   #2, DS$PRINTB                           ;
                                 0000'  CF   9F 0002A          PUSHAB  P.AAK                                   ; 0187
                           64           01   FB 0002E          CALLS   #1, DS$PRINTB                           ;
                                        52   D4 00031          CLRL    I                                       ; 0193
                           53    00CC C542   D0 00033 1$:       MOVL    EXPECT_PCB[I], R3                       ; 0204
                           51    012C C542   D0 00039          MOVL    ACTUAL_PCB[I], R1                       ;
                50         65    01           52 EF 0003F      EXTZV   I, #1, PCB_ERR_FLAG, R0                 ; 0201
                                 01           50 D1 00044      CMPL    R0, #1                                  ;
                                              17 12 00047      BNEQ    2$                                      ;
                           7E    53           51 CD 00049      XORL3   R1, R3, -(SP)                           ; 0204
                                              51 DD 0004D      PUSHL   R1                                      ;
                                              53 DD 0004F      PUSHL   R3                                      ;
                                    6C A542   DD 00051         PUSHL   INIT_PCB[I]                             ;
                                        52    DD 00055         PUSHL   I                                       ;
                                 0000'  CF   9F 00057          PUSHAB  P.AAL                                   ;
                           64           06   FB 0005B          CALLS   #6, DS$PRINTB                           ;
                                              11 11 0005E      BRB     3$                                      ; 0201
                                              51 DD 00060 2$:  PUSHL   R1                                      ; 0211
                                              53 DD 00062      PUSHL   R3                                      ;
                                    6C A542   DD 00064         PUSHL   INIT_PCB[I]                             ;
                                        52    DD 00068         PUSHL   I                                       ;
                                 0000'  CF   9F 0006A          PUSHAB  P.AAM                                   ;
                           66           05   FB 0006E          CALLS   #5, DS$PRINTX                           ;
                           BE    52           17 F3 00071 3$:  AOBLEQ  #23, I, 1$                              ; 0193
                                              07 11 00075      BRB     5$                                      ; 0178
                                 0000'  CF   9F 00077 4$:       PUSHAB  P.AAN                                   ; 0215
                           64           01   FB 0007B          CALLS   #1, DS$PRINTB                           ;
                                 04     A5   D5 0007E 5$:       TSTL    REG_ERR_FLAG                            ; 0219
                                        5B   13 00081          BEQL    9$                                      ;
                                 024C   C5   9F 00083          PUSHAB  ACTUAL_REG                              ; 0225
                                 0000'  CF   9F 00087          PUSHAB  P.AAO                                   ;
                           64           02   FB 0008B          CALLS   #2, DS$PRINTB                           ;
                                 0000'  CF   9F 0008E          PUSHAB  P.AAP                                   ; 0229
                           64           01   FB 00092          CALLS   #1, DS$PRINTB                           ;
                                        52   D4 00095          CLRL    I                                       ; 0230
                           53    01EC C542   D0 00097 6$:       MOVL    EXPECT_REG[I], R3                       ; 0239
                           51    024C C542   D0 0009D          MOVL    ACTUAL_REG[I], R1                       ;
                50   04    A5   01           52 EF 000A3      EXTZV   I, #1, REG_ERR_FLAG, R0                 ; 0236
                                 01           50 D1 000A9      CMPL    R0, #1                                  ;
                                              18 12 000AC      BNEQ    7$                                      ;
                           7E    53           51 CD 000AE      XORL3   R1, R3, -(SP)                           ; 0239
                                              51 DD 000B2      PUSHL   R1                                      ;
                                              53 DD 000B4      PUSHL   R3                                      ;
                                 018C C542    DD 000B6         PUSHL   INIT_REG[I]                             ;
                                        52    DD 000BB         PUSHL   I                                       ;
                                 0000'  CF   9F 000BD          PUSHAB  P.AAQ                                   ;
                           64           06   FB 000C1          CALLS   #6, DS$PRINTB                           ;
                                              12 11 000C4      BRB     8$                                      ; 0231
                                              51 DD 000C6 7$:  PUSHL   R1                                      ; 0243
                                              53 DD 000C8      PUSHL   R3                                      ;
                                 018C C542    DD 000CA         PUSHL   INIT_REG[I]                             ;
                                        52    DD 000CF         PUSHL   I                                       ;
                                 0000'  CF   9F 000D1          PUSHAB  P.AAR                                   ;
                           66           05   FB 000D5          CALLS   #5, DS$PRINTX                           ;
                           BB    52           17 F3 000D8 8$:  AOBLEQ  #23, I, 6$                              ; 0230
```

```
                                    07  11 000DC          BRB     10$                                     ; 0219
                              0000' CF  9F 000DE 9$:      PUSHAB  P.AAS                                   ; 0247
                           64       01  FB 000E2          CALLS   #1, DS$PRINTB                           ;
                             18     AC  DD 000E5 10$:     PUSHL   EXPECTED_EXCEPT                         ; 0253
                              0000' CF  9F 000E8          PUSHAB  P.AAT                                   ;
                           64       02  FB 000EC          CALLS   #2, DS$PRINTB                           ;
                             1C     AC  DD 000EF          PUSHL   ACTUAL_EXCEPT                           ; 0256
                              0000' CF  9F 000F2          PUSHAB  P.AAU                                   ;
                           64       02  FB 000F6          CALLS   #2, DS$PRINTB                           ;
                                        04 000F9          RET                                             ; 0162

; Routine Size:  250 bytes,    Routine Base:  CODE + 0000


;          0259
;          0260  ! Exception Handler
;          0261
;          0262  GLOBAL ROUTINE HANDLER (sig : ref vector, mech) =
;          0263      BEGIN
;          0264      GLOBAL
;          0265          ACTUAL_PSL,
;          0266          OPR_ABORT;
;          0267      EXTERNAL
;          0268          TEMP_PSL;
;          0269      MAP
;          0270          MECH : REF BLOCK [, BYTE];    ! Make it byte addressable
;          0271                                        ! to satisfy macro defined
;          0272                                        . in library
;          0273
;          0274
;          0275      OPR_ABORT = 0;                    ! Initialize abort flag
;          0276      ACTUAL_PSL = .SIG [3];            ! Send back the PSL received after
;          0277                                        ! instruction execution
;          0278      SELECTONE 1 OF                    ! Only one exception code will be valid
;          0279          SET
;          0280            [ .SIG [1] EQL SS$_ROPRAND] :
;          0281  !
;          0282  !               The  exception is passed to the main routine
;          0283  !               implicitly in ACTUAL_EXCEPT
;          0284  .
;          0285            BEGIN
;          0286                  ACTUAL_EXCEPT =  RESV_OPR;
;          0287                  OPR_ABORT = 1;  ! Set exception flag
;          0288            END;
;          0289  !
;          0290  ! ACTUAL_EXCEPT remains None if this handler is called because of BPT instruction
;          0291  !
;          0292            [ .SIG [1] EQL SS$_BREAK] :
;          0293                  ACTUAL_EXCEPT =  NONE;
;          0294
;          0295            [OTHERWISE] :
;          0296
;          0297  !       If it is not an exception normally obtained from a LDPCTX or SVPCTX
;          0298  .
;          0299                  RETURN SS$_RESIGNAL ! Something unexpected occured, tell DS to
;          0300                                      ! take care of it
;          0301          TES;
```

```
;          0302      .MECH [CHF$L_MCH_FRAME] = 0;    !  Disable handler.
;          0303
;          0304      SIG [2] = SAVE_RESULTS;      ' Restore PC to return to Execute at Save_Results
;          0305      SS$_CONTINUE
;          0306      END;                         . End of Exception Handler


                                                        .PSECT  GLOBALS,NOEXE,2

                                                002B8 ACTUAL_PSL::
                                                        .BLKB   4
                                                002BC OPR_ABORT::
                                                        .BLKB   4

                                                        .EXTRN  TEMP_PSL

                                                        .PSECT  CODE,NOWRT,2

                                                    0000 00000          .ENTRY  HANDLER, Save nothing                   ; 0262
                                            0000'   CF   D4 00002       CLRL    OPR_ABORT                               ; 0275
                                         51   04   AC   D0 00006       MOVL    SIG, R1                                 ; 0276
                                    0000'   CF   0C   A1   D0 0000A     MOVL    12(R1), ACTUAL_PSL                      ;
                               00000454     8F   04   A1   D1 00010     CMPL    4(R1), #1108                            ; 0280
                                                  0E   12   00018       BNEQ    1$                                      ;
                                    0000'   CF   0000'  CF   9E 0001A   MOVAB   RESV_OPR, ACTUAL_EXCEPT                 ; 0285
                                    0000'   CF         01   D0 00021     MOVL    #1, OPR_ABORT                           ; 0287
                                                  19   11   00026       BRB     3$                                      ; 0278
                               00000414     8F   04   A1   D1 00028 1$:  CMPL    4(R1), #1044                            ; 0292
                                                  09   12   00030       BNEQ    2$                                      ;
                                    0000'   CF   0000'  CF   9E 00032   MOVAB   NONE, ACTUAL_EXCEPT                     ; 0278
                                                  06   11   00039       BRB     3$                                      ;
                                         50   0918   8F   3C 0003B 2$:  MOVZWL  #2328, R0                              ; 0299
                                                        04   00040       RET                                             ;
                                         50   08   AC   D0 00041 3$:    MOVL    MECH, R0                                ; 0302
                                              04   B0   D4 00045         CLRL    @4(R0)                                  ;
                                    08   A1 00000000G  EF   9E 00048     MOVAB   SAVE_RESULTS, 8(R1)                     ; 0304
                                         50         01   D0 00050       MOVL    #1, R0                                  ; 0262
                                                        04   00053       RET                                             ;

; Routine Size:  84 bytes,    Routine Base:  CODE + 00FA


;          0307  !
;          0308  ! Compare PCB routire
;          0309  !
;          0310  ROUTINE COMPARE_PCB : NOVALUE =
;          0311  ! Compares Actual_Pcb against Expect_Pcb and Actual_Regs against Expect_Regs
;          0312  ! and sets Pcb_Err_Flag and/or Reg_Err_Flag, respectively, if any data
;          0313  ! discrepency is discovered.  For example if Actual_Pcb[R0] does not
;          0314  ! correspond with Expect_Pcb[R0] then Pcb_Err_Flag bit four is set, because
;          0315  ! R0 is the fourth data field in a PCB.
;          0316
;          0317  BEGIN
;          0318  PCB_ERR_FLAG = 0;                ! Begin with no errors
;          0319  REG_ERR_FLAG = 0;
;          0320
```

```
;          0321  !
;          0322  !
;          0323      SELECT 1 OF
;          0324          SET
;          0325  !
;          0326  !       Do not check Ksp in the Pcb, it probably won't be the same
;          0327  !       as in Expect_Pcb
;          0328  !
;          0329          [.ACTUAL_PCB [PTX$L_ESP] NEQ .EXPECT_PCB [PTX$L_ESP]] :
;          0330                  PCB_ERR_FLAG [1] = 1;
;          0331
;          0332          [.ACTUAL_PCB [PTX$L_SSP] NEQ .EXPECT_PCB [PTX$L_SSP]] :
;          0333                  PCB_ERR_FLAG [2] = 1;
;          0334
;          0335          [.ACTUAL_PCB [PTX$L_USP] NEQ .EXPECT_PCB [PTX$L_USP]] :
;          0336                  PCB_ERR_FLAG [3] = 1
;          0337          TES;
;          0338
;          0339  ! Check the general registers saved in the PCB
;          0340
;          0341      INCR I FROM 4 TO 16 DO
;          0342          BEGIN
;          0343          MAP
;          0344              ACTUAL_PCB : VECTOR,        ! Make it easier to reference longwds
;          0345              EXPECT_PCB : VECTOR;
;          0346  !
;          0347  !       Check each longword in the PCB from R0 to AP against the expected
;          0348  !       value. Expect_Pcb [FP] probably is not correct.
;          0349  !
;          0350          IF .ACTUAL_PCB [.I] NEQ .EXPECT_PCB [.I] THEN
;          0351              PCB_ERR_FLAG [.I] = 1
;          0352          END;
;          0353
;          0354  ! Check memory management data fields
;          0355
;          0356      IF .CHECK_STACK GEQ 2 THEN
;          0357          SELECT 1 OF
;          0358              SET
;          0359              [.ACTUAL_PCB [PTX$L_P0BR] NEQ .EXPECT_PCB [PTX$L_P0BR]] :
;          0360                  PCB_ERR_FLAG [18] = 1;
;          0361
;          0362              [.ACTUAL_PCB [PTX$V_P0LR] NEQ .EXPECT_PCB [PTX$V_P0LR]] :
;          0363                  PCB_ERR_FLAG [19] = 1;
;          0364
;          0365              [.ACTUAL_PCB [PTX$V_MBZ1] NEQ 0] :
;          0366                  PCB_ERR_FLAG [20] = 1;
;          0367
;          0368              [.ACTUAL_PCB [PTX$V_ASTLVL] NEQ .EXPECT_PCB [PTX$V_ASTLVL]] :
;          0369                  PCB_ERR_FLAG [21] = 1;
;          0370
;          0371              [.ACTUAL_PCB [PTX$V_MBZ2] NEQ 0] :
;          0372                  PCB_ERR_FLAG [22] = 1;
;          0373
;          0374              [.ACTUAL_PCB [PTX$L_P1BR] NEQ .EXPECT_PCB [PTX$L_P1BR]] :
;          0375                  PCB_ERR_FLAG [23] = 1;
;          0376
;          0377              [.ACTUAL_PCB [PTX$V_P1LR] NEQ .EXPECT_PCB [PTX$V_P1LR]] :
```

```
;          0378                PCB_ERR_FLAG [24] = 1;
;          0379
;          0380            [.ACTUAL_PCB [PTX$V_MBZ3] NEQ 0] :
;          0381                PCB_ERR_FLAG [25] = 1;
;          0382
;          0383            [.ACTUAL_PCB [PTX$V_PME] NEQ .EXPECT_PCB [PTX$V_PME]] :
;          0384                PCB_ERR_FLAG [26] = 1;
;          0385            TES;
;          0386  !
;          0387  ! End of PCB check
;          0388  !
;          0389
;          0390  !
;          0391  ! Now check if registers got everything they were supposed to
;          0392  !
;          0393      SELECT 1 OF
;          0394          SET
;          0395  !
;          0396  !       The Ksp in the registers should be correct
;          0397  !
;          0398          [.ACTUAL_REG [PTX$L_KSP] NEQ .EXPECT_REG [PTX$L_KSP]] :
;          0399                  REG_ERR_FLAG [0] = 1;
;          0400
;          0401          [.ACTUAL_REG [PTX$L_ESP] NEQ .EXPECT_REG [PTX$L_ESP]] :
;          0402                  REG_ERR_FLAG [1] = 1;
;          0403
;          0404          [.ACTUAL_REG [PTX$L_SSP] NEQ .EXPECT_REG [PTX$L_SSP]] :
;          0405                  REG_ERR_FLAG [2] = 1;
;          0406
;          0407          [.ACTUAL_REG [PTX$L_USP] NEQ .EXPECT_REG [PTX$L_USP]] :
;          0408                  REG_ERR_FLAG [3] = 1
;          0409          TES;
;          0410
;          0411  ! Check the general registers saved in the REG
;          0412
;          0413          INCR I FROM 4 TO 16 DO          ! R0 to AP
;          0414              BEGIN
;          0415              MAP
;          0416                  ACTUAL_REG : VECTOR,    ! Make it easy to compare longword
;          0417                  EXPECT_REG : VECTOR;    ! offsets
;          0418  !
;          0419  !       Check all resulting general registers against expected values
;          0420  !
;          0421              IF .ACTUAL_REG [.I] NEQ .EXPECT_REG [.I] THEN
;          0422                  REG_ERR_FLAG [.I] = 1
;          0423              END;
;          0424
;          0425  ! Check memory management data fields
;          0426
;          0427      SELECT 1 OF
;          0428          SET
;          0429          [.ACTUAL_REG [PTX$L_P0BR] NEQ .EXPECT_REG [PTX$L_P0BR]] :
;          0430                  REG_ERR_FLAG [18] = 1;
;          0431
;          0432          [.ACTUAL_REG [PTX$V_P0LR] NEQ .EXPECT_REG [PTX$V_P0LR]] :
;          0433                  REG_ERR_FLAG [19] = 1;
;          0434
```

```
;          0435  !       Skip MBZ bits, they don't exist in hardware, but wish to keep
;          0436  !       consistant with Pcb_Err_Flag
;          0437
;          0438          [.ACTUAL_REG [PTX$V_ASTLVL] NEQ .EXPECT_REG [PTX$V_ASTLVL]] :
;          0439                  REG_ERR_FLAG [21] = 1;
;          0440
;          0441          [.ACTUAL_REG [PTX$L_P1BR] NEQ .EXPECT_REG [PTX$L_P1BR]] :
;          0442                  REG_ERR_FLAG [23] = 1;
;          0443
;          0444          [.ACTUAL_REG [PTX$V_P1LR] NEQ .EXPECT_REG [PTX$V_P1LR]] :
;          0445                  REG_ERR_FLAG [24] = 1;
;          0446
;          0447          [.ACTUAL_REG [PTX$V_PME] NEQ .EXPECT_REG [PTX$V_PME]] :
;          0448                  REG_ERR_FLAG [26] = 1;
;          0449
;          0450          TES;
;          0451
;          0452  ! End of REG check, end of Compare routine
;          0453
;          0454  END;


                                                0004 00000 COMPARE_PCB:
                                                           .WORD   Save R2                                  ; 0310
                                      52    0000' CF  9E 00002       MOVAB   PCB_ERR_FLAG, R2                 ;
                                            62  7C 00007            CLRQ    PCB_ERR_FLAG                     ; 0318
                                00D0  C2    0130  C2  D1 00009       CMPL    ACTUAL_PCB+4, EXPECT_PCB+4       ; 0329
                                            03  13 00010             BEQL    1$                               ;
                                      62          02  88 00012       BISB2   #2, PCB_ERR_FLAG                 ; 0330
                                00D4  C2    0134  C2  D1 00015 1$:   CMPL    ACTUAL_PCB+8, EXPECT_PCB+8       ; 0332
                                            03  13 0001C             BEQL    2$                               ;
                                      62          04  88 0001E       BISB2   #4, PCB_ERR_FLAG                 ; 0333
                                00D8  C2    0138  C2  D1 00021 2$:   CMPL    ACTUAL_PCB+12, EXPECT_PCB+12     ; 0335
                                            03  13 00028             BEQL    3$                               ;
                                      62          08  88 0002A       BISB2   #8, PCB_ERR_FLAG                 ; 0336
                                      50          04  D0 0002D 3$:   MOVL    #4, I                            ; 0341
                                00CC  C240  012C  C240  D1 00030 4$:   CMPL    ACTUAL_PCB[I], EXPECT_PCB[I]     ; 0350
                                            04  13 00039             BEQL    5$                               ;
                          00            62          50  E2 0003B       BBSS    I, PCB_ERR_FLAG, 5$              ; 0351
                          ED            50          10  F3 0003F 5$:   AOBLEQ  #16, I, 4$                      ; 0341
                                        02    FC    A2  91 00043       CMPB    CHECK_STACK, #2                  ; 0356
                                            03  1E 00047             BGEQU   6$                               ;
                                          0091  31 00049             BRW     15$                              ;
                                011C  C2    017C  C2  D1 0004C 6$:   CMPL    ACTUAL_PCB+80, EXPECT_PCB+80     ; 0359
                                            04  13 00053             BEQL    7$                               ;
                                  02    A2          04  88 00055       BISB2   #4, PCB_ERR_FLAG+2               ; 0360
50      0120    C2                16          00  EF 00059 7$:   EXTZV   #0, #22, EXPECT_PCB+84, R0          ; 0362
50      0180    C2                16          00  ED 00060       CMPZV   #0, #22, ACTUAL_PCB+84, R0           ;
                                            04  13 00067             BEQL    8$                               ;
                                  02    A2          08  88 00069       BISB2   #8, PCB_ERR_FLAG+2               ; 0363
                                  C0    8F    0182  C2  93 0006D 8$:   BITB    ACTUAL_PCB+86, #192              ; 0365
                                            04  13 00073             BEQL    9$                               ;
                                  02    A2          10  88 00075       BISB2   #16, PCB_ERR_FLAG+2              ; 0366
50      0123    C2                03          00  EF 00079 9$:   EXTZV   #0, #3, EXPECT_PCB+87, R0           ; 0368
```

```
                50      0183    C2              03              00  ED 00080        CMPZV   #0, #3, ACTUAL_PCB+87, R0                       ;
                                                                04  13 00087        BEQL    10$                                             ;
                                        02      A2              20  88 00089        BISB2   #32, PCB_ERR_FLAG+2                             ; 0369
                                        F8      8F      0183    C2  93 0008D 10$:   BITB    ACTUAL_PCB+87, #248                             ; 0371
                                                                05  13 00093        BEQL    11$                                             ;
                                        02      A2      40      8F  88 00095        BISB2   #64, PCB_ERR_FLAG+2                             ; 0372
                                      0124      C2      0184    C2  D1 0009A 11$:   CMPL    ACTUAL_PCB+88, EXPECT_PCB+88                    ; 0374
                                                                05  13 000A1        BEQL    12$                                             ;
                                        02      A2      80      8F  88 000A3        BISB2   #128, PCB_ERR_FLAG+2                            ; 0375
                50      0128    C2              16              00  EF 000A8 12$:   EXTZV   #0, #22, EXPECT_PCB+92, R0                     ; 0377
                50      0188    C2              16              00  ED 000AF        CMPZV   #0, #22, ACTUAL_PCB+92, R0                      ;
                                                                04  13 000B6        BEQL    13$                                             ;
                                        03      A2              01  88 000B8        BISB2   #1, PCB_ERR_FLAG+3                              ; 0378
                                      7FC0      8F      018A    C2  B3 000BC 13$:   BITW    ACTUAL_PCB+94, #32704                           ; 0380
                                                                04  13 000C3        BEQL    14$                                             ;
                                        03      A2              02  88 000C5        BISB2   #2, PCB_ERR_FLAG+3                              ; 0381
                50      012B    C2              01              07  EF 000C9 14$:   EXTZV   #7, #1, EXPECT_PCB+95, R0                     ; 0383
                50      018B    C2              01              07  ED 000D0        CMPZV   #7, #1, ACTUAL_PCB+95, R0                       ;
                                                                04  13 000D7        BEQL    15$                                             ;
                                        03      A2              04  88 000D9        BISB2   #4, PCB_ERR_FLAG+3                              ; 0384
                                      01EC      C2      024C    C2  D1 000DD 15$:   CMPL    ACTUAL_REG, EXPECT_REG                          ; 0398
                                                                04  13 000E4        BEQL    16$                                             ;
                                        04      A2              01  88 000E6        BISB2   #1, REG_ERR_FLAG                                ; 0399
                                      01F0      C2      0250    C2  D1 000EA 16$:   CMPL    ACTUAL_REG+4, EXPECT_REG+4                      ; 0401
                                                                04  13 000F1        BEQL    17$                                             ;
                                        04      A2              02  88 000F3        BISB2   #2, REG_ERR_FLAG                                ; 0402
                                      01F4      C2      0254    C2  D1 000F7 17$:   CMPL    ACTUAL_REG+8, EXPECT_REG+8                      ; 0404
                                                                04  13 000FE        BEQL    18$                                             ;
                                        04      A2              04  88 00100        BISB2   #4, REG_ERR_FLAG                                ; 0405
                                      01F8      C2      0258    C2  D1 00104 18$:   CMPL    ACTUAL_REG+12, EXPECT_REG+12                    ; 0407
                                                                04  13 0010B        BEQL    19$                                             ;
                                        04      A2              08  88 0010D        BISB2   #8, REG_ERR_FLAG                                ; 0408
                                                50              04  D0 00111 19$:   MOVL    #4, I                                           ; 0413
                                    01EC[C240]      024C[C240]      D1 00114 20$:   CMPL    ACTUAL_REG[I], EXPECT_REG[I]                    ; 0421
                                                                05  13 0011D        BEQL    21$                                             ;
                        00      04      A2              50  E2 0011F        BBSS    I, REG_ERR_FLAG, 21$                            ; 0422
                        EC              50              10  F3 00124 21$:   AOBLEQ  #16, I, 20$                                    ; 0413
                                      023C      C2      029C    C2  D1 00128        CMPL    ACTUAL_REG+80, EXPECT_REG+80                    ; 0429
                                                                04  13 0012F        BEQL    22$                                             ;
                                        06      A2              04  88 00131        BISB2   #4, REG_ERR_FLAG+2                              ; 0430
                50      0240    C2              16              00  EF 00135 22$:   EXTZV   #0, #22, EXPECT_REG+84, R0                     ; 0432
                50      02A0    C2              16              00  ED 0013C        CMPZV   #0, #22, ACTUAL_REG+84, R0                      ;
                                                                04  13 00143        BEQL    23$                                             ;
                                        06      A2              08  88 00145        BISB2   #8, REG_ERR_FLAG+2                              ; 0433
                50      0243    C2              03              00  EF 00149 23$:   EXTZV   #0, #3, EXPECT_REG+87, R0                     ; 0438
                50      02A3    C2              03              00  ED 00150        CMPZV   #0, #3, ACTUAL_REG+87, R0                       ;
                                                                04  13 00157        BEQL    24$                                             ;
                                        06      A2              20  88 00159        BISB2   #32, REG_ERR_FLAG+2                             ; 0439
                                      0244      C2      02A4    C2  D1 0015D 24$:   CMPL    ACTUAL_REG+88, EXPECT_REG+88                    ; 0441
                                                                05  13 00164        BEQL    25$                                             ;
                                        06      A2      80      8F  88 00166        BISB2   #128, REG_ERR_FLAG+2                            ; 0442
                50      0248    C2              16              00  EF 0016B 25$:   EXTZV   #0, #22, EXPECT_REG+92, R0                     ; 0444
                50      02A8    C2              16              00  ED 00172        CMPZV   #0, #22, ACTUAL_REG+92, R0                      ;
                                                                04  13 00179        BEQL    26$                                             ;
                                        07      A2              01  88 0017B        BISB2   #1, REG_ERR_FLAG+3                              ; 0445
                50      024B    C2              01              07  EF 0017F 26$:   EXTZV   #7, #1, EXPECT_REG+95, R0                     ; 0447
                50      02AB    C2              01              07  ED 00186        CMPZV   #7, #1, ACTUAL_REG+95, R0                       ;
```

```
                                        04  13 0018D          BEQL    27$                                     ;
                              07   A2     04  88 0018F          BISB2   #4, REG_ERR_FLAG+3                      ; 0448
                                          04 00193 27$:       RET                                             ; 0310

; Routine Size:  404 bytes,    Routine Base:  CODE + 014E


;          0455
;          0456  ! Begin main set-up code
;          0457
;          0458  ! ++
;          0459  !       Test comments
;          0460  ! See ENKAXL.MAR for test comments.
;          0461  ; The subtests are independent of each other.
;          0462  !
;          0463  ! --
;          0464
;          0465  GLOBAL ROUTINE TEST_CONTROL =
;          0466      BEGIN
;          0467
;          0468      BUILTIN
;          0469          MFPR,                   ! Move from Process register
;          0470          MTPR;                   ! Move to Process register
;          0471
;          0472  ! Pattern set-up to initialize memory management registers before
;          0473  ! LDPCTX instruction ie. P0LR, P0LR, P1BR, P1LR
;          0474
;          0475      OWN
;          0476          INIT_MNGT:VECTOR[4] INITIAL(%X'8A5A5A58',
;          0477                                      %X'040A5A5A',
;          0478                                      %X'8A5A5A58',
;          0479                                      %X'000A5A5A');
;          0480      LOCAL
;          0481          STATUS,                 ! Returns status codes from DS calls
;          0482          DATA,                   ! Temporarily holds fill characters when
;          0483                                  ! defining REG_DATA
;          0484          ORIG_PCBB,              ! Holds the Pcbb of the process context
;          0485                                  ! previous to the first subtest
;          0486          TEMP_LONGWORD;          ! Temporary longword storage
;          0487  ! BEGIN code
;          0488
;          0489      IF .DSA$V_USER THEN         ! Can only be run in stand-alone
;          0490          $DS_ABORT (ARG - PROGRAM);
;          0491
;          0492  !
;          0493  !    Memory Management off, so we may have our own Kernel stack ptr
;          0494  !    If mem mgt can not be shut off (operator requested it on) then
;          0495  !    we do not want to continue the test
;          0496  !
;          0497      STATUS = $DS_MMOFF;
;          0498      IF NOT .STATUS THEN
;          0499          BEGIN
;        P 0500          $DS_ERRHARD (MSGADR = $ASCIC('Unable to shut off memory management.'),
;        L 0501                       PRLINK = PRINT_ERR);
; %PRINT:          Test 11, Subtest 0, Error 1
;          0502          $DS_ABORT (ARG = PROGRAM);
;          0503          END;
```

```
;          0504
;          0505  !
;          0506  ! REG_DATA is used to initialize the registers or a Pcb to a valid data
;          0507  ! pattern. Don't fool around with the KSP, don't know where we would end up.
;          0508
;          0509      REG_DATA [PTX$L_ESP] = 0;                   ! should not try to access
;          0510      REG_DATA [PTX$L_SSP] = 0;                   ! any other stack pointer
;          0511      REG_DATA [PTX$L_USP] = 0;                   ! except on Priv_instr aborts
;          0512
;          0513  !
;          0514  !   Fill the registers with various values, the value in FP will not be
;          0515  !   loaded into the registers.
;          0516  !
;          0517      DATA = 0;                       ! Start R0 with all zeroes
;          0518      INCR I FROM 4 TO 17 DO          ! The longword offsets of gen regs in PCB
;          0519          BEGIN
;          0520          MAP
;          0521              REG_DATA : VECTOR;  ! Make longword offsets easier
;          0522
;          0523          REG_DATA [.I] = .DATA;
;          0524          DATA = .DATA + %X'11111111'     ! Effectively, add hex 11 to the
;          0525                                          ! fill character for the next register
;          0526          END;
;          0527
;          0528
;          0529  !
;          0530  !   Leave the memory management registers at the current valid value.
;          0531  !
;          0532      MFPR (PR$_P0BR, REG_DATA [PTX$L_P0BR]);     ! Get the current value of
;          0533                                                  ! P0 Base Register
;          0534      MFPR (PR$_P0LR, TEMP_LONGWORD);             ! Store P0 length register in
;          0535                                                  ! a longword
;          0536      REG_DATA [PTX$V_P0LR] = .TEMP_LONGWORD<0,22>;! Extract the field
;          0537                                                  ! containing the value and
;          0538                                                  ! put it in Reg_Data
;          0539      MFPR (PR$_ASTLVL,  TEMP_LONGWORD);  ! MFPR always wants a longword
;          0540      REG_DATA [PTX$V_ASTLVL] = .TEMP_LONGWORD<0,3>;!AST level
;          0541
;          0542      MFPR (PR$_P1BR,  REG_DATA [PTX$L_P1BR]);
;          0543      MFPR (PR$_P1LR, TEMP_LONGWORD);
;          0544      REG_DATA [PTX$V_P1LR] = .TEMP_LONGWORD<0,22>;! P1 Base Register
;          0545      MFPR (PR$_PME, TEMP_LONGWORD);              ! Don't want to screw up
;          0546                                                  ! anyone's performance
;          0547                                                  ! monitering, so keep it the
;          0548                                                  ! same
;          0549      REG_DATA [PTX$V_PME] = .TEMP_LONGWORD<0,1>;
;          0550
;          0551  !  All done with defining Reg_Data
;          0552  !
;          0553  ! INIT_PCB is used to initialize PCB variables that might be changed by the
;          0554  ! execution of an instruction. It contains a background of 'Z's which no
;          0555  ! instruction produces so it is easy to see if something was not loaded in
;          0556  ! correctly.  Init_Pcb becomes Init_Reg for the Ldpctx instructions.
;          0557  !
;          0558
;          0559      CH$FILL (%C'Z', PTX$K_LENGTH, INIT_PCB);
;          0560
```

```
;          0561
;          0562  ! ++
;          0563  !
;          0564  ! CHECK_STACK determines if the interrupt and kernel stacks should be
;          0565  ! monitered before and after instruction execution. CHECK_STACK is an
;          0566  ! implicit parameter to the EXECUTE routine in ENKAXL
;          0567  !       Check_stack = 0  K -> I  start on kernel stack and end up on interrupt
;          0568  !                   = 1  I -> I  start on interrupt stack, end on interrupt
;          0569  !                   = 2  K -> K  start on kernel stack, end on kernel
;          0570  !                   = 3  I -> K  start on interrupt stack, end on kernel
;          0571  !
;          0572  ! --
;          0573
;          0574
;          0575  !
;          0576
;          0577
;          0578  !
;          0579  !
;          0580  ! Set up for LDPCTX subtests
;          0581  ! The contents of Actual_pcb will be loaded into the registers upon execution
;          0582  ! of the Ldpctx instruction
;          0583  !
;          0584      MFPR (PR$_PCBB, ORIG_PCBB);         ! Save the original PCB location
;          0585      MTPR (%REF(ACTUAL_PCB), PR$_PCBB);
;          0586
;          0587  !
;          0588  ! The registers should be initialized with background before each execution
;          0589  ! of Ldpctx, so fill Init_Reg with 'Z's from Init_pcb
;          0590  !
;          0591  ! Make sure memory management longwords in the PCB get loaded with the right
;          0592  ! initial pattern
;          0593  !
;          0594      CH$MOVE (32, INIT_MNGT, INIT_PCB + 80);
;          0595      CH$MOVE (PTX$K_LENGTH, INIT_PCB, INIT_REG);
;          0596  !
;          0597  ! Use typical data to load registers
;          0598  !
;          0599  !     CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;          0600
;          0601  !+
;          0602  ! SUBTEST DESCRIPTION:
;          0603  !       Subtest 1
;          0604  !       Out of bounds P0 Base Register, test reserved operand abort,
;          0605  !       LDPCTX.
;          0606  !
;          0607  ! SUBTEST STEPS:
;          0608  !
;          0609  !       Initialize the variables, background in Init_Regs, and data in
;          0610  !       Init_Pcb. The registers will be loaded with background data before
;          0611  !       the LDPCTX execution.
;          0612  !       Modify the P0 Base Register in Init_Pcb and Actual_Pcb so the value
;          0613  !       is out of bounds (set last byte to 1).
;          0614  !       Start on the interrupt stack with kernel mode privileges.
;          0615  !       When the execution gets to load P0BR from Actual_Pcb to registers
;          0616  !       it will check the validity of the longword.  In this case it is
;          0617  !       invalid and DS is called to handle a reserved operand abort, DS
```

```
;          0618  !       calls the exception handler in this module, which sets Actual_-
;          0619  !       Except to Resv_Opr abort.  The flow of control returns to the
;          0620  !       code that checks the stack for validity in Execute.  When Execute
;          0621  !       finishes it returns a status code of one for success and zero if
;          0622  !       it has found something wrong with the stack. The subtest code
;          0623  !       calls Print_Err if any errors are encountered, the test is aborted
;          0624  !       if Execute did not come back sucessfully.
;          0625  !
;          0626  ! ERRORS:
;          0627  !
;          0628  !       Did not switch to Kernel stack
;          0629  !       Unexpected exception - If the implementation of the instruction
;          0630  !               used checks for the POBR error, the wrong exception took
;          0631  !               priority.  If the check is not included the exception
;          0632  !               received would relate to the out of bounds POBR value
;          0633  !               loaded into the registers.
;          0634  !
;          0635  ! DEBUG:
;          0636  !
;          0637  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
;          0638  !
;          0639  !-
;          0640      $DS_BGNSUB
;          0641  !
;          0642  !    Out-of-bounds POBR should get a reserved operand abort.
;          0643  !
;          0644          CHECK_STACK = 3;            ! Start on interrupt stack, end on kernel,
;          0645                                      ! LDPCTX
;          0646          MODE = 0;                   ! All the privileges you'll ever need
;          0647          ACTUAL_EXCEPT = NONE;                       ! Clear exceptions
;          0648          EXPECTED_EXCEPT = RESV_OPR;                 ! Expect a resv_opr_abort
;          0649  !
;          0650  !       Use normal data pattern to load the registers
;          0651  !
;          0652          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;          0653          INIT_PCB [83,6,2,0] = 1;        !Make POBR out-of-bounds
;          0654  !
;          0655  !       PCB should remain the same whether it is loaded into the registers or
;          0656  !       not
;          0657  !
;          0658          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;          0659          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);   ! Init Actual_Pcb
;          0660  !
;          0661  !       Initialize Actual_Reg with background, Init_regs still all 'Z's
;          0662  !
;          0663          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
;          0664  !
;          0665  !       Registers should not get any data loaded in
;          0666  !
;          0667          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;          0668
;          0669          IF NOT EXECUTE () THEN
;          0670              $DS_ABORT (ARG = PROGRAM);
;          0671  !
;          0672  !       Registers are in an unknown state, so no check
;          0673  !
;          0674          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
```

```
;          0675              BEGIN
;        P 0676              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;        P 0677                           PRLINK = PRINT_ERR,
;        P 0678                           P1 = SUB_MSG_1,
;        P 0679                           P2 = .EXPECTED_EXCEPT,
;        L 0680                           P3 = .ACTUAL_EXCEPT);
; %PRINT:          Test 11, Subtest 1, Error 2
;          0681              END;
;          0682          $DS_ENDSUB;
;          0683
;          0684  !+
;          0685  ! SUBTEST DESCRIPTION:
;          0686  !       Subtest 2
;          0687  !       Invalid P0 Base Register value, start on interrupt stack,
;          0688  !       rest of data valid. Test for reserved operand abort.
;          0689  !
;          0690  ! SUBTEST STEPS:
;          0691  !
;          0692  !       Initialize as in subtest 1, set Actual_Pcb P0BR bit 1 to 1,
;          0693  !       an invalid value, it must be zero if it is to be loaded into
;          0694  !       the registers.
;          0695  !       If the correct exception is not in Actual_Except, Print_Err is
;          0696  !       called.  The data is unknown at this point, some of it is
;          0697  !       probably already loaded, but we don't know how much, so no
;          0698  !       data check is made.
;          0699  !
;          0700  ! ERRORS:
;          0701  !
;          0702  !       Did not switch to kernel stack - should have ended up on kernel
;          0703  !               stack, even if resv_opr exception was received, see
;          0704  !               subtest 1.
;          0705  !       Unexpected exception - If the check is not made in the implementa-
;          0706  !               tion another exception may occur because of the bad value
;          0707  !               for P0BR in the registers.
;          0708  !
;          0709  !
;          0710  ! DEBUG:
;          0711  !
;          0712  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
;          0713  !
;          0714  !-
;          0715      $DS_BGNSUB
;          0716  !
;          0717  !    Out of bounds P0BR should get a reserved operand abort
;          0718  !
;          0719          CHECK_STACK = 3;        ! Start on interrupt, end up on kernel stack,
;          0720                                  ! LDPCTX
;          0721          MODE = 0;               ! Kernel mode
;          0722          ACTUAL_EXCEPT = NONE;                   ! Clear exceptions
;          0723          EXPECTED_EXCEPT = RESV_OPR;             ! Expect a reserved operand abort
;          0724  !
;          0725  !       Use normal data, then corrupt P0BR
;          0726  !
;          0727          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;          0728          (INIT_PCB [PTX$L_P0BR])<0,1> = 1;               ! Make P0BR out-of-bounds
;          0729  !
;          0730  !       The Pcb should remain unchanged
```

```
;          0731  !
;          0732          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;          0733          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);   ! Init Actual_Pcb
;          0734                                                          ! with corrupt P0BR
;          0735
;          0736  !       Initialize Actual_Reg with background from Init_reg
;          0737  !
;          0738          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
;          0739  !
;          0740  !       The registers should not complete loading in the data from Init_pcb
;          0741  !
;          0742          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;          0743
;          0744          IF NOT EXECUTE () THEN
;          0745              $DS_ABORT (ARG = PROGRAM);
;          0746
;          0747          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
;          0748              BEGIN
;        P 0749              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;        P 0750                           PRLINK = PRINT_ERR,
;        P 0751                           P1 = SUB_MSG_2,
;        P 0752                           P2 = .EXPECTED_EXCEPT,
;        L 0753                           P3 = .ACTUAL_EXCEPT);
; %PRINT:          Test 11, Subtest 2, Error 3
;          0754                END;
;          0755      $DS_ENDSUB;
;          0756
;          0757  !+
;          0758  ! SUBTEST
;          0759  !       Subtest 3
;          0760  !       Test reserved operand abort, set MBZ two to all ones. Execute
;          0761  !       from the interrupt stack.
;          0762  !
;          0763  ! SUBTEST STEPS:
;          0764  !
;          0765  !       Initialize variables as in subtest 1, set Actual_Pcb [MBZ2] to
;          0766  !       negative one (all bits set), set Check_Stack to start on Interrupt
;          0767  !       stack. During execution DS should be called with a reserved oper-
;          0768  !       and abort, as in subtest 9. If the check is not made in the
;          0769  !       implementation tested another exception may occur before the break-
;          0770  !       point instruction which sends control to the exception handler after
;          0771  !       execution of the instruction is complete, this would result in an
;          0772  !       unexpected exception which may actually be valid. If control is
;          0773  !       not on the kernel stack when the Execute Check Results code is
;          0774  !       executed, Execute returns with a failure code, and the test is
;          0775  !       aborted.  If Actual_Except is not Resv_Opr_Abort, Print_Err is
;          0776  !       called.
;          0777  !
;          0778  ! ERRORS:
;          0779  !
;          0780  !       No exception - The implementation does not perform this check.
;          0781  !       Unexpected Exception - Wrong exception is getting priority.
;          0782  !       Did not Switch to Kernel stack - see subtest 1.
;          0783  !
;          0784  ! DEBUG:
;          0785  !
;          0786  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
```

```
;          0787  !
;          0788  !-
;          0789      $DS_BGNSUB
;          0790  !
;          0791  !    MBZ2 not equal to zero should get a reserved operand abort
;          0792  !
;          0793          CHECK_STACK = 3;        ! Start on interrupt, end on kernel, LDPCTX
;          0794          MODE = 0;               ! Kernel mode
;          0795          ACTUAL_EXCEPT = NONE;                   ! Clear exceptions
;          0796          EXPECTED_EXCEPT = RESV_OPR;             ! Expect a resv_opr_abort
;          0797  !
;          0798  !       Load in typical data with corrupted MBZ2
;          0799  !
;          0800          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;          0801          INIT_PCB [PTX$V_MBZ2] = - 1;    ! Fill Must Be Zero field with ones
;          0802  !
;          0803  !       The Pcb should not be touched on a Ldpctx
;          0804  !
;          0805          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;          0806          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);   !Init Actual_Pcb with
;          0807                                                          ! corrupt MBZ2 field
;          0808  !
;          0809  !       Initialize Actual_Reg with background 'Z's from Init_reg
;          0810  !
;          0811          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
;          0812  !
;          0813  !       The registers should not load all of the data
;          0814  !
;          0815          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;          0816
;          0817          IF NOT EXECUTE () THEN
;          0818              $DS_ABORT (ARG = PROGRAM);
;          0819  !
;          0820  !       The contents of the regsters after Resv_opr_abort are unknown
;          0821  !       so do not run Compare_pcb
;          0822  !
;          0823          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
;          0824              BEGIN
;        P 0825              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;        P 0826                           PRLINK = PRINT_ERR,
;        P 0827                           P1 = SUB_MSG_3,
;        P 0828                           P2 = .EXPECTED_EXCEPT,
;        L 0829                           P3 = .ACTUAL_EXCEPT);
; %PRINT:       Test 11, Subtest 3, Error 4
;          0830                END;
;          0831      $DS_ENDSUB;
;          0832
;          0833  !+
;          0834  ! SUBTEST DESCRIPTION:
;          0835  !       Subtest 4
;          0836  !       Reserved Operand Abort test, Astlevel reserved to Digital.
;          0837  !       Execute off of kernel stack.
;          0838  !
;          0839  ! SUBTEST STEPS:
;          0840  !
;          0841  !       Initialize variables as in subtest 1, set Actual_Pcb [Astlevel]
;          0842  .       to five, a value reserved to Digital. Set Check_Stack to start
```

```
;       0843  !       Execution from the Kernel stack. If the check is included in
;       0844  !       the implementation tested, a reserved operand abort should call
;       0845  !       DS as in subtest 1, when an attempt is made to load in an illegal
;       0846  !       value for Astlevel. If the check is not included, no exception
;       0847  !       should appear in Actual_Except. The data should not be checked
;       0848  !       as it is in an unknown state.
;       0849  !
;       0850  ! ERRORS:
;       0851  !
;       0852  !       Stray from kernel stack - stack corrupted
;       0853  !       No exception - reserved operand check not preformed for Astlevel
;       0854  !       Unexpected Exception - Wrong exception took precedence over
;       0855  !               reserved operand abort.
;       0856  !
;       0857  ! DEBUG:
;       0858  !
;       0859  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
;       0860  !
;       0861  !-
;       0862      $DS_BGNSUB
;       0863  !
;       0864  !    Reserved ASTLVL should get a reserved operand abort
;       0865  !
;       0866          CHECK_STACK = 2;        ! Start on kernel, end on kernel stack, LDPCTX
;       0867          MODE = 0;               ! Kernel mode
;       0868          ACTUAL_EXCEPT = NONE;                   ! Clear exceptions
;       0869          EXPECTED_EXCEPT = RESV_OPR;             ! Expect a resv_opr_abort
;       0870  !
;       0871  !       Use typical data with a oorrupt Astlevel
;       0872  !
;       0873          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;       0874          INIT_PCB [PTX$V_ASTLVL] = 5;    ! ASTLVL reserved to Digital
;       0875  !
;       0876  !       PCB should not be altered
;       0877  !
;       0878          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;       0879          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);   ! PCB to be loaded in
;       0880                                                          ! has an illegal
;       0881                                                          ! Astlevel
;       0882  !
;       0883  !       Fill Actual_Reg with background from Init_reg
;       0884  !
;       0885          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
;       0886  !
;       0887  !       Registers should not get all of INIT_pcb loaded in
;       0888  !
;       0889          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;       0890
;       0891          IF NOT EXECUTE () THEN
;       0892              $DS_ABORT (ARG = PROGRAM);
;       0893  !
;       0894  !       Do not compare unknown data received
;       0895  !
;       0896          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
;       0897              BEGIN
;     P 0898              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;     P 0899                           PRLINK = PRINT_ERR,
```

```
;       P 0900                          P1 = SUB_MSG_4,
;       P 0901                          P2 = .EXPECTED_EXCEPT,
;       L 0902                          P3 = .ACTUAL_EXCEPT);
; %PRINT:       Test 11, Subtest 4, Error 5
;         0903              END;
;         0904      $DS_ENDSUB;
;         0905
;         0906  !+
;         0907  ! SUBTEST DESCRIPTION:
;         0908  !       Subtest 5
;         0909  !       Test for Reserved Operand Abort, P1 Base Register out of bounds,
;         0910  !       start execution of LDPCTX from the kernel stack.
;         0911  !
;         0912  ! SUBTEST STEPS:
;         0913  !
;         0914  !       Initialize as in subtest 5. Set P1 region Base Register field in
;         0915  !       Actual_Pcb to zero, set Check_Stack to start on the kernel stack.
;         0916  !       LDPCTX will load Actual_Pcb into the registers until it comes to
;         0917  !       the P1BR field, then a Reserved Operand Abort exception will occur
;         0918  !       and control will go to the exception handler in this module.
;         0919  !       Actual_Except will be set to Reserved Operand Abort.  Control goes
;         0920  !       to the
;         0921  !
;         0922  ! ERRORS:
;         0923  !
;         0924  !       Unexpected exception - exception other than Resv_Op or Priv_Instr.
;         0925  !       Stray from Kernel stack - stack corrupted
;         0926  !       Wrong exception - Priv_Instr or no exception, Resv_Op expected
;         0927  !
;         0928  ! DEBUG:
;         0929  !
;         0930  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
;         0931  !
;         0932  !-
;         0933     $DS_BGNSUB
;         0934  !
;         0935  !    P1 Base Register out of bounds, should get a reserved operand abort
;         0936  !
;         0937          CHECK_STACK = 2;          ! Start on kernel, end on kernel stack, LDPCTX
;         0938          MODE = 0;                 ! Kernel mode
;         0939          ACTUAL_EXCEPT = NONE;                     ! Clear exceptions
;         0940          EXPECTED_EXCEPT = RESV_OPR;               ! Expect resv_opr_abort
;         0941  .
;         0942  .       Use with typical test pattern, and corrupt P1BR
;         0943  .
;         0944          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;         0945          (INIT_PCB [PTX$L_P1BR])<30,2> = 1;        ! Invalid value in P1BR, keep bit one clear
;         0946  !
;         0947  .       LDPCTX does not alter Pcb it loads in
;         0948  .
;         0949          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;         0950          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);    ! Init Pcb to be
;         0951                                                           ! loaded with bad P1BR
;         0952  !
;         0953  !       Initialize Actual_Reg with 'Z's from Init_reg
;         0954  .
;         0955          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
```

```
;          0956  !
;          0957  !       Resulting registers are unknown and may contain some data
;          0958  !
;          0959          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;          0960
;          0961          IF NOT EXECUTE () THEN
;          0962              $DS_ABORT (ARG = PROGRAM);
;          0963  !
;          0964  !       Don't check unknown results, just want to make sure got the right
;          0965  !       exceptions
;          0966  .
;          0967          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
;          0968              BEGIN
;        P 0969              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;        P 0970                           PRLINK = PRINT_ERR,
;        P 0971                           P1 = SUB_MSG_5,
;        P 0972                           P2 = .EXPECTED_EXCEPT,
;        L 0973                           P3 = .ACTUAL_EXCEPT);
; %PRINT:          Test 11, Subtest 5, Error 6
;          0974              END;
;          0975      $DS_ENDSUB;
;          0976
;          0977  !+
;          0978  ! SUBTEST DESCRIPTION:
;          0979  !       Subtest 6
;          0980  !       Test Reserved Operand Abort, invalid P1BR - bit 0 set. Start on
;          0981  !       interrupt stack.
;          0982  !
;          0983  ! SUBTEST STEPS:
;          0984  !
;          0985  !       Initialize as in subtest 1, set Actual_Pcb [P1BR] bit 0 to one, this
;          0986  !       is a Must Be Zero field. The subtest will call Execute and LDPCTX
;          0987  !       will execute normally until it tries to load in the P1 Base Register
;          0988  !       value. A reserved operand abort will occur, the handler will catch the
;          0989  !       exception and record it in Actual_except. A check will be made to be
;          0990  !       sure the switch from interrupt to kernel stack was made correctly.
;          0991  !       Control returns to the subtest and a check is made on the exception
;          0992  !       received vs. the expected exception.
;          0993  !
;          0994  ! ERRORS:
;          0995  !
;          0996  !       Unexpected exception - LDPCTX got an exception other than reserved
;          0997  !               operand abort or privileged instruction fault.
;          0998  !       Incorrect switch to the kernel stack- see subtest 1.
;          0999  !       Wrong exception - Got Privileged Instruction or no exception when
;          1000  !               should have gotten Reserved Operand Abort.
;          1001  !
;          1002  ! DEBUG:
;          1003  !
;          1004  !       *** HELPFUL HINTS FOR TRACKING HARDWARE FAULTS ***
;          1005  !
;          1006  !-
;          1007      $DS_BGNSUB
;          1008  !
;          1009  !    P1 Base Register invalid, bit 0 set; Reserved operand abort
;          1010  !
;          1011          CHECK_STACK = 3;        ! Start on interrupt stack, end on kernel,
```

```
;          1012                                 ! LDPCTX
;          1013          MODE = 0;              ! Kernel mode
;          1014          ACTUAL_EXCEPT = NONE;                  ! Clear exceptions
;          1015          EXPECTED_EXCEPT = RESV_OPR;            ! Expect resv_opr_abort
;          1016  !
;          1017  !       Use normal test pattern with P1BR invalid
;          1018  !
;          1019          CH$MOVE (PTX$K_LENGTH, REG_DATA, INIT_PCB);
;          1020          (INIT_PCB [PTX$L_P1BR])<0,1> = 1 ;             ! Set 0 bit
;          1021  !
;          1022  !       Ldpctx does not alter Pcb loaded
;          1023  !
;          1024          CH$MOVE (PTX$K_LENGTH, INIT_PCB, EXPECT_PCB);
;          1025          CH$MOVE (PTX$K_LENGTH, INIT_PCB, ACTUAL_PCB);  ! Init Pcb to be
;          1026                                                         ! loaded with invalid
;          1027                                                         ! P1BR
;          1028  !
;          1029  !       Initialize Actual_Reg with background from Init_reg
;          1030  !
;          1031          CH$MOVE (PTX$K_LENGTH, INIT_REG, ACTUAL_REG);
;          1032  !
;          1033  !       The registers resulting from Ldpctx execution are unknown and may or
;          1034  !       may not contain data
;          1035  !
;          1036          CH$MOVE (PTX$K_LENGTH, INIT_REG, EXPECT_REG);
;          1037
;          1038          IF NOT EXECUTE () THEN
;          1039              $DS_ABORT (ARG = PROGRAM);
;          1040  !
;          1041  !       Do not check unknown results
;          1042  !
;          1043          IF (.EXPECTED_EXCEPT NEQ .ACTUAL_EXCEPT) THEN
;          1044              BEGIN
;        P 1045              $DS_ERRHARD (MSGADR = FAIL_LD_M,
;        P 1046                           PRLINK = PRINT_ERR,
;        P 1047                           P1 = SUB_MSG_6,
;        P 1048                           P2 = .EXPECTED_EXCEPT,
;        L 1049                           P3 = .ACTUAL_EXCEPT);
; %PRINT:          Test 11, Subtest 6, Error 7
;          1050                END;
;          1051      $DS_ENDSUB;
;          1052
;          1053      MTPR (ORIG_PCBB, PR$_PCBB); ! Restore original PCB
;          1054      SS$_NORMAL
;          1055      END;                        ! To the Initial Exception handler, goes to


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

74  75  68  73  20  6F  74  20  65  6C  62  61  6E  55  25  002C6 P.AAV:  .ASCII  \%Unable to shut off memory management.\    ;
6E  61  6D  20  79  72  6F  6D  65  6D  20  66  66  6F  20  002D5                                                            ;
                        2E  74  6E  65  6D  65  67  61  002E4                                                            ;

                                                                .PSECT  $OWN$,NOEXE,2

                    000A5A5A  8A5A5A58  040A5A5A  8A5A5A58  00000 INIT_MNGT:
```

```
                                                        .LONG   -1973790120, 67787354, -1973790120, 678490  ;

                                                        .EXTRN  DS$ABORT, DS$MMOFF
                                                        .EXTRN  DS$ERRHARD, DS$BGNSUB
                                                        .EXTRN  DS$ENDSUB

                                                        .PSECT  CODE,NOWRT,2

                                        0FFC 00000      .ENTRY  TEST_CONTROL, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0465
                                                                R10,R11
                    5B 00000000G    9F  9E 00002         MOVAB   a#DS$BGNSUB, R11                            ;
                    5A 00000000G    9F  9E 00009         MOVAB   a#DS$ERRHARD, R10                           ;
                    59 00000000G    9F  9E 00010         MOVAB   a#DS$ABORT, R9                              ;
                    58      0000'   CF  9E 00017         MOVAB   NONE, R8                                    ;
                    57      0000'   CF  9E 0001C         MOVAB   INIT_PCB, R7                                ;
          03 0000FE03   9F              04  E1 00021      BBC     #4, a#^X0000FE03, 1$                        ; 0489
                        69              00  FB 00029      CALLS   #0, DS$ABORT                                ; 0490
             00000000G  9F              00  FB 0002C 1$:  CALLS   #0, a#DS$MMOFF                             ; 0497
                        17              50  E8 00033      BLBS    STATUS, 2$                                  ; 0498
                                        7E  7C 00036      CLRQ    -(SP)                                       ; 0501
                                        7E  7C 00038      CLRQ    -(SP)                                       ;
                                        7E  7C 0003A      CLRQ    -(SP)                                       ;
                                FCDE    CF  9F 0003C      PUSHAB  PRINT_ERR                                   ;
                                02C6    C8  9F 00040      PUSHAB  P.AAV                                       ;
                    7E                  01  7D 00044      MOVQ    #1, -(SP)                                   ;
                    6A                  0A  FB 00047      CALLS   #10, DS$ERRHARD                             ;
                    69                  00  FB 0004A      CALLS   #0, DS$ABORT                                ; 0502
                                    A4  A7  7C 0004D 2$:  CLRQ    REG_DATA+4                                  ; 0509
                                    AC  A7  D4 00050      CLRL    REG_DATA+12                                 ; 0511
                                        50  7D 00053      MOVQ    #4, I                                       ; 0518
                                A0 A740     81  7E 00056 3$:  MOVAQ   (DATA)+, REG_DATA[I]                    ; 0523
                                    51 11111109 E1  9E 0005B      MOVAB   286331145(R1), DATA                 ; 0524
                            F0          50  11  F3 00062      AOBLEQ  #17, I, 3$                          ; 0518
                                F0      A7  08  DB 00066      MFPR    #8, REG_DATA+80                     ; 0532
                                        50  09  DB 0006A      MFPR    #9, TEMP_LONGWORD                   ; 0534
          F4    A7                  16  00  50  F0 0006D      INSV    TEMP_LONGWORD, #0, #22, REG_DATA+84 ; 0536
                                        50  13  DB 00073      MFPR    #19, TEMP_LONGWORD                  ; 0539
          F7    A7                  03  00  50  F0 00076      INSV    TEMP_LONGWORD, #0, #3, REG_DATA+87  ; 0540
                                F8      A7  0A  DB 0007C      MFPR    #10, REG_DATA+88                    ; 0542
                                        50  0B  DB 00080      MFPR    #11, TEMP_LONGWORD                  ; 0543
          FC    A7                  16  00  50  F0 00083      INSV    TEMP_LONGWORD, #0, #22, REG_DATA+92 ; 0544
                                        50  3D  DB 00089      MFPR    #61, TEMP_LONGWORD                  ; 0545
          FF    A7                  01  07  50  F0 0008C      INSV    TEMP_LONGWORD, #7, #1, REG_DATA+95  ; 0549
        0060    8F          5A      8F  6E  00  2C 00092      MOVC5   #0, (SP), #90, #96, INIT_PCB        ; 0559
                                            67      0009A                                                   ;
                                        56  10  DB 0009B      MFPR    #16, ORIG_PCBB                      ; 0584
                                        50      C7  9E 0009E  MOVAB   ACTUAL_PCB, R0                      ; 0585
                                        10  50  DA 000A3      MTPR    R0, #16                             ;
                            50      A7      CF  20  28 000A6  MOVC3   #32, INIT_MNGT, INIT_PCB+80         ; 0594
                        0120        C7      67      0060  8F  28 000AD  MOVC3   #96, INIT_PCB, INIT_REG     ; 0595
                                            01  DD 000B5      PUSHL   #1                                  ;
                                            0B  DD 000B7      PUSHL   #11                                 ;
                                        6B  02  FB 000B9      CALLS   #2, DS$BGNSUB                       ;
                                    90  A7      03  B0 000BC  MOVW    #3, CHECK_STACK                     ; 0644
                                    88  A7      68  9E 000C0  MOVAB   NONE, ACTUAL_EXCEPT                 ; 0595
                                    8C  A7  06  A8  9E 000C4  MOVAB   RESV_OPR, EXPECTED_EXCEPT           ;
                            67      A0  A7      0060  8F  28 000C9  MOVC3   #96, REG_DATA, INIT_PCB        ; 0652
```

```
      53   A7                02                06                01  F0 000D0          INSV    #1, #6, #2, INIT_PCB+83                          ; 0653
                        60    A7                67          0060  8F  28 000D6          MOVC3   #96, INIT_PCB, EXPECT_PCB                        ; 0658
                      00C0    C7                67          0060  8F  28 000DD          MOVC3   #96, INIT_PCB, ACTUAL_PCB                        ; 0659
                      01E0    C7          0120  C7          0060  8F  28 000E5          MOVC3   #96, INIT_REG, ACTUAL_REG                        ; 0663
                      0180    C7          0120  C7          0060  8F  28 000EF          MOVC3   #96, INIT_REG, EXPECT_REG                        ; 0667
                                                      00000000G   EF  16 000F9          JSB     EXECUTE                                          ; 0669
                                                03                50  E8 000FF          BLBS    R0, 4$                                           ;
                                                69                00  FB 00102        4$: CALLS   #0, DS$ABORT                                     ; 0670
                                          88    A7          8C    A7  D1 00105          CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT                   ; 0674
                                                                    1A  13 0010A          BEQL    5$                                               ;
                                                                    7E  7C 0010C          CLRQ    -(SP)                                            ; 0680
                                                                    7E  D4 0010E          CLRL    -(SP)                                            ;
                                                              88    A7  DD 00110          PUSHL   ACTUAL_EXCEPT                                    ;
                                                              8C    A7  DD 00113          PUSHL   EXPECTED_EXCEPT                                  ;
                                                              2E    A8  9F 00116          PUSHAB  SUB_MSG_T                                        ;
                                                            FC01    CF  9F 00119          PUSHAB  PRINT_ERR                                        ;
                                                              1E    A8  9F 0011D          PUSHAB  FAIL_LD_M                                        ;
                                                7E                  02  7D 00120          MOVQ    #2, -(SP)                                        ;
                                                6A                  0A  FB 00123          CALLS   #10, DS$ERRHARD                                  ;
                                                                    01  DD 00126        5$: PUSHL   #1                                               ; 0681
                                                                    0B  DD 00128          PUSHL   #11                                              ;
                                    00000000G   9F                  02  FB 0012A          CALLS   #2, @#DS$ENDSUB                                  ;
                                                                    02  DD 00131          PUSHL   #2                                               ; 0682
                                                                    0B  DD 00133          PUSHL   #11                                              ;
                                                6B                  02  FB 00135          CALLS   #2, DS$BGNSUB                                    ;
                                          90    A7                  03  B0 00138          MOVW    #3, CHECK_STACK                                  ; 0719
                                          88    A7                  68  9E 0013C          MOVAB   NONE, ACTUAL_EXCEPT                              ; 0682
                                          8C    A7          06      A8  9E 00140          MOVAB   RESV_OPR, EXPECTED_EXCEPT                        ;
                              67          A0    A7          0060    8F  28 00145          MOVC3   #96, REG_DATA, INIT_PCB                          ; 0727
                                          50    A7                  01  88 0014C          BISB2   #1, INIT_PCB+80                                  ; 0728
                        60    A7                67          0060    8F  28 00150          MOVC3   #96, INIT_PCB, EXPECT_PCB                        ; 0732
                      00C0    C7                67          0060    8F  28 00157          MOVC3   #96, INIT_PCB, ACTUAL_PCB                        ; 0733
                      01E0    C7          0120  C7          0060    8F  28 0015F          MOVC3   #96, INIT_REG, ACTUAL_REG                        ; 0738
                      0180    C7          0120  C7          0060    8F  28 00169          MOVC3   #96, INIT_REG, EXPECT_REG                        ; 0742
                                                      00000000G   EF  16 00173          JSB     EXECUTE                                          ; 0744
                                                03                50  E8 00179          BLBS    R0, 6$                                           ;
                                                69                00  FB 0017C          CALLS   #0, DS$ABORT                                     ; 0745
                                          88    A7          8C    A7  D1 0017F        6$: CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT                   ; 0747
                                                                    1A  13 00184          BEQL    7$                                               ;
                                                                    7E  7C 00186          CLRQ    -(SP)                                            ; 0753
                                                                    7E  D4 00188          CLRL    -(SP)                                            ;
                                                              88    A7  DD 0018A          PUSHL   ACTUAL_EXCEPT                                    ;
                                                              8C    A7  DD 0018D          PUSHL   EXPECTED_EXCEPT                                  ;
                                                              57    A8  9F 00190          PUSHAB  SUB_MSG_2                                        ;
                                                            FB87    CF  9F 00193          PUSHAB  PRINT_ERR                                        ;
                                                              1E    A8  9F 00197          PUSHAB  FAIL_LD_M                                        ;
                                                7E                  03  7D 0019A          MOVQ    #3, -(SP)                                        ;
                                                6A                  0A  FB 0019D          CALLS   #10, DS$ERRHARD                                  ;
                                                                    02  DD 001A0        7$: PUSHL   #2                                               ; 0754
                                                                    0B  DD 001A2          PUSHL   #11                                              ;
                                    00000000G   9F                  02  FB 001A4          CALLS   #2, @#DS$ENDSUB                                  ;
                                                                    03  DD 001AB          PUSHL   #3                                               ; 0755
                                                                    0B  DD 001AD          PUSHL   #11                                              ;
                                                6B                  02  FB 001AF          CALLS   #2, DS$BGNSUB                                    ;
                                          90    A7                  03  B0 001B2          MOVW    #3, CHECK_STACK                                  ; 0793
                                          88    A7                  68  9E 001B6          MOVAB   NONE, ACTUAL_EXCEPT                              ; 0755
                                          8C    A7          06      A8  9E 001BA          MOVAB   RESV_OPR, EXPECTED_EXCEPT                        ;
```

```
                        67      A0      A7      0060    8F  28 001BF        MOVC3   #96, REG_DATA, INIT_PCB                 ; 0800
                                57      A7        F8    8F  88 001C6        BISB2   #248, INIT_PCB+87                       ; 0801
                    60  A7              67      0060    8F  28 001CB        MOVC3   #96, INIT_PCB, EXPECT_PCB               ; 0805
                  00C0  C7              67      0060    8F  28 001D2        MOVC3   #96, INIT_PCB, ACTUAL_PCB               ; 0806
                  01E0  C7      0120    C7      0060    8F  28 001DA        MOVC3   #96, INIT_REG, ACTUAL_REG               ; 0811
                  0180  C7      0120    C7      0060    8F  28 001E4        MOVC3   #96, INIT_REG, EXPECT_REG               ; 0815
                                        00000000G       EF  16 001EE        JSB     EXECUTE                                 ; 0817
                                        03              50  E8 001F4        BLBS    R0, 8$                                  ;
                                        69              00  FB 001F7        CALLS   #0, DS$ABORT                            ; 0818
                                88      A7        8C    A7  D1 001FA 8$:    CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT          ; 0823
                                                        1A  13 001FF        BEQL    9$                                      ;
                                                        7E  7C 00201        CLRQ    -(SP)                                   ; 0829
                                                        7E  D4 00203        CLRL    -(SP)                                   ;
                                                  88    A7  DD 00205        PUSHL   ACTUAL_EXCEPT                           ;
                                                  8C    A7  DD 00208        PUSHL   EXPECTED_EXCEPT                         ;
                                                  7F    A8  9F 0020B        PUSHAB  SUB_MSG_3                               ;
                                                FB0C    CF  9F 0020E        PUSHAB  PRINT_ERR                               ;
                                                  1E    A8  9F 00212        PUSHAB  FAIL_LD_M                               ;
                                        7E              04  7D 00215        MOVQ    #4, -(SP)                               ;
                                        6A              0A  FB 00218        CALLS   #10, DS$ERRHARD                         ;
                                                        03  DD 0021B 9$:    PUSHL   #3                                      ; 0830
                                                        0B  DD 0021D        PUSHL   #11                                     ;
                            00000000G   9F              02  FB 0021F        CALLS   #2, @#DS$ENDSUB                         ;
                                                        04  DD 00226        PUSHL   #4                                      ; 0831
                                                        0B  DD 00228        PUSHL   #11                                     ;
                                        6B              02  FB 0022A        CALLS   #2, DS$BGNSUB                           ;
                                90      A7              02  B0 0022D        MOVW    #2, CHECK_STACK                         ; 0866
                                88      A7              68  9E 00231        MOVAB   NONE, ACTUAL_EXCEPT                     ; 0831
                                8C      A7        06    A8  9E 00235        MOVAB   RESV_OPR, EXPECTED_EXCEPT               ;
                        67      A0      A7      0060    8F  28 0023A        MOVC3   #96, REG_DATA, INIT_PCB                 ; 0873
  57    A7                    03              00          05  F0 00241        INSV    #5, #0, #3, INIT_PCB+87                 ; 0874
                    60  A7              67      0060    8F  28 00247        MOVC3   #96, INIT_PCB, EXPECT_PCB               ; 0878
                  00C0  C7              67      0060    8F  28 0024E        MOVC3   #96, INIT_PCB, ACTUAL_PCB               ; 0879
                  01E0  C7      0120    C7      0060    8F  28 00256        MOVC3   #96, INIT_REG, ACTUAL_REG               ; 0885
                  0180  C7      0120    C7      0060    8F  28 00260        MOVC3   #96, INIT_REG, EXPECT_REG               ; 0889
                                        00000000G       EF  16 0026A        JSB     EXECUTE                                 ; 0891
                                        03              50  E8 00270        BLBS    R0, 10$                                 ;
                                        69              00  FB 00273        CALLS   #0, DS$ABORT                            ; 0892
                                88      A7        8C    A7  D1 00276 10$:   CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT          ; 0896
                                                        1B  13 0027B        BEQL    11$                                     ;
                                                        7E  7C 0027D        CLRQ    -(SP)                                   ; 0902
                                                        7E  D4 0027F        CLRL    -(SP)                                   ;
                                                  88    A7  DD 00281        PUSHL   ACTUAL_EXCEPT                           ;
                                                  8C    A7  DD 00284        PUSHL   EXPECTED_EXCEPT                         ;
                                                00A7    C8  9F 00287        PUSHAB  SUB_MSG_4                               ;
                                                FA8F    CF  9F 0028B        PUSHAB  PRINT_ERR                               ;
                                                  1E    A8  9F 0028F        PUSHAB  FAIL_LD_M                               ;
                                        7E              05  7D 00292        MOVQ    #5, -(SP)                               ;
                                        6A              0A  FB 00295        CALLS   #10, DS$ERRHARD                         ;
                                                        04  DD 00298 11$:   PUSHL   #4                                      ; 0903
                                                        0B  DD 0029A        PUSHL   #11                                     ;
                            00000000G   9F              02  FB 0029C        CALLS   #2, @#DS$ENDSUB                         ;
                                                        05  DD 002A3        PUSHL   #5                                      ; 0904
                                                        0B  DD 002A5        PUSHL   #11                                     ;
                                        6B              02  FB 002A7        CALLS   #2, DS$BGNSUB                           ;
                                90      A7              02  B0 002AA        MOVW    #2, CHECK_STACK                         ; 0937
                                88      A7              68  9E 002AE        MOVAB   NONE, ACTUAL_EXCEPT                     ; 0904
```

```
                                8C   A7      06    A8  9E 002B2          MOVAB   RESV_OPR, EXPECTED_EXCEPT                ;
                          67    A0   A7    0060    8F  28 002B7          MOVC3   #96, REG_DATA, INIT_PCB                  ; 0944
          5B   A7               02         06      01  F0 002BE          INSV    #1, #6, #2, INIT_PCB+91                 ; 0945
                 60   A7                    67    0060  8F  28 002C4          MOVC3   #96, INIT_PCB, EXPECT_PCB               ; 0949
               00C0   C7                    67    0060  8F  28 002CB          MOVC3   #96, INIT_PCB, ACTUAL_PCB               ; 0950
               01E0   C7          0120      C7    0060  8F  28 002D3          MOVC3   #96, INIT_REG, ACTUAL_REG               ; 0955
               0180   C7          0120      C7    0060  8F  28 002DD          MOVC3   #96, INIT_REG, EXPECT_REG               ; 0959
                                             00000000G   EF  16 002E7          JSB     EXECUTE                                 ; 0961
                                       03                50  E8 002ED          BLBS    R0, 12$                                 ;
                                       69                00  FB 002F0          CALLS   #0, DS$ABORT                            ; 0962
                                8B   A7      8C    A7    D1  002F3 12$:       CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT          ; 0967
                                             1B    13    002F8              BEQL    13$                                     ;
                                             7E    7C    002FA              CLRQ    -(SP)                                   ; 0973
                                             7E    D4    002FC              CLRL    -(SP)                                   ;
                                          88   A7  DD    002FE              PUSHL   ACTUAL_EXCEPT                           ;
                                          8C   A7  DD    00301              PUSHL   EXPECTED_EXCEPT                         ;
                                        00CA   C8  9F    00304              PUSHAB  SUB_MSG_5                               ;
                                        FA12   CF  9F    00308              PUSHAB  PRINT_ERR                               ;
                                          1E   A8  9F    0030C              PUSHAB  FAIL_LD_M                               ;
                                       7E                06  7D  0030F          MOVQ    #6, -(SP)                               ;
                                       6A                0A  FB  00312          CALLS   #10, DS$ERRHARD                         ;
                                                         05  DD  00315 13$:     PUSHL   #5                                      ; 0974
                                                         0B  DD  00317          PUSHL   #11                                     ;
                                00000000G   9F           02  FB  00319          CALLS   #2, a#DS$ENDSUB                         ;
                                                         06  DD  00320          PUSHL   #6                                      ; 0975
                                                         0B  DD  00322          PUSHL   #11                                     ;
                                       6B                02  FB  00324          CALLS   #2, DS$BGNSUB                           ;
                                90   A7                  03  B0  00327          MOVW    #3, CHECK_STACK                         ; 1011
                                88   A7                  68  9E  0032B          MOVAB   NONE, ACTUAL_EXCEPT                     ; 0975
                                8C   A7      06          A8  9E  0032F          MOVAB   RESV_OPR, EXPECTED_EXCEPT               ;
                          67    A0   A7    0060          8F  28  00334          MOVC3   #96, REG_DATA, INIT_PCB                 ; 1019
                                58   A7                  01  88  0033B          BISB2   #1, INIT_PCB+88                         ; 1020
                 60   A7                    67    0060  8F  28  0033F          MOVC3   #96, INIT_PCB, EXPECT_PCB               ; 1024
               00C0   C7                    67    0060  8F  28  00346          MOVC3   #96, INIT_PCB, ACTUAL_PCB               ; 1025
               01E0   C7          0120      C7    0060  8F  28  0034E          MOVC3   #96, INIT_REG, ACTUAL_REG               ; 1031
               0180   C7          0120      C7    0060  8F  28  00358          MOVC3   #96, INIT_REG, EXPECT_REG               ; 1036
                                             00000000G   EF  16  00362          JSB     EXECUTE                                 ; 1038
                                       03                50  E8  00368          BLBS    R0, 14$                                 ;
                                       69                00  FB  0036B          CALLS   #0, DS$ABORT                            ; 1039
                                88   A7      8C    A7    D1  0036E 14$:       CMPL    EXPECTED_EXCEPT, ACTUAL_EXCEPT          ; 1043
                                             1B    13    00373              BEQL    15$                                     ;
                                             7E    7C    00375              CLRQ    -(SP)                                   ; 1049
                                             7E    D4    00377              CLRL    -(SP)                                   ;
                                          88   A7  DD    00379              PUSHL   ACTUAL_EXCEPT                           ;
                                          8C   A7  DD    0037C              PUSHL   EXPECTED_EXCEPT                         ;
                                        00F0   C8  9F    0037F              PUSHAB  SUB_MSG_6                               ;
                                        F997   CF  9F    00383              PUSHAB  PRINT_ERR                               ;
                                          1E   A8  9F    00387              PUSHAB  FAIL_LD_M                               ;
                                       7E                07  7D  0038A          MOVQ    #7, -(SP)                               ;
                                       6A                0A  FB  0038D          CALLS   #10, DS$ERRHARD                         ;
                                                         06  DD  00390 15$:     PUSHL   #6                                      ; 1050
                                                         0B  DD  00392          PUSHL   #11                                     ;
                                00000000G   9F           02  FB  00394          CALLS   #2, a#DS$ENDSUB                         ;
                                       10                56  DA  0039B          MTPR    ORIG_PCBB, #16                          ; 1053
                                       50                01  D0  0039E          MOVL    #1, R0                                  ; 0465
                                                             04  003A1          RET                                             ;
```

```
; Routine Size:  930 bytes,    Routine Base:  CODE + 02E2

;         1056                                    . $DS_ENDTEST
;         1057  $DS_ENDMOD;
;         1058  END
;         1059  ELUDOM


                                                                .PSECT  $TSTCNT,NOWRT,NOEXE,  OVR,2

                                                00000000B  00000 L_TSTCNT:
                                                                .LONG   11                                              ;


;                                       PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  $PLIT$                                748  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  GLOBALS                               704  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  CODE                                 1668  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                                  16  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $TSTCNT                                 4  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  OVR,NOPIC,ALIGN(2)



;                                   LIBRARY STATISTICS
;
;                                                 -------- Symbols --------     Blocks
;       File                                      Total    Loaded   Percent      Read
;
;  SYS$SYSROOT:[SYSLIB]LIB.L32;4                  11296        13         0       414
;  SYS$SYSROOT:[SYSLIB]DIAG.L32;234                 761        15         1        46



;                                       COMMAND QUALIFIERS

;       BLISS /LIS ENKAXK

; Size:         1668 code + 1472 data bytes
; Run Time:       00:28.4
; Elapsed Time:   00:51.9
; Memory Used:  669 pages
; Compilation Complete
```

Table of contents

```
0000     1
0000     2 .TITLE ENKAXL Ldpctx Execute
0000     3 .IDENT  /1-3/
0000     4 .LIST   MEB
0000     5 .NLIST  CND
0000     6 .DEFAULT DISPLACEMENT, LONG     ;***  CHANGE THIS TO LONG FOR DEBUG ***
0000     7
0000     8 ;++
0000     9 ; COPYRIGHT (C) 1982
0000    10 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000    11 ;
0000    12 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000    13 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000    14 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    15 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    16 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    17 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    18 ; REMAIN IN DEC.
0000    19 ;
0000    20 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    21 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    22 ; CORPORATION.
0000    23 ;
0000    24 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    25 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    26 ;++
0000    27
0000    28 ;++
0000    29 ; FACILITY:     VAX DIAGNOSTIC.
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ; The Load Process Context (LDPCTX) instruction test is part of the
0000    33 ; Privileged Instuction Set , which tests the valadity of data being
0000    34 ; loaded into the Process Control Block during a LDPCTX operation,
0000    35 ; according to its definition in the System Reference Manual.
0000    36 ;
0000    37 ;
0000    38 ; ENVIRONMENT:  VAX DIAGNOSTIC SUPERVISOR.
0000    39 ;
0000    40 ; AUTHOR:       Kevin Martin            9-NOV-1982      Version 1
0000    41 ;
0000    42 ;
0000    43 ; MODIFIED BY:
0000    44 ;
0000    45 ;               Tai-Chun Pan    01-Apr-83       Version 1.2
0000    46 ;
0000    47 ;               Added quick flag on Test 7(TU58). If quick flag is set
0000    48 ;               will skip subtest 4 through subtest 11.  Added event flag 3.
0000    49 ;               If event flag 3 is set, will override protection,
0000    50 ;               i.e., go ahead and write on tape.  If run APT & event flag 3
0000    51 ;               is clear & tape is not scratch, will report an error.
0000    52 ;               Modified Unibus sizer on Test 8 & 9. make test run faster.
0000    53 ;
0000    54 ;
0000    55 ;               Tai-Chun Pan    03-Aug-83       Version 1.3
0000    56 ;
0000    57 ;               fixed bugs on TEST 7 and error summary routine call when
```

```
                                0000     58 ;                   running under APT.
                                0000     59 ;
                                0000     60 ;--
```

```
                     62          .SBTTL  DECLARATIONS
                     63
                     64 .LIBRARY         \SYS$LIBRARY:LIB\
                     65 .LIBRARY         \SYS$LIBRARY:DIAG\      ; Vax Family Diagnostic Library
                     66 .LIBRARY         \ENKAX\                 ; Library used by Enkax modules
                     67
                     68 ;+
                     69 ; MACROS:
                     70 ;-
                     74          $DS_DSADEF
                     75          $DS_SCBDEF
                     76          $PRDEF
                     77          $PSLDEF
                     78          ENKAX_SECDEF
                     79 ;
                     80 ; User defined macros
                     81 ;
                     82
                     83          $DEFINI     PTX
                     84                  $DEF     PTX$L_KSP, .BLKL, 1
0004                 85                  $DEF     PTX$L_ESP, .BLKL,1
0008                 86                  $DEF     PTX$L_SSP, .BLKL,1
000C                 87                  $DEF     PTX$L_USP, .BLKL,1
0010                 88                  $DEF     PTX$L_R0, .BLKL, 1
0014                 89                  $DEF     PTX$L_R1, .BLKL, 1
0018                 90                  $DEF     PTX$L_R2, .BLKL, 1
001C                 91                  $DEF     PTX$L_R3, .BLKL, 1
0020                 92                  $DEF     PTX$L_R4, .BLKL, 1
0024                 93                  $DEF     PTX$L_R5, .BLKL, 1
0028                 94                  $DEF     PTX$L_R6, .BLKL, 1
002C                 95                  $DEF     PTX$L_R7, .BLKL, 1
0030                 96                  $DEF     PTX$L_R8, .BLKL, 1
0034                 97                  $DEF     PTX$L_R9, .BLKL, 1
0038                 98                  $DEF     PTX$L_R10, .BLKL,1
003C                 99                  $DEF     PTX$L_R11, .BLKL,1
0040                100                  $DEF     PTX$L_AP, .BLKL, 1
0044                101                  $DEF     PTX$L_FP, .BLKL, 1
0048                102                  $DEF     PTX$L_PC, .BLKL, 1
004C                103                  $DEF     PTX$L_PSL, .BLKL,1
0050                104                  $DEF     PTX$L_P0BR, .BLKL,1
0054                105                  $DEF     PTX$L_P0LRASTL, .BLKL,1
0058                106                  $VIELD   PTX,0, <-
0058                107                           <P0LR, 22>-
0058                108                           <MBZ1,2>-
0058                109                           <ASTLVL,3>-
0058                110                           <MBZ2,5>>
0058                111                  $DEF     PTX$L_P1BR, .BLKL,1
005C                112                  $DEF     PTX$L_P1LRPME, .BLKL,1
0060                113                  $VIELD   PTX,0, <-
0060                114                           <P1LR,22>-
0060                115                           <MBZ3,9>-
0060                116                           <PME,1>>
0060                117                  $DEF     PTX$K_SIZE
0060                118                  $DEFEND PTX
                    119
                    120 ;
                    121 ; Macros
```

```
                     0000   122 ;
                     0000   123 .MACRO  SAVE_R  PCB              ; Routine to call Save_regs with param PCB
                     0000   124         PUSHAB  PCB              ; Put PCB address on stack, don't
                     0000   125                                  ; disturb regs
                     0000   126         JSB     SAVE_REGS        ; Do the Save_regs routine
                     0000   127         ADDL2   #4, SP           ; Pop PCB address off the stack
                     0000   128 .ENDM   SAVE_R
                     0000   129
                     0000   130 ;
                     0000   131 ;
                     0000   132 ;
                     0000   133
                     0000   134 .MACRO  PUSH_R  PCB              ; Routine to call Push_regs with param PCB
                     0000   135         PUSHAB  PCB              ; Put PCB address on stack as parameter
                     0000   136         JSB     PUSH_REGS        ; Do Push_regs routine
                     0000   137         ADDL2   #4, SP           ; Clean off stack
                     0000   138 .ENDM   PUSH_R
                     0000   139
                     0000   140 ;
                     0000   141 ; Declarations
                     0000   142 ;
                     0000   143
                     0000   144 $DS_BGNMOD      ENV= CEP_REPAIR, TN = 11
                            ;;;     Macro-32 Diagnostic macro library Version 6.10 'DIAG.MLB (137)"
                     0000   145
                     0000   146 ;+
                     0000   147 ; PROGRAMMER DEFINED LOCAL AND GLOBAL STORAGE
                     0000   148 ;-
                     0000   149 ;
                     0000   150 ;*****************************************************************************
                     0000   151 ;
                     0000   152 ; EXTERNAL
                     0000   153 ;
                     0000   154 ; All from ENKAXK
                     0000   155 ;
                     0000   156 ;   HANDLER : WORD,            Address of Exception handler routine
                     0000   157 ;                               from ENKAXK
                     0000   158 ;   CHECK_STACK : BYTE         Should a check be made for stack corruption?
                     0000   159 ;
                     0000   160 ;   INIT_REG : PCB,            What the registers should contain
                     0000   161 ;                               before instruction execution
                     0000   162 ;   ACTUAL_REG : PCB,          The place where registers are saved
                     0000   163 ;                               after the instruction
                     0000   164 ;   EXPECT_REG : PCB,          What the registers should contain after
                     0000   165 ;                               the instruction
                     0000   166 ;   INIT_PCB : PCB,            What the pcb contains before the instruction
                     0000   167 ;   ACTUAL_PCB : PCB,          The place where the Pcb is saved after
                     0000   168 ;                               the instruction
                     0000   169 ;   ABORT : BYTE               If an error occurs on the stack, stop testing
                     0000   170 ;   OPR_ABORT: LONG            Flag to check for abort
                     0000   171 ;
                     0000   172 ;*****************************************************************************
                     0000   173 ;
                     0000   174 ; LOCAL ( For Execute )
                     0000   175 ;
                     0000   176
00000060             0000   177 TEMP_REGS:      .BLKB 96         ; Holds the contents of the registers
```

```
                                0060   178                             ; until Execute finishes its subtest
                    00000000    0060   179 TEMP_PC:          .LONG 0   ; Temporary PC pointer
                    00000000    0064   180 TEMP_PSL::        .LONG 0   ; Temporary process status longword
                00000000 00000000  0068   181 TEMP_R0:        .QUAD 0   ; Temporary storage for R0 and R1
                    00000000    0070   182 INIT_PSL:         .LONG 0   ; Used to initialize mode and stack for
                                0074   183                             ; each subtest
                    00000000    0074   184 INIT_PC:          .LONG 0   ; Determines which instruction will be
                                0078   185                             ; executed
                    00000000    0078   186 INST_SP:          .LONG 0   ; The SP the old stack is set to if there
                                007C   187                             ; is a stack switch during the instruction
                          00    007C   188 NULL:             .BYTE 0   ; For null messages in DS_Errhard
                    00000000    007D   189 TEMP_LONGWORD:    .LONG 0   ; Hold fill characters
                    00000000    0081   190 PUSH_PC:          .LONG 0   ; Hold pushed PC and PSL
                    00000000    0085   191 PUSH_PSL:         .LONG 0   ; after LDPCTX.
                2E 65 6E 6F 4E 00'  0089   192 NONE: .ASCIC 'None.'    ; Used to check against expected exception
                          05    0089
                    00000000    008F   193 TEMP_SV_PC:       .LONG 0   ; Hold PC and PSL before it is popped
                    00000000    0093   194 TEMP_SV_PSL:      .LONG 0   ; off by SVPCTX
                    00000000    0097   195 TEMP_ESP:         .LONG 0   ; Hold executive stack address
                    00000000    009B   196 TEMP_SSP:         .LONG 0   ; Hold supervisor stack address
                    00000000    009F   197 TEMP_USP:         .LONG 0   ; Hold user stack address
                                00A3   198
                                00A3   199 ;
                                00A3   200 ;    ASCIC INFO AND ERROR MESSAGES
                                00A3   201 ;
                                00A3   202
                                00A3   203
50 44 4C 20 4E 4F 20 4C 49 41 46 00'  00A3   204 CHK_1: .ASCIC 'FAIL ON LDPCTX, stray from kernel stack.'
66 20 79 61 72 74 73 20 2C 58 54 43   00AF
73 20 6C 65 6E 72 65 6B 20 6D 6F 72   00BB
                     2E 6B 63 61 74   00C7
                                 28   00A3
73 20 74 70 75 72 72 65 74 6E 49 00'  00CC   205 CHK_2: .ASCIC 'Interrupt stack corrupted, wrong ISP.'
74 70 75 72 72 6F 63 20 6B 63 61 74   00D8
53 49 20 67 6E 6F 72 77 20 2C 64 65   00E4
                           2E 50      00F0
                                 25   00CC
65 68 63 74 69 77 73 20 74 6F 4E 00'  00F2   206 CHK_21: .ASCIC 'Not switched from Interrupt to Kernel stack.'
72 65 74 6E 49 20 6D 6F 72 66 20 64   00FE
6E 72 65 4B 20 6F 74 20 74 70 75 72   010A
         2E 6B 63 61 74 73 20 6C 65   0116
                                 2C   00F2
73 20 74 70 75 72 72 65 74 6E 49 00'  011F   207 CHK_PC: .ASCIC 'Interrupt stack corrupted,  wrong PC.'
74 70 75 72 72 6F 63 20 6B 63 61 74   012B
50 20 67 6E 6F 72 77 20 20 2C 64 65   0137
                           2E 43      0143
                                 25   011F
63 61 74 73 20 6C 65 6E 72 65 4B 00'  0145   208 CHK_PSL: .ASCIC 'Kernel stack corrupted, wrong PSL.'
2C 64 65 74 70 75 72 72 6F 63 20 6B   0151
   2E 4C 53 50 20 67 6E 6F 72 77 20   015D
                                 22   0145
                                0168   209
```

```
                                0168   211 .SBTTL  <Subroutines>
                                0168   212
                                0168   213 SAVE_REGS :
                                0168   214 ;
                                0168   215 ; This routine is called from a macro (Save_R) called from Execute, The
                                0168   216 ; macro accepts a PCB address as an argument and pushes it on the stack
                                0168   217 ; Saves the current Registers into the specified PCB. This subtest is
                                0168   218 ; called at the beginning of Execute to preserve the current registers,
                                0168   219 ; and again after the instruction under test is executed to store the
                                0168   220 ; register results in a PCB for later comparison with expected data.
                                0168   221 ; Save_Regs does not alter any of the registers when executed.
                                0168   222 ;
                                0168   223
                                0168   224
                      03   BB   0168   225         PUSHR   #^M<R0,R1>                      ; Save registers 0 & 1,
                                016A   226                                                 ; so we can overwrite
                   50   0C AE   D0   016A   227         MOVL    12(SP),R0                   ; Set up pointer to PCB,
                                016E   228                                                 ; below R1 in stack
                 10 A0   6E   D0   016E   229         MOVL    (SP), PTX$L_R0 (R0)         ; Move saved R0 into PCB's R0
              14 A0   04 AE   D0   0172   230         MOVL    4(SP), PTX$L_R1(R0)         ; Save R1 into PCB's R1
                                0177   231 ;
                                0177   232 ;       If there are no internal processer stack pointers, the next
                                0177   233 ;       four instructions will merely NOP. So fill up the space with
                                0177   234 ;       expected data, if the stack pointers exist as internal
                                0177   235 ;       registers then the MFPR will put the requested data in.
                                0177   236 ;
          00000000'EF   D0   0177   237         MOVL    EXPECT_REG + PTX$L_KSP,-        ; Fill with expected data for
                      60        017D   238                 PTX$L_KSP(R0)               ; a transparent possible NOP
                   60   00   DB   017E   239         MFPR    #PR$_KSP, PTX$L_KSP(R0)     ; Save internal registers
             00000000'EF   00   DB   0181   240         MFPR    #PR$_KSP, EXPECT_REG + -    ; Save Ksp value in Expect_Reg
                                0188   241                 PTX$L_KSP
          00000004'EF   D0   0188   242         MOVL    EXPECT_REG + PTX$L_ESP,-
                   04 A0        018E   243                 PTX$L_ESP(R0)
            04 A0   01   DB   0190   244         MFPR    #PR$_ESP, PTX$L_ESP(R0)
          00000008'EF   D0   0194   245         MOVL    EXPECT_REG + PTX$L_SSP,-
                   08 A0        019A   246                 PTX$L_SSP(R0)
            08 A0   02   DB   019C   247         MFPR    #PR$_SSP, PTX$L_SSP(R0)
          0000000C'EF   D0   01A0   248         MOVL    EXPECT_REG + PTX$L_USP,-
                   0C A0        01A6   249                 PTX$L_USP(R0)
            0C A0   03   DB   01A8   250         MFPR    #PR$_USP, PTX$L_USP(R0)
                                01AC   251
                3FFF 8F   BB   01AC   252         PUSHR   #^X3FFF                         ; Put regs not saved (R2-FP) in
                                01B0   253                                                 ; memory,and save R0 from being
                                01B0   254                                                 ; overwritten by MOVC3
       18 A0   08 AE   30   28   01B0   255         MOVC3   #4*12, 8(SP), PTX$L_R2(R0)      ; Copy into PCB specified
                3FFF 8F   BA   01B6   256         POPR    #^X3FFF                         ; Clear the stack after saved
                                01BA   257
                50 A0   08   DB   01BA   258         MFPR    #PR$_P0BR, PTX$L_P0BR(R0)       ; Save memory management
                54 A0   09   DB   01BE   259         MFPR    #PR$_P0LR, PTX$L_P0LRASTL(R0)   ; registers
                   51   13   DB   01C2   260         MFPR    #PR$_ASTLVL, R1                 ; Put ASTLVL data in a temp
             03    18   51   F0   01C5   261         INSV    R1, #PTX$V_ASTLVL, #PTX$S_ASTLVL,-
                      54 A0        01C9   262                 PTX$L_P0LRASTL(R0)          ; then put in longword
                58 A0   0A   DB   01CB   263         MFPR    #PR$_P1BR, PTX$L_P1BR(R0)
                5C A0   0B   DB   01CF   264         MFPR    #PR$_P1LR, PTX$L_P1LRPME(R0)
                   51   3D   DB   01D3   265         MFPR    #PR$_PME, R1                    ; Put PME in a temp
             01    1F   51   F0   01D6   266         INSV    R1, #PTX$V_PME, #PTX$S_PME,-    ; then put in longword
                      5C A0        01DA   267                 PTX$L_P1LRPME(R0)
```

```
                    03   BA  01DC   268          POPR    #^M<R0,R1>                      ; Pop all regs saved
                         05  01DE   269          RSB
                             01DF   270
                             01DF   271 ;
                             01DF   272 ; Loads data into the registers from a Pcb argument pushed on the stack
                             01DF   273 ;
                             01DF   274
                             01DF   275 PUSH_REGS:
                             01DF   276
                50   04 AE   D0  01DF   277          MOVL    4(SP),R0                ; Set up pointer to PCB
                             01E3   278 ;
                             01E3   279 ;       Push in expected data   If there are no Internal register
                             01E3   280 ;       stack pointers, the MTPR will be a NOP.  The data in that case
                             01E3   281 ;       might as well be correct
                             01E3   282 ;
                             01E3   283 ;       Don't change the Ksp register, otherwise end up in never-never
                             01E3   284 ;       land.
                             01E3   285 ;
                             01E3   286
        04 A0   00000004'EF  D0  01E3   287          MOVL    EXPECT_REG + PTX$L_ESP, -
                             01EB   288                  PTX$L_ESP(R0)
                01   04 A0   DA  01EB   289          MTPR    PTX$L_ESP(R0), #PR$_ESP
                             01EF   290
        08 A0   00000008'EF  D0  01EF   291          MOVL    EXPECT_REG + PTX$L_SSP, -
                             01F7   292                  PTX$L_SSP(R0)
                02   08 A0   DA  01F7   293          MTPR    PTX$L_SSP(R0), #PR$_SSP
                             01FB   294
        0C A0   0000000C'EF  D0  01FB   295          MOVL    EXPECT_REG + PTX$L_USP, -
                             0203   296                  PTX$L_USP(R0)
                03   0C A0   DA  0203   297          MTPR    PTX$L_USP(R0), #PR$_USP
                             0207   298
                   51   10   D0  0207   299          MOVL    #PTX$L_AP/4, R1         ; Set counter, do not change FP
                      6041   DD  020A   300 LOOP:     PUSHL   (R0)[R1]                ; Push the PCB's regs onto stack
FFF3 51   FFFFFFFF 8F   06   F1  020D   301          ACBL    #PTX$L_R2/4, #-1, R1, LOOP
                             0217   302                                              ; From AP to R2
                   1FFC 8F   BA  0217   303          POPR    #^X1FFC                 ; Pop the PCB defined regs into the
                             021B   304                                              ; current registers
                             021B   305
                08   50 A0   DA  021B   306          MTPR    PTX$L_P0BR(R0), #PR$_P0BR       ; Save memory management reg.
                   16   00   EF  021F   307          EXTZV   #PTX$V_P0LR, #PTX$S_P0LR,-      ; Put PCB defined P0LR
                51   54 A0        0222   308                  PTX$L_P0LRASTL(R0), R1          ; into R1
                   09   51   DA  0225   309          MTPR    R1, #PR$_P0LR                   ; Put PCB's P0LR into register
                   03   18   EF  0228   310          EXTZV   #PTX$V_ASTLVL, #PTX$S_ASTLVL,-  ; Put PCB defined ASTLVL
                51   54 A0        022B   311                  PTX$L_P0LRASTL(R0), R1          ; into longword temp.
                   13   51   DA  022E   312          MTPR    R1, #PR$_ASTLVL                 ; Put ASTLVL data in register
                0A   58 A0   DA  0231   313          MTPR    PTX$L_P1BR(R0), #PR$_P1BR       ; Stick P1BR in register
                   16   00   EF  0235   314          EXTZV   #PTX$V_P1LR, #PTX$S_P0LR,-      ; Put PCB defined P1BR
                51   5C A0        0238   315                  PTX$L_P1LRPME(R0), R1          ; in a longword
                   0B   51   DA  023B   316          MTPR    R1, #PR$_P1LR                   ; Put P1BR longwd in reg
             5C A0 01   1F   EF  023E   317          EXTZV   #PTX$V_PME, #1, PTX$L_P1LRPME(R0),-
                      51        0243   318                  R1                              ; Make PCB PME a longword
                   3D   51   DA  0244   319          MTPR    R1, #PR$_PME                    ; Put PME longwd in reg
                             0247   320
                50   10 A0   7D  0247   321          MOVQ    PTX$L_R0(R0),R0                 ; Put R0 and R1 into regs
                             05  024B   322          RSB
```

```
                     024C   324 $DS_SBTTL       <Load Process Context>
                     024C               .SBTTL  TEST 11:  Load Process Context
                     024C               $D2_SBTTL       011,<Load Process Context>,<PAGE>
                    00000000                   .PSECT  TEST_011, PAGE, NOWRT
                     0018
                     0018   325
                     0018   326 ;++
                     0018   327 ; TEST DESCRIPTION:
                     0018   328 ;
                     0018   329 ;
                     0018   330 ; ASSUMPTIONS:
                     0018   331 ;
                     0018   332 ;       The MFPR and MTPR privileged instructions ran successfully.  It
                     0018   333 ;       is also assumed the user is in kernel mode and in standalone.
                     0018   334 ;
                     0018   335 ; TEST STEPS:
                     0018   336 ;
                     0018   337 ;       The Bgntest in this module forces an exception in order to get
                     0018   338 ;       on the interrupt stack.  From the initial exception handler
                     0018   339 ;       TEST_CONTROL which resides in the Bliss module (ENKAXL) is
                     0018   340 ;       called.  TEST_CONTROL fakes a Bgntest in its module in order
                     0018   341 ;       that it may run all of the LDPCTX subtests.
                     0018   342 ;       Each subtest sets up initial and expected data for the registers
                     0018   343 ;       and the Process Control Block which will receive or give data.
                     0018   344 ;       The subtest also requests which stack it will start execution
                     0018   345 ;       off of (through CHECK_STACK) and the mode (Kernel, executive,
                     0018   346 ;       supervisor or user, through MODE) that it wishes to be in at
                     0018   347 ;       the start of the instruction.  After the set up a common EXECUTE
                     0018   348 ;       routine is called.
                     0018   349 ;
                     0018   350 ;       EXECUTE evaluates CHECK_STACK and MODE and sets up a PSL
                     0018   351 ;       accordingly. It sets the PC to the section of code which will
                     0018   352 ;       execute the LDPCTX .
                     0018   353 ;       The new PSL and PC are pushed on the stack and an exception is
                     0018   354 ;       forced.  This causes the PSL and PC on the stack to become
                     0018   355 ;       current so we start the instruction on the desired stack and mode
                     0018   356 ;       and pointing at the correct code.  The flow of execution will
                     0018   357 ;       go to the exception handler (in ENKAXK) regardless of what
                     0018   358 ;       happens in the instruction.  If an expected or unexpected
                     0018   359 ;       exception occurs during execution and control goes to the handler
                     0018   360 ;       otherwise, a breakpoint exception is forced after the instruction.
                     0018   361 ;       The handler records the exception for which it was called in
                     0018   362 ;       Actual_Except or resignals DS if it is unknown.  The handler
                     0018   363 ;       returns to the Save_Results label in Execute. The register
                     0018   364 ;       results are saved in Actual_Regs.  A check is made to ensure
                     0018   365 ;       we landed on the stack desired correctly as specified by the
                     0018   366 ;       SRM. A return code is put in R0, a value of one is normal,
                     0018   367 ;       successful completion.  Execute then returns to the subtest
                     0018   368 ;       it was called from
                     0018   369 ;
                     0018   370 ;         The program aborts if Execute was not successful,  since it
                     0018   371 ;       is likely the stacks will be useless.  If Execute is successful
                     0018   372 ;       Compare_Pcb is called.  Compare_Pcb checks Actual_Pcb against
                     0018   373 ;       Expected_Pcb and Actual_Regs against Expected_Regs.  The routine
                     0018   374 ;       returns Pcb_Err_Flag and Reg_Err_Flag with one or more bits set
                     0018   375 ;       if there are any discrepencies between actual and expected data.
                     0018   376 ;       If there is any difference between actual and expected data or
```

```
                                0018   377 ;       exception then the subtest calls an error printout routine (in
                                0018   378 ;       ENKAXK).
                                0018   379 ;
                                0018   380 ; ERRORS:
                                0018   381 ;
                                0018   382 ;       *** DETAILED DISCRIPTION OF THE ERRORS DETECTABLE AND REPORTED ***
                                0018   383 ;
                                0018   384 ;General
                                0018   385 ;
                                0018   386 ;       ERROR 01:  User is not in kernel mode
                                0018   387 ;       ERROR 02:  Program can not turn memory management off
                                0018   388 ;                  memory
                                0018   389 ;       ERROR 03:  Discrepency in Actual and Expected data or exception.
                                0018   390 ;
                                0018   391 ;Ldpctx
                                0018   392 ;
                                0018   393 ;       ERROR 04:  Did not stay on kernel stack during Ldpctx
                                0018   394 ;       ERROR 05:  Did not get on to the kernel stack during Ldpctx
                                0018   395 ;       ERROR 06:  Interrupt stack pointer set incorrectly when switching
                                0018   396 ;                  to kernel stack on Ldpctx
                                0018   397 ;       ERROR 07:  Interrupt stack PC not set properly,on Ldpctx
                                0018   398 ;       ERROR 08:  Kernel stack PSL not set properly, on Ldpctx
                                0018   399 ;
                                0018   400 ; DEBUG:
                                0018   401 ;
                                0018   402 ;       THIS SECTION WILL CONTAIN INSTRUCTIONS ON HOW TO USE THIS
                                0018   403 ;       TEST IN DEBUGGING THE UNIT UNDER TEST.
                                0018   404 ;
                                0018   405 ;--
                                0018   406 $DS_BGNTEST     SECTION= <DEFAULT, LDPCTX> REGMASK = <>
                                0018       DATA_011:
                    00000000    0018                       .LONG   0                ; TEST ARGUMENT TABLE TERMINATOR
                                001C       TEST_011::
                        0000    001C                       .WORD   ^M<>             ; ENTRY MASK
                                001E   407
                                001E   408
00000097'EF     01      DB      001E   409           MFPR    #PR$_ESP, TEMP_ESP       ; Save stack address in
0000009B'EF     02      DB      0025   410           MFPR    #PR$_SSP, TEMP_SSP       ; temporary locations
0000009F'EF     03      DB      002C   411           MFPR    #PR$_USP, TEMP_USP       ; for later restoration.
                                0033   412
                                0033   413 $DS_SETVEC_S    #SCB$L_BREAK, INIT_HNDLR, #1    ; Set up initial BPT
                    01      DD  0033                   PUSHL   #1
          00000060'EF       DF  0035                   PUSHAL  INIT_HNDLR
                    2C      DD  003B                   PUSHL   #SCB$L_BREAK
00000000'9F     03      FB      003D                   CALLS   #3, @#DS$SETVEC
                                0044   414                                        ; instruction handler address
                        03      0044   415           BPT                          ; Get a BPT exception that gets
                                0045   416                                        ; us to a handler that puts us
                                0045   417                                        ; on the interrupt stack
                                0045   418 BACK:                                  ;
01      00000097'EF     DA      0045   419           MTPR    TEMP_ESP, #PR$_ESP       ; Restore stack addresses
02      0000009B'EF     DA      004C   420           MTPR    TEMP_SSP, #PR$_SSP       ;
03      0000009F'EF     DA      0053   421           MTPR    TEMP_USP, #PR$_USP       ;
                                005A   422
                    02A8    31  005A   423           BRW     END_TEST                 ; After the last subtest is done
                                005D   424                                        ; an REI returns us to here and
                                005D   425                                        ; the test is over
```

```
                                 005D   426 .ALIGN LONG                                ; DS setvec routine requires it to
                                 0060   427                                            ; be longword aligned
                                 0060   428 !INIT_HNDLR:
                                 0060   429         $DS_CLRVEC_S    #SCB$L_BREAK    ; DS uses this vec so leave it
                          2C DD  0060                 PUSHL   #SCB$L_BREAK
              00000000'9F 01 FB  0062                 CALLS   #1, @#DS$CLRVEC
                                 0069   430                                            ; clean
              00000000'EF 00 FB  0069   431         CALLS   #0, TEST_CONTROL        ; Go to Bliss module and do all
                                 0070   432                                            ; of the subtests, return when
                                 0070   433                                            ; all done
                    6E D2 AF DE  0070   434         MOVAL   BACK, (SP)              ; Make sure BPT is not executed again
                                 0074   435                                            ; on the next try
                             02  0074   436         REI                             ; Return to the instruction following
                                 0075   437                                            ; the BPT instruction
                                 0075   438
                                 0075   439
                                 0075   440
```

```
                                0075   442 EXECUTE::
                                0075   443
                                0075   444 ;
                                0075   445 ;
                                0075   446 ;
                                0075   447 ;
                                0075   448 ;
                                0075   449 ; CODE BEGINS
                                0075   450
                                0075   451
          00000060'EF   6E   D0 0075   452         MOVL    (SP), TEMP_PC   ; Save PC for restoration at the end of
                                007C   453                                 ; the subtest
               00000064'EF   DC 007C   454         MOVPSL  TEMP_PSL        ; Save PSL for later restoration
                                0082   455
                                0082   456         SAVE_R  TEMP_REGS       ; Save contents of registers for
               00000000'EF   9F 0082                PUSHAB  TEMP_REGS               ; Put TEMP_REGS address on stack, don't
               00000168'EF   16 0088                JSB     SAVE_REGS       ; Do the Save_regs routine
                   5E   04   C0 008E                ADDL2   #4, SP          ; Pop TEMP_REGS address off the stack
                                0091   457                                 ; restoration after instruction execution
                                0091   458 ;
                                0091   459 ;
                                0091   460 ;*******************************************************************************
                                0091   461 ;
                                0091   462 ; Set up previous mode for each test case
                                0091   463 ;
                                0091   464 ; NOTE: The mode is always kernel for now.
                                0091   465 ;
                                0091   466 CHECK_STK:
               00000070'EF   DC 0091   467         MOVPSL  INIT_PSL                ; Get a normal PSL value
     02   16   00000000'EF   F0 0097   468         INSV    MODE, #PSL$V_PRVMOD, #PSL$S_PRVMOD,-
               00000070'EF            009F   469                 INIT_PSL                ; Define previous mode as requested
                                00A4   470                                         ; for each subtest
                                00A4   471 ;
                                00A4   472 ;
                                00A4   473 ;*******************************************************************************
                                00A4   474 ;
                                00A4   475 ;
                                00A4   476 ;       Now set up the stacks to kernal or interrupt as requested
                                00A4   477 ;
                                00A4   478 ;
          0B 00000000'FF   E9   00A4   479         BLBC    CHECK_STACK, CHK_STK_2  ; If low bit set, Check_Stack is
                                00AB   480                                         ; one or three so get on
                                00AB   481                                         ; interrupt stack
                                00AB   482 ;
                                00AB   483 ;
                                00AB   484 ;
00000070'FF    01   1A   01   F0 00AB   485         INSV    #1, #PSL$V_IS, #1,INIT_PSL      ; The instruction will start
                                00B4   486                                         ; on the interrupt stack
                          09   11   00B4   487         BRB     REG_READY
                                00B6   488 ;
                                00B6   489 ;
                                00B6   490 CHK_STK_2:
00000070'EF    01   1A   00   F0 00B6   491         INSV    #0,#PSL$V_IS, #1,INIT_PSL       ; If low bit clear, Check_stack
                                00BF   492                                         ; is zero or two so get on
                                00BF   493                                         ; kernal stack
                                00BF   494 ;
                                00BF   495 ;
```

```
                                00BF   496 REG_READY:
                                00BF   497          PUSH_R  INIT_REG                ; Set up patterns for
                 00000000'EF    9F  00BF           PUSHAB  INIT_REG                ; Put INIT_REG address on stack as parameter
                 000001DF'EF    16  00C5           JSB     PUSH_REGS       ; Do Push_regs routine
                     5E   04    C0  00CB           ADDL2   #4, SP          ; Clean off stack
                                00CE   498                                 ; subtests into the regs
                                00CE   499 ;
                                00CE   500 ;
                                00CE   501
                                00CE   502
     00000074'EF    00000111'EF DE  00CE   503          MOVAL   LDPCTX_PC, INIT_PC      ; Otherwise go to LDPCTX instruction
                                00D9   504                                 ; after general instruction set-up
                                00D9   505 ;
                                00D9   506 ;
                                00D9   507 ;*****************************************************************************
                                00D9   508 ;
                                00D9   509 ;
                                00D9   510 ;
                                00D9   511 ; Set up an exception handler to get us on the interrupt stack so we
                                00D9   512 ; can change the IS bit in the current PSL
                                00D9   513
            00000068'EF    50   7D  00D9   514          MOVQ    R0,TEMP_R0              ; Save R0, R1
                                00E0   515 ;
                                00E0   516          $DS_SETVEC_S    #SCB$L_BREAK, ISTK_HNDLR, #1
                      01        DD  00E0                   PUSHL   #1
                 000000F4'EF    DF  00E2                   PUSHAL  ISTK_HNDLR
                      2C        DD  00E8                   PUSHL   #SCB$L_BREAK
     00000000'9F      03        FB  00EA                   CALLS   #3, @#DS$SETVEC
                               03   00F1   517          BPT                             ; Sets up stack and mode according
                                00F2   518                                 ; to PSL then goes to LDPCTX or
                                00F2   519                                 ; SVPCTX instruct
                                00F2   520          .ALIGN LONG                     ; DS Exception likes the handler
                                00F4   521                                 ; longword aligned
                                00F4   522 ISTK_HNDLR:
                                00F4   523          $DS_CLRVEC_S    #SCB$L_BREAK    ; Don't end up here on next BPT
                      2C        DD  00F4                   PUSHL   #SCB$L_BREAK
     00000000'9F      01        FB  00F6                   CALLS   #1, @#DS$CLRVEC
     6E     00000074'EF    D0  00FD   524          MOVL    INIT_PC, (SP)           ; Go to right instruction code
04   AE     00000070'EF    D0  0104   525          MOVL    INIT_PSL, 4(SP)         ; Be in the correct mode and
                                010C   526                                 ; stack when you get there
                               02   010C   527          REI                             ; Get to the instruction-under-test
                                010D   528
                    5A5A5A5A    010D   529          .LONG   ^X5A5A5A5A              ; If PC is off by a few bits
                                0111   530                                 ; these buffers stop the instruction
                                0111   531 ;
                                0111   532 ;
                                0111   533 ;*****************************************************************************
                                0111   534 ;
                                0111   535 ;
                                0111   536 LDPCTX_PC:
     50     00000068'EF    7D  0111   537          MOVQ    TEMP_R0, R0             ; Save R0, R1
                                0118   538 ;
     00000078'EF    5E    D0  0118   539          MOVL    SP, INST_SP             ; Save stack pointer before LDPCTX
                                011F   540 ;
                                011F   541 ; Set up PSL for new process
                                011F   542 ;
          0000004C'EF       DC  011F   543          MOVPSL  ACTUAL_PCB + PTX$L_PSL
```

```
                                    0125   544 ;
                                    0125   545 ; Set up FP, KSP, & PC for new process
                                    0125   546 ;
            00000044'EF   5D   D0   0125   547          MOVL    FP, ACTUAL_PCB + PTX$L_FP
                                    012C   548
            00000000'EF   00   DB   012C   549          MFPR    #PR$_KSP, ACTUAL_PCB + PTX$L_KSP
                                    0133   550
     00000048'EF   00000153'EF   D0   0133   551          MOVL    LD_PC, ACTUAL_PCB + PTX$L_PC
                                    013E   552 ;
                                    013E   553 ;*******************************************************************************
                                    013E   554 ;
00000000'FF   00000089'EF   05   29 013E   555          CMPC3   #5, NONE, @EXPECTED_EXCEPT      ; Check if excepton is expected
                            07   13 014A   556          BEQL    LD_PC                           ; If not , execute LDPCTX
            6D   00000000'EF   DE   014C   557          MOVAL   HANDLER, (FP)                   ; Othewiswe enable handler
                                    0153   558                                                  ; This makes sure the handler is
                                    0153   559                                                  ; is set up just before it is used
                                    0153   560
                                    0153   561 ;
                                    0153   562 ;*******************************************************************************
                                    0153   563 ;
                                    0153   564 LD_PC:
                            06      0153   565          LDPCTX                                  ; Execute Load process context
                                    0154   566 ;
            00000081'EF   8E   D0   0154   567          MOVL    (SP)+,PUSH_PC                   ; Save pushed PC
            00000085'EF   8E   D0   015B   568          MOVL    (SP)+,PUSH_PSL                  ; Save pushed PSL
                                    0162   569
                                    0162   570 ;
                                    0162   571 ;*******************************************************************************
                                    0162   572 ;
                                    0162   573 ;
         6D   00000000'EF   DE      0162   574          MOVAL   HANDLER, (FP)                   ; Enable handler
                                    0169   575
                            03      0169   576          BPT                                     ; Go to the exception handler if
                                    016A   577                                                  ; didn't get an exception from LDPCTX
                                    016A   578                                                  ; Return to Save_Results
                     5A5A5A5A       016A   579          .LONG   ^X5A5A5A5A
                                    016E   580 ;
                                    016E   581 ;*******************************************************************************
                                    016E   582 ;
                                    016E   583 ;
                                    016E   584 SAVE_RESULTS::                               ; Global, referenced in ENKAXK
                                    016E   585          SAVE_R  ACTUAL_REG
            00000000'EF   9F        016E               PUSHAB  ACTUAL_REG                  ; Put ACTUAL_REG address on stack, don't
            00000168'EF   16        0174               JSB     SAVE_REGS       ; Do the Save_regs routine
                5E   04   C0        017A               ADDL2   #4, SP          ; Pop ACTUAL_REG address off the stack
                                    017D   586                                              ; registers is saved into Actual_Reg
                                    017D   587 ;
                                    017D   588 ;*******************************************************************************
                                    017D   589 ;
                                    017D   590 ; Check status after instruction execution to make sure the instruction
                                    017D   591 ; ended up in the right state .
                                    017D   592 ;
                                    017D   593 CHK_STK_RSLT:
                                    017D   594 ;
                                    017D   595 ;
  00000000'EF   01   1A   EF        017D   596          EXTZV   #PSL$V_IS, #1, -            ; Get Interrupt Stack status from
                     51             0185   597                  ACTUAL_PSL,R1               ; PSL saved after instr, leave in R1
```

```
                                0186   598 ;
                                0186   599 ;*****************************************************************************
                                0186   600 ;
                                0186   601 ;
                                0186   602 ;
                                0186   603 ;
                                0186   604 ; Check if started on the kernal stack and ended on the kernal stack
                                0186   605 ;
                                0186   606
          02   00000000'EF   91  0186   607          CMPB    CHECK_STACK, #2          ; Check if Check_Stack = 2
                       32   12  018D   608          BNEQ    CHK_STK_3                ; if not Check_stack = 3
                    2C 51   E9  018F   609          BLBC    R1, 65$                  ; If psl<is> is set then we
                                0192   610                                           ; strayed from kernel stack
                                0192   611          $DS_ERRHARD_S MSGADR = CHK_1,-
                                0192   612                        PRLINK = PRINT_ERR,-
                                0192   613                        P1 = #NULL,-
                                0192   614                        P2 = EXPECTED_EXCEPT,-
                                0192   615                        P3 = ACTUAL_EXCEPT
             00000000'EF    DD  0192                    PUSHL   ACTUAL_EXCEPT
             00000000'EF    DD  0198                    PUSHL   EXPECTED_EXCEPT
             0000007C'8F    DD  019E                    PUSHL   #NULL
             00000000'EF    DF  01A4                    PUSHAL  PRINT_ERR
             000000A3'EF    DF  01AA                    PUSHAL  CHK_1
                       00   DD  01B0                    PUSHL   #0
                       01   DD  01B2                    PUSHL   #$ER
                            01B4       ;;; TEST 11, SUBTEST 0, ERROR 1
          00000000'9F    07   FB  01B4                    CALLS   #$$M, @#DS$ERRHARD
                    00EB   31  01BB   616          BRW     ABORTS
                    0040   31  01BE   617 65$:     BRW     L_END
                                01C1   618 ;
                                01C1   619 ;*****************************************************************************
                                01C1   620 ;
                                01C1   621 ;
                                01C1   622 ; Check if stared on interrupt stack and ended on kernal stack
                                01C1   623 ;
                                01C1   624 CHK_STK_3:
                                01C1   625
          0000007D'EF    04   DB  01C1   626          MFPR    #PR$_ISP, TEMP_LONGWORD
00000078'EF    0000007D'EF   D1  01C8   627          CMPL    TEMP_LONGWORD, INST_SP
                       2C   13  01D3   628          BEQL    L_END
                                01D5   629          $DS_ERRHARD_S MSGADR = CHK_2,-
                                01D5   630                        PRLINK = PRINT_ERR,-
                                01D5   631                        P1 = #NULL,-
                                01D5   632                        P2 = EXPECTED_EXCEPT,-
                                01D5   633                        P3 = ACTUAL_EXCEPT
             00000000'EF    DD  01D5                    PUSHL   ACTUAL_EXCEPT
             00000000'EF    DD  01DB                    PUSHL   EXPECTED_EXCEPT
             0000007C'8F    DD  01E1                    PUSHL   #NULL
             00000000'EF    DF  01E7                    PUSHAL  PRINT_ERR
             000000CC'EF    DF  01ED                    PUSHAL  CHK_2
                       00   DD  01F3                    PUSHL   #0
                       02   DD  01F5                    PUSHL   #$ER
                            01F7       ;;; TEST 11, SUBTEST 0, ERROR 2
          00000000'9F    07   FB  01F7                    CALLS   #$$M, @#DS$ERRHARD
                    00A8   31  01FE   634          BRW     ABORTS
                                0201   635 ;
                                0201   636 ; Check if ended on kernal stack
```

```
                                0201    637 ;
                                0201    638 L_END:
          00000000'EF    D5     0201    639          TSTL    OPR_ABORT          ;Check if reserved operand
                                0207    640                                     ;occured.
                      2F    12  0207    641          BNEQ    15$                ;If not  equal then PSL is no good
                   2F 51    E9  0209    642          BLBC    R1, 20$            ; If PSL<IS> is set then there was no
                               020C    643                                     ; switch to kernel stack
                               020C    644          $DS_ERRHARD_S MSGADR = CHK_21,-
                               020C    645                        PRLINK = PRINT_ERR,-
                               020C    646                        P1 = #NULL,-
                               020C    647                        P2 = EXPECTED_EXCEPT,-
                               020C    648                        P3 = ACTUAL_EXCEPT
          00000000'EF    DD     020C                       PUSHL   ACTUAL_EXCEPT
          00000000'EF    DD     0212                       PUSHL   EXPECTED_EXCEPT
          0000007C'8F    DD     0218                       PUSHL   #NULL
          00000000'EF    DF     021E                       PUSHAL  PRINT_ERR
          000000F2'EF    DF     0224                       PUSHAL  CHK_21
                    00   DD     022A                       PUSHL   #0
                    03   DD     022C                       PUSHL   #$ER
                               022E         ;;; TEST 11, SUBTEST 0, ERROR 3
    00000000'9F    07    FB     022E                       CALLS   #$$M, @#DS$ERRHARD
                  0071    31     0235    649          BRW     ABORTS
                               0238    650 ;
                               0238    651 ;*****************************************************************************
                               0238    652 ;
                               0238    653 ;
                               0238    654 ; Check pushed PC & PSL during LDPCTX operation
                               0238    655 ;
                      0072    31     0238    656 15$:     BRW     ENDS
                               0238    657 20$:
                               0238    658
00000048'EF    00000081'EF    D1     0238    659          CMPL    PUSH_PC, ACTUAL_PCB + PTX$L_PC  ; Get pushed PC, does it
                               0246    660                                                     ; it match PC in PCB
                      2B    13     0246    661          BEQL    30$
                               0248    662          $DS_ERRHARD_S MSGADR = CHK_PC,-
                               0248    663                        PRLINK = PRINT_ERR,-
                               0248    664                        P1 = #NULL,-
                               0248    665                        P2 = EXPECTED_EXCEPT,-
                               0248    666                        P3 = ACTUAL_EXCEPT
          00000C00'EF    DD     0248                       PUSHL   ACTUAL_EXCEPT
          00000000'EF    DD     024E                       PUSHL   EXPECTED_EXCEPT
          0000007C'8F    DD     0254                       PUSHL   #NULL
          00000000'EF    DF     025A                       PUSHAL  PRINT_ERR
          0000011F'EF    DF     0260                       PUSHAL  CHK_PC
                    00   DD     0266                       PUSHL   #0
                    04   DD     0268                       PUSHL   #$ER
                               026A         ;;; TEST 11, SUBTEST 0, ERROR 4
    00000000'9F    07    FB     026A                       CALLS   #$$M, @#DS$ERRHARD
                    36    11     0271    667          BRB     ABORTS
                               0273    668 ;
                               0273    669 30$:
0000004C'EF    00000085'EF    D1     0273    670          CMPL    PUSH_PSL, ACTUAL_PCB + PTX$L_PSL ; Get pushed PSL, should
                               027E    671                                                      ; match the one in PCB created
                      2D    13     027E    672          BEQL    ENDS
                               0280    673          $DS_ERRHARD_S MSGADR = CHK_PSL-
                               0280    674                        PRLINK = PRINT_ERR,-
                               0280    675                        P1 = #NULL,-
```

```
                                0280   676                          P2 = EXPECTED_EXCEPT,-
                                0280   677                          P3 = ACTUAL_EXCEPT
                00000000'EF   DD  0280                       PUSHL   ACTUAL_EXCEPT
                00000000'EF   DD  0286                       PUSHL   EXPECTED_EXCEPT
                0000007C'8F   DD  028C                       PUSHL   #NULL
                00000000'EF   DF  0292                       PUSHAL  PRINT_ERR
                00000145'EF   DF  0298                       PUSHAL  CHK_PSL
                        00    DD  029E                       PUSHL   #0
                        05    DD  02A0                       PUSHL   #$ER
                                02A2           ;;; TEST 11, SUBTEST 0, ERROR 5
              00000000'9F   07  FB  02A2                       CALLS   #$$M, @#DS$ERRHARD
                                02A9   678 ; End of status check
                                02A9   679 ;
                                02A9   680 ;*******************************************************************************
                                02A9   681 ;
                                02A9   682
                                02A9   683 ;
                                02A9   684 ; If EXECUTE fails
                                02A9   685 ;
                        50    94  02A9   686 ABORTS: CLRB    R0                      ; Failed to complete normally
                        03    11  02AB   687         BRB     EXIT
                                02AD   688
                                02AD   689 ;
                                02AD   690 ; If EXECUTE passes
                                02AD   691 ;
                    50  01    D0  02AD   692 ENDS:   MOVL    #1, R0                  ; Normal successful completion
                                02B0   693
                                02B0   694 ;
                                02B0   695 ; Restore registers saved , before EXECUTE was called
                                02B0   696 ;
00000010'EF         50    D0    02B0   697 EXIT:   MOVL    R0, TEMP_REGS + PTX$L_R0        ; Put current value of
                                02B7   698                                             ; R0 in registers that will return
00000068'EF         50    D0    02B7   699         MOVL    R0,TEMP_R0                  ; Save return status for Execute
                                02BE   700         PUSH_R  TEMP_REGS                   ; Restore the registers as they
          00000000'EF     9F    02BE               PUSHAB  TEMP_REGS                   ; Put TEMP_REGS address on stack as paramete
          000001DF'EF     16    02C4               JSB     PUSH_REGS       ; Do Push_regs routine
              5E    04    C0    02CA               ADDL2   #4, SP          ; Clean off stack
                                02CD   701                                             ; were when Execute was called
                                02CD   702
                                02CD   703 ;
                                02CD   704 ; Set up stack to return from EXECUTE
                                02CD   705 ;
                                02CD   706         $DS_SETVEC_S    #SCB$L_BREAK,-      ; Get on interrupt stack so we
                                02CD   707                         END_HANDLR, #1      ; can change back to the stack
                        01    DD  02CD                       PUSHL   #1
          000002E8'EF         DF  02CF                       PUSHAL  END_HANDLR
                        2C    DD  02D5                       PUSHL   #SCB$L_BREAK
              00000000'9F   03  FB  02D7                       CALLS   #3, @#DS$SETVEC
                                02DE   708                                             ; we started Execute on.
                        03      02DE   709         BPT                                 ; Go to End_Handlr
                                02DF   710 RETURN:
6E          00000060'EF   D0    02DF   711         MOVL    TEMP_PC, (SP)               ; Restore saved PC
                        05      02E6   712         RSB                                 ; Return to the subtest that called
                                02E7   713
                                02E7   714         .ALIGN LONG
                                02E8   715
                                02E8   716 END_HANDLR:
```

```
                                 02E8   717          $DS_CLRVEC_S    #SCB$L_BREAK    ; Let DS take care of any
                          2C     DD     02E8                 PUSHL   #SCB$L_BREAK
              00000000'9F 01     FB     02EA                 CALLS   #1, @#DS$CLRVEC
                                 02F1   718                                          ; unexpected exceptions from here
              50  00000068'EF    D0     02F1   719          MOVL    TEMP_R0, R0     ; Reload status saved
                                 02F8   720 ;
                                 02F8   721 ; Set up PSL for proper return
                                 02F8   722 ;
           04 AE  00000064'EF    D0     02F8   723          MOVL    TEMP_PSL, 4(SP) ; Put the PSL stack and mode back
                   6E    DC AF   DE     0300   724          MOVAL   RETURN, (SP)    ; like we found it
                                 02     0304   725          REI
                                 0305   726
                                 0305   727 ; Time to get out of Execute
                                 0305   728
                                 0305   729
                                 0305   730 ;*****************************************************************************
                                 0305   731 ;                           END     OF      TEST
                                 0305   732 ;*****************************************************************************
                                 0305   733
                                 0305   734 END_TEST:
                                 0305   735
                                 0305   736 $DS_ENDTEST                           ; End of test
                          50     01     D0     0305                 MOVL    #1, R0          ; NORMAL EXIT
                                 0308           TEST_011_X::
              00000000'9F 6E     FA     0308                 CALLG   (SP), @#DS$BREAK
                                 04     030F                 RET                     ; RETURN TO TEST SEQUENCER
                                 0310   737
                                 0310   738 $DS_ENDMOD                            ; End of module
                          00000000                       .PSECT  $TSTCNT, NOEXE, NOWRT, OVR, LONG
                    0000000B  0000                       .LONG   $TN
                                 0004   739
                                 0004   740 .END
```

```
$$$                             = 00000000
$$M                             = 00000007
$$N                             = 00000001
$$S                             = FFFFFFFF
$ENV                            = 00000001  G
$ER                             = FFFFFFFF
$MO                             = 00000001  G
$ST                             = 00000000
$TN                             = 0000000B
ABORTS                            000002A9 R      03
ACTUAL_EXCEPT                     ******** X    03
ACTUAL_PCB                        ******** X    03
ACTUAL_PSL                        ******** X    03
ACTUAL_REG                        ******** X    03
ALL                             = 00000008  G
BACK                              00000045 R      03
BASE_011                          00000000 R      03
BIT...                          = 00000002
CEP_FUNCTIONAL                  = 00000000
CEP_REPAIR                      = 00000001
CHECK_STACK                       ******** X    03
CHECK_STK                         00000091 R      03
CHK_1                             000000A3 R      01
CHK_2                             000000CC R      01
CHK_21                            000000F2 R      01
CHK_PC                            0000011F R      01
CHK_PSL                           00000145 R      01
CHK_STK_2                         000000B6 R      03
CHK_STK_3                         000001C1 R      03
CHK_STK_RSLT                      0000017D R      03
CPU$V_COMET                     = 00000002
CPU$V_HS                        = 00000000
CPU$V_STAR                      = 00000001
DATA_011                          00000018 R      03
DEFAULT                         = 00000000  G
DS$BREAK                          ******** X    03
DS$CLRVEC                         ******** X    03
DS$ERRHARD                        ******** X    03
DS$SETVEC                         ******** X    03
DSA$AL_APTMAIL                    0000FE00
DSA$AT_APTTXT                     0000FA00
DSA$GL_APTCOM                     0000FE04
DSA$GL_DEVLEN                     0000FE58
DSA$GL_ERRNO                      0000FE44
DSA$GL_EVENT                      0000FE48
DSA$GL_FLAGS                      0000FE00
DSA$GL_MSGTYP                     0000FE40
DSA$GL_PASSES                     0000FE08
DSA$GL_PASSNO                     0000FE54
DSA$GL_SECTNO                     0000FE10
DSA$GL_SID                        0000FE14
DSA$GL_SUBTNO                     0000FE4C
DSA$GL_TESTNO                     0000FE50
DSA$GL_UNITS                      0000FE0C
DSA$GQ_MSGPTR                     0000FE68
DSA$GT_DEVNAM                     0000FE5C
DSA$M_APT                       = 80000000
DSA$M_BELL                      = 00000008
DSA$M_BINARY                    = 00000002
DSA$M_CRD_AUTOTEST              = 00000800
DSA$M_CRD_MENUTEST              = 00010000
DSA$M_CRD_TRACE                 = 00020000
DSA$M_DEBUG                     = 04000000
DSA$M_HALT                      = 00000001
DSA$M_IE1                       = 00000010
DSA$M_IE2                       = 00000020
DSA$M_IE3                       = 00000040
DSA$M_IES                       = 00000080
DSA$M_LOAD_DEBUGGER             = 40000000
DSA$M_LOOP                      = 00000004
DSA$M_NORPT                     = 08000000
DSA$M_OPER                      = 00001000
DSA$M_PASSO                     = 20000000
DSA$M_PROMPT                    = 00002000
DSA$M_QA                        = 00008000
DSA$M_QUICK                     = 00000100
DSA$M_SEARCH                    = 00004000
DSA$M_TRACE                     = 00000400
DSA$M_USER                      = 10000000
DSA$M_VERIFY                    = 00000200
DSA$V_APT                       = 0000001F
DSA$V_BELL                      = 00000003
DSA$V_BINARY                    = 00000001
DSA$V_CRD_AUTOTEST              = 0000000B
DSA$V_CRD_MENUTEST              = 00000010
DSA$V_CRD_TRACE                 = 00000011
DSA$V_DEBUG                     = 0000001A
DSA$V_HALT                      = 00000000
DSA$V_IE1                       = 00000004
DSA$V_IE2                       = 00000005
DSA$V_IE3                       = 00000006
DSA$V_IES                       = 00000007
DSA$V_LOAD_DEBUGGER             = 0000001E
DSA$V_LOOP                      = 00000002
DSA$V_NORPT                     = 0000001B
DSA$V_OPER                      = 0000000C
DSA$V_PASSO                     = 0000001D
DSA$V_PROMPT                    = 0000000D
DSA$V_QA                        = 0000000F
DSA$V_QUICK                     = 00000008
DSA$V_SEARCH                    = 0000000E
DSA$V_TRACE                     = 0000000A
DSA$V_USER                      = 0000001C
DSA$V_VERIFY                    = 00000009
ENDS                              000002AD R      03
END_HANDLR                        000002E8 R      03
END_TEST                          00000305 R      03
ENV$M_DOMAIN                    = 00000002
ENV$M_LEVEL                     = 00000001
ENV$M_SUPER                     = 000003FC
ENV$S_DOMAIN                    = 00000001
ENV$S_LEVEL                     = 00000001
ENV$S_SUPER                     = 00000008
ENV$V_DOMAIN                    = 00000001
```

```
ENV$V_LEVEL                     = 00000000
ENV$V_SUPER                     = 00000002
ENV$_CPU                        = 00000000
ENV$_FUNCTIONAL                 = 00000000
ENV$_REPAIR                     = 00000001
ENV$_SUPER                      = 00000001
ENV$_SYSTEM                     = 00000001
EXECUTE                           00000075 RG    03
EXIT                              000002B0 R     03
EXPECTED_EXCEPT                   ******** X     03
EXPECT_REG                        ******** X     01
HALT                            = 00000004 G
HANDLER                           ******** X     03
INIT_HNDLR                        00000060 R     03
INIT_PC                           00000074 R     01
INIT_PSL                          00000070 R     01
INIT_REG                          ******** X     03
INST_SP                           00000078 R     01
IPR                             = 00000006 G
ISTK_HNDLR                        000000F4 R     03
LDPCTX                          = 0000000B G
LDPCTX_PC                         00000111 R     03
LD_PC                             00000153 R     03
LOOP                              0000020A R     01
L_END                             00000201 R     03
MANUAL                          = 00000001 G
MCHK                            = 00000005 G
MEM                             = 00000009 G
MODE                              ******** X     03
MSG_011                           00000002 R     03
NONE                              00000089 R     01
NULL                              0000007C R     01
OPR_ABORT                         ******** X     03
POWER                           = 00000003 G
PR$S_SID_ECO                    = 00000009
PR$S_SID_PL                     = 00000003
PR$S_SID_SN                     = 0000000C
PR$S_SID_TYPE                   = 00000008
PR$V_SID_ECO                    = 0000000F
PR$V_SID_PL                     = 0000000C
PR$V_SID_SN                     = 00000000
PR$V_SID_TYPE                   = 00000018
PR$_ACCR                        = 00000029
PR$_ACCS                        = 00000028
PR$_ASTLVL                      = 00000013
PR$_CADR                        = 00000025
PR$_CAER                        = 00000027
PR$_CMIERR                      = 00000017
PR$_CRBT                        = 00000043
PR$_CSRD                        = 0000001D
PR$_CSRS                        = 0000001C
PR$_CSTD                        = 0000001F
PR$_CSTS                        = 0000001E
PR$_CSWP                        = 00000042
PR$_ESP                         = 00000001
PR$_ICCS                        = 00000018
PR$_ICR                         = 0000001A
PR$_IPL                         = 00000012
PR$_ISP                         = 00000004
PR$_KSP                         = 00000000
PR$_MAPEN                       = 00000038
PR$_MCESR                       = 00000026
PR$_MCTL1                       = 00000044
PR$_MCTL2                       = 00000045
PR$_MDCR1                       = 00000049
PR$_MEAR                        = 00000048
PR$_MECCR                       = 0000004B
PR$_MEDR                        = 0000004A
PR$_MGEN                        = 00000046
PR$_MTBER                       = 00000047
PR$_NICR                        = 00000019
PR$_P0BR                        = 00000008
PR$_P0LR                        = 00000009
PR$_P1BR                        = 0000000A
PR$_P1LR                        = 0000000B
PR$_PAMACC                      = 00000040
PR$_PAMLOC                      = 00000041
PR$_PCBB                        = 00000010
PR$_PME                         = 0000003D
PR$_RXCS                        = 00000020
PR$_RXDB                        = 00000021
PR$_SBIER                       = 00000034
PR$_SBIFS                       = 00000030
PR$_SBIMT                       = 00000033
PR$_SBIQC                       = 00000036
PR$_SBIS                        = 00000031
PR$_SBISC                       = 00000032
PR$_SBITA                       = 00000035
PR$_SBR                         = 0000000C
PR$_SCBB                        = 00000011
PR$_SID                         = 0000003E
PR$_SID_TYP730                  = 00000003
PR$_SID_TYP750                  = 00000002
PR$_SID_TYP780                  = 00000001
PR$_SID_TYP7VV                  = 00000004
PR$_SID_TYPMAX                  = 00000004
PR$_SIRR                        = 00000014
PR$_SISR                        = 00000015
PR$_SLR                         = 0000000D
PR$_SSP                         = 00000002
PR$_TBCHK                       = 0000003F
PR$_TBDR                        = 00000024
PR$_TBIA                        = 00000039
PR$_TBIS                        = 0000003A
PR$_TODR                        = 0000001B
PR$_TXCS                        = 00000022
PR$_TXDB                        = 00000023
PR$_UBRESET                     = 00000037
PR$_USP                         = 00000003
PR$_WCSA                        = 0000002C
PR$_WCSD                        = 0000002D
PRINT_ERR                         ********   X   03
PSL$C_EXEC                      = 00000001
PSL$C_KERNEL                    = 00000000
```

```
PSL$C_SUPER                     = 00000002
PSL$C_USER                      = 00000003
PSL$M_C                         = 00000001
PSL$M_CM                        = 80000000
PSL$M_CURMOD                    = 03000000
PSL$M_DV                        = 00000080
PSL$M_FPD                       = 08000000
PSL$M_FU                        = 00000040
PSL$M_IPL                       = 001F0000
PSL$M_IS                        = 04000000
PSL$M_IV                        = 00000020
PSL$M_N                         = 00000008
PSL$M_PRVMOD                    = 00C00000
PSL$M_SAFBITS                   = 000037FF
PSL$M_TBIT                      = 00000010
PSL$M_TP                        = 40000000
PSL$M_V                         = 00000002
PSL$M_Z                         = 00000004
PSL$S_CURMOD                    = 00000002
PSL$S_IPL                       = 00000005
PSL$S_PRVMOD                    = 00000002
PSL$V_C                         = 00000000
PSL$V_CM                        = 0000001F
PSL$V_CURMOD                    = 00000018
PSL$V_DV                        = 00000007
PSL$V_FPD                       = 0000001B
PSL$V_FU                        = 00000006
PSL$V_IPL                       = 00000010
PSL$V_IS                        = 0000001A
PSL$V_IV                        = 00000005
PSL$V_N                         = 00000003
PSL$V_PRVMOD                    = 00000016
PSL$V_TBIT                      = 00000004
PSL$V_TP                        = 0000001E
PSL$V_V                         = 00000001
PSL$V_Z                         = 00000002
PTX$K_SIZE                        00000060
PTX$L_AP                          00000040
PTX$L_ESP                         00000004
PTX$L_FP                          00000044
PTX$L_KSP                         00000000
PTX$L_P0BR                        00000050
PTX$L_P0LRASTL                    00000054
PTX$L_P1BR                        00000058
PTX$L_P1LRPME                     0000005C
PTX$L_PC                          00000048
PTX$L_PSL                         0000004C
PTX$L_R0                          00000010
PTX$L_R1                          00000014
PTX$L_R10                         00000038
PTX$L_R11                         0000003C
PTX$L_R2                          00000018
PTX$L_R3                          0000001C
PTX$L_R4                          00000020
PTX$L_R5                          00000024
PTX$L_R6                          00000028
PTX$L_R7                          0000002C
PTX$L_R8                          00000030
PTX$L_R9                          00000034
PTX$L_SSP                         00000008
PTX$L_USP                         0000000C
PTX$S_ASTLVL                    = 00000003
PTX$S_MBZ1                      = 00000002
PTX$S_MBZ2                      = 00000005
PTX$S_MBZ3                      = 00000009
PTX$S_P0LR                      = 00000016
PTX$S_P1LR                      = 00000016
PTX$S_PME                       = 00000001
PTX$V_ASTLVL                    = 00000018
PTX$V_MBZ1                      = 00000016
PTX$V_MBZ2                      = 0000001B
PTX$V_MBZ3                      = 00000016
PTX$V_P0LR                      = 00000000
PTX$V_P1LR                      = 00000000
PTX$V_PME                       = 0000001F
PUSH_PC                           00000081 R      01
PUSH_PSL                          00000085 R      01
PUSH_REGS                         000001DF R      01
REG_READY                         000000BF R      03
RETURN                            000002DF R      03
SAVE_REGS                         00000168 R      01
SAVE_RESULTS                      0000016E RG     03
SCB$L_ACCESS                      00000020
SCB$L_ARITH                       00000034
SCB$L_BREAK                       0000002C
SCB$L_CHME                        00000044
SCB$L_CHMK                        00000040
SCB$L_CHMS                        00000048
SCB$L_CHMU                        0000004C
SCB$L_COMPAT                      00000030
SCB$L_KNLSTK                      00000008
SCB$L_MACHCHK                     00000004
SCB$L_OPCCUS                      00000014
SCB$L_OPCDEC                      00000010
SCB$L_POWER                       0000000C
SCB$L_RADRMOD                     0000001C
SCB$L_ROPRAND                     00000018
SCB$L_RXDB                        000000F8
SCB$L_SFTLVL1                     00000084
SCB$L_SFTLVL10                    000000A8
SCB$L_SFTLVL11                    000000AC
SCB$L_SFTLVL12                    000000B0
SCB$L_SFTLVL13                    000000B4
SCB$L_SFTLVL14                    000000B8
SCB$L_SFTLVL15                    000000BC
SCB$L_SFTLVL2                     00000088
SCB$L_SFTLVL3                     0000008C
SCB$L_SFTLVL4                     00000090
SCB$L_SFTLVL5                     00000094
SCB$L_SFTLVL6                     00000098
SCB$L_SFTLVL7                     0000009C
SCB$L_SFTLVL8                     000000A0
SCB$L_SFTLVL9                     000000A4
SCB$L_TBIT                        00000028
```

```
SCB$L_TIMER                     000000C0
SCB$L_TRANSL                    00000024
SCB$L_TXDB                      000000FC
SCB$L_ZERO                      00000000
SEP_FUNCTIONAL                = 00000002
SEP_REPAIR                    = 00000003
SIZ...                        = 00000008
TEMP_ESP                        00000097 R      01
TEMP_LONGWORD                   0000007D R      01
TEMP_PC                         00000060 R      01
TEMP_PSL                        00000064 RG     01
TEMP_R0                         00000068 R      01
TEMP_REGS                       00000000 R      01
TEMP_SSP                        0000009B R      01
TEMP_SV_PC                      0000008F R      01
TEMP_SV_PSL                     00000093 R      01
TEMP_USP                        0000009F R      01
TEST_011                        0000001C RG     03
TEST_011_X                      00000308 RG     03
TEST_CONTROL                    ********   X    03
TU58                          = 00000002  G
UBI                           = 0000000A  G
WCS                           = 00000007  G
```

! Psect synopsis .

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000024C (588.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	0000FE70 (65136.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
TEST_011	00000310 (784.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
DISPATCH	00000018 (24.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
$TSTCNT	00000004 (4.)	05 (5.)	NOPIC USR OVR REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG

! Symbol Cross Reference !

SYMBOL	VALUE	DEFINITION		REFERENCES...							
$$$	=00000000	324	(2)								
$$M	=00000007	677	(2)	406	(2)	615	(2)	633	(2)	648	(2)
				666	(2)	677	(2)				
$$N	=00000001	677	(2)	615	(2)	633	(2)	648	(2)	666	(2)
				677	(2)	78	(2)				
$$S	=FFFFFFFF	736	(2)	324	(2)	406	(2)				
$ENV	=00000001	144	(2)								
$ER	=FFFFFFFF	736	(2)	#-615	(2)	#-633	(2)	#-648	(2)	#-666	(2)
				#-677	(2)						
$MO	=00000001	738	(2)	738	(2)						
$ST	=00000000	736	(2)	406	(2)	615	(2)	633	(2)	648	(2)
				666	(2)	677	(2)	736	(2)		
$TN	=0000000B	738	(2)	324	(2)	406	(2)	615	(2)	633	(2)
				648	(2)	666	(2)	677	(2)	736	(2)
				738	(2)						
ABORTS	000002A9-R	686	(2)	#-616	(2)	#-634	(2)	#-649	(2)	#-667	(2)
ACTUAL_EXCEPT	00000000-XR			#-615	(2)	#-633	(2)	#-648	(2)	#-666	(2)
				#-677	(2)						
ACTUAL_PCB	00000000-XR			#-543	(2)	#-547	(2)	#-549	(2)	#-551	(2)
				#-659	(2)	#-670	(2)				
ACTUAL_PSL	00000000-XR			597	(2)						
ACTUAL_REG	00000000-XR			585	(2)						
ALL	=00000008	78	(2)								
BACK	00000045-R	418	(2)	434	(2)						
BASE_011	00000000-R	324	(2)	406	(2)						
BIT...	=00000002	144	(2)	110	(2)	116	(2)	144	(2)	74	(2)
CEP_FUNCTIONAL	=00000000	144	(2)								
CEP_REPAIR	=00000001	144	(2)	144	(2)						
CHECK_STACK	00000000-XR			#-479	(2)	#-607	(2)				
CHECK_STK	00000091-R	466	(2)								
CHK_1	000000A3-R	204	(2)	615	(2)						
CHK_2	000000CC-R	205	(2)	633	(2)						
CHK_21	000000F2-R	206	(2)	648	(2)						
CHK_PC	0000011F-R	207	(2)	666	(2)						
CHK_PSL	00000145-R	208	(2)	677	(2)						
CHK_STK_2	000000B6-R	490	(2)	#-479	(2)						
CHK_STK_3	000001C1-R	624	(2)	#-608	(2)						
CHK_STK_RSLT	0000017D-R	593	(2)								
CPU$V_COMET	=00000002	74	(2)								
CPU$V_HS	=00000000	74	(2)								
CPU$V_STAR	=00000001	74	(2)								
DATA_011	00000018-R	406	(2)	406	(2)						
DEFAULT	=00000000	78	(2)	406	(2)						
DS$BREAK	00000000-XR			736	(2)						
DS$CLRVEC	00000000-XR			429	(2)	523	(2)	717	(2)		
DS$ERRHARD	00000000-XR			615	(2)	633	(2)	648	(2)	666	(2)
				677	(2)						
DS$SETVEC	00000000-XR			413	(2)	516	(2)	707	(2)		
DSA$AL_APTMAIL	0000FE00	74	(2)								
DSA$AT_APTTXT	0000FA00	74	(2)								

```
DSA$GL_APTCOM                   0000FE04      74      (2)
DSA$GL_DEVLEN                   0000FE58      74      (2)
DSA$GL_ERRNO                    0000FE44      74      (2)
DSA$GL_EVENT                    0000FE48      74      (2)
DSA$GL_FLAGS                    0000FE00      74      (2)
DSA$GL_MSGTYP                   0000FE40      74      (2)
DSA$GL_PASSES                   0000FE08      74      (2)
DSA$GL_PASSNO                   0000FE54      74      (2)
DSA$GL_SECTNO                   0000FE10      74      (2)
DSA$GL_SID                      0000FE14      74      (2)
DSA$GL_SUBTNO                   0000FE4C      74      (2)
DSA$GL_TESTNO                   0000FE50      74      (2)
DSA$GL_UNITS                    0000FE0C      74      (2)
DSA$GQ_MSGPTR                   0000FE68      74      (2)
DSA$GT_DEVNAM                   0000FE5C      74      (2)
DSA$M_APT                      =80000000      74      (2)
DSA$M_BELL                     =00000008      74      (2)
DSA$M_BINARY                   =00000002      74      (2)
DSA$M_CRD_AUTOTEST             =00000800      74      (2)
DSA$M_CRD_MENUTEST             =00010000      74      (2)
DSA$M_CRD_TRACE                =00020000      74      (2)
DSA$M_DEBUG                    =04000000      74      (2)
DSA$M_HALT                     =00000001      74      (2)
DSA$M_IE1                      =00000010      74      (2)
DSA$M_IE2                      =00000020      74      (2)
DSA$M_IE3                      =00000040      74      (2)
DSA$M_IES                      =00000080      74      (2)
DSA$M_LOAD_DEBUGGER            =40000000      74      (2)
DSA$M_LOOP                     =00000004      74      (2)
DSA$M_NORPT                    =08000000      74      (2)
DSA$M_OPER                     =00001000      74      (2)
DSA$M_PASSO                    =20000000      74      (2)
DSA$M_PROMPT                   =00002000      74      (2)
DSA$M_QA                       =00008000      74      (2)
DSA$M_QUICK                    =00000100      74      (2)
DSA$M_SEARCH                   =00004000      74      (2)
DSA$M_TRACE                    =00000400      74      (2)
DSA$M_USER                     =10000000      74      (2)
DSA$M_VERIFY                   =00000200      74      (2)
DSA$V_APT                      =0000001F      74      (2)
DSA$V_BELL                     =00000003      74      (2)
DSA$V_BINARY                   =00000001      74      (2)
DSA$V_CRD_AUTOTEST             =0000000B      74      (2)
DSA$V_CRD_MENUTEST             =00000010      74      (2)
DSA$V_CRD_TRACE                =00000011      74      (2)
DSA$V_DEBUG                    =0000001A      74      (2)
DSA$V_HALT                     =00000000      74      (2)
DSA$V_IE1                      =00000004      74      (2)
DSA$V_IE2                      =00000005      74      (2)
DSA$V_IE3                      =00000006      74      (2)
DSA$V_IES                      =00000007      74      (2)
DSA$V_LOAD_DEBUGGER            =0000001E      74      (2)
DSA$V_LOOP                     =00000002      74      (2)
DSA$V_NORPT                    =0000001B      74      (2)
DSA$V_OPER                     =0000000C      74      (2)
DSA$V_PASSO                    =0000001D      74      (2)
DSA$V_PROMPT                   =0000000D      74      (2)
```

```
DSA$V_QA                        =0000000F      74      (2)
DSA$V_QUICK                     =00000008      74      (2)
DSA$V_SEARCH                    =0000000E      74      (2)
DSA$V_TRACE                     =0000000A      74      (2)
DSA$V_USER                      =0000001C      74      (2)
DSA$V_VERIFY                    =00000009      74      (2)
ENDS                             000002AD-R    692     (2)     #-656     (2)     #-672     (2)
END_HANDLR                       000002E8-R    716     (2)       707     (2)
END_TEST                         00000305-R    734     (2)     #-423     (2)
ENV$M_DOMAIN                    =00000002      144     (2)
ENV$M_LEVEL                     =00000001      144     (2)
ENV$M_SUPER                     =000003FC      144     (2)
ENV$S_DOMAIN                    =00000001      144     (2)
ENV$S_LEVEL                     =00000001      144     (2)
ENV$S_SUPER                     =00000008      144     (2)
ENV$V_DOMAIN                    =00000001      144     (2)       144     (2)
ENV$V_LEVEL                     =00000000      144     (2)       144     (2)
ENV$V_SUPER                     =00000002      144     (2)
ENV$_CPU                        =00000000      144     (2)       144     (2)
ENV$_FUNCTIONAL                 =00000000      144     (2)       144     (2)
ENV$_REPAIR                     =00000001      144     (2)       144     (2)
ENV$_SUPER                      =00000001      144     (2)
ENV$_SYSTEM                     =00000001      144     (2)       144     (2)
EXECUTE                          00000075-R    442     (2)
EXIT                             000002B0-R    697     (2)     #-687     (2)
EXPECTED_EXCEPT                  00000000-XR                     555     (2)     #-615     (2)     #-633     (2)     #-648     (2)
                                                               #-666     (2)     #-677     (2)
EXPECT_REG                       00000000-XR                   #-237     (2)     #-240     (2)     #-242     (2)     #-245     (2)
                                                               #-248     (2)     #-287     (2)     #-291     (2)     #-295     (2)
HALT                            =00000004      78      (2)
HANDLER                          00000000-XR                     557     (2)       574     (2)
INIT_HNDLR                       00000060-R    428     (2)       413     (2)
INIT_PC                          00000074-R    184     (2)     #-503     (2)     #-524     (2)
INIT_PSL                         00000070-R    182     (2)     #-467     (2)       469     (2)       485     (2)       491     (2)
                                                               #-525     (2)
INIT_REG                         00000000-XR                     497     (2)
INST_SP                          00000078-R    186     (2)     #-539     (2)     #-627     (2)
IPR                             =00000006      78      (2)
ISTK_HNDLR                       000000F4-R    522     (2)       516     (2)
LDPCTX                          =0000000B      78      (2)       406     (2)
LDPCTX_PC                        00000111-R    536     (2)       503     (2)
LD_PC                            00000153-R    564     (2)     #-551     (2)     #-556     (2)
LOOP                             0000020A-R    300     (2)     #-301     (2)
L_END                            00000201-R    638     (2)     #-617     (2)     #-628     (2)
MANUAL                          =00000001      78      (2)
MCHK                            =00000005      78      (2)
MEM                             =00000009      78      (2)
MODE                             00000000-XR                   #-468     (2)
MSG_011                          00000002-R    324     (2)       406     (2)
NONE                             00000089-R    192     (2)       555     (2)
NULL                             0000007C-R    188     (2)     #-615     (2)     #-633     (2)     #-648     (2)     #-666     (2)
                                                               #-677     (2)
OPR_ABORT                        00000000-XR                   #-639     (2)
POWER                           =00000003      78      (2)
PR$S_SID_ECO                    =00000009      76      (2)
PR$S_SID_PL                     =00000003      76      (2)
PR$S_SID_SN                     =0000000C      76      (2)
```

```
PR$S_SID_TYPE                   =00000008      76      (2)
PR$V_SID_ECO                    =0000000F      76      (2)
PR$V_SID_PL                     =0000000C      76      (2)
PR$V_SID_SN                     =00000000      76      (2)
PR$V_SID_TYPE                   =00000018      76      (2)
PR$_ACCR                        =00000029      76      (2)
PR$_ACCS                        =00000028      76      (2)
PR$_ASTLVL                      =00000013      76      (2)    #-260    (2)    #-312    (2)
PR$_CADR                        =00000025      76      (2)
PR$_CAER                        =00000027      76      (2)
PR$_CMIERR                      =00000017      76      (2)
PR$_CRBT                        =00000043      76      (2)
PR$_CSRD                        =0000001D      76      (2)
PR$_CSRS                        =0000001C      76      (2)
PR$_CSTD                        =0000001F      76      (2)
PR$_CSTS                        =0000001E      76      (2)
PR$_CSWP                        =00000042      76      (2)
PR$_ESP                         =00000001      76      (2)    #-244    (2)    #-289    (2)    #-409    (2)    #-419    (2)
PR$_ICCS                        =00000018      76      (2)
PR$_ICR                         =0000001A      76      (2)
PR$_IPL                         =00000012      76      (2)
PR$_ISP                         =00000004      76      (2)    #-626    (2)
PR$_KSP                         =00000000      76      (2)    #-239    (2)    #-240    (2)    #-549    (2)
PR$_MAPEN                       =00000038      76      (2)
PR$_MCESR                       =00000026      76      (2)
PR$_MCTL1                       =00000044      76      (2)
PR$_MCTL2                       =00000045      76      (2)
PR$_MDCR1                       =00000049      76      (2)
PR$_MEAR                        =00000048      76      (2)
PR$_MECCR                       =0000004B      76      (2)
PR$_MEDR                        =0000004A      76      (2)
PR$_MGEN                        =00000046      76      (2)
PR$_MTBER                       =00000047      76      (2)
PR$_NICR                        =00000019      76      (2)
PR$_P0BR                        =00000008      76      (2)    #-258    (2)    #-306    (2)
PR$_P0LR                        =00000009      76      (2)    #-259    (2)    #-309    (2)
PR$_P1BR                        =0000000A      76      (2)    #-263    (2)    #-313    (2)
PR$_P1LR                        =0000000B      76      (2)    #-264    (2)    #-316    (2)
PR$_PAMACC                      =00000040      76      (2)
PR$_PAMLOC                      =00000041      76      (2)
PR$_PCBB                        =00000010      76      (2)
PR$_PME                         =0000003D      76      (2)    #-265    (2)    #-319    (2)
PR$_RXCS                        =00000020      76      (2)
PR$_RXDB                        =00000021      76      (2)
PR$_SBIER                       =00000034      76      (2)
PR$_SBIFS                       =00000030      76      (2)
PR$_SBIMT                       =00000033      76      (2)
PR$_SBIQC                       =00000036      76      (2)
PR$_SBIS                        =00000031      76      (2)
PR$_SBISC                       =00000032      76      (2)
PR$_SBITA                       =00000035      76      (2)
PR$_SBR                         =0000000C      76      (2)
PR$_SCBB                        =00000011      76      (2)
PR$_SID                         =0000003E      76      (2)
PR$_SID_TYP730                  =00000003      76      (2)
PR$_SID_TYP750                  =00000002      76      (2)
PR$_SID_TYP780                  =00000001      76      (2)
```

```
PR$_SID_TYP7VV                  =00000004      76      (2)
PR$_SID_TYPMAX                  =00000004      76      (2)
PR$_SIRR                        =00000014      76      (2)
PR$_SISR                        =00000015      76      (2)
PR$_SLR                         =0000000D      76      (2)
PR$_SSP                         =00000002      76      (2)     #-247    (2)     #-293    (2)     #-410    (2)     #-420    (2)
PR$_TBCHK                       =0000003F      76      (2)
PR$_TBDR                        =00000024      76      (2)
PR$_TBIA                        =00000039      76      (2)
PR$_TBIS                        =0000003A      76      (2)
PR$_TODR                        =0000001B      76      (2)
PR$_TXCS                        =00000022      76      (2)
PR$_TXDB                        =00000023      76      (2)
PR$_UBRESET                     =00000037      76      (2)
PR$_USP                         =00000003      76      (2)     #-250    (2)     #-297    (2)     #-411    (2)     #-421    (2)
PR$_WCSA                        =0000002C      76      (2)
PR$_WCSD                        =0000002D      76      (2)
PRINT_ERR                        00000000-XR                    615    (2)      633    (2)      648    (2)      666    (2)
                                                                677    (2)
PSL$C_EXEC                      =00000001      77      (2)
PSL$C_KERNEL                    =00000000      77      (2)
PSL$C_SUPER                     =00000002      77      (2)
PSL$C_USER                      =00000003      77      (2)
PSL$M_C                         =00000001      77      (2)
PSL$M_CM                        =80000000      77      (2)      77      (2)
PSL$M_CURMOD                    =03000000      77      (2)
PSL$M_DV                        =00000080      77      (2)
PSL$M_FPD                       =08000000      77      (2)      77      (2)
PSL$M_FU                        =00000040      77      (2)
PSL$M_IPL                       =001F0000      77      (2)
PSL$M_IS                        =04000000      77      (2)
PSL$M_IV                        =00000020      77      (2)
PSL$M_N                         =00000008      77      (2)
PSL$M_PRVMOD                    =00C00000      77      (2)
PSL$M_SAFBITS                   =000037FF      77      (2)
PSL$M_TBIT                      =00000010      77      (2)
PSL$M_TP                        =40000000      77      (2)      77      (2)
PSL$M_V                         =00000002      77      (2)
PSL$M_Z                         =00000004      77      (2)
PSL$S_CURMOD                    =00000002      77      (2)
PSL$S_IPL                       =00000005      77      (2)
PSL$S_PRVMOD                    =00000002      77      (2)     #-468    (2)
PSL$V_C                         =00000000      77      (2)
PSL$V_CM                        =0000001F      77      (2)
PSL$V_CURMOD                    =00000018      77      (2)
PSL$V_DV                        =00000007      77      (2)
PSL$V_FPD                       =0000001B      77      (2)
PSL$V_FU                        =00000006      77      (2)
PSL$V_IPL                       =00000010      77      (2)
PSL$V_IS                        =0000001A      77      (2)     #-485    (2)     #-491    (2)     #-596    (2)
PSL$V_IV                        =00000005      77      (2)
PSL$V_N                         =00000003      77      (2)
PSL$V_PRVMOD                    =00000016      77      (2)     #-468    (2)
PSL$V_TBIT                      =00000004      77      (2)
PSL$V_TP                        =0000001E      77      (2)
PSL$V_V                         =00000001      77      (2)
PSL$V_Z                         =00000002      77      (2)
```

```
PTX$K_SIZE                      00000060      117     (2)
PTX$L_AP                        00000040      100     (2)   #-299     (2)
PTX$L_ESP                       00000004      85      (2)   #-242     (2)   #-243     (2)   #-244     (2)   #-287     (2)
                                                            #-288     (2)   #-289     (2)
PTX$L_FP                        00000044      101     (2)   #-547     (2)
PTX$L_KSP                       00000000      84      (2)   #-237     (2)   #-238     (2)   #-239     (2)   #-241     (2)
                                                            #-549     (2)
PTX$L_P0BR                      00000050      104     (2)   #-258     (2)   #-306     (2)
PTX$L_P0LRASTL                  00000054      105     (2)   #-259     (2)     262     (2)       308     (2)       311     (2)
PTX$L_P1BR                      00000058      111     (2)   #-263     (2)   #-313     (2)
PTX$L_P1LRPME                   0000005C      112     (2)   #-264     (2)     267     (2)       315     (2)       317     (2)
PTX$L_PC                        00000048      102     (2)   #-551     (2)   #-659     (2)
PTX$L_PSL                       0000004C      103     (2)   #-543     (2)   #-670     (2)
PTX$L_R0                        00000010      88      (2)   #-229     (2)   #-321     (2)   #-697     (2)
PTX$L_R1                        00000014      89      (2)   #-230     (2)
PTX$L_R10                       00000038      98      (2)
PTX$L_R11                       0000003C      99      (2)
PTX$L_R2                        00000018      90      (2)     255     (2)   #-301     (2)
PTX$L_R3                        0000001C      91      (2)
PTX$L_R4                        00000020      92      (2)
PTX$L_R5                        00000024      93      (2)
PTX$L_R6                        00000028      94      (2)
PTX$L_R7                        0000002C      95      (2)
PTX$L_R8                        00000030      96      (2)
PTX$L_R9                        00000034      97      (2)
PTX$L_SSP                       00000008      86      (2)   #-245     (2)   #-246     (2)   #-247     (2)   #-291     (2)
                                                            #-292     (2)   #-293     (2)
PTX$L_USP                       0000000C      87      (2)   #-248     (2)   #-249     (2)   #-250     (2)   #-295     (2)
                                                            #-296     (2)   #-297     (2)
PTX$S_ASTLVL                    =00000003     110     (2)   #-261     (2)   #-310     (2)
PTX$S_MBZ1                      =00000002     110     (2)
PTX$S_MBZ2                      =00000005     110     (2)
PTX$S_MBZ3                      =00000009     116     (2)
PTX$S_P0LR                      =00000016     110     (2)   #-307     (2)   #-314     (2)
PTX$S_P1LR                      =00000016     116     (2)
PTX$S_PME                       =00000001     116     (2)   #-266     (2)
PTX$V_ASTLVL                    =00000018     110     (2)   #-261     (2)   #-310     (2)
PTX$V_MBZ1                      =00000016     110     (2)
PTX$V_MBZ2                      =0000001B     110     (2)
PTX$V_MBZ3                      =00000016     116     (2)
PTX$V_P0LR                      =00000000     110     (2)   #-307     (2)
PTX$V_P1LR                      =00000000     116     (2)   #-314     (2)
PTX$V_PME                       =0000001F     116     (2)   #-266     (2)   #-317     (2)
PUSH_PC                          00000081-R   190     (2)   #-567     (2)   #-659     (2)
PUSH_PSL                         00000085-R   191     (2)   #-568     (2)   #-670     (2)
PUSH_REGS                        000001DF-R   275     (2)     497     (2)     700     (2)
REG_READY                        000000BF-R   496     (2)   #-487     (2)
RETURN                           000002DF-R   710     (2)     724     (2)
SAVE_REGS                        00000168-R   213     (2)     456     (2)     585     (2)
SAVE_RESULTS                     0000016E-R   584     (2)
SCB$L_ACCESS                    00000020      75      (2)
SCB$L_ARITH                     00000034      75      (2)
SCB$L_BREAK                     0000002C      75      (2)   #-413     (2)   #-429     (2)   #-516     (2)   #-523     (2)
                                                            #-707     (2)   #-717     (2)
SCB$L_CHME                      00000044      75      (2)
SCB$L_CHMK                      00000040      75      (2)
SCB$L_CHMS                      00000048      75      (2)
```

```
SCB$L_CHMU                      0000004C      75      (2)
SCB$L_COMPAT                    00000030      75      (2)
SCB$L_KNLSTK                    00000008      75      (2)
SCB$L_MACHCHK                   00000004      75      (2)
SCB$L_OPCCUS                    00000014      75      (2)
SCB$L_OPCDEC                    00000010      75      (2)
SCB$L_POWER                     0000000C      75      (2)
SCB$L_RADRMOD                   0000001C      75      (2)
SCB$L_ROPRAND                   00000018      75      (2)
SCB$L_RXDB                      000000F8      75      (2)
SCB$L_SFTLVL1                   00000084      75      (2)
SCB$L_SFTLVL10                  000000A8      75      (2)
SCB$L_SFTLVL11                  000000AC      75      (2)
SCB$L_SFTLVL12                  000000B0      75      (2)
SCB$L_SFTLVL13                  000000B4      75      (2)
SCB$L_SFTLVL14                  000000B8      75      (2)
SCB$L_SFTLVL15                  000000BC      75      (2)
SCB$L_SFTLVL2                   00000088      75      (2)
SCB$L_SFTLVL3                   0000008C      75      (2)
SCB$L_SFTLVL4                   00000090      75      (2)
SCB$L_SFTLVL5                   00000094      75      (2)
SCB$L_SFTLVL6                   00000098      75      (2)
SCB$L_SFTLVL7                   0000009C      75      (2)
SCB$L_SFTLVL8                   000000A0      75      (2)
SCB$L_SFTLVL9                   000000A4      75      (2)
SCB$L_TBIT                      00000028      75      (2)
SCB$L_TIMER                     000000C0      75      (2)
SCB$L_TRANSL                    00000024      75      (2)
SCB$L_TXDB                      000000FC      75      (2)
SCB$L_ZERO                      00000000      75      (2)
SEP_FUNCTIONAL                  =00000002     144     (2)
SEP_REPAIR                      =00000003     144     (2)
SIZ...                          =00000008     144     (2)     110     (2)     116     (2)     144     (2)     74      (2)
TEMP_ESP                        00000097-R    195     (2)   #-409     (2)   #-419     (2)
TEMP_LONGWORD                   0000007D-R    189     (2)   #-626     (2)   #-627     (2)
TEMP_PC                         00000060-R    179     (2)   #-452     (2)   #-711     (2)
TEMP_PSL                        00000064-R    180     (2)   #-454     (2)   #-723     (2)
TEMP_R0                         00000068-R    181     (2)   #-514     (2)   #-537     (2)   #-699     (2)   #-719     (2)
TEMP_REGS                       00000000-R    177     (2)     456     (2)   #-697     (2)     700     (2)
TEMP_SSP                        0000009B-R    196     (2)   #-410     (2)   #-420     (2)
TEMP_SV_PC                      0000008F-R    193     (2)
TEMP_SV_PSL                     00000093-R    194     (2)
TEMP_USP                        0000009F-R    197     (2)   #-411     (2)   #-421     (2)
TEST_011                        0000001C-R    406     (2)     406     (2)
TEST_011_X                      00000308-R    736     (2)
TEST_CONTROL                    00000000-XR             (2)     431     (2)
TU58                            =00000002      78     (2)
UBI                             =0000000A      78     (2)
WCS                             =00000007      78     (2)
```

! Macros Cross Reference !

MACRO	SIZE	DEFINITION	REFERENCES...
$D1_BTEST	1	406 (2)	406 (2)
$D1_ERR	1	615 (2)	615 (2) 633 (2) 648 (2) 666 (2) 677 (2)
$D1_ERR_S	1	615 (2)	615 (2) 633 (2) 648 (2) 666 (2) 677 (2)
$D1_ETEST	1	736 (2)	736 (2)
$D1_SBTTL	2	324 (2)	324 (2)
$D2_BDATA	1	406 (2)	406 (2)
$D2_BTEST	1	406 (2)	406 (2)
$D2_SBTTL	1	324 (2)	324 (2)
$DEF	1	144 (2)	100 (2) 101 (2) 102 (2) 103 (2) 104 (2)
			105 (2) 111 (2) 112 (2) 117 (2) 74 (2)
			75 (2) 84 (2) 85 (2) 86 (2) 87 (2)
			88 (2) 89 (2) 90 (2) 91 (2) 92 (2)
			93 (2) 94 (2) 95 (2) 96 (2) 97 (2)
			98 (2) 99 (2)
$DEFEND	1	74 (2)	118 (2) 74 (2) 75 (2) 76 (2) 77 (2)
$DEFINI	1	74 (2)	74 (2) 75 (2) 76 (2) 77 (2) 83 (2)
$DS_BGNMOD	1	144 (2)	144 (2)
$DS_BGNTEST	1	406 (2)	406 (2)
$DS_BREAK	1	736 (2)	736 (2)
$DS_CLRVEC_S	1	429 (2)	429 (2) 523 (2) 717 (2)
$DS_DSADEF	1	74 (2)	74 (2)
$DS_DSBDEF	1	74 (2)	
$DS_ENDDATA	1	406 (2)	406 (2)
$DS_ENDMOD	1	738 (2)	738 (2)
$DS_ENDTEST	1	736 (2)	736 (2)
$DS_ENVDEF	2	144 (2)	144 (2)
$DS_ERRHARD_S	1	611 (2)	611 (2) 629 (2) 644 (2) 662 (2) 673 (2)
$DS_SBTTL	1	324 (2)	324 (2)
$DS_SCBDEF	1	75 (2)	75 (2)
$DS_SECDEF	1	78 (2)	78 (2)
$DS_SETVEC_S	1	413 (2)	413 (2) 516 (2) 706 (2)
$EQU	1	144 (2)	144 (2) 76 (2) 77 (2)
$EQULS1	1	144 (2)	144 (2)
$EQULST	1	144 (2)	144 (2)
$GBLINI	2	74 (2)	144 (2) 74 (2) 75 (2) 76 (2) 77 (2)
			83 (2)
$PRDEF	1	76 (2)	76 (2)
$PSLDEF	1	77 (2)	77 (2)
$PTXDEF	1	118 (2)	
$PUSHADR	1	615 (2)	615 (2) 633 (2) 648 (2) 666 (2) 677 (2)
$VIELD	1	74 (2)	106 (2) 113 (2) 144 (2) 74 (2)
$VIELD1	1	144 (2)	110 (2) 116 (2) 144 (2) 74 (2)
ENKAX_SECDEF	1	78 (2)	78 (2)
PUSH_R	1	134 (2)	497 (2) 700 (2)
SAVE_R	1	123 (2)	456 (2) 585 (2)

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	21	00:00:00.06	00:00:00.18
Command processing	29	00:00:00.21	00:00:00.55
Pass 1	740	00:00:09.08	00:00:12.14
Symbol table sort	0	00:00:00.49	00:00:00.50
Pass 2	197	00:00:02.20	00:00:02.51
Symbol table output	47	00:00:00.26	00:00:00.28
Psect synopsis output	4	00:00:00.02	00:00:00.02
Cross-reference output	85	00:00:01.70	00:00:02.24
Assembler run totals	1127	00:00:14.06	00:00:18.44

The working set limit was 1000 pages.
66458 bytes (130 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 365 non-local and 4 local symbols.
740 source lines were read in Pass 1, producing 23 object records in Pass 2.
70 pages of virtual memory were used to define 35 macros.

Macro library statistics

Macro library name	Macros defined
WRKD$0:[PAN.ENKAX]ENKAX.MLB;72	1
SYS$SYSROOT:[SYSLIB]DIAG.MLB;812	22
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	0
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	8
TOTALS (all libraries)	31

568 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

/LIS/CRO ENKAXL