
SE
335

SE 335

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
WEEK ONE	INTRO MEMORY STRUCTURES	MEMORY STRUCTURES	JOB TABLES LAB	FILE SYSTEM	DISPATCHER
	LUNCH				
	MEMORY STRUCTURES	MEMORY STRUCTURES	DIRECTORY LAB	FILE SYSTEM	MEMORY MANAGEMENT
	MODULES				SYSTEM REFERENCE READING

OMEWORK: READING CHAPTER 6 OF TABLES MANUAL

WEEK TWO	MEMORY MANAGEMENT	INTERRUPT SYSTEM	TERMINAL I/O	DUMP ANALYSIS	MGMT ROUND TABLE
	LUNCH				
	MEMORY MANAGEMENT INTERFACE WITH I/O SYSTEM VIRTUAL MEMORY	I/O SYSTEM	SYSTEM SUMMARY	DUMP ANALYSIS	DUMP ANALYSIS

OMEWORK: SYSTEM SUMMARY

HEWLETT hp PACKARD

HP 3000 INTRODUCTION TO MPE INTERNALS

PREREQUISITES: THE STUDENT MUST HAVE ATTENDED ALL LEVEL I S.E. COURSES. IN ADDITION, THE SE SHOULD HAVE A GOOD WORKING KNOWLEDGE OF SPL AND DATA STACKS AND HAVE AT LEAST THREE MONTHS FIELD EXPERIENCE SINCE HIS LAST S.E. COURSE.

COURSE OBJECTIVES: TO TEACH THE TECHNICALLY ORIENTED STUDENT TO BETTER WORK WITH THE MPE OPERATING SYSTEM. SPECIFICALLY, THIS COURSE WILL ENABLE THE S.E. TO ASSIST IN TECHNICAL CONSULTATION, PROBLEM ANALYSIS, AND IDENTIFICATION OF BOTH HARDWARE AND SOFTWARE PROBLEMS BY:

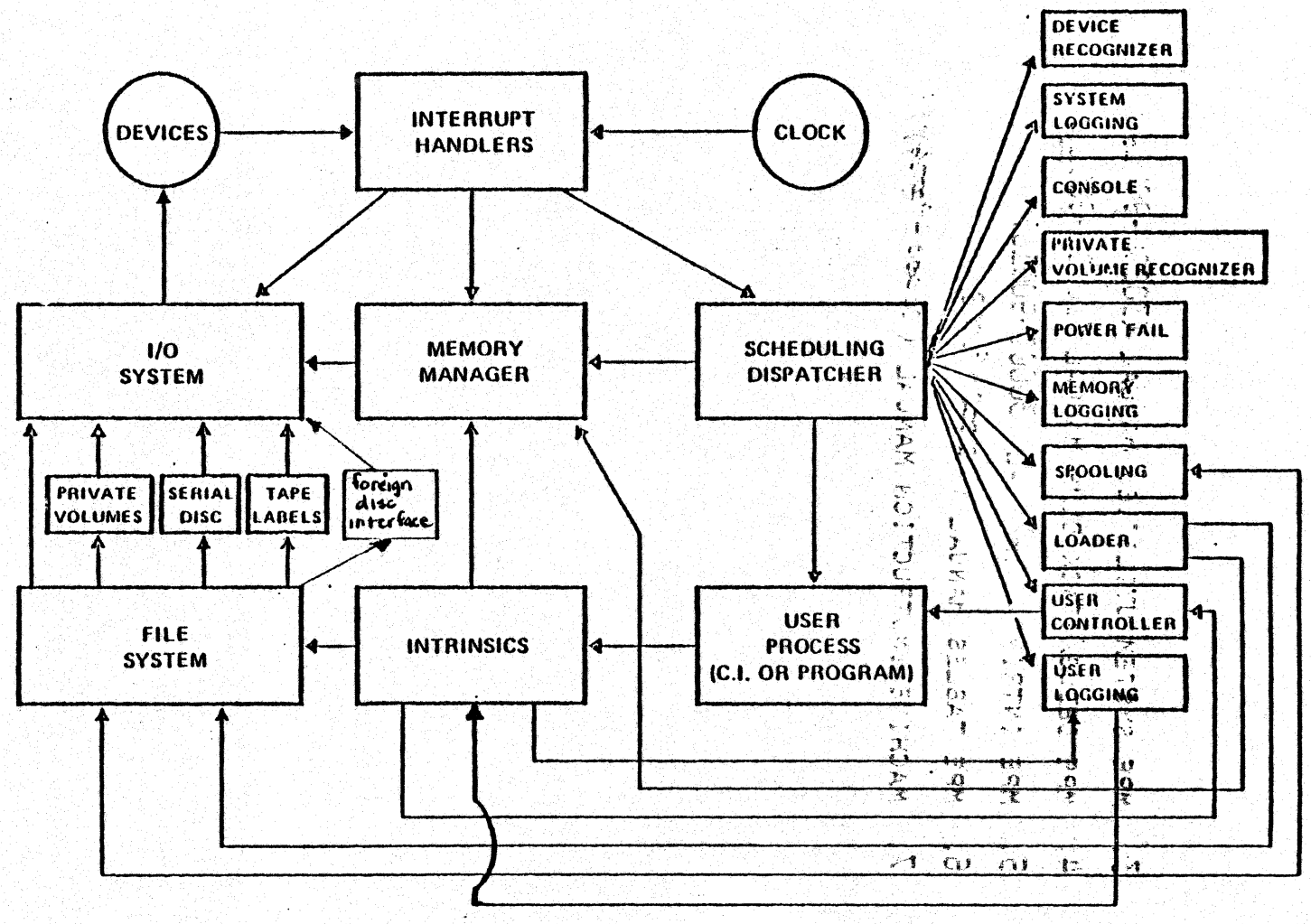
A. IDENTIFYING THE FUNCTIONS OF VARIOUS MPE MODULES TO BE ABLE TO LOCALIZE SYSTEM PROBLEMS.
B. EARNING THE FUNCTION OF CRITICAL TABLES AND THEIR STRUCTURE.
C. LEARNING A METHODOLOGY OF DUMP ANALYSIS.
D. KNOWING THE HARDWARE STRUCTURE OF THE SERIES II/III, 30/33/44 PROCESSORS.
E. SYSTEM PROGRAM AND SL INSTALLATIONS. PERFORMING SYSDUMP INSTALLATIONS OF SYSTEM PROGRAMS AND SYSTEM SL MODULES.

COURSE LENGTH: 2 WEEKS

COURSE MATERIALS:

1. SOFTWARE POCKET GUIDE (30000-90049)
2. SPL REFERENCE MANUAL (30000-90024)

3. MPE SYSTEM UTILITIES MANUAL (30000-90014)
4. MPE DEBUG/STACK DUMP MANUAL (30000-90012)
5. MPE INTRINSICS MANUAL (30000-90010)
6. MPE TABLES MANUAL (32002-90003)
7. MACHINE INSTRUCTION MANUAL (30000-90036)


DEVICES
INTERRUPT HANDLERS
CLOCK
I/O SYSTEM
MEMORY MANAGER
SCHEDULING DISPATCHER
PRIVATE VOLUMES
SERIAL DISC
TAPE LABELS
foreign disc interface
FILE SYSTEM
INTRINSICS
USER PROCESS (C.I. OR PROGRAM)
DEVICE RECOGNIZER
SYSTEM LOGGING
CONSOLE
PRIVATE VOLUME RECOGNIZER
POWER FAIL
MEMORY LOGGING
SPOOLING
LOADER
USER CONTROLLER
USER LOGGING

WHAT IS MPE?

MULTIPROGRAMMING EXECUTIVE

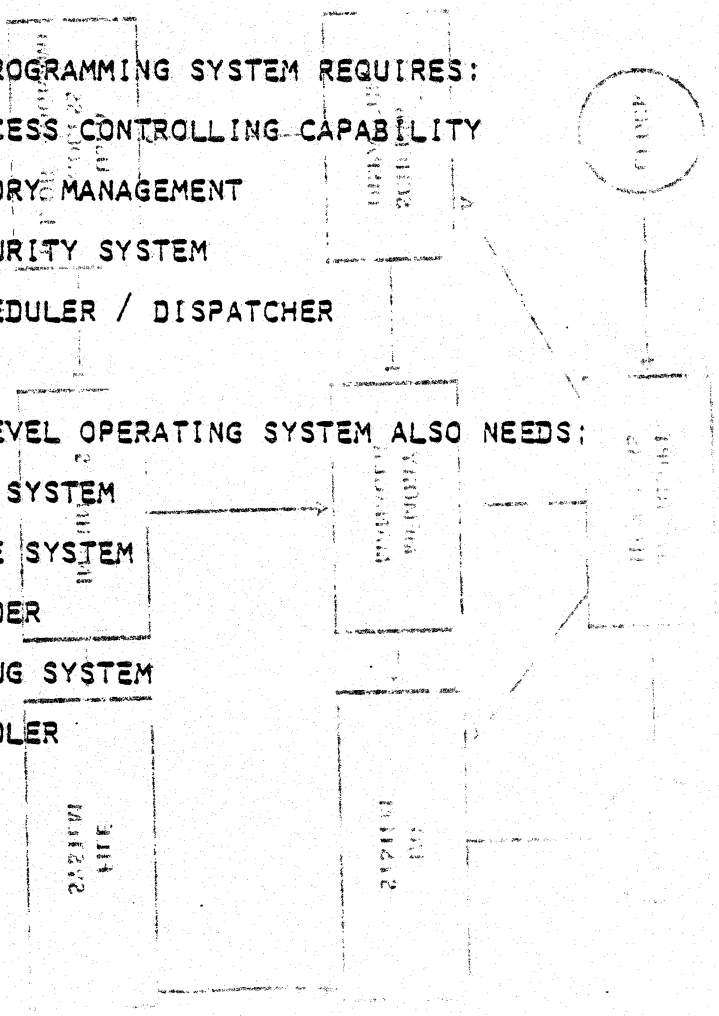
MULTIPROGRAMMING - ALLOWS MULTIPLE PROGRAMS TO RUN CONCURRENTLY.

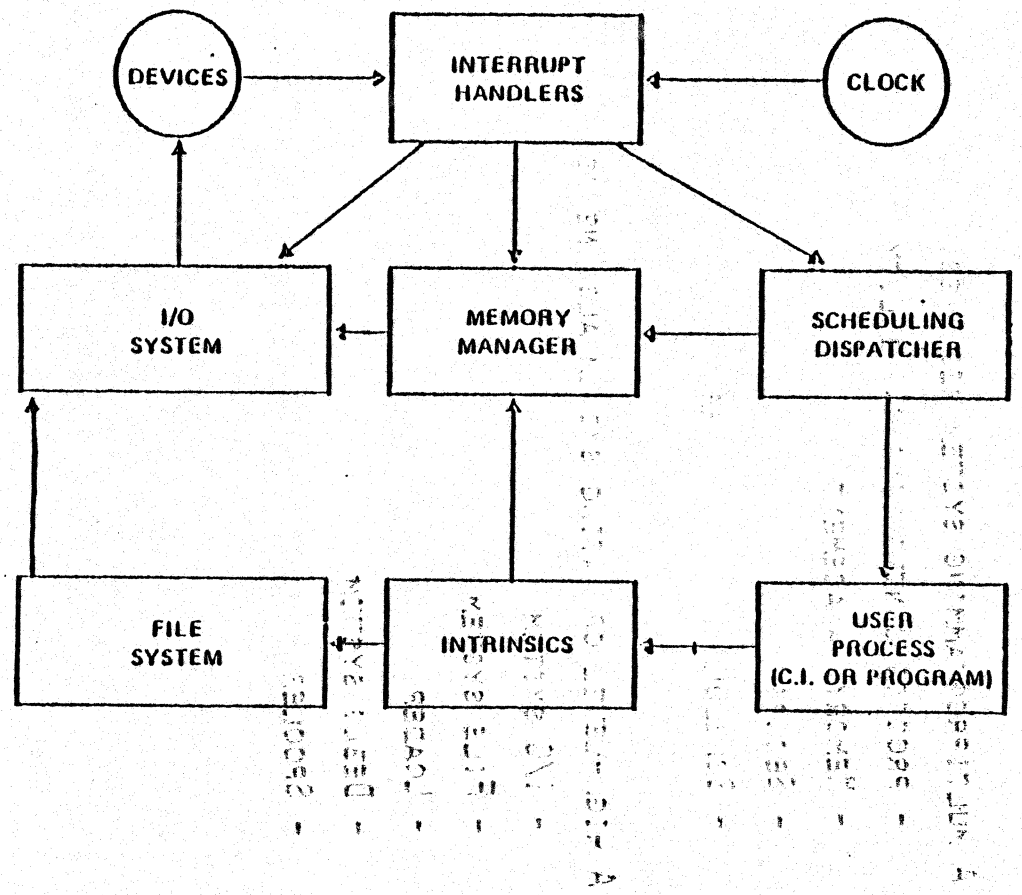
A MULTIPROGRAMMING SYSTEM REQUIRES:

- PROCESS CONTROLLING CAPABILITY
- MEMORY MANAGEMENT
- SECURITY SYSTEM
- SCHEDULER / DISPATCHER

A HIGH-LEVEL OPERATING SYSTEM ALSO NEEDS:

- I/O SYSTEM
- FILE SYSTEM
- LOADER
- DEBUG SYSTEM
- SPOOLER




DEVICES
INTERRUPT HANDLERS
CLOCK
I/O SYSTEM
MEMORY MANAGER
SCHEDULING DISPATCHER
FILE SYSTEM
INTRINSICS
USER PROCESS (C.I. OR PROGRAM)

INTERRUPT HANDLERS-

UPON INTERRUPT OF THE CPU, DIFFERENT ROUTINES ARE ACTIVATED TO SERVICE THAT INTERRUPT.

I/O SYSTEM-

CONTAINS THE CODE OF DEVICE DRIVERS. THEY HANDLE THE CODE INTERFACE BETWEEN THE DEVICE AND THE FILESYSTEM (CALLS TO ATTACHIO)

MEMORY MANAGER-

MANAGES THE CONTENTION BETWEEN VIRTUAL STORAGE (DISC) AND THE MAIN MEMORY RESOURCES.

SCHEDULING DISPATCHER

MANAGES PROCESSES THEIR PRIORITY, AND QUANTUM.
MANAGES THE AS,BS,CS,DS,& ES SUBQUEUES IN THE MULTIPROGRAMMING ENVIRONMENT.

USER PROCESS

CONSISTING OF A STACK, ONE OR MORE CODE SEGMENTS, AND SOMETIMES AN EXTRA DATA SEGMENT THE USER PROCESS IS THE USER-INTERFACE TO THE REST OF MPE.

INTRINSICS

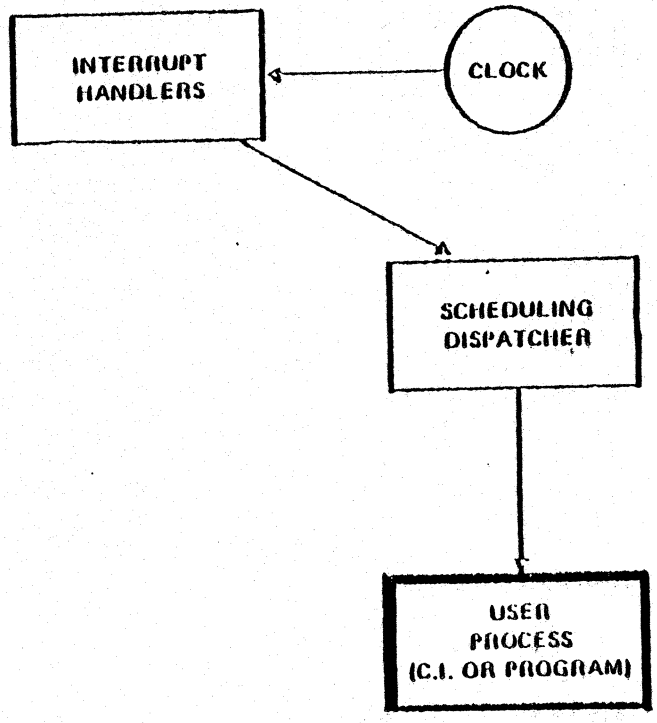
UTILITY PROCEDURES FOR INFORMATION I/O OR CONTROL OF THE MACHINE.

FILE SYSTEM

INTRINSIC INTERFACE TO PHYSICAL I/O.

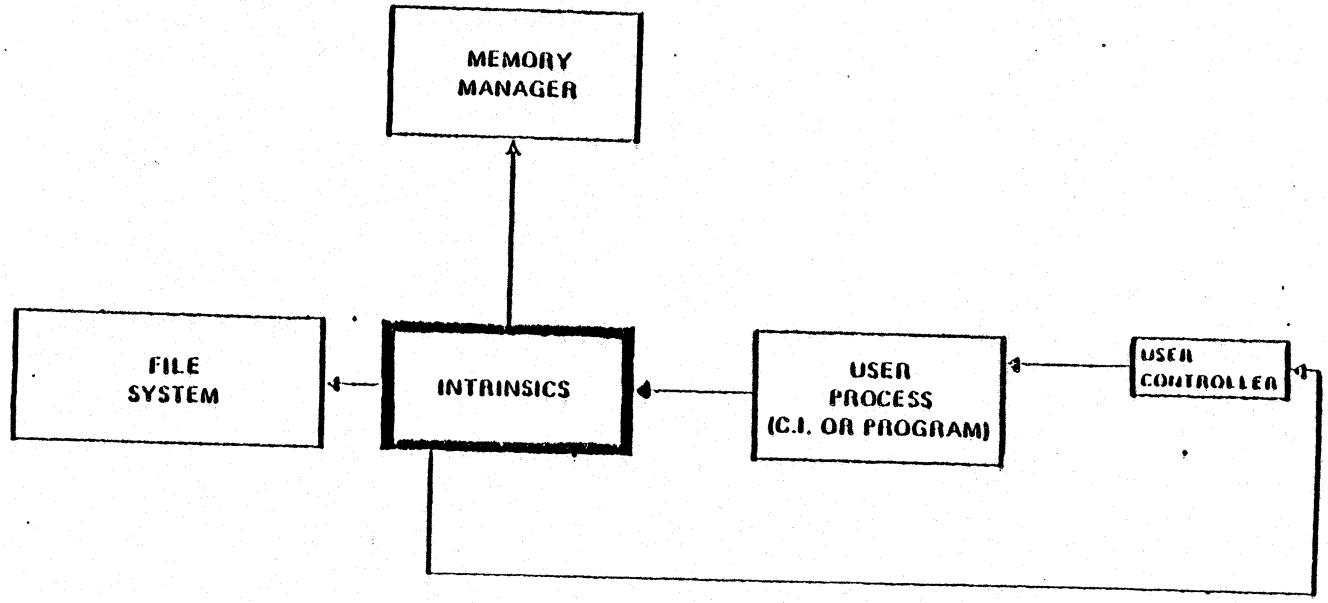
CLOCK

THE SYSTEM CLOCK IS A HARDWARE DEVICE THAT INTERRUPTS THE CPU EVERY 100 MILLI-SECONDS. WHEN AN INTERRUPT OCCURRS FROM THE CLOCK, A ROUTINE CALLED, "TICK," IS EXECUTED. TICK UPDATES SYSTEM TIME THEN ACTIVATES THE DISPATCHER FOR SCHEDULING OF PROCESSES, AT CERTAIN TIMES.


INTERRUPT HANDLERS
CLOCK
SCHEDULING DISPATCHER
USER PROCESS (C.I. OR PROGRAM)

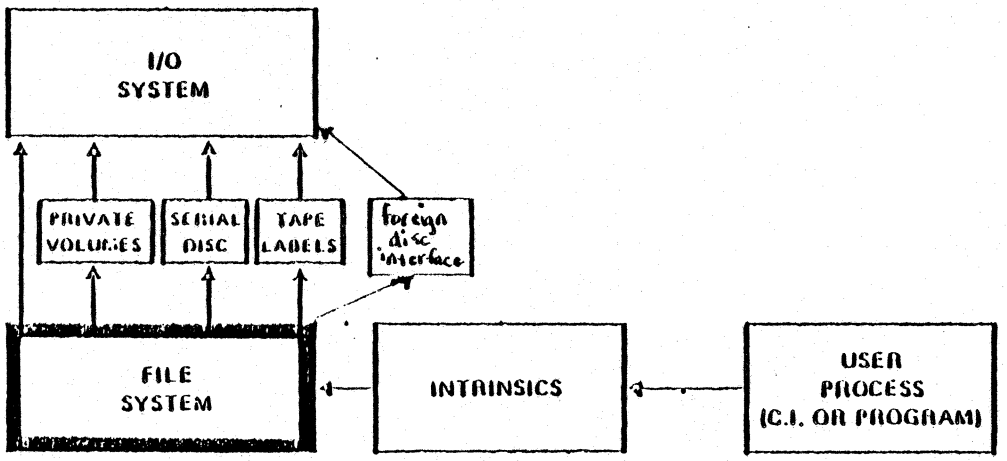
USER PROCESS--

1. A USER PROCESS IS LAUNCHED TO HAVE THE CPU.
2. EVERY 100 MILLISECONDS THE SYSTEM CLOCK GENERATES AN INTERRUPT.
3. THE INTERRUPT HANDLER FOR THE CLOCK LAUNCHES THE DISPATCHER.
4. THE DISPATCHER CHECKS FOR A HIGHER PRIORITY PROCESS TO LAUNCH.
5. IF NONE, THEN CHECKS THE CURRENT PROCESS. IF IT EXCEEDED ITS TIME SLICE, THEN ITS PRIORITY IS LOWERED.
6. THE DISPATCHER THEN LAUNCHES THE HIGHEST PRIORITY PROCESS.


MEMORY MANAGER
FILE SYSTEM
INTRINSICS
USER PROCESS (C.I. OR PROGRAM)
USER CONTROLLER

INTRINSICS

1. DIRECT USER CALLABLE PROCEDURES USED BY A PROGRAM.
2. THESE PROCEDURES CHECK ON THE VALIDITY AND CAPABILITY OF THE REQUEST.
3. THE INTRINSICS THEN MAKE CALLS TO THE FILESYSTEM (FREAD, FOPEN, ETC.) OR TO MEMORY MANAGER (GETDSEG, ALTDSEG, ETC.).
4. UPON COMPLETION, THEY RETURN TO THE USER.


I/O
SYSTEM
PRIVATE
VOLUMES
SERIAL
DISC
TAPE
LABELS
foreign
disc
interface
FILE
SYSTEM
INTRINSICS
USER
PROCESS
(C.I. OR PROGRAM)

FILE SYSTEM

1. A USER PROGRAM WISHES EITHER I/O OR SOME CONTROL OF A DEVICE.

2. THE USER PROGRAM CALLS FILE SYSTEM INTRINSICS.

3. THE FILE SYSTEM INTRINSICS CALL PROCEDURES IN THE FILE SYSTEM.

4. IF THE I/O IS TO A DEVICE OTHER THAN DISC OR A LABELED TAPE, THEN CONTROL PASSES, AFTER CHECKING FOR VALIDITY, TO THE I/O SYSTEM.

5. IF THE REQUEST IS A DISC FILE, THE FILE SYSTEM CHECKS WHERE IT IS BY WAY OF THE DIRECTORY.

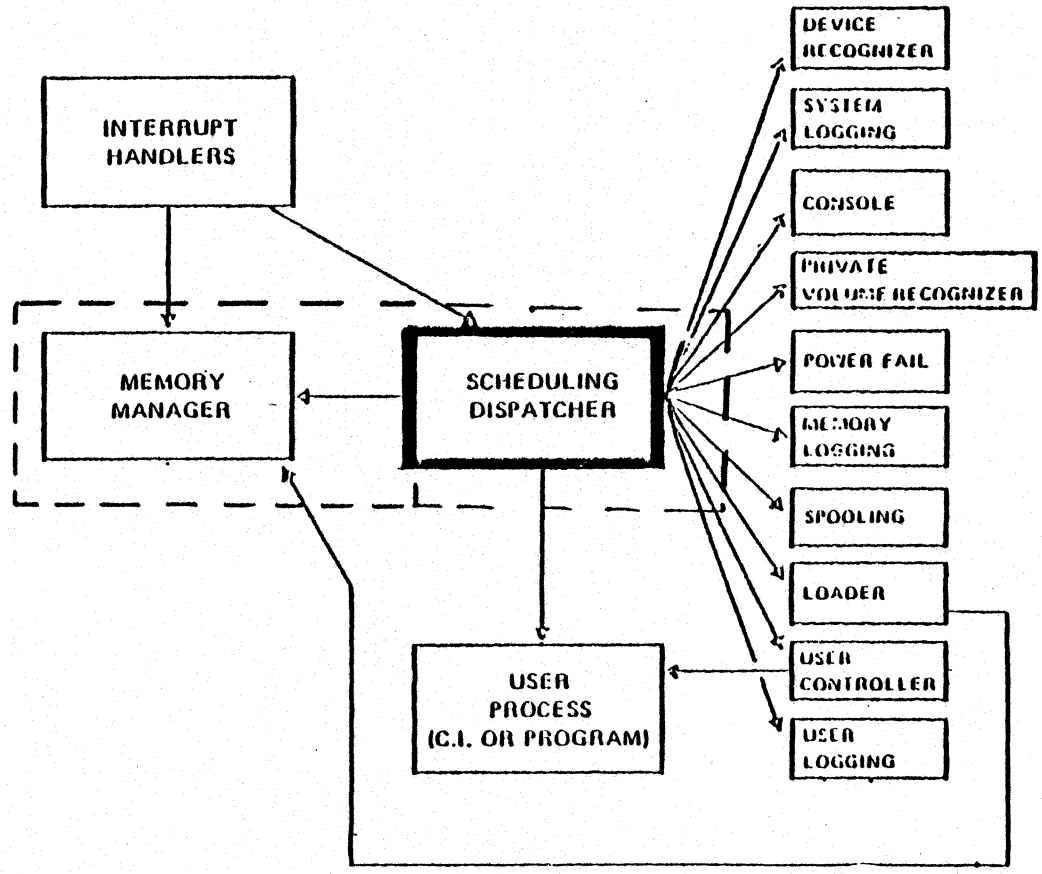
6. IF IT IS A PRIVATE VOLUME THEN THE FILE SYSTEM CHECKS TO MAKE SURE THE VOLUME IS MOUNTED, FINDS WHERE THE FILE IS LOCATED, THEN USES THE I/O SYSTEM TO MANIPULATE THE FILE.

7. IF IT IS A SERIAL DISC, FILE SYSTEM ALLOCATES THE DISC DRIVE AND WRITES OR READS SEQUENTIALLY TO THE DISC.

8. IF IT IS A LABELED TAPE, THE FILE SYSTEM CHECKS IF THE VOLUME IS MOUNTED ON THE DRIVE. IF NOT, IT REQUESTS THAT IT BE MOUNTED. IF MOUNTED, ACTIVATES THE I/O SYSTEM TO BEGIN MANIPULATING THE TAPE FILE.

9. IN ADDITION TO ALL OF THE ABOVE, THE FILE SYSTEM KEEPS TRACK OF:

   A. BEGINNING AND END OF FILES.
   B. BLOCKING AND DEBLOCKING LOGICAL AND PHYSICAL RECORDS.
   C. CURRENT RECORD POINTERS.
   D. WHICH FILES ARE OPEN.
   E. PHYSICAL CHARACTERISTICS OF THE FILES.


INTERRUPT HANDLERS
MEMORY MANAGER
SCHEDULING DISPATCHER
USER PROCESS (C.I. OR PROGRAM)
DEVICE RECOGNIZER
SYSTEM LOGGING
CONSOLE
PRIVATE VOLUME RECOGNIZER
POWER FAIL
MEMORY LOGGING
SPOOLING
LOADER
USER CONTROLLER
USER LOGGING

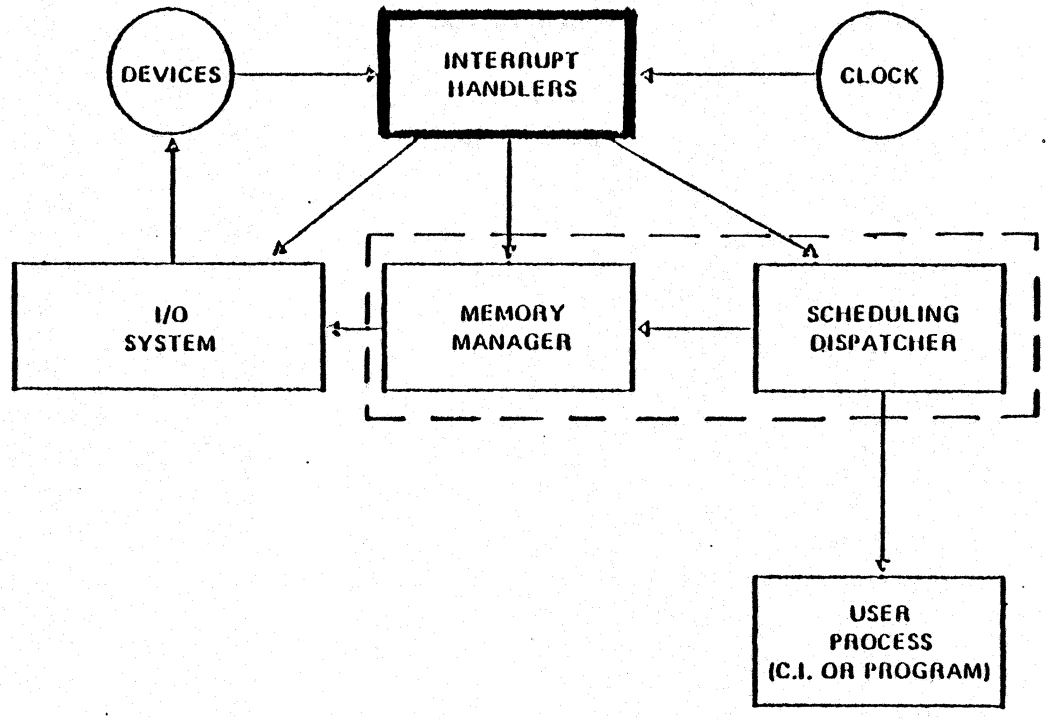
SCHEDULING DISPATCHER-

1. ALL PROCESSES IN THE SYSTEM ARE LAUNCHED VIA THE DISPATCHER.

2. DISPATCHER DETERMINES BY PRIORITY AND STATUS, (SUSPEND, ASLEEP, ETC.), WHICH PROCESS WILL RUN

3. DISPATCHER MANAGES THE PROCESS QUEUES AND THEIR MANIPULATION.

4. DISPATCHER IS ACTIVATED BY INTERRUPT HANDLERS.

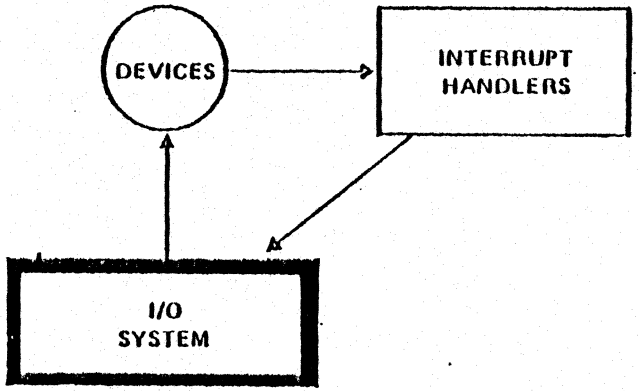

DEVICES
INTERRUPT HANDLERS
CLOCK
I/O SYSTEM
MEMORY MANAGER
SCHEDULING DISPATCHER
USER PROCESS (C.I. OR PROGRAM)

INTERRUPT HANDLERS-

ON THE 3000 THERE ARE TWO TYPES OF CPU INTERRUPTS:

1. EXTERNAL INTERRUPTS - COME FROM DEVICES OR THE SYSTEM CLOCK. A SPECIAL AREA IN LOW MEMORY CONTAINS TRANSFER POINTS DEPENDING ON THE DEVICE INTERRUPTING-

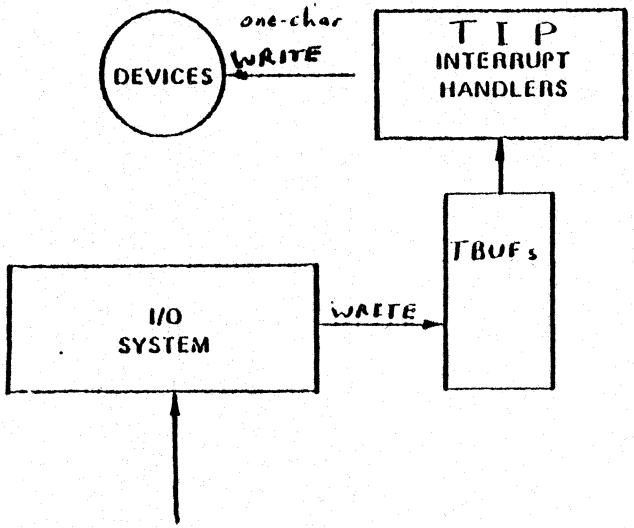
2. INTERNAL INTERRUPTS - ABNORMAL CONDITIONS DETECTED BY THE MICROCODE CAUSE CERTAIN INTERRUPT ROUTINES TO PROCESS THOSE CONDITIONS.


DEVICES
INTERRUPT HANDLERS
I/O SYSTEM

PROGRAMMED I/O OR SIO

WRITE OR READ--

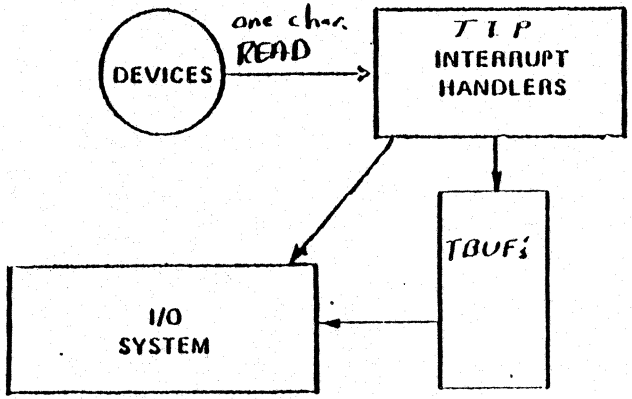
1. A REQUEST FOR A WRITE OR READ TO A DEVICE IS ISSUED.
2. I/O SYSTEM'S MONITOR AWAKENS AND INITIATES I/O.
3. THE INITIATOR ASSEMBLES AN I/O PROGRAM FOR THE DEVICE.
4. THE SELECTOR OR MULTIPLEXOR CHANNEL RUNS THE I/O PROGRAM.
5. WHEN THE I/O IS COMPLETE, AN SIO INSTRUCTION INTERRUPTS THE CPU.
6. THE INTERRUPT HANDLER AWAKES THE I/O SYSTEM'S MONITOR FOR THE DEVICE.
7. THE COMPLETOR OF THE MONITOR RETURNS TO THE USER SIGNALLING I/O COMPLETE.



Terminal Write

TERMINAL WRITE

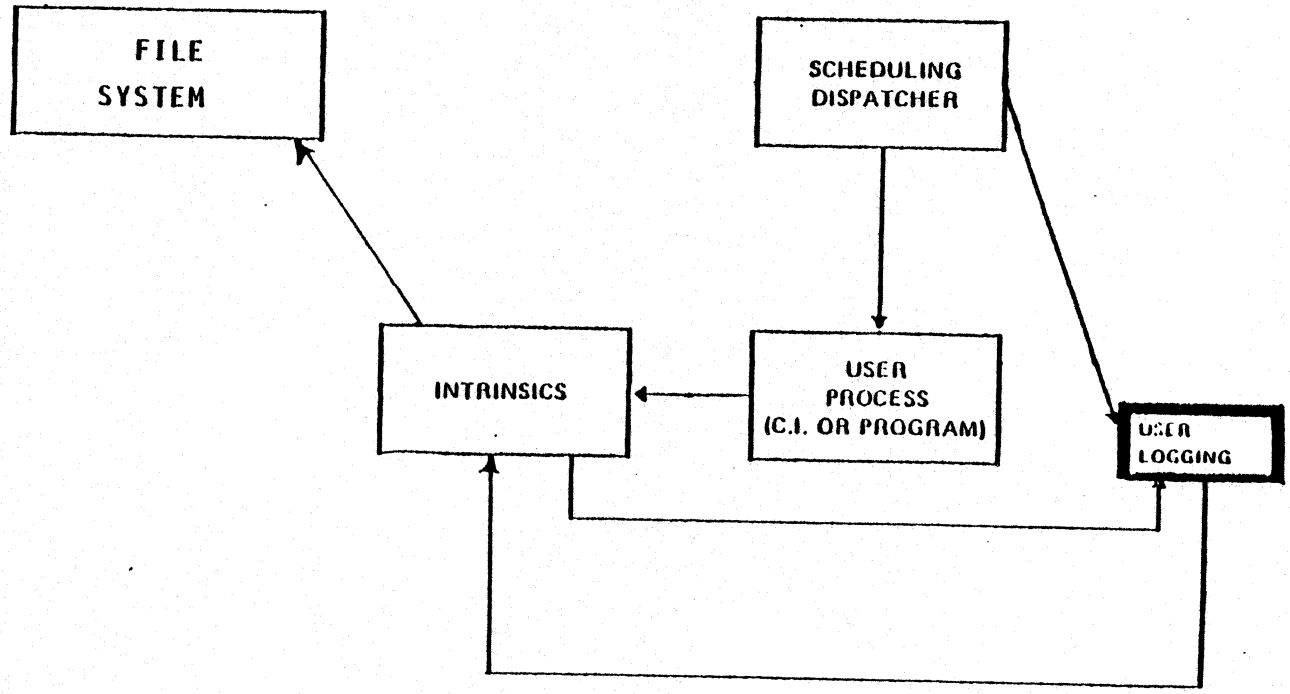
1. WRITE IS INITIATED BY THE I/O SYSTEM BY FILLING TBUF'S AND RETURNING I/O COMPLETION STATUS.
2. THE INTERRUPT SYSTEM WRITES A CHARACTER TO THE TERMINAL.
3. THE ATC THEN INTERRUPTS BACK TO CPU.
4. THE INTERRUPT HANDLER OBTAINS ANOTHER CHARACTER AND WRITES TO THE TERMINAL.
5. THIS REPEATS UNTIL ALL CHARACTERS HAVE BEEN TRANSFERRED TO THE TERMINAL.



Terminal Read

TERMINAL READ

1. USER POSTS A TERMINAL READ.
2. THE I/O SYSTEM SCHEDULES A READ FOR A TERMINAL.
3. WHEN A CHARACTER IS TYPED, THE DEVICE INTERRUPTS THE CPU.
4. INTERRUPT HANDLER STORES CHARACTER IN TBUF.
5. INTERRUPT HANDLER CHECKS IF I/O COMPLETE.
6. IF NOT, RETURNS TO I/O SYSTEM FOR ANOTHER READ. IF I/O IS COMPLETE, GOES TO DISPATCHER.
7. I/O SYSTEM PASSES DATA FROM TBUF TO USER'S STACK.


FILE
SYSTEM
SCHEDULING
DISPATCHER
INTRINSICS
USER
PROCESS
(C.I. OR PROGRAM)
USER
LOGGING

USER LOGGING-

--USED TO LOG TRANSACTIONS BY A USER.

1. TYPICAL USE IS TO RECORD TRANSACTIONS TO A FILE OR DATA BASE.

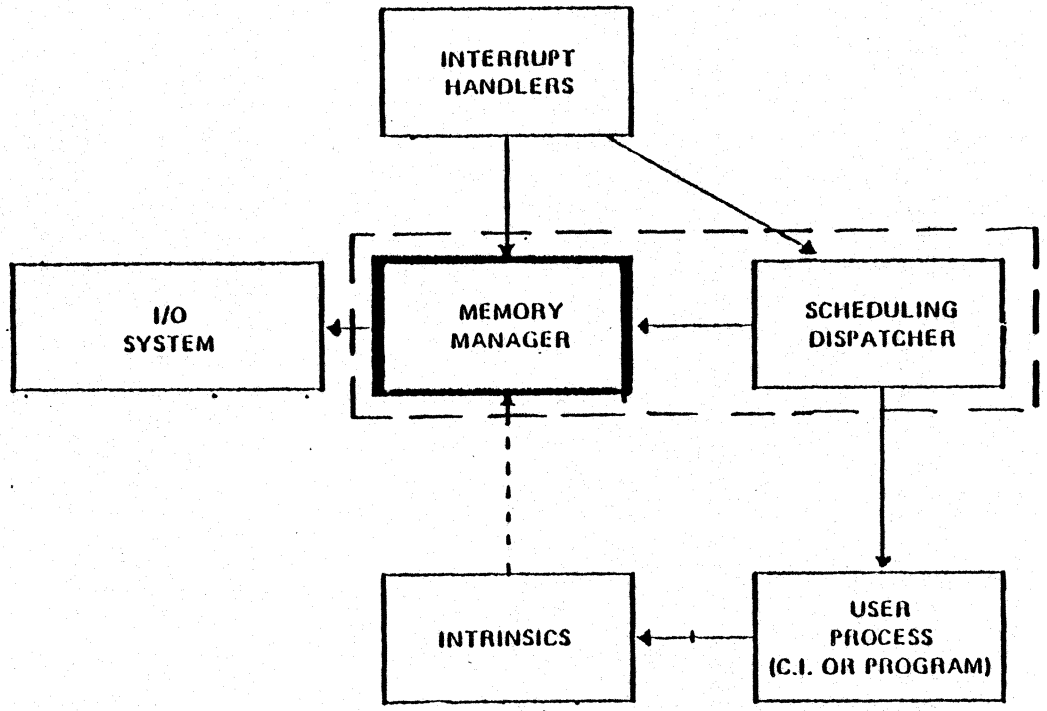
2. THE USER CAN EXECUTE INTRINSICS TO LOG TRANSACTIONS.

3. USER LOGGING GOES THROUGH THE FILE SYSTEM.

4. LOGGING BUFFERS THE REQUESTS TO LOG BY STORING THE RECORDS TO BE POSTED IN A DATA SEGMENT.

5. WHEN THE SEGMENT IS FULL OF REQUESTS, THE USER LOGGING PROCESS POSTS THEM TO TAPE OR DISC DIRECTLY THROUGH THE FILE.

6. IF A LOG BUFFER BECOMES FULL IT WILL SUSPEND A PROCESS USING IT UNTIL THE I/O CAN COMPLETE.


INTERRUPT HANDLERS
I/O SYSTEM
MEMORY MANAGER
SCHEDULING DISPATCHER
INTRINSICS
USER PROCESS (C.I. OR PROGRAM)

MEMORY MANAGER

1. MEMORY MANAGER IS A SET OF PROCEDURES THAT ARE ONLY CALLED BY DISPATCHER.

2. MAM HANDLES THE FIND OF, RESERVING, AND FILLING AVAILABLE REGIONS OF MEMORY FOR CODE AND DATA SEGMENTS.

3. MAM ALSO DOES GARBAGE COLLECTION.

4. MAM HAS ITS OWN INTERFACE TO THE I/O SYSTEM. IT DOES NOT CALL ATTACHIO.

INTRO TO MPE INTERNALS

STUDENT STUDY GUIDE

TITLE: MPE MODULES INTRODUCTION

OBJECTIVES: UPON COMPLETION OF THIS LESSON THE STUDENT SHOULD BE ABLE TO:

1. UNDERSTAND WHAT AN OPERATING SYSTEM IS AND THE FUNCTION OF ONE.
2. BE ABLE TO NARROW A SYMPTOM DOWN TO A FUNCTIONAL MODULE.

THERE ARE 4 TYPES
OF S/W INSTALLATION
TAPES

TYPE	CONTENTS	DISTRIBUTION TIME FRAME	CUSTOMER UPDATE STRATEGY
A. SYSTEM I.T.	S/W FOR ENTIRE SYSTEM CONFIGURATION EX: C.00.01 (MPE IV	ONCE A QUARTER OR WHEN APPLICABLE	MANDATORY UPDATE
B. PRODUCT I.T.	S/W FOR ENTIRE SYSTEM CONFIGURATION BUT ONLY THAT S/W FOR A PARTICULAR PRODUCT ENHANCED.	WHEN APPLICABLE EX: C.00.06 (CUB)	UPDATE ONLY THOSE CUSTOMERS THAT PURCHASED THAT PRODUCT.
C. DELTA I.T.	THIS I.T. CONTAINS ONLY FIXES TO EXISTING BUGS ON CURRENT S/W.	EVERY 2 MONTHS (60 DAYS) EX: C∆1	EVERY 2 MONTHS OR WHENEVER FSC'S DEEM NECESSARY.
D. PRODUCT TAPE	ONLY THAT S/W TO SUPPORT A PARTICULAR PRODUCT (NOT A COLD-LOAD)	WHEN APPLICABLE HP-IB EX: CARD READER PROD. TAPE	ONLY THOSE CUSTOMERS THAT PURCHASED THE PRODUCT.

POSSIBLE S/W TAPES

* SOURCE : CONTAINS SOURCE CODE FOR OPERATING SYSTEM AND INCLUDE FILES
  EX: S005002C.HP32002.SUPPORT
  & PAOP002C.HP32002.SUPPORT

* MMT : STANDS FOR MASTER MAINTENANCE TAPE. CONTAINS THE FOLLOWING FILES:
  - MOOM002C.HP32002.SUPPORT
    (THIS FILE, WHEN MERGED WITH THE CORRESPONDING SOURCE FILE AT COMPILE, WILL CREATE AN UP-TO-DATE HARDCOPY SOURCE LISTING.)
  - UOOU002C.HP32002.SUPPORT
    THIS IS THE UPDATED USL FILE FOR MODULE INITIAL.
  - POOP002C.HP32002.SUPPORT
    THIS IS THE UPDATED PROGRAM FILE FOR MODULE INITIAL.
  - JOOJ002C.HP32002.SUPPORT
    THIS IS THE JOB FILE THAT, WHEN STREAMED, WILL CREATE THE LISTING FOR MODULE INITIAL.

* I.T. : THIS IS THE S/W INSTALLATION TAPE THAT CONTAINS THE MPE OPERATING SYSTEM.
  (THIS TAPE TAKES THE PLACE OF "NOON" FILES.)

* SRB : SOFTWARE RELEASE BULLETIN; THIS TAPE CONTAINS INFO ON CURRENT STATUS OF THE S/W.

S/W TAPE DESCRIPTORS

* V. U. F.

  - FIX LEVEL
  - UPDATE LEVEL
  - VERSION LEVEL

  THESE TWO POSITIONS (NUMERICAL) ARE INCREMENTED AS S/W IS FIXED AND ENHANCED.

  (THIS LETTER POSITION IS INCREMENTED WHEN A MAJOR CHANGE TO THE S/W IS MADE.)

  EXAMPLE: C.00.02 (D-MIT)

* DATE CODE (REVISION): THIS NUMBER REPRESENTS THE WEEK OF A PARTICULAR YEAR A SPECIFIC RELEASE OF CODE FREEZES (I.E., THE FREEZE DATE IS THE TIME WHEN NO OTHER CHANGES CAN BE SUBMITTED TO THE SOURCE CODE).

  EXAMPLE: CURRENT YEAR: 1981

  BASE YEAR: 1960

  2131

  ↳ 31ST WEEK OF 1981 THIS CODE FROZEN.

  2131 IS THE DATECODE FOR C.00.02 (D-MIT)

MPE MODULES
NUMBERING CONVENTION

SXXS002C - SOURCE FILE FOR MODULE XX.

MXXM002C - MAINTENANCE FILE FOR MODULE XX.

UXXU002C - USL FILE FOR MODULE XX.

PXXP002C - PROGRAM FILE FOR MODULE XX.

JXXJ002C - JOB FILE FOR MODULE XX.

MPEMIT2 - DOCUMENTATION FILE FOR MPE.

ALL FILES ARE FOUND IN THE GROUP HP32002 OF THE SUPPORT ACCOUNT.

HP32033 IS FOR THE SERIES 33/30/40/44

HP32000 IS FOR THE SERIES I.

```
PROGRAM    SLCREF2

UNSUPPORTED

:FILE SL=SL.PUB.SYS,OLD
:FILE LIST;DEV=LP
:RUN SLCREF.2
```

PROCEDURE NAME OR ENTRY POINT

SEGMENT NAME OF WHERE CODE RESIDES

WHERE REFERENCED

ABORTDSEG	DATASEG	MORGUE
ABORTIO	NRIO	ALLOCUTIL
ABORTIOX	NRIO	FILESYS2
ABORTJOB	PINT	DSMISC
ABORTMAIL	PROCMAIL	MORGUE
ABORTPROCIO	NPIO	MORGUE
ABORTPROG	PINT	CIINIT

SLCREF

SL CROSS REFERENCE -- SL.PUB.SYS

REFERENCES = 7021 SEGMENTS = 168 ENTRY POINTS = 2607 MEMORY = 23897

ABORT	ABORTDUMP	CHECKER	FIRMWARESIM1	ININ.PUB.SYS	KERNELC	MISCSEGC
ABORT3270	TTSINTR	TTSSON.PUB.SYS				
ABORTDSEG	KERNELD	MORGUE				
ABORTIO	NRIO	ALLOCUTIL	DEVREC.PUB.SYS	DSMISC	OPHI	PROGEN.PUB.SYS
	SPOOLING					
ABORTIO3270	IOMONITOR3270	TTSINTR				
ABORTIOX	NRIO	FILESYS2	FILESYS7			
ABORTJOB	PROCSEG	DSMISC	IOTERM0.PUB.SYS	SPOOLCOMS1		
ABORTMAIL	PROCSEG	MORGUE				
ABORTPROCESS	KERNELC					
ABORTPROCIO	NRIO	MORGUE	PROCSEG	UCOP.PUB.SYS		
ABORTPROG	PROCSEG	CIINIT	CIUSERUTIL	DSSEG4		
ABORTRIN	MORGUE					
ABORTTIMEREQ	HARDRES	BIPC	CSUTILTY	HIOLPRT0.PUB.SYS	HIOLPRT1.PUB.SYS	HIOLPRT2.PUB.SYS
	HIOTAPE0.PUB.SYS	IOINP0.PUB.SYS	IOLPRT0.PUB.SYS	IOMONITOR3270	IOMPS0 PUB SYS	IOMPTRM0.PUB.SYS
	IOPRPN0.PUB.SYS	IOTAPE0.PUB.SYS	KERNELC	MPMON.PUB.SYS		
ABSOLUTESEG	DEBUG					
ACCCHECK	DIRC	FILESYS6A	LISTDIR2.PUB.SYS	RESTORE	STORE	
ACCEPT	DSSEG5	DS2026.PUB.SYS	DSTEST.PUB.SYS			
ACCEPT'FREE'C	COBLIB16					
ACCEPT'INPUT'C	COBLIB16					
ACHRLL'	CLIB'01					
ACHRLPB'	CLIB'01					
ACHRLS'	CLIB'01					
ACHRSL'	CLIB'01					
ACHRSPB'	CLIB'01					
ACHRSS'	CLIB'01					
ACTIVATE	PROCSEG	B'BASICIN	BASIC.PUB.SYS	COBEDIT.PUB.SYS	COMSYS5	DSCOPY.PUB.SYS
	DSSEG5	DSTEST.PUB.SYS	IOCDPN0.PUB.SYS	MPTEST.PUB.SYS	MRJEMON.PUB.SYS	SEGUTIL
	SPOOK.PUB.SYS	SYSDUMP.PUB.SYS	TDP.PUB.SYS	TRACPROG.PUB.SYS	TTSSON.PUB.SYS	TTSUSER.PUB.SYS
	XRTCGEN.PUB.SYS					
ADDASS	OPHI	OPMED				

PROCEDURE SEGMENT segments or (program files).pub.sys that call or refs procedure

FORMUSERID	CIUSERUTIL					
FPOINT	FILESYS2	APL.PUB.SYS	B'FILEIO	BASIC.PUB.SYS	COBLIB18	COBOLII.PUB.SYS
	CSDUMP.PUB.SYS	DBRECOV.PUB.SYS	DSCOPY.PUB.SYS	DSM.PUB.SYS	DSSEG4	EDITOR.PUB.SYS
	FORMSPEC.PUB.SYS	IMAGE08	KSAM01	KSAM05	KSAMUTIL.PUB.SYS	LOGSEG0
	OPMED	QUERY.PUB.SYS	RPG'IMAGE'IO	RPG'INIT	RPG'IO	SCRIBE.PUB.SYS
	SEGPROC.PUB.SYS	SEGUTIL	STORE	SYSDUMP.PUB.SYS	TDP.PUB.SYS	TRACPROG.PUB.SYS
	TTSCONFG	TTSMON.PUB.SYS	V3000'6	XL2100.PUB.SYS		
FPROCTERM	FILESYS7	MORGUE	OPHI	TTSMON.PUB.SYS		
FQUIESCE'IO	FILESYS1A	FILESYS2	FILESYS3			
FQUIESCEIO	FILESYS5	FILESYS7				
FREAD	FILESYS1A	APL.PUB.SYS	ASOCTABL.PUB.SYS	B'BASICIN	B'FILEIO	BASIC.PUB.SYS
	BASICOMP.PUB.SYS	BUILDINT.PUB.SYS	CIPREPRUN	CLIB'03	COBCNV.PUB.SYS	COBEDIT.PUB.SYS
	COBLIB06	COBLIB18	COBOL.PUB.SYS	COBOLII.PUB.SYS	COMSYS2	CSDUMP.PUB.SYS
	CSLIST.PUB.SYS	CXSTOREST	DBDRIVER.PUB.SYS	DBDUMP.PUB.SYS	DBLOAD.PUB.SYS	DBRECOV.PUB.SYS
	DBRESTOR.PUB.SYS	DBSCHEMA.PUB.SYS	DBSTORE.PUB.SYS	DBUNLOAD.PUB.SYS	DBUTIL.PUB.SYS	DCAINTR
	DEL'OPEN'CLOSE	DEL'TERM'I'O	DNLDSYS	DNLDUSER	DOWNLOAD	DPAN2.PUB.SYS
	DPAN4.PUB.SYS	DS2026.PUB.SYS	DSCOPY.PUB.SYS	DSM.PUB.SYS	DSSEG3	DSSEG4
	DSTEST.PUB.SYS	EDITOR.PUB.SYS	FCOPY.PUB.SYS	FORMGEN.PUB.SYS	FORMSPEC.PUB.SYS	FORTRAN.PUB.SYS
	HELPUSER	IMAGE08	IOCDPNO.PUB.SYS	KSAM06	KSAM07	LINETEST.PUB.SYS
	LISTLOG2.PUB.SYS	LOADER1	LOGSEG0	MAKECAT.PUB.SYS	MERGE.PUB.SYS	MERGELIB
	MPCONFIG.PUB.SYS	MPMON.PUB.SYS	MPTEST.PUB.SYS	MRJE.PUB.SYS	MRJELOGR.PUB.SYS	MRJEMISC2
	MRJEMON.PUB.SYS	MRJEOUT.PUB.SYS	OPLOW	OPMED	OPT.PUB.SYS	PATCH.PUB.SYS
	PSLCREF.PUB.SYS	QUERY.PUB.SYS	RECOVER2.PUB.SYS	REFORMAT.PUB.SYS	REFSPEC.PUB.SYS	RESTORE
	RESTORE.PUB.SYS	RJE.PUB.SYS	RPG'INIT	RPG'IO	RPG.PUB.SYS	SCRIBE.PUB.SYS
	SEGUTIL	SORT.PUB.SYS	SORTLIB	SPL.PUB.SYS	SPOOK.PUB.SYS	SPOOLING
	STORE	SYSDUMP.PUB.SYS	TDP.PUB.SYS	TRACE0'	TRACE1'	TRACPROG.PUB.SYS
	TTSCONFG	TTSSON.PUB.SYS	TTSUSER.PUB.SYS	UTILITY1	UTILITY2	V3000'1
	V3000'3	V3000'6	XA2100.PUB.SYS	XL2100.PUB.SYS	XRTCGEN.PUB.SYS	XSORT.PUB.SYS
FREADBACKWARD	FILESYS1A					
FREADBYKEY	KSAM02	APL.PUB.SYS	COBLIB18	DSSEG4	FORMSPEC.PUB.SYS	KSAM06
	KSAM07	REFORMAT.PUB.SYS	REFSPEC.PUB.SYS	RPG'IMAGE'IO		
FREADC	KSAM02	CONVERT.PUB.SYS	DSSEG4	FCOPY.PUB.SYS	RPG'IO	
FREADDIR	FILESYS1A	APL.PUB.SYS	APLSEG01	ASOCTABL.PUB.SYS	B'FILEIO	BASIC.PUB.SYS
	BASICOMP.PUB.SYS	BUILDINT.PUB.SYS	CIORGMAN	CLIB'03	COBCNV.PUB.SYS	COBLIB06
	COBLIB18	COBOL.PUB.SYS	COBOLII.PUB.SYS	COMSYS3	CSDUMP.PUB.SYS	DBDRIVER.PUB.SYS
	DBRECOV.PUB.SYS	DBRESTOR.PUB.SYS	DBSTORE.PUB.SYS	DBUTIL.PUB.SYS	DEL'EDITS	DEL'FORM'I'O
	DEL'OPEN'CLOSE	DHANDLE.PUB.SYS	DPAN2.PUB.SYS	DPAN4.PUB.SYS	DSSEG4	DSTEST.PUB.SYS
	EDITOR.PUB.SYS	FORMAINT.PUB.SYS	FORMGEN.PUB.SYS	FORTRAN.PUB.SYS	HELPUSER	IMAGE03
	IMAGE04	IMAGE05	IMAGE06	IMAGE07	IMAGE08	IMAGE11
	INPDPAN.PUB.SYS	KSAM01	KSAM02	KSAM03	KSAM04	KSAMUTIL.PUB.SYS
	LOAD.PUB.SYS	LOADER1	LOGSEG0	MAKECAT.PUB.SYS	MEMLOGAN.PUB.SYS	MEMLOGP.PUB.SYS
	MRJE.PUB.SYS	MRJELOGR.PUB.SYS	MRJEMISC2	MRJEOUT.PUB.SYS	OPHI	PATCH.PUB.SYS
	PSLCREF.PUB.SYS	QUERY.PUB.SYS	RJE.PUB.SYS	RPG'IO	RPG.PUB.SYS	SCRIBE.PUB.SYS
	SEGPROC.PUB.SYS	SEGUTIL	SLPATCH.PUB.SYS	SORTLIB	SPL.PUB.SYS	SPOOK.PUB.SYS
	SPOOLCOMS1	SPOOLING	SYSDUMP.PUB.SYS	TDP.PUB.SYS	TTSMON.PUB.SYS	UDC
	USER	V3000'5	V3000'6	V3000'7		
FREADKEY	KSAM04	KSAMUTIL.PUB.SYS				
FREADKEYNEXT	KSAM04					
FREADLABEL	FILESYS3	APL.PUB.SYS	COBLIB02	COBLIB06	COBLIB18	DBDUMP.PUB.SYS
	DBRESTOR.PUB.SYS	DBSTORE.PUB.SYS	DBUTIL.PUB.SYS	DSCOPY.PUB.SYS	DSSEG4	EDITOR.PUB.SYS
	FCOPY.PUB.SYS	FILESYS6A	HELPUSER	IMAGE01	IMAGE03	IMAGE04
	IMAGE05	IMAGE06	IMAGE07	IMAGE08	IMAGE09	IMAGE11

T	DESCRIPTION	MPE V.U.F.	MPE FIXES
1646		A.00.05	
1701		A.01.00	
1705			
1709		A.01.01	
1728		A.01.FX	
1737	SPECIAL DS	A.01.02	
1745	SPECIAL MRJE		
1814		B.00.00	1- 66
1331		B.00.01	67- 134
1906		B.00.02	135- 498
1912	SERIES 33 RELEASE	B.00.02	135- 789
1913		B.01.00	500- 789
TACOMM		B.01.DC	790- 792
ATHENA		B.01.01	800-1282
BRUNO		B.01.01	1300- 499
CHEETAH	PRERELEASE	C.00.00	1500-2055
CHEETAH		C.00.01	2056-2097
C-DELTA	FIRST FIX MIT	C.00.03	2300-2381
D-MIT	FIRST EPOC/7976	C.00.02	2500-2730
S.64	GEMINI MIT	C.00.08	3000-3048
CUB	LINUS	C.00.06	3500
D-DELTA1		C.00.04	2800-2874

GROUPS IN SUPPORT ACCOUNT

HP30010	IML
HP30020	IML
HP30126	CALCOMP PLOTTER
HP30130	RJE/3000
HP30131	CS
HP30301	PROG CONTROLLER
HP30361	PROG CONTROLLER
HP32002	MPE III
HP32050	SEGMENTER
HP32100	SPL
HP32101	BASIC
HP32102	FORTRAN
HP32103	BASIC COMPILER
HP32104	RPG
HP32105	APL
HP32150	BUILDINT
HP32190	DS/3000
HP32192	MRJE
HP32193	MTS
HP32199	DISCCOPY
HP32201	EDITOR
HP32204	STAR
HP32205	SCIENTIFIC LIBRARY
HP32206	DEL/3000
HP32207	INDEX
HP32208	KSAM
HP32209	V/3000
HP32211	COMPILER LIBRARY
HP32212	FCOPY
HP32213	COBOL
HP32214	SORT/MERGE
HP32215	IMAGE
HP32216	QUERY
HP32222	TRACE
HP32223	XA2100
HP32226	XL2100
HP32229	IML
HP32230	DIAGNOSTICS(SERIES II AND III)
HP32231	DIAGNOSTICS(SERIES 30 /33)
HP32232	COBOL LIBRARY
HP32233	COBOL II
HP32238	OPT/3000
HP32243	DATACAPTURE
HP32260	MATERIALS MANAGEMENT 3000
HP32900	SIS/3000
HP32901	SAS/3000
HP32902	CIS/3000
HP36578	TDP/3000
HPONLN	DIAGNOSTICS(SERIES I)
HPOFFLN	DIAGNOSTICS(SERIES I)
UPGRADE	SERIES I TO II CONVERSION
CREATOR	INSTALLATION UTILITIES

HEWLETT hp PACKARD

INCLUDE FILES

FILES CONTAINING SOURCE USED BY A NUMBER OF MODULES

USED AT COMPILE WITH SOURCE MODULES

REFERENCED IN SOURCE MODULE BY STATEMENT

$INCLUDE include file name

MODULE NO.	MODULE NAME	TYPE OF SYSTEM	INCLIO PA0P00C2C	INCLVMC PA1P00C2C	INCLPCB PA2P00C2C	INCLSLL PA3P00C2C	INCLST PA4P00C2C	INCLMEAS PA5P00C2C	INCLVMLD PA6P00C2C	INCLICS PA7P00C2C	INCLHARD PA8P00C2C	INCLMSG PA9P00C2C	RINSINCL PB0P00C2C
00	INITIAL			o					o				
09	PROGEN									o			
10	ININ	III			o		o	o		o		o	
10	ININ	33			o		o	o		o		o	
27	IOMDISC1							o					
55	HARDRES	III	o					o			o	o	
55	HARDRES	33	o					o			o	o	
60	PROCSEG									o			o
63	PCREATE				o	o	o	o		o			
64	MORGUE												o
73	RINS							o					o
85	OPCOMMND									o			
92	KERNELC		o		o	o	o	o		o		o	
93	KERNELD			o	o	o	o	o	o			o	
95	MISCSEGC				o		o						
96	MEASSEG							o					

```
00000001  00000 0   $CONTROL MAP,CODE                                                <<01040>>
00000002  00000 0   << ININ -- MODULE 10 >>                                          <<01040>>
00000003  00000 0   << SUN MAY 6, 1979 4:45 >>                                       <<01040>>
00000090  00000 0   <<COPYRIGHT     "(C) COPYRIGHT HEWLETT-PACKARD CO. 1976. ",          >>
00000091  00000 0   <<      "THIS PROGRAM MAY BE USED WITH ONE COMPUTER SYSTEM AT A ",    >>
00000092  00000 0   <<      "TIME AND SHALL NOT OTHERWISE BE RECORDED, TRANSMITTED OR ",  >>
00000093  00000 0   <<      "STORED IN A RETRIEVAL SYSTEM.  COPYING OR OTHER REPRODUCTION ">>
00000094  00000 0   <<      "OF THIS PROGRAM EXCEPT FOR ARCHIVAL PURPOSES IS PROHIBITED ", >>
00000095  00000 0   <<      "WITHOUT THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.">>
00000863  00000 0   <<" (C) COPYRIGHT HEWLETT-PACKARD COMPANY,
00000864  00000 0     1977.  ALL RIGHTS RESERVED.  NO PART OF
00000865  00000 0     THIS PROGRAM MAY BE PHOTOCOPIED, REPRO-
00000866  00000 0     DUCED OR TRANSLATED TO ANOTHER PROGRAM
00000867  00000 0     LANGUAGE WITHOUT THE PRIOR WRITTEN CON-
00000868  00000 0     SENT OF HEWLETT-PACKARD COMPANY.">>
00001000  00000 0   $CONTROL MAIN=ININ
00002000  00000 0   $THIRTY
00003000  00000 0   BEGIN
00004000  00000 1   <<
00005000  00000 1
00006000  00000 1         INTERNAL INTERRUPT SEGMENT
00007000  00000 1
00008000  00000 1   >>
00009000  00000 1
00010000  00000 1   DEFINE  ASMB = ASSEMBLE#,
00011000  00000 1           F = ABSOLUTE#,
00011100  00000 1           PDISABLE=ASMB(PSDB)#,                                    <<JB.IV>>
00011200  00000 1           PENABLE=ASMB(PSEB)#,                                     <<JB.IV>>
00012000  00000 1           DISABLE = ASMB( SED 0 )#,
00013000  00000 1           ENABLE = ASMB( SED 1 )#,                                 <<01040>>
00013100  00000 1           LMEM = ASMB( LSEA )#,                                    <<01040>>
00013150  00000 1         LRV = ABS(%17).(1:15)#;                                    <<JB.IV>>
          00000 1   $EDIT                                                           <<JB.IV>>
00014000  00000 1   $INCLUDE MEASINCL                                               <<JB.IV>>
00001000  00000 1
00002000  00000 1   <<*************************************************************
00003000  00000 1   *  The following equates came from the SINCLUDE file named   *
00004000  00000 1   *                                                             *
00005000  00000 1   *                      MEASINCL                               *
00006000  00000 1   *                                                             *
00007000  00000 1   *************************************************************
00008000  00000 1   *  Definition of # of Classes, subclasses, subclass sizes     *
00009000  00000 1   *  NOTE: This information is maintained in the Tables Manual, *
00010000  00000 1   *  any changes should be reflected in the manual               *
00011000  00000 1   *************************************************************>>
00012000  00000 1
00013000  00000 1
00013010  00000 1   EQUATE LDEVTABSIZE       = 256,
00014000  00000 1          CLASSCOUNT        = 2,
00015000  00000 1          GLOBAL'HEADSIZE   = CLASSCOUNT+1,
00016000  00000 1          STD'SUBCLASS'OVHD = 3,        << 3 WORD ENTRY 0 >>
00017000  00000 1          CLASS0'SUBCLASSCNT= 4,        << GLOBAL STATS, DISC STATS >>
00018000  00000 1          CLASS0'HEADSIZE   = CLASS0'SUBCLASSCNT+1,
00019000  00000 1          COSUB0'SEGRELOFF  = LDEVTABSIZE+      <<THIS OFFSET POINTS>>
00019010  00000 1                              GLOBAL'HEADSIZE+  <<TO FIRST ITEM OF >>
00020000  00000 1                              CLASS0'HEADSIZE+  <<SUBCLASS,KERNEL   >>
00021000  00000 1                              STD'SUBCLASS'OVHD,<<USE THIS TO UPDATE>>
```

```
00023000  00000 1                                                <<GLOBAL COUNTERS   >>
00025000  00000 1          CLASS0'SUB0SIZE   = 100,              << 0-29 PROC LAUNCH/STOP >>
00026000  00000 1                                                << 30-49 PAUSE >>
00027000  00000 1                                                << 50-69 SWAP >>
00028000  00000 1                                                << 70-79 CPU USAGE >>
00029000  00000 1                                                << 80-99 MISC COUNTERS >>
00030000  00000 1          CLASS0'SUB1SIZE   = 20,               << DISC ACTIVITY >>
00030100  00000 1          CLASS0'SUB2SIZE   = 2,                << LP ACTIVITY >>
00030300  00000 1          CLASS0'SUB3SIZE   = 4,                << TAPE ACTIVTY >>
00031000  00000 1          COSUB1'SEGRELOFF  = COSUB0'SEGRELOFF+ <<POINTS TO ITEM 0  >>
00032000  00000 1                              CLASS0'SUB0SIZE+  <<OF CLASS0 SUB 1   >>
00033000  00000 1                              STD'SUBCLASS'OVHD,
00033001  00000 1          COSUB2'SEGRELOFF  = COSUB1'SEGRELOFF+ <<POINTS TO ITEM 0  >>
00033002  00000 1                              CLASS0'SUB1SIZE+  <<OF CLASS0 SUB 2   >>
00033003  00000 1                              STD'SUBCLASS'OVHD,
00033004  00000 1          COSUB3'SEGRELOFF  = COSUB2'SEGRELOFF+ <<POINTS TO ITEM 0  >>
00033005  00000 1                              CLASS0'SUB2SIZE+  <<OF CLASS0 SUB 3   >>
00033006  00000 1                              STD'SUBCLASS'OVHD,
00033010  00000 1          CLASS1'SUBCLASSCNT= 6,
00034000  00000 1          CLASS1'SUB0SIZE   = 44,       << CPU BURST >>
00035000  00000 1          CLASS1'SUB1SIZE   = 44,       << CPU PAUSE >>
00036000  00000 1          CLASS1'SUB2SIZE   = 44,       << THINK TIME >>
00037000  00000 1          CLASS1'SUB3SIZE   = 44,       << RESPONSE TIME DISTRIBUTION >>
00038000  00000 1          CLASS1'SUB4SIZE   = 50,       << CPU QUEUE INFORMATION >>
00039000  00000 1          CLASS1'SUB5SIZE   = 68,       << INTERNAL SYNC RESOURCE STATS >>
00040000  00000 1
00041000  00000 1          MEASSIR           = 24,
00042000  00000 1
00042200  00000 1          NUMDEVTYPES       = 3;    <<NUM INTF KNOWS ABOUT>>
00042300  00000 1
00043000  00000 1   DEFINE SYSMEASINFO'TABIDX = ABSOLUTE(%1261)#,
00044000  00000 1          GCLASSENABLEDMASK  = ABSOLUTE(%1262)#,
00045000  00000 1          MEASSTATXDSBANK    = ABSOLUTE(%1263)#,
00046000  00000 1          MEASSTATXDSBASE    = ABSOLUTE(%1264)#;
00047000  00000 1
00048000  00000 1   DEFINE CLASS0  = (0:1)#,
00049000  00000 1          CLASS1  = (1:1)#,
00050000  00000 1          CLASS2  = (2:1)#,
00051000  00000 1          CLASS3  = (3:1)#,
00052000  00000 1          CLASS4  = (4:1)#,
00053000  00000 1          CLASS5  = (5:1)#,
00054000  00000 1          CLASS6  = (6:1)#,
00055000  00000 1          CLASS7  = (7:1)#,
00056000  00000 1          CLASS8  = (8:1)#,
00057000  00000 1          CLASS9  = (9:1)#,
00058000  00000 1          CLASS10 = (10:1)#,
00059000  00000 1          CLASS11 = (11:1)#,
00060000  00000 1          CLASS12 = (12:1)#,
00061000  00000 1          CLASS13 = (13:1)#,
00062000  00000 1          CLASS14 = (14:1)#;
00063000  00000 1
00063100  00000 1   DEFINE CFIELD = (0:4)#,    <<CLASS FIELD OF LDEVTAB>>
00063200  00000 1          SFIELD = (4:4)#,    <<SUBCLASS FIELD OF LDEVTAB>>
00063300  00000 1          EFIELD = (8:8)#;    <<ENTRY FILED OF LDEVTAB>>
00063340  00000 1
00064000  00000 1   <<*****************************************************
00065000  00000 1   *                                                     *
```

```
00066000  00000 1   *                  END OF MEASINCL FILE                        *
00067000  00000 1   ************************************************************>>
00015000  00000 1   EQUATE  PCBSIZE = 16,
                    .
***** WARNING #001 *****  SEQUENCE ERROR

00016000  00000 1            CPCB = 4,
00017000  00000 1            PCBB = 3,
00017100  00000 1            DSTB = 2,                                          <<01040>>
00018000  00000 1             TRACELABEL = %1257,
00019000  00000 1            QI = 5,
00020000  00000 1            ZI = 6;
00020100  00000 1   EQUATE ICSPOISABLECNTCELL=18;                               <<JB.IV>>
00020200  00000 1                                                               <<JB.IV>>
00021000  00000 1
00022000  00000 1   INTEGER  X = X,
00023000  00000 1            P = Q-2,     << STACK MARKER P VALUE >>
00024000  00000 1            DELTAQ = Q+0,
00025000  00000 1            STATUS = Q-1,<< STACK MARKER STATUS >>
00026000  00000 1            PARAM = Q+1; << INTERRUPT PARAMETER >>
00026001  00000 1   EQUATE ICS'QI=5;                                             <<JB.IV>>
00026010  00000 1   EQUATE SYSBASE=%1000,                                        <<JB.IV>>
00026011  00000 1          SLIX=1,                                              <<JB.IV>>
00026012  00000 1          SYSXL=SYSBASE+SLIX,                                  <<JB.IV>>
00026020  00000 1          DSTIX=2,                                             <<JB.IV>>
00026030  00000 1          SYSDST=SYSBASE+DSTIX,                                <<JB.IV>>
00026040  00000 1          PCBIX=3,                                             <<JB.IV>>
00026050  00000 1          SYSPCB=SYSBASE+PCBIX,                                <<JB.IV>>
00026051  00000 1          DFCIX=%32,                                           <<JB.IV>>
00026052  00000 1          SYSDFC=SYSBASE+DFCIX,                                <<JB.IV>>
00026060  00000 1          XDSTIX=7,                                            <<JB.IV>>
00026070  00000 1          SYSXDSTIX=SYSBASE+XDSTIX,                            <<JB.IV>>
00026071  00000 1          CSTXBLKIX=%51,                                       <<JB.IV>>
00026072  00000 1          SYSCSTXBLK=SYSBASE+CSTXBLKIX,                        <<JB.IV>>
00026073  00000 1          SYSWAITTODISPMSG=%1053;                              <<JB.IV>>
00026074  00000 1                                                               <<JB.IV>>
00026075  00000 1   DEFINE MEMTRAPFLAG=(4:1)#;                                  <<JB.IV>>
00026076  00000 1                                                               <<JB.IV>>
00026080  00000 1   <<SYSTEM TABLE PTRS FOR LST ACCESS>>                        <<JB.IV>>
00026090  00000 1                                                               <<JB.IV>>
00026092  00000 1   LOGICAL CLOCKALG=DB+%355;                                   <<JB.IV>>
00026100  00000 1   INTEGER POINTER DST=DSTIX,                                  <<JB.IV>>
00026110  00000 1                  PCB=PCBIX,                                   <<JB.IV>>
00026111  00000 1                  CSTXBLK=CSTXBLKIX,                           <<JB.IV>>
00026120  00000 1                  XDST=XDSTIX;                                 <<JB.IV>>
00026130  00000 1                                                               <<JB.IV>>
00026131  00000 1   INTEGER DSTSYSBASEINX=DB+DSTIX,                            <<JB.IV>>
00026132  00000 1           SLSYSBASEINX=DB+SLIX,                               <<JB.IV>>
00026134  00000 1                  DFC=DB+DFCIX;                                <<JB.IV>>
00026140  00000 1   <<PCB INFO>>                                                <<JB.IV>>
00026150  00000 1   INTEGER ARRAY PCBSLLPTR(*)=DB+1,                            <<JB.IV>>
00026160  00000 1                PCBPBX(*)=DB+%14;                              <<JB.IV>>
00026170  00000 1                                                               <<JB.IV>>
00026180  00000 1   DEFINE SLLPTR=PCBSLLPTR(X)#,                                <<JB.IV>>
00026190  00000 1          PBX=PCBPBX(X)#;                                      <<JB.IV>>
00026200  00000 1                                                               <<JB.IV>>
00026201  00000 1   EQUATE PCBRESABORTINFOWORDNUM=0;                            <<JB.IV>>
```

JOB FILES

```
:JOB KERNELC,FIELD.SUPPORT,HP32002;RESTART;OUTCLASS=,2
:COMMENT
:COMMENT
:COMMENT                KERNELC - MODULE 92
:COMMENT
:COMMENT
:PURGE P92P002C
:PURGE U92U002C
:BUILD U92U002C;CODE=USL;DISC=256,16
:FILE NEW=TEXT,NEW;TEMP;REC=-80,16,F,ASCII;DISC=9000,16,16
:FILE INCLIO=PA0P002C.HP32002.SUPPORT
:FILE INCLPCB=PA2P002C.HP32002.SUPPORT
:FILE INCLSLL=PA3P002C.HP32002.SUPPORT
:FILE INCLST=PA4P002C.HP32002.SUPPORT
:FILE INCLMEAS=PA5P002C.HP32002.SUPPORT
:FILE INCLICS=PA7P002C.HP32002.SUPPORT
:FILE INCLMSG=PA9P002C.HP32002.SUPPORT
:SPL M92M002C.HP32002.SUPPORT,U92U002C,,S92S002C.HP32002.SUPPORT,*NEW
:RUN CROSSREF.PUB.SUPPORT
:PREP U92U002C,P92P002C;PMAP;CAP=IA,BA,PM,PH,DS
:COMMENT
:EOJ
```

GETTING PROGRAM CROSSREF LISTINGS

A. INDIVIDUAL PROGRAMS

```
:FILE LIST; DEV = LP
:FILE TEXT = NEW FILE
:RUN CROSSREF.PUB.SYS
```

B. MPE SOURCE (STREAM JOB JOOJOO2A)

```
:FILE NEW = TEXT, NEW; REC= - 80, 16, F, ASCII;
            DISC = 20000, 16, 16; TEMP
:SPL M7OMOO2A,U7OUOO2A,,S7OSOO2A,  *NEW
:RUN CROSSREF.PUB.SYS
```

NOTE: MUST HAVE CROSSREF ON THE SYSTEM TO GET SYSTEM LISTINGS OR THE JOBS WILL ABORT

PROGRAM CROSSREF USAGE

A. TO FIND INFORMATION ON IDENTIFIER.
   (IS IT A PROCEDURE, VALUE OR VARIABLE?)

B. TO LOCATE WHERE AN IDENTIFIER IS USED.

C. TO LOCATE THE STARTING POINT OF AN INTERNAL PROCEDURE OR SUBROUTINE.

D. WHEN USED WITH AN SLCREF2 LISTING, TO LOCATE ALL CALLS IN THE SYSTEM TO A PROCEDURE.

MODULES COMMON TO SERIES II/III AND SERIES 30/33

INITIAL	00	PCREATE	63	MEASSEG	96	ASOCTABL	AS
SYSDUMP	01	MORGUE	64	FILEIO	97	CATALOG	CA
SEGPROC	02	BIPC	65	DEBUGUTL	98	IOCNDPN0	IC
SEGDVR	03	IPC	66	INCLIO	A0	DISKED2	DE
LOAD	05	CHECKER	69	INCLVMC	A1	DPAN4	DN
UCOP	07	UTILITY	70	INCLPCB	A2	FREE2	FR
DEVREC	08	SEGUTIL	71	INCLSLL	A3	CICAT	HE
PROGEN	09	LOADER1	72	INCLST	A4	LISTDIR2	LD
MEMLOGP	11	RINS	73	INCLMEAS	A5	LISTEQ2	LE
LOG	12	JOBTABLE	74	INCLVMLD	A6	LISTLOG2	LL
HIOLPRT2	21	DEBUG	75	INCLICS	A7	MEMLOGAN	ML
HIOPPRT0	23	NURSERY	76	INCLHARD	A8	MEMTIMER	MT
PVPROC	31	FIRMWARE	78	INCLMSG	A9	PATCH	PA
VINIT	32	SPOOLING	79	RINSINCL	B0	RECOVER2	RC
HIOTAPE0	35	SPOOLCOM	80			SADUTIL	SA
HIOLPRT0	36	PVSYS	81			SLPATCH	SL
HIOMDSC1	37	UDC	82			SPOOK	SP
HIOLPRT1	38	USER	83				
HIOFLOP0	39	HELPUSER	84				
MAKECAT	40	OPCOMMND	85				
HIOTAPE1	41	LABSEG	86				
FILEACC	50	SDISC	87				
CI	51	LOGSEG0	90				
RESTORE	52	LOGSEG1	91				
DIRC	53	KERNELC	92				
ALLOCATE	54	KERNELD	93				
ABORTDMP	58	MISCSEGC	95				
MESSAGE	59						
PROCSEG	60						

MODULES WITH THE SAME NAME, BUT DIFERENT CODE FOR SERIES II/III AND SERIES 30/33

ININ	10	NRIO	62
PFAIL	30	MEASIO	88
HARDRES	55		

MODULES UNIQUE TO SERIES II/III

IOPTRD0	13	IOLPRT0	19
IOPTPN0	14	IOCDRD0	20
IOPLOT0	15	IOTERM0	22
IOMDISC0	16	IOPRPN0	24
IOFDISC0	17	IOREM0	25
IOTAPE0	18	IOMDISC1	27

MODULES UNIQUE TO SERIES 30/33

HIOTERM0	26	SDFGEN	34
SDFLOAD	33	[illegible]	[illegible]

- MODULES 00-49 ARE SYSTEM PROGRAMS.
- THEY ARE REPLACED WITH SYSDUMP PROGRAM CHANGES.
- SOME RUN AS SYSTEM PROCESSES.
- SOME ARE ACCESSED BY PROCEDURE CALL.
- SOME OF THOSE USED SHOW UP IN THE LOAD MAP.
- ALL SHOW UP WITH: LISTF @.PUB.SYS

- MODULES 50-99 ARE SYSTEM LIBRARY (SL) MODULES.
- THEY ARE REPLACED WITH SYSDUMP SYSTEM SL CHANGES.
- ALL SHOW UP IN THE LOAD MAP ONLY.
- SOME MODULES CAUSE MULTIPLE ENTRIES IN THE SYSTEM SYSTEM LIBRARY (SL)

```
FILESYS1      ,U50U002C.HP32002.SUPPORT,S,32
FILESYS4      ,U50U002C.HP32002.SUPPORT,S,32
FILESYS5      ,U50U002C.HP32002.SUPPORT,S,32
FILESYS6      ,U50U002C.HP32002.SUPPORT,S,32
FILESYS6A     ,U50U002C.HP32002.SUPPORT,S,32
FILESYS7      ,U50U002C.HP32002.SUPPORT,S,32
CIALTORG      ,U51U002C.HP32002.SUPPORT,S,32
CICOMSYS      ,U51U002C.HP32002.SUPPORT,S,32
CIERR         ,U51U002C.HP32002.SUPPORT,S,32
CIFILEB       ,U51U002C.HP32002.SUPPORT,S,32
CIFILEM       ,U51U002C.HP32002.SUPPORT,S,32
CIINIT        ,U51U002C.HP32002.SUPPORT,S,32
CILISTF       ,U51U002C.HP32002.SUPPORT,S,32
CIMISC        ,U51U002C.HP32002.SUPPORT,S,32
CIORGMAN      ,U51U002C.HP32002.SUPPORT,S,32
CIPREPRUN     ,U51U002C.HP32002.SUPPORT,S,32
CISUBS        ,U51U002C.HP32002.SUPPORT,S,32
CISYSMGR      ,U51U002C.HP32002.SUPPORT,S,32
CIUSERUTIL    ,U51U002C.HP32002.SUPPORT,S,32
CXSTOREST     ,U52U002C.HP32002.SUPPORT,S,32
RESTORE       ,U52U002C.HP32002.SUPPORT,S,32
STORE         ,U52U002C.HP32002.SUPPORT,S,32
DIRC          ,U53U002C.HP32002.SUPPORT,S,32
ALLOCATE      ,U54U002C.HP32002.SUPPORT,S,32
ALLOCUTIL     ,U54U002C.HP32002.SUPPORT,S,32
HARDRES       ,U55U033C.HP32033.SUPPORT,C,32
ABORTDUMP     ,U58U002C.HP32002.SUPPORT,S,32
MESSAGE       ,U59U002C.HP32002.SUPPORT,S,32
PROCSEG       ,U60U002C.HP32002.SUPPORT,S,32
NRIO          ,U62U033C.HP32033.SUPPORT,S,32
PCREATE       ,U63U002C.HP32002.SUPPORT,S,32
MORGUE        ,U64U002C.HP32002.SUPPORT,S,32
BIPC          ,U65U002C.HP32002.SUPPORT,S,32
IPC           ,U66U002C.HP32002.SUPPORT,S,32
CHECKER       ,U69U002C.HP32002.SUPPORT,S,32
UTILITY1      ,U70U002C.HP32002.SUPPORT,S,32
UTILITY2      ,U70U002C.HP32002.SUPPORT,S,32
LOADER1       ,U72U002C.HP32002.SUPPORT,S,32
RINS          ,U73U002C.HP32002.SUPPORT,S,32
JOBTABLE      ,U74U002C.HP32002.SUPPORT,S,32
DEBUG         ,U75U002C.HP32002.SUPPORT,S,32
NURSERY       ,U76U002C.HP32002.SUPPORT,S,32
FIRMWARESIM1  ,U78U002C.HP32002.SUPPORT,S,32
FIRMWARESIM2  ,U78U002C.HP32002.SUPPORT,S,32
SPOOLING      ,U79U002C.HP32002.SUPPORT,S,32
SPOOLCOMS1    ,U80U002C.HP32002.SUPPORT,S,32
SPOOLCOMS2    ,U80U002C.HP32002.SUPPORT,S,32
PVCOMSEG      ,U81U002C.HP32002.SUPPORT,S,32
PVSYSD        ,U81U002C.HP32002.SUPPORT,S,32
PVSYSM        ,U81U002C.HP32002.SUPPORT,S,32
UDC           ,U82U002C.HP32002.SUPPORT,S,32
USER          ,U83U002C.HP32002.SUPPORT,S,32
```

A00A033C.HP32033.SUPPOR

```
HELPUSER      ,U84U002C.HP32002.SUPPORT,P,32
OPLOW         ,U85U002C.HP32002.SUPPORT,S,32
OPMED         ,U85U002C.HP32002.SUPPORT,S,32
OPHI          ,U85U002C.HP32002.SUPPORT,S,32
LABSEG        ,U86U002C.HP32002.SUPPORT,S,32
SDISC         ,U87U002C.HP32002.SUPPORT,S,32
MIO'SEGMENT   ,U88U033C.HP32033.SUPPORT, ,32
LOGSEG0       ,U90U002C.HP32002.SUPPORT,S,32
LOGSEG1       ,U91U002C.HP32002.SUPPORT,S,32
KERNELC       ,U92U002C.HP32002.SUPPORT,C,32
KERNELD       ,U93U002C.HP32002.SUPPORT,S,32
MISCSEGC      ,U95U002C.HP32002.SUPPORT,C,32
MEASSEG       ,U96U002C.HP32002.SUPPORT, ,32
FILESYS1A     ,U97U002C.HP32002.SUPPORT,S,32
FILESYS2      ,U97U002C.HP32002.SUPPORT,S,32
FILESYS3      ,U97U002C.HP32002.SUPPORT,S,32
DEBUGUTL      ,U98U002C.HP32002.SUPPORT,S,32
```

A00A033C.HP32033.SUPPORT

```
INITIAL   ,P00P033C.HP32033.SUPPORT
SYSDUMP   ,P01P002C.HP32002.SUPPORT
LOAD      ,P05P002C.HP32002.SUPPORT
UCOP      ,P07P002C.HP32002.SUPPORT
DEVREC    ,P08P002C.HP32002.SUPPORT
PROGEN    ,P09P002C.HP32002.SUPPORT
ININ      ,P10P033C.HP32033.SUPPORT
MEMLOGP   ,P11P002C.HP32002.SUPPORT
LOG       ,P12P002C.HP32002.SUPPORT
HIOLPRT2  ,P21P002C.HP32002.SUPPORT
HIOPPRT0  ,P23P002C.HP32002.SUPPORT
HIOTERM0  ,P26P033C.HP32033.SUPPORT
HIOCDRD0  ,P28P033C.HP32033.SUPPORT
PFAIL     ,P30P033C.HP32033.SUPPORT
PVPROC    ,P31P002C.HP32002.SUPPORT
PVINIT    ,P32P002C.HP32002.SUPPORT
SDFLOAD   ,P33P033C.HP32033.SUPPORT
SDFGEN    ,P34P033C.HP32033.SUPPORT
HIOTAPE0  ,P35P002C.HP32002.SUPPORT
HIOLPRT0  ,P36P002C.HP32002.SUPPORT
HIOMDSC1  ,P37P002C.HP32002.SUPPORT
HIOLPRT1  ,P38P002C.HP32002.SUPPORT
HIOFLOP0  ,P39P002C.HP32002.SUPPORT
MAKECAT   ,P40P002C.HP32002.SUPPORT
HIOTAPE1  ,P41P002C.HP32002.SUPPORT
HIOCTAP0  ,P42P033C.HP32033.SUPPORT
HIOMDSC2  ,P43P002C.HP32002.SUPPORT
SDFCHECK  ,PSDFCHCK.HP32033.SUPPORT
HIOASLP0  ,HIOASLP0.HP32196.SUPPORT
HIOTERM1  ,HIOTERM1.HP32196.SUPPORT
```

A01A033C.HP32033.SUPPORT

TO LOCATE FREAD INTRINSIC CODE:

1. FIND FREAD DOWN LEFT
   COLUMN OF SLCREF.

   SEGMENT NAME IS
   FILESYS1A (WHERE
   FREAD CODE RESIDES)

   2. USING THE FILE A00A002C.HP32002.SUPPORT
      LOCATE FILESYS1A AND FIND CORRESPONDING
      USL FILE NAME.

      FILESYS1A = U50U002C

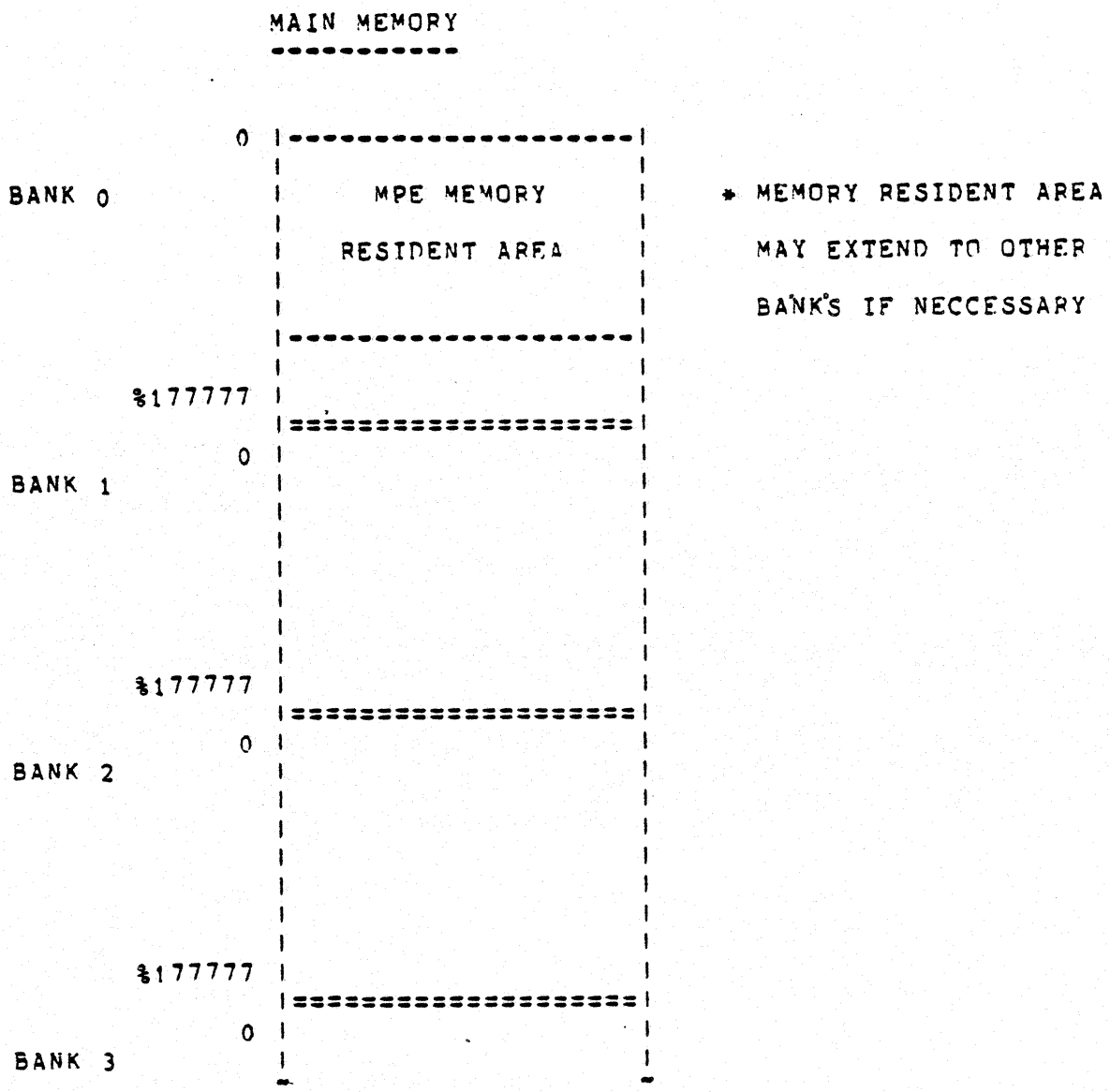
      THIS IS THE MODULE # THAT
      CONTAINS THE CODE FOR SEGMENT
      FILESYS1A. YOU NOW KNOW THAT
      YOU WILL NEED THE FOLLOWING
      FILES:

      S50S002C
      M50M002C
      J50J002C (LOOK AT JOB FILE TO SEE WHICH INCLUDE FILES (IF ANY) ARE NEEDED.)

      NOW STREAM THE JOB FILE. THE RESULT WILL BE THE SOURCE LISTING FOR
      MODULE 50, FILACC.

ONCE YOU HAVE THE LISTING FOR MODULE 50, FILEACC, USE THE CROSSREFERENCE TO LOCATE THE EDITOR LINE # WHERE FREAD BEGINS.

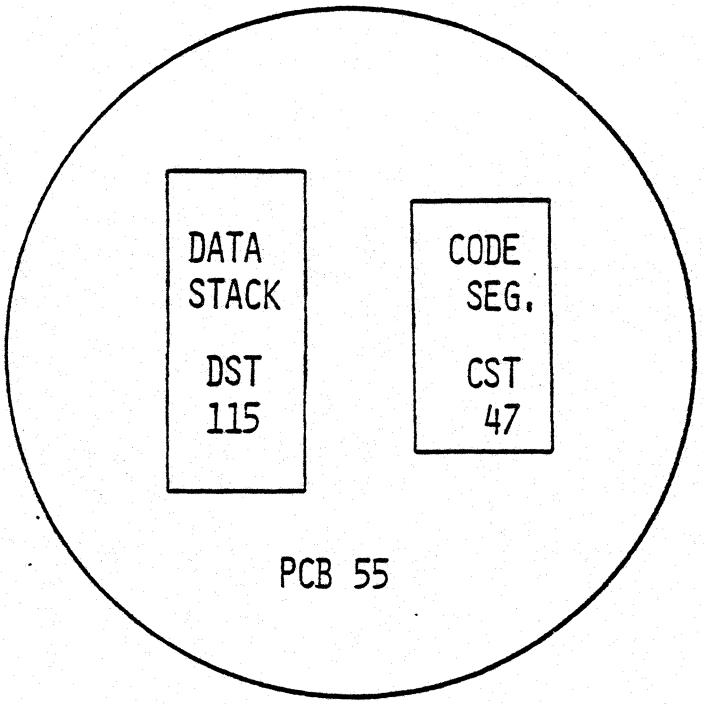
MPE MEMORY STRUCTURES


MAIN MEMORY
0
BANK 0
MPE MEMORY
RESIDENT AREA
%177777
0
BANK 1
%177777
0
BANK 2
%177777
0
BANK 3
* MEMORY RESIDENT AREA
MAY EXTEND TO OTHER
BANK'S IF NECCESSARY
ADDRESSABLE UP TO 64K BANKS

SEGMENT RESTRICTIONS

* MUST RESIDE IN ONE BANK
* OCCUPIES CONTIGUOUS MEMORY LOCATIONS

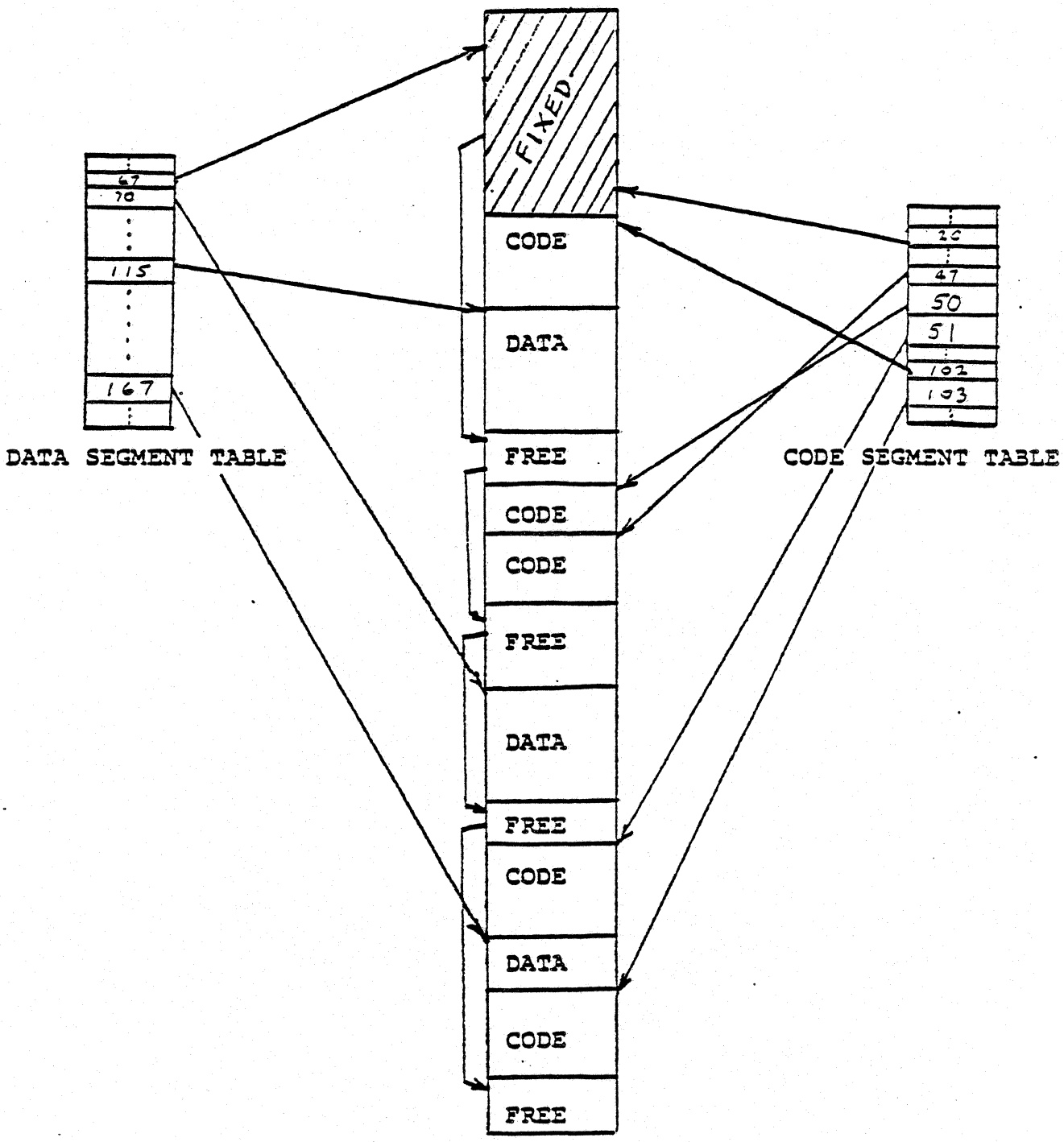
WHAT IS A PROCESS?



CODE

DATA

ENTRY IN PROCESS CONTROL BLOCK (PCB OR PIN)



MPE MEMORY LAYOUT MEMORY SEGMENTS

CODE SEGMENT TABLE
(CST)

* EACH ENTRY IS 4 WORDS IN LENGTH.

* CONTAINS INFORMATION ON
  - LOCATION (MEMORY OR DISC)
  - PRESENCE, ABSENCE OR RECOVERABLE OVERLAY (NOT CURRENTLY IMPLEMENTED)
  - TRACEFLAG (NO LONGER USED)
  - LENGTH/4

* POINTERS IN ABSOLUTE LOCATION Ø AND DST 1

* USED IF CST # IS <%300 (%277 ENTRIES MAX)

* RESERVED FOR SYSTEM AND SL (BOTH USER AND SYSTEM)

DATA SEGMENT TABLE
(DST)

* EACH ENTRY IS 4 WORDS IN LENGTH.

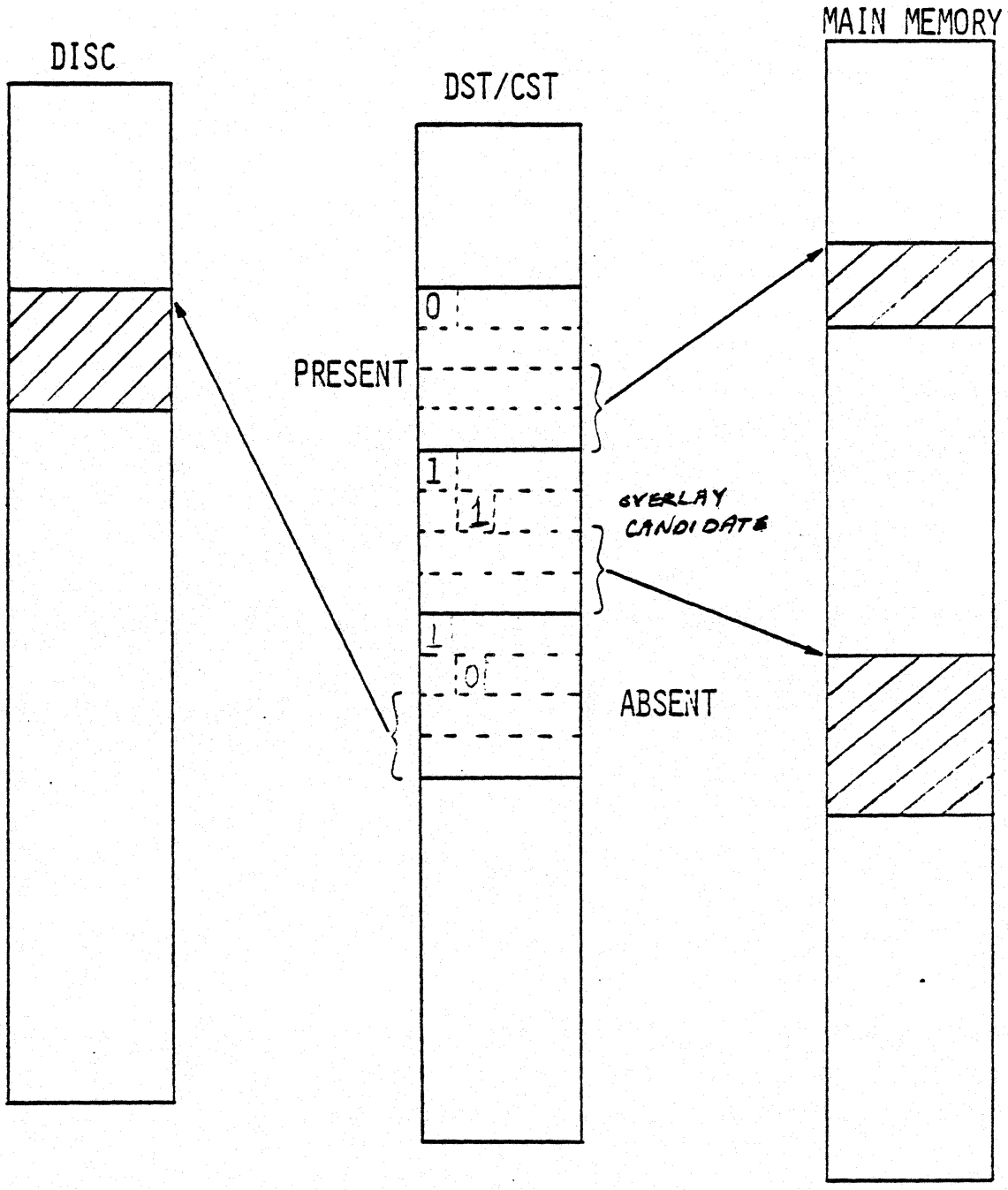
* CONTAINS INFORMATION ON
  - LOCATION (MEMORY OR DISC)
  - PRESENCE, ABSENCE OR ~~IROC~~ (IN MEMORY)
  - DISC COPY VALID (NOW REDUNDANT)
  - VIRTUAL MEMORY SIZE

* FIRST FREE DST # IS %75.

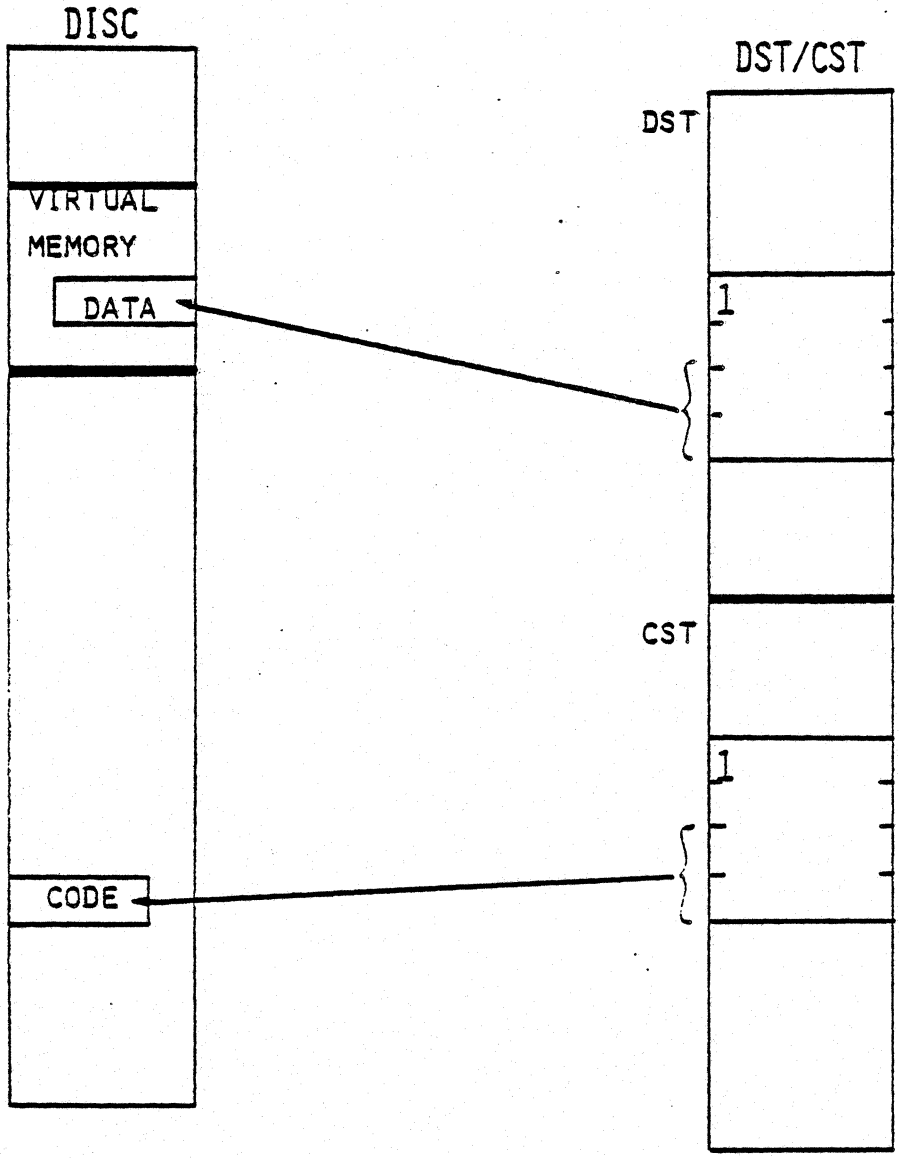
* POINTER IN ABSOLUTE LOCATION 2 AND DST #2.

* SIZE IS CONFIGURABLE.

PRESENT, ABSENT, AND OVERLAY Candidates



DATA AND CODE LOCATIONS ON DISC



DATA DESCRIPTIONS

* STACK - DATA SEGMENT WITH TOS

* XDS - DATA SEGMENT PECULIAR TO A PROCESS ie, pointed to by DB (as in split-stack mode).

* DS - DATA SEGMENT USED BY THE SYSTEM

IMPORTANT

TO REMEMBER THE DIFFERENCES

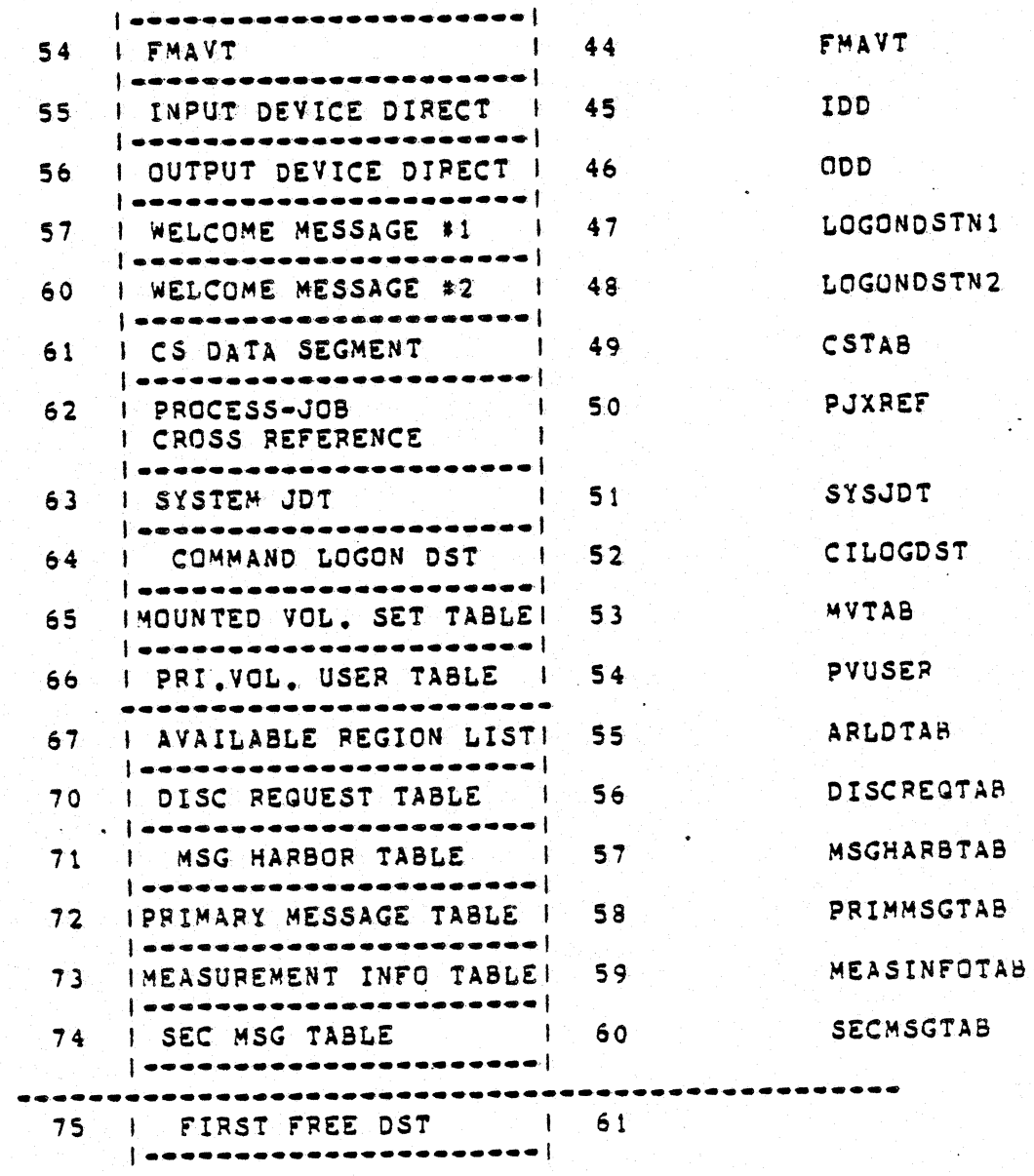
Fixed DST Entry Assignments

OCTAL		DECIMAL	TABLE NAME
0		0	
1	CST	1	CST
2	DST	2	DST
3	PCB	3	PCB
4	CSTX	4	CSTX
5	SYSTEM GLOBAL AREA	5	SYS
6	CORE	6	CORE
7	ICS	7	ICS
10	SYSTEM BUFFERS	8	SBUF
11	UCOP REQUEST QUEUE	9	UCPQ
12	PROCESS-PROCESS COMMUNICATION TABLE	10	PPCOM
13	I/O QUEUE	11	IOQ
14	TERMINAL BUFFERS	12	TBUF
15	LOGICAL-PHYSICAL DEVICE TABLE	13	LPDT
16	LOGICAL DEVICE AND CLASS TABLE	14	LDT
17	DRIVER LINKAGE TABLE	15	DLT
20	I/O RESOURCE TABLES	16	BUSY, HEAD, TAIL
21	DISC FREE SPACE	17	
22	LOADER SEGMENT TABLE	18	LST
23	TIMER REQUEST LIST	19	TPL
24	DIRECTORY	20	DDS

DST ALLOCATION (CONT.)

25	DIRECTORY SPACE	21	
26	RIN TABLE	22	RIN
27	SWAPTABLE	23	SWAPTAB
30	JOB PROCESS COUNT	24	JPCNT
31	JOB MASTER TABLE	25	JMAT
32	TAPE LABEL TABLE	26	VDD
33	LOG TABLE	27	LOGTAB
34	REPLY INFORMATION TABLE	28	RIT
35	VOLUME TABLE	29	VTAB
36	BREAKPOINT TABLE	30	STOP
37	LOG BUFFER1	31	
40	LOG BUFFER2	32	
41	LOG ID TABLE	33	LIDTAB
42	Association Table	34	
43	CST BLOCK	35	CSTBLK
44	JOB CUTOFF TABLE	36	JCUT
45	SYSTEM JIT	37	SJIT
46	SPECIAL REQ TABLE	38	SRTTAB
47	VIRTUAL DISC SPACE MANAGEMENT TABLE	39	VDSMTAB
50	//////////////////////	40	
51	ARSBM TABLE	41	ARSBMTAB
52	ILT-DIT	42	ILTDIT
53	SIR TABLE	43	SIR

DST ALLOCATION



MPE IV

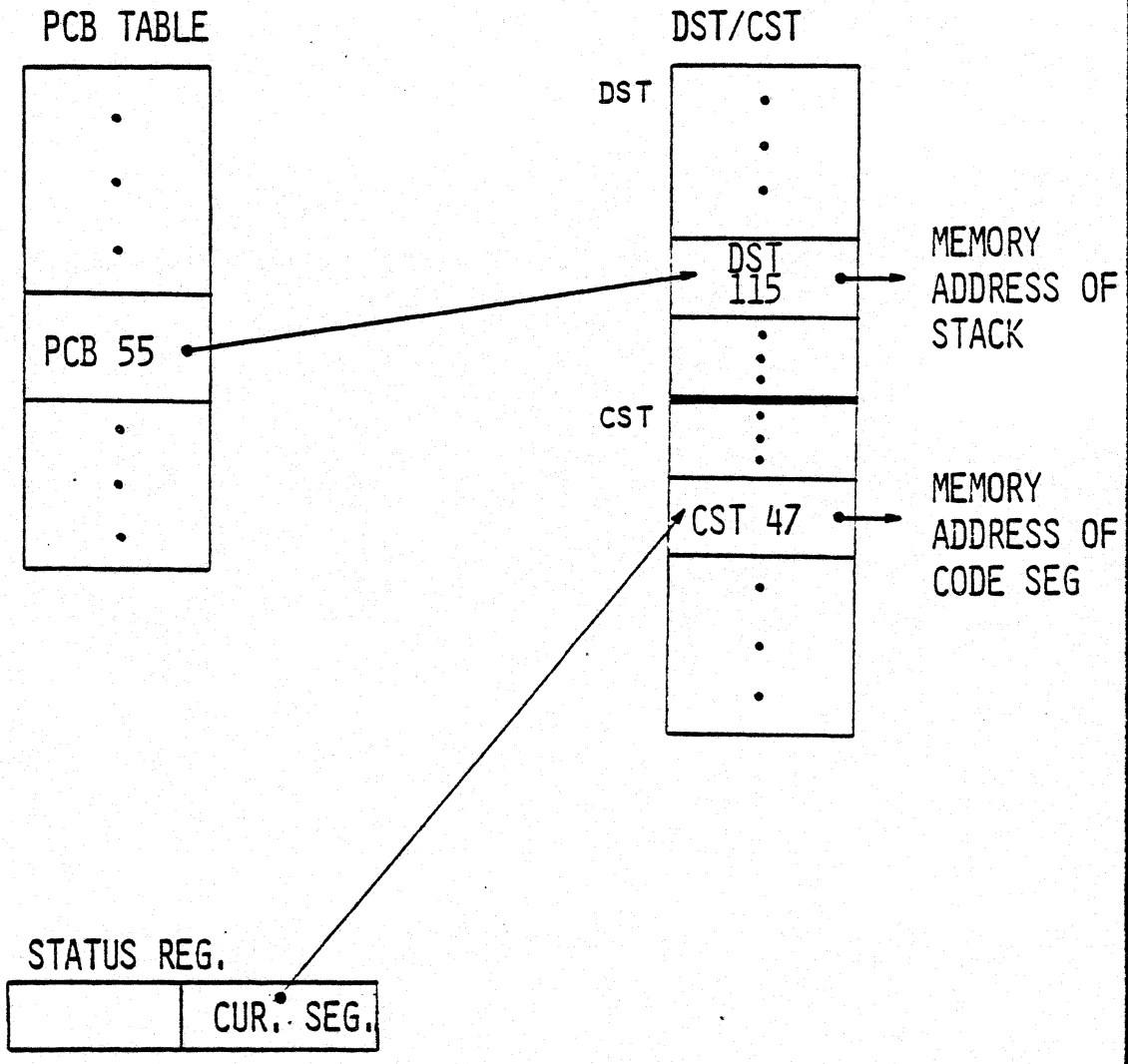
PROCESS CONTROL BLOCK TABLE

PROCESS CONTROL BLOCK TABLE

1 ENTRY FOR EACH PROCESS CURRENTLY
ON THE SYSTEM

POINTS TO DST (DATA SEGMENT TABLE) ENTRIES
FOR STACK AND EXTRA DATA SEGMENT OF PROCESS

CURRENTLY EXECUTING PROCESS



PROCESS CONTROL BLOCK
(PCB)

* USED BY DISPATCHER TO SELECT APPROPRIATE CANDIDATE FOR THE CPU.
* ENTRY IS %20 WORDS IN LENGTH.
* CONTAINS INFORMATION THE DISPATCHER MUST HAVE TO DETERMINE CANDIDATE:
  - PROCESS STATUS
  - PROCESS FAMILY (BROTHER, SON, FATHER)
  - QUEUE TYPE
  - PRIORITY
* POINTER IN ABSOLUTE LOCATION 3, DST #3

SOME IMPORTANT PCB FLAGS

SAR:	SCHEDULING ATTENTION REQUIRED
CRIT:	PROCESS SET CRITICAL & CANNOT BE ABORTED
BIO:	PROCESS BLOCKED (WAITING ON COMPLETION) FOR I/O.
MEM:	PROCESS WAITING FOR MEMORY.
EVENT FLAGS:	AN EVENT TYPE (SAME AS FLAGS IN WAKEMASK) THAT A PROCESS MAY SPECIFY TO BE AWAKENED ON.
DISPQ:	PROCESS ON DISPATCHING QUEUE
LQ:	PROCESS IN LINEAR QUEUE
C:	PROCESS IN C QUEUE
D:	PROCESS IN D QUEUE
E:	PROCESS IN E QUEUE

PSEUDO INTERRUPTS

"INTERRUPTS ARE HARDWARE/FIRMWARE EVENTS THAT ARE RECOGNIZED BY THE CPU OR ITS MICROCODE.

PSEUDOINTERRUPTS ARE SOFTWARE EVENTS THAT ARE RECOGNIZED BY THE DISPATCHER."

PSEUDO INTERRUPTS

HARD KILL:

SOMEONE CALLED QUITPROG INTRINSIC TO ABORT THE ENTIRE PROGRAM TREE.

SOFT KILL:

THIS PROCESS IS TERMINATING.

STOP:

THIS PROCESS'S FATHER IS TERMINATING (AND SO WILL WE, SHORTLY).

HYBERNATE:

THIS PROCESS TREE IS IN BREAK.

CONTROL Y:

THIS PROCESS WILL BE SERVICING A Y^c.

BREAK:

THIS PROCESS WILL BE SERVICING A BREAK.

HEWLETT PACKARD

SETCRITICAL

- TURNS ON CR BIT IN PCB.
- DISALLOWS PSEUDOINTERRUPTS
  BUT ALLOWS DISP.
- RETURNS THE OLD CRITICAL STATE.

IF A PSEUDOINTERRUPT OCCURS WHILE CRITICAL OR HOLDING A LOCKED SIR, DISPATCHER WILL SET OV BIT IN PCB INSTEAD OF INTERRUPTING.

WILL NOT PREVENT BREAK FROM AWAKING FATHER.

HEWLETT PACKARD

RESETCRITICAL

- RESETS CR BIT.
- YOU PASS IT THE OLD CRITICAL STATE.
- IF OV = 1, DO A DISP
  TO PROCESS THE PENDING
  PSEUDOINTERRUPT.

WARNING: ABORT WHILE CRITICAL.
IS A SYSTEM FAILURE 311 !

PROCESS CONTROL BLOCK

WHAT TO LOOK FOR:

PORTS "LOCKED UP":

PROCESS IS NOT IN MEMORY (AND WILL NOT FIT).

PROCESS IS WAITING FOR IO, RIN, OR IS IMPEDED.

PROCESS WAS NOT BLOCKED, WAS RUNNING (PROGRAM LOOP??)

WHAT WAS PROCESS DOING?

LOOK AT CST EXTENSION BLOCK TO SEE IF YOU RECOGNIZE THE PATTERN (YOU MAY HAVE TO ASK CUSTOMER WHAT WAS RUNNING).

LOOK AT WORKING SET POINTER TO SEE NAMES OF SEGMENTS IN THE WORKING SET (TO LOOK FOR RINS, ETC)

WHICH PROCESS WAS EXECUTING ?

LOOK FOR "*" BESIDE PROCESS ID NUMBER. IF NONE, THEN CPU WAS IN DISPATCHER, NO PROCESS WAS ACTIVE.

CONSOLE WOULD NOT FUNCTION

PROCESS 1 (PROGENITOR) IS THE PROCESS THAT HANDLES THE CONSOLE, EXAMINE THIS ENTRY.

WHO WAS ACCESSING A SPECIFIC TABLE

LOOK AT XDS COLUMN FOR PROCESS IN SPLIT STACK MODE POINTING TO THIS DST #.

HOW MANY JOBS AND SESSIONS WERE LOGGED ON?

COUNT PROCESSES WITH PTYPE = UMAIN.

HOW MANY PROGRAMS WERE EXECUTING?

COUNT PROCESSES WITH PTYPE = USONM.

LAB

IN THE CLASS DUMP, MAP OUT THE PROCESS TREE STRUCTURE FOR THE PROCESSES ON THE SYSTEM.

ALSO, LOCATE SEVERAL PRESENT SEGMENTS IN MAIN MEMORY.

CODE SEGMENT TABLE

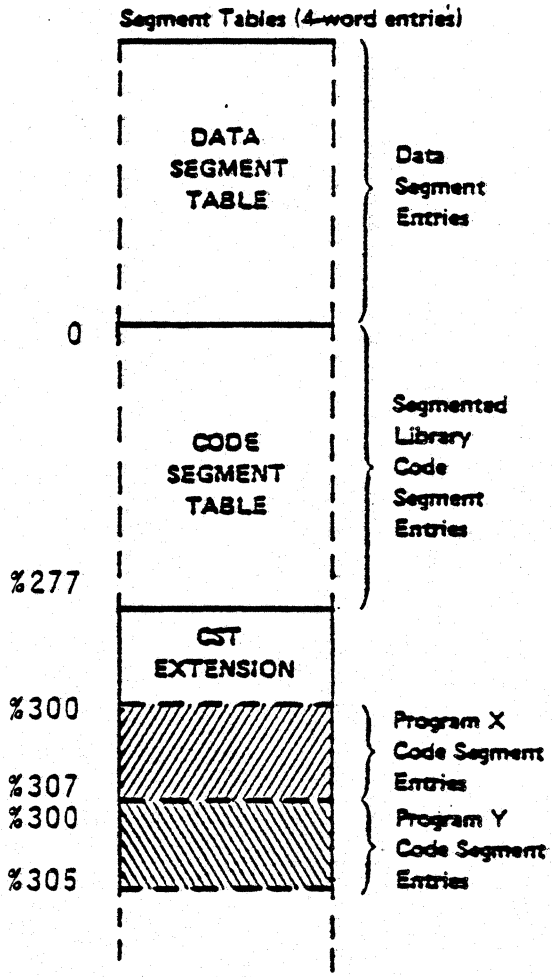
VS

EXTENDED CODE SEGMENT TABLE

CODE SEGMENTS IN SL's (MPE, GROUP SL's, ETC.) GET CST ENTRIES. (0-%277)

SEGMENTS IN USER PROGRAM FILES GET XCST ENTRIES. (%300-%377)

XCST


Segment Tables (4-word entries)
DATA SEGMENT TABLE
Data Segment Entries
0
CODE SEGMENT TABLE
Segmented Library Code Segment Entries
%277
CST EXTENSION
%300
Program X Code Segment Entries
%307
%300
Program Y Code Segment Entries
%305

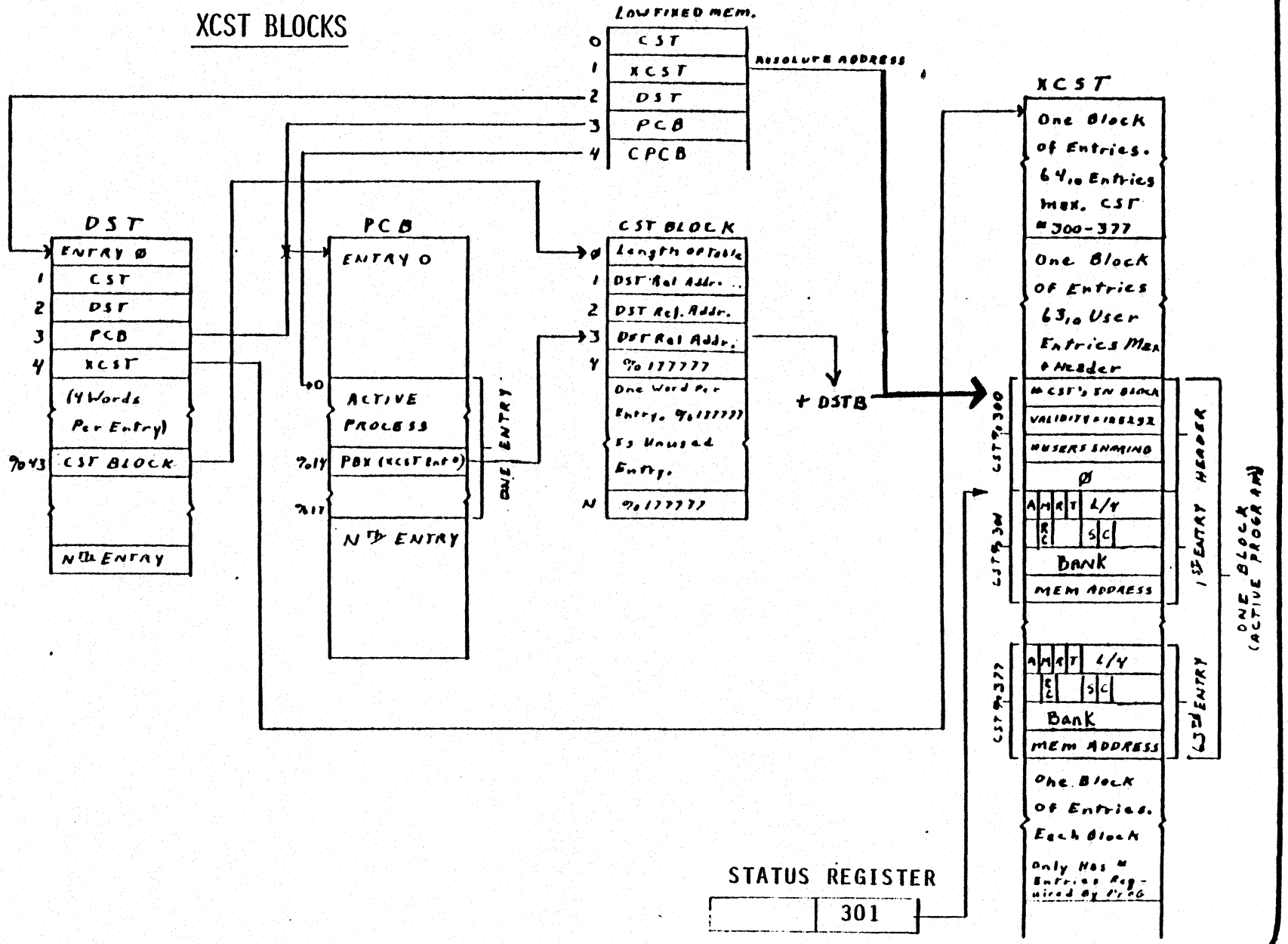
EXTENDED CODE SEGMENT TABLE
(XCST)

- SAME FORMAT AS CST ENTRIES.
- SIZE IS CONFIGURABLE.
- USED FOR USER PROGRAMS WHERE CST # >%300.
- CONTAINS 1 BLOCK OF ENTRIES PER PROGRAM.
- FIRST POSSIBLE ENTRY IN BLOCK IS %301 AND LAST IS %377 (MAX OF 63 SEGMENTS/PROGRAM).
- ENTRY %300 IN EACH BLOCK IS ZEROTH ENTRY FOR THAT BLOCK.
- POINTER IN ABSOLUTE LOCATION 1, DST #4.
- POINTER IN LOCATION 1 VALID ONLY WHEN USER HAS CPU RESOURCES.
- FOR PROGRAMS REQUIRING MORE THAN 63 SEGMENTS, THE USER MUST BREAK INTO MULTIPLE PROGRAMS (SPECIAL CAPABILITIES USUALLY REQUIRED).

CST BLOCK TABLE
(CSTBT)

* ONE WORD PER ENTRY.
* ZEROTH ENTRY IS LENGTH OF TABLE.
* EACH ENTRY IS THE SEGMENT TABLE RELATIVE ADDRESS OF CST %300 FOR THE PROGRAM IT REFLECTS.
* UNUSED ENTRIES CONTAIN A -1 (%177777)
* POINTER IN PCB (%14) FOR THE USER CODE SEGMENTS IS THROUGH THE CSTBT.
* POINTED TO BY DST %43.
* CURRENT CSTBT INDEX IN SYSDB %30

XCST BLOCKS

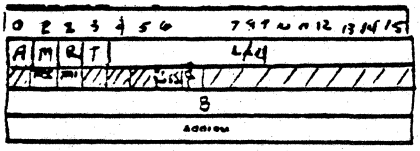

LOW FIXED MEM.
0 CST
1 XCST
2 DST
3 PCB
4 CPCB
ABSOLUTE ADDRESS
DST
ENTRY Ø
1 CST
2 DST
3 PCB
4 XCST
(4 Words Per Entry)
%043 CST BLOCK
NTH ENTRY
PCB
ENTRY 0
0
ACTIVE PROCESS
%017 PBX (XCST ENT #)
%117
N TH ENTRY
ONE ENTRY
CST BLOCK
Ø Length of Table
1 DST Rel Addr.
2 DST Rel. Addr.
3 DST Rel Addr.
4 %0 177777
One Word Per Entry. %0 177777
53 Unused Entry.
N %0 177777
+ DSTB
XCST
One Block of Entries. 64₁₀ Entries max. CST #300-377
One Block of Entries 63₁₀ User Entries Max + Header
CST %300
# CST's IN BLOCK
VALIDITY = 188292
USERS SHARING
Ø
A M R T L/4
R C S C
BANK
MEM ADDRESS
CST %301
CST %377
1ST ENTRY HEADER
63RD ENTRY
ONE BLOCK (ACTIVE PROGRAM)
One Block of Entries. Each Block Only Has # Entries As Required By Prog
STATUS REGISTER
301

LAB

USING THE UNFORMATTED TABLES (FORMATTED PCB O.K.),
LOCATE IN THE CLASS DUMP THE CODE SEGMENT OF SOME
PROCESS

3 formats to Remember ! Quick reference

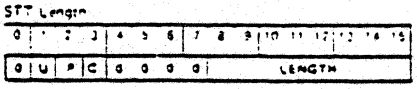
CODE SEGMENT TABLE



A Absence Bit - 1 = segment is absent from main memory
M Mode Bit - 1 = segment executes in Privileged Mode (Code only)
R Reference Bit - 1 = segment has been referenced (set by microcode)
T Trace Bit - 1 = trace feature is used. Checked by PCAL instruction
L Length Field - segment length divided by 4
B Base Address - Points to memory page in resident or near memory in which segment resides
ADDRESS - Absolute address of PB within B if the segment is present, otherwise the 3rd and 4th words contain the absolute disc address

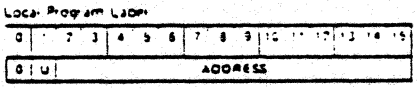
SEGMENT TRANSFER TABLE Words

STT Length



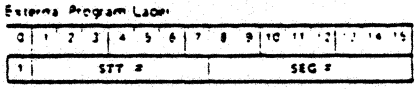
U Uncallable bit
P Patch Area Bit = 1 if patch area exists
C Code Segment Bit (0 = SL code segment, 1 = program code segments)
LENGTH Maximum = 225 (Calls from external segments may reference only the first 127 entries PL 1 thru PL 127 (IPL 0 STTL)

Local Program Label



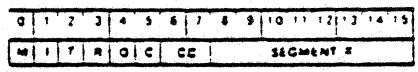
U Uncallable bit
ADDRESS PB relative + only

External Program Label



STT # STT entry number in target segment maximum = 127
SEG # Target segment

STATUS Word



M Mode bit (1 = for privileged mode)
I Interrupt enable (1)/disable(0) external
T Trap enable(1)/disable(0) user
R Right Stack Opcode bit (pending = 1)
O Overflow bit
C Carry bit
CC Condition Code
SEGMENT # currently executing

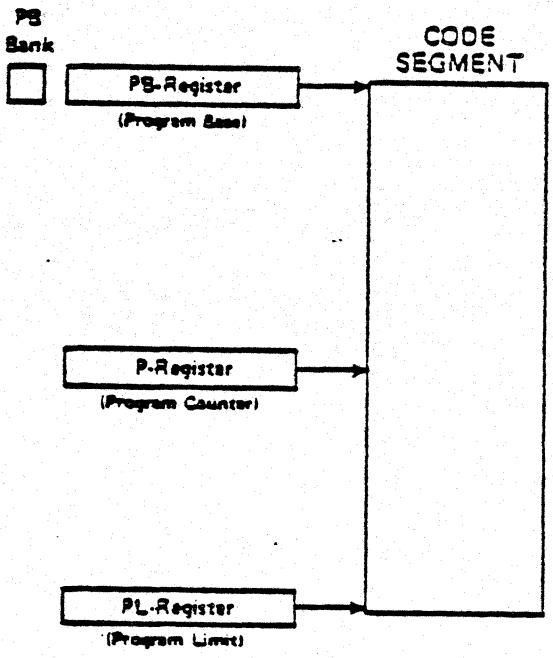
ROC - RECOVERABLE OVERLAY CANDIDATE
IMI - IN MOTION IN
SYS - SYSTEM SEGMENT
CORE - CORE RESIDENT

CODE SEGMENTS

DATA SEGMENTS

CODE SEGMENTS

CODE SEGMENT POINTING REGISTERS



IF A PROCESS IS EXECUTING CODE IN A CERTAIN CODE SEGMENT, HOW DOES IT CALL A PROCEDURE IN THE SAME SEGMENT OR IN ANOTHER SEGMENT?

BY UTILIZING THE SEGMENT TRANSFER TABLE (STT) ATTACHED TO EACH CODE SEGMENT.

SEGMENT TRANSFER TABLE (STT)

- PHYSICALLY RESIDES IN EACH CODE SEGMENT.
- ENTRY SIZE IS 1 WORD.
- ZEROTH ENTRY AT PL.
  - CONTAINS NUMBER OF ENTRIES - ZEROTH ENTRY
- ONE ENTRY FOR EACH PCAL INSTRUCTION IN SEGMENT
- N-TH ENTRY AT PL-N.
- TWO FORMATS
  - INTERNAL
  - EXTERNAL
- EXTERNALS ARE SATISFIED AND FILLED IN AT :PREP AND :RUN TIME

SEGMENT TRANSFER TABLE Words

STT Length

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	0	0	0	0	0	0	LENGTH							

U Uncallable bit
LENGTH Maximum = 255 (Calls from external segments may reference only the first 128 entries, PL thru PL-127.)

Local Program Label

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	U	ADDRESS													

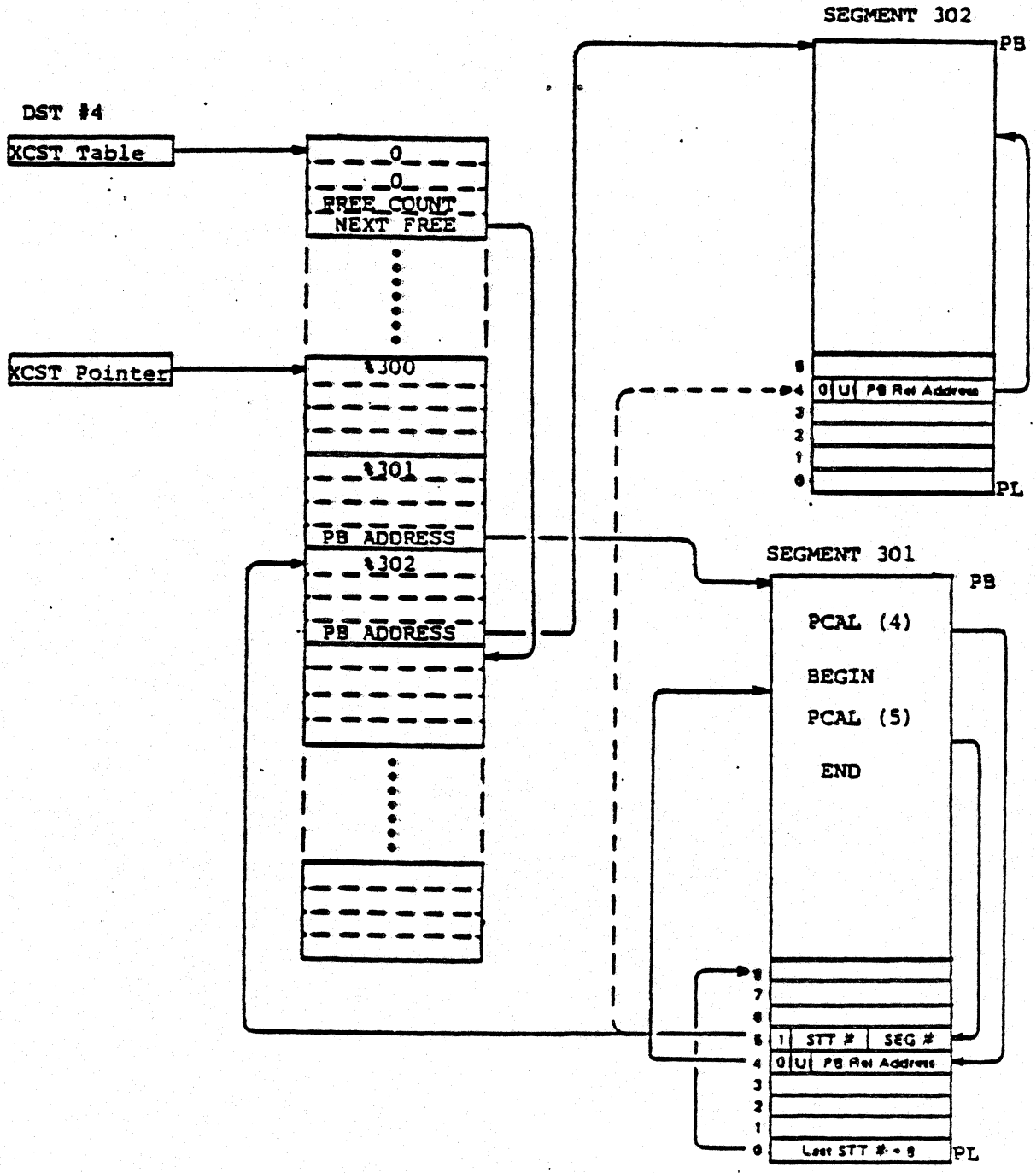
U Uncallable bit
ADDRESS PB relative, + only

External Program Label

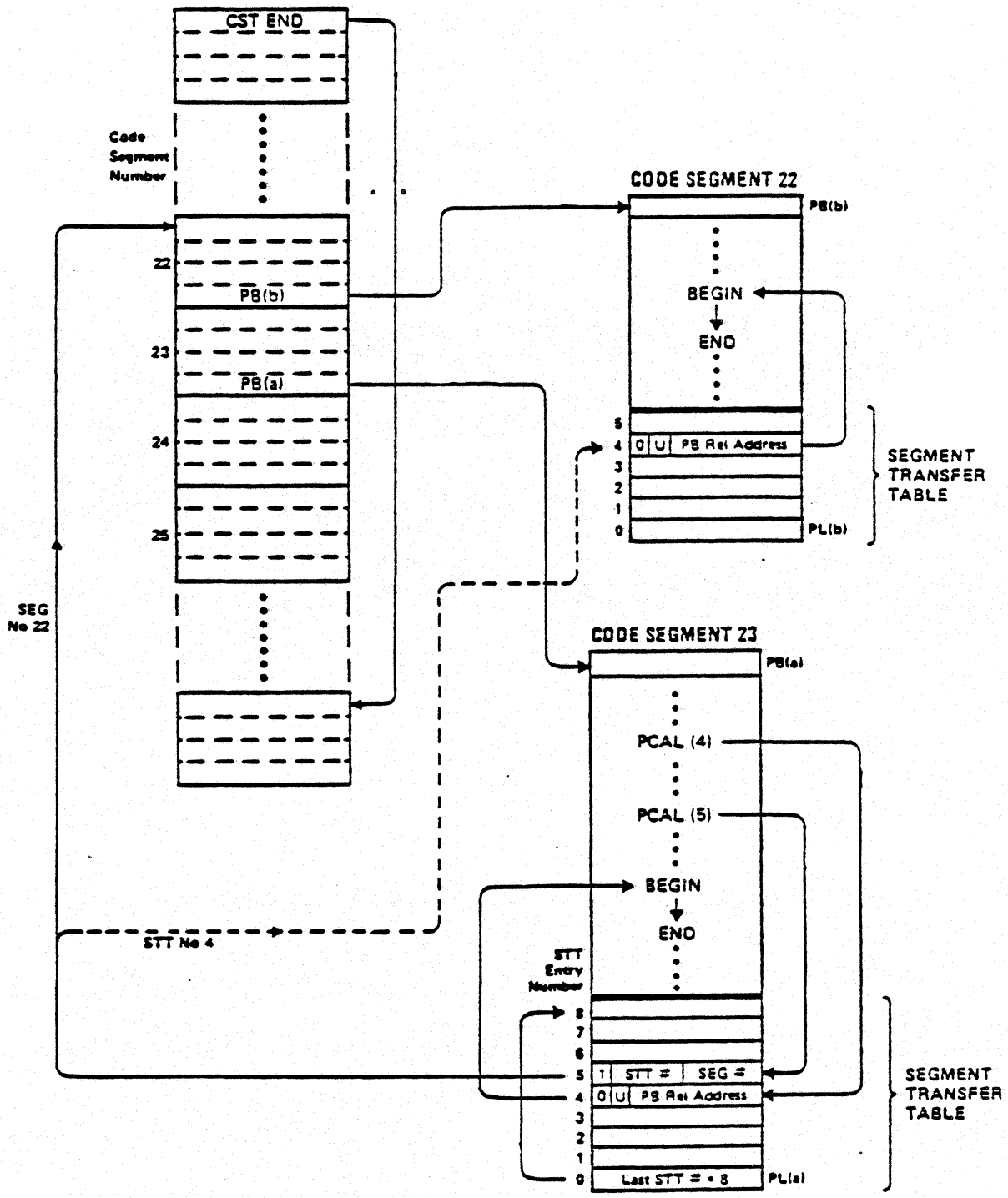
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	STT #							SEG #							

STT # STT entry number in target segment, maximum = 127
SEG # Target segment

CODE SEGMENT TRANSFERS



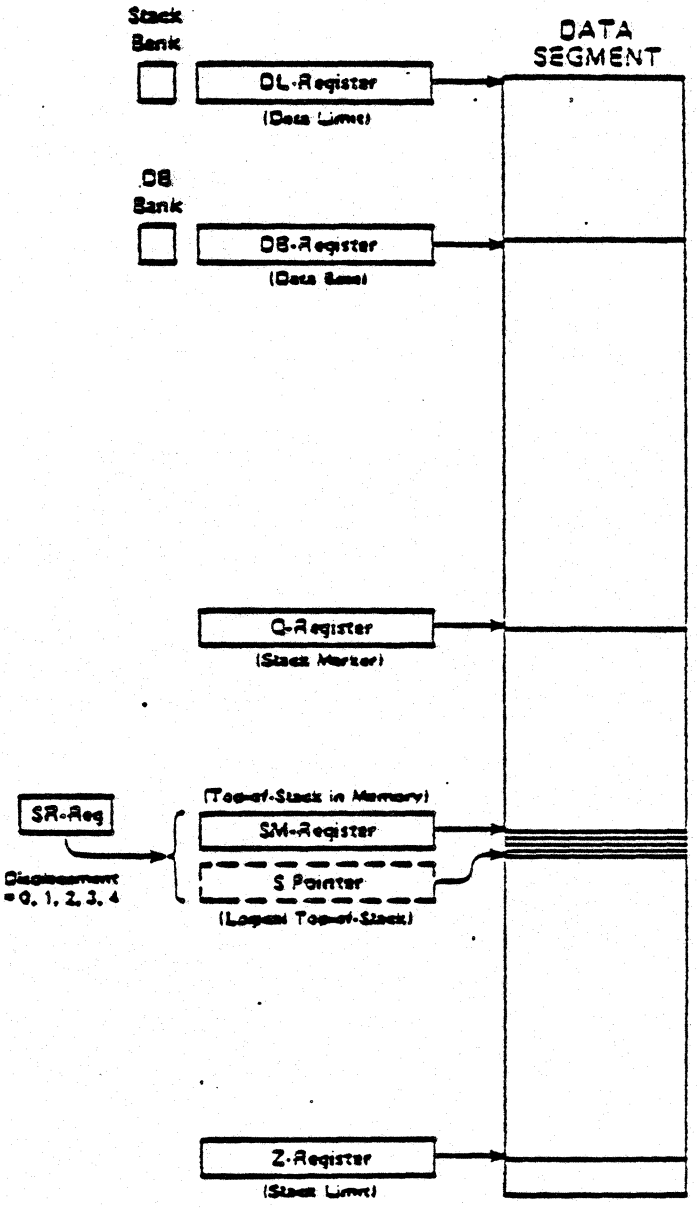
CODE SEGMENT TRANSFERS



DATA SEGMENTS

STACK

DATA SEGMENT POINTING REGISTERS



THE STACK

THE STACK FOR A PROCESS CONTAINS THE BULK OF PROCESS INFORMATION IN AN AREA OF THE STACK REFERRED TO AS THE PROCESS CONTROL BLOCK EXTENSION (PCBX). THE PCBX IS LOCATED IN THE STACK SEGMENT STARTING AT WORD 0 AND EXTENDING TO BUT NOT INCLUDING DL.

THE GENERAL PHILOSOPHY OF MPE WAS TO PLACE THOSE DATA ITEMS IN THE PCB THAT MUST BE PRESENT TO MAKE SCHEDULING DECISIONS, AND TO PLACE PROCESS SPECIFIC INFORMATION IN THE PCBX (SINCE THE STACK WOULD BE IN MEMORY WHEN THE PROCESS WAS EXECUTING).

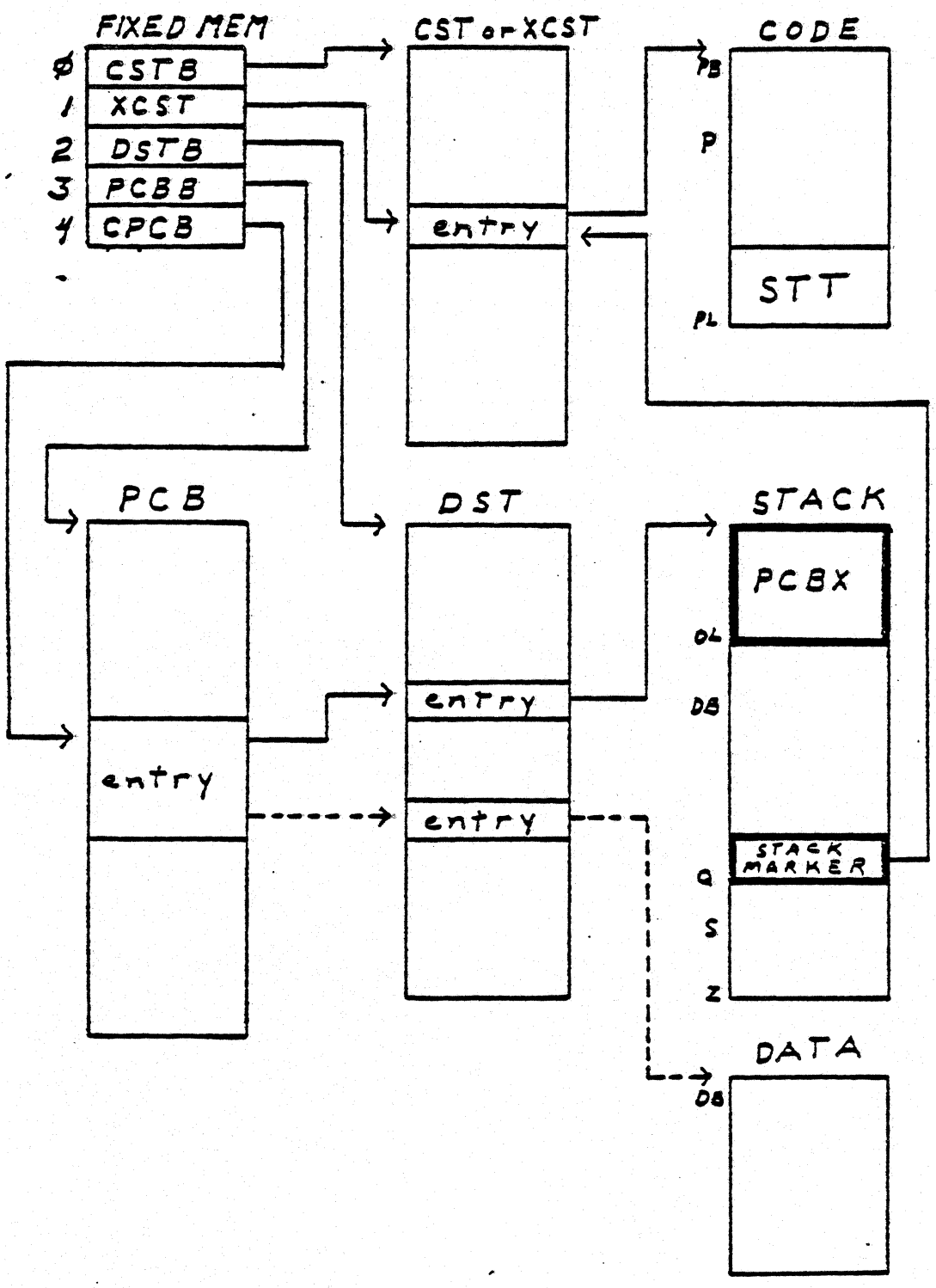
THE HEADER INFORMATION FOR THE STACK, ALONG WITH THE STACK MARKERS, ARE PRINTED BEHIND THE PCB IN THE DUMP FOR ALL THE STACKS THAT ARE IN MEMORY. THE CURRENT PROCESS STACK IS PRINTED FIRST, FOLLOWED BY ALL THE STACKS IN MEMORY, SORTED IN PCB ENTRY ORDER.

THERE ARE TWO SECTIONS PRINTED FOR EACH STACK, NAMELY:

HEADER AND PCBX INFORMATION.

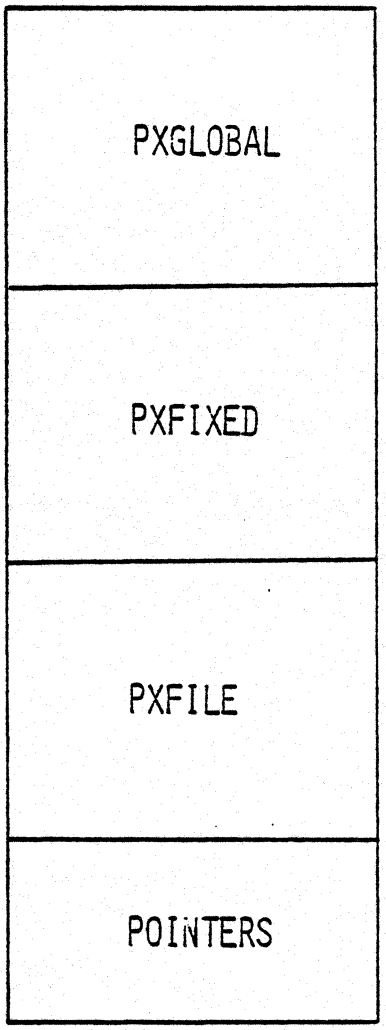
STACK MARKERS.

EXECUTING PROCESS

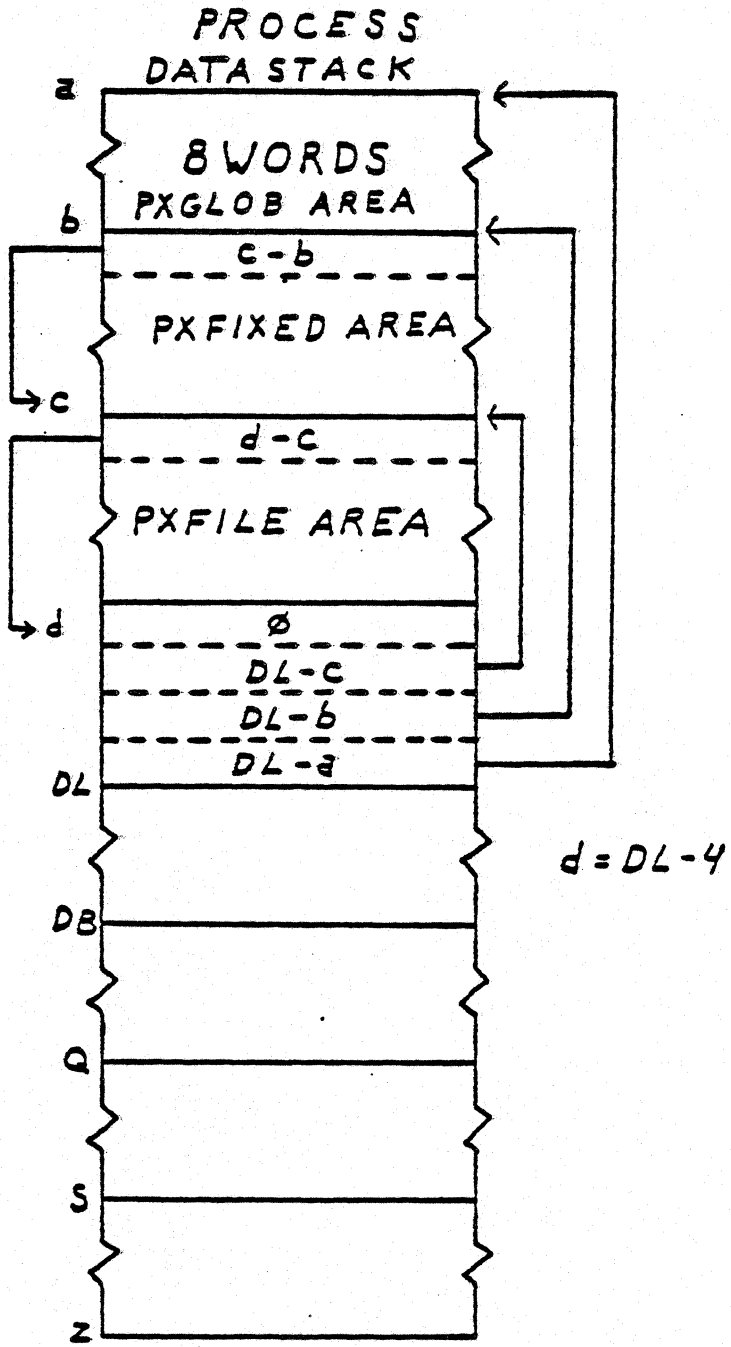


PCBX

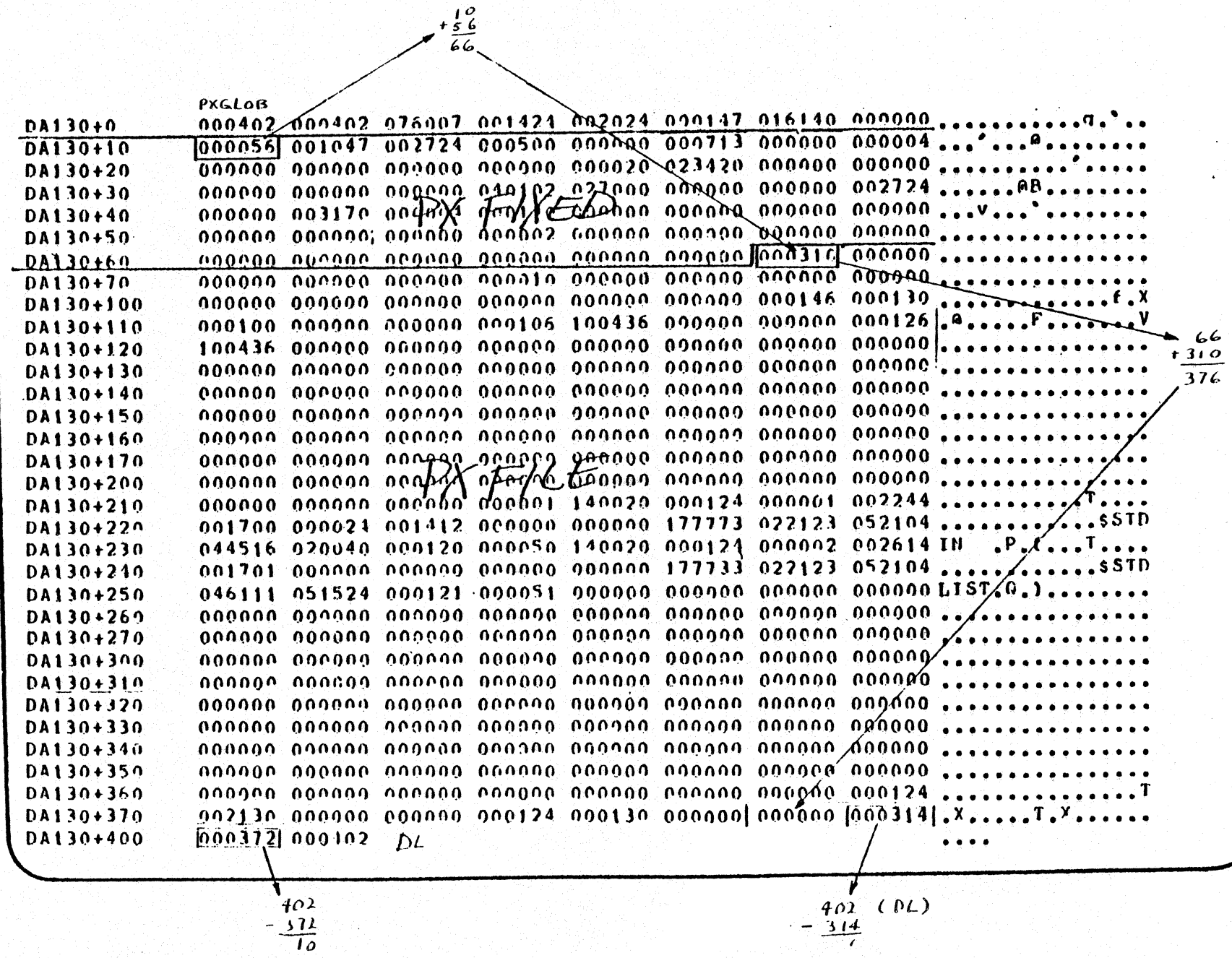
1/STACK



LOCATING PCBX AREAS



PROCESS CONTROL BLOCK EXTENTION



	PXGLOB								
DA130+0	000402	000402	076007	001421	002024	000147	016140	000000	..... q . ..
DA130+10	000056	001047	002724	000500	000000	000713	000000	000004	... ... @ .....
DA130+20	000000	000000	000000	000000	000020	023420	000000	000000	..... ' .....
DA130+30	000000	000000	000000	040102	027000	000000	000000	002724	..... @B .....
DA130+40	000000	003170	004004	000000	000000	000000	000000	000000	... v ... ' .....
DA130+50	000000	000000	000000	000002	000000	000000	000000	000000	.....
DA130+60	000000	000000	000000	000000	000000	000000	000310	000000	.....
DA130+70	000000	000000	000000	000010	000000	000000	000000	000000	.....
DA130+100	000000	000000	000000	000000	000000	000000	000146	000130	..... f X
DA130+110	000100	000000	000000	000106	100436	000000	000000	000126	@ ..... F V
DA130+120	100436	000000	000000	000000	000000	000000	000000	000000	.....
DA130+130	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+140	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+150	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+160	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+170	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+200	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+210	000000	000000	000000	000001	140020	000124	000001	002244	..... T ....
DA130+220	001700	000021	001412	000000	000000	177773	022123	052104	..... $STD
DA130+230	044516	020040	000120	000050	140020	000124	000002	002614	IN .P (... T ....
DA130+240	001701	000000	000000	000000	000000	177733	022123	052104	..... $STD
DA130+250	046111	051524	000121	000051	000000	000000	000000	000000	LIST Q) .....
DA130+260	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+270	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+300	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+310	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+320	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+330	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+340	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+350	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+360	000000	000000	000000	000000	000000	000000	000000	000124	..... T
DA130+370	002130	000000	000000	000124	000130	000000	000000	000314	. X ..... T . X .....
DA130+400	000372	000102	DL						....

PROCESS CONTROL BLOCK EXTENSION -- STACK HEADER DATA

THE HEADER LINE CONTAINS THE FOLLOWING INFORMATION:

DL AND DB REGISTER VALUES (SEGMENT RELATIVE).

JMAT INDEX JOB MASTER TABLE INDEX (KEEPS TRACK OF JOBS AND SESSIONS. IS FORMATTED BY SHOWJOB)

JPCNT INDEX JOB PROCESS COUNT INDEX (JPCNT KEEPS TRACK OF THE NUMBER OF PROCESSES USED BY A JOB/SESSION).

JOB INPUT AND OUTPUT LOGICAL DEVICES. IN DECIMAL, THE LDEV OF THE DEVICE FOR $STDIN AND $STDLIST. MAY BE PSEUDO LDEVS IF SPOOLED (SEE LOGICAL TO PHYSICAL DEVICE TABLE). REAL LDEVS CAN BE LOOKED UP IN SYSDUMP IO CONFIGURATION LISTING (FROM DUMPJOB)..

JDT AND JIT DST INDEXES. THE DST NUMBER OF THE JOB INFORMATION TABLE AND THE JOB DIRECTORY TABLE.

JOB TYPE IF A SESSION OR JOB, THE JOB/SESSION NUMBER IS PRINTED (IN THE SAME FORM AS A SHOWJOB).

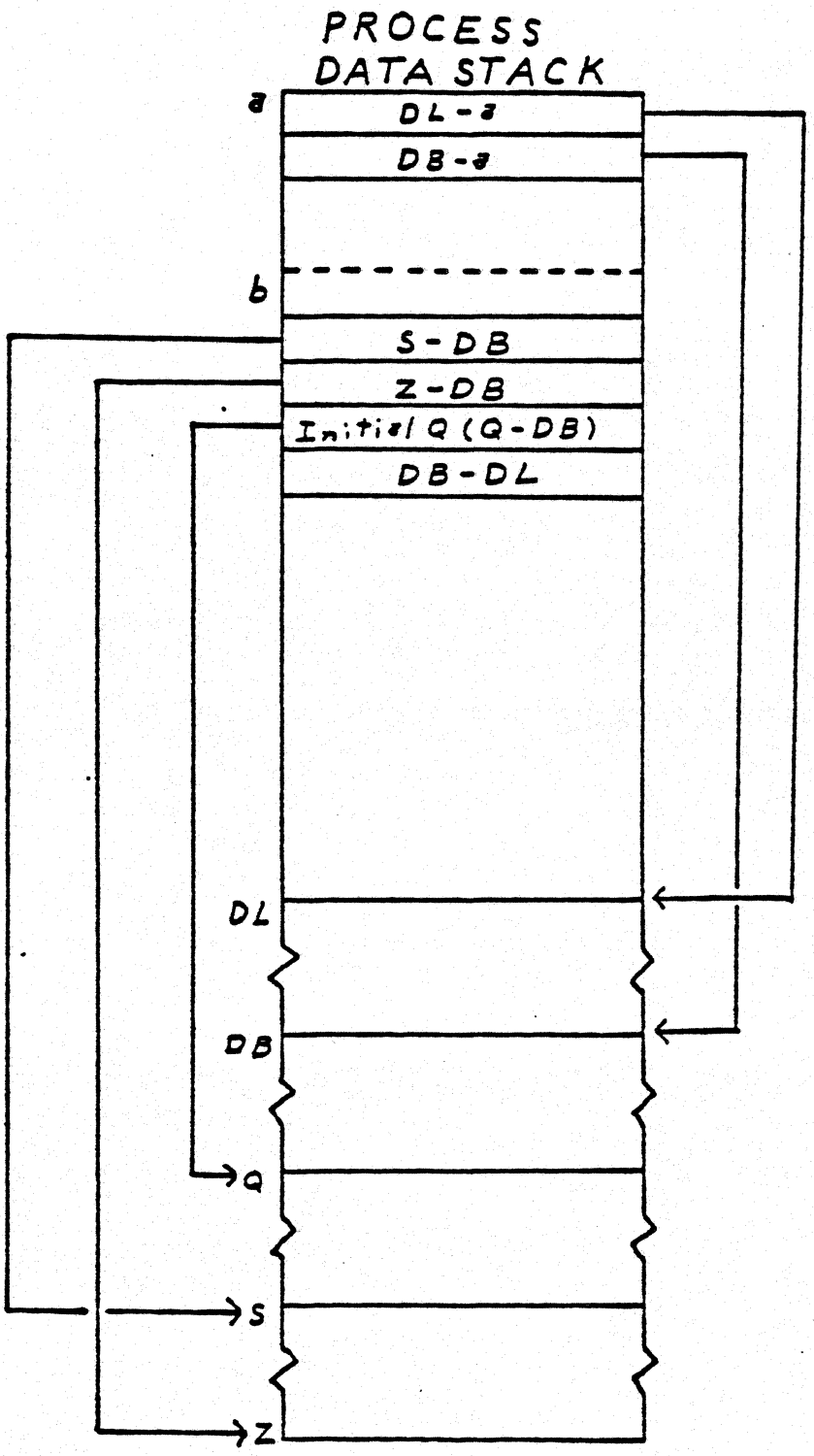
DUPLICAT YES IF $STDIN AND $STDLIST ARE A DUPLICATIVE PAIR.

INTERACT YES IF AN INTERACTIVE SESSION.

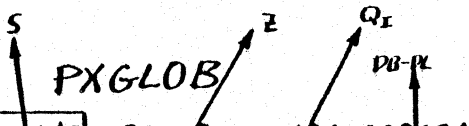
INIT Q LOCATION (RELATIVE TO DB REGISTER) OF THE FIRST (OLDEST) STACK MARKER. THIS MARKER SHOULD ALWAYS BE A MARKER FOR THE PROCEDURE TERMINATE, PLACED ON THE STACK TO KEEP THE USER FROM DOING TOO MANY EXITS (ANY PROGRAM THAT QUITS WITHOUT CALLING TERMINATE WILL EXECUTE THIS MARKER AND TERMINATE NORMALLY).

JCUT INDEX THE JOB CUTOFF INDEX. IF ZERO, THEN NO TIMELIMIT IS ASSIGNED TO THIS JOB/SESSION.

LOCATING STACK LIMITS



PROCESS CONTROL BLOCK EXTENTION



DA130+0	000402	000412	076007	001121	002024	000147	016140	000000	..... q. ..
DA130+10	000056	001047	002724	000500	000000	000713	000000	000004	... ... @ .....
DA130+20	000000	000000	000000	000000	000020	023420	000000	000000	..... ' .....
DA130+30	000000	000000	000000	010102	027000	000000	000000	002724	..... AB .....
DA130+40	000000	003170	004000	000000	000000	000000	000000	000000	... v ... ' .....
DA130+50	000000	000000	000000	000002	000000	000300	000000	000000	.....
DA130+60	000000	000000	000000	000000	000000	000000	000316	000000	.....
DA130+70	000000	000000	000000	000010	000000	000000	000000	000000	.....
DA130+100	000000	000000	000000	000000	000000	000000	000146	000130	..... f. X
DA130+110	000100	000000	000000	000106	100436	000000	000000	000126	. q ..... F V
DA130+120	100436	000000	000000	000000	000000	000000	000000	000000	.....
DA130+130	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+140	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+150	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+160	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+170	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+200	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+210	000000	000000	000000	000001	140020	000124	000001	002244	..... T ....
DA130+220	001700	000021	001412	000000	000000	177773	022123	052104	..... $STD
DA130+230	044516	020040	000120	000050	140020	000121	000002	002614	IN . P . (... T ....
DA130+240	001701	000000	000000	000000	000000	177733	022123	052104	..... $STD
DA130+250	046111	051524	000121	000051	000000	000000	000000	000000	LIST . Q .) .....
DA130+260	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+270	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+300	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+310	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+320	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+330	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+340	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+350	000000	000000	000000	000000	000000	000000	000000	000000	.....
DA130+360	000000	000000	000000	000000	000000	000000	000000	000124	..... T
DA130+370	002130	000000	000000	000124	000130	000000	000000	000314	. X ..... T . X .....
DA130+400	000372	000102	[PB, PL]						....

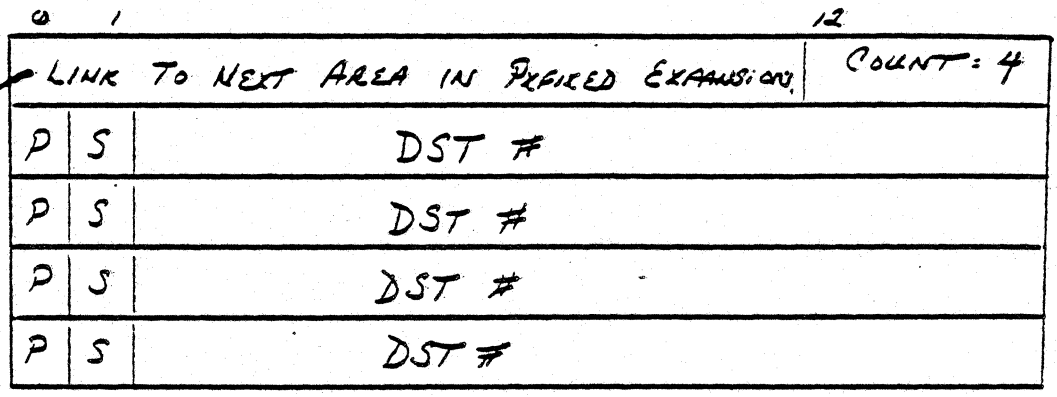
M 45

HANDLING EXTRA DATA SEGMENTS

```
GETDSEG

    . . . .
    . . . .

    PXDSEG (get new entry)
    . . . .
    . . . .


PXDSEG

    Set pointers to first XDS block in PXPINED  <<setup>>
 L1:Scan through block for unassigned entry
    IF no more entries THEN
        Bump to next block
        IF bump successful THEN GO TO L1
        GETPXSEG:  <<have to create a new block>>
        Initialize new block
        GO TO L1:
        <<found entry>>
        RETURN ENTRY #  <<index into block>>


GETPXSEG

        Scan bit map for free block
        IF found THEN return block address
           ELSE
              Initialize bit map in anticipation of stack
              expansion
              Expand stack   <<up to 3 times of 128w each>>
              RETURN NEWLY CREATED BLOCK ADDRESS
```

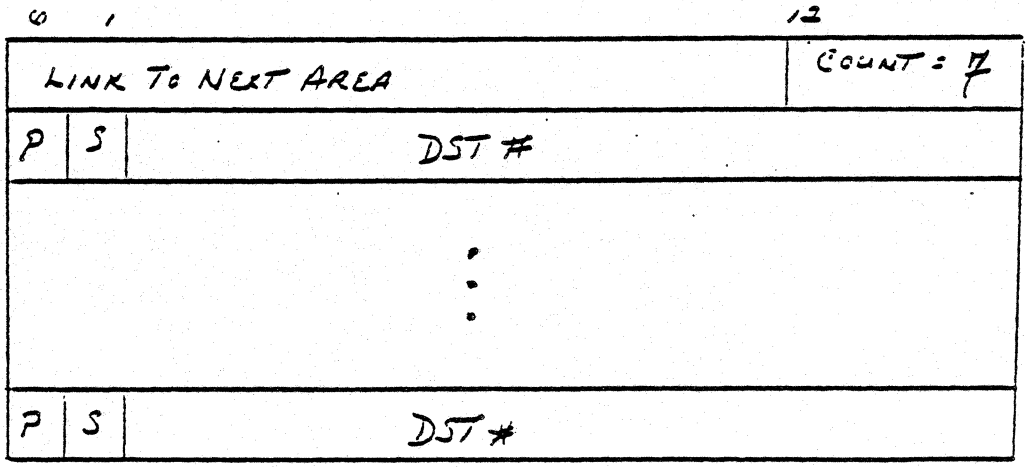
Extra Data Segment Entries in PREFIXED Area

In PREFIXED Area



P : off if segment opened by privileged !
S : on if segment is shared

In PREFIXED Expansion Area



LINK is PREFIXED area relative pointer to next set of entries.

Extra Data Segment Entries in PXFIXED Area

Notes:

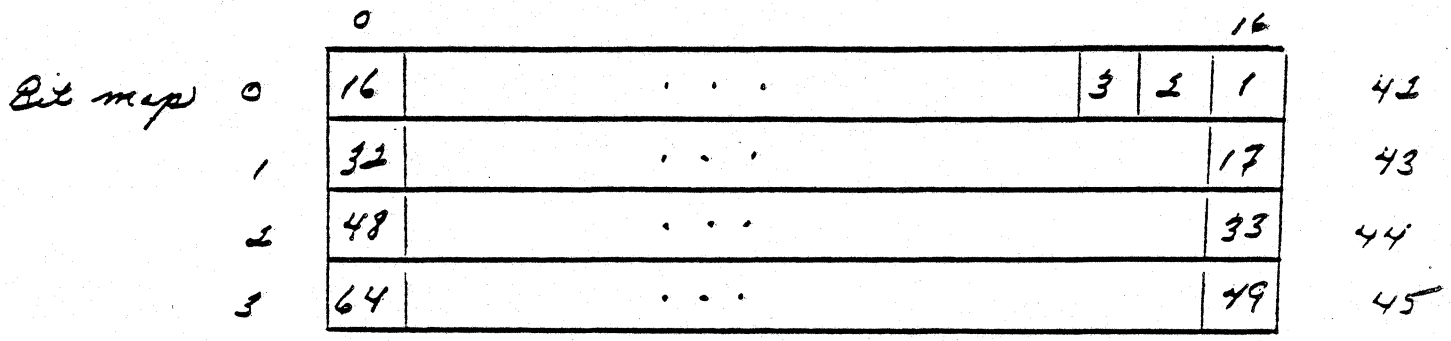
1. There is room for 4 extra data segment entries in the PXFIXED area of the PCBx.

2. If more entries are needed, they are placed in the PXFIXED expansion area, and are linked together by the first 12 bits of the first word of each group.

3. The 12 bit links are relative to the start of the PXFIXED area.

4. In the PXFIXED expansion area, extra entries are allocated 8 (words) at a time.

In PXFIXED area



Notes:

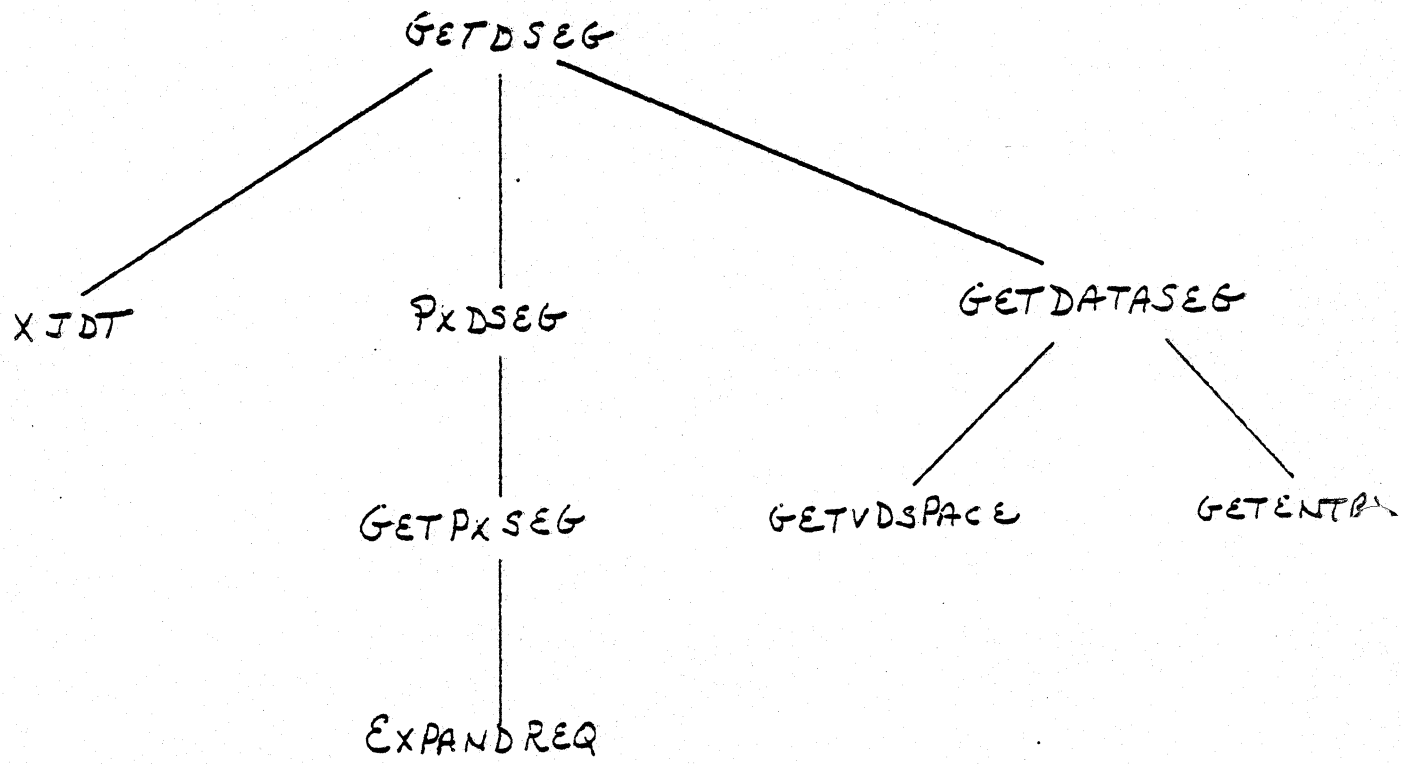
1. Bit map is used to keep track of allocated and unallocated areas (of 8 words) in PXFIXED expansion area.

2. Bit i on indicates i^{th} group of 8 words is free for use. Bit i off indicates i^{th} group of 8 words is in use.

3. Bit map is used only in procedure GETPXSEG to allocate a new area of 8 words.

4. Bit map could handle maximum of 64 8 word areas (or 512 words). However

PxFIXED expansion area is only allowed to expand up to 3 extra sectors (384 words).


GETDSEG
XJDT
PXDSEG
GETDATASEG
GETPXSEG
GETVDSPACE
EXPANDREQ

```
procedure GETDSEG (INDEX, LENGTH, ID );
  logical INDEX, ID;
  integer LENGTH;

  begin
    if ID <> 0 then {shared data seg}
      begin
        search JDT for data seg entry
          having ID (with XJDT(0, ··· ));
        if found then
          begin
            search process' PREFIXED area for
              data seg entry having ID (with
              PXDSEG (6, ·· ));
            if found then set CCG and go to fin
              else decrement segment sequence
              count (with XJDT (2, ·· ));
          end;
      end;

  if maximum number of data segs is
    exceeded then set error and go to error;
```

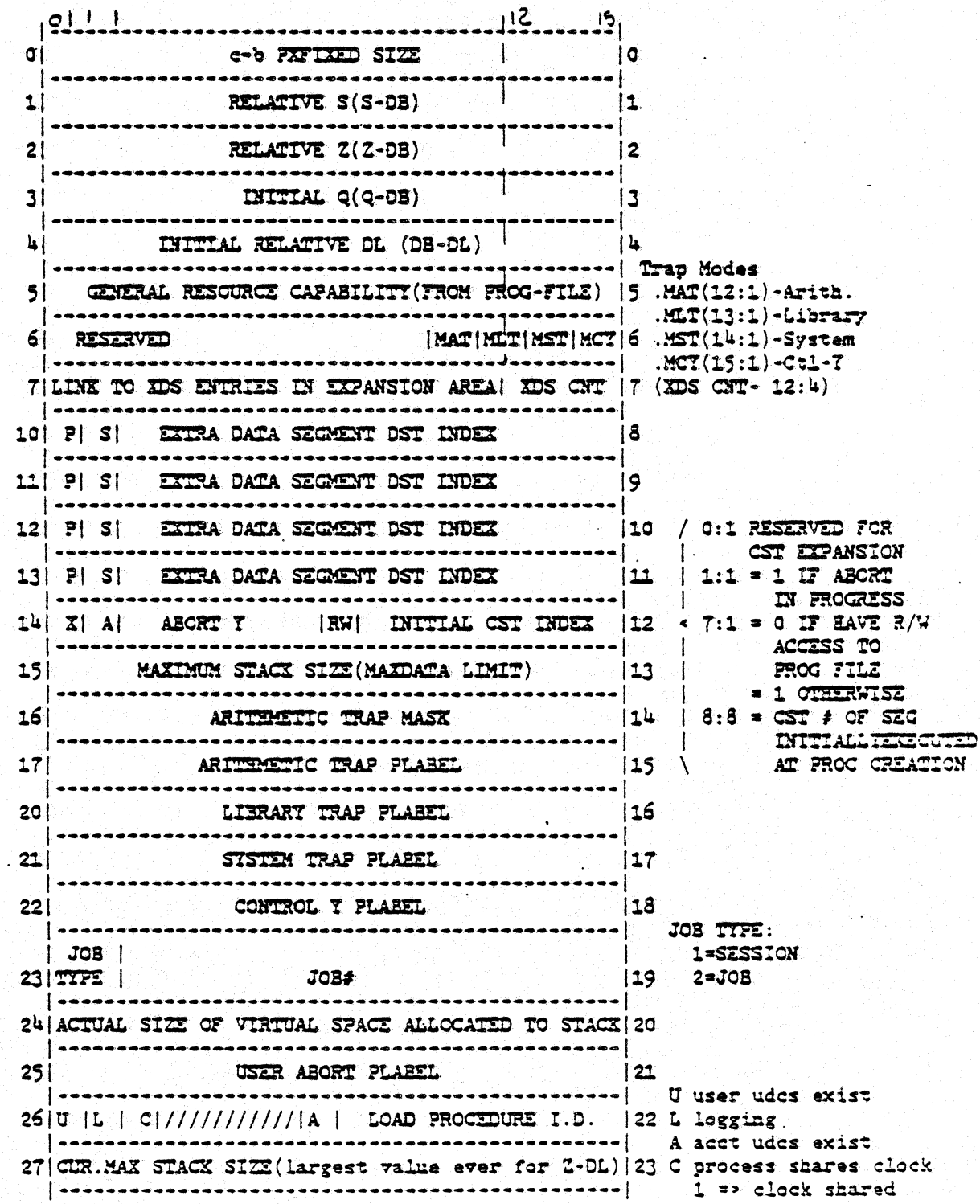
M45.7

```
if ID <> 0 then { shared data seg }
   begin
      set shared bit for PXFIXED entry;
      if entry exists in JDT for data seg
         ( determined with XJDT(2, ... ))
         then go to fin;
   end;

if privileged and LENGTH < 0
   then adjust LENGTH and go to contx;

if bad LENGTH then set error code
   and go to error;

contx : adjust LENGTH;

   try to allocate new DST entry (with GETDATASEG)
   if not successful then set error code
      and go to error;

   if ID = 0 then go to contx;

   try to put entry in JDT (with XJDT(1, ...));
```

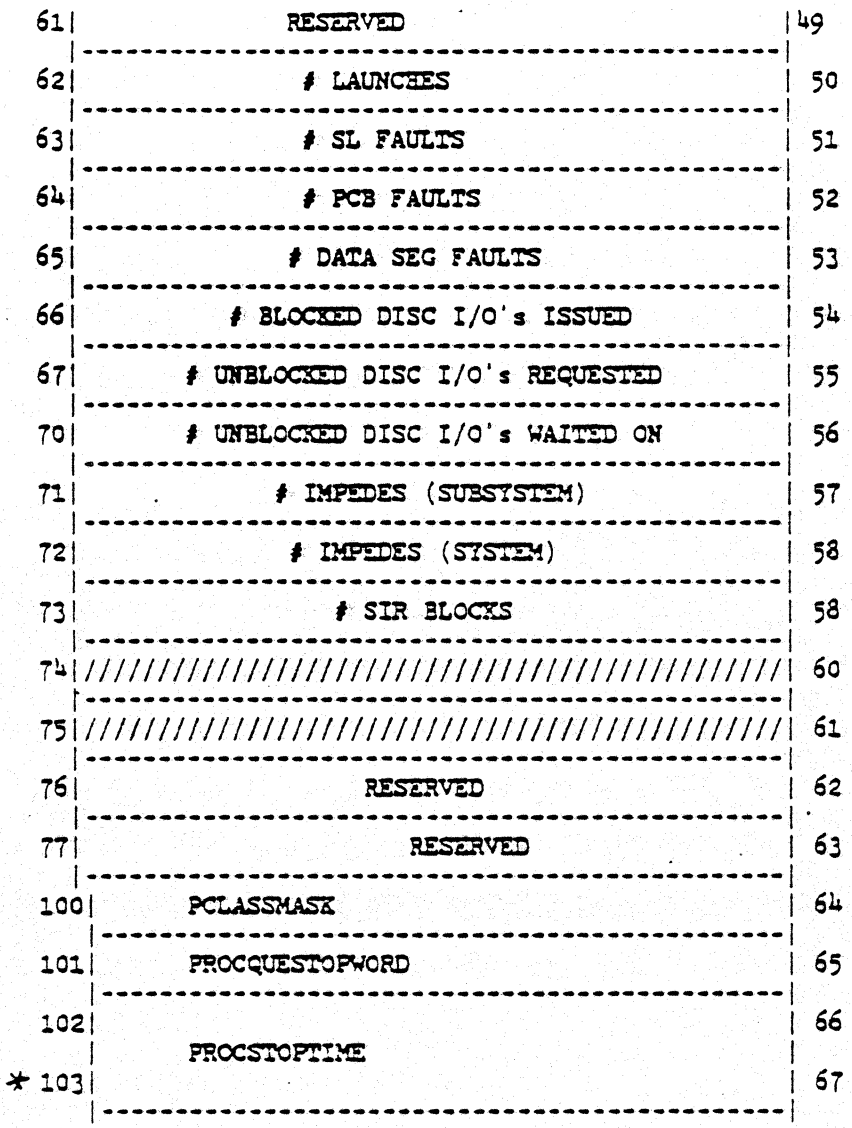
```
        if entry was not already in JDT
            then go to cont1
        else
            begin
                release the new DST entry just
                    obtained since we don't need it;
                set DST1 to DST# found in JDT;
            end;

cont:   set CCG;
        set LENGTH;

cont1:  place new entry in PREFIXED area (with
            PLDSEG (DST1, ... ));
        go to fin;

error:  set CCL;

fin:    establish return condition code;

    end of GETDSEG;
```

7.3.3 PXFIXED ASSIGNMENTS

The PXFIXED portion of the pcbx contains specific information and control information.



M45.91



* 4 word. bit map follows. Then XDS expansion area upto 384 wrds.

NOTES: P = 0 if opened by priv user
S = 1 if data seg is sharable

PCLASSMASK = BIT MASK OF CLASSES THIS PROCESS HAS ENABLED
PROCQUESTOPWORD.(0:4) = PROCESS PRIORITY: 7 => L QUEUE
6 => C QUEUE
2 => D QUEUE
1 => E QUEUE
.(4:12)= REASON STOPPED: 1 => STOP SEG FAULT
2 => STOP DISC WAIT
3 => BLOCKED I/O, NCN TERMINAL
4 => TERMINAL READ
5 => STOP IMPEDE
6 => STOP ACTIVE
PROCSTOPTIME = DBL WORD TIMESTAMP OF WHEN PROCESS STOPPED FOR REASON GIVEN IN PROCQUESTOPWORD

M.4C

PXFILE AREA OF STACK

* THE FILE SYSTEM'S DATA BASE AREA

* CONSISTS OF:

  OVERHEAD
  CONTROL BLOCK TABLE
  ACTIVE FILE TABLE
  AVAILABLE SPACE

* A CONTROL BLOCK CONTAINS INFORMATION ON THE FILE THAT IS BEING ACCESSED AND ALSO, INFORMATION ON HOW THE FILE SHOULD BE ACCESSED.

* EVERY FILE ACCESSED BY A PROCESS WILL HAVE AN ENTRY IN THE ACTIVE FILE TABLE. THIS ENTRY IS USED BY THE FILE SYSTEM TO CHARACTERIZE THE FILE ACCESS AND TO GIVE THE LOCATION OF ITS CONTROL BLOCKS.

LAB

IN THE CLASS DUMP, TRACE THROUGH THE POINTERS IN THE PCBX FOR SEVERAL PROCESS STACKS. LOCATE THE PARTS OF THE PCBX AND THE STACK REGISTERS.

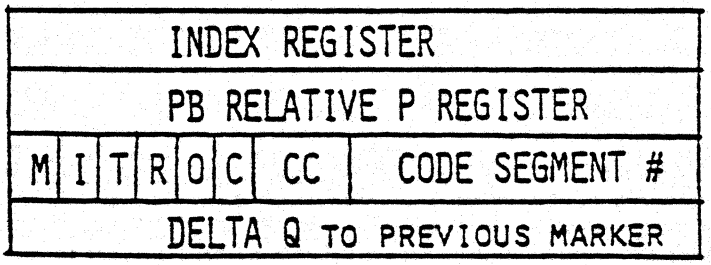
WE HAVE SEEN HOW MPE HANDLES CODE SEGMENTS DURING A PCAL BY USING THE SEGMENT TRANSFER TABLE. BUT HOW DOES MPE KEEP TRACK OF CONDITIONS ON THE STACK PRIOR TO THE PCAL SO THAT THE CONDITIONS MAY BE RETURNED UPON EXITING THE PCALED PROCEDURE?

BY MEANS OF THE STACK MARKER.

STACK MARKER

A STACK MARKER IS A GROUP OF DATA PLACED ON THE TOP OF THE STACK BY FIRMWARE TO MARK THE SET OF CONDITIONS NEEDED TO RESTORE THE STACK REGISTERS WHEN THE CALLED PROCEDURE COMPLETES. SPECIFICALLY, A STACK MARKER IS PLACED ON THE STACK WHEN A PCAL (PROCEDURE CALL) INSTRUCTION IS EXECUTED; AND IS REMOVED WHEN AN EXIT (RETURN TO CALLER) INSTRUCTION IS EXECUTED.

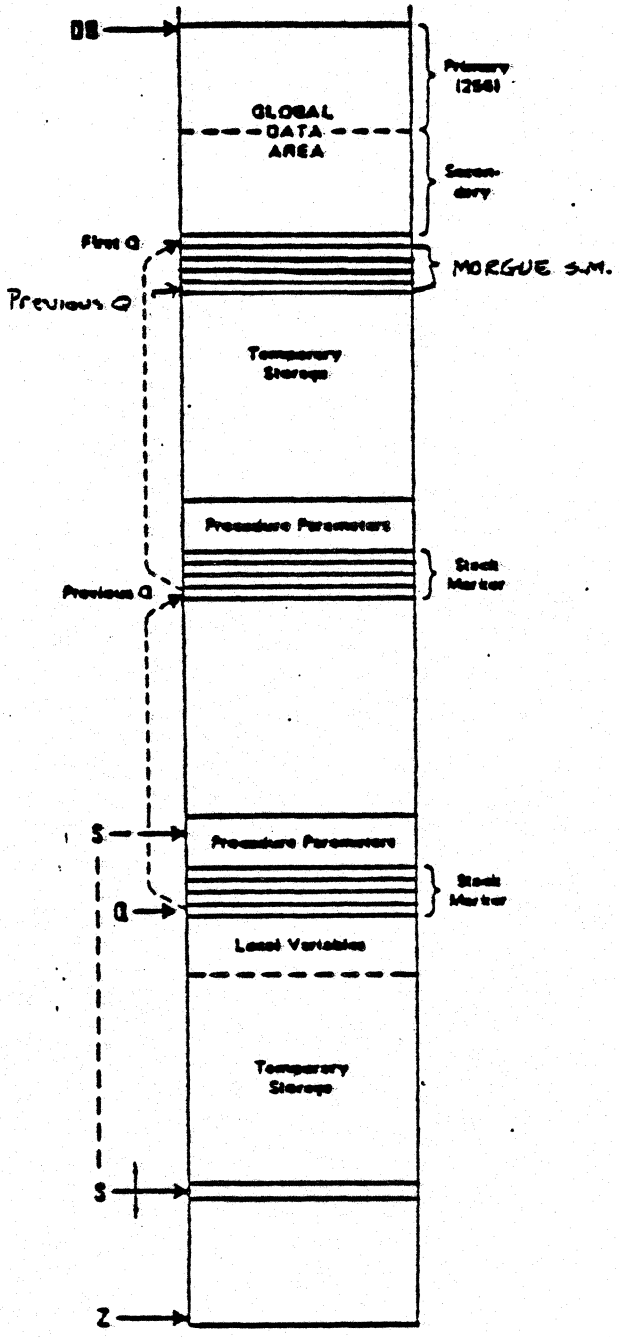
A STACK MARKER IS FOUR WORDS LONG, AND CONTAINS THE FOLLOWING:



IT IS IMPORTANT TO REMEMBER:

> A STACK MARKER IS PLACED ON THE STACK WHEN LEAVING A PROCEDURE.

STACK MARKERS



STACK MARKERS-DUMP EXAMPLE

```
$$$$$$$$ DSI  7o $$$$$$$$
*****APLBX: ************************************************************************
***PXGLOBAL:
073225: 000444 000444 177777 000024 000024 000063 006045 000000
***PXFIXED:
073233: 000120 000027 002357 000000 000000 177777 000000 000000  073243: 000000 000000
073253: 000000 000000 000000 000000 007000 000000 000000 007000  073263: 000000 000066
073273: 000000 000000 000000 000000 000000 000000 000000 000000  073303: 000000 000000
LINES 073313 - 073332 SAME AS ABOVE

073333: 000000 000000 000000 000000 000000 000000 000000 000000  073343: 000000 000000
***PXFILE: (CONTAINS NO CONTROL BLOCKS)
073353: 000310 000000 000000 000000 000000 000000 000000 000000  073363: 000000 000000
073373: 000000 000000 000000 000000 000000 000000 000000 000000  073403: 000000 000000
LINES 073413 - 073652 SAME AS ABOVE

073653: 000000 000000 000000 000000 000000 000000 000000 000000
**PXPOINTERS:
073663: 000000 000314 000434 000444
****DL REGISTER: ************************************************************************
****DB REGISTER: ************************************************************************
073667(000000): 000000 Q=

073670(MARKER): 000000 001015 140041 000004 MURGUE (37)                          -------
073674(000005): 000000 000000 000020 000100

073700(MARKER): 000001 000710 140437 000010 MKTQ (35)                            -------
073704(000015): 000000 001000 001260 000000 001000

073711(MARKER): 177756 017471 101074 000011 KLRNFLC (15)                         -------
****S REGISTER: ************************************************************************
073715(000026): 000000 001000 141037 000025 000000 000031 000400 177777 000110 000120 0
073731(000042): 000000 000000 046104 042526 020043 050040 047117 052040 051105 040504 0
073745(000056): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 0
```

SUMMARY

1. WE HAVE A USER PROGRAM . . .

2. USER CALLS AN INTRINSIC . . .

3. WE NOW EXIT BACK TO USER . . .

EXAMINING A STACK MARKER

THE QUESTION MOST ASKED WHEN EXAMINING A STACK MARKER IS:

WHAT CODE WAS EXECUTING ??

THE ANSWER TO THIS QUESTION PROVIDES A NUMBER OF CLUES TO WHAT IS HAPPENING. FOR INSTANCE, WAS THE CODE A USER PROGRAM, MPE, OR A SUBSYSTEM? IF IT WAS MPE, WHAT PROCEDURE WAS EXECUTING?

TO ANSWER THIS QUESTION, WE MUST FIRST FIGURE OUT WHAT SEGMENT (CODE SEGMENT) THE STACK MARKER WAS LEAVING. THIS INFORMATION CAN BE EXTRACTED FROM THE LOW ORDER 8 BITS OF THE STATUS REGISTER. IF THIS VALUE IS BETWEEN %301 AND %377, THEN THE SEGMENT IS IN THE CST EXTENSION, AND REPRESENTS CODE FROM A PROGRAM. IF THIS VALUE IS BETWEEN 1 AND %300, THEN THE SEGMENT IS IN THE CST, AND REPRESENTS A SEGMENT IN AN SL (MPE AND PRIV SUBSYSTEMS ARE IN AN SL, SPECIFICALLY SL.PUB.SYS). IF THE SEGMENT IS IN THE CST, WE MAY LOOK AT THE FORMATTED CST TABLE, OR THE LOADMAP (PRINTED BY DUMPJOB) TO FIND THE SEGMENT WHOSE NAME CORRESPONDS TO THE SEGMENT NUMBER. IF THE CST AND LOADMAP DO NOT HAVE NAMES THAT GO TO THIS HIGH OF A NUMBER, THEN THE SEGMENT IS A DYNAMICALLY ALLOCATED SEGMENT IN AN SL - BUT WE HAVE NO EASY WAY OF FINDING OUT WHAT (ASKING THE CUSTOMER IF HE HAS SEGMENTS IN AN SL). WE KNOW THE VALUE OF P IN THE MARKER, BUT HOW DO WE RELATE THIS TO A PROCEDURE? TO FIND THE PROCEDURE NAME, WE WILL NEED A PMAP OF THE SEGMENT IN QUESTION.

PMAP

A PMAP FOR A SEGMENT SHOWS THE PHYSICAL LOCATION WITHIN THE SEGMENT OF ALL THE PROCEDURES THAT RESIDE IN THE SEGMENT. THE PMAP FURTHER SHOWS THE STARTING LOCATION (SEGMENT RELATIVE) OF EACH PROCEDURE. WE MUST FIND A PROCEDURE WHOSE ADDRESS FOR "CODE" IS JUST LESS THAN THE DESIRED P ADDRESS WE ARE SEARCHING FOR. WHEN WE FIND SUCH A PROCEDURE, WE SUBTRACT THE BASE ADDRESS FOR CODE FROM THE ADDRESS P IN THE MARKER TO OBTAIN THE PROCEDURE RELATIVE ADDRESS OF THE DESIRED MARKER.

EXAMPLE:

SUPPOSE WE HAVE A STACK MARKER:

000000 000677 107445 000023

FROM THE CST IN THE EXAMPLE DUMP, WHAT IS THE NAME OF THE SEGMENT THAT WAS EXECUTING WHEN THIS MARKER WAS PLACED ON THE STACK??

PMAP

ASSUME THAT THIS IS THE PMAP FOR THE SEGMENT IN THE STACK MARKER ON THE PREVIOUS PAGE. WHERE DID THE PCAL OCCUR?

MAIN	0			
NAME	STT	CODE	ENTRY	SEG
NRIO	1	0	0	
TERMINATE*	2			?
SEGMENT LENGTH		4		
NRIO	1			
NAME	STT	CODE	ENTRY	SEG
IOTABLEINFO	1	0	0	
SDFINIT	2	136	136	
VALIDDEVTYPE	3	137	161	
SETSYSDB	21			?
CHECKLDEV	22			?
RESETDB	23			?
ASCII	24			?
GETDEVINFO	25			?
INITIO	4	343	364	
ATTACHIO	26			?
IOMESSPROC	5	470	470	
AWAKE	27			?
SUDDENDEATH	30			?
GENMSG	31			?
AWAKEIO	32			?
LOG	33			?
RETURNTRUE	34			?
WAIT	35			?
ABORTIOX	6	670	670	
RETURNIOQ	36			?
WAITFORIO	37			?
ABORTIO	7	774	774	
ABORTPROCIO	10	1140	1140	
GETSYSBUF	11	1273	1273	
HELP	40			?
IOIMPEDE	41			?
FGETSYSBUF	12	1273	1276	
TERMINIT	13	1446	1453	
MPXWRITE	42			?
DSETCONTROL	43			?
IOFAILURE	44			?
INITCHANNEL	14	1624	1624	
MPXCONTROL	45			?
SETTERMTYPE	15	1636	2004	
RESETBREAKBITS	16	2150	2150	
IOCONTROL	17	2173	2173	
DEVICESTATUS	20	2211	2211	
SEGMENT LENGTH		2324		

CST 45

000000 000677 107445 000023

PB Relative P=677

677
-670
7

∴ 7 words into code for ABORTIOX

A CONCLUDING NOTE ON STACK MARKERS:

THE PROCESS YOU HAVE LEARNED HAS BEEN AUTOMATED FOR YOU BY DPAN. THE STACK MARKERS IN THE EXAMPLE DUMP SHOW THE FOUR WORD MARKER PLUS THE FOLLOWING INFORMATION:

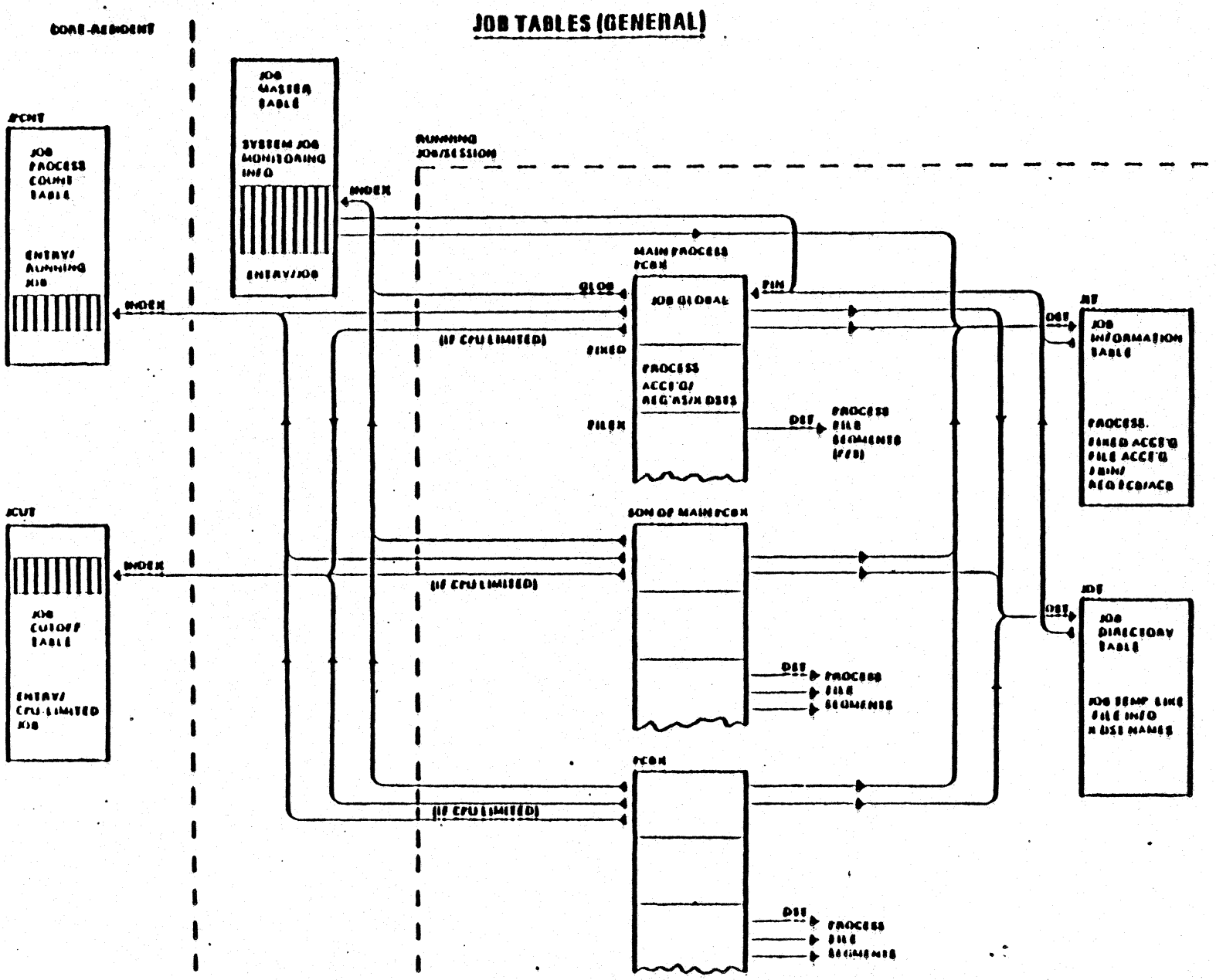
- THE SEGMENT NUMBER (EXTRACTED FROM STATUS)
- THE SEGMENT NAME (FROM THE LOADMAP)

THIS INFORMATION IS PROVIDED FOR YOU IF THE LOADMAP IS CORRECT FOR THE DUMP.

STACK MARKERS CAN BE READ FROM TOP TO BOTTOM TO READ MOST RECENT TO OLDEST (THE MOST RECENT MARKER IS ON TOP).

JOB RELATED TABLES

* JOB MASTER TABLE (JMAT)
* JOB PROCESS COUNT TABLE (JPCNT)
* JOB CUT-OFF TABLE (JCUT)
* JOB INFORMATION TABLE (JIT)
* JOB DIRECTORY TABLE (JDT)


JOB TABLES (GENERAL)
CORE-RESIDENT
JOB MASTER TABLE
SYSTEM JOB MONITORING INFO
ENTRY/JOB
#CNT
JOB PROCESS COUNT TABLE
ENTRY/ RUNNING JOB
RUNNING JOB/SESSION
INDEX
MAIN PROCESS PCB
JOB GLOBAL
PROCESS ACCESS/ REASSIGN DSES
GLOB
PIN
DST
(IF CPU LIMITED)
FIXED
FILEN
PROCESS FILE SEGMENTS (FDS)
JCUT
JOB CUTOFF TABLE
ENTRY/ CPU-LIMITED JOB
SON OF MAIN PCB
PCB
JIT
JOB INFORMATION TABLE
PROCESS.
FIXED ACCESS
FILE ACCESS
MAIN
REASSIGNS
JDT
JOB DIRECTORY TABLE
JOB TEMP FILE FILE INFO
DSE NAMES

JOB MASTER TABLE (JMAT)

* ONE TABLE PER SYSTEM
* ONE TABLE ENTRY PER JOB/SESSION
* POINTED TO BY DST %31
* ENTRY 0 (HEADER) CONTAINS SYSTEM JOB INFORMATION IN ADDITION ABOUT THE TABLE ITSELF
* DIFFERENT TYPES OF ENTRIES TO DEFINE STATE OF PARTICULAR JOB/SESSION
* CURRENT STATE DESCRIPTORS OF A JOB/SESSION ARE:

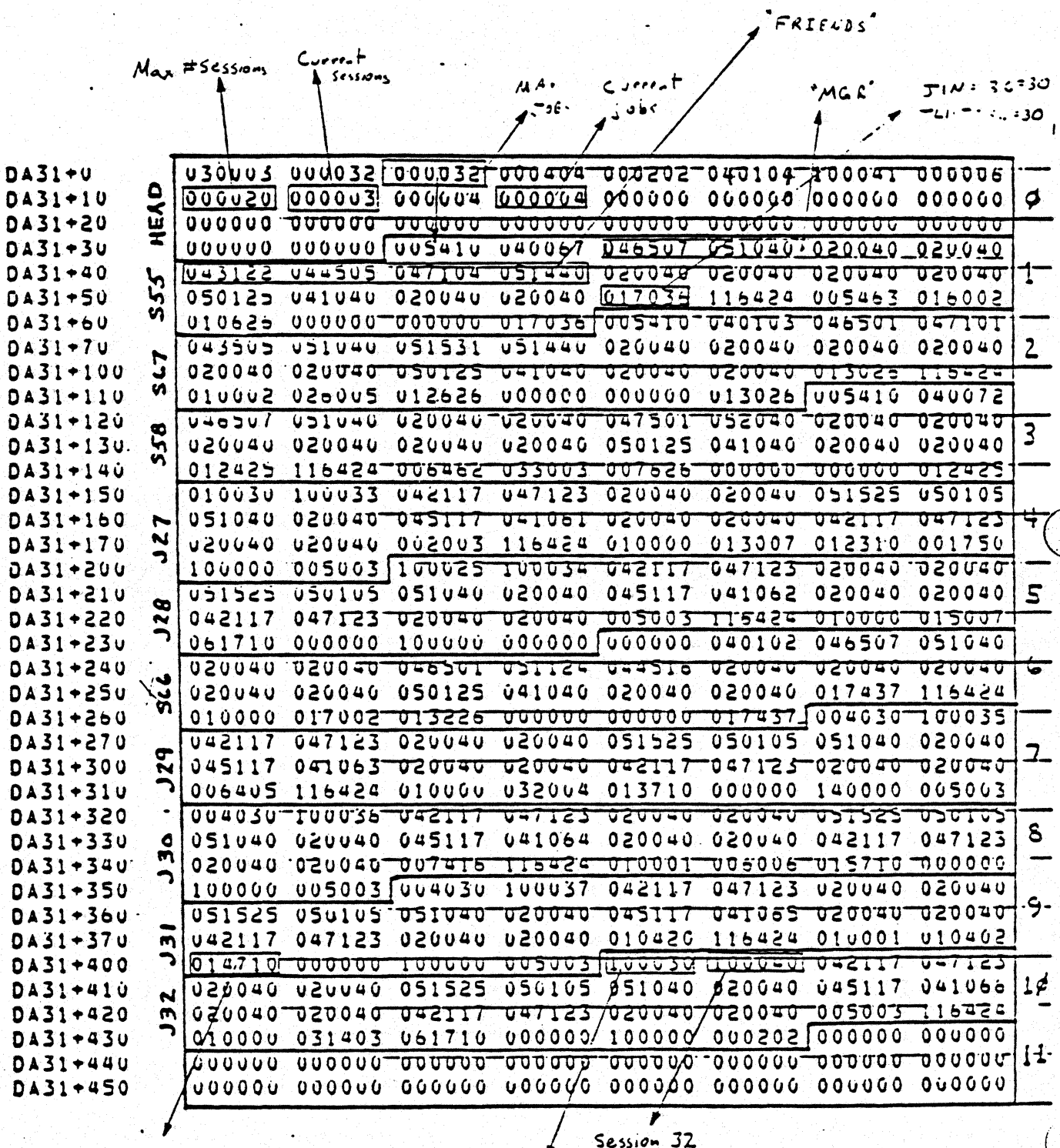
%0 –	UNUSED
%1 –	JOB/SESSION INTRODUCED
%2 –	JOB/SESSION EXECUTING
%3 –	TERMINATED OR TERMINATING
%4 –	SUSPENDED
%40–	WAITING (DEFERRED)
%50–	ERROR
%60–	BEING INITIALIZED (C.I.)

:SHOWJOB ANALYZES THE JMAT

```
JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#J27    SUSP       10S  LP       TUE  4:00P  JOB1,DONS.SUPER
#S55    EXEC       30   30       TUE 11:51A  MGR.FRIENDS
#S65    EXEC       22   22       TUE  3:55P  DONS.SUPER
#S58    EXEC       21   21       TUE  1:50P  MGR.OAT
#J29    EXEC       10R  LP       TUE  4:00P  JOB3,DONS.SUPER
#J30    EXEC       10S  LP       TUE  4:01P  JOB4,DONS.SUPER
#J31    EXEC       10S  LP       TUE  4:01P  JOB5,DONS.SUPER
#J32    WAIT:1   8 10S  LP       TUE  4:00P  JOB6,DONS.SUPER
#J28    WAIT   0 5 10S  LP       TUE  4:00P  JOB2,DONS.SUPER

9 JOBS:
    0 INTRO
    2 WAIT; INCL 1 DEFERRED
    6 EXEC; INCL 3 SESSIONS
    1 SUSP
JOBFENCE= 6; JLIMIT= 4; SLIMIT= 16
```

HEWLETT PACKARD

JMAT



JOB INFORMATION TABLE (JIT)

*ONE TABLE PER JOB/SESSION - EXTRA DATA SEGMENT

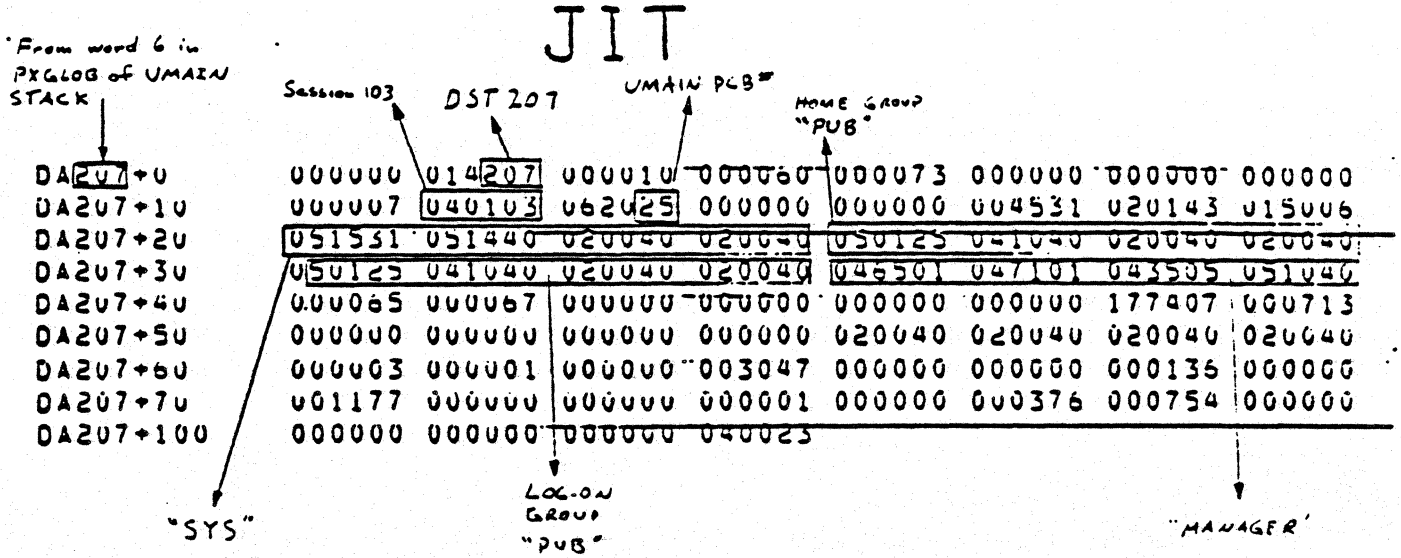
*DST INDEX IN PCBX(6),(6:15)

*EACH TABLE IS %75 WORDS

*CONTAINS JOB/SESSION RELATED INFORMATION

- ACCOUNTING
- DIRECTORY POINTERS
- JOB ATTRIBUTES
- JOB CAPABILITIES
- PRIVATE VOLUME DATA

JIT


From word 6 in PXGLOB of UMAIN STACK
Session 103
DST 207
UMAIN PCB#
HOME GROUP "PUB"
DA207+0 000000 014207 000010 000060 000073 000000 000000 000000
DA207+10 000007 040103 062025 000000 000000 004531 020143 015006
DA207+20 051551 051440 020040 020040 050125 041040 020040 020040
DA207+30 050125 041040 020040 020040 046501 047101 043505 051040
DA207+40 000065 000067 000000 000000 000000 000000 177407 000713
DA207+50 000000 000000 000000 000000 020040 020040 020040 020040
DA207+60 000003 000001 000000 003047 000000 000000 000136 000000
DA207+70 001177 000000 000000 000001 000000 000376 000754 000000
DA207+100 000000 000000 000000 040023
"SYS"
LOG-ON GROUP "PUB"
"MANAGER"

JOB DIRECTORY TABLE
(JDT)

*ONE TABLE PER JOB/SESSION – EXTRA DATA SEGMENT

*DST INDEX IN PCBX(5),(6:15)

*INITIAL ALLOCATION IS %50 WORDS – EXPANDS WITH USAGE

*CONTAINS FIVE SUBTABLES

1. JOB DATA SEGMENT DIRECTORY (JDSD)

   *ONE ENTRY FOR EACH EXTRA DATA SEGMENT FOR THE PROCESS (USER XDS CAN HAVE NAME)

2. JOB TEMPORARY FILE DIRECTORY (JTFD)

   *ONE ENTRY FOR EACH TEMPORARILY OPENED FILE ($OLDPASS,$NEWPASS,ETC.)

3. JOB FILE EQUATION TABLE (JFEQ)

   *ONE ENTRY FOR EACH ":FILE" COMMAND

4. JOB LINE EQUATION TABLE (JLEQ)

   *ONE ENTRY FOR EACH ":CLINE" COMMAND

5. JOB CONTROL WORD TABLE (JJCW)

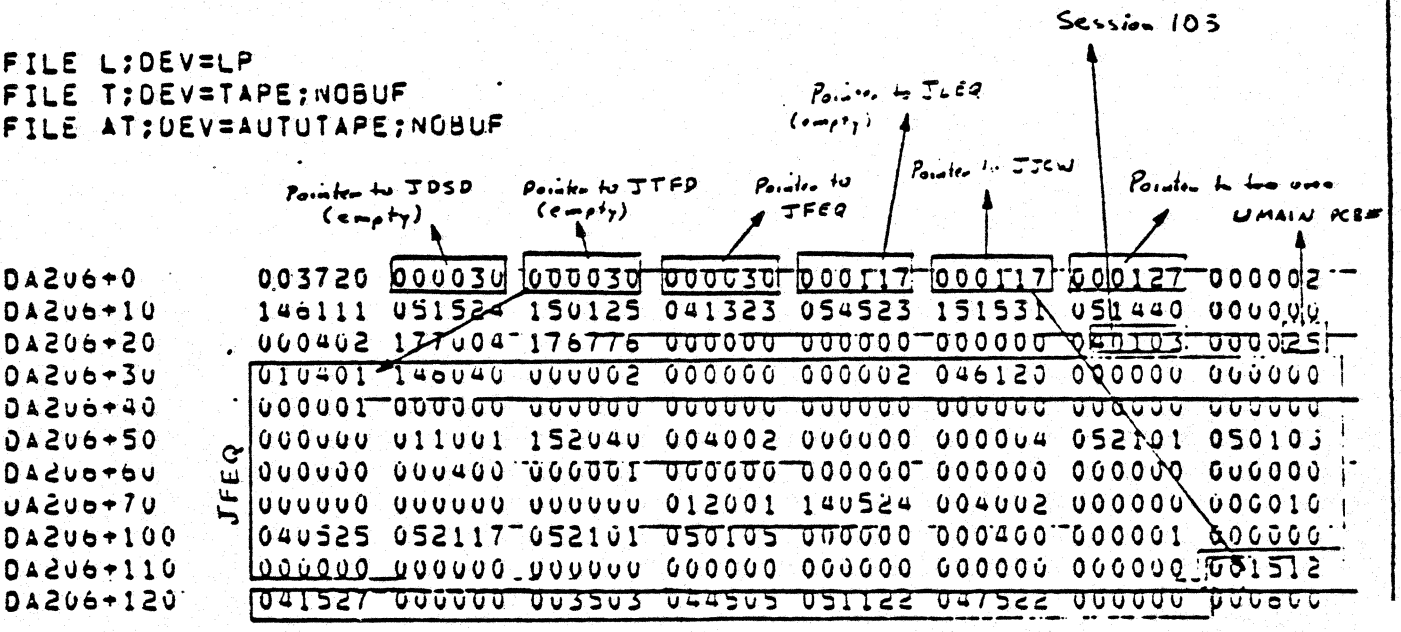
   *ONE ENTRY FOR EACH USER NAMED JCW

   *INITIALLY GIVEN TWO ENTRIES

      . JCW – SYSTEM JCW
      . CIERR – C.I. ERROR JCW

JDT

```
FILE L;DEV=LP
FILE T;DEV=TAPE;NOBUF
FILE AT;DEV=AUTUTAPE;NOBUF
```



JOB PROCESS COUNT TABLE

(JPCNT)

*ONE TABLE PER SYSTEM (CORE RESIDENT)

*ONE TABLE ENTRY PER RUNNING JOB

*POINTED TO BY DST %30 AND SYSGLOB %15

*EACH ENTRY IS ONE BYTE (8 BITS)

*ENTRY 0 IS MAXIMUM NUMBER OF RUNNING JOBS

*ENTRY 1 IS TOTAL NUMBER OF FREE ENTRIES

*ENTRY POINTED TO BY PCBX(4).(0:8)

JOB CUT-OFF TABLE (JCUT)

* ONE TABLE PER SYSTEM (CORE RESIDENT)
* ONE TABLE ENTRY PER CPU LIMITED JOB
* POINTED TO BY DST %44 AND SYSGLOB %13
* LIMIT SET BY

```
";TIME = (CPU SECS)" IN
    :JOB
    :HELLO
```

* ENTRY INDEX IN PCBX(7),(0:8)

Job Table Lab

1. Look in the JMAT header and look at:

   A) The session limit
   B) The job limit
   C) Next assignable session #/ job #

2. Find your entry in the JMAT:

   A) Display in ASCII and make sure you see your user and account name.

   B) Notice words 22 and 31. Are they the Same? why?

   C) What is the state of your job?

   D) What is your job main PIN?

3. Find your JIT:

   A) Does the job main PIN displayed in the JIT agree with the value in the JMAT?

   B) What user capabilities do you have?

   C) Change your user capabilities and see the effects.

4. Create some temporary files and file equations.

   A) Find your JDT and look at the Contents.

5. Give your session a time limit.*

   A) Look at your entry in the JCUT.

*Note--be sure to make your time limit long enough so that you can find your entry in the JCut before being logged off!

LAB

IN THE CLASS DUMP, TRACE THROUGH ALL OF THE JOB TABLES FOR THE CURRENT PROCESS, EXTRACTING THE IMPORTANT INFORMATION.

SIR TABLE

THE SIR TABLE (STANDS FOR SYSTEM INTERNAL RESOURCE) IS A TABLE OF SEMAPHORES (SWITCHES) THAT ALLOW USERS TO INSURE THAT THEY HAVE EXCLUSIVE ACCESS TO A RESOURCE (USUALLY A SYSTEM TABLE).

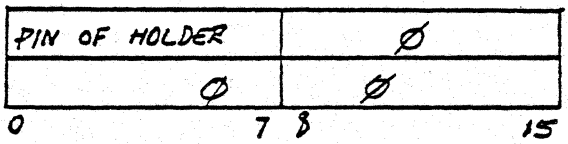
FOR EXAMPLE, BEFORE MAKING MODIFICATIONS TO THE JMAT, THE USER FIRST LOCKS THE JMAT SIR, THEN MODIFIES THE JMAT, AND THEN UNLOCKS THE JMAT SIR. IF SOMEONE ELSE TRIES TO LOCK THE JMAT SIR WHILE THE FIRST USER STILL HAS IT LOCKED, THE SECOND PROCESS WILL BECOME IMPEDED. THE DPAN SIR LISTING WILL CLEARLY SHOW WHAT PROCESS HAS THE SIR, ITS NAME, AND WHAT PROCESSES ARE IN LINE FOR THE SIR.

THE EXAMPLE (NON EMPTY) SHOWS SEVERAL ERROR CONDITIONS THAT ARE DETECTED AND PRINTED. MANY OF THE "LOCK UPS" ARE IN FACT CAUSED BY DEADLOCKS (MANY PROCESSES WAITING FOR SIRs, RINs AND/OR MEMORY). THESE LOCKUPS ARE EASY TO FALL INTO - THIS IS THE REASON FOR REQUIRING MULTIPLE RIN CAPABILITY FOR USERS TO PERFORM MULTIPLE LOCKS. (THE MR QUESTION IS A GOOD ONE TO ASK ABOUT IF PCB SHOWS WAIT ON RL OR RG).

THE SIR TABLE IS FIXED IN SIZE, AND SIR MEANINGS ARE PERMANENTLY ASSIGNED BY NUMBER TO SPECIFIC ENTRIES.

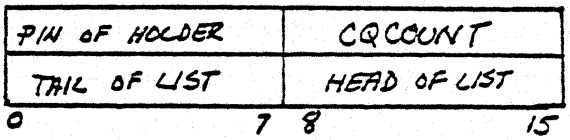
SIRS - SYSTEM INTERNAL RESOURCES

- CONTAIN TWO WORDS/ENTRY
- ONE ENTRY FOR EACH SIR
- ENTRY IS ZEROS IF FREE
- ENTRY IS:



FOR AN ENTRY THAT IS LOCKED BUT NO OTHER PROCESSES ARE REQUESTING IT.

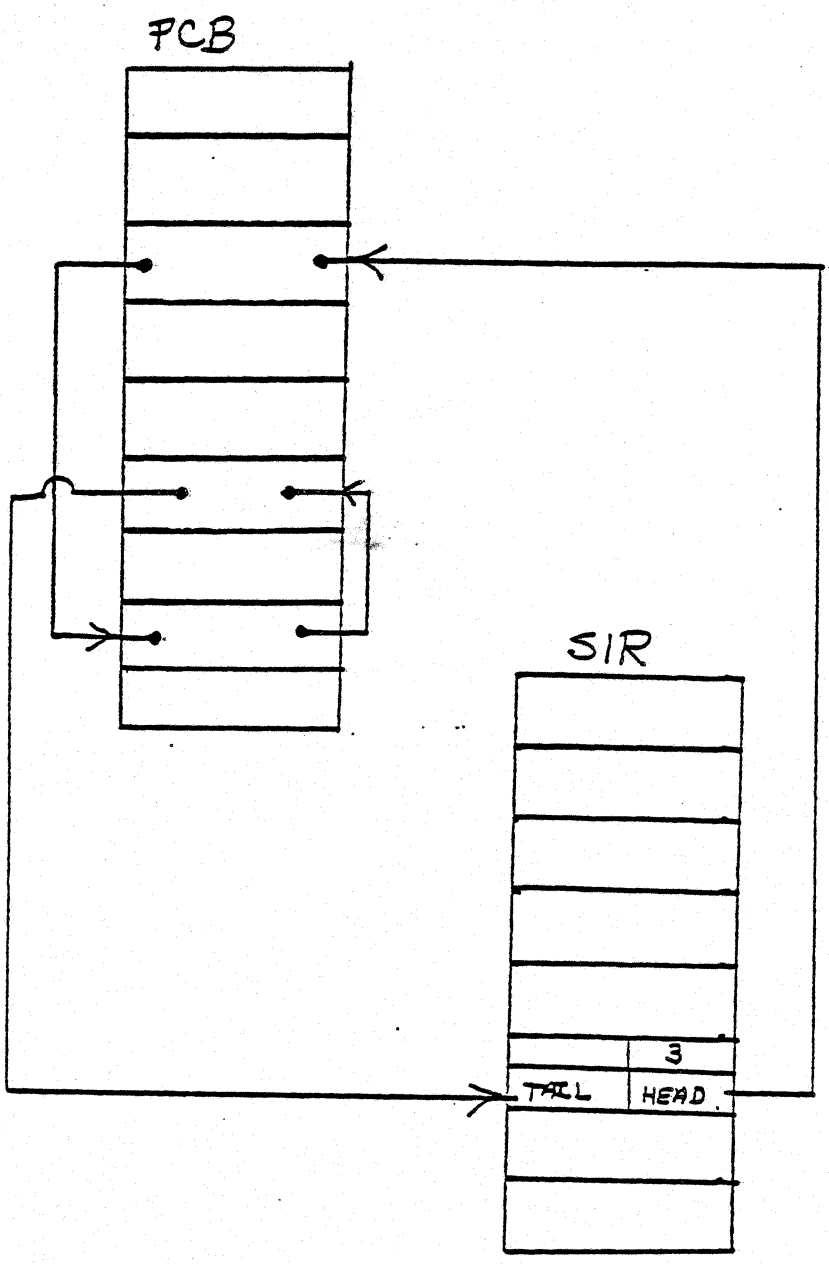
- ENTRY IS:



IF A PROCESS HAS A SIR AND THERE IS ONE OR MORE PROCESSES IMPEDED

CQOUNT - THE NUMBER OF PROCESSES IMPEDED FOR THIS SIR.

TAIL/HEAD OF IMPEDED LIST - PCB INDEX OF FIRST THROUGH LAST OF PROCESS IMPEDED FOR THIS SIR.


PCB
SIR
3
TAIL
HEAD

SE 335 QUIZ

1. What four (4) types of files must be in the SUPPORT account in order to compile an MPE IV source listing using the :STREAM command?

2. What program must be run to find out in what MPE modules procedures reside?

3. In what file do the MPE modules 50-99 reside?

4. What are the basic components of an MPE process?

5. What two locations contain the address of the beginning of the CODE SEGMENT TABLE?

6. Where does the number of processes running a given program reside?

7. How would you find the location of the JOB MASTER TABLE in the Memory Dump (not using the index).

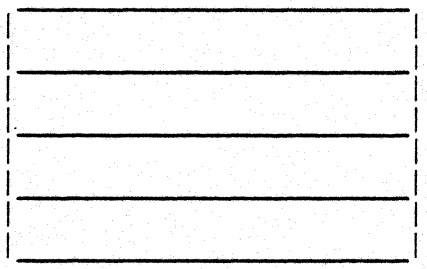
How would you find a given process' entry in the JOB MASTER TABLE in a DUMP?

8. Where would you look in the DUMP to find a process' scheduling priority?

9. What segment number will be in the STATUS REGISTER for a one segment program that is currently executing?

10. In what table would you find the DST-relative address of the block of XCST entries for a given program?

11. What information is found in a four (4) word stack marker?



12. What table is used to find the PB-relative location of the current program pointer of a called program?

    Where does the table reside?

13. What are the four major parts of a process' PCBX area?

14. In what table could you find the capabilities of a user logged on to the system?

15. What data storage can a process use other than its stack?

DEBUG

DEBUG

** MODE:

* THE REPRESENTATION MODE FOR OUTPUT VALUES IN THE D, M AND "=" COMMANDS HAS BEEN EXTENDED TO PERMIT THE MNEMONIC DISPLAY OF CODE.

* "C" IS USED TO INDICATE THIS MODE.

EX: ?D P,C
P+0 045403 LOAD Q+3,X

HEWLETT PACKARD

DEBUG

** EXPANSION OF B, C AND R COMMANDS:

* ALLOWS USERS TO SET AND CLEAR BREAKPOINTS IN OTHER PROCESSES:

B [[PIN.] [[SEGTYPE] SEGMENT.] OFFSET [:[@][CNT]] [,...]

C [[PIN.] [[SEGTYPE] SEGMENT.] OFFSET [,...]

R [[PIN.] [[SEGTYPE] SEGMENT.] OFFSET [:[@][CNT]] [,...]

PIN THE PIN OF THE PROCESS IN WHICH THE BREAKPOINT IS TO BE SET --PM ONLY.

EXAMPLE: ?B 15.3.2 BREAK IN PIN #15, SEGMENT=3, OFFSET OF 2.

DEBUG

** EXPANSION OF B AND R COMMANDS :

* TO EXPAND THE CONDTIONAL BREAKPOINT FEATURE:

B [[PIN.] [[SEGTYPE] SEGMENT.] OFFSET [:[@] [COND]] [,...]

R [[PIN.] [[SEGTYPE] SEGMENT.] OFFSET [:[@] [COND]]

EXAMPLE: ?B 3.2:DB+3=5

BREAK IN CURRENT PROCESS, SEGMENT=3, OFFSET OF 2, WHEN THE LOCATION DB+3 IS EQUAL TO 5.

PRIV-MODE DEBUG LAB

WRITE A PROGRAM THAT CALLS DEBUG AND FIND:

A. JOBTABLE INFO FOR YOUR PROCESS

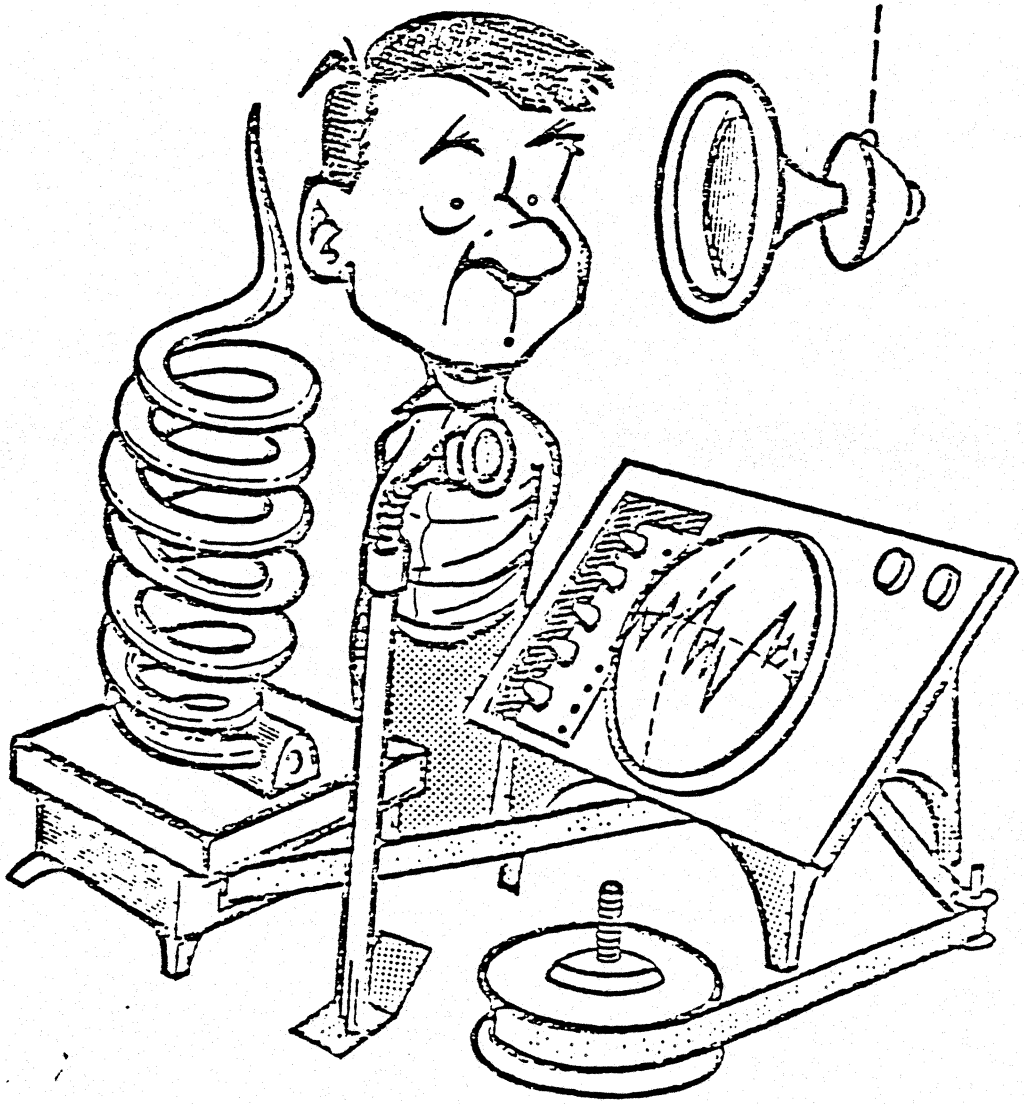
B. JOBTABLE INFO FOR YOUR PROCESS' FATHER.

JOBTABLE INFO TO FIND:

1. JOB TYPE & STATE
2. YOUR USER, GROUP & ACCT.
3. PRIORITY
4. JIT DST #
5. USER CAPABILITIES
6. ANY FILE EQUATIONS?
7. HOW ABOUT JOB CONTROL WORDS?
8. ANY TEMPORARY FILES?
9. NUMBER OF SESSIONS AND JOBS.
10. WHAT IS YOUR JMAT INDEX, JIT & JDT DST #

DIRECTORY SECRETS

(Who uses it and what is inside?)



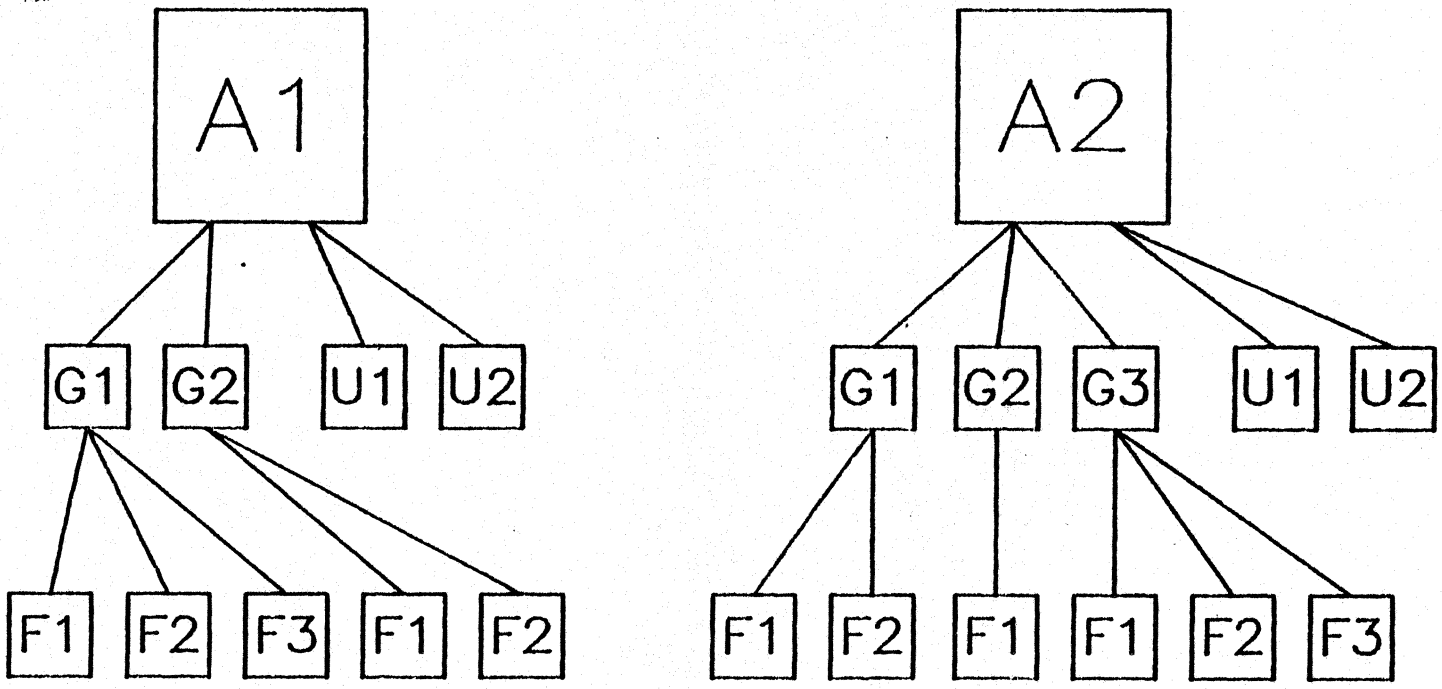
DIR-1

THE DIRECTORY

What is it
Why does it exist
How does it work
Why do we care

Reference
Systems Tables Ref. Manual
(32002-90003)

ACCOUNTING STRUCTURE



ACCOUNT ENTRY

- Name
- Password
- Group Pointer
- User Pointer
- Capabilities/Attributes
- Disc Space Count/Limit
- CPU Time Count/Limit
- Connect Time Count/Limit
- File Security

GROUP ENTRY

- Name
- Password
- Counts/Limits
- File Index Pointer
- Security
- Capability

USER ENTRY

- Name
- Password
- Capability
- Local Attributes
- Max Job Priority
- UDC Exist Flag

FILE LABEL

- NAME
- Lockword
- Security Matrix
- Creation Date
- Access Date
- Modification Date
- File Code
- File Limit
- Rec Size
- Block Size
- Extent Map

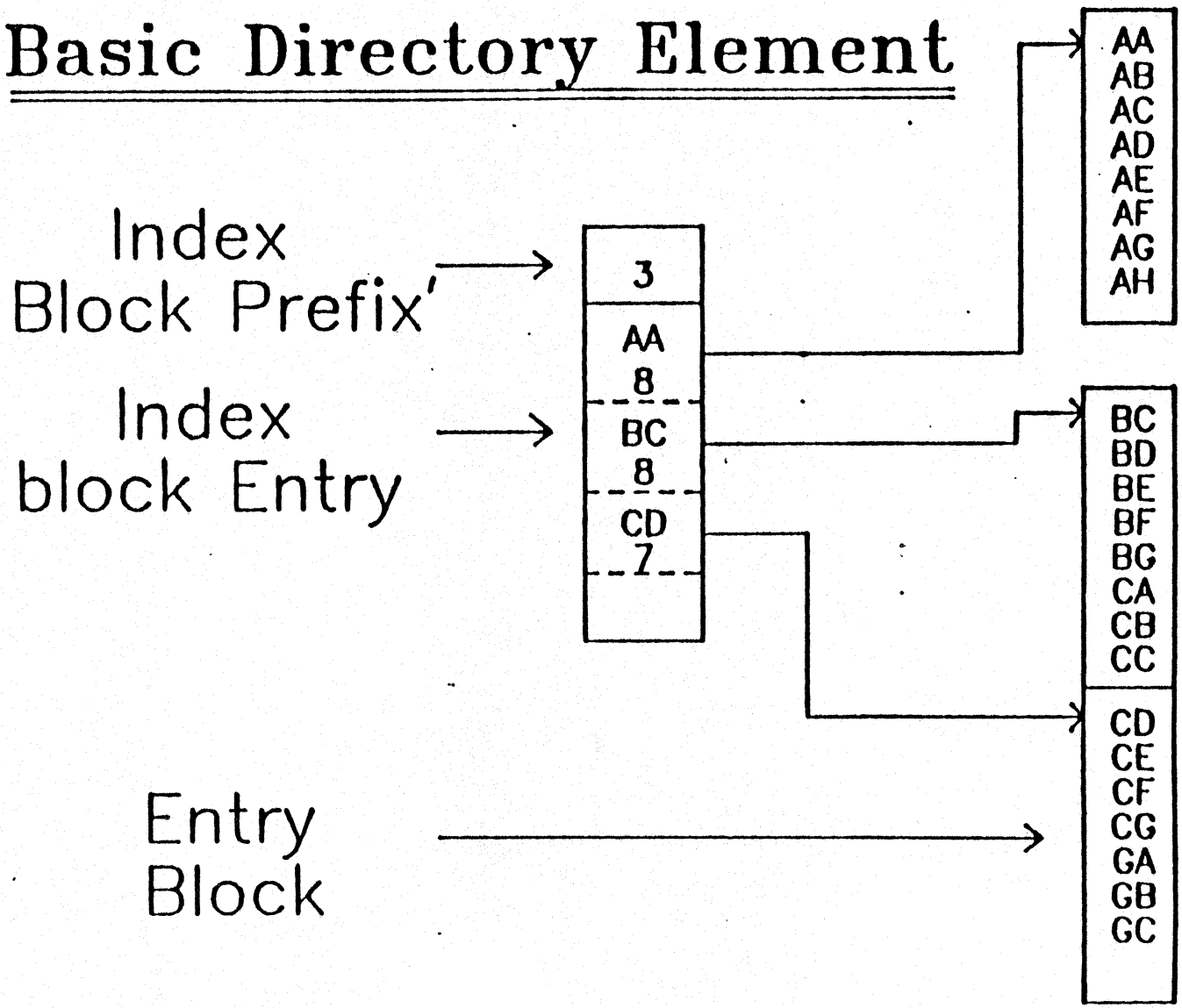
Reference:
Software Pocket Guide
(30000-90049)

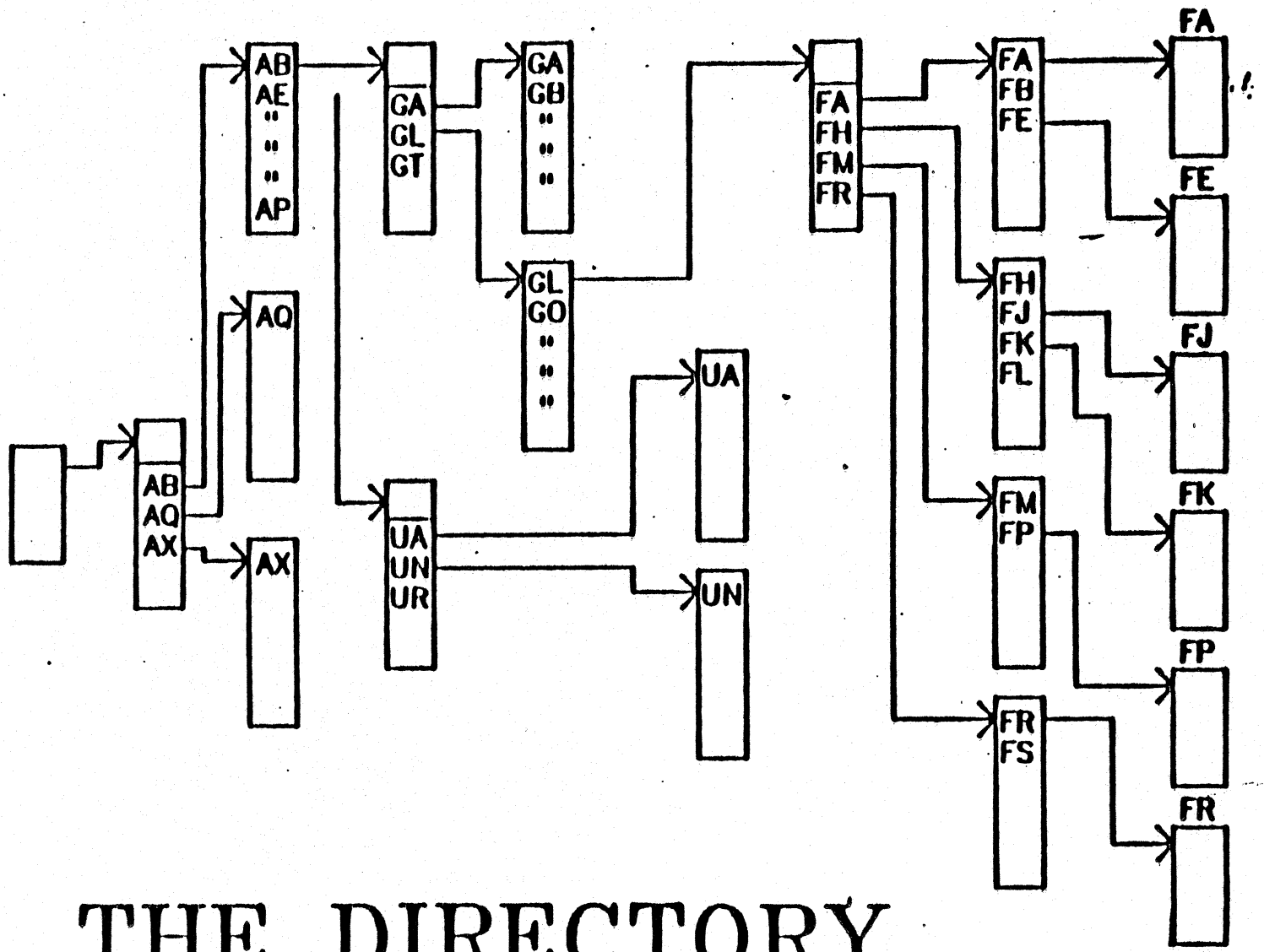
FILE ENTRY

Name – 4 words

Label Pointer – 2 words

Basic Directory Element

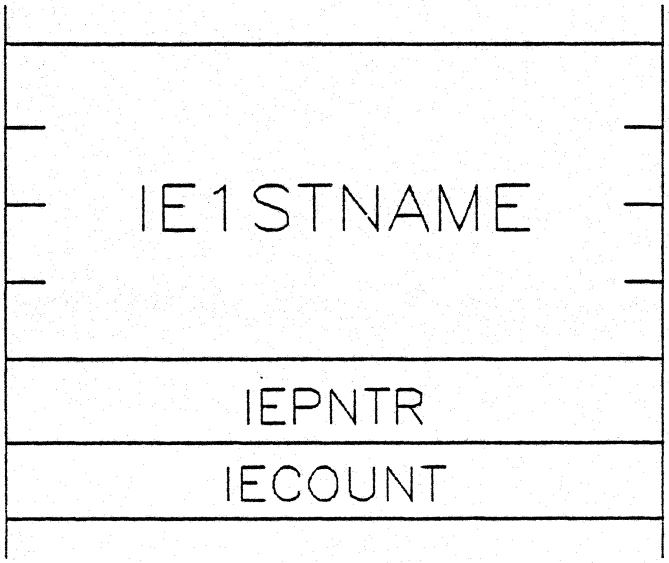




THE DIRECTORY

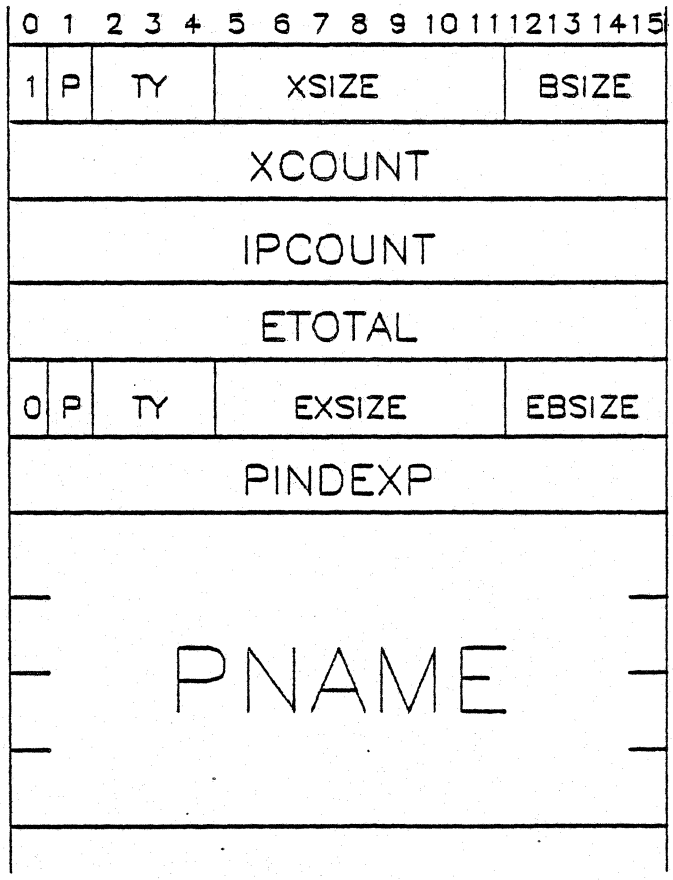
INDEX ENTRY

(6 Words)

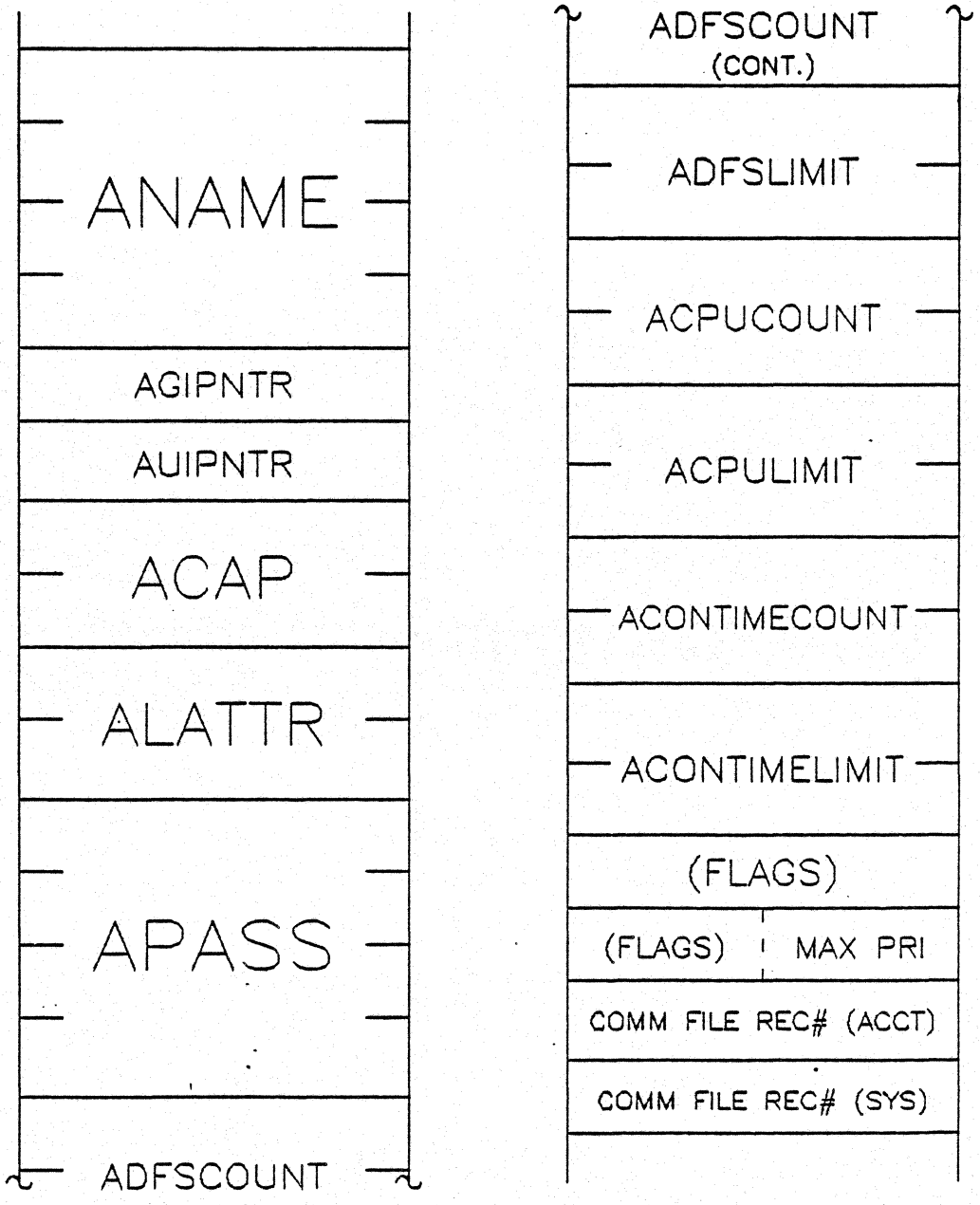


INDEX BLOCK PREFIX

(10 WORDS)

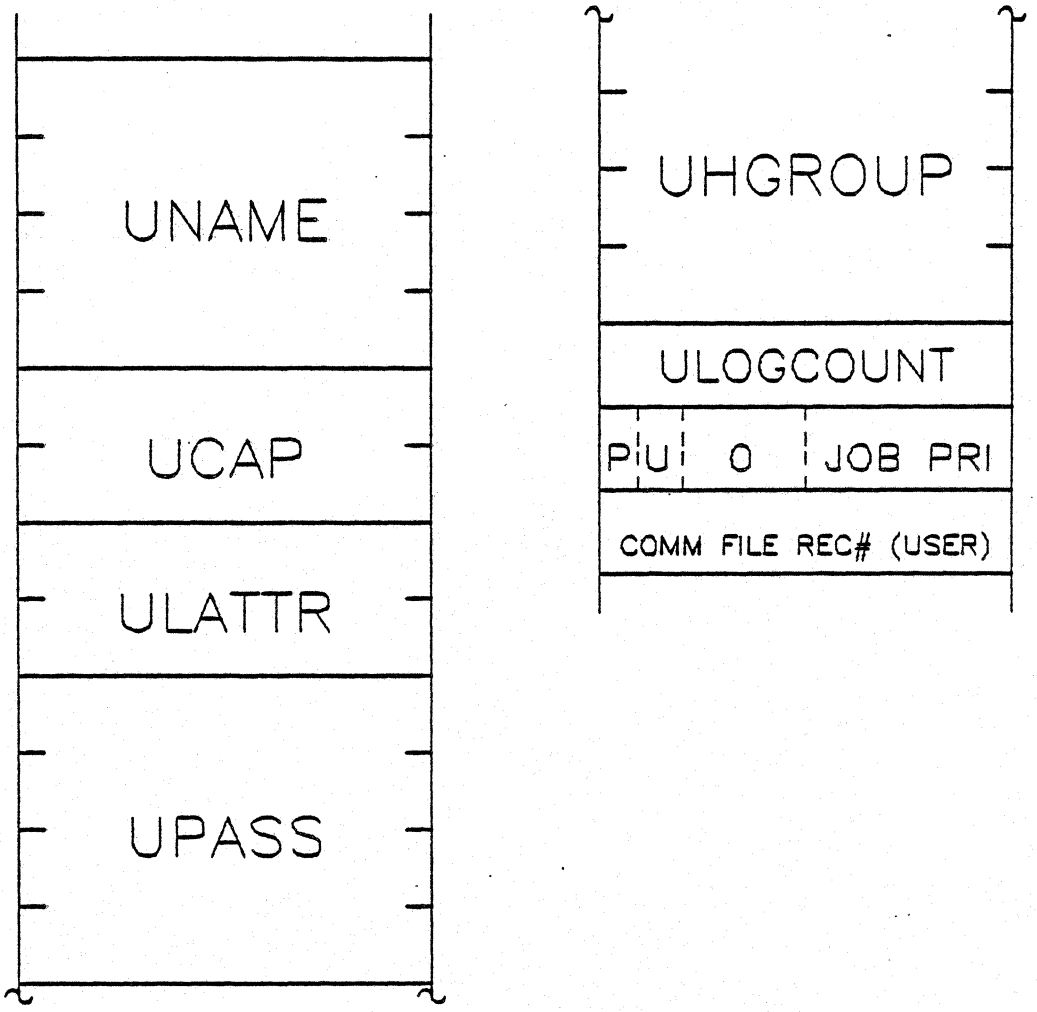


ACCOUNT ENTRY
(30 WORDS)

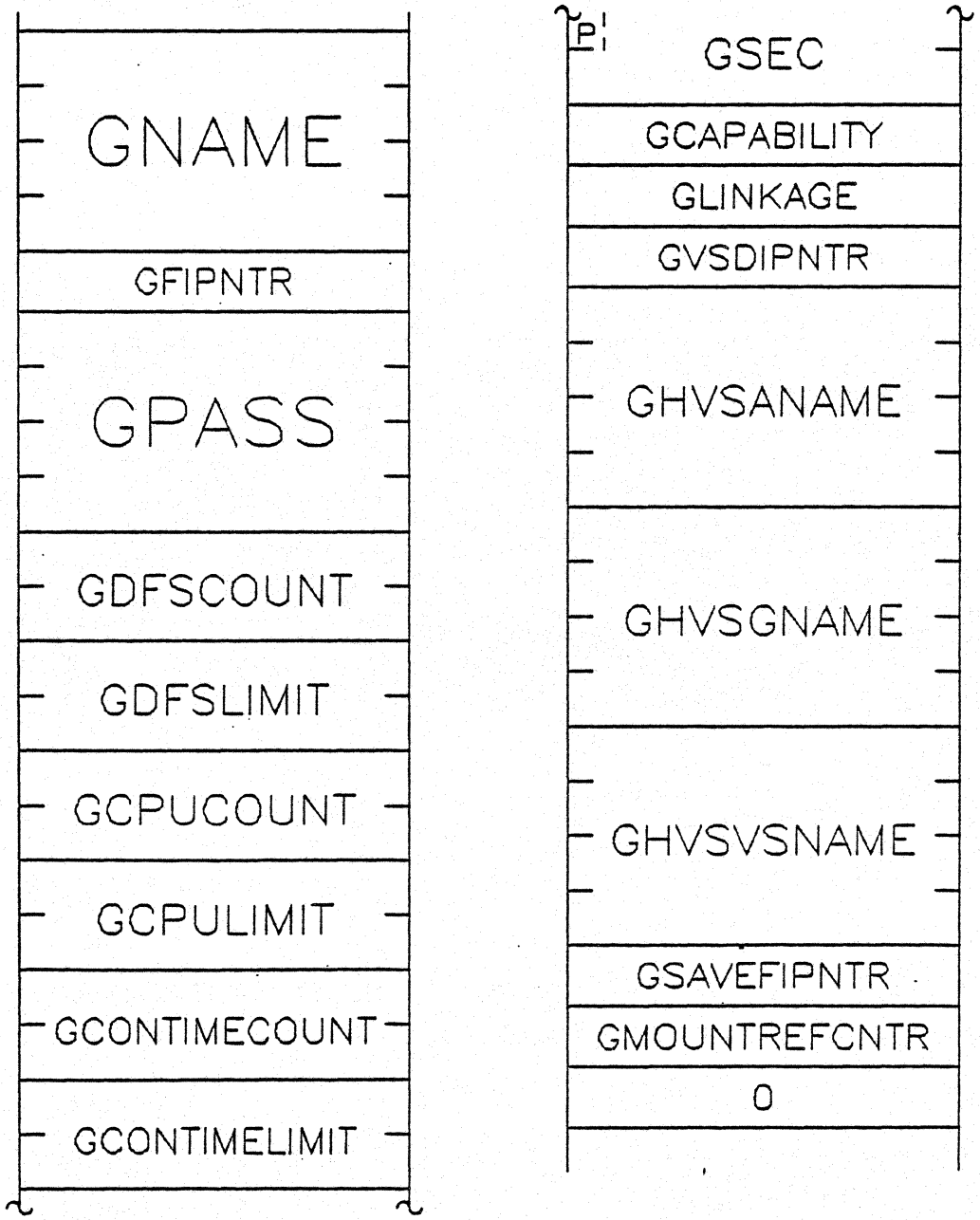


USER ENTRY

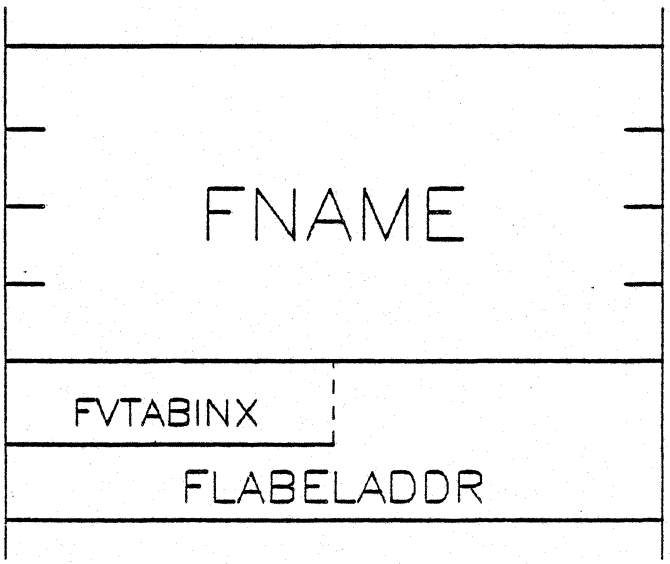
(19 WORDS)

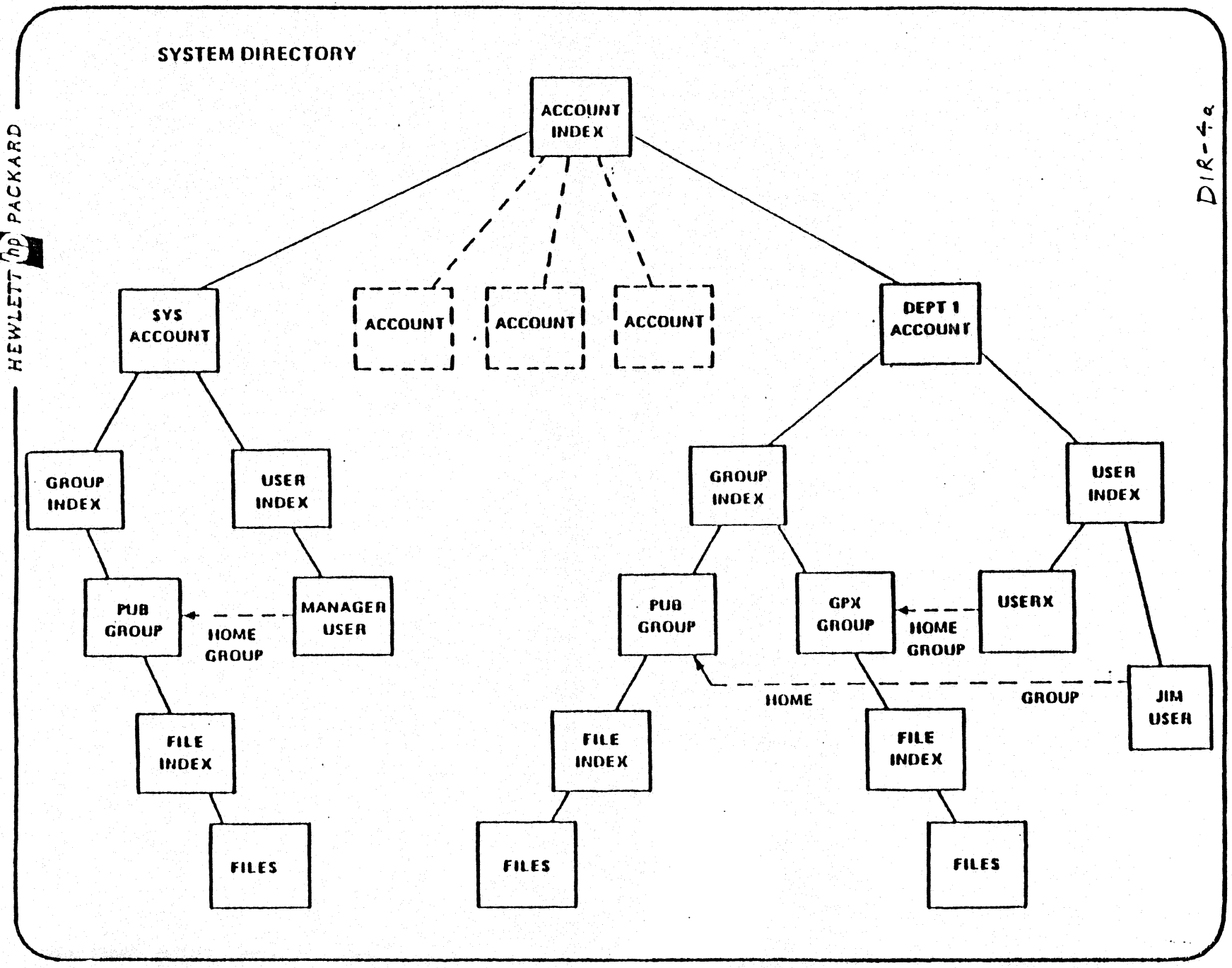


GROUP ENTRY
(41 WORDS)



FILE ENTRY
(6 WORDS)




SYSTEM DIRECTORY
HEWLETT PACKARD
DIR-4a
ACCOUNT INDEX
SYS ACCOUNT
ACCOUNT
ACCOUNT
ACCOUNT
DEPT 1 ACCOUNT
GROUP INDEX
USER INDEX
GROUP INDEX
USER INDEX
PUB GROUP
HOME GROUP
MANAGER USER
PUB GROUP
GPX GROUP
HOME GROUP
USERX
HOME
GROUP
JIM USER
FILE INDEX
FILE INDEX
FILE INDEX
FILES
FILES
FILES

DIRECTORY

ACCOUNT INDEX BLOCK

SECTOR %3

	%0	110143		INDEX BLOCK INFO
	%1	000003		# OF INDEX PTRS
INDEX	%2	000022		# OF ACCESSORS
BLOCK	%3	000034		ENTRY TOTALS
	%4	010743		ENTRY BLOCK INFO
PREFIX	%5	000000		INDEX PTR OF FATHER
	%6			
	%7			NAME OF
	%10			FATHER
	%11			
	%12	A	A	
INDEX	%13	6	4	ENTRY
BLOCK	%14			NAME
ENTRY	%15			
	%16	000010		PTR TO ENTRY BLOCK
	%17	000007		# OF ENTRIES IN BLK
	%20	G	A	
INDEX	%21	M	E	ENTRY
BLOCK	%22	S		NAME
ENTRY	%23			
	%24	001267		PTR TO ENTRY BLOCK
	%25	000011		# OF ENTRIES IN BLK
	%26	S	B	
INDEX	%27	A	Y	ENTRY
BLOCK	%30	C	O	NAME
	%31	M		
ENTRY	%32	000173		PTR TO ENTRY BLOCK
	%33	000014		# OF ENTRIES IN BLK
	%34			

ect.

ACCOUNT ENTRY BLOCK

SECTOR %10

	%0	A	A	
	%1	6	4	ACCOUNT
ACCOUNT	%2			NAME
ENTRY	%3			
	%4	000026		ACCT GRP INDEX PTR
	%5	000027		ACCT USER INDEX PTR
	%35			
	%36	A	C	
	%37	C	O	ACCOUNT
ACCOUNT	%40	U	N	NAME
	%41	T	1	
ENTRY	%42	002003		ACCT GRP INDEX PTR
	%43	002004		ACCT USER INDEX PTR
	%73			
	%74	B	E	
	%75	E		ACCOUNT
	%76			NAME
ACCOUNT	%77			
ENTRY	%100	001614		ACCT GRP INDEX PTR
	%101	001615		ACCT USER INDEX PTR

ect.

NOTE SECTOR ADDRESSES ARE DIRBASE RELATIVE

DIRECTORY

ACCOUNT ENTRY BLOCK
SECTOR %10

	%0	A	A	
	%1	6	4	ACCOUNT
ACCOUNT	%2			NAME
ENTRY	%3			
	%4	0 0 0 0 2 6		ACCT GRP INDEX PTR
	%5	0 0 0 0 2 7		ACCT USER INDEX PTR
	%35			
	%36	A	C	
ACCOUNT	%37	C	O	ACCOUNT
ENTRY	%40	U	N	NAME
	%41	T	1	
	%42	0 0 2 0 0 3		ACCT GRP INDEX PTR
	%43	0 0 2 0 0 4		ACCT USER INDEX PTR
	%73			
	%74	B	E	
ACCOUNT	%75	E		ACCOUNT
ENTRY	%76			NAME
	%77			
	%100	0 0 1 6 1 4		ACCT GRP INDEX PTR
	%101	0 0 1 6 1 5		ACCT USER INDEX PTR

ect.

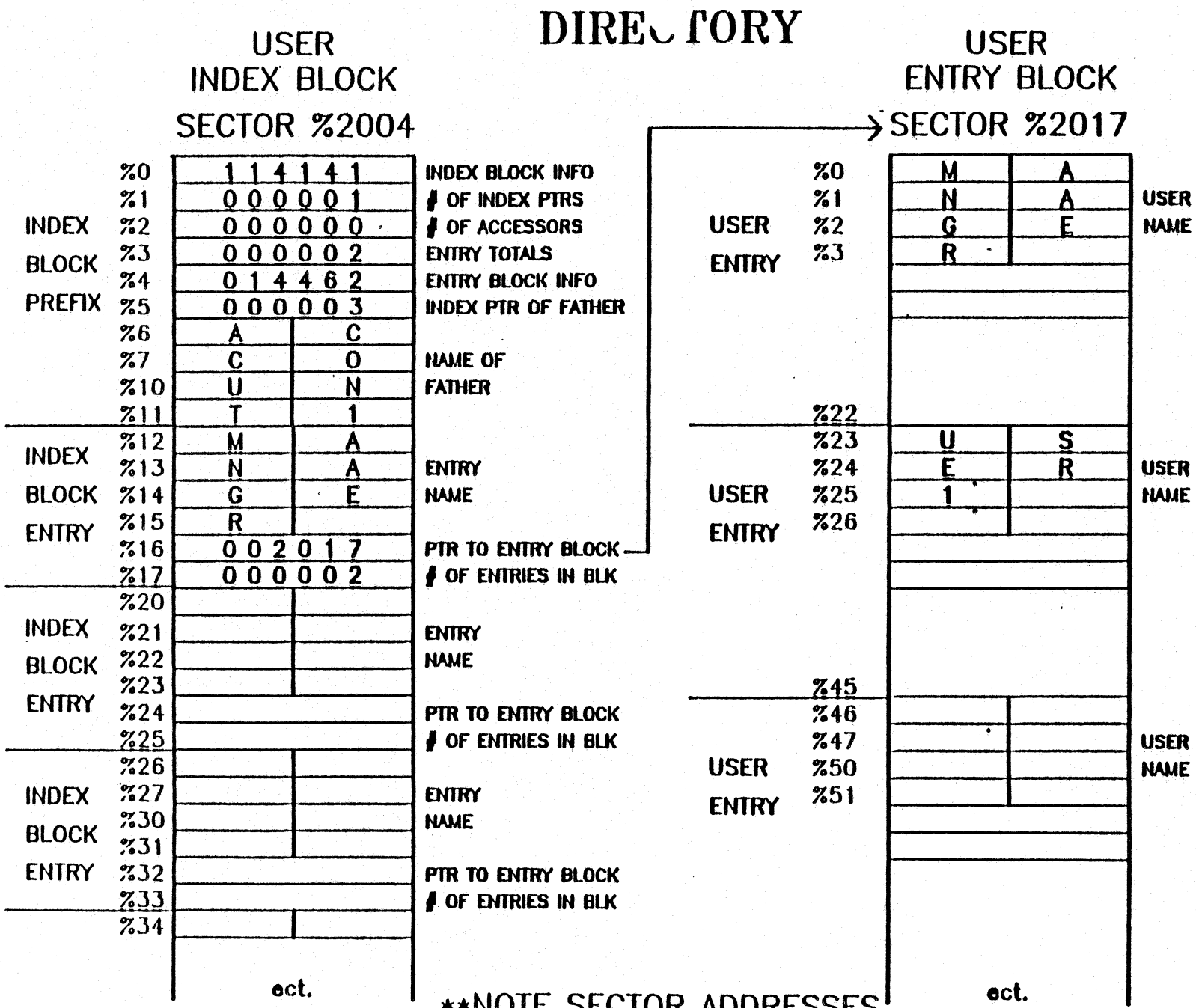
USER INDEX BLOCK
SECTOR %2004

	%0	1 1 4 1 4 1		INDEX BLOCK INFO
	%1	0 0 0 0 0 1		# OF INDEX PTRS
INDEX	%2	0 0 0 0 0 0		# OF ACCESSORS
BLOCK	%3	0 0 0 0 0 2		ENTRY TOTALS
	%4	0 1 4 4 6 2		ENTRY BLOCK INFO
PREFIX	%5	0 0 0 0 0 3		INDEX PTR OF FATHER
	%6	A	C	
	%7	C	O	NAME OF
	%10	U	N	FATHER
	%11	T	1	
	%12	M	A	
INDEX	%13	N	A	ENTRY
BLOCK	%14	G	E	NAME
ENTRY	%15	R		
	%16	0 0 2 0 1 7		PTR TO ENTRY BLOCK
	%17	0 0 0 0 0 2		# OF ENTRIES IN BLK
	%20			
INDEX	%21			ENTRY
BLOCK	%22			NAME
	%23			
ENTRY	%24			PTR TO ENTRY BLOCK
	%25			# OF ENTRIES IN BLK
	%26			
INDEX	%27			ENTRY
BLOCK	%30			NAME
	%31			
ENTRY	%32			PTR TO ENTRY BLOCK
	%33			# OF ENTRIES IN BLK
	%34			

ect.

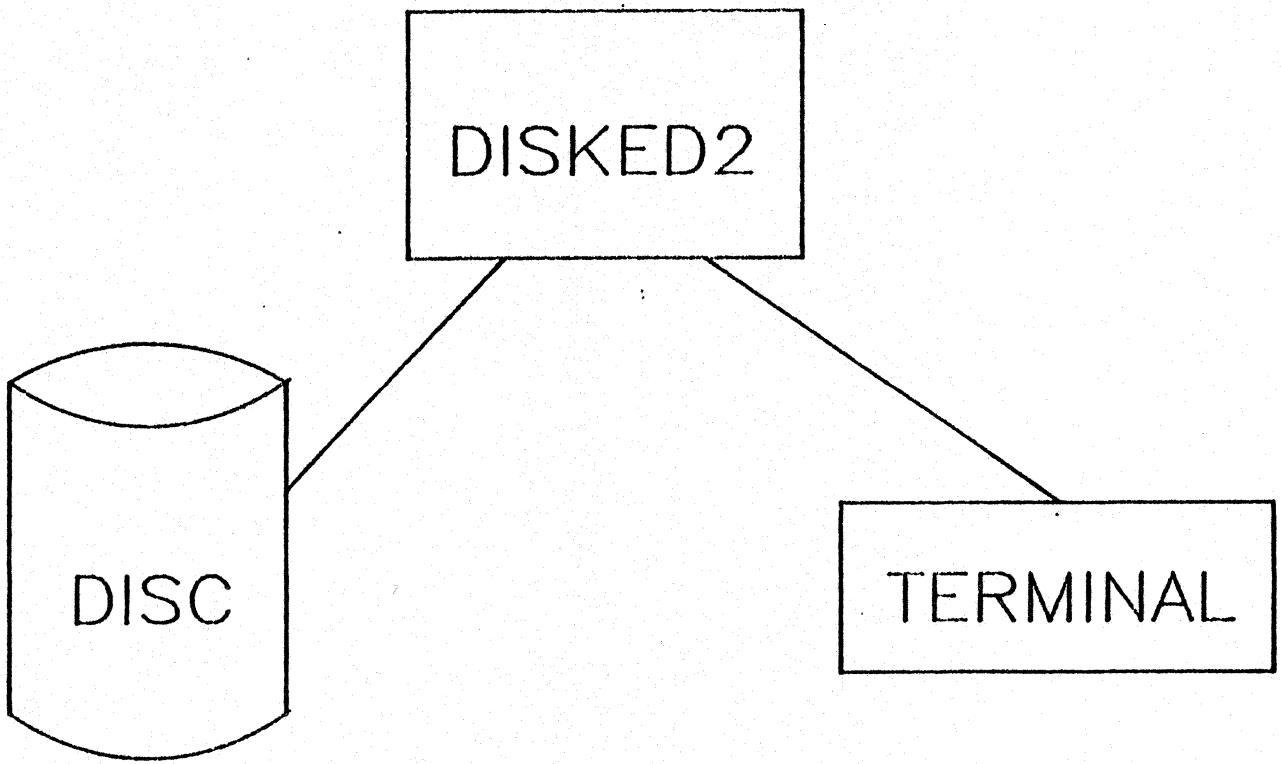
NOTE SECTOR ADDRESSES ARE DIRBASE RELATIVE

DIRECTORY



NOTE SECTOR ADDRESSES ARE DIRBASE RELATIVE

DISKED2 (DISK EDIT)



DISKED2

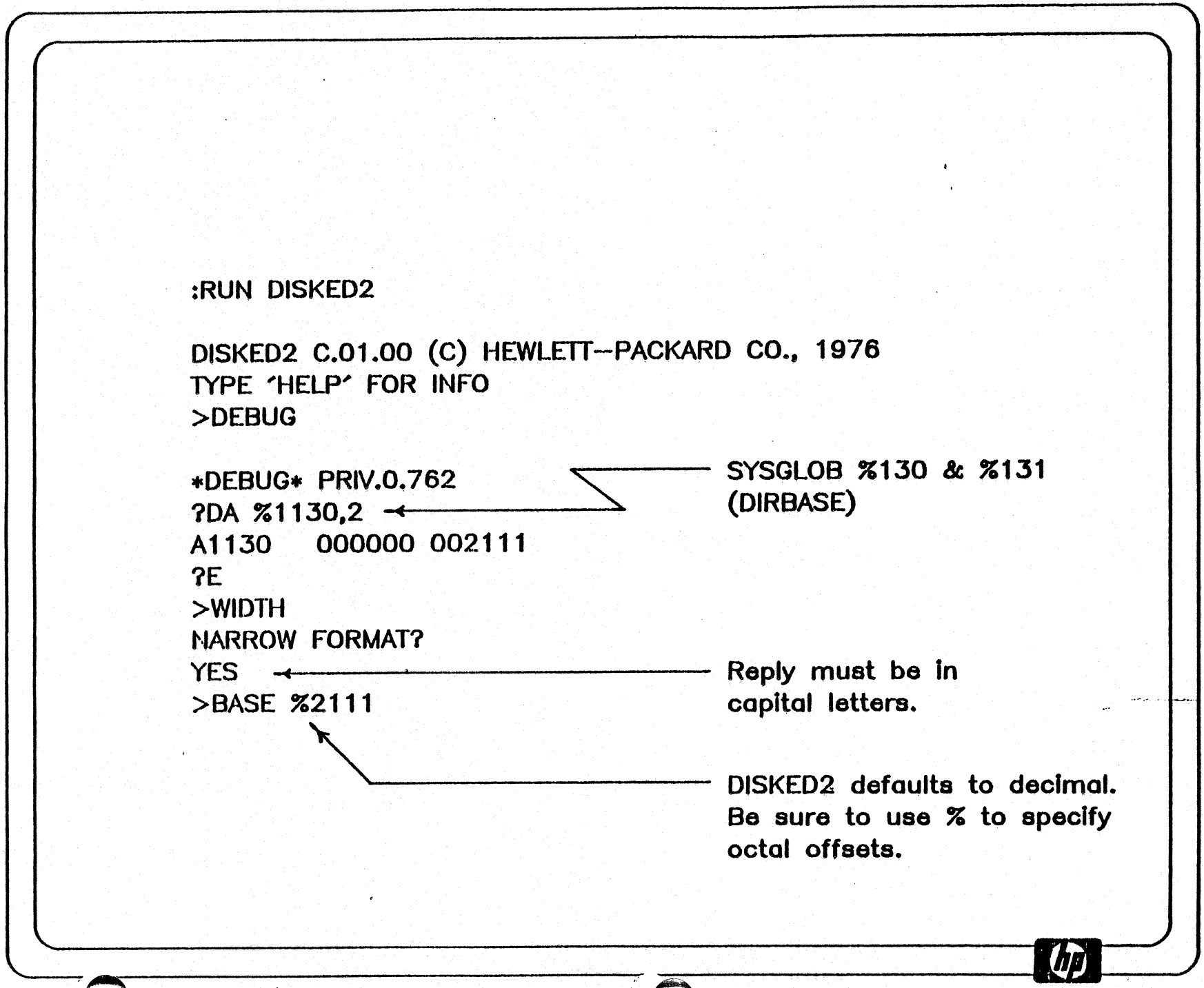
DESCRIPTION:

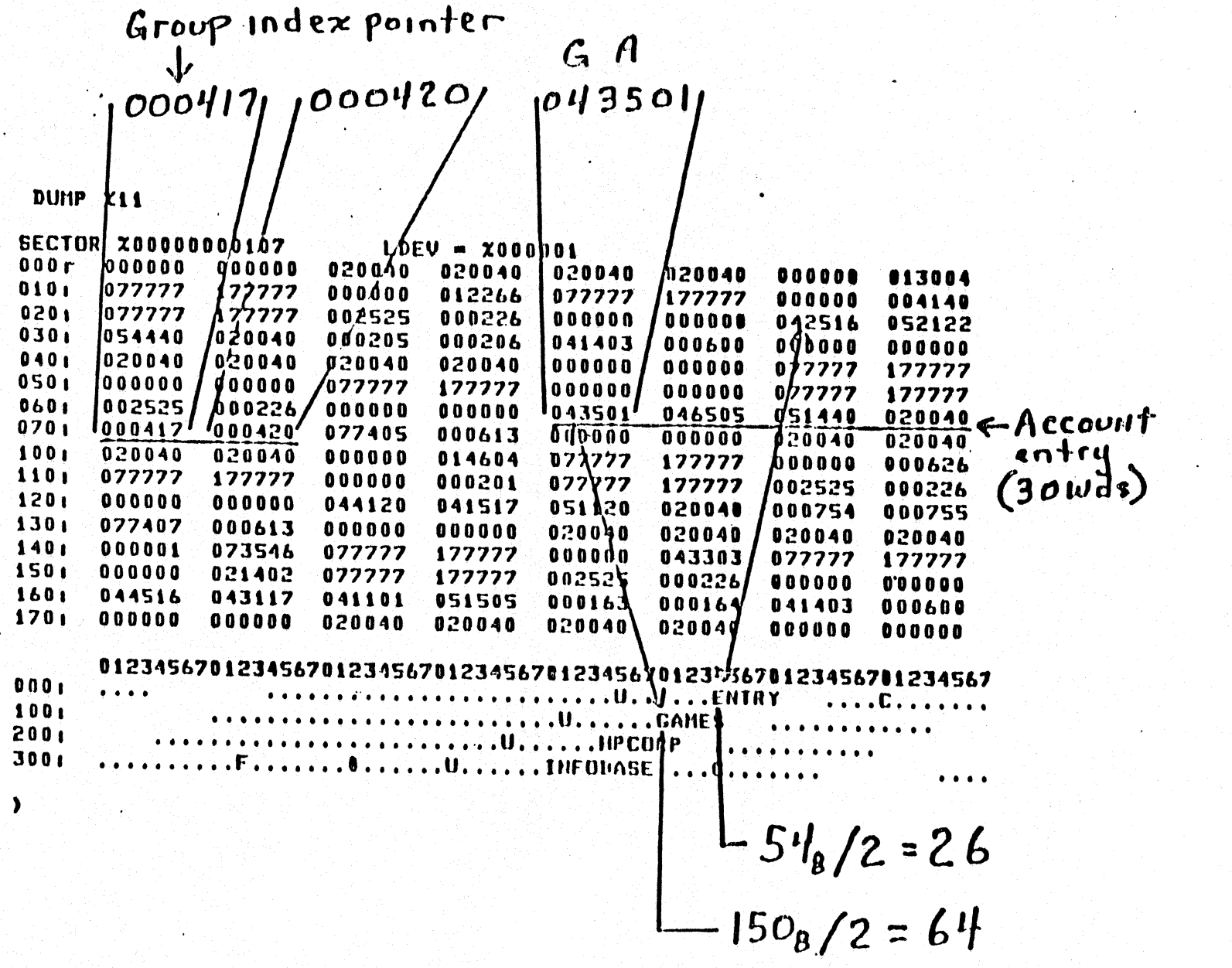
Allows you to do online display or modification of data on discs.

CONSIDERATIONS:

Modification of disc can cause system failure or loss of data.

```
:RUN DISKED2

DISKED2 C.01.00 (C) HEWLETT-PACKARD CO., 1976
TYPE 'HELP' FOR INFO
>HELP

DISKED2 allows todump and/or modify : file contents or
any disc sector (sys. mgr capability is required).

B[ASE] [<ABS SEC #>]
DEBUG
DISC <LOG DEV #>
D[UMP] [ [<REL SEC #>] [, <# OF SEC>] ] OR [<'ALL'>] [, A=ASCII ]
    (AT LEAST ONE PARAMETER MUST BE PRESENT.)
F[ILE] <FILENAME>
L[IST] [<DEVICE CLASS>] OR [<LOG DEV #>]
M[ODIFY] <SEC NUM, REL WORD ADDR [,NUM OF WORDS]>
    (NEW VALUE STARTS WITH : # - DECIMAL, ' - ASCII)
W[IDTH]
E[XIT]
```


:RUN DISKED2
DISKED2 C.01.00 (C) HEWLETT-PACKARD CO., 1976
TYPE 'HELP' FOR INFO
>DEBUG
DEBUG PRIV.0.762
?DA %1130,2
A1130 000000 002111
?E
>WIDTH
NARROW FORMAT?
YES
>BASE %2111
SYSGLOB %130 & %131 (DIRBASE)
Reply must be in capital letters.
DISKED2 defaults to decimal. Be sure to use % to specify octal offsets.


Group index pointer
000417
000420
G A
043501
DUMP %11
SECTOR %00000000107 LDEV = %000101
000: 000000 000000 020040 020040 020040 020040 000000 013004
010: 077777 177777 000000 012266 077777 177777 000000 004140
020: 077777 177777 002525 000226 000000 000000 042516 052122
030: 054440 020040 000205 000206 041403 000600 000000 000000
040: 020040 020040 020040 020040 000000 000000 077777 177777
050: 000000 000000 077777 177777 000000 000000 077777 177777
060: 002525 000226 000000 000000 043501 046505 051440 020040
070: 000417 000420 077405 000613 000000 000000 020040 020040
100: 020040 020040 000000 014604 077777 177777 000000 000626
110: 077777 177777 000000 000201 077777 177777 002525 000226
120: 000000 000000 044120 041517 051120 020040 000754 000755
130: 077407 000613 000000 000000 020040 020040 020040 020040
140: 000001 073546 077777 177777 000000 043303 077777 177777
150: 000000 021402 077777 177777 002525 000226 000000 000000
160: 044516 043117 041101 051505 000163 000164 041403 000600
170: 000000 000000 020040 020040 020040 020040 000000 000000
Account entry (30wds)
01234567012345670123456701234567012345670123456701234567012345670123456701234567
000: ....U..J...ENTRY ....C.....
100: .....U.....GAMES .....
200: .....U.....HPCORP .....
300: .....F.....0.....U.....INFOBASE ...C..... ....
)
54B/2 = 26
150B/2 = 64

000424

```
 DUMP X417

SECTOR X00000000515        LDEV = X000001
000:  104141  000001  000000  000002  005222  000003  043501  046505
010:  051440  020040  040504  053040  020040  020040  000424  000002
020:  000002  000002  000002  000002  000002  000002  000002  000002
030:  000002  000002  000002  000002  000002  000002  000002  000002
040:  000002  000002  000002  000002  000002  000002  000002  000002
050:  000002  000002  000002  000002  000002  000002  000002  000002
060:  000002  000002  000002  000002  000002  000002  000002  000002
070:  000002  000002  000002  000002  000002  000002  000002  000002
100:  000002  000002  000002  000002  000002  000002  000002  000002
110:  000002  000002  000002  000002  000002  000002  000002  000002
120:  000002  000002  000002  000002  000002  000002  000002  000002
130:  000002  000002  000002  000002  000002  000002  000002  000002
140:  000002  000002  000002  000002  000002  000002  000002  000002
150:  000002  000002  000002  000002  000002  000002  000002  000002
160:  000002  000002  000002  000002  000002  000002  000002  000002
170:  000002  000002  000002  000002  000002  000002  000002  000002

      0123456701234567012345670123456701234567012345670123456701234567
000:  ..........CAMES   ADV     ................................
100:  ................................................................
200:  ................................................................
300:  ................................................................
```

← Index block entry (group) (Lwde)

Base = 76
+ 3
101

\# of blocks ↓ 000002

\# of accts ↓ 000024

001634

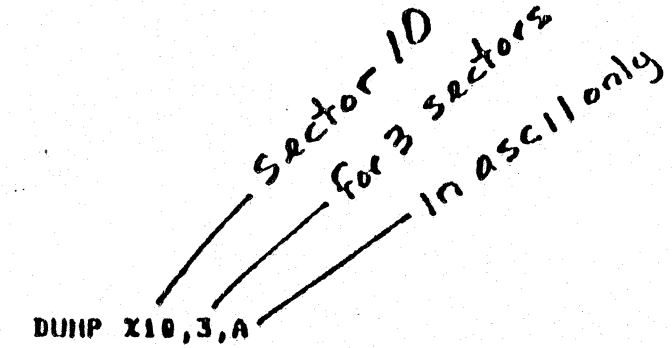
000010

```
)DUMP 3

SECTOR X00000000101          LDEV = X000001
000|  110143  000002  000007  000024  110743  000000  020040  020040
010|  020040  020040  041105  047103  044040  020040  000010  000012
020|  047117  051124  044102  040531  001634  000012  000012  000012
030|  000012  000012  000012  000012  000012  000012  000012  000012
040|  000012  000012  000012  000012  000012  000012  000012  000012
050|  000012  000012  000012  000012  000012  000012  000012  000012
060|  000012  000012  000012  000012  000012  000012  000012  000012
070|  000012  000012  000012  000012  000012  000012  000012  000012
100|  000012  000012  000012  000012  000012  000012  000012  000012
110|  000012  000012  000012  000012  000012  000012  000012  000012
120|  000012  000012  000012  000012  000012  000012  000012  000012
130|  000012  000012  000012  000012  000012  000012  000012  000012
140|  000012  000012  000012  000012  000012  000012  000012  000012
150|  000012  000012  000012  000012  000012  000012  000012  000012
160|  000012  000012  000012  000012  000012  000012  000012  000012
170|  000012  000012  000012  000012  000012  000012  000012  000012

      0123456701234567012345670123456701234567012345670123456701234567
000|  ............        BENCH    ....NORTHDAY.......................
100|  ................................................................
200|  ................................................................
300|  ................................................................

)
```

Index block Prefix (10 wds)

Index block entry (6 wds)

$24_8 / 2 = 12_8$




```
DUMP X10,3,A

SECTOR X00000000106       LDEV = X000001
000:  BENCH   ............          ..........................U......COCO
100:       ....C.......          ..........................U......CONVAID
200:  ....C.......          ..........................U......DREEACTG....
300:  C.......          ..........................U......EASTBAY ........

SECTOR X00000000107       LDEV = X000001
000:  ....          ..........................U......ENTRY    ....C.......
100:           ..........................U......GAMES  ............
200:       ..........................U......HPCORP  ............
300:  ..........F.......0......U......INFOBASE....C.......           ....

SECTOR X00000000110       LDEV = X000001
000:  ......................U......LIB      ....C.......          ........
100:  ...2.......1.....U.....................................................
200:  .......................................................................
300:  .......................................................................
```

000421

```
)DUMP X424

SECTOR X00000000522       LDEV = X000001
000:  040504  053040  020040  020040  000706  020040  020040  020040
010:  020040  000000  002102  077777  177777  000000  000552  077777
020:  177777  000000  000107  077777  177777  002041  004102  000613
030:  000000  000724  020040  020040  020040  020040  020040  020040
040:  020040  020040  020040  020040  020040  020040  000000  000000
050:  000000  050125  041040  020040  020040  000421  020040  020040
060:  020040  020040  000000  012502  077777  177777  000000  000054
070:  077777  177777  000000  000072  077777  177777  020143  015006
100:  000713  000000  000423  020040  020040  020040  020040  020040
110:  020040  020040  020040  020040  020040  020040  020040  000000
120:  000000  000000  000000  000000  000000  000000  000000  000000
130:  000000  000000  000000  000000  000000  000000  000000  000000
140:  000000  000000  000000  000000  000000  000000  000000  000000
150:  000000  000000  000000  000000  000000  000000  000000  000000
160:  000000  000000  000000  000000  000000  000000  000000  000000
170:  000000  000000  000000  000000  000000  000000  000000  000000

      0123456701234567012345670123456701234567012345670123456701234567
000:  ADV     ..        ...B..............G.....I.B......
100:              ......PUD     ..        ...B.....,.....I.... ...
200:  ......                    ..................................
300:  ..........................................................
```

Group entry (11 wds)

$122_8/2 = 51$

000514

```
DUMP X421

SECTOR X000000000517        LDEV = X000001
000:  100142  000005  000000  000223  000142  000417  050125  041040
010:  020040  020040  040502  040507  042514  020040  000514  000040
020:  041522  040520  051440  020040  000516  000040  046501  051111
030:  042524  020040  000520  000040  047125  041117  040524  020040
040:  000522  000025  051122  042503  047522  042040  000524  000036
050:  000036  000036  000036  000036  000036  000036  000036  000036
060:  000036  000036  000036  000036  000036  000036  000036  000036
070:  000036  000036  000036  000036  000036  000036  000036  000036
100:  000036  000036  000036  000036  000036  000036  000036  000036
110:  000036  000036  000036  000036  000036  000036  000036  000036
120:  000036  000036  000036  000036  000036  000036  000036  000036
130:  000036  000036  000036  000036  000036  000036  000036  000036
140:  000036  000036  000036  000036  000036  000036  000036  000036
150:  000036  000036  000036  000036  000036  000036  000036  000036
160:  000036  000036  000036  000036  000036  000036  000036  000036
170:  000036  000036  000036  000036  000036  000036  000036  000036

      0123456701234567012345670123456701234567012345670123456701234567
000:  ............PUB    ABACEL  .L. CRAPS   .N. MARKET  .P. NUBOAT
100:  .R..RRECORD .T.....................................................
200:  ..................................................................
300:  ..................................................................
```

Index block entry (files) (6 wds)

001001 122501

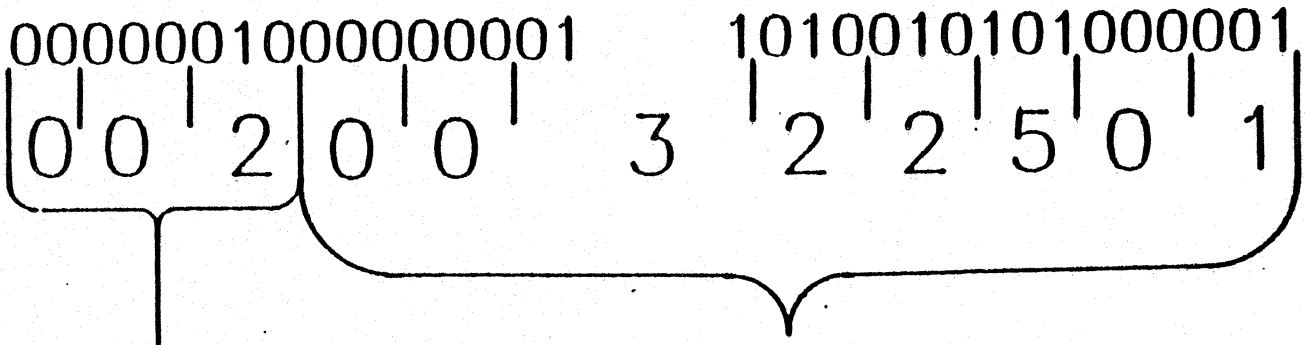
Vol LDEV

Volume Table Index

```
)DUMP %514

SECTOR %00000000612        LDEV = %000001
000:  040502  040507  042514  020040  000402  100210  040502  040523
010:  042440  020040  001001  121650  040504  053103  047515  020040
020:  000402  100225  040504  053105  047124  020040  001001  121666
030:  040504  053123  040526  042440  000402  100625  040515  040532
040:  042440  020040  001001  122461  040515  047522  052040  020040
050:  000402  101225  040516  044515  040514  020040  001001  122501
060:  040516  044515  040514  052040  000402  101257  040516  044515
070:  046061  020040  001001  122530  040516  044515  046062  020040
100:  000402  101351  041101  043505  046123  020040  001001  122704
110:  041101  052116  052515  020040  000402  101461  041105  040516
120:  051440  020040  001001  122720  041111  047523  044516  020040
130:  000402  101501  041111  051104  040531  020040  001001  122732
140:  041112  020040  020040  020040  000402  101533  041114  040503
150:  045512  040513  001001  122745  041114  045101  041513  020040
160:  000402  101557  041125  052124  047516  020040  001001  123003
170:  041501  047116  047516  020040  000402  101614  041501  053105

      0123456701234567012345670123456701234567012345670123456701234567
000:  ABAGEL  ....ABASE   ....ADVCON  ....ADVENT  ....ADVSAVE ....AMAZ
100:  E   ...1AMORT   ....ANIMAL  ...AANIMALT ....ANIML1  ...XANIML2
200:  ....BAGELS  ....BATNUM  ...1BEANS   ....BIOSIN  ...BIRDAY  ....
300:  BJ      ...BLACKJAK....BLJACK  ....BUTTON  ....CANNON  ....CAVE

)
```

File entry (6 wds)

$124_8/2 =$



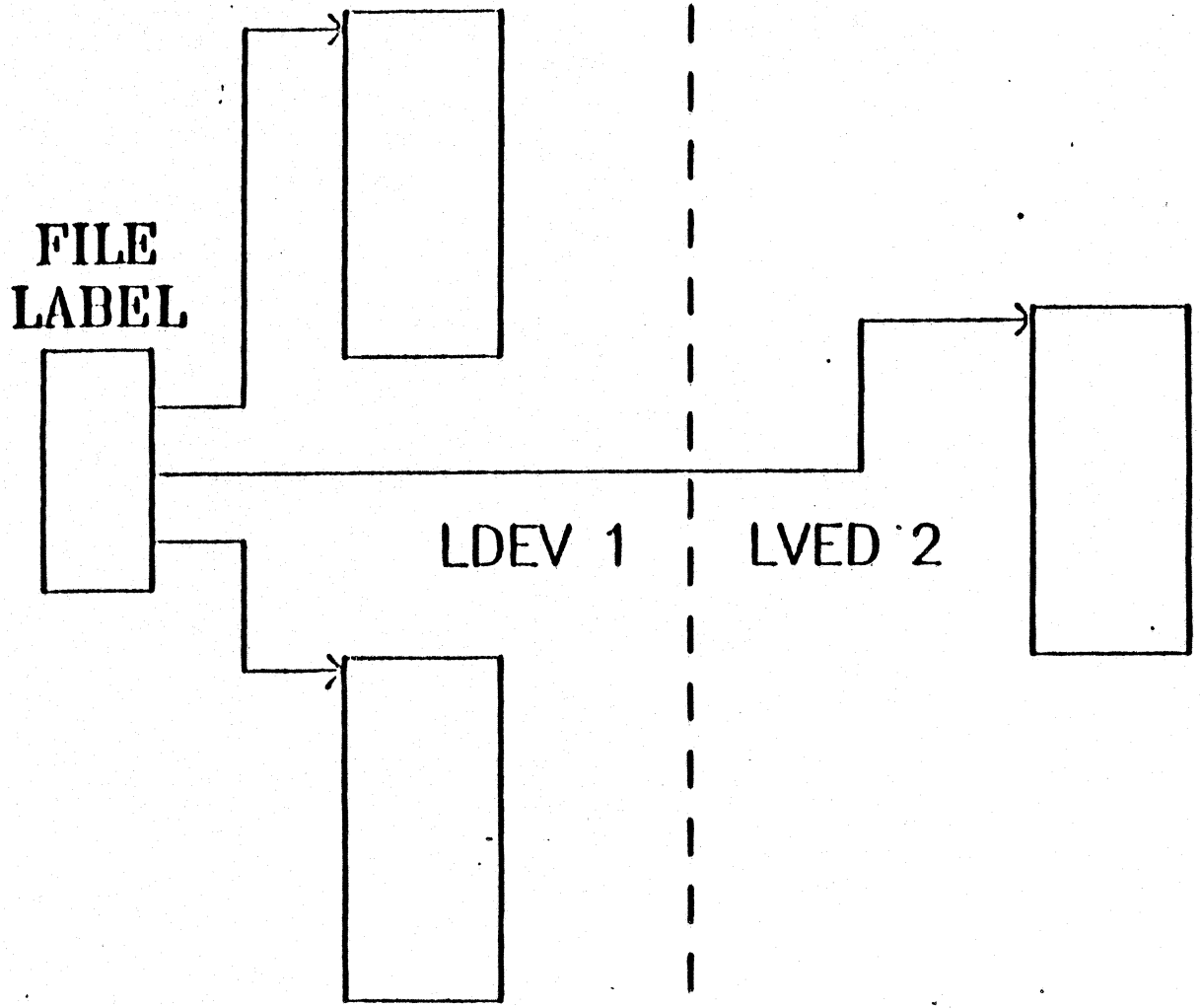
File label address

Volume table index "NOT LDEV #!!!"

File Label

```
DISC 2          <- LDEV
)BASE 8
)DUMP X322501

SECTOR X00000322501        LDEV = X000002
000: 040516 044515 040514 020040 050125 041040 020040 020040
010: 043501 046505 051440 020040 050114 040531 042522 020040
020: 020040 020040 020040 020040 020202 004040 000001 117017
030: 117073 117073 002005 000000 004400 000000 000000 000026
040: 000000 000000 017655 017512 002001 177400 000200 000400
050: 000027 000027 000000 000026 000001 122501 000000 000000
060: 000000 000000 000000 000000 000000 000000 000000 000000
070: 000000 000000 000000 000000 000000 000000 000000 000000
100: 000000 000000 000000 000000 000000 000000 000000 000000
110: 000000 000000 000000 000000 000000 000000 000000 000000
120: 000000 000000 000000 000000 000000 000000 000000 000000
130: 000000 000000 000000 000000 000000 000000 000000 000000
140: 000000 000000 000000 000000 000000 000000 000000 000000
150: 000000 000000 000000 000000 000000 000000 000000 000000
160: 000000 000000 000000 000000 000000 000000 000000 000000
170: 000000 000000 000000 000000 042111 051503 020000 047111

     0123456701234567012345670123456701234567012345670123456701234567
000: ANIMAL  PUB     GAMES   PLAYER          .. .....|.|............
100: ..A.....J.A........A.......A.A......................................
200: ...........................................................
300: ......................................................DISC  .NI
```

- 1st extent ptr. Always pts to File label itself.
- original date
- # of extents -1
- last extent
- 1?? words total, 120₁₀
- File, Group, Acct, Creator
- Device Name? Could be LDEV#

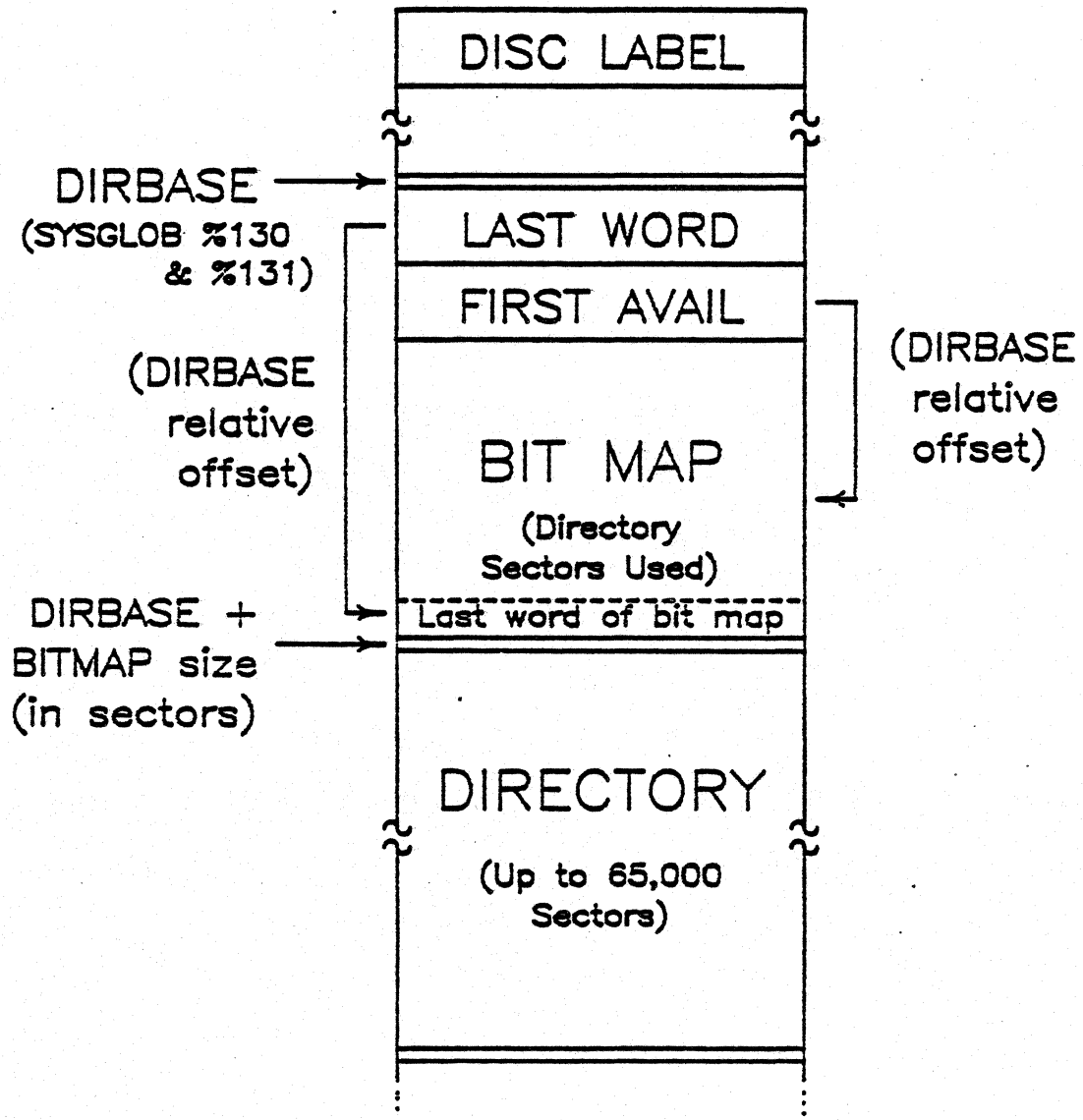


FILE EXTENTS

LISTF ,-1

```
:LISTF BASIC,-1
F  =  BASIC
041101 051511 041440 020040 000400 113173                    BASIC.......
041101 051511 041440 020040 050125 041040 020040 020040 BASIC...PUB....
051531 051440 020040 020040 046501 047101 043505 051040 SYS.....MANAGER
020040 020040 020040 020040 020202 004040 000001 117054 ..............
117074 117073 002005 000000 004000 000000 000000 000542 ..............I
000000 000000 040743 017514 002001 177400 000200 000400 ....A..L.......
000543 000543 000000 000542 000400 113173 000000 000000 .C.C...b.......
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 000000 000000 000000 000000 ..............
000000 000000 000000 000000 042111 051503 020040 020040 ........DISC...
```

DISC DIRECTORY LAYOUT



LAB

WALK THROUGH THE DIRECTORY

FINDING ALL OF THE POINTERS

TO THE FILE _____.

REFERENCE

MATERIAL

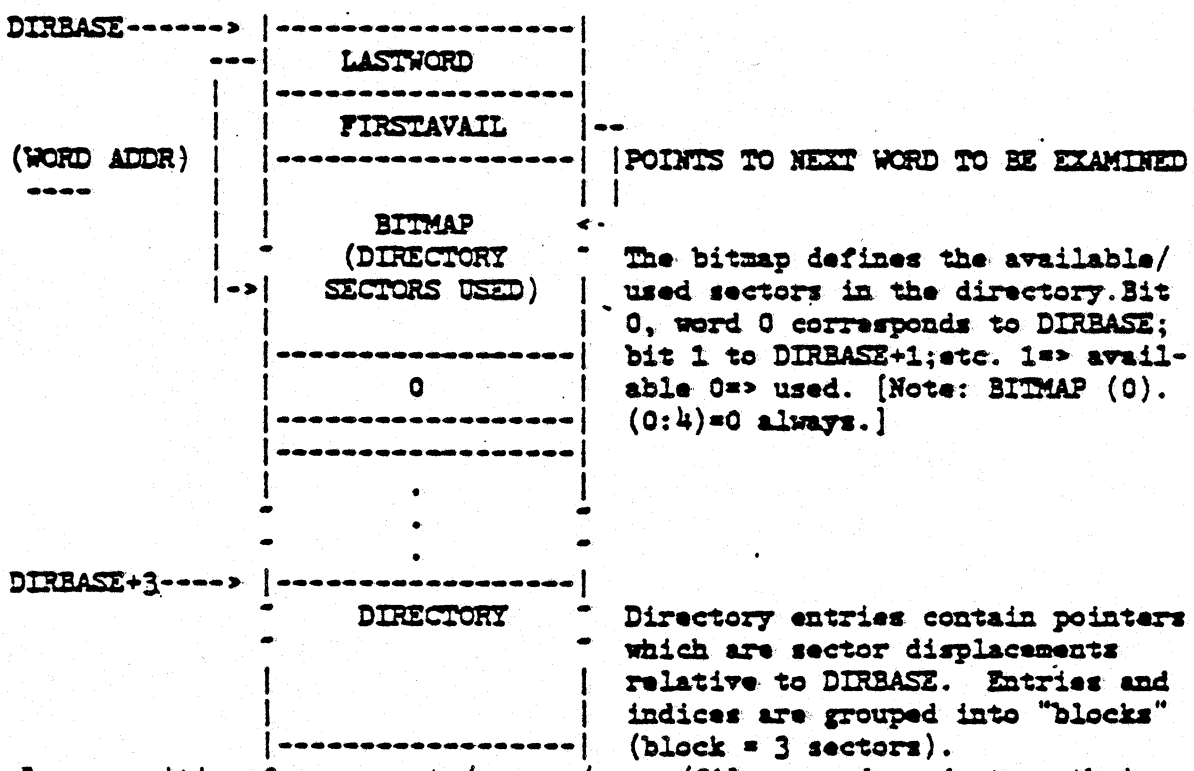
CHAPTER 4 DIRECTORY

DIRECTORY

Directory on disc consists of a contiguous area

SYSGLOB cells:

DIRBASE<---------absolute disk addr of base [SYSGLOB+%130 AND %131]



The capacities for accounts/groups/users/files are dependent on their block sizes, described in the directory data segment.

SYSSAIBSIZE	System acct index block size (sectors)
SYSAUIBSIZE	Acct. user index block size (sectors)
SYSAGIBSIZE	Acct. group index block size (sectors)
SYSGFIBSIZE	Group file index block size (sectors)
SYSGVSIBSIZE	Group volume set definition ind. blk. size(sectors)
SYSAEBSIZE	Acct. entry block size (sectors)
SYSUEBSIZE	User entry block size (sectors)
SYSGEBSIZE	Group entry block size (sectors)
SYSFEBSIZE	File entry block size (sectors)
SYSMAXBSIZE	Maximum of above. (used to initialize DDS.)
SYSVSEBSIZE	Volume set definition entry block size (sectors)

These values are used once for the creation of the (root) system, account index or new systems. This root index is always at address DIRBASE+3.

D I R E C T O R Y I M S

Larry Birenbaum
R.J. Vannucci
S. Yamakoshi

September 2, 1981

4. DIRECTORY

4.1 INTRODUCTION

4.1.1 REQUIREMENTS

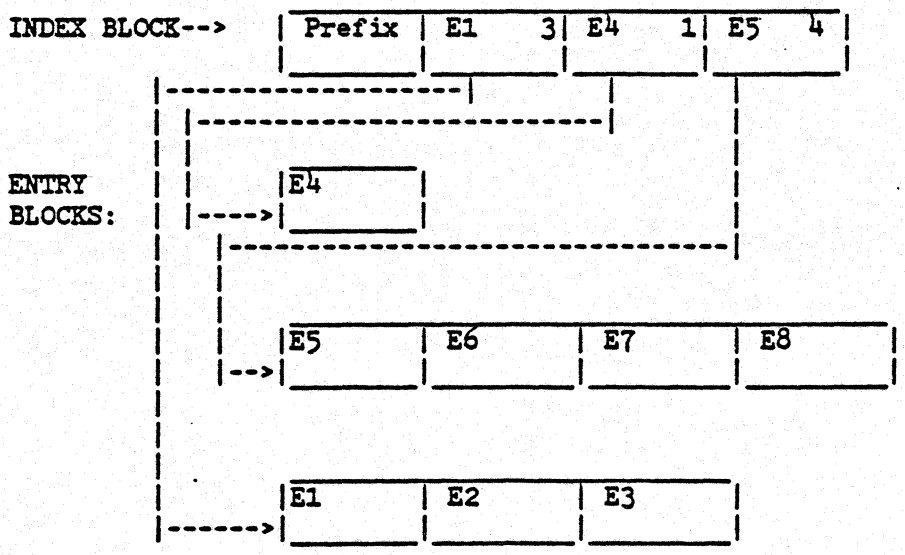
The logical requirements for a file directory and organizational directory were evolved concurrently. Accounts were to be the unique partition; belonging to each account would be a set of users and a set of groups; and every file and every volume set definition belonged to a group. The user possessed no files directly, but rather he accessed files of groups. The connection of logon groups and users served to give the user a local file domain, and also provided an entity to which resources were accumulated and limited.

The directory can be thought of as consisting of 2 trees, one overlayed upon the other. One tree is the account/user tree, the other is the account/group/file-volume set definition tree. This causes the maintenance of the MPE directory to be somewhat non-standard from the usual tree manipulations (e.g., tree scanning). In some maintenance routines, it is necessary to distinguish which subtree under an account is the desired target.

4.2 OVERVIEW

4.2.1 DATA STRUCTURE

The MPE directory is a hierarchial data structure in the form of two trees, each making use of a common first level (account), see Figure 4.2.1-A. Descending down the structure, it can be described as consisting of a system root and its subtrees of account nodes. Each account node consists of the account entry and subtrees of user nodes and subtrees of group nodes. The user nodes are "leafs". The group nodes consist of the group entry and subtrees of volume set definition entries and of file entries. Each file entry is a pointer to the file label and is a leaf as far as the directory is concerned. In the program listings, the term "level" is sometimes used in place of the more correct term, "subtree". Level, when used according to its proper definition, is always considered to be relative to the system root. Each subtree is implemented by an indexed sequential structure in which every block of nodes (entries) is pointed to by an index entry, the set of indices for a subtree being contained in one block. Pictorially, the structure for each subtree looks like:



A detailed diagram of the entire organization can be seen in Figure 4.2.1-A, 4.2.1-B, and 4.2.1-C.

The actual information that the directory must keep is contained in entries. Entries are distributed among a set of entry blocks for each subtree. The entries are maintained alphabetically within each block. Every block is pointed to by an index, which contains the name of the first entry of "its" block and a pointer to the block. The indices are kept alphabetically in an index block, of which there is one for each subtree. The entry blocks are thus logically arranged alphabetically (according to the indices) even though the actual blocks are in fact not contiguous. [Every index block is preceded by a prefix containing necessary access information; it is described below.] Note that a logical subtree is represented by an index block.

As might be inferred, space is allocated/deallocated in blocks, each consisting of an integral number of contiguous sectors. Blocks that are logically contiguous, however, need not be physically contiguous. The total space available for a directory consists of one contiguous area on the master volume of a volume set in which all space is managed by the directory routines by means of a simple space bitmap. This bitmap is the first three sectors of the directory disc area.

Note that any entry of the directory can be retrieved by either performing a global search starting from the root node, or by performing a shorter search on a subtree, assuming there exists a means of "pointing" to the desired subtree. Because every subtree is equivalent to an index block, pointers to subtrees are pointers to index blocks.

The provisions necessary for listing and quick access (ie. maintenance of pointers) for MPE directories consist of pointers to the appropriate index block. This is done via the JIT (Job Information Table). Shorter access for local group files is achieved by keeping a pointer to the user's logon group/file index block. Pointers for the purpose of listing are pointers to the relevant index block and the name being listed. The only problem in using "pointers" is that it must be guaranteed that the item which is being pointed to cannot be moved (including deleted). Therefore, the following mechanism is employed: Only pointers to index blocks can be considered safe; this protection mechanism is achieved by means of incrementing an "index pointer count" associated with each index block whenever that block is being pointed to. As long as the count is non-zero, the index block cannot be moved, or deleted. Note that pointers to actual indices, or to entries, or to entry blcoks cannot be considered secure because (without a similar facility) indices, entries, and entry blocks can be moved or deleted when the directory is not locked. Although an index block may be "frozen", its indices are not and may be moved and deleted.

Figure 4.2.1-A OVERVIEW OF DIRECTORY

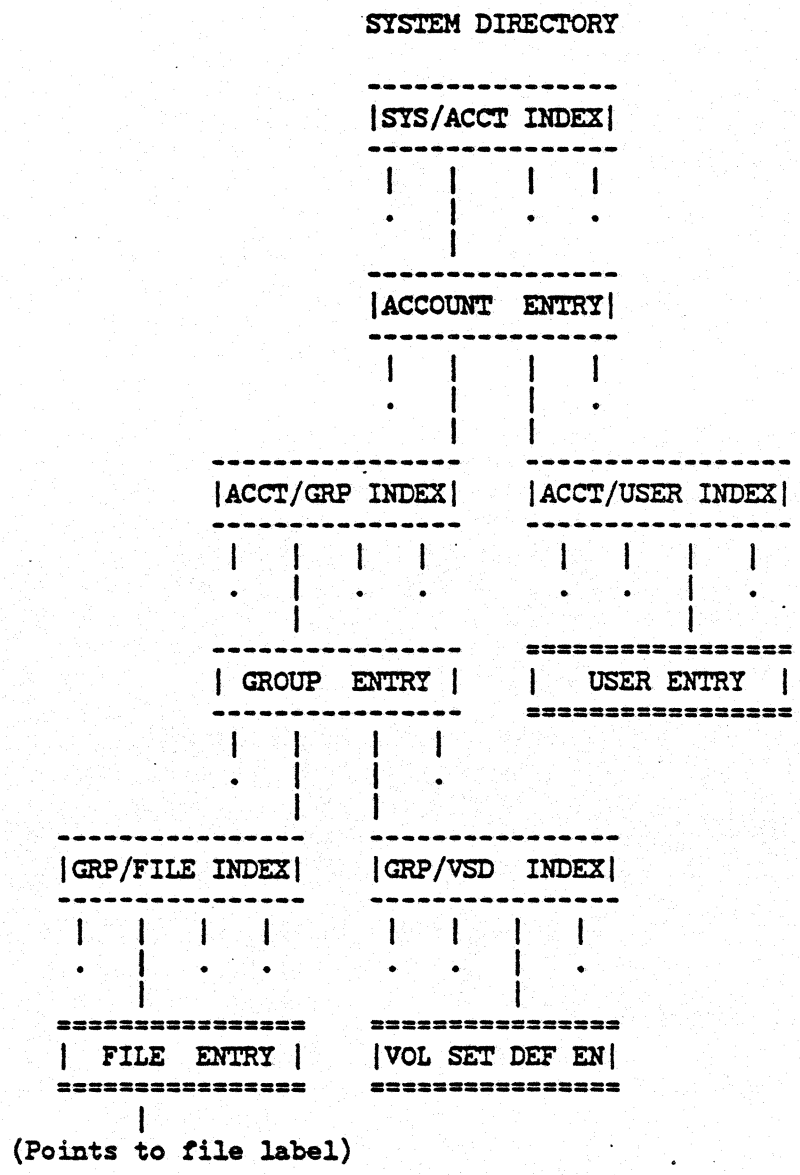
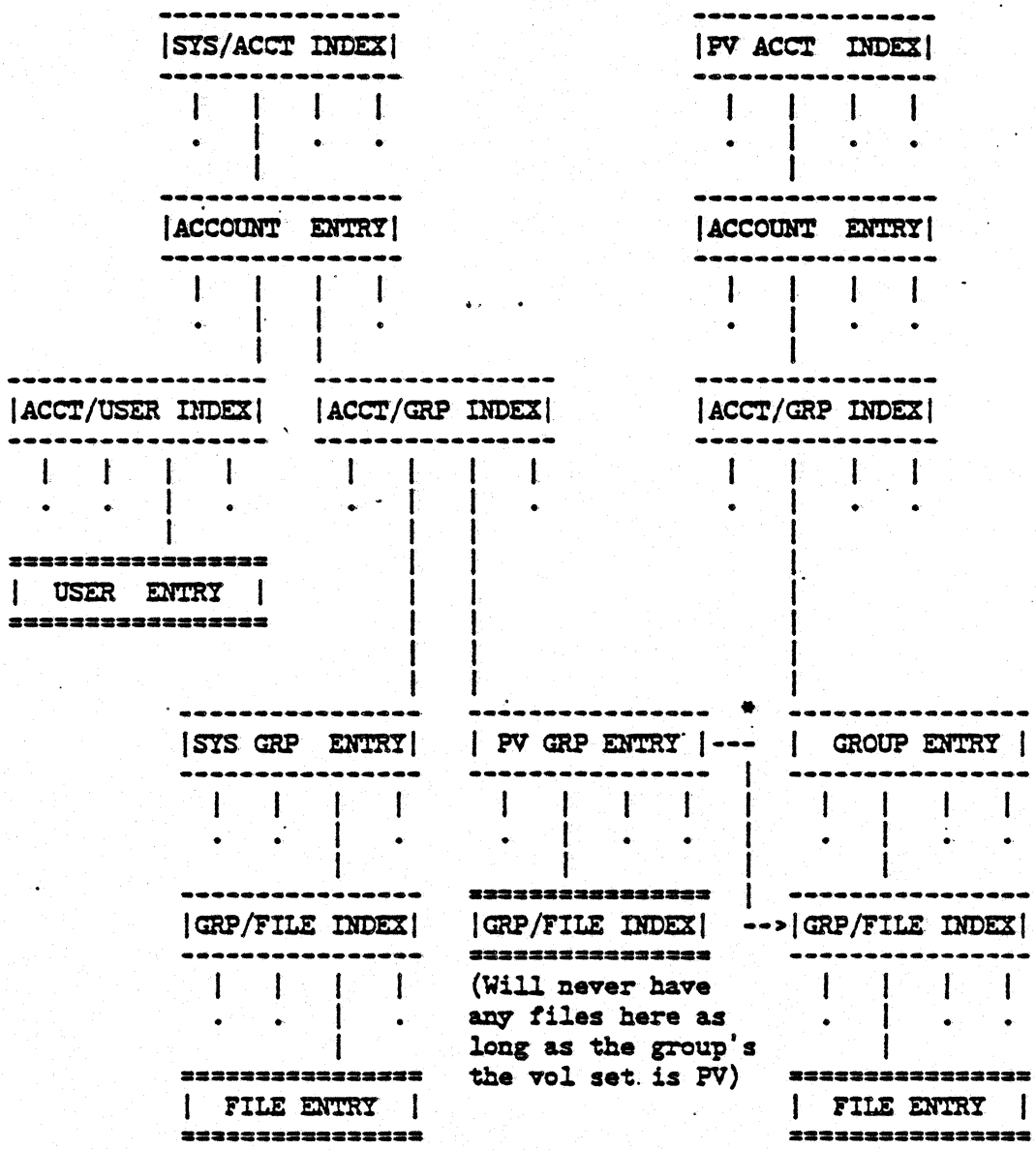
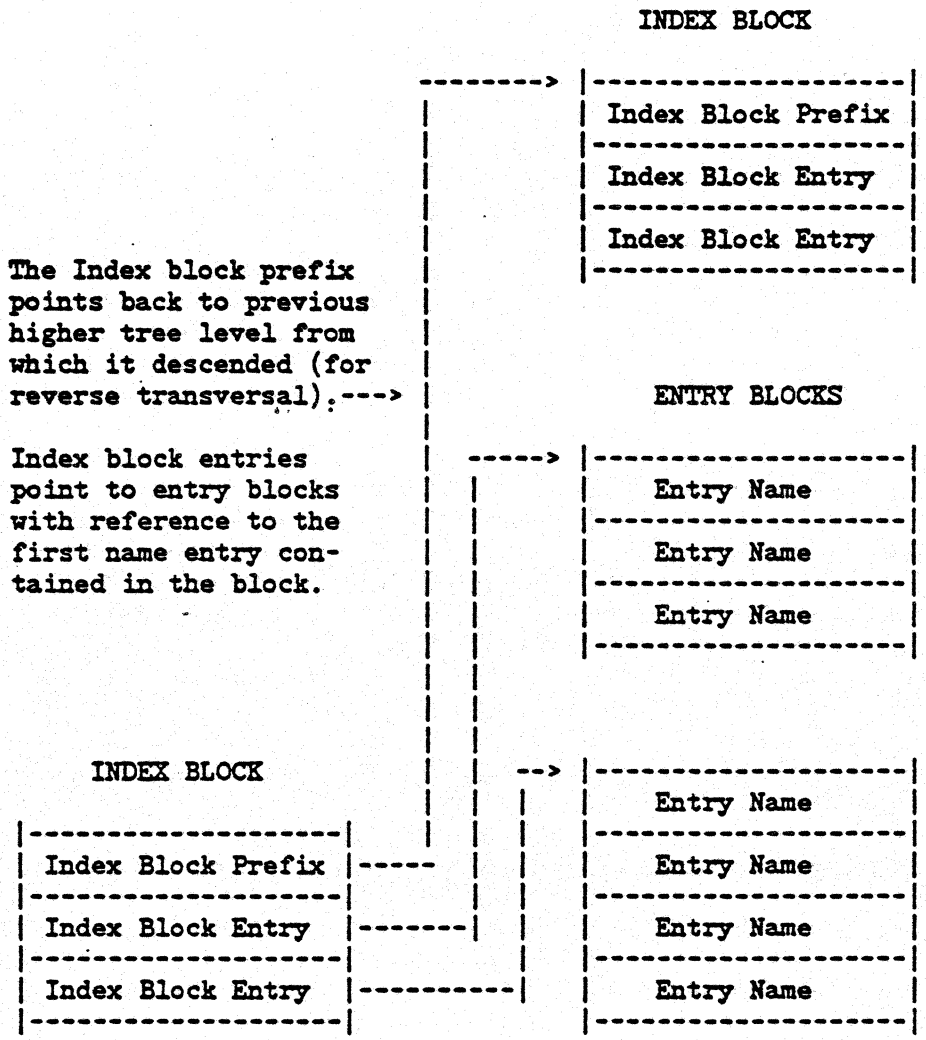


Figure 4.2.1-B SYSTEM DIRECTORY WITH PRIVATE VOLUMES



* The SYSTEM group entry will point to the Private Volume Group/File index if the group is a Private Volume Group and the Volume Set/Class is logically Mounted and Bound.

Figure 4.2.1-C DIRECTORY POINTER STRUCTURE



4.2.2 DIRECTORY INFORMATION

The directory serves essentially two independent purposes:

1. File partitioning and security; and
2. MPE access, capability restrictions, resource accumulation, and for maintaining group file space definitions (i.e., volume set definitions).

Consequently, each directory node has information pertaining to these requirements. This information is diagrammed in Figures 4.2.2-A, B, C, D, E, F, G, and H. Some notes on the contents of the directory entries follows:

NAMES - All names in the directory (as in MPE, in general), are represented internally by four continuous words containing the name in ASCII, left-justified and padded with blanks.

INDEX POINTERS - These are one-word positive integers that are directory-base relative addresses of subtrees. Thus account entries have two INDEX POINTERS, one for the group subtree and one for the user subtree. Logically, these pointers can be thought of as representing subtrees and the actual interpretation of them will be deferred.

PASSWORDS - These are used for establishing MPE logon access only. The user and account passwords are always required, if non-null; and the group password is required when accessing a non-home group. A "null" password is represented by all blanks.

CAPABILITIES and LOCAL ATTRIBUTES - The bits of account and user capabilities correspond one-for-one to the bits returned by the WHO intrinsic when capability is returned, see Figure 4.2.2-F, Attributes/Capabilities. Groups' capabilities are only the so-called, "capability class attributes", which is the second word of the full two-word capbility.

COUNTS and LIMITS - these are double-word, positive integers, each representing the indicated usage count or limit. The maintenance of these variables is described later. Where an "unlimited" quantity is required it is represented by %17777777777, permitting no special test to be necessary for the unlimited case.

FILE SECURITIES - These words contain bit representations of file security "permission matrices". The bit correspondence is diagrammed. File security is described in Subsection 4.5.2.

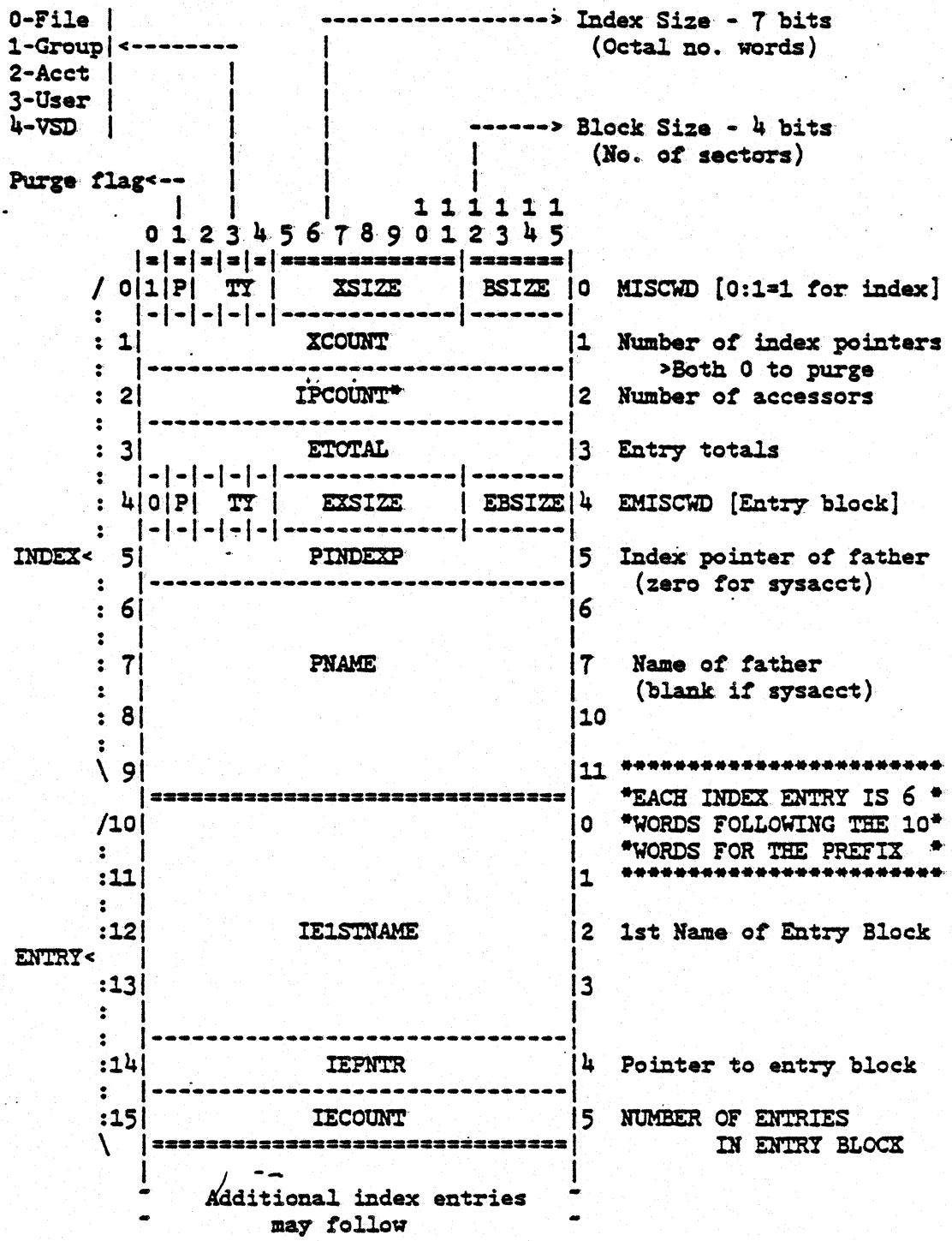
MAXIMUM PRIORITY - This is the numerical priority value which most significant factor to be considered. Given the hierarchial organization, it was desirable, of course, to be able for the user to make relatively fast accesses to his local (logon group) file domain, and to his account tile domain. Essentially it was determined that a means for "pointer" directly to a group/file level, and to an account/group level was desirable.

This means for achieving this tied in closely to another typical file directory problem: In performing a listing operation it is desirable to keep a "pointer" to the entry being listed, so that the next entry can be easily retrieved. The essential problem of keeping such a pointer is that - while depending on the pointer - some directory operation can occur asynchronously that might shift some directory entries (e.g. purge, insert), thus making the "pointer" invalid unless some provision is made explicitly for this case.

Additionally, there is the requirement that the directory structure on the disc maintain its integrity across abrupt system failures (crashes, non =SHUTDOWN shutdowns, etc.). For this reason, directory vulnerability is minimized, with return from directory routines signifying a completely recoverable directory structure on disc.

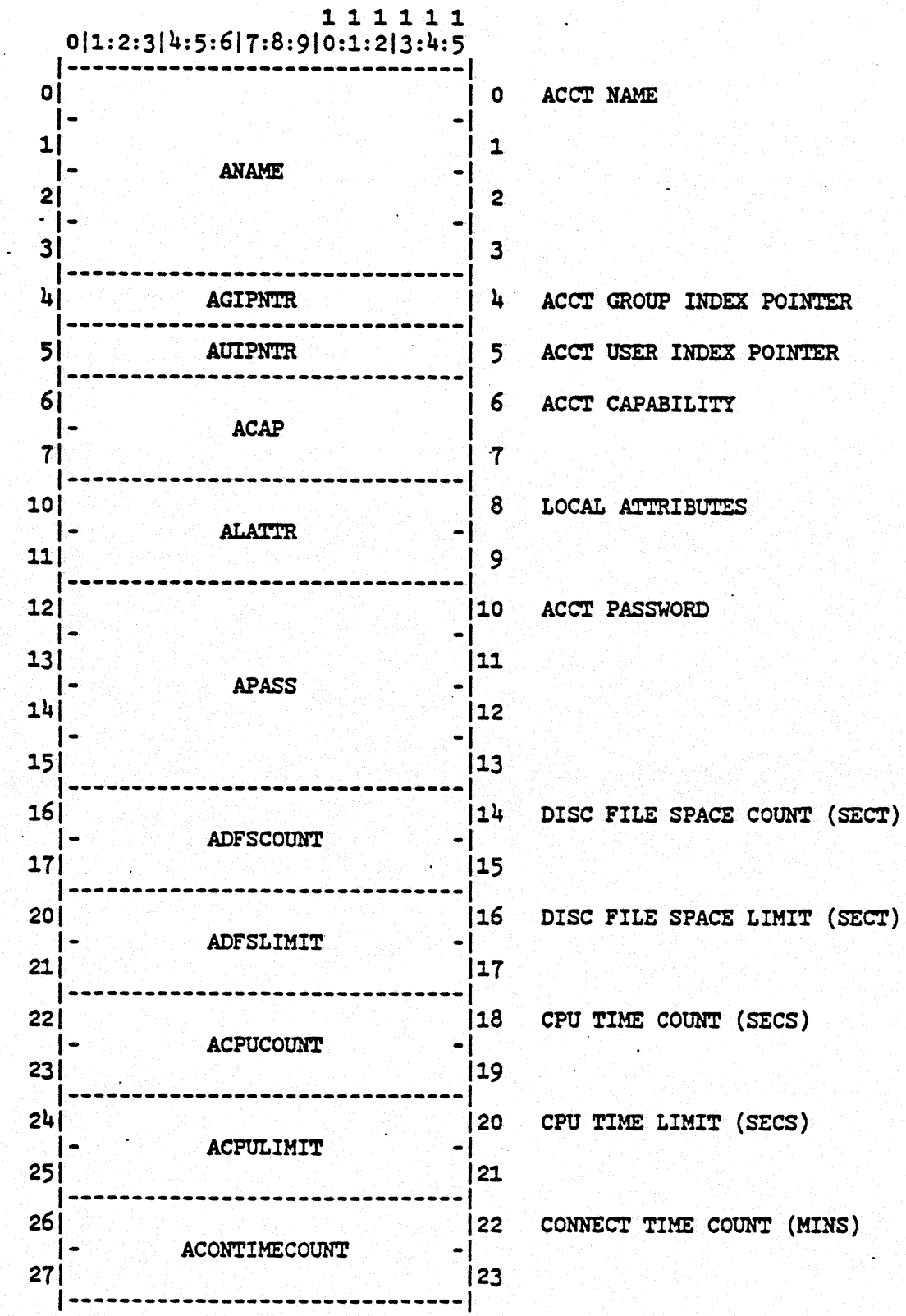
The preceding discussion is intended to give the reader a background of the design requirements, and thus a better understanding of the ultimate structure employed.

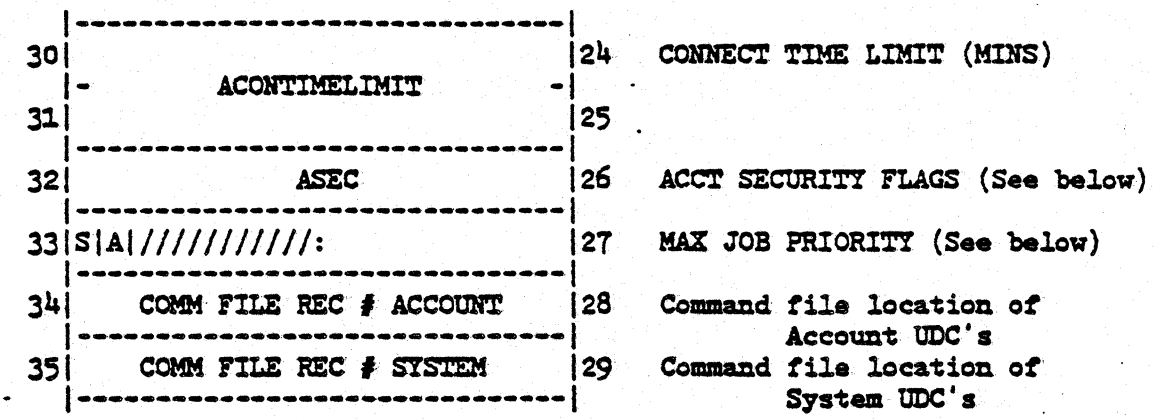
Figure 4.2.2-A INDEX PREFIX and ENTRY



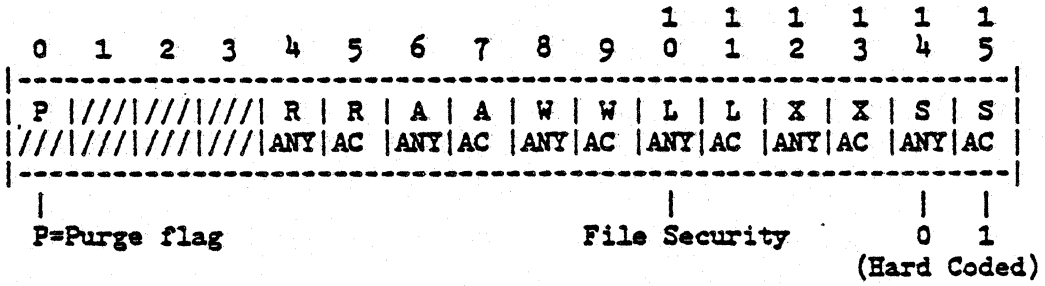
* IPCOUNT: the count is incremented by each access that uses and relies upon a pointer to the index block (eg., LISTF), and is guaranteed not to be purged while the count is not zero.

Figure 4.2.2-B ACCOUNT ENTRY





ACCOUNT SECURITY FLAGS

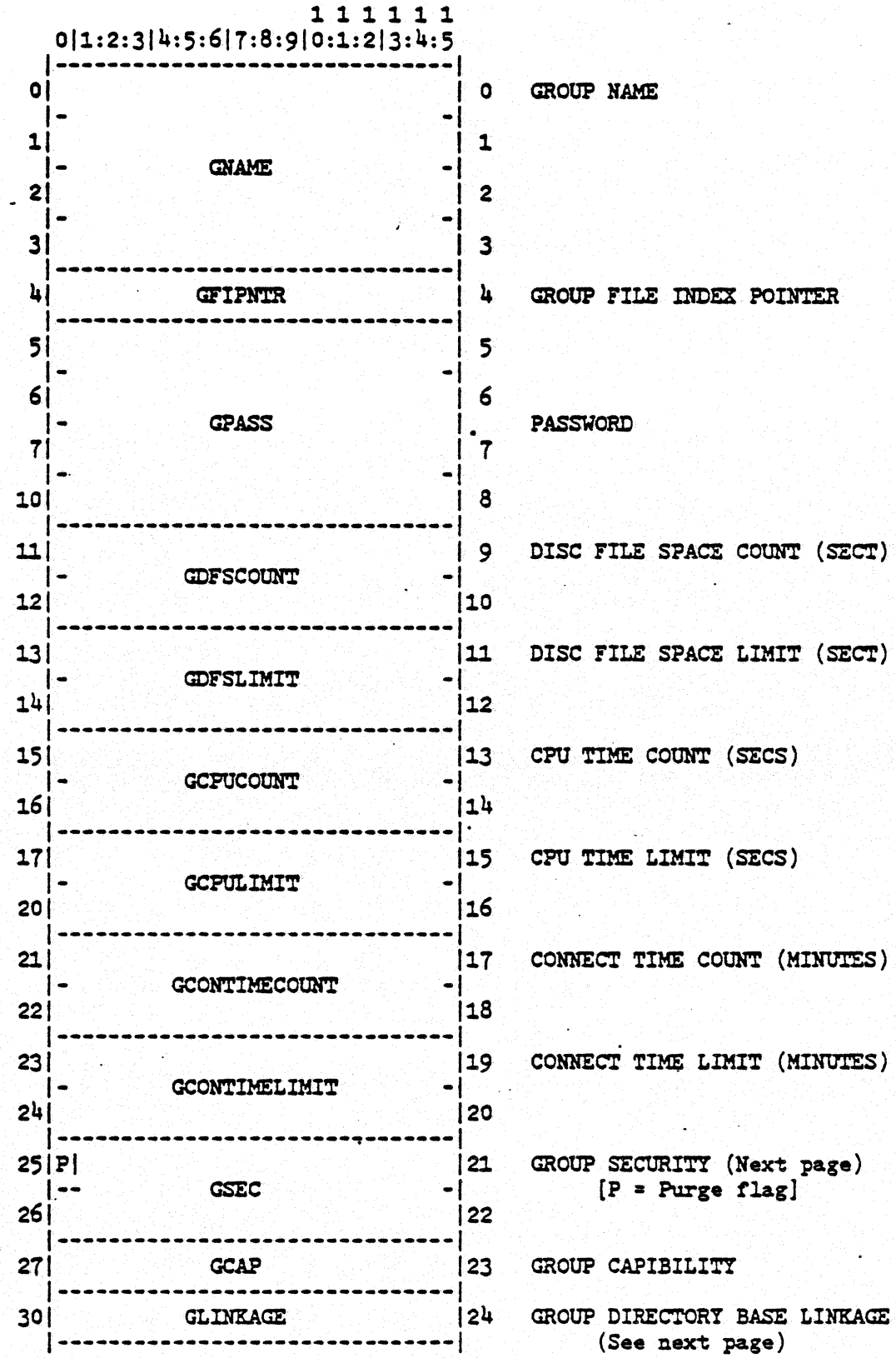


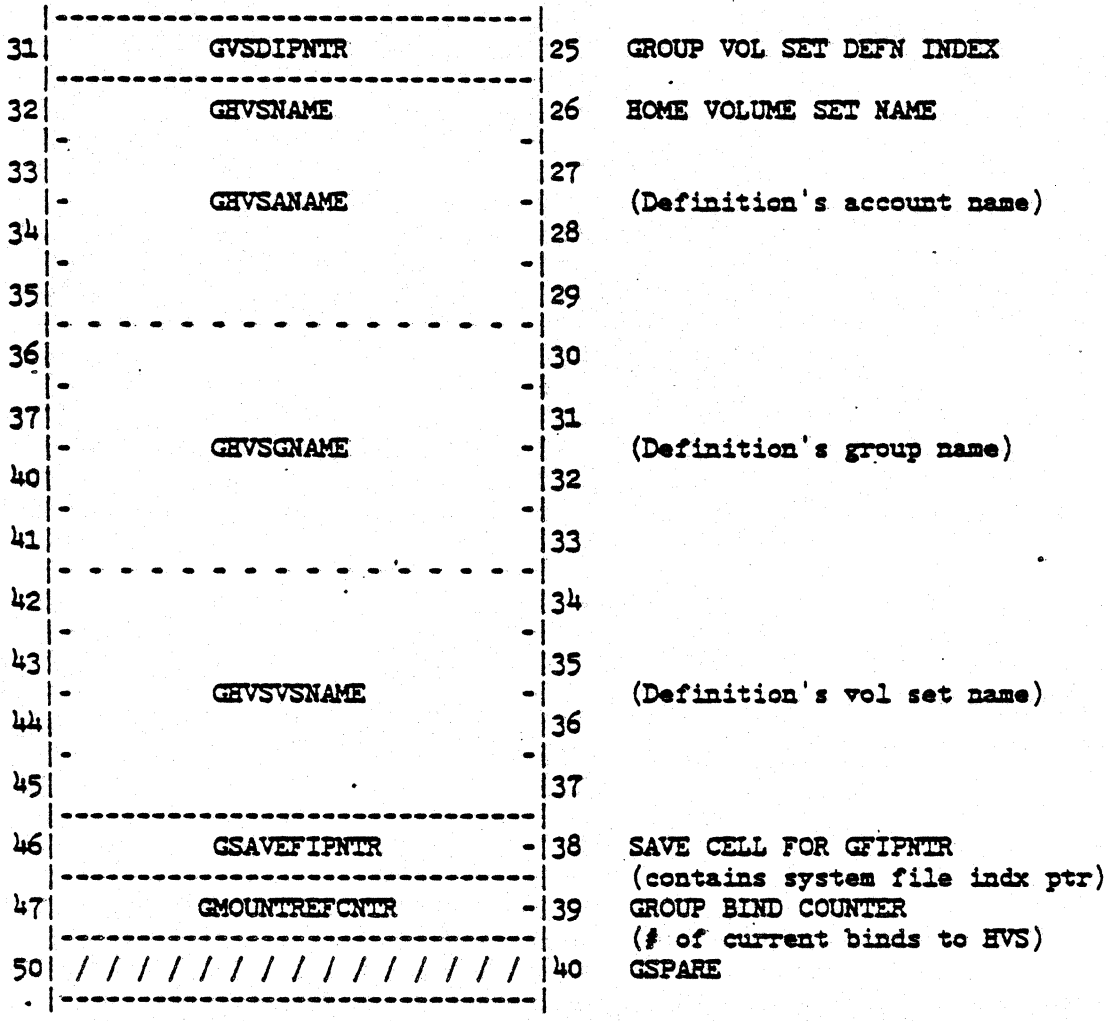
MAX JOB PRIORITY bits:

S (0:1) = 1 If system level UDC's exist (only in "SYS" account)

A (1:1) = 1 If account level UDC's exist for account

Figure 4.2.2-C GROUP ENTRY





G R O U P S E C U R I T Y M A S K

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	///	R	R	R	R	R	A	A	A	A	A	W	W	W	W
///	///	ANY	AC	AL	GU	GL	ANY	AC	AL	GU	GL	ANY	AC	AL	GU
W	L	L	L	L	L	X	X	X	X	X	S	S	S	S	S
GL	ANY	AC	AL	GU	GL	ANY	AC	AL	GU	GL	ANY	AC	AL	GU	GL

P=Purge flag

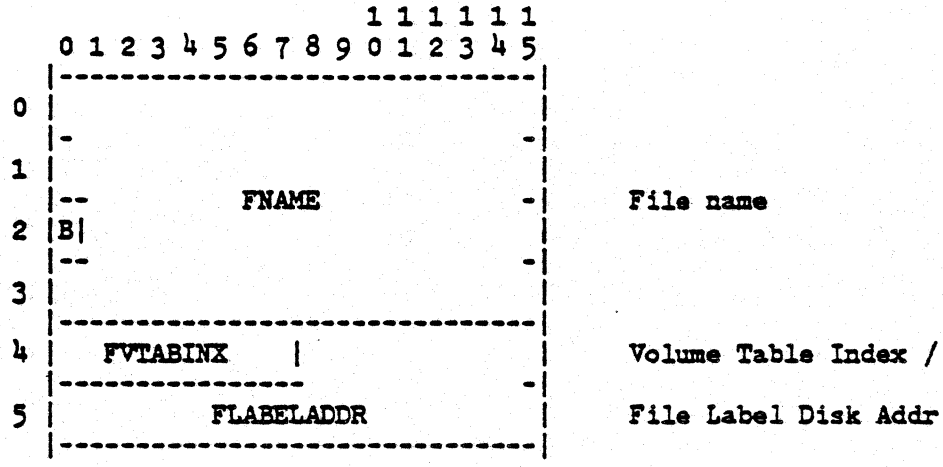
G L I N K A G E W O R D

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PV	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/
								M	V	T	A	B	X		

(0:1)=0 HVS is System
=1 HVS is Private Volume

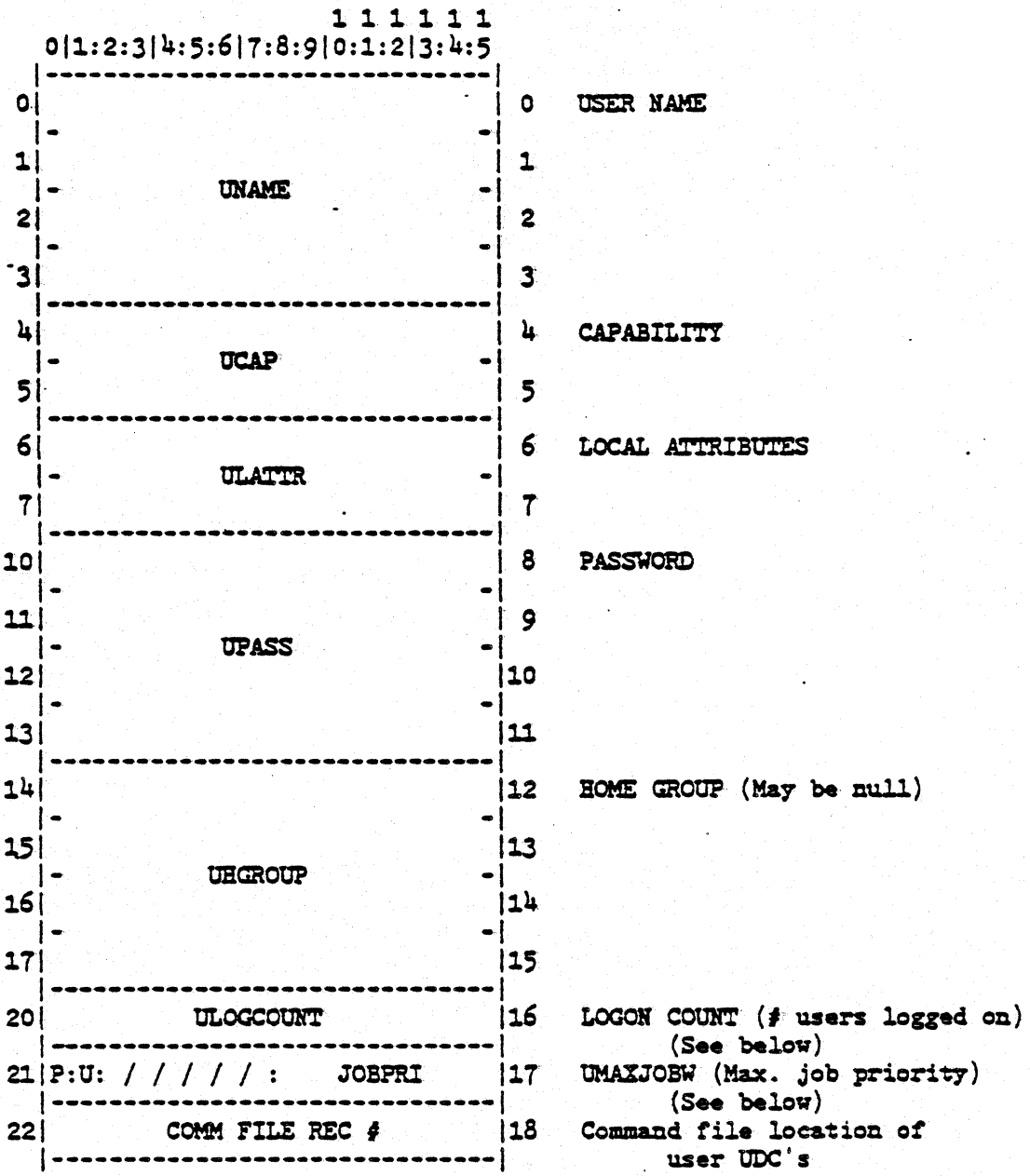
(8:8)=0 if not PV or NOT Bound
<>0 if PV and Bound

Figure 4.2.2-D FILE ENTRY



B: [word 2, bit 0:1 of FNAME] if set to a value, 1, indicates that the file label it points to is defective in some way. This bit is set by DIRECSETFLAG procedure and is called by FRESTORE or FLABIOERR. When flagged, the file label is no longer accessible and the file entry will be deleted dur- in a RECOVER lost disk space.

Figure 4.2.2-E USER ENTRY



ULOGCOUNT: Initialized to 1 for MANAGER.SYS so this user cannot be purged.

UMAXJOBW: P = Purge flag
U = User UDC exists

Figure 4.2.2-F ATTRIBUTES/CAPABILITY

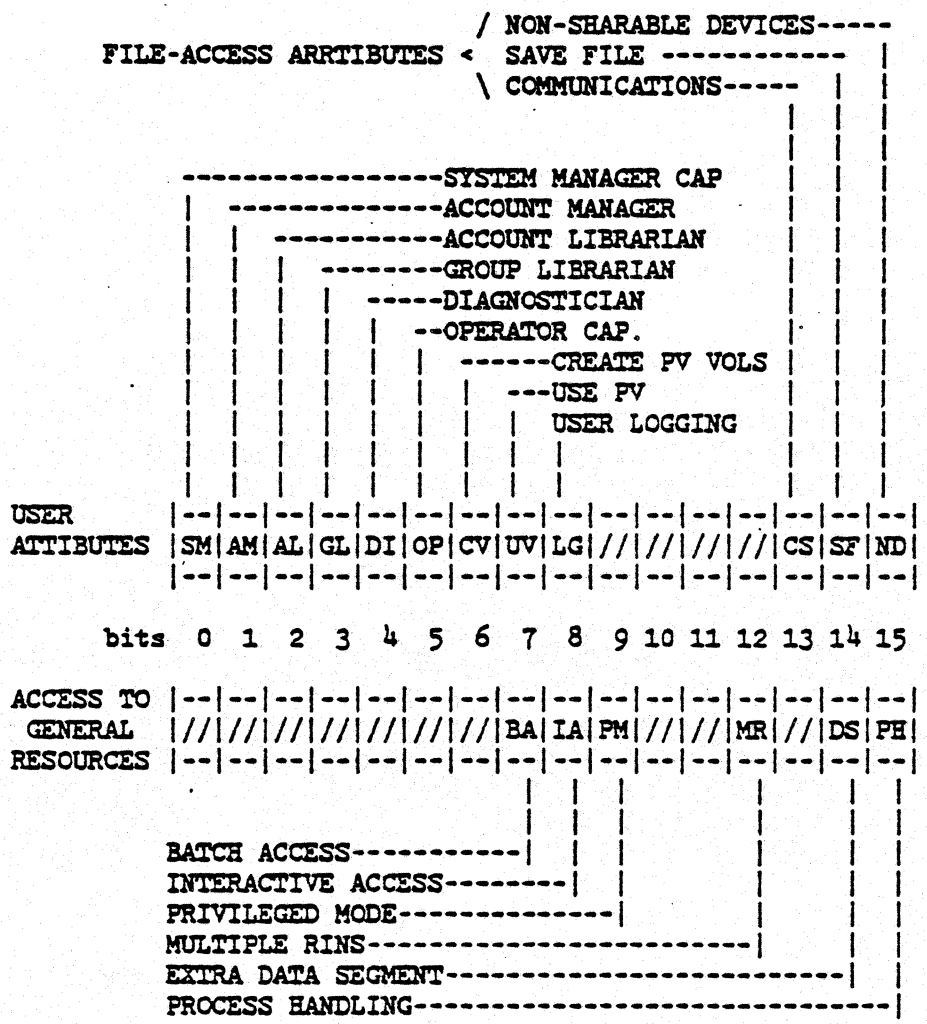
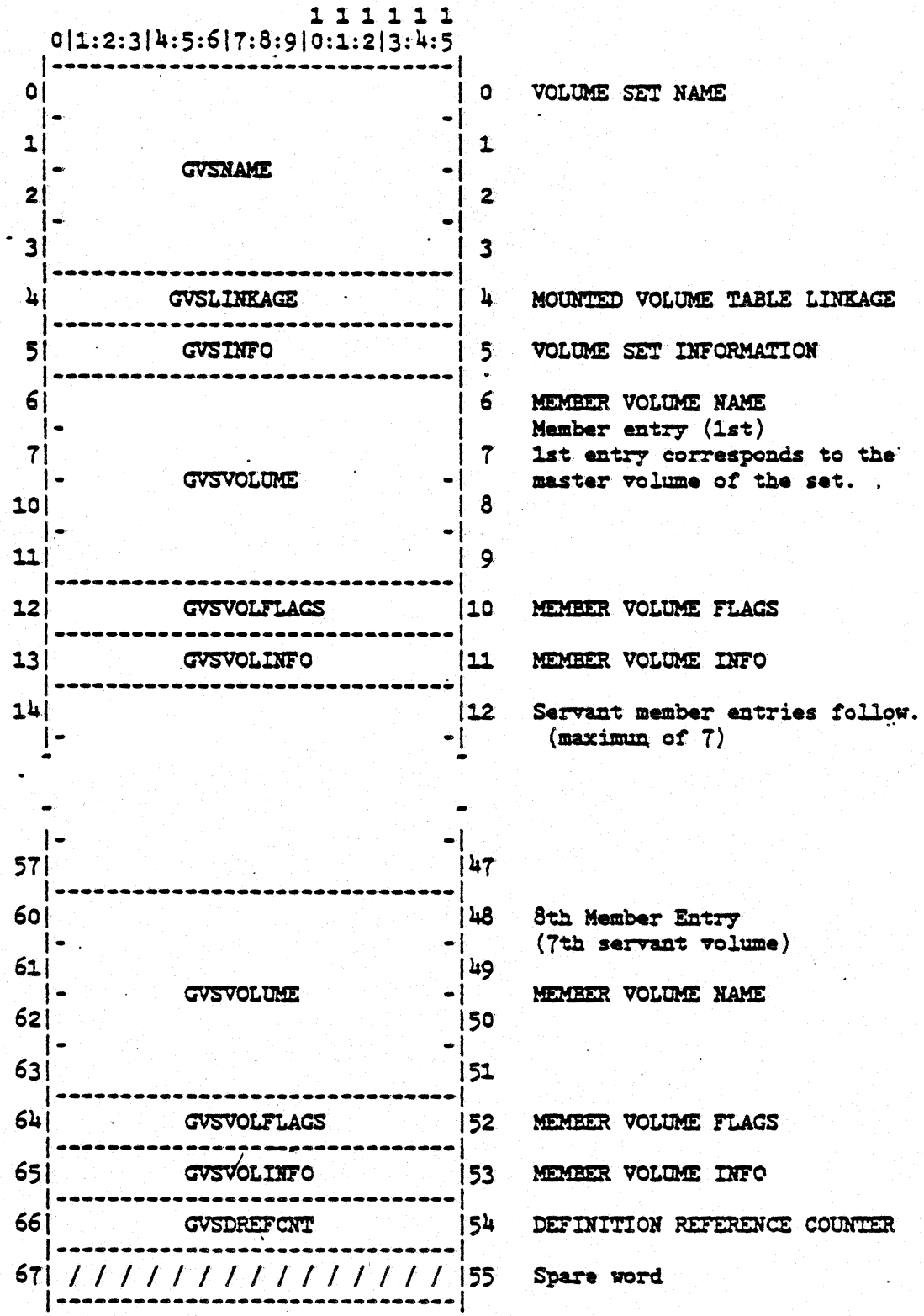
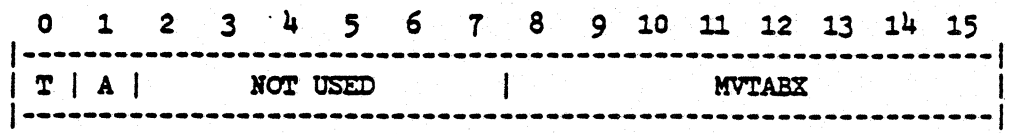


Figure 4.2.2-G VOLUME SET DEFINITION ENTRY



G V S L I N K A G E

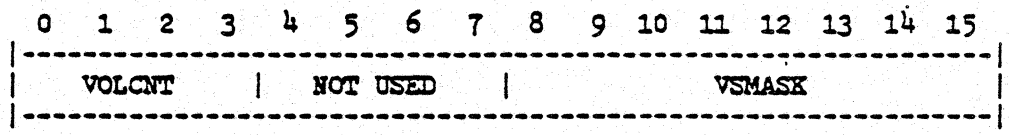


T - TYPE
1 = Volume Set Definition
0 = Volume Set Class

A - ALLOCATING FLAG
0 = not initialally allocating (not 1st user of set)
1 = 1st user of set allocating resources (transitional)

MVTABX - Mounted Volume Table Index
0 if volume set not logically mounted

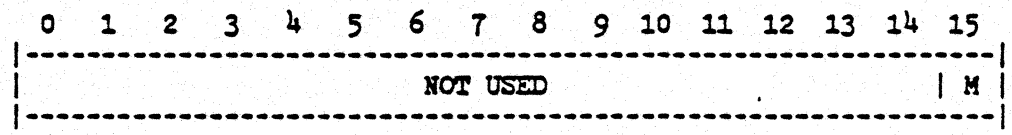
G V S I N F O



VOLCNT - Number of members in set

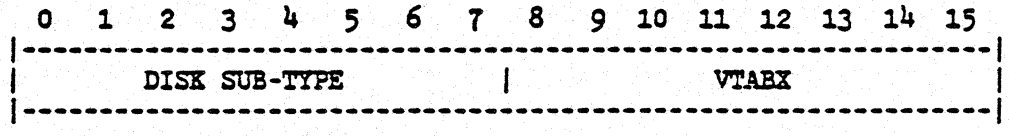
VSMASK - Bit mask of volume member usage
Order is from right to left
i.e. bit 15 is 1st member, bit 14 is 2nd member ...

G V S V O L F L A G S



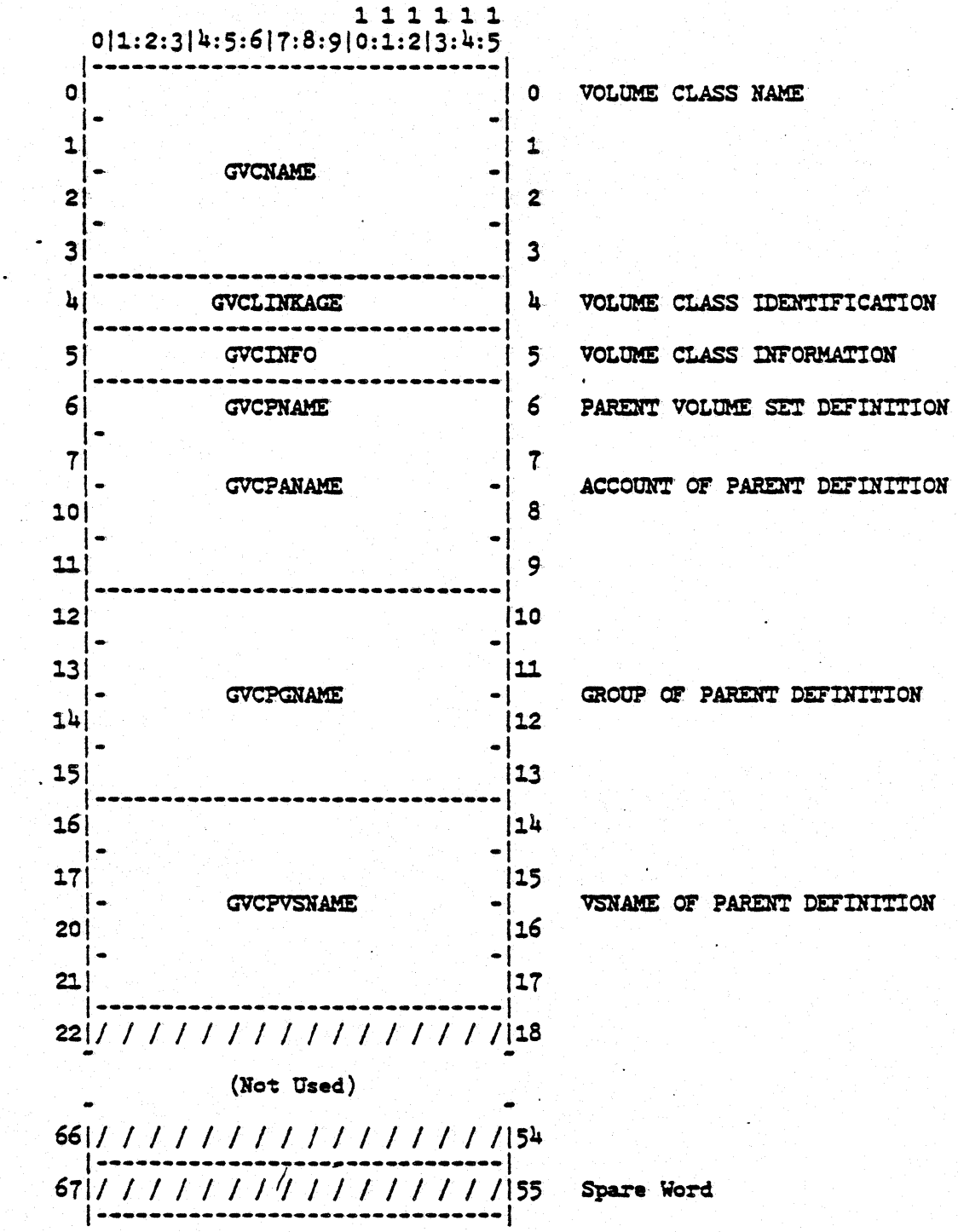
M - Member Mounted Flag
0 = not mounted
1 = mounted

G V S V O L I N F O

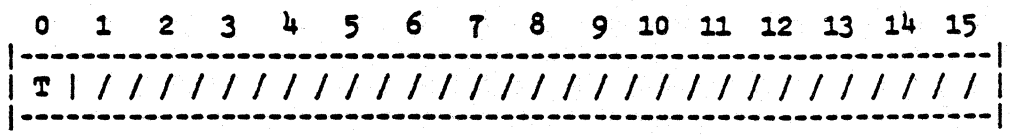


VTABX - Volume Table Index

Figure 4.2.2-H VOLUME SET CLASS ENTRY

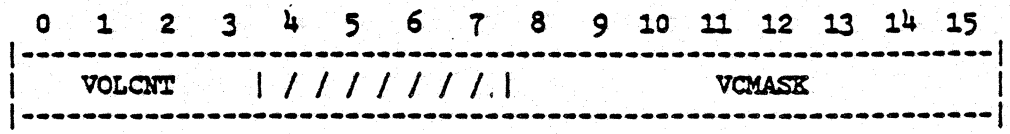


G V C L I N K A G E



T - TYPE
1 = Volume Set Definition
0 = Volume Set Class

G V C I N F O

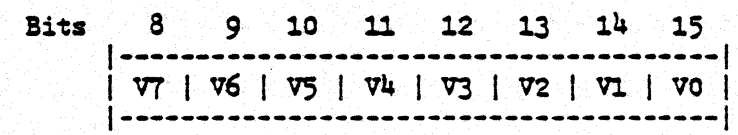


VOLCNT - Number of members in set

VCMASK - Bit mask of volume member usage
Order is from right to left
i.e. bit 15 is 1st member, bit 14 is 2nd member ...

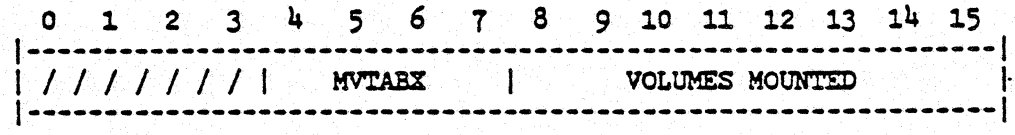
V O L U M E M A S K F O R M A T

Used in Mounted Volume Table, PVUSER Table, File Control Block (FCB), Volume Set/Class definition, Volume Table, and as a passed Private Volume related parameter between procedures (PVINFO word)



Volumes 0-7 in a volume set (one master and 7 slaves), where V0 is the master volume. Zero is not mounted or non-member, one is mounted or member of the volume set.

P V I N F O W o r d



4.3 DIRECTORY SYSTEM MANAGEMENT

Management of the directory, which includes finding, inserting, and purging of entries, is handled in two system data segments. DST %24 and %25 are used by the directory routines and are initialized at the start of the system by INITIAL.

One segment, called the Directory Space Data Segment, is used to manage the available and used sectors for the entire directory on the disk. This segment is used for all I/O operations involving the Bitmap for the directory, (shared by both system and private packs).

The other data segment is the Directory Data Segment which is used as the global variable area as well as for Input/Output operations.

INITIAL, during a startup of the system, will go through the entire system directory on disc and set to zero all of the various reference counts. There remains a problem for Private Volume directories, as INITIAL does not maintain those in the same fashion. In many cases, the disc is not even on-line. In the event of a system crash, the Private Volume directory may be in an inconsistent state where the counts were not properly decremented.

The Directory SIR is 8.

Figure 4.3-A DIRECTORY DISC LAYOUT

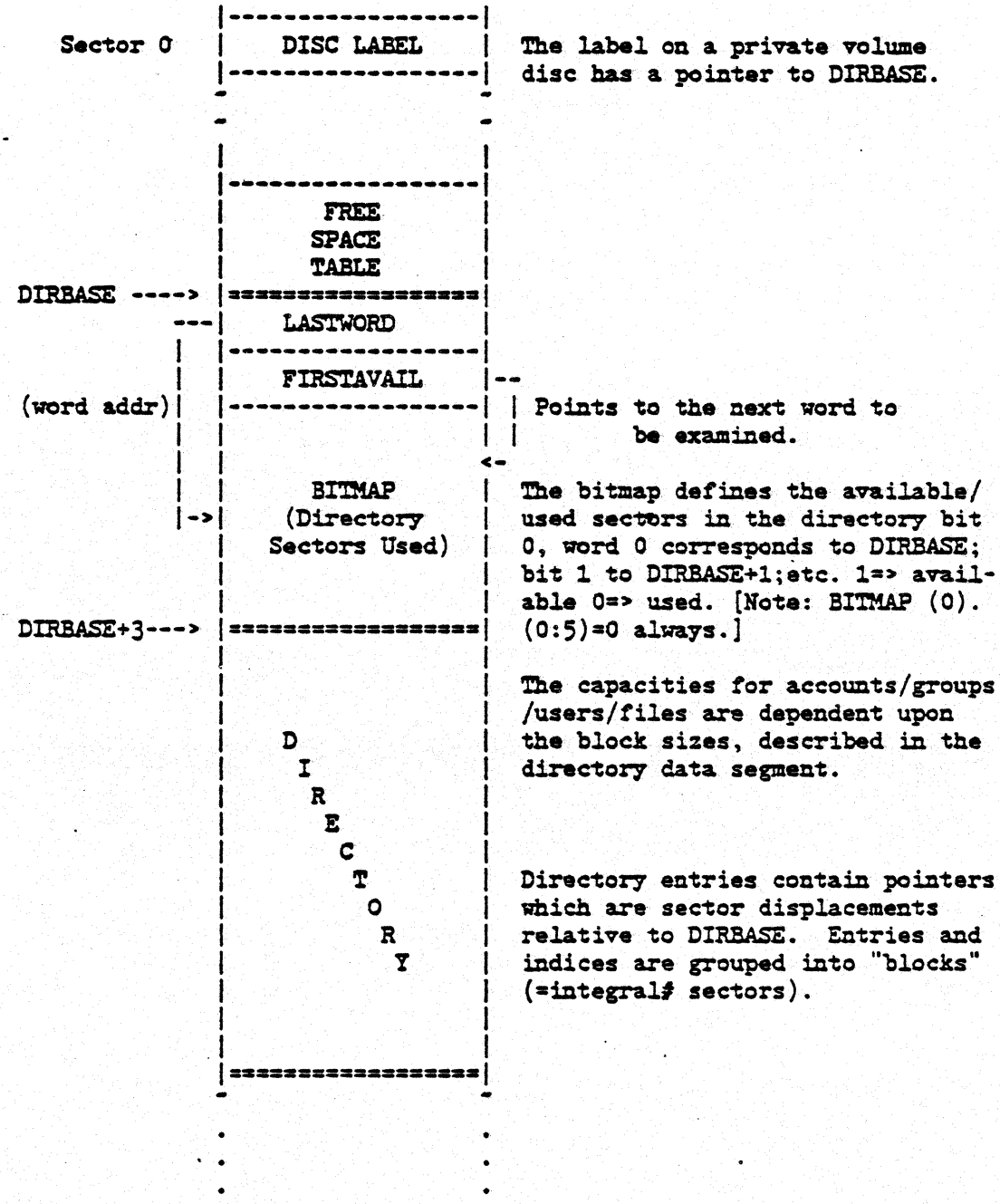
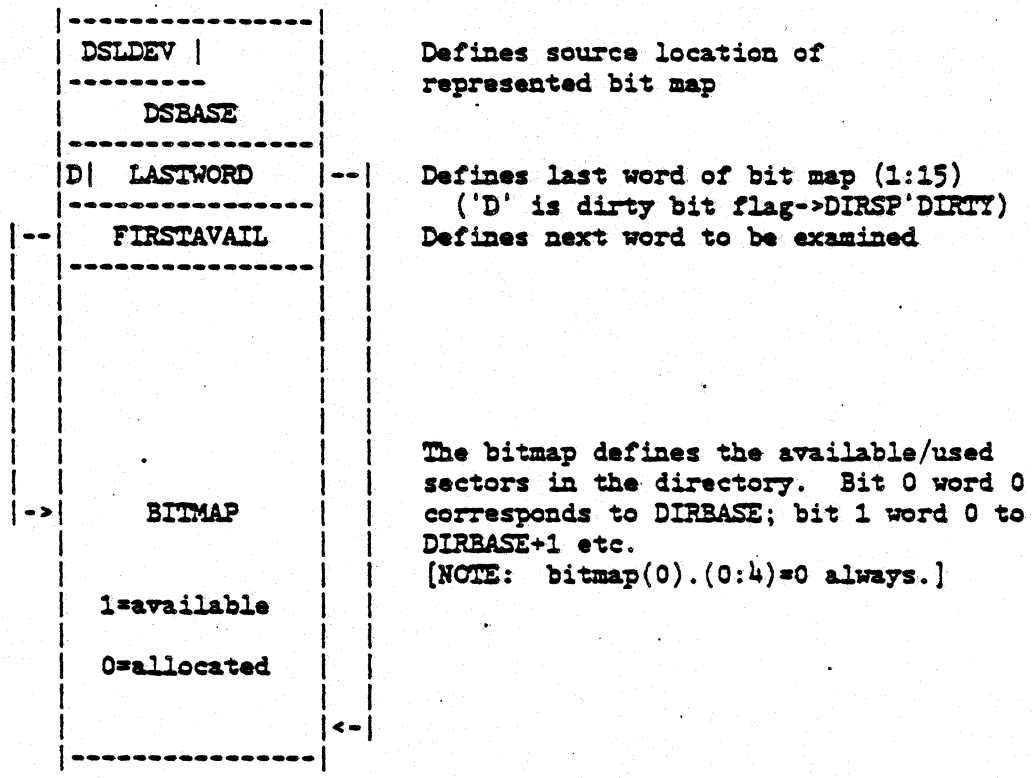


Figure 4.3-B DIRECTORY SPACE DATA SEGMENT (DSDS)

DST = 21 (%25)



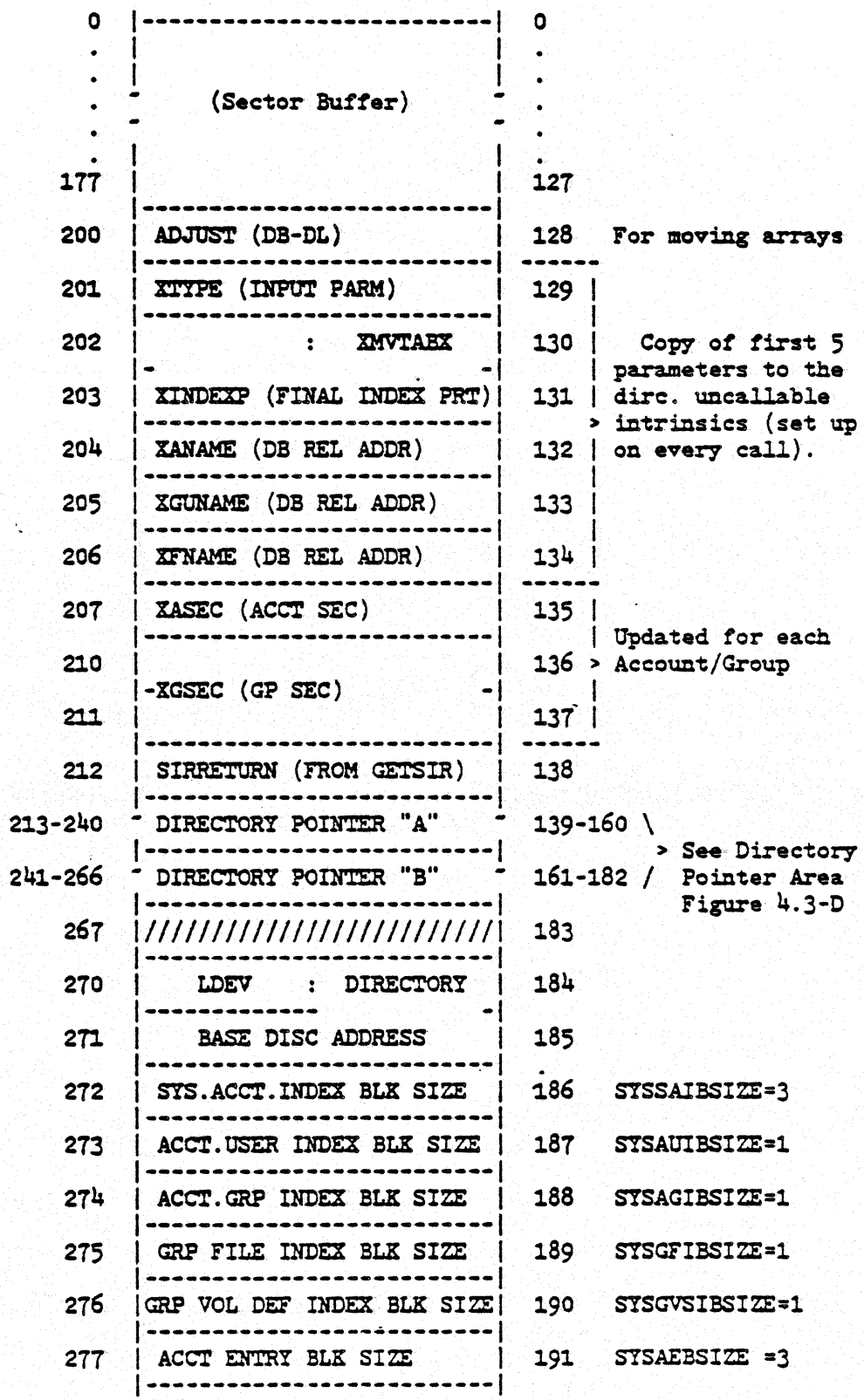
DIRSPSIZE (data segment size) is defined as

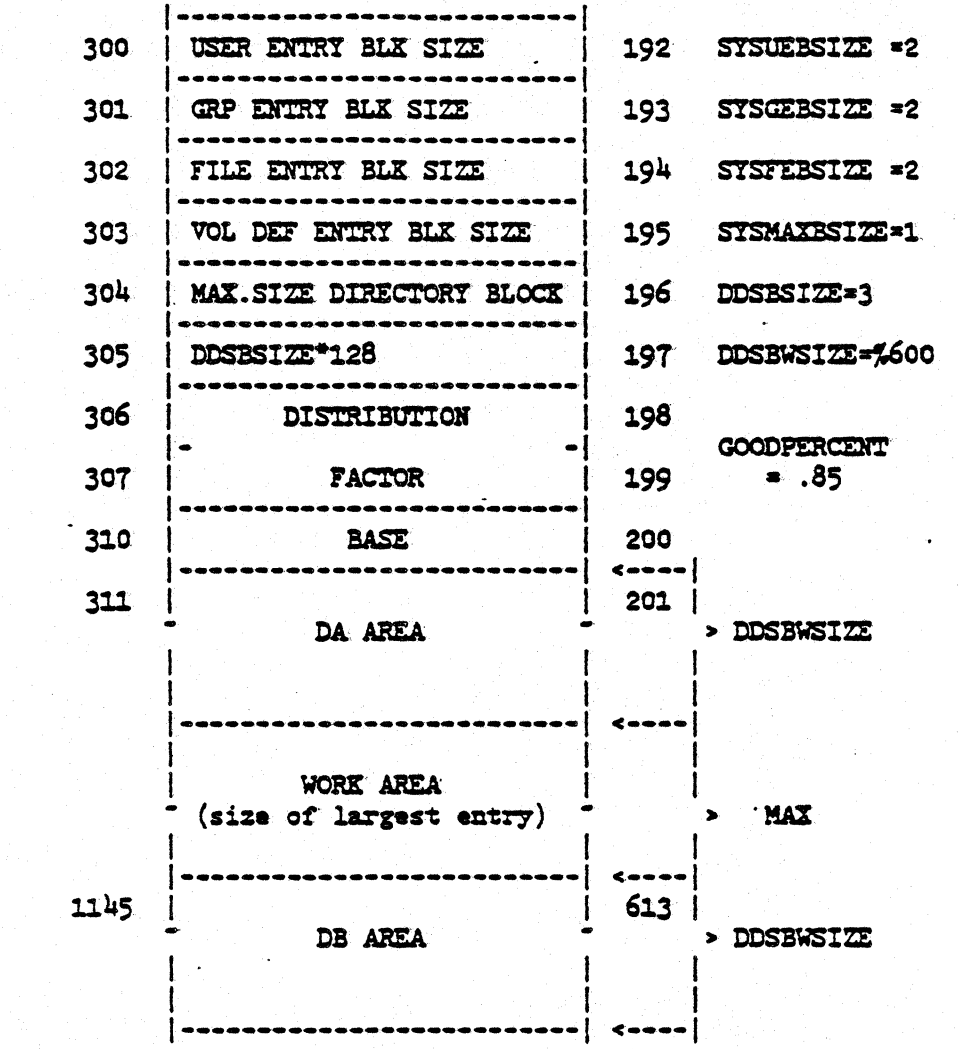
SYSACCTINDEX * 128

where SYSACCTINDEX=3.

Figure 4.3-C DIRECTORY DATA SEGMENT (DDS)

DST = 20 (%24)

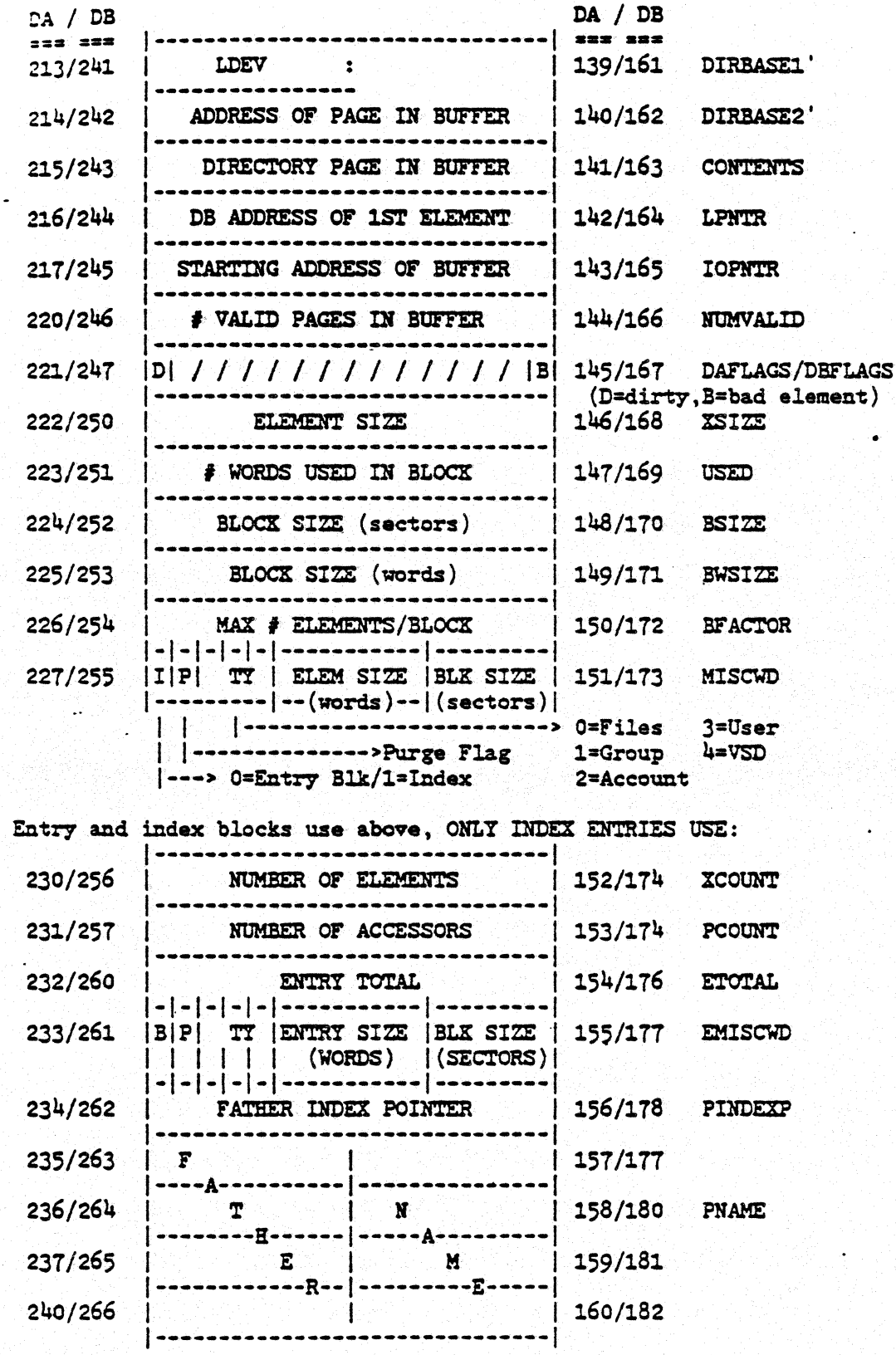




	*	SYSSAIBSIZE	System/Account Index
		SYSAUIBSIZE	Account/User Index
		SYSAGIBSIZE	Account/Group Index
		SYSGFIBSIZE	Group/File Index
BLOCK SIZE		SYSGVSIBSIZE	Group/VSD Index
NUMBER OF SECTORS	*	SYSAEBSIZE	Account Entry
		SYSUEBSIZE	User Entry
		SYSGEBSIZE	Group Entry
		SYSFEBSIZE	File Entry
		SYSVSEBSIZE	VSD Entry
		SYSMAXBSIZE	Max. of above to init. DDS

* These values are used once for the creation of the system (root), account index or new systems. This root index is always at address DIRBASE+3.

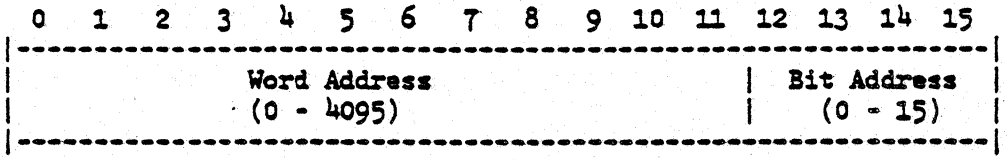
Figure 4.3-D DIRECTORY POINTER AREA [DA OR DB]



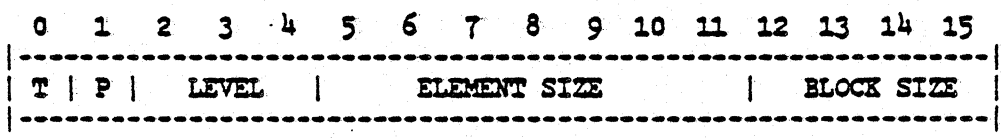
Entry and index blocks use above, ONLY INDEX ENTRIES USE:

Figure 4.3-E MISCWORD and TYPE Word Formats

Directory Bitmap Addressing



M I S C W O R D Format
Index and Entry Element Descriptors

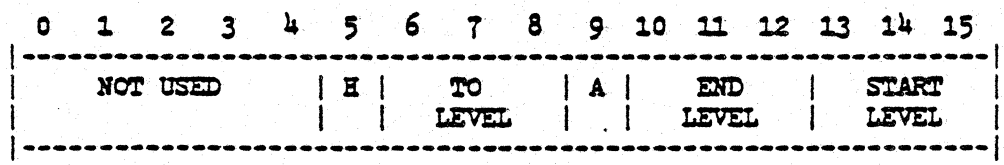


T - TYPEF: (0:1) XSIZEF: (5:7) BSIZEF: (12:4)
0 = Entry
1 = Index

P - IPURGEFLAGF: (1:1)

LEVEL - LEVELF: (2:3)
0 = Filelevel
1 = Grouplevel
2 = Accountlevel
3 = Userlevel
4 = Vsdlevel (Volume Set Definition)

TYPE Word Format



H - HITFLAG: (5:1)
TO LEVEL - TOLEVELF: (6:3)
A - ALLFLAG: (9:1)
END LEVEL - ENDLEVELF: (10:3)
ENDLEVELFX: (9:4)
START LEVEL - STARTLEVELF: (13:3)
0 = No Index - Start at root node
1 = Use Account index
2 = Use Group Index

DEFINITIONS

DIRBASE consists of a logical device number of the master volume of a volume set and the sector displacement to the base of a particular directory. The master volume of the system volume set, SYSVS, is defined to be logical device number 1).

POINTER is a one-word, positive, DIRBASE-relative sector displacement. DIRBASE+POINTER yields sector address.

A PAGE is the smallest quantum of allocatable space, i.e a sector. PAGE when used in a term such as "PAGE number" refers to the DIRBASE-relative sector displacement of a location.

A BLOCK is a contiguous group of PAGES; manipulated as one linear area - and is the basic unit of I/O transfer. ENTRIES and INDICES are grouped into BLOCKS. Directory blocks can be of different sizes.

SYSACCTINDEX is the fixed POINTER of the directory root; i.e. a POINTER to the system account INDEX BLOCK.

ENTRIES are the actual information-contining constituents of the directory. The essential purpose of the directory is to maintain ENTRIES; all othe directory components exist in order to make this maintenance possible. There are five kinds of ENTRIES, each of a different size: ACCOUNT ENTRIES, GROUP ENTRIES, USER ENTRIES, VOLUME SET DEFINITION ENTRIES (the abbriviation VSD will be used), and FILE ENTRIES. An ENTRY may contain a POINTER to subtrees. The MPE directory FILE ENTRY should not be confused with file labels: directory FILE ENTRIES contain disc addresses of file labels. ENTRIES are grouped into ENTRY BLOCKS.

An INDEX is a 6-word item associated with every ENTRY BLOCK. It contains the name of the first ENTRY in the BLOCK, a counter of the number of ENTRIES in the BLOCK and a POINTER to the ENTRY BLOCK. INDICES are grouped into INDEX BLOCKS.

ELEMENT is the generic term for an item which is either an INDEX or an ENTRY.

The five types of subtrees of the MPE directory are (INDEX BLOCKS) known as SYSTEM ACCOUNT INDEX (directory root), ACCOUNT USER INDEX, ACCOUNT GROUP INDEX, GROUP VOLUME SET DEFINITION INDEX, and GROUP FILE INDEX. These terms refer to a specific INDEX BLOCK unless used in a descriptive context. Every ACCOUNT ENTRY contains a POINTER to its ACCOUNT USER INDEX BLOCK and a POINTER to its ACCOUNT GROUP INDEX BLOCK, etc.

The INDEX BLOCK PREFIX (or simply PREFIX) is the first part of every INDEX BLOCK and contains information necessary to access the INDICES, and the ENTRIES of the ENTRY BLOCK to which the INDICES point.

4.4 BASIC ALGORITHMS

4.4.1 SPACE MANAGEMENT

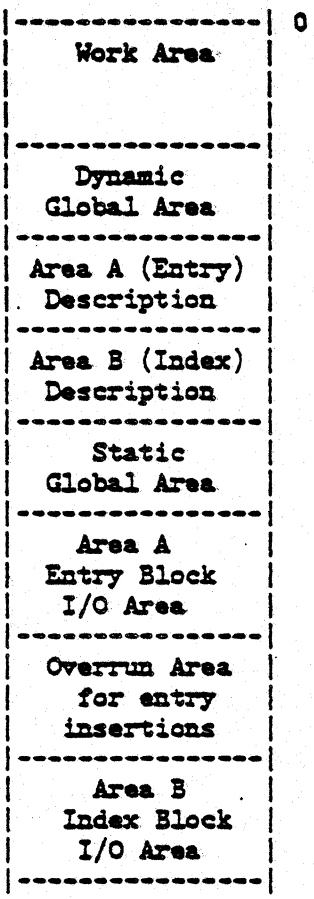
The first three sectors of the directory disc area contain a space bitmap for the entire area. [The system account index always exists and is the first block created for a null directory. Therefore SYSACCTINDEX is always 3.] The DIRECTORY SPACE DATA SEGMENT is a data segment whose length is three (3) sectors plus two (2) words in length. This data segment is maintained in virtual memory. The two extra words prefix the data segment and contains the disk address and logical device number of its parent directory. This mechanism allows for the maintenance of multiple directories. The bitmap portion of the data segment (base+2) is written to the appropriate volume whenever the bitmap has been changed by either an allocation or deallocation of the directory space. Directory space management is entirely accomplished in this data segment.

The format of this area is shown in Figure 4.3-B. A bit set to 1 implies that the corresponding page is available. The size of the directory disc area is implicitly defined by the number of sectors available as indicated by the bits on between (and including) the fourth (relative to zero) word and LASTWORD. FIRSTAVAIL is a is a cycling pointer to the first word that is to be interrogated for availability: space allocation takes place cyclically to minimize fragmentation.

4.4.2 DIRECTORY DATA SEGMENT

The directory data segment (DDS) is shown in detail in Figure 4.3-C and 4.3-D. This data segment is used as the I/O buffer for the directory, and all directory management is effected in it. It can be divided into the following areas:

DDS - Directory Data Segment



AREA A and AREA B are used as the directory I/O buffers and are large enough to accomodate the largest directory block. Two areas are required primarily because of the entry insertion algorithm which splits blocks into two blocks at times. This algorithm is described below. AREA A is used solely for entry blocks and AREA B is used primarily for index blocks (in addition to entry block splits). The OVERRUN AREA is as large as the largest entry and also is provided for entry insertion.

AREA A DESCRIPTION and AREA B DESCRIPTION are (directly-addressable) variables that contain various attributes and characteristics of their respective areas. (e.g. number or elements, dirty flag, page number, etc.) For index blocks, this information is obtained from the index block prefix. For entry blocks this information is obtained from the entry block's index and the index's block prefix. These areas are set whenever a block is read into the DDS, and always reflect the correct attributes. The last part of these DESCRIPTION AREAS have space for index prefixes. When the block they contain is an index block, this area is used as temporary storage during certain types of entry insertion situations.

The STATIC GLOBAL AREA contains configuration parameters and is never changed after a COLD LOAD.

The DYNAMIC GLOBAL AREA contains information pertaining to the last call of a directory uncallable intrinsic. As will be seen later, there is considerable commonality of parameters and necessary setup among these uncallable intrinsics. This area contains some of this information that is needed by some of the internal directory procedures.

The first 128 words of the DDS are used for two purposes. The beginning area is used for submission and return entry information to/from internal directory routines. The entire area is also available as a work area for the "recipient" procedures of DIRECSCAN (as will be explained later).

4.5 PRIMITIVE FOR BASIC MANIPULATIONS

In light of the data structure described above, the algorithms to manipulate the directory follow directly. This section considers the cases of finding, purging and inserting entries into the first level of a subtree. The descriptions are overviews of the operations of internal directory primitives. The next sections describe extensions necessary for the hierarchial structure and special file system considerations.

Given a pointer to subtree and a first level name to find, the first step is to read the designated index block (into area B) and find the index whose name is the greatest among those which are equal to or less than the target name. This index points to the entry block that must contain the desired entry. The entry block is read into area A, and the entry is searched for.

In order to purge a first level entry from a designated subtree the first step is to find the entry, as described above. Once it has been located, the entry is removed from the entry block by moving the succeeding entries of the entry block so that the space occupied by the target entry is overrlayed. Several variables (e.g. counts) are updated to the index block (and the DDS description areas). If the entry was the only element in the entry block, the block is deallocated and its index is removed. Index blocks are never deallocated by any of the primitive facilities because they are considered to be a necessary component of their parent entries. (Index block deallocation is done by higher level routines.)

The insertion primitive is the most complicated. Again, the first step is to locate the entry block into which the new entry should be inserted (if the target name is less than the first index name, the first entry block is considered.) If it can fit into that entry block, it is inserted (counts are adjusted, etc.) and the operation is complete. If no entry blocks currently exist, one is allocated, an index is created for it in the index block, and the entry is inserted into the new entry block. If an entry won't fit into an entry block then a re-adjustment is required: The logically neighboring block with the lowest number of entries is chosen, and the total space utilization of the two blocks is checked against a "good percentage" (85%). If utilization is less than this percentage, the entries of the two blocks (including the new one) are equally distributed between the two existent entry blocks. If the space utilization criterion is not satisfied, the neighboring block is not satisfied, the neighboring block is no longer considered, and the entries (of the old entry block plus the new entry) are equally distributed between the existant block and a newly created one.

5.1 TREE-STRUCTURE CONSIDERATION FOR BASIC MANIPULATIONS

Once the algorithms for the primitive manipulations against one level have been established, it is necessary to extend them to cover the entire tree structure. For MPE, the important constituent of a directory node is the information it contains. For directory manipulations, the important consideration is that a directory node may have subtree (depending on the level of the node - The MPE directory is further distinguished from a generalized tree structure in that subtrees permissable for a given level are pre-defined).

The following descriptions apply to a basic class of directory uncallable intrinsics which primarily operated directly on the directory data structure, ignoring entry-dependent considerations described below. These uncallable intrinsics are DIRECFIND, DIRECINSERT and DIRECPURGE.

Both global references and subtree-type references are allowed for these functions. The latter type of specification permits a search to begin at a designated subtree directly, thus avoiding several accesses. [This form is primarily used when accessing logon group of logon account files; i.e. when a fully-qualified reference is not supplied.] For the implementation of this concept, a common initialization routine is used [DIRSTARTOFF]. It analyzes the caller's specification, performs any necessary directory accesses and returns (to the directory uncallable intrinsic) the final entry name and associated index block pointer for the target. Once the target is so distinguished, the pertinent directory uncallable intrinsic then completes the operation by performing the necessary juggling to effect the requested insert, purge or find. The following overviews describe the operations implemented after the target has been "marked" as just explained.

The find operation simply returns the desired node, using the appropriate primitive.

Insertion uses the insert primitive to insert the entry at the designated level. But depending on the level, additional provisions for subtrees must be made. Specifically, a group file index must be created when inserting groups; an account user index and an account group index must be created when inserting accounts. Simple entry insertions are made for files and users.

Purging is complicated by the requirement that when an entry is purged, its subtrees, if any, must also be purged. When purging user or file entries, only the entry is deleted and no further actions need be taken. However, when purging accounts or groups, their respective subtrees must first be purged. To do this, the designated tree is scanned in endorder [endorder is the

traversal of a tree by visiting all subtrees before visiting the root, where a visit of a tree means traversal of it in endorder]. The target root (i.e. the account or group entry to be removed) is deleted only if all its subtrees have been successfully removed.

4.5.2 ACCOUNTING AND FILE SECURITY CONSIDERATIONS

The following considerations have been made for permanent file space accounting and file security (both functionally defined to the ERS). These special actions are invoked through special directory uncallable intrinsics [DIRECFINDFILE, DIRECPURGEFILE and DIRECINSERTFILE], that are available to MPE callers manipulating file entries and requiring these functions. These routines are to be distinguished from the basic complement of directory uncallable instrinsics described above, which operate on directory entries without regard to accounting and security. [There is one exception to the last comment: purging a group adjusts the account's file space accordingly.] Note that these special considerations are made only when dealing with files.

The special provisions for finding a file consist of returning additional information regarding file security at the account and group level. Otherwise it is identical to a general directory find of a file entry.

When inserting or purging files under these provisions, however, the file space counts of the account entry and group entry must be adjusted. Consequently, such accesses are always global because the group and account entries must be visited anyway to update the counts. In addition, insertions insures that the user has SAVE FILE (SF) access to the group. When purging, the caller has supposedly determined that the user has WRITE access.

4.5.3 SCANNING AND UPDATING

Certain commands are designed to operate on a set of directory entries [:LISTF, :LISTGROUP, :LISTUSER, :LISTACCT, :STORE, :RESTORE, :REPORT, :RESETACCT, LISTVS]. The concept of set is defined in the ERS and a consistent syntax is employed: the final directory level is implied by the command; lower levels precede higher levels, separated by periods; omitted trailing specifications imply "logon ..." and leading "@"s mean all..." at the corresponding level.

There is one directory uncallable intrinsic that is used for all these functions, DIRECSCAN. As arguments, it takes a "recipient procedure" and a parameter array. It scans the designated subtree in preorder, visiting each node [preorder is the traversal of a tree by visiting its root node before traversing its subtrees]. This routine visits a node by calling the "recipient procedure", passing it the visited entry and the parameter array, which has no relevance to the directory routines.

The recipeint procedure can, each time it is invoked, simply examine the contents of the directory node [:LISTF, :LISTGROUP, :LISTUSER, :LISTACCT, :REPORT, :STORE, LISTVS], or actually modify it obeying certain restrictions described in a later section [RESETACCT, :RESTORE]. Because the entry can be changed, this routine can be used as an update procedure [:RESTORE]. The directory can be released by the recipient procedure if necessary [such would be the case when return to the directory routine might be after a relatively long period of time (e.g. listing)]. This directory routine uses the pointer reference count mechanism described earlier to ensure that the current index block is not removed, when the directory is unlocked.

For later discussion, it will be helpful to note at this time that the specification of a subtree of targets requires the specification of a root and the leaf level. The root specification depends on whether this is a global, logon account or logon group specification.

..5.4 LIMITATIONS

It may be seen from the description of the insertion algorithm that once the index block is full, that subtree can accommodate no more entry blocks. Thus the following situation exists when an entry cannot be inserted: the entry block that should accommodate the entry is full, a distribution with a neighboring block fails the good percentage test, and no more blocks can be allocated because there is no more room for indices. But not all the entry blocks are full at this point, so that we are getting partial utilization of all the available space. The space utilization in these cases will be discussed later.

As mentioned previously, entry blocks are not all of the same size. The block size depends on the block type. In this implementation, the following values were chosen:

	VARIABLE NAME	WORDS/ ENTRY	BLOCK FACTOR	NO. OF SECTORS
SYSTEM ACCOUNT INDEX	SYSSAIBSIZE	6	62	3
ACCOUNT USER INDEX	SYSAUIBSIZE	6	19	1
ACCOUNT GROUP INDEX	SYSAGIBSIZE	6	19	1
GROUP FILE INDEX	SYSGFIBSIZE	6	41	2
VSD INDEX	SYSGVSIBSIZE	6	19	1
ACCOUNT ENTRY	SYSAEBSIZE	30	13	3
USER ENTRY	SYSUEBSIZE	19	13	2
GROUP ENTRY	SYSGEBSIZE	25	6	2
FILE ENTRY	SUSFEBSIZE	6	42	2
VSD ENTRY	SYSVSEBSIZE	56	3	1
(MAXIMUM BLOCK SIZE)	SYSMAXBSIZE	-	-	3

Obviously, a tradeoff existed in the establishment of these values: potential available space vs. potential wasted space. These values were set in consideration of:

1. The need to accommodate more of certain elements (e.g. files).
2. The expected variance in number of elements at a given level.
3. The space utilization percentages for certain index/entry block sizes [the average space utilization and space utilization when full are dependent on the index block and entry block factors].
4. The lower the level the more critical is the value, because there are more blocks of that type (e.g. the system account index could've been much larger than 3 at small cost in space, since there's only one).

The block sizes are revelant primarily in determining block factors:

BLOCKFACTOR= FLOOR[((BLOCKSIZE*128)[-PREFIX])/ELEMENTSIZE]

The theoretical entry limit at a level is the (index factor) * (entry block factor). But as was noted above, an insertion failure will occur before the entry blocks are entirely full. Simulation has shown that the entry blocks will be between 79-82% full when the first failure occurs at a level. Thus, the practical limit of number of entries at a level is about 80%; theoretical limit. The ERS shows these practical limits for entries per level, based on 80.5%. Note that this is the expected utilization as the first failure: it may be possible to still add entries which will be accepted if they fall within partically full entry blocks, or distribution is possible.

Simulation has also shown that, before failure, the entry blocks are 76-80% full on the average. So to calculate the space used during normal operation, the effective entry block factor to use should be about 78% of the actual block factor. The formula for the space used by the directory in the ERS, uses these practical block factors in the ceiling expressions, which are the number of entry blocks used by the indicated entries. The formula is a simplification of:

space bitmap	3
+ index/entries for accounts	+3+3*a/10.15
+ (#accounts) * (index/entries for groups)	+a*(1+2*g/7.8)
+ (#accounts) * (index/entries for users)	+a*(1+2*u/10.13)
+ (#groups) * (index/entries for files)	+a*g*(2+2*f/32.75)

4.6 DIRECTORY UNCALLABLE INTRINSICS

4.6.1 COMMON CONVENTIONS

4.6.1.1 ENVIRONMENT

All directory uncallable intrinsics must be called from the stack. If called with any of the following SIRs locked, the caller must respect this nested SIR order:

FILESIR (37)=> DIRECTORY (8)=> LPDT (9)=> DISC SPACE(12)

All directory routines always acquire the directory SIR, but don't directly invoke any procedures which acquire any other SIRs, except, of course, the "recipient procedure" of DIRECSCAN.

The following intrinsics are always invoked by the directory uncallable intrinsics:

GETSIR/RELSIR	- to acquire and release the directory
EXCHANGEDB	- to shift to the DDS and the directory space data segment, if necessary
ATTACHIO	- if an I/O is required. I/O will not be required only if the necessary directory block(s) is already in the DDS.

The directory routines access SYSGLOB+%130 and +%131 as DIRBASE when the volume set being accessed is the system volume set. This is an exception to the normal algorithm for establihsing DIRBASE for a volume set and is used merely for efficiency.

When a Private Volume directory is being manipulated, the DIRBASE is retrieved from the disc label at sector 0 (word %17). Once the PV disc is mounted on the system, the DIRBASE for this volume can be located in the Mounted Volume Table.

4.6.1.2 RETURNS

All directory uncallable intrinsics, except DIRCLOGON/ DIRECLOGOFF, are double procedures and use the following returns [shown are (S-0), S-1) and "CC"; these will be referenced in the various intrinsic descriptions to show the possible returns from the particular intrinsic]:

cc	(S-0)	(S-1	Meaning
CCE	0	0	Successfull return.
CCL	0	0	I/O error. (Currently never returned. The routines call SUDDENDEATH (4) upon detecting any I/O error.)
CCG	1	0	Duplicate name on insertion.
	2	n	Non-existent name at some point in search. (S-1) indicates the level of the non-existent node: n=0 file =3 user 1 group =4 VSD 2 account
	3	n	User does not have SAVE FILE access to <n>: 1 group 2 account
	4	n	No room. Insertion failed because subtree cannot accomodate any more entry blocks. n% of total entry space actually in use.
	5	0	No room. More than 65K entries. Currently never returned because of block sizes).
	6	n	No room. System directory has no room to accommodate <n> contiguous blocks.
	7	0	Entry cannot be purged because it (or some constituent of its subtree) is in use.
	8	n	Permanent file space limit would be exceeded for <n>: n= 1 group 2 account

Every routine can return CCE, CCL and certain CCG failures. Those CCG failures possible are noted in the descriptions. For any CCG failure, the directory is left in the state when called - no part of the operation has been effected.

4.6.2 GENERAL DIRECTORY ENTRY MANIPULATION

4.6.2.1 CALLING CONVENTIONS

All of the following uncallable intrinsics use a call of the following form (appended intrinsic-relevant parameters):

```
DOUBLE PROCEDURE directory (TYPE, LINKAGE'INDEXP,
ANAME, GUNAME, FNAME...);

    VALUE TYPE, LINKAGE'INDEXP, ...;
    INTEGER TYPE ...;
    DOUBLE LINKAGE'INDEXP ...;
    ARRAY ANAME, GUNAME, FNAME, ...;
```

These initial parameters are common to all these intrinsics and serve to define the target by specifying:

1. What level is the desired target (file, group, account, user, volume set definition):

2. If this is a shortened search, where to start. This occurs in the case of finding an entry in the logon group or logon account. The JIT contains index block directory pointers to the logon account group index block (subtree of groups in logon account). One of these pointers can be passed to the directory routines to bypass one of two levels of the search.

The parameters serve these purposes as follows:

TYPE. (10:3) = ENDLEVEL = the final target level:

0 file
1 group
2 account
3 user
4 volume set definition

TYPE. (13.3) = STARTLEVEL = specification of starting subtree for search:

0 GLOBAL - Full global search: ANAME [GUNAME [FNAME]] used (depending on level). Index ignored.
1 LOGON ACCOUNT - Group or file logon account. LINKAGE'INDEXP is a mounted volume table index and directory pointer for the appropriate account group index block (cf. JIT). GUNAME [FNAME] used as required.
2 LOGON GROUP - File in logon group. LINKAGE'INDEXP (as above) locates the appropriate group file index block (cf. JIT). FNAME used.

LINKAGE'INDEX may contain an index (MVTABX), other than zero (0 implies the System Volume Set), into a table (Mounted Volume Table) that contains, among other information, directory base address and the logical device number of the target directory, and a block pointer address, as defined by TYPE.(13:3). All name parameters are 4-word arrays, right padded with blanks, containing:

ANAME	account name
GUNAME	group or user name, depending on TYPE (10:3)=3
FNAME	file name

Each is used as indicated by TYPE.(13:3).

If the parameter MVTABX is passed as an outright parameter, it is used as an index into the Mounted Volume Table for fetching a DIRBASE for the appropriate volume set. It is assumed that the volume set has already been mounted, setting up the MVTAB entry, by prior MPE routines.

4.6.2.2 INSERTION

The following uncallable intrinsic inserts an entry into any level of the directory:

```
DOUBLE PROCEDURE DIRECINSERT (TYPE, LINKAGE'INDEXP,
ANAME, GUNAME, FNAME, NTRY, MVTABX);

   VALUE TYPE, LINKAGE'INDEXP, MVTABX;
   INTEGER TYPE, MVTABX;
   DOUBLE LINKAGE'INDEXP;
   ARRAY ANAME, GUNAME, FNAME, NTRY;
   OPTION EXTERNAL, VARIABLE;
```

TYPE, LINKAGE'INDEXP, ANAME, GUNAME, FNAME are described above. NTRY is words 4 thru 'n' of the appropriate entry (see Figures 4.2.2-B,C,D,E,G,H) ['n' is the last word]. Space for index pointers (as required must be provided, but their contents are supplied by DIRECINSERT: account user index blocks and account group index blocks are established for new accounts, and group file index blocks and volume set definition index blocks are established for new groups.

The possible returns are:

```
CCE
CCL
CCG 1,2,4,5,6
```

4.6.2.3 FINDING

The following uncallable intrinsic locates and returns any directory entry:

```
DOUBLE PROCEDURE DIRECFIND (TYPE, LINKAGE'INDEXP,
ANAME, GUNAME, FNAME, NTRY);

   VALUE TYPE, LINKAGE'INDEXP;
   INTEGER TYPE;
   DOUBLE LINKAGE'INDEXP;
   ARRAY ANAME, GUNAME, FNAME, NTRY;
   OPTIONAL EXTERNAL;
```

TYPE, LINKAGE'INDEXP, ANAME, GUNAME, FNAME are described above.

NTRY must be large enough to accommodate the entire entry, which will be returned on successful completion.

The possible returns are:

```
CCE
CCL
CCG 2
```

4.6.2.4 PURGING

The following uncallable intrinsic attempts to purge the designated entry and all of its subtrees from the directory:

```
DOUBLE PROCEDURE DIRECPURGE (TYPE, LINKAGE'INDEXP,
ANAME, GUNAME, FNAME, MVTABX);

   VALUE TYPE, LINKAGE'INDEXP, MVTABX;
   INTEGER TYPE, MVTABX;
   DOUBLE LINKAGE'INDEXP;
   ARRAY ANAME, GUNAME, FNAME;
   OPTION EXTERNAL, VARIABLE;
```

All parameters are described above.

Purging proceeds as follows:

FILES - the directory entry is removed.
VSD - the directory entry is removed.
GROUPS - All files of the designated group are FDELETEd and removed from the directory. All volume set definitions created under the designated group are removed from the directory, and the group entry is removed. [In performing a DIRECPURGE of a group, the follwing filespace accounting consideration is made: the Number of sectors released via FDELETE are accumulated and subtracted from (the group entry, which remains if any of its files are in use, and) the account entry. This is the only content-dependent action performed by any of the uncallable intrinsics in this section.]

USERS - The entry is removed.
ACCOUNTS - All users and groups of the designated account entry are removed, unless any of the groups and users were "in-use".

Certain conditions will preclude the removal of an entry (e.g., access counts not zero). No father node will be removed. The intrinsic obtains the File SIR (before the directory SIR), when purging account, group or file entries.

The possible returns are:

CCE
CCL
CCG 2,7

4.6.2.5 SCANNING/UPDATING

For background information about the following routine, refer to the discussion "DIRECTORY-OVERVIEW". The following routine transverses a designated subtree in preorder, visiting (by invoking a parameter-procedure) each entry encountered.

```
DOUBLE PROCEDURE DIRECSCAN (TYPE, LINKAGE'INDEXP,
ANAME, GUNAME, FNAME, RECIP, PARMS, MVTABX);

  VALUE TYPE, LINKAGE'INDEXP, MVTABX;
  INTEGER TYPE, MVTABX;
  DOUBLE LINKAGE'INDEXP;
  ARRAY ANAME, GUNAME, FNAME, PARMS;
  INTEGER PROCEDURE RECIP;
  OPTION EXTERNAL, VARIABLE;
```

The first five parameters are used to define the subtree desired as will be described later. The second to last two are used for the "visit" of a node.

For each target entry hit in the designated subtree, the following procedure, provided by the caller, will be invoked:

```
INTEGER PROCEDURE RECIP (NTRY, LEVEL, PARMSDISPL,
SIRS);

  VALUE LEVEL, PARMSDISPL, SIRS;
  ARRAY NTRY;
  INTEGER LEVEL, PARMSDISPL;
  DOUBLE SIRS;
```

The procedure is passed to DIRECSCAN via the RECIP parameter. It is invoked with DB at the directory data segment (DST #20) and the directory SIR locked. NTRY is a pointer to the current directory entry being visited (note that this is a pointer into the DDS); and LEVEL indicates the kind of entry:

```
0 file
1 group
2 account
3 user
4 volume set defintion
```

PARMS - as passed to DIRECSCAN - is an array of information relevant only to the caller's application and will be passed to RECIP by DIRECSCAN. It can contain parameters required by RECIP as well as space for "OWN variables" of RECIP. None of its contents is pertinent to DIRECSCAN. Since PARMS is a stack array, it is "passed" to RECIP as a negative, DIRECSCAN, Q-relative displacement, PARMSDISPL. RECIP must access the first element of PARMS from the DDS.

For example:

in RECIP:

```
        .
        .
        .
    INTEGER ARRAY
        ARRZ0 (*) = Q+0,
        ARRZ1 (*) = Q+1;
    INTEGER
        DELTAQ = Q+0;
        .
        .
        .
    PARMSDISPL := PARMSDISPL -DELTAQ;
        .
        .
        .
    TOS := ARRQ0 (PARMSDISPL);
    TOS := ARRQ0
    TOS := ARRQ1 (PARMSDISPL);
        .
        .
        .
```

The directory can be released* (unless the DDS is changed) by RECIP, using SIRS as follows:

```
    TOS := SIRS;
    RELSIR (*,*);
```

RECIP can simply examine NTRY (e.g. to list it), can alter NTRY, or alter any part of the DDS*, provided of course, that the contents of the DDS remain consistent. [Refer to the DDS description in "DIRECTORY-OVERVIEW" for definitions of the references that follow.]

RECIP can make use of WORKAREA, for its own needs.

RECIP can change AREA A and/or AREA A DESCRIPTION*, but must set DB+145 ["DADIRTY"] to TRUE if either is changed. [NTRY will always be in AREA when RECIP is called.]

RECIP can change AREA B and/or AREA B DESCRIPTION*, but must set DB+167 ["DBDIRTY"] to TRUE if either is changed. [The index block for the subtree containing NTRY will always be in AREA B when RECIP is called.]

RECIP can invoke any directory routines, from the stack. [The caller must respect all SIR orders, however.]

RECIP can, in general, avail itself of any information in the DDS; e.g. DYNAMIC GLOBAL AREA and STATIC GLOBAL AREA.

[RECIP is not as unrestricted as indicated above for certain, "bracketted" visits of particular DIRECSCAN invocations, specified below.] RECIP cannot release the directory (using SIRS) if it alters the DDS, but should release it if performing actions possibly causing an indefinite wait (eg. non-disc I/O, GENMSG, etc.). RECIP returns an indication to DIRECSCAN of how to proceed with the traversal, and also an indication of whether the directory was unlocked or not:

RECIP.(15:1) = SIR action
- 0 directory SIR was released
- 1 directory SIR was not released

RECIP.(13:2) = scan continuation
- 0 continue traversal
- 1 skip this entry's subtrees (i.e. visit the entry's "brother" next)
- 2 stop traversal (i.e. return to DIRCSCAN caller).
  At the file entry level, this will return you to the next group (if search is for "@"). If you specify "2" in the RECIP return at the group level, then you will be returned to the next account. If you specify "2" at the account level, you would be returned to the caller.

RECIP should not re-acquire the directory SIR once it is released. RECIP should release all resources it has acquired before returning to DIRECSCAN. DB must be at the DDS on return.

It is permissable for RECIP to call other directory routines, but it must be remembered the contents of the DDS could (most likely would) change as a result of these call (i.e., the contents pointed to by NTRY could be changed between references to NTRY if either another directory were called or if the directory SIR was released).

DIRECSCAN Subtree Definition Parameters

In order to specify the subtree desired, it is necessary to supply:

a. Domain - global, logon group, logon account.
b. Subtree root - e.g. an account; all groups of an account etc.
c. Leaf level.

TYPE is extended to supply this information. [NOTE: in the previous uncallable intrinsic, only domain and one node had to be supplied.] TYPE specifies the following request to DIRECSCAN.

"VISIT <tolevel> OF [<allflag>]/endlevel> STARTING AT <startlevel>" where

<startlevel> = TYPE.(13:3)
0 System root.
1 Account group index (supplied)
2 Group file index (supplied)
<endlevel> = TYPE.(10:3)
0 Files
1 Groups
2 Accounts
3 Users
4 Volume set definitions
<allflag> = TYPE.(9:1) meaning "all of <endlevel>"
<tolevel> = TYPE.(6:3) (leaflevel)
0 Files
1 Groups
2 Accounts
3 Users
4 Volume set definitions

In addition, <hitflag> (TYPE.(5:1)) indicates that every node encounted (ie. those hit in finding the root) should also be visited.

The preceding table shows, in brackets, the additional entries visited when TYPE.(5:1) is set. RECIP can only examine or alter NTRY for these visits (setting DADIRTY if altered) and must leave the DDS otherwise in tact. An extra GETSIR is performed before calling RECIP for these entries so that an attempt to RELSIR will be ineffective.

The table is simply a summary of the effects of TYPE extended to include <allflag>, <tolevel> and <hitflag>. It is useful when the desired subtree is pre-determined. However, the formulation of TYPE by sub-fields may be necessary if the subtree is not known before hand (e.g. user specification). The reader is referred to the command interpreter procedure, PRODUCEPARMS for a routine which analyzes a subtree specification and forms the appropriate DIRECSCAN parameters.

The possible returns from DIRSCAN are:

```
CCE
CCL
CCG 2
```

CCE will be returned if the tree is found. This includes a null tree (never invoking RECIP) or a traversal stopped by RECIP.

4.6.3 SPECIAL FILE ENTRY MANIPULATIONS

The following uncallable intrinsics have been provided to account for special considerations for:

a. permanent file space accounting
b. file security

All these intrinsics apply to files (actually directory file entries), and three of them always use global domains. For these last three, the TYPE and LINKAGE'INDEXP - which are 0, 0D implicitly - have been replaced with NUMSECTS (a double), a variable used for file space accounting, and a dummy variable DUMMY (an integer) for maintaining consistancy in the number and order of the required (formal to actual) parmeters.

To minimize the possibility of the directory containing a file entry pointing to garbage, the file should be fully created on disc before DIRECINSERTFILEing and removed from disc after DIRECPURGEFILEing.

4.6.3.1 INSERTION

The following uncallable intrinsic is used to insert a file entry into the directory:

```
DOUBLE PROCEDURE DIRECTINSERTFILE (NUMSECTS, DUMMY,
AN, GN, FN, FADDR, MVTABX);

    VALUE NUMSECTS, DUMMY, FADDR, MVTABX;
    DOUBLE NUMSECTS, FADDR;
    INTEGER DUMMY, MVTABX;
    ARRAY AN, GN, FN;
    OPTION EXTERNAL, VARIABLE;
```

NUMSECTS is a positive double integer containing the number of sectors that are allocated and are to be accounted for.

DUMMY is a variable used to satisfy the normal calling sequence to DIRSTARTOFF;

AN, GUN, FN are the account name, group name, and file name for the insertion. Note that they must be supplied (even for logon group/account).

FADDR is a double contianing the VTAB index in (0:8) of the first word, and the sector address in the remainder of FADDR.

The routine increments the account and group file space counts by NUMSECTS; checks the counts against the limits, ensures that the user has SAVE access to the group, and (if no errors exist) inserts FADDR in the directory.

Possible returns are:

```
CCE
CCL
CCG 1, 2, 3, 4, 5, 6
```

4.6.3.2 FINDING

The following uncallable intrinsic returns a file address and file securities:

```
DOUBLE PROCEDURE DIRECFINDFILE (TYPE, LINKAGE'INDEXP,
                        AN, GN, FRETURN, MVTABX);

    VALUE TYPE, LINKAGE'INDEXP, MVTABX;
    INTEGER TYPE, MVTABX;
    DOUBLE LINKAGE'INDEXP;
    ARRAY, AN, GN, FN, FRETURN;
    OPTION EXTERNAL, VARIABLE;
```

The first five parameters have the same meaning and usage as described in DIRECFIND.

FRETURN will contain the following on a successful return:

```
  **************
0 * FILE       *
1 * POINTER    *    2
4 * ACCT SECURITY*
  **************
```

The numbers on the right show the amount returned depending on TYPE.(13:3), the search domain. In other words, each account/group encountered will be used for FRETURN. These securities are to be used along with the file security to determine the user's access to the file, using ACCCHECK. [If FRETURN (2:4) is initialized to the logon group and logon accuount securitites (c.f. JIT) then the true group/account securities for this access will be found in FRETURN (2:4) on return from DIRECFINDFILE.]

This intrinsic makes no other provisions for security or filespace accounting. The possible returns are:

```
CCE
CCL
CCG 2
```

4.6.3.3 PURGING

The following intrinsic is used to remove a file entry from the directory and decrease the account/group file-space counts by the sectors used for the file:

```
DOUBLE PROCEDURE DIRECPURGEFILE (NUMSECTS, DUMMY,
AN, GN, FN, MVTABX);

    VALUE NUMSECTS, DUMMY, MVTABX;
    DOUBLE NUMSECTS;
    INTEGER DUMMY, MVTABX;
    ARRAY AN, GN, FN;
    OPTION EXTERNAL, VARIABLE;
```

NUMSECTS is negative double integer containing the number of sectors that will be released.

DUMMY is used as described under the discussion under DIRECINSERTFILE (above).

AN, GN and FN are the account name, group name and user name all of which must be supplied and existent.

To use this intrinsic, it is necessary to have DIRECFINDFILEd the file at some time. NUMSECTS, derived from the file label, is passed to DIRECSCAN in order to decrease the account and group filespace counts. No provision for file security is needed, since the caller has supposedly already determined that the user has WRITE access to the file.

The possible returns are:

```
CCE
CCL
```

[No CCG return is applicable between AN, GN, FN must exist since the file exists, and the caller has accounted for file security and in-use.]

4.6.3.4 EXTENT ADDITION AND REMOVAL

The following intrinsic is used to adjust the account and group filespace counts, intended for extent addition and removal:

```
DOUBLE PROCEDURE DIRECADJUST (NUMSECTS, DUMMY AN,
GN, MVATEX);

    VALUE NUMSECTS, DUMMY, MVTAEX;
    DOUBLE NUMSECTS;
    ARRAY AN, GN;
    OPTION EXTERNAL, VARIABLE;
```

NUMSECTS is the positive/negative sector adjustment to be applied against the account and group;

DUMMY same as above.

AN and GN are the account name and group name.

This routine simply adjusts and checks the filespace counts of the indicated account and group. Again, AN and GN are assumed to be existent since they are derived from some existent file.

The possible returns are:

CCE
CCL
CCG 8

4.6.4 LOGON and LOGOFF

It is necessary to perform some directory manipulations for logon and logoff that would be rather cumbersome and inefficient using only the above uncallable intrinsics. For this reason, the folloiwng intrinsics were implemented. It was noted earlier that in order to prevent purging of users, accounts and groups that were logged-on (providing fast-access group and account file domains) certain counts were maintained. The following two intrinsics adjust these counts for the designated user/account/group, and perform other actions necessary for logon and logoff.

The following intrinsic

1. Finds and returns to pre-defined locations:
   a. the user entry to stack DB+30
   b. the account entry to stack DB+50
   c. the group entry to stack DB+80

2. Increments (to prevent purging):
   a. the user entry logon count
   b. the account group index block pointer count
   c. the group file index block pointer count

This procedure is called by NURSERY, which also establish-the JIT pointers to the Group/File indices for quicker search for files.

```
INTEGER PROCEDURE DIRECLOGON (MASK, JMATENTRY, CONTIME,
                  CPUTIME, AENTRY, UENTRY, GENTRY);

    VALUE MASK, CONTIME, CPUTIME;
    INTEGER MASK;
    ARRAY JMATENTRY, AENTRY, UENTRY, GENTRY;
    DOUBLE CONTIME, CPUTIME;
    OPTION EXTERNAL:

    INPUT PARAMETERS:
       <MASK>
          LOGON - MUST BE 0.
          LOGOFF
             = 0 ACCT/USER/GROUP EXIST,
             = 1 ACCT/USER EXIST, NO GROUP,
             = 2 NO ACCT,
             = 3 ACCT EXISTS, NO USER,
             = 4 ACCT/USER EXIST, NO HOME GROUP SPEC.,
       <JMATENTRY> THE FULL JMATENTRY IN STACK.
          USED TO GET THE GROUP AND USER NAMES.
       <CONTIME> AND <CPUTIME>
          LOGON - IGNORED,
          LOGOFF - TIMES USED FOR UPDATE (IF MASK = 0).

Returns:  (Condition code unchanged)
   DIRECLOGON=

   0 fully successful (all counts incremented)
```

1 group does not exist (only user entry logon count incremented)
2 user (but not group) does not exist (no counts incremented)
3 account exist, but not user (no accounts incremented)
4 account/user exist, but no home group specified

For logoff, the next intrinsic performs the following actions as directed by the caller's indication of which entries were found on a logon (attempt).

The following intrinsic

1. Finds the account entry, user entry, and group entry
2. Decrements
   a. the user entry logon count
   b. the account group index block pointer count
   c. the group file index block pointer count
3. Updates and checks the account and group connect and CPU times.

This procedure is called by MORGUE.

```
LOGICAL PROCEDURE DIRECLOGOFF (MASK, JMATENTRY,
CONTIME, CPUTIME,);

    VALUE MASK, CONTIME, CPUTIME;
    INTEGER MASK;
    ARRAY UMATENTRY;
    DOUBLE CONTIME, CPUTIME;
    OPTION EXTERNAL;
```

MASK is the integer returned by DIRECLOGON. JMATENTRY is a stack image of the pertinent JMAT entry.

CONTIME and CPUTIME are (if MASK=0) the connect time and CPU time.

Returns:
If MASK <> 0 then 0 If MASK = 0 then

(15:1) Account connect time exceeded
(14:1) Account CPU time exceeded
(13:1) Group connect time exceeded
(12:1) Group CPU time exceeded

[Note that the time counts are always incremented, and the "time exceeded" indications are just warnings.]

4.7 IMPLEMENTATION

The following discussions concerns the actual implementation of the directory routines. Overviews of the basis algorithms used have already been given, and much of the actual details will therefore be self-evident from the listings directly. This discussion, then, serves as a guide to the listings, and the listings should be used in conjunction with the following descriptions.

4.7.1 NAMING CONVENTIONS

Being uncallable intrinsics (i.e. procedures), the directory routines allocate no global storage. They do, however, make use of global storage available to them in the DIRECTORY DATA SEGMENT (DDS), and the DIRECTORY SPACE DATA SEGMENT. These two data segments are first initialized by the configurator, and the information is subsequently maintained by the directory routines. The information of these data segments is assumed to be consistent, and no special checks for garbage data is made (in the data segments and implicitly in the directory as a whole).

The first pages of the listing contain definitions of entry and index content displacements and fields; some system global information; definitions of the DDS and DIRECTORY SPACE DATA SEGMENT (DIRSPACE) contents; flags and flag fields for the directory routines; and miscellaneous stack declarations. The descriptions that follow will present the naming conventions employed. The next section will describe the meanings of some of the variables.

Word displacements into elements are denoted by prefixing the content identifier by a letter representing the element type: A, G, F, U, I (index), Element sizes are EQUATED to identifiers named "xSIZE". Field definitions are suffixed by "...F", such as xPURGEFLAGF. The content displacements of index block prefixes are prefixed by "PRE...".

Each level of the directory is represented by a unique number that is used consistently:

0 Files.
1 Groups.
2 Accounts.
3 Users.
4 Volume set definitions.

Refer to the diagram given earlier in this Section of the DDS and to Figure 4.3-C and 4.3-D for definitions of the areas and variables described below. The DDS contains two pairs of "twin" areas: AREA A/AREA A DESCRIPTION and AREA B/AREA B DESCRIPTION. The two description areas contain the exact same kind of information, positioned in the same relative order. The displacements from the beginning of

the description areas for both areas are defined by a common set of EQUATES. The names were chosen to represent the contents corresponding word. Each description area is (not coincidentally) entirely directly addressable. These directly-addressable variables are defined for each area in terms of these displacements. AREA A DESCRIPTION variables are preceded by "DA...", and AREA B DESCRIPTION variables are preceded by "DB...". Those directory routines which are generalized to operate on either area use displacements relative to BASE, a variable setup to point to the pertinent area. Where possible, of course, the directly-addressable variable is used. This usually is the case when the target is an index (must be in B) or an entry known to be in A (entries appear in B only when splitting blocks on insertion)). When necessary, AREA A is represented by 0 and AREA B is represented by 1.

[When possible, each content displacement is defined by reference to the area displacement, so that variable insertions/deletions would be easier.]

Procedures have been named according to a loosely-defined convention regarding their "internalness". Specifically, the most basic procedures are names "D..."; the next higher level procedures are named "DIR..."; and only uncallable intrinsics are named "DIREC...".

4.7.2 DATA STRUCTURE

An overview of the basic data structure used has already been given. The following description augments the listings and gives some of the implementation details regarding the data structure. Only those variables which are not already self-evident will be described.

4.7.2.1 DIRECTORY BLOCKS

The structure of the directory on the disc has been described in a previous subsection, where it was stated that the size of directory blocks is not constant. Otherwise, the standard block concept was used, in which each block simply is the physical unit of transfer and contains an integral number of entries (c.f. records). A minor exception to this is index blocks which contain a prefix.

The prefix is very important. A diagram of an index block prefix appears in Figure 4.2.2-A. It contains two words which describe the characteristics of this index block, and all of its entry blocks [MISCWD and EMISCWD]: level, element size, and block size. The high order bit of these words indicated whether this applies to an index block (1) or an entry block (0). The only other piece of information that is necessary to fully characterize the block is the number of entries that it contains: for index blocks, this is in the prefix, for entry blocks this is in the entry block's index.

Note that the "MISCWD" word and element count fully characterize a block. [Note also that not only can blocks at different levels be of different size, but they can be different even on the same level.] The only restriction is that all entry blocks of an index block be of the same size.

Additional information appearing in the index prefix is the pointer count (indicating the number of accessors relying on a pointer to the index block), a count of the total number of entries, and the index pointer and name of the "father" of this index block. It will be shown later that an index pointer and name are sufficient to locate any entry in the directory.

The indices immediately follow the index block prefix in the index block, and are diagrammed in Figure 4.2.2-A.

4.7.2.2 DIRECTORY SPACE DATA SEGMENT

The directory space data segment (DIRSPACE) is shown in Figure 4.3-B. The home address of the directory that the data segment represents is maintained in the data segment itself. This address is maintained in the form of a double word containing the logical device number and directory base disk address of the area's parent

directory. The address is initialized whenever the the data is read into the data segment. This address is then used as the target address whenever the data segment's content is changed. This mechanism allows for the maintenance of space maps of any number of directories.

Directory space is monitored using the bitmap in DIRSPACE which lies between word 4 and LASTWORD (inclusively). FIRSTWORD is a cycling pointer that points to the first word to examine for availability. The word pointer to by LASTWORD must be followed by a word of zeros. The bitmap shows allocation of the entire directory disc area: word (0).(0:1) represents the first sector (0), word(1).(1:1) represents sector 17, etc. Because DIRSPACE is the first three sectors of the directory disc area and the system root index is the first allocated and must be 3 sectors, word(0).(0:5) must be zero.

4.7.2.3 DIRECTORY DATA SEGMENT (DDS)

A diagram of the DDS is shown in Figure 4.3-C. Its basic purpose is to serve as a buffer and working area for the directory management routines. A breakdown into its major areas and their use has been described above. Following is a more detailed, implementation-oriented description of the contents of this data segment.

The first 128 words are used in addition to a miscellaneous work area as the area in which stack arrays are moved, for passing to and from integral directory routines. These arrays include names and/or whole entries.

The next area, the "Dynamic Global Area", is set up on every call to a directory uncallable intrinsic. It contains information pertinent to the particular call and useful or necessary information that can be used by internal routines or to correctly complete the request. This includes (DB-DL) for array moves, an exact copy of the first 5 parameters to the uncallable intrinsic (with INDEXP adjusted to the final target index), security matrices that are initialized and updated as each account/group is encountered, and the return value from the initial GETSIR (DIRSIR).

As was described earlier, the DDS contains two buffers, A and B, - and their corresponding "description" areas. The description areas fully characterize the contents of the buffers. The meanings of the information contained is evident from the descriptions in Figure 4.3-C. These description areas are used for both index blocks and entry blocks. Index blocks, however, make use of an "extended" description area to contain information derived from the index block prefix. When an entry block is read into the DDS, the index block prefix "miscwd", EMISCWD, is used to set the MISCWD in the appropriate

description area for the entry block.

Following the two description areas are the "static global area" containing invariant global information and the two buffers (separated by an "overrun area").

4.7.3 SPACE MANAGEMENT

DIRXXXBITMAP simply updates the disc copy of the directory space bitmap, if it is "dirty", by writing out the current DIRSPACE bitmap onto the directory disc area. This is done whenever the bitmap is changed, due to an allocation/deallocation.

DIRXXXLLOCATE is a utility procedure used by directory allocation or deallocation which simply changes PPSIZE pages starting at PNTRIN to SETTO [i.e. changes the bits in the bit map corresponding to those pages to 0 (allocate) or 1 (deallocate)]. It SUDDENDEATHs if it detects an allocation of an allocated page, or a deallocation of a deallocated page. It marks the data segment "dirty" before returning tho the caller. It is the caller's responsibility to update the bitmap on disk, when appropriate, by calling DIRXXXBITMAP with a function of "write".

DIRALLOCATE attempts to allocate PPSIZE contiguous pages, and (if successful) returns the directory pointer of the first page. The CC returns are:

CCE - Successful allocation. Pointer returned.
CCL - Can't find PPSIZE contiguous pages. 0 returned.
CCG - PPSIZE is too large for the DDS as configured. 0 returned.

The subroutine FIND searches between the words of the bitmap delimited by LOWLIN and UPLIM inclusive. It does this by examining two words at a time, and therefore may examine UPLIM+1 (this is why LASTWORD must be followed by a zero). In examining the bits, it keeps track of the number of acceptable (i.e. contiguous) pages in SIZE, resetting it whenever it comes across an allocated page (0). If it finds PPSIZE contiguous pages, it updates FIRSTAVAIL in DIRSPACE (to point to the word containing the newly allocated pages, allocates them, using DIRXXLLOCATE), and returns to the caller of DIRALLOCATE.

A return to the procedure body implies a failure to find the pages between (and indcluding) LOWLIM and UPLIM.

The procedure body of DIRALLOCATE uses FIND to examine, first, the bitmap following FIRSTAVAIL, and, if that fails, the portion before FIRSTAVAIL.

DIRDEALLOCATE simply calls DIRXXXLLOCATE to set the appropriate bits back to 1. The PPSIZE pages starting at PNTR must already have been allocated.

4.7.4 PRIMITIVE UTILITIES

DIRWRITE writes out the blocks contained in WHICH [WHICH defines the buffer area in question: 0=A (entry block), 1=B (index block)]. The target address for the write is derived from the combinition of the appropriate area's (either A or B) directory base and logical device number and directory base relative displacement address. Each of the description areas keep track of their respective block's "source" address. Because the block is written out, the area can be considered "clean", and the dirty flag is always reset. If the block is an index block, then information in the description area relating to the prefix updates the prefix in the buffer area before the block is written. The routine SYSABORTs if any error is detected in the write (the IOCB error code can be found in the X register).

DIRREAD will perform the function of returning to the caller the requested block (either Index or Entry, depending on WHICH) in the appropriate I/O area of the Directory Data Segment. The combination of the directory base relative disk address, passed through the parameter PNTR, and the directory base address and logical device number, contained in the Dynamic Global Area of the DDS, determine the source address from which the requested directory block should be obtained. These addresses are stored in their respective description areas for reference by subsequent DIRREADs and DIRWRITES. DIRREAD reads at least one block's worth of pages into WHICH, starting at PNTR in the directory disc area. It returns immediately if the pertinent buffer already contains the requested block. If the block is "dirty", it calls DIRWRITE to update the disc. [The long sequence of code in comments was an attempt to achieve a very minor (if any) performance increase by not issuing a disc read if all the pages of the desired block are imbedded anywhere in the DDS.] The disc read is for the maximum number of pages that the DDS can hold. Again, if an I/O error is detected the routine SYSABORTs, with the IOCB error code in the X register. Once the appropriate block is read into DDS, the corresponding description area must be set up. For index blocks, this information is obtained from the index block prefix. Therefore:

> For index blocks, EXCOUNT and EEMISCWD are ignored.

The characteristics of a particular entry block are defined by IECOUNT of the entry block's index, and EMISCWD of that index's index block prefix. Therefore:

> For entry blocks, EXCOUNT is the number of entries in the entry block (derived from the entry block's index), and EEMISCWD is the word characterizing the entry block, as defined by EMISCWD of the index block prefix.

So, from the last two parameters (entry blocks) or from the index block prefix, the entire description area is initialized.

DIRNEWINDEX sets up description area B to be a null index block with the specified attributes.

IBSIZE is the index block size.
ILEVEL is the level.
EBSIZE is the entry block size.
ESIZE is the entry size.

The caller must ensure that area B is clear and can accept the initialization; and must also have moved the father name and index pointer into DBPNAME and DBPINDEXP. Once B is initialized, it is written out onto the area DIRNEWINDEX has obtained uisng DIRALLOCATE. The returns are:

CCE - okay.
CCG - IBSIZE or EBSIZE is too large for DDS as configured.
CCL - can't allocate IBSIZE contiguous pages.

DIRSCAN is the general routine that searches area A or area B for a given element:

ENTRYNAME (a misnomer) is the DDS address of the element name.

TYPE'WHICH. (13:2) is the type of request:
0 Search for exact element match.
1 If no exact match, return next element.
2 If no exact macth, return preceding element.
TYPE'WHICH. (15:1) is the area to search:
0 A.
1 B.
[All combinations of TYPE'WHICH are symbolically defined, globally].

A DDS pointer is returned:

CCG - Exact element returned.
CCL - Next [preceding element returned.
CCE - No exact or next [preceding] element found. Pointer to "pseudo element," beyond last [before first] returned.

Currently, a simple linear search is made for the element exactly matching, or the "lowest" element "greater" than the target, ENTRYNAME. That will be the exact hit, or the place where the target, logically, falls. [Some improvement might be obtained by simply changing the implementation of this routine to a binary search.]

4.7.5 MANAGEMENT PRIMITIVES

4.7.5.1 INSERTION

DIRINSERT is the basic routine that inserts an entry into a given subtree. The subtree or directory page into which the entry is to be inserted is specified by the parameter, INDEXPOINTER. When this procedure is called, the entry to be added must be located in the DDS at DB+0 (with DB set to the DDS). OD return is successful. If failure occurs, a CCG return of 0, 1, 4, 5, 6 is given.

ZINSERT is the subroutine which does the actual "insert and spread" of a specified index or entry block. It puts the entire ELEMENT into the location PNTR (if it is non-zero) in area WHICH, moving all the elements following PNTR one element width to make room for the new one. If PNTR is zero, ZINSERT will determine PNTR. Note that the caller of ZINSERT has ensured that the element will fit and found the place for it at PNTR (when not zero). When PNTR is zero, ZINSERT utilizes DIRSCAN to find the point where ELEMENT should be inserted (by looking for exact/next). ZINSERT returns the pointer where ELEMENT was inserted (as supplied or determined by ZINSERT).

ZNEWNTRYBLOCK is the subroutine which allocates a new entry block and creates the proper index pointer entry for it. The name of the first entry of the block (used for the index reference) is at NAME. INDEXPLACE (if non zero) is the pointer (in area B) where the new index should be placed. If INDEXPLACE is zero, then the new place is determined in the routine. If successful, the routine returns the pointer for the new entry. If some failure occurs, the routine returns directly to the caller of DIRINSERT with the proper error indication.

ZSET is the subroutine which initializes the following variables in anticipation of a "distribution". On Entry to this routine, ZT is the total number of entries (including the new one) involved in the distribution:

ZTOTAL := ZT * entrysize
ZH1 and ZH2 are a division of ZT into halves.
(if ZT is odd, ZH2 will be the greater one)
ZHALF1 and ZHALF2 are the word equivalents of ZH1 and ZH2.

ZDISTRIBUTE is the routine which divides an overflowing entry block in area A to equal entry blocks using both area A and B. It then writes these new entry blocks out to the directory on the disc.

The main procedure body follows the algorithm described earlier for insertion. It first reads in the designated index block, makes a check that there are fewer than 65K entries in the subtree, and tries to locate the entry by using DIRSCAN on the index block, looking for an exact/preceding hit. If there is no such block, it will consider the first entry block for the insertion. If no entry block exist at all, it will create a new one using ZNEWENTRYBLOCK and insert the new entry into the new entry target block in the normal fashion.

Once an entry block has been determined, a test is made to see if the entry will fit into the entry block. If it will, the entry block is read in, the entry is inserted (using ZINSERT), some variables are updated to reflect the new entry, the name in the index is changed (in case the new entry is the first of the block), and the entry block and index block are written out.

If the entry doesn't fit into the entry block, then a distribution with an existing neighbor block, or distribution with a newly-allocated entry block must occur. It will be distributed with a neighbor if one exists and if the total number of entries of the two blocks (plus the new entry) is less than the GOODPERCENT of the total number of entries that can be accommodated. [If there is only one neighbor, it is chosen; if there are 2, the one with the lower number of entries is chosen.] NOTE: at this point, no entry block has been read yet, all calculations were from the entry counts of the indices.

If a distribution of two existing blocks is attempted, the lower one is read into area A, and the higher one is read into the DDS the address immediately following the last entry of the lower block. This results in one, overflowing entry block in A, into which the new entry is to be inserted. Area B may be destroyed by this operation (if it isn't, it will be by the following distribution). After kluging some area A description variables so that the entire (overflowing) block is considered, the new entry is inserted with ZINSERT. Assuming no duplicate entry exists, the whole mess is now distributed into entry blocks in both area A and area B by ZDISTRIBUTE, which writes out the two entry blocks; the indexes are corrected and the index block is written out.

When distribution with a new block is required, the existing entry block is read into area A; space for a new entry block, and a new index, is established by ZNEWENTRYBLOCK; the index is updated and the index block written out. A null entry block representing the newly-allocated one is set up in area B. The existing entry block is "overflowed" by a ZINSERT of the new entry, but it is distributed between the old and new block by ZDISTRIBUTE (writing out both entry blocks). The index block has already been updated and written out.

4.7.5.2 FINDING

DIRFIND will locate an entry of a specified subtree. The subtree is represented by the index block pointer, INDEXPOINTER, and the entry name is expected to be found in DDS DB+0 through DB+3. If not found, 0D is returned, otherwise,

(S-0) = Address of entry [always in area A].
(S-1) = Address of entry's entry block index [always in area B].

The algorithm consists simply of reading the index block and locating the containing entry block (DIRSCAN exact/preceding): reading the entry block and locating the entry.

4.7.5.3 DELETING

DIRREMOVE simply removes ELEMENT from are <WHICH>. It sets DXDIRTY, decrements DXXCOUNT, and adjust DXUSED. It removes the element by performing a move of the following entries over the target. If this is an entry block that is thereby depleted, its space is deallocated. [Remember that while entry blocks are deallocated when empty, index blocks remain as long as their parent exists.]

4.7.6 UNCALLABLE INTRINSICS

The routines described above are general procedures that operate on the data structure of the directory, ignoring the contents of the entries that they manipulate. In fact, they are generally ignorant even of the hierarchial structure, and operate only on one level of a subtree. The following procedures effect the extensions necessary for the hierarchial structure and other content-dependent considerations (i.e. permanent file space and security). Obviously, these routines make use of the "one-level" procedures described above.

Generally, the implementations of the actual uncallable intrinsics follow directly from the descriptions already given. Therefore, only comments and unusual operations will be described for the uncallable intrinsics.

4.7.6.1 TWO UTILITY PROCEDURES

DIRSTARTOFF is used by almost every uncallable intrinsic. A great part of the parameter specification of the different uncallable intrinsics is identical (TYPE, LINKAGE'INDEXP, ANAME, GUNAME, FNAME, ...). This is because these parameters esentailly serve to define one target, which is to be inserted, found or removed. Thus the setup for the operation is the same for the different intrinsics, with deviations occuring once the target is established. The specification of this target can be non-trivial (admitting to any "starting point" and any target level). The essential purpose of DIRSTARTOFF, then, is to thread through the directory hierarchy, performing some useful functions as it goes, and winding up with the final target defined by one name and one index defining the subtree.

DIRSTARTOFF analyzes the specification part (first 5 parameters) for directory uncallable intrinsics, and performs any directory accesses necessary to leave:

XLINKAGE'INDEXP (double)

XMVTABX - If<>0 then the directory base is obtained from the associated Mounted Volume Table entry.
If XMVTABX = 0 then the directory base is that of the system directory and is constructed from the contents of the system global cell %130-%131.
The determined directory base address is used to initialize the DDS variable (double) DIRBASE.

XINDEXP points to the index block pointer of the target entry, and DB+0 thru DB+3 to the target name.

In addition, the following DDS variables are initialized:

ADJUST to the stack (DB-DL) [for moves];
XTYPE to the TYPE parameter:
XANAME, XGUNAME and XFNAME to the corresponding stack addresses;
XASEC and XGSEC to the secutiry matrices of the account/group as (if) encountered: -1, -1D indicate not accountered; and
SIRRETURN to the initial GETSIR (DIRSIR), performed

Input parameters:

PARR is stack address of intrinsic's parameters:
NUMSECTS, if specified, is added to account and group filespace counts.
RECIP and PARRS, if specified, causes each entry actually encountered, to be visited by RECIP (passing in PARMS), like DIRECSCAN; and if only PARMS specified, then S access to group is verified.

This procedure returns 0D, if successful, or a double word identical to the CCG returns for the uncallable intrinsics described earlier.

Implementation follows directly from the description given above. Basically, it consists of an initialization portion, followed by a switch transferring control to the code necessary to handle the "start" level of TYPE.(13:3). Each level automatically drops to the lower level, until the "end" level is hit. At each level, the following processing is performed:

1. Actually find the relevant entry,
2. Check for save access,
3. Bump the filespace count,
4. Visit the entry,
5. And set XINDEXP to the next "subtree".

DIRRESET is used by routines when they determine that the filespace counts of the [group [and account]] of the current access need to be adjusted, by NUMSECTS. Using DBPINDEXP and DBNAME, it traces backward, until the root is reached.

4.7.6.2 INSERTING

DIRECINSERT performs a DIRSTARTOFF and - once index blocks are allocated for groups and users (accounts) or files (groups) - does a DIRINSERT of the entry. If any error is detected, any index blocks allocated must be deallocated.

DIRECINSERTFILE calls DIRSTARTOFF (with the options of incrementing filespace counts and checking save access) and then performs a simple DIRINSERT of the directory file pointer entry, as supplied by the caller.

4.7.6.3 FINDING

DIRECFIND performs a simple DIRFIND, after the target is positioned by a standard call to DIRSTARTOFF.

DIRECTFINDFILE performs the same actions that DIRECFIND does, except that the file pointer and file security (possibly) are returned (instead of the entire entry). Both group and account, just group, or no file securities are returned depending on "startlevel" as described in the previous definition.

4.7.6.4 PURGING

As described earlier, the algorithm for purging consists of scanning the target subtree in endorder, deleting a node only if the following conventions: each returns a double representing the total number of file sectors removed (as returned by FDELETE), with CARRY set if the entire entry/subtree has been removed. The purge procedures are divided into: a procedure which removes a subtree, five procedures designed to remove entries, and the actual uncallable intrinsic procedure. As might be expected, the routines for the "higher" levels make use of the "lower" level routines, possibly causing several layers of recursion: the uncallable intrinsic calls an entry purge procedure, which may call the purge subtree procedure, which in turn will call an entry purge procedure, which may again call the purge subtree procedure, and so on.

DIRPURGESCAN is the "purge subtree" procedure. The index block for the target subtree is located in area B on entry, and the procedure is passed the appropriate "entry purge" procedure in PURGER. It simply scans B's index block, reading in each entry block and attempting to purge each entry using PURGER. If an entry block becomes depleted, the index is removed (the entry block has been deallocated by DIRREMOVE called from PURGER. Otherwise DIRPURGESCAN updates the directory by writing out each non-empty entry block. Before returning, area A is restored to what it had on entry. The (possibly empty) index block is left in area B.

The following five entry purge routines have the following in common: they require the entry to be purged to be in area A and pointed to by NTRY, and if area B must be destroyed by the purge operation for the entry, it must be restored by the "entry purge" routine before exiting. [The double result and CARRY conventions are still obeyed.] Each entry is removed by DIRREMOVE (which will set DADIRTY of deallocate the entry block when depleted), or if it cannot be removed because it is in use, DADIRTY is set if the entry is at all changed [e.g. setting a (currently unused) "purge" flag].

DDELVSD simpley purges the volume set definition entry.

DDELFILE determines the disc LDN from the VTAB and calls FDELETE.

DDELUSER purges the user entry.

DDEGROUP saves the pointer for B, invokes DIRPURGESCAN on the group entry's group file index block, and removes the entry or flags it depending on the result from DIRPURGESCAN.

DDELACCT saves B's CONTENTS, invokes DIRPURGESCAN for account user index, invokes DIRPURGESCAN for the account group index, and either removes the account entry or flags it depending on both returns from DIRPURGESCAN.

DIREPURGE and DIRECPURGEFILE differ only slightly and are implemented by the same procedure. DIRECPURGE performs a standard DIRSTARTOFF for positioning, whereas DIRECPURGEFILE performs a DIRSTARTOFF with adjustment. In both cases, the entry is located and removed: by calling the appropriate "entry purge" routine for groups, users and accounts, or by simply DIRREMOVEing file entries (FDELETE is invoked only when purging groups and accounts). It is assumed that the file is FOPENed when purge is required and the caller is responsible for deallocating the files's disc space after return from DIRECPURGE[FILE]. If the entry is fully purged, the index is updated, or removed, and the index block is written out. If the entry was not fully removed, an indication is returned to the caller, and if it was a DIRECPURGEFILE - the filespace counts are returned to the values they were before DIRSTARTOFF adjusted them. The account's filespace count is adjusted for the files removed by a DIRECPURGE of a group.

4.7.6.5 SCANNING

IN order for DIRECSCAN to traverse the specified subtree in "preorder" as described in the preceding definition, it makes use of two procedures; one that scans a subtree, and one that actually performs the visit.

DIRSCANTREE traverses the tree specified by INDEX to the level, LEAFLEVEL, in preorder. For each entry encountered in the traversal, DIRSCANTREE invokes DIRDOENTRY (with RECIP, PARMS and GETSIRRESULT), which is to perform the actual visit.

DIRSCANTREE performs a normal scan by reading in the appropriate index block, reading in each entry block, and "hitting" each entry. But because RECIP can release the directory, the DDS and the directory may have changed. For example, perhaps the directory was released by RECIP and some entry close to the current point of traversal is removed. In other words, nothing about the directory can be guaranteed. For this reason, DIRSCANTREE uses the pointer reference count mechanism to ensure that at least the current index block will remain, and finds the "next" entry by saving the "next" name (in the stack) and performing the find using only the index block pointer (known to be existent) and a "candidate" for the next name (which may or may not exist). Like DIRDOENTRY, DIRSCANTREE adjusts PARMS so that it alwlays contains the correct Q-relative displacement for RECIP. The index is always restored after returning from DIRDOENTRY, and before leaving DIRSCANTREE the pointer reference count is decremented. Note that the use of DIRSCANTREE and DIRDOENTRY in this way results in a recursion similar to that effected by the purge routines, but purging is a traversal in endorder.

DIRDOENTRY is responsible for performing a proper visit of the entry using RECIP. Entry ELEMENT of level, LEAFLEVEL, is visited by passing PARMS (adjusted) to RECIP, as described earlier. DIRDOENTRY saves information it (or subsequent routines) need, performs the call (possibly re-locks the directory SIR), and restores some information: and either invokes DIRSCANTREE for ELEMENT's subtree or simply returns, depending on whether LEAFLEVEL has yet been reached or not. The reason DIRDOENTRY is separate from DIRSCANTREE is that either of them may be called from DIRECSCAN.

DIRECSCAN calls DIRSTARTOFF in a standard fashion, or requesting visits, depending on "hitflag", TYPE.(5:1). When DIRSTARTOFF visits entries, it is depending on DDS addresses to continue its operation. It does not perform the save/restore that the normal scan procedures do, and that is why RECIP is restricted in the actions it can perform when operating on entries visited by DIRSTARTOFF. DIRECSCAN then calls DIRSCANTREE or

DIRDOENTRY depening on whether a "pure" subtree is being requested or an entry (and its subtrees). "@" is a specification for a subtree, as opposed to an explicit entry specification [see the DIRECSCAN chart accompanying the uncallable intrinsic definition]. If returning from either routine with an unrecorded change in the DDS, the relevant block is written out.

4.8 MISCELLANEOUS

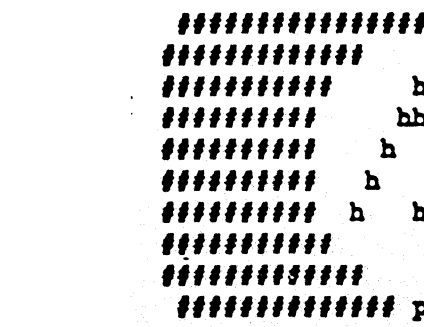
DIRECADJUST performs the requested adjustment of the filespace counts by simply calling DIRSTARTOFF with the "adjust option". Note that the specification request to DIRSTARTOFF if for "all files", so that no access is attempted for a file.

DIRECLOGON/LOGOFF, performs the actions defined earlier. Because it does not require any positioning, DIRSTARTOFF is not invoked, and DIRECLOGON/LOGOFF does any necessary initializing actions itself.

4.8.1 IMPROVEMENTS and EXTENSIONS

Retaining the basic directory structure and algorithms just described, it may be profitable to incorporate some of the following extensions. In most cases (with the possible exception of the binary search), the performance increase or additional capability must be considered a "frill" at the time of this writing: either the performance increase is marginal (if existent), or the case occurs too infrequently to justify alterations at this time.

1. Propagate an I/O error detected in DWRITEBITMAP, DIRWRITE or DIRREAD back to the caller.
2. Change DIRSCAN so that it uses a more efficient search than a linear one (e.g. binary).
3. For insertion, if two existent blocks fail the "GOODPERCENT" test but there is room for one more entry and no new entry block can be allocated, insert and distribute with existent blocks.
4. For insertion, if target entry would be the last one of a full block, try to insert in the next block.
5. For insertion, develop some scheme of distributing the entire subtree before failing because of no room.
6. Change high level routine so that an index block is not allocated until it is needed.
7. Return actual cause of the in-use failure from DIRECPURGE.
8. Change DIRSTARTOFF so that the special entry visits due to "hitflag", do not restrict RECIP actions.



H E W L E T T

P A C K A R D

COMPUTER SYSTEMS . 19447 Pruneridge Avenue, Box 17000, Cupertino, CA 95014

==========

From: Sam Yamakoshi

Date: September 28, 1981

To: DISTRIBUTION

Subject: Design Specification for Directory Expansion

The investigation review meeting for the MPE Directory Expansion project was held in the Alviso Room on September 22, 1981. A draft copy of the design specification for this project (attached) was reviewed. An external specification is also attached.

Issues of compatibility, performance, and future utility requirements were discussed. These issues are addressed in the design specifications.

The scheduled date for completion of this project is 11/30/81. As a result, unless the MIT for the HP7935 (BFD) slips beyond its target MR date, the directory expansion (phase I) will not be available until after the HP7935 has been released.

-hp-

EXTERNAL SPECIFICATION

for

DIRECTORY EXPANSION

by: Sam Yamakoshi

date: Sept. 24, 1981

The external changes to MPE with the directory expansion only occur to the disc allocation messages in SYSDUMP, VINIT, and INITIAL. In each of these programs, the dialogue will change from a 6000 sector maximum for the directory to a 65,000 sector limit.

SYSDUMP: Changing the current directory disc allocation the message will now appear as,

DIRECTORY USED = <YYYY>,MIN = <ZZZZ>,MAX = <XXXX>

where <XXXX> cannot exceed 65,000.

VINIT: During the initialization of a private volume disc pack,

ENTER DIRECTORY SIZE (SECTORS - 384 TO 65,000)?

INITIAL: Any errors messages will now reflect a 65,000 sector limit for the directory.

DESIGN SPECIFICATION

for

DIRECTORY EXPANSION

by: Sam Yamakoshi

date: Sept. 21, 1981

DESIGN SPECIFICATIONS

1. PRODUCT IDENTIFICATION

1.1 PROJECT NAME: DIRECTORY EXPANSION

1.2 DESCRIPTION: Increase the maximum size of an allowable MPE directory from (current) 6000 sectors to 65,000 sectors of disc space.

1.3 PROJECT PERSONNEL:

- 1.3.1 Project Engineer: Sam Yamakoshi
- 1.3.2 Project Manager: Bill Dalton
- 1.3.3 Product Manager: John Chisholm

2. GENERAL DESIGN OVERVIEW

2.1 DESIGN APPROACH: This will be a multi-phase implementation.

2.1.1 The first phase will be a pure expansion of the size of the directory from a current maximum of 6000 sectors up to a limit of 65,000 [one word]. Without any other changes, this would at least allow systems to reach the maximum at each node of the directory accounting structure.

2.1.2 The second phase will be a MIT update which will include a modified set of directory routines which incorporates a "link list" for the directory index nodes. The current maximum number of accounts, groups, and so on are set by the fixed size of an index block entry. If the index blocks are extendible, a system would have unlimited (other than the actual size) number of accounts, groups, etc.

2.1.3 The last phase of the directory expansion would be to expand the 65,000 sector limit to a double word entry.

2.2 MAJOR MODULES:

2.2.1 DIRC [53]: Modifications to the directory routines to handle larger bitmaps and to include the "link lists" in the search algorithms.

2.2.2 INITIAL [00]: For RELOADs with any of the directory modification operations, the directory routines in INITIAL need to be changed.

2.2.3 SYSDUMP [01]: This module will require changes to handle the increased or decreased directories.

2.2.4 SADUTIL: Include the "link list" directory search routines.

2.2.5 LISTDIR2: Changes for LISTDIR2 will also be needed for displaying the contents of an MPE directory.

2.3 MAJOR DATA STRUCTURES:

2.3.1 DISC LAYOUT: The relative position of the directory on disc will be the same (following the Free Space Table). The bitmap may, however, be from 3 to 33 sectors to handle directories of from 6000 to a maximum of 65,000 sectors. This would cause the start of the directory data portion to be a variable (but calculable) distance from DIRBASE (the pointer to the beginning).

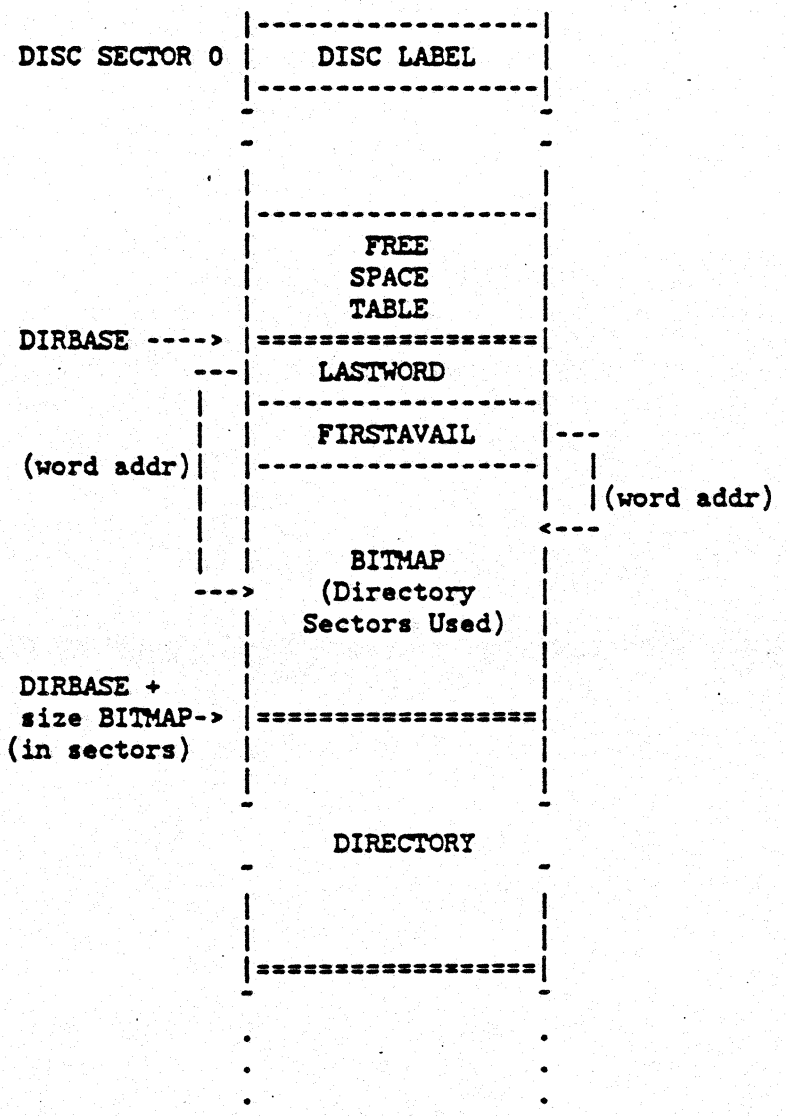


Figure 2.3.1

2.3.2 DIRECTORY SPACE DATA SEGMENT: The system directory bitmap I/O buffer [DST %25].

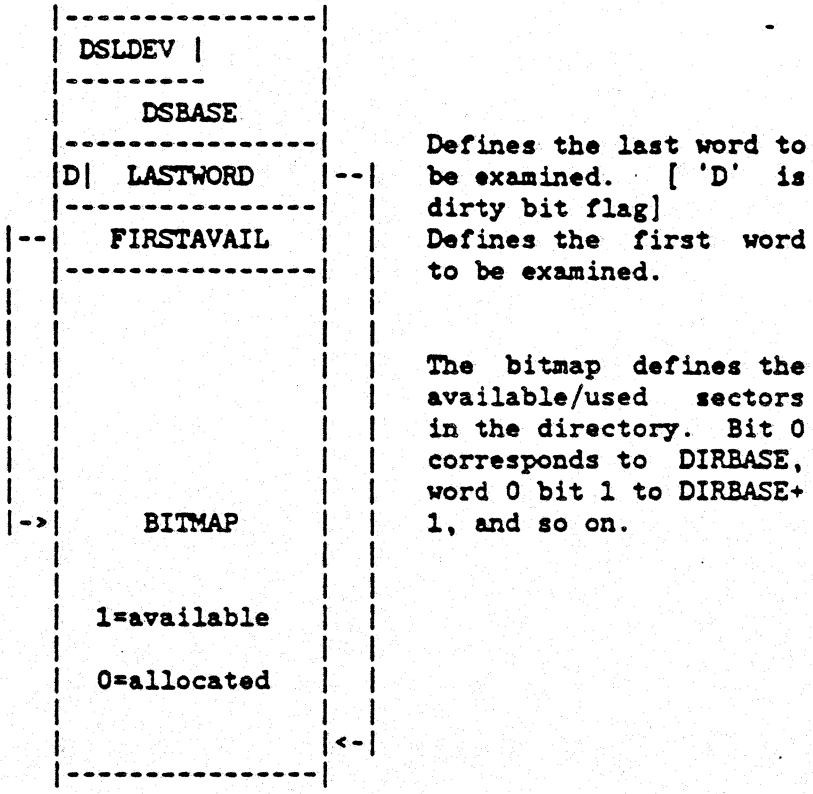


Figure 2.3.2

DIRSPSIZE (data segment size) is defined as
= SYSACCTINDEX *128
where SYSACCTINDEX is 3.

The change required here would be to words 0 and 1 of the data segment buffer. The logical device and directory base sector address are needed because both the system and private volumes utilize this segment for all directory operations. The directory address will now reflect which "page" of up to 33 sectors is the one being examined for space.

2.3.3 DIRECTORY DATA SEGMENT: The system directory I/O buffer area for directory index and entry blocks [DST %24].

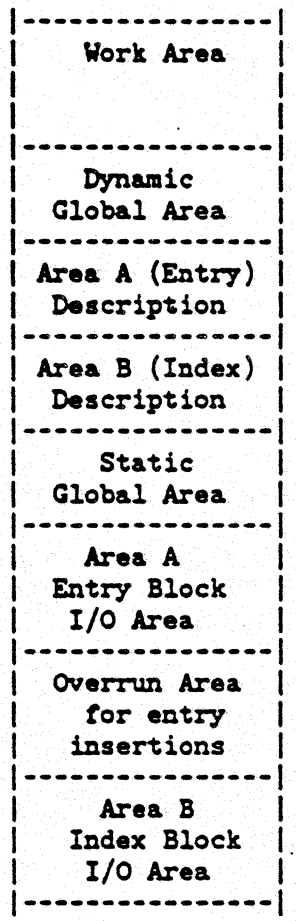


Figure 2.3.3

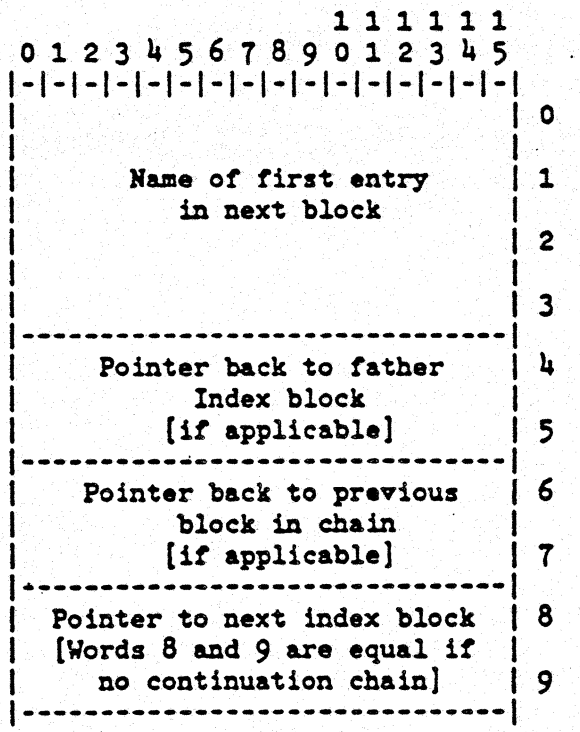
There will likely be changes made to the dynamic global (and static) cells to implement the new search algorithms.

2.3.4 INDEX BLOCK ENTRIES: There is at each node of the directory, index entries which contain the first entry name within an entry block and a pointer to that block. This hierarchical key structure shortens the search for a target entry to only the block that should contain the entry.

2.3.4.1 CURRENT INDEX BLOCK SIZES: Listed below are the index blocks, the number of 6 word entries per block, the number of sectors and words used per block, and then the number of unused words (of the entire index block) which will be used for "link list" chaining of indices.

Index Type	Blk	#wds/#sect	#unused
System Account	62	372/ 3	12
Account User	19	114/ 1	14
Account Group	19	114/ 1	14
Group File	41	246/ 2	10
Group VSD	19	114/ 1	14

2.3.4.2 "LINK LIST" FIELD: The last ten words of each index block will be used to chain links forward and backward.



2.4 PERFORMMANCE CONSIDERATION:

2.4.1 DIRECTORY RETRIEVAL RESPONSE TIME:

Directory search and retrieval operations under the new expanded directory scheme will perform the same as current releases. Performance will become degraded when "chaining" of indices is required. This becomes necessary when the prescribed maximum is reached:

	Avail.	of	ABS
Accounts per system	632	of	806
Users per account	200	of	247
Groups per account	96	of	114
Files per group	1385	of	1722
VSD per group	32	of	57

Some means to "re-structure" the directory when the number of chains becomes relatively "long" must be provided, as this will cause excessive "seeks" on the disc. Re-structuring would consist of building the index block as contiguous sectors on disc.

2.4.2 COMPATIBILITY WITH CURRENT RELEASES:

2.4.2.1 INSTALLING A NEW SYSTEM ON AN EXISTING DISC:

This will not be a problem, as the new "chained" indices can be added to an existing structure. No RELOAD is required for installing the new directory routines.

2.4.2.2 INSTALLING AN EARLIER RELEASE OF MPE ON A DISC WITH A NEW DIRECTORY:

This may occur with the "Mounting" of a Private Volume disc on an earlier system, or with the "backward" update of a system disc. Neither one of these will destroy the new directory (necessitating a RELOAD). For directories more than 6000 sectors, the bitmap will extend into the area where SYSACCTINDEX normally exists (this is required on earlier systems to begin the search).

If a "chain" link has been created in a directory (of 6000 sectors or less), we need to prevent an earlier release of INITIAL from bringing up the system because of the possibility of a "RECOVER LOSS DISC SPACE". This would remove all files which exist in an extended

link in the directory. I am investigating a couple of possibilities in this area.

HEWLETT hp PACKARD

HP 3000 SERIES II/III
OPERATING SYSTEM SPECIALIST
TRAINING COURSE
STUDENT STUDY GUIDE

TITLE: File System

OBJECTIVE: At the completion of this module the student should be able to:

1) Analyze a PBCX of a user process in a system dump to determine:
   A) what files are being used by the process.
   B) The current state of those files.
2) Be able to analyze a system failure dump showing a failure found by the file system and determine a workaround.

TIME: Three Days

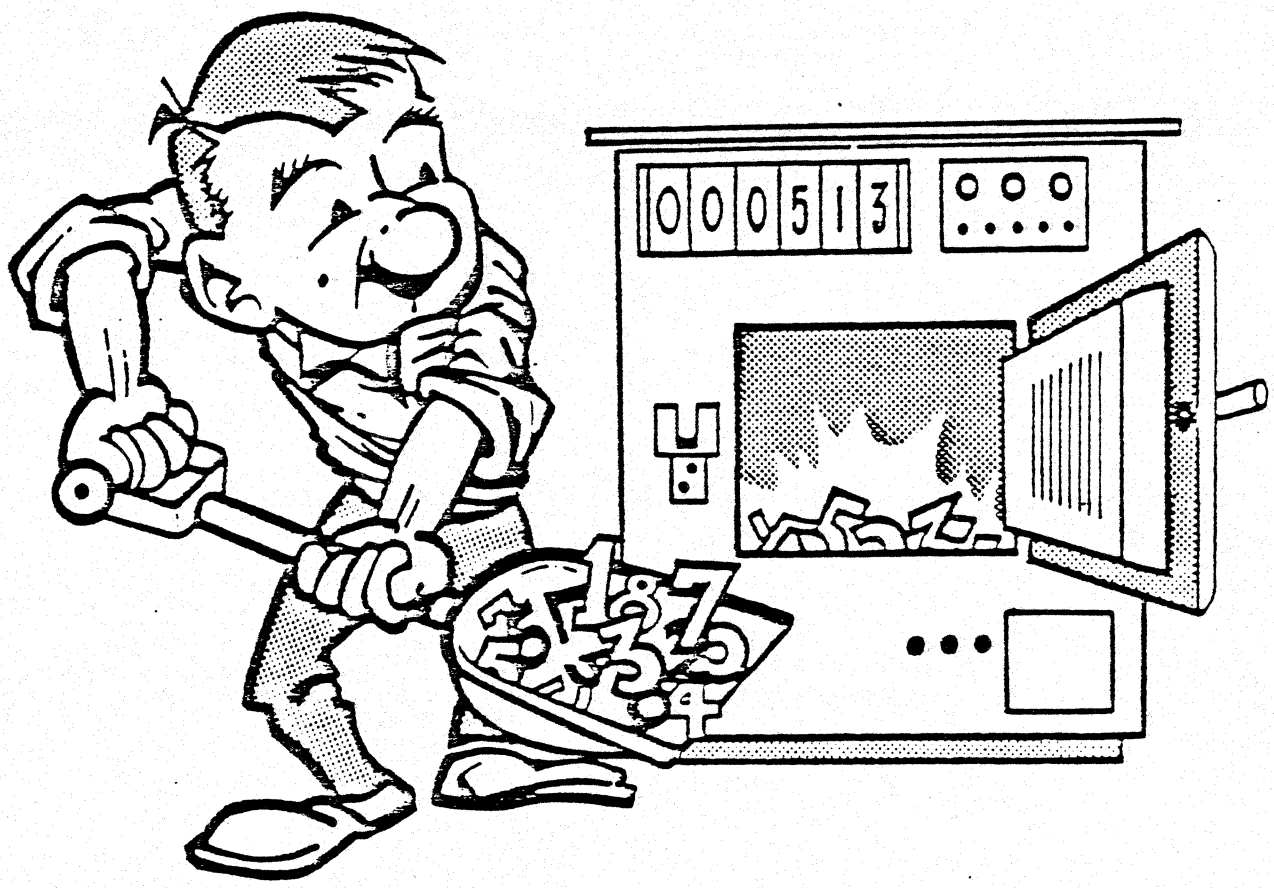
REFERENCES: Student handouts
MPE Intrinsics Manual

INTRODUCTION:

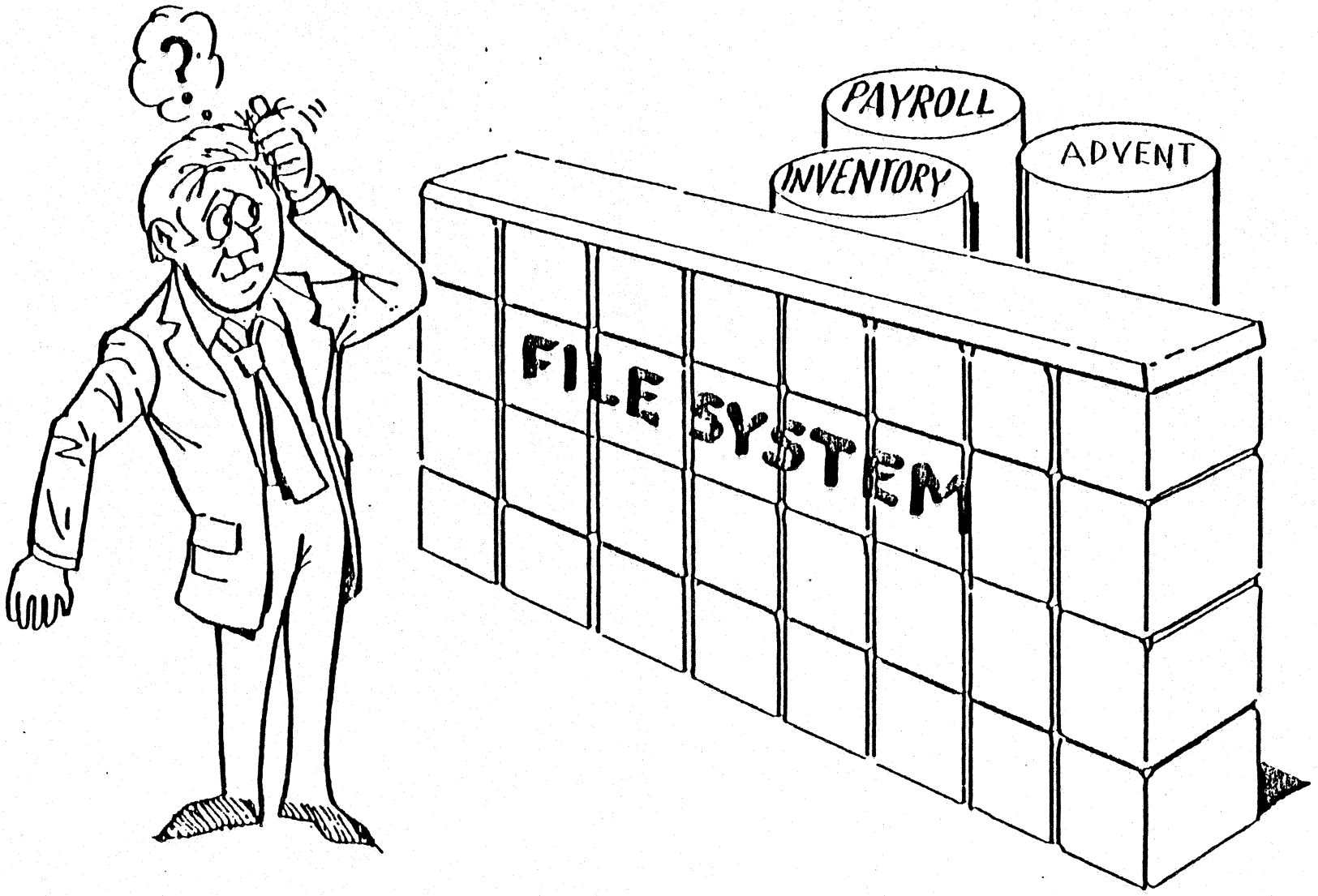
EVERY FILE (EXCEPT $NULL) INVOLVES A PHYSICAL DEVICE. THE FILE SYSTEM IS A MODULE OF CODE THAT PRIMARILY INTERFACES BETWEEN THE USER AND THE I/O SYSTEM SO THAT THE USER/PROGRAMMER DOES NOT HAVE TO PUT IN EXTENSIVE WORK ACCORDING TO WHAT TYPE OF PHYSICAL DEVICE THE FILE WILL RESIDE. THIS INCLUDES THE PROGRAMMER BECAUSE HE/SHE DOES NOT HAVE TO PAY MUCH ATTENTION TO WHAT DEVICE TYPE WILL BE USED WHEN HE WRITES THE PROGRAM AND THE USER BECAUSE HE/SHE CAN CHANGE WHAT TYPE OF DEVICE IS USED WITHOUT HAVING THE PROGRAMMER CHANGE THE PROGRAM.

THIS MODULE IS INTENDED TO FAMILIARIZE THE STUDENT WITH HOW THE FILE SYSTEM INTERNALLY HANDLES THE TABLES IT NEEDS TO ACCOMPLISH THE WORK OF PASSING DATA AND CONTROL INFORMATION BETWEEN THE USER AND THE I/O SYSTEM.

THE FILE SYSTEM



THE FILE SYSTEM HELPS FULFILL THE PRIMARY FUNCTION OF AN OPERATING SYSTEM



THE PRIMARY FUNCTION OF AN OPERATING SYSTEM IS TO STAND BETWEEN A USER AND HIS DATA

* WHAT IS THE DIVISION POINT BETWEEN THE USER, THE FILE SYSTEM AND THE I/O SYSTEM?

* BETWEEN THE USER AND THE FILE SYSTEM IT IS THE FILE SYSTEM INTRINSICS

FCARD	FREADSEEK
FCHECK	FRELATE
FCLOSE	FRENAME
FCONTROL	FSETMODE
FGETINTO	FSPACE
FLOCK	FUNLOCK
FOPEN	FUPDATE
FPOINT	FWRITE
FREAD	FWRITEDIR
FREADDIR	FWRITELABEL
FREADLABEL	PRINTFILEINFO

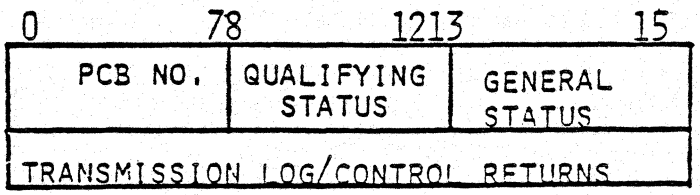
* BETWEEN THE FILE SYSTEM AND THE I/O SYSTEM IT IS PROCEDURE ATTACHIO.

* WHAT DOES ATTACHIO DO AND WHAT DOES IT RETURN?

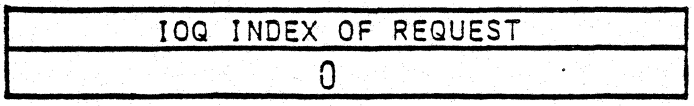
* ATTACHIO ALLOWS THE FILE SYSTEM TO INITIATE BLOCKED OR UNBLOCKED I/O REQUESTS BY BUILDING AN IOQ ENTRY AND AWAKING THE MONITOR TO HANDLE THE REQUEST.

* ATTACHIO RETURNS TWO WORDS:

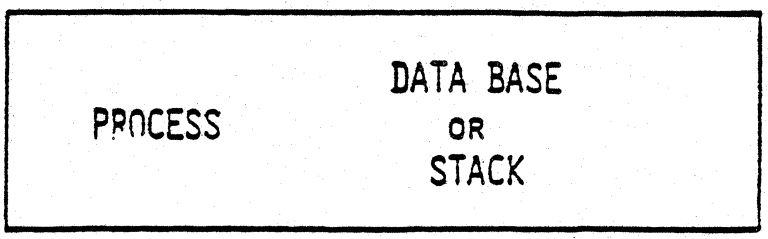
BLOCKED I/O

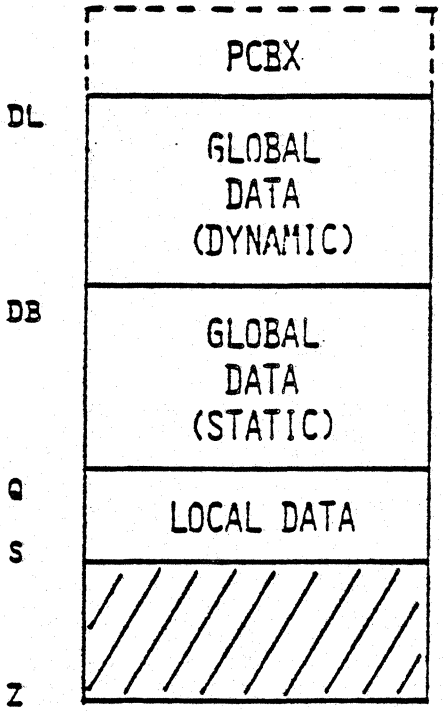


UNBLOCKED I/O



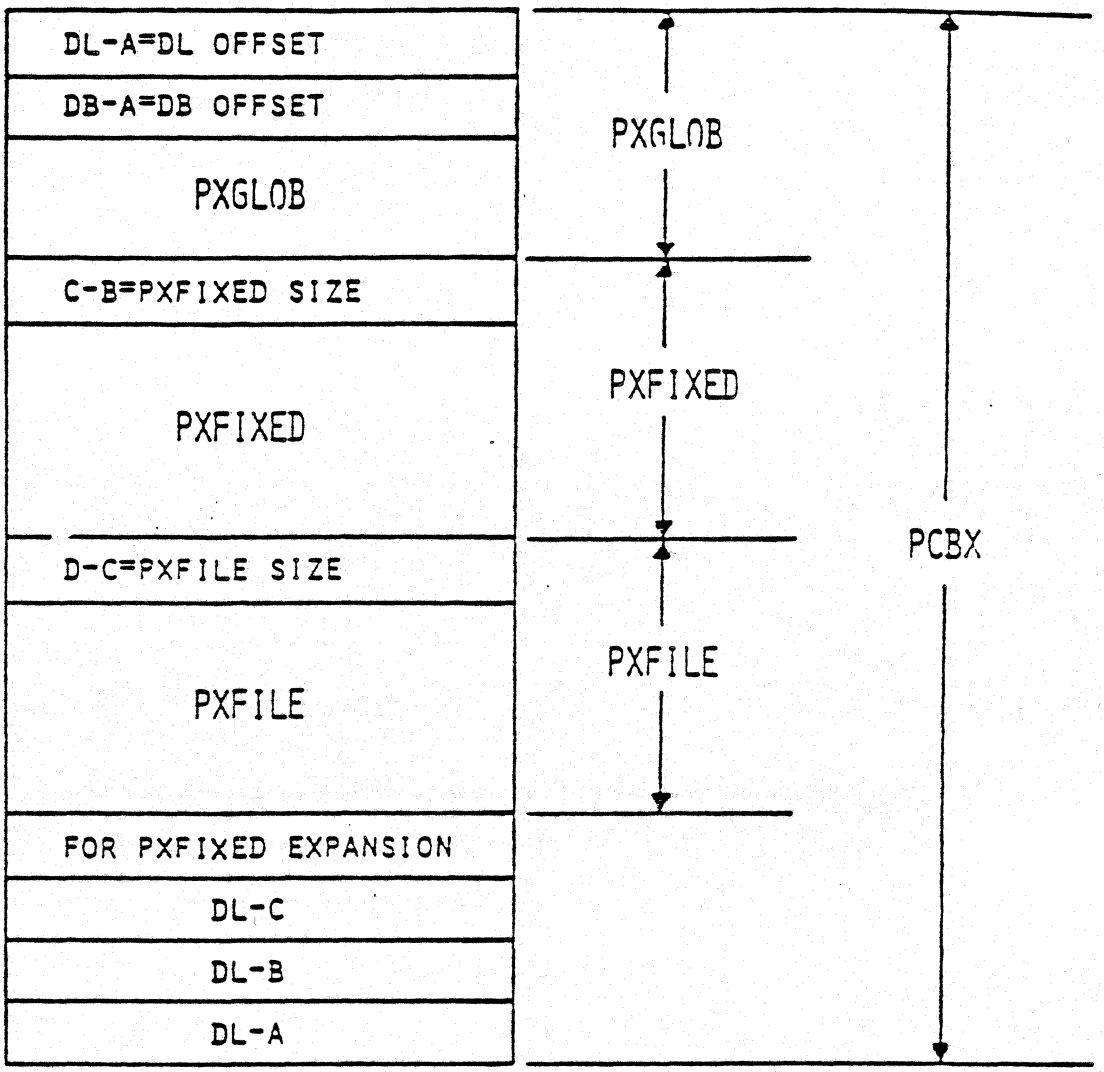
- WHERE IS THE FILE SYSTEM'S DATA BASE?





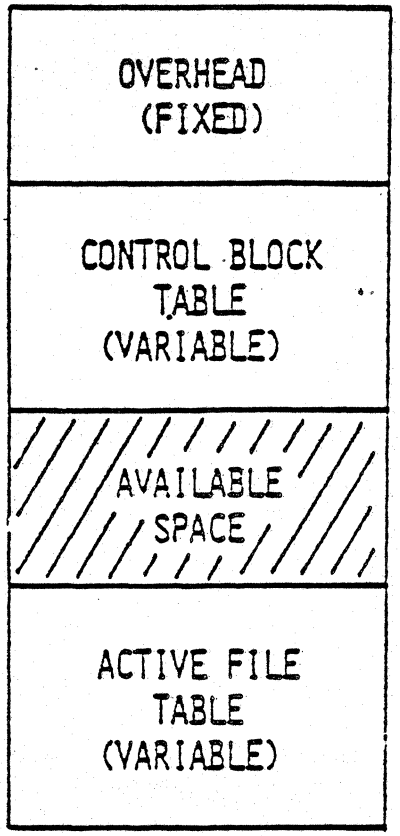
- PCBX STANDS FOR PROCESS CONTROL BLOCK EXTENSION
- PCBX CONTAINS PROCESS STATE INFORMATION THAT NEED NOT BE IN MEMORY UNLESS THE PROCESS IS EXECUTING
- PCBX IS INVISIBLE AND INACCESSIBLE TO USER MODE PROCESS
- PCBX IS ACCESSIBLE TO PRIVILEDGED MODE PROCESS

- WHAT IS THE FORMAT OF THE PCBX?



- THE PCBX HOLDS A VARIETY OF STATE INFORMATION FOR VARIOUS PARTS OF MPE
- THE PRINCIPAL USER OF THE PCBX IS THE FILE SYSTEM
- AS THE FILE SYSTEM'S STORAGE REQUIREMENTS CHANGE, IT CAN EXPAND OR CONTRACT THE SIZE OF THE PXFILE AREA
- THE PXFILE IS (ALMOST) USED EXCLUSIVELY BY THE FILE SYSTEM

- WHAT IS THE FORMAT OF THE PXFILE?

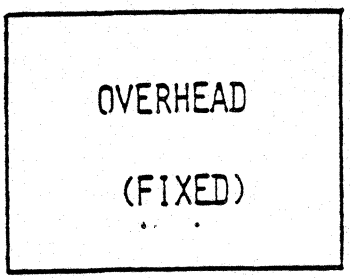


- THE PXFILE CONSISTS OF FOUR PARTS:
  1. OVERHEAD
  2. CONTROL BLOCK TABLE
  3. ACTIVE FILE TABLE
  4. AVAILABLE SPACE

- THE CONTROL BLOCK TABLE AND THE ACTIVE FILE TABLE EXPAND AND CONTRACT, GIVING OR TAKING SPACE FROM THE AVAILABLE SPACE AREA

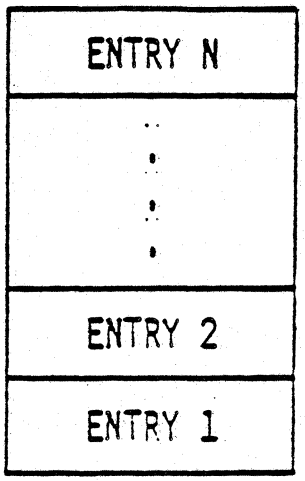
- WHEN THE AVAILABLE SPACE AREA IS EXHAUSTED, THE PXFILE IS EXPANDED AND THE ACTIVE FILE TABLE RELOCATED

- WHAT IS THE FUNCTION AND FORMAT OF THE OVERHEAD PART OF THE PXFILE?



- THE OVERHEAD PART CONTAINS:
  1. PXFILE MANAGEMENT INFORMATION:
     LENGTH OF PXFILE
     LENGTH OF ACTIVE FILE TABLE
  2. MISC. FILE SYSTEM STATE INFORMATION:
     - LAST FOPEN ERROR NUMBER
     - LAST RESPONDING NO-WAIT I/O FILE NUMBER
     - LOCATION OF AUXILIARY CONTROL BLOCK TABLES
     - NOCB FLAG
  3. MISC. NON FILE SYSTEM STATE INFORMATION
     - LAST COPEN ERROR NUMBER
     - LAST DOPEN ERROR NUMBER
     - LAST KOPEN ERROR NUMBER
     - OTHER CS AND DS STATE INFORMATION

- WHAT IS THE FUNCTION AND FORMAT OF THE ACTIVE FILE TABLE (AFT)?



- FOR EACH PROCESS THERE IS A ONE-TO-ONE CORRESPONDENCE BETWEEN A FILE NUMBER AND AN AFT ENTRY NUMBER

- THE AFT IS NEGATIVELY INDEXED BY FILE NUMBER TO OBTAIN THE CORRESPONDING AFT ENTRY

- AN AFT ENTRY CONTAINS INFORMATION WHICH GROSSLY CHARACTERIZES FILE ACCESS

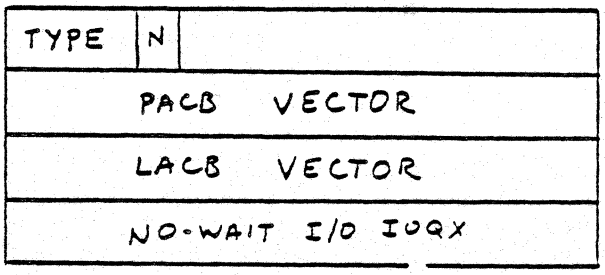
- MOST IMPORTANTLY, AN AFT ENTRY CONTAINS THE LOCATION OF THE CONTROL BLOCKS NECESSARY FOR FILE ACCESS

- SPECIFICALLY, AN AFT ENTRY CONTAINS:
  - AFT ENTRY TYPE NUMBER
  - $NULL FLAG
  - LACB VECTOR
  - PACB VECTOR
  - PENDING NO-WAIT I/O IOQX

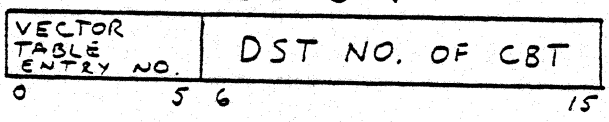
0 = LOCAL FILE
1 = REMOTE FILE
2 = DSNUM
3 = DSNUM (No Wait)
4 = CSFILE
5 = CSFILE (With auto dial)
6 = KSAM

- A FREE AFT ENTRY CONTAINS ZERO'S; A USED AFT ENTRY CONTAINS AT LEAST ONE NON-ZERO WORD

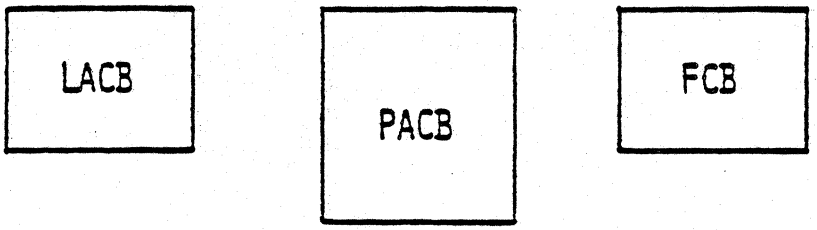
AFT ENTRY



VECTOR

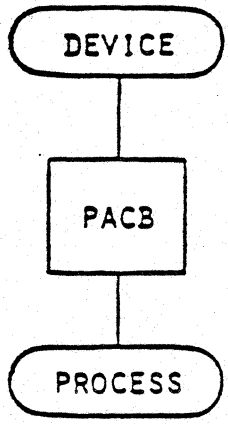


- WHAT IS THE FUNCTION OF A CONTROL BLOCK?

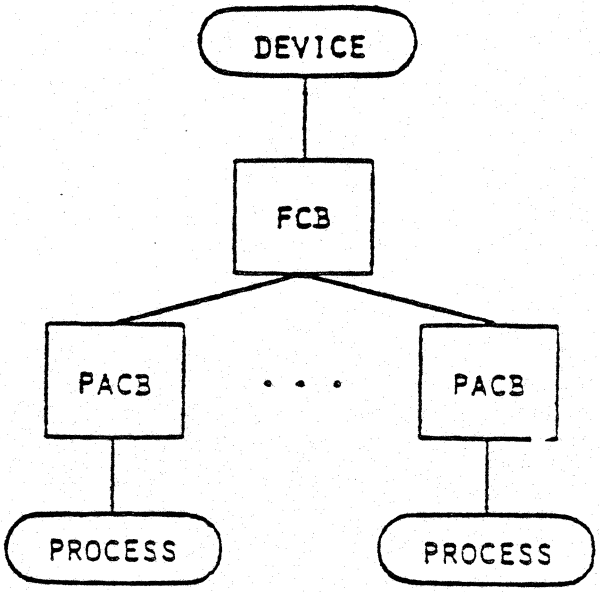


- CONTROL BLOCKS ARE USED TO COORDINATE FILE ACCESS
- CONTROL BLOCKS CONTAIN PRACTICALLY ALL OF THE FILE SYSTEM's DATA BASE
- CONTROL BLOCKS COME IN THREE FLAVORS:
  LACB - LOGICAL ACCESS CONTROL BLOCK
  PACB - PHYSICAL ACCESS CONTROL BLOCK
  FCB - FILE CONTROL BLOCK
- CONTROL BLOCKS ARE DYNAMIC ENTITIES: GENERALLY,
  -THEY ARE CREATED WHEN A FILE ACCESS IS ESTABLISHED (FOPEN)
  -THEY ARE CONSULTED WHEN A FILE IS ACCESSED (E.G. FREAD)
  -THEY ARE DESTROYED WHEN A FILE ACCESS IS TERMINATED (FCLOSE)
- CONTROL BLOCKS ARE EITHER:
  LOCAL - ACCESS INFORMATION FOR A PARTICULAR PROCESS
  GLOBAL - ACCESS INFORMATION COMMON TO ALL ACCESSORS
- FROM Ø TO 3 CONTROL BLOCKS ARE NECESSARY FOR ACCESSING EACH FILE

- WHAT ARE THE RELATIONSHIPS BETWEEN DEVICES, PROCESSES, CONTROL BLOCKS AND FILES?



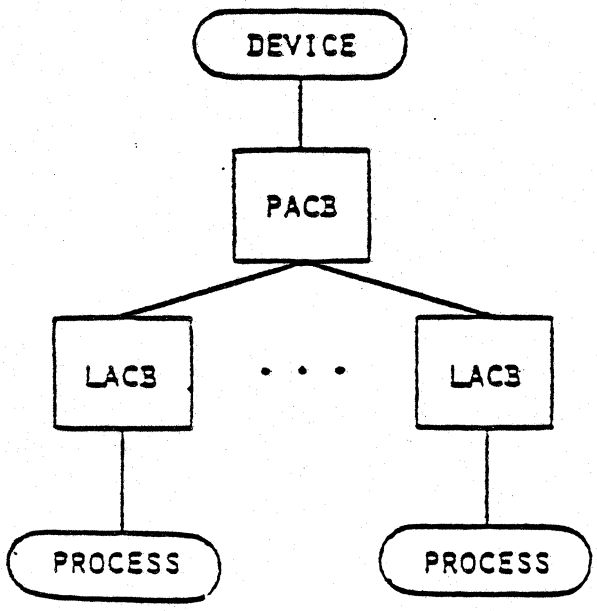
NON-DISC, NON-MULTI



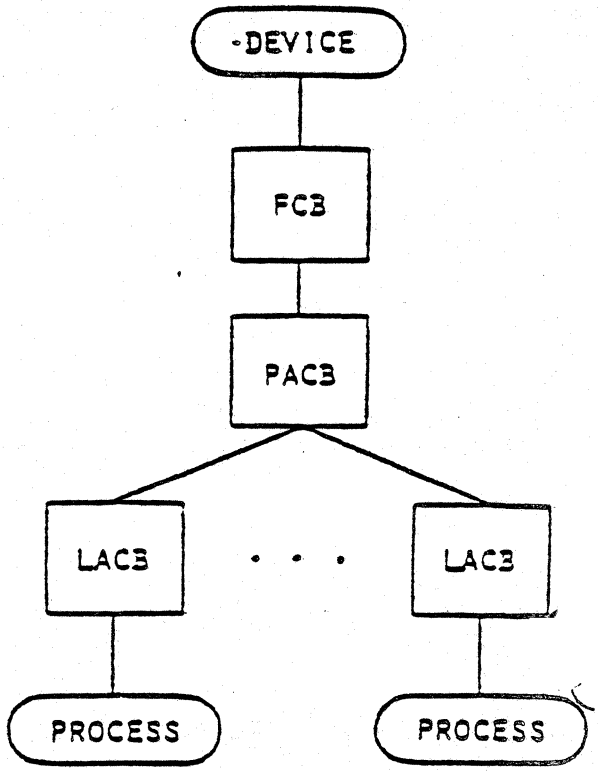
DISC, NON-MULTI

- FILE ACCESS INFORMATION WHICH IS UNIQUE AND LOCAL TO EACH ACCESSOR (PROCESS) IS CONTAINED IN THE PACB
- FILE INFORMATION WHICH IS COMMON TO EACH SHARABLE FILE IS CONTAINED IN THE FCB
- THE ONLY DEVICE WHICH SUPPORTS SHARABLE FILES IS THE DISC
- HENCE THE ONLY FILE WHICH REQUIRES AN FCB IS A DISC FILE; IN FACT, THERE IS EXACTLY ONE FCB FOR EACH DISC FILE BEING ACCESSED
- THE PACB REPRESENTS THE ACCESSOR
- THE FCB REPRESENTS THE FILE

- WHAT ARE THE RELATIONSHIPS BETWEEN DEVICES, PROCESSES, CONTROL BLOCKS AND FILES? (CONT)



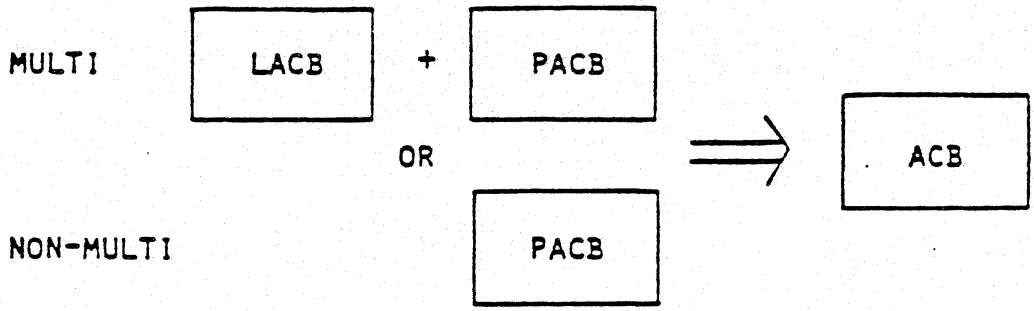
NON-DISC, MULTI



DISC, MULTI

- THE MULTI-ACESS CAPABILITY PERMITS THE SHARING OF THE ACCESS TO A FILE (AS OPPOSED TO THE SHARING OF THE FILE ITSELF)
- FILE ACCESS INFORMATION WHICH IS UNIQUE AND LOCAL TO EACH INDIVIDUAL ACCESSOR IS CONTAINED IN THE LACB
- FILE ACCESS INFORMATION WHICH IS COMMON TO EACH SET OF ACCESSORS IS CONTAINED IN THE PACB
- THE PACB REPRESENTS THE (SHARED) ACCESS
- THE LACB REPRESENTS THE (UNIQUE) ACCESSOR

- HOW IS AN ACB COMPOSED FROM A LACB AND PACB?



- ONE OF THE MAJOR IMPLEMENTATION HEADACHES FACED BY THE FILE SYSTEM IS THE ENVIRONMENT SWITCH (EXCHANGEDB) REQUIRED IN ORDER TO ACCESS A CONTROL BLOCK

- SINCE UP TO THREE CONTROL BLOCKS MAY BE ACCESSED FOR EACH INTRINSIC CALL, IT IS BENEFICIAL TO COMBINE TWO OR MORE CONTROL BLOCKS INTO A SINGLE CONTROL BLOCK FOR THE PURPOSE OF EFFICIENT ACCESS

- THIS POLICY IS IMPLEMENTED WITH RESPECT TO THE LACB AND PACB

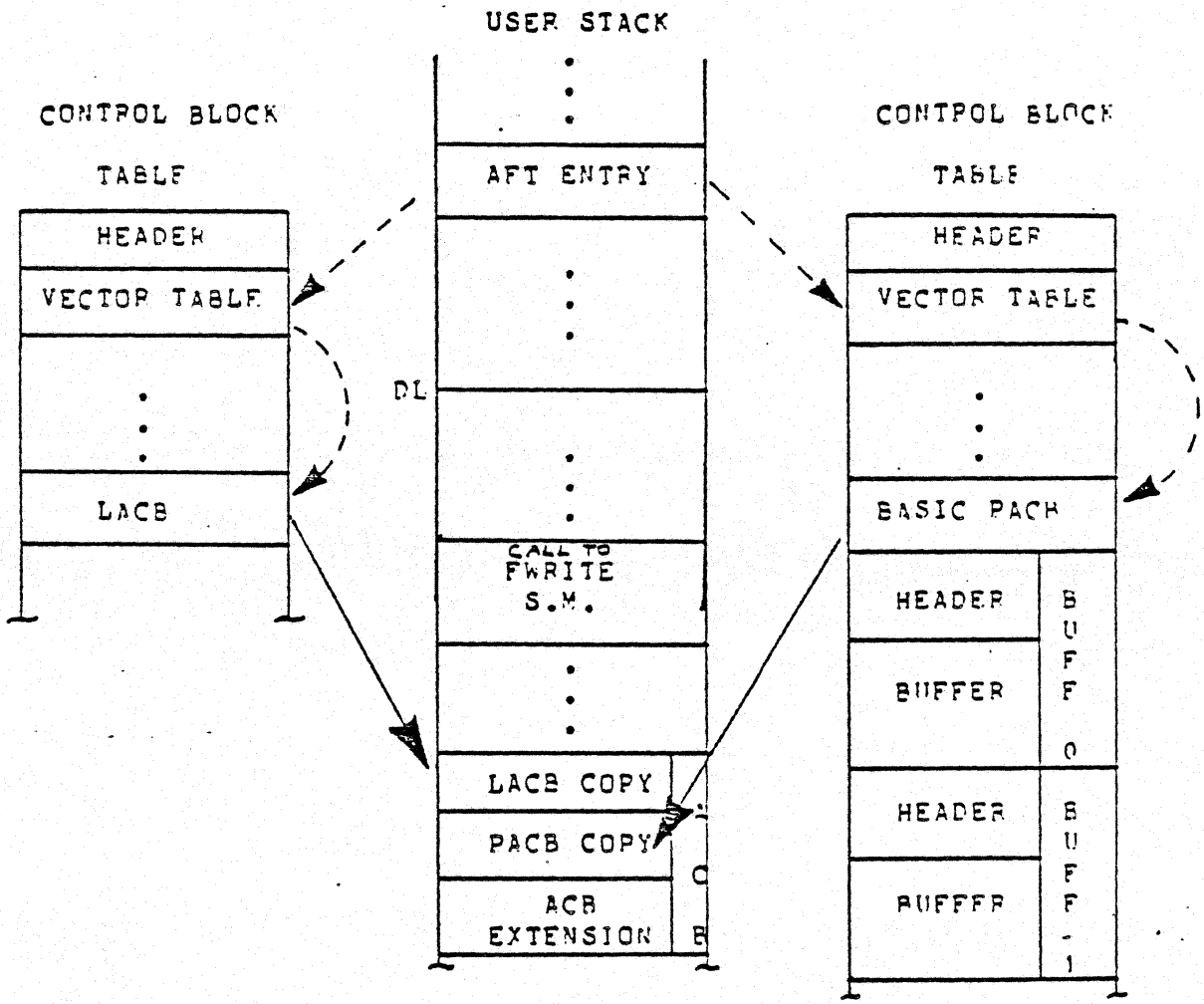
- WHEN THE PACB IS FIRST ACCESSED AND AN LACB EXISTS, THE APPROPRIATE FIELDS IN THE PACB ARE OVERLAYED BY THE CORRESPONDING FIELDS IN THE LACB; THE RESULTING CONTROL BLOCK IS CALLED AN ACB.

- WHEN ACCESS TO THE ACB IS COMPLETE, THE LACB IS UPDATED WITH THE (POSSIBLY MODIFIED) FIELDS OF THE ACB

- THE ACB IS A DYNAMIC, TEMPORAL ENTITY

MPE IV ACB

ON MPE IV, THE ACB PHYSICALLY RESIDES ON THE STACK OF THE CALLER:



NOTE: Q-RELATIVE LOCATION OF ACB IS DEPENDENT UPON INTRINSIC LOCAL STORAGE

Location of ACB by intrinsic:

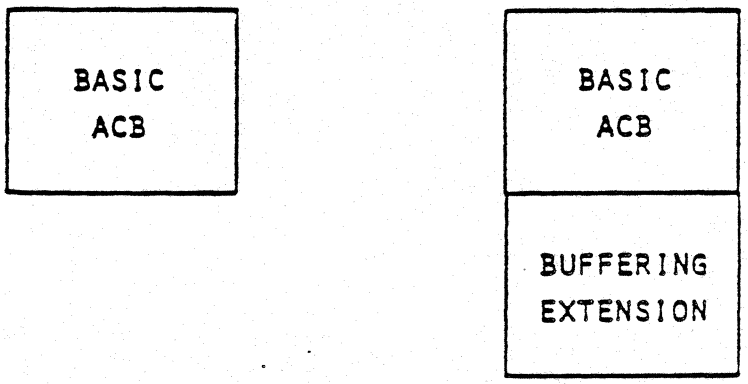
Intrinsic	Location of ACB
FREAD	Q+10
FWRITE	Q+ 9
FREADDIR/FWPITEDIR	Q+11
FUPDATE	Q+10
FBREAK	Q+6
FUNBREAK	Q+ 6
FRESETEOF	Q+ 6
FREDASEEK	Q+ 9
FSPACE	Q+10
FPOINT	Q+ 8
FCONTROL	Q+55
FSETMODE	Q+8
FRELATE	Q+10
FCHECK	Q+18
FGETINFO	Q+38
FREADLABEL/FWRITELABEL	Q+19
FLOCK	Q+10
FUNLOCK	Q+ 9
IOWAIT/IODONTWAIT	Q+21

HINT: Look for file name and ACB first word.
IOMOVE assumes ACB copy is located at Q-63:

```
56 Words = ACB
 3 Words = Parms to IOMOVE
 4 Words = stack marker
---------
63 Words
```

Warning: TOS values must not be stacked before calling IOMOVE.

- WHAT DOES AN ACB LOOK LIKE?



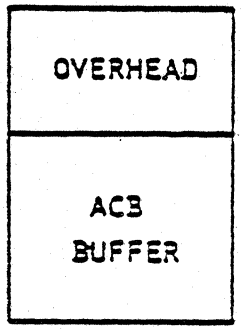
- THE ACB COMES IN TWO FLAVORS:
  - BASIC ACB - UNBUFFERED ACCESS
  - BASIC ACB + BUFFERING EXTENSION - BUFFERED ACCESS

- THE BUFFERING EXTENSION PHYSICALLY RESIDES WITH THE PACB (SINCE THE ACB IS A RESULT OF OVERLAYING THE LACB INFO ONTO THE PACB).

- THE BASIC ACB CONTAINS SUCH INFORMATION AS:

LOCAL FILE NAME	FILE POINTER
RECORD SIZE	ERROR CODE
BLOCK SIZE	RECORD TRANSFER COUNT
BLOCKING FACTOR	BLOCK TRANSFER COUNT
LOGICAL DEVICE NR.	LAST I/O STATUS
VOLUME TABLE INDEX	LAST I/O TRANSMISSION LOG
DEVICE TYPE	EOF FLAG
AOPTIONS	
FOPTIONS	

* WHAT DOES AN ACB BUFFERING EXTENSION LOOK LIKE?



* THE BUFFERING EXTENSION IS COMPOSED OF FROM 1 TO 16 BLOCK BUFFERS

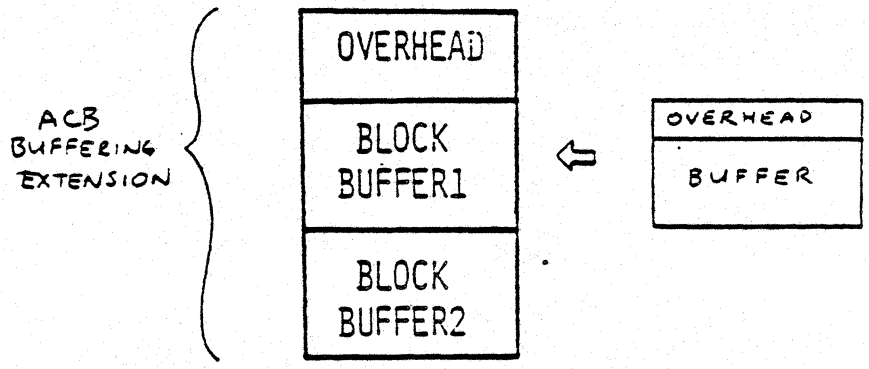
* EACH BLOCK BUFFER CONTAINS:
  - OVERHEAD - BUFFER MANAGEMENT INFORMATION
  - BUFFER - BUFFER FOR ONE BLOCK OF DATA

* THE OVERHEAD PART CONTAINS:

  BLOCK NUMBER
  LDEV AND SECTOR NUMBER (DISC ONLY)
  IOQ ENTRY INDEX (WHEN I/O IS PENDING)
  IOCB (WHEN I/O IS COMPLETED)

  I/O MODE FLAG (READ OR WRITE)
  BUFFER DIRTY FLAG
  I/O IN PROGRESS FLAG

* HOW DOES THE FILE SYSTEM DO ANTICIPATORY READS?



* WHEN CALLING ANY DATA TRANSFERING INTRINSIC (READ OR WRITE) THE CALLER PASSES A RECORD NUMBER.

  - FREAD - IMPLICIT RECORD NUMBER
    BY VIRTUE OF SEQUENTIAL ACCESS
  - FREADDIR - PASSED BY THE USER.

* CALCULATE WHAT BLOCK CONTAINS THE RECORD. (REMEMBER THAT IN UNDEFINED A BLOCK IS A RECORD).

* LOOK AT THE OVERHEAD AND FIND OUT WHICH BUFFER IS USED FOR THAT BLOCK.

* INITIATE AN I/O REQUEST FOR THAT BLOCK (ATTACHIO).

* HAS THE I/O COMPLETED? (LOOK AT THE BUFFER EXTENSION I/O IN PROGRESS FLAG [.(8).(15:1)]).

  -- IF I/O COMPLETE THEN WAIT FOR THE PROCESS TO ASK FOR THE DATA.

- HOW ARE ANTICIPATORY READS HANDLED FOR DIRECT RECORD ACCESS (FREADDIR)?

- THE FILE SYSTEM DOES NOT DO ANTICIPATORY READS WHEN THE FILE IS RANDOMLY ACCESSED BECAUSE THE FILE SYSTEM DOESN'T KNOW WHAT THE SEQUENCE WILL BE.

- FREADSEEK INTRINSIC SERVES AS A SHOULDER TAP TO THE FILE SYSTEM TO CAUSE ANTICIPATORY READS.

- FWRITEDIR CAN BE PRECEDED BY FREADSEEK BECAUSE THERE IS A DIRTY FLAG THAT CAUSES THE BUFFER TO BE WRITTEN IF IT HAS BEEN MODIFIED.

* WHEN AND HOW ARE FILE AREAS INITIALIZED?

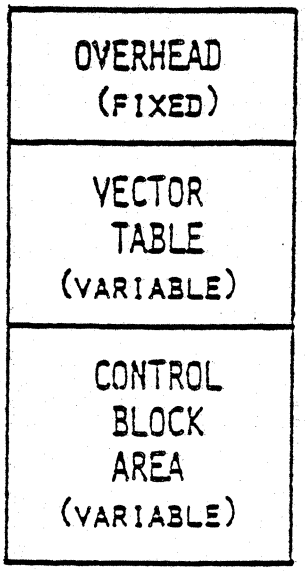
* FILES ARE INITIALIZED ONLY ON AN "AS NEEDED" BASIS.

* IF FOPEN A NEW FILE AND WRITE ONE RECORD, THE BLOCK IS INITIALIZED IN THE BUFFER AND GETS WRITTEN WHEN THE BLOCK IS WRITTEN.

* IF FOPEN OLD FILE AND MOVE THE POINTER INTO NEXT EXTENT THEN THE JUST ALLOCATED EXTENT WILL BE INITIALIZED BY REQUEST TO THE DRIVER.

* IF OPEN NEW FILE AND MOVE POINTER TO RECORD 100 (FOR EXAMPLE) THEN THE DISC AREA WILL BE INITIALIZED BY THE DRIVER BECAUSE POTENTIALLY ANY OF THE RECORDS COULD BE READ AND THE INITIALIZING PREVENTS READING GARBAGE.

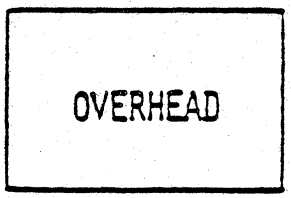
- WHAT DOES A CONTROL BLOCK TABLE (CBT) LOOK LIKE?



- A CONTROL BLOCK TABLE IS USED TO FACILITATE THE MANAGEMENT OF CONTROL BLOCKS
- A CONTROL BLOCK TABLE CAN BE LOCATED IN TWO PLACES:
  1. SUB-SECTION OF PXFILE AREA
  2. TOTALLY CONTAINED IN A DATA SEGMENT
- A CONTROL BLOCK TABLE CONSISTS OF THREE PARTS:
  1. OVERHEAD
  2. VECTOR TABLE
  3. CONTROL BLOCK AREA
- THE CONTROL BLOCK AREA IS EXPANDABLE AND CONTRACTABLE
- THE VECTOR TABLE IS ALSO EXPANDABLE AND CONTRACTABLE BUT IN A RESTRICTED SENSE: SPACE MUST BE TAKEN FROM THE CONTROL BLOCK AREA

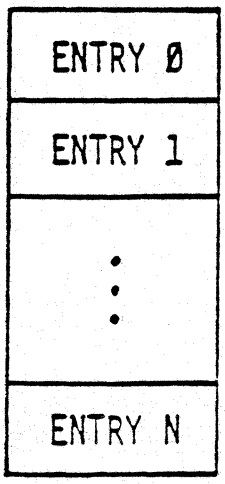
```
MP 3000 II MEMORY DUMP B00.00 OF SYS VER R  UPDATE  00  FIX 02  DUMP TIME  4/29/79,  1147PM                BANK  2     PAGE  210
(C) HEWLETT-PACKARD CO. 1976

116174: 177777 000000 000000 000002 000000 000000 000000 000000  116204: 000000 000000 000000 000000 000000 000000
***PXFILE: (ZERO TABLE ENTRIES ARE NOT PRINTED)
116212: 000510 000000 000000 000000 000000 000030 000000 000000  116222: 000000 000000 000000 000000 000000 000000 000000 000000
116232: 000267 000150 000100 000000 000000
------- FILE VECTOR TABLE:              ENTRY  ADDRESS  LOCK  BRK  LOCK COUNT/PIN   HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
116237: 000106 100426 000000 000000        0      106    LOCK                  1  26
116243: 000126 100426 000000 000000        1      126    LOCK                  1  26
------- CONTROL BLOCKS:
116337(000105): 000001 140070 000155 000001 002244 001700 000024 001412 000000 000000 177773 022123  116337:.....m.................BB
116353(000121): 052104 044516 020040 000110 000044 140020 000155 000002 002614 001701 000000 000000  116353:IDIN  .H.$...m..........
116367(000135): 000000 000000 000000 022123 052104 046111 051524 000111 000045 000121 000000 002001  116367:......$STDLIST.I.%.Q....
116403(000151): 177766 000010 000401 004150 000004 000000 000000 000000 000216 000000 000000 000000  116403:.......h.................
116417(000165): 000216 000000 000217 000401 000400 000217 000001 050125 041040 020040 020040 051531  116417:..............PUB      SY
116433(000201): 051440 020040 020040 000401 010314 100061 001400 000000 000014 000000 000014 000000  116433:S      .....1..........
116447(000215): 000014 004000 000767 000000 000001 000400 000200 006150 000000 000000 000000 037401  116447:..............h.......?.
116463(000231): 000001 002001 002424 000000 000001 000000 000000 043103 047520 054440 020040 000000  116463:..............FCOPY    ..
116477(000245): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 177777  116477:........................
116513(000261): 177777 000000 000000 000000 000000 000000                                            116513:............
116521: 000000 000000 000000 000000 000000 000000 000000 000000  116531: 000000 000000 000000 000000 000000 000001 140020
116541: 000155 000001 002244 001700 000024 001412 000000 000000  116551: 177773 022123 052104 044516 020040 000110 000044 140020
116561: 000155 000002 002614 001701 000000 000000 000000 000000  116571: 177762 022123 052104 046111 051524 000111 000045 000040
116601: 000000 002001 177766 000010 000401 000156 000004 000000  116611: 000000 000000 000310 000000 000000 000000 000020 000000
116621: 000043 003001 000400 000043 000001 050125 041040 020040  116631: 020040 051531 051440 020040 020040 000401 032512 000000
116641: 000000 000000 000000 000000 000000 000000 000000 000000  116651: 000000 000000 000000 000000 000000 000000 000000 000000
116661: 000000 000000 000000 000000 000000 000000 000000 000000  116671: 000000
------- AVAILABLE FILE TABLE:          FNUM   FTYPE  $NULL    PACB V     LACB V    IOQX
116672: 000000 000153 000000 000000        6    FILE             0  153     0    0
116706: 000000 000160 000000 000000        5    FILE             0  160     0    0
116702: 000000 000157 000000 000000        4    FILE             0  157     0    0
116712: 000000 000155 002150 000000        2    FILE             0  155     1  150
116716: 000000 000155 000150 000000        1    FILE             0  155     0  150
**PXPOINTERS:
116722: 000000 000514 000572 000602
****DL REGISTER: ************************************************************************************************************
****DB REGISTER: ************************************************************************************************************
116726(000000): 000002 042105 041125 043440 006520 054456 050125 041056 051531 051415 020040 020040  1167261..DEBUG .PY.PUB.SYS.
116742(000014): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  116742:
LINES 116756 - 116771 SAME AS ABOVE
```

- WHAT DOES THE OVERHEAD PART OF A CBT LOOK LIKE?



- THE OVERHEAD PART CONTAINS INFORMATION NECESSARY FOR THE MANAGEMENT OF THE CBT
- SPECIFICALLY, IT CONTAINS:
  - CBT SIZE
  - DST NUMBER OF DATA SEGMENT CONTAINING CBT
  - VECTOR TABLE SIZE
  - CBT LOCK

- WHAT DOES THE VECTOR TABLE PART OF A CBT LOOK LIKE?



- THE VECTOR TABLE IS USED TO LOCATE AND CONTROL THE ACCESS TO CONTROL BLOCKS IN THE CBT

- THE ADDRESS OF A CONTROL BLOCK IS COMPOSED OF TWO PARTS:
  1. DST NUMBER OF CBT (10 BITS)
  2. VECTOR TABLE INDEX (6 BITS)

CONTROL BLOCK VECTOR (16 BITS)

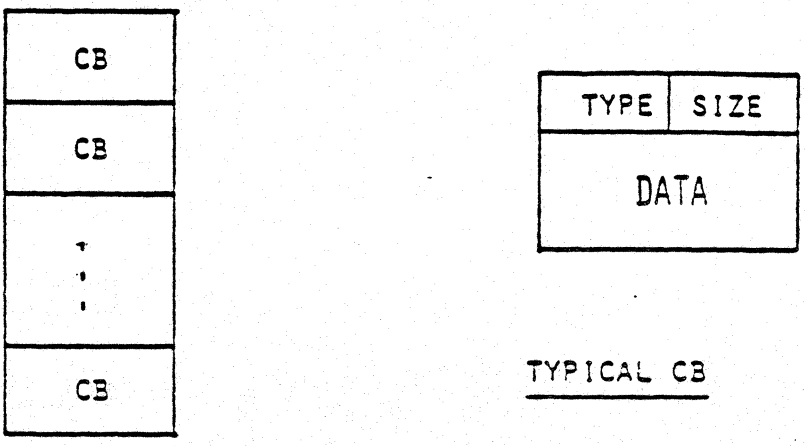
- SPECIFICALLY, A VECTOR TABLE ENTRY CONTAINS:
  - CONTROL BLOCK ADDRESS
  - CONTROL BLOCK LOCK

HEWLETT PACKARD

****** PCBX AND STACK MARKERS FOR DST150 (PCB 26) ******

```
MP 3000 II MEMORY DUMP B00.00 OF SYS VER B  UPDATE  00  FIX 02  DUMP TIME  4/29/79,  1147PM                    BANK  2     PAGE  210
(C) HEWLETT-PACKARD CO. 1976

1161741: 177777 000000 000000 000002 000000 000000 000000 000000  1162041: 000000 000000 000000 000000 000000 000000
***PXFILE: (ZERO TABLE ENTRIES ARE NOT PRINTED)
1162121: 000510 000000 000000 000000 000000 000030 000000 000000  1162221: 000000 000000 000000 000000 000000 000000 000000 000000
1162321: 000267 000150 000100 000000 000000
------- FILE VECTOR TABLE:                ENTRY  ADDRESS  LOCK  BRK  LOCK COUNT/PIN   HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
1162371: 000106 100426 000000 000000        0      106    LOCK              1  26
1162431: 000126 100426 000000 000000        1      126    LOCK              1  26
------- CONTROL BLOCKS:
116337(000105): 000001 140020 000155 000001 002244 001700 000024 001412 000000 000000 177773 022123  116337:.....m.................SS
116353(000121): 052104 046516 020040 000110 000044 140020 000155 000002 002614 001701 000000 000000  116353:TDIN  .H.$...m..........
116367(000135): 000000 000000 000000 022123 052104 046111 051524 000111 000045 000121 000000 002001  116367:......#STDLIST.I.%.Q....
116403(000151): 177766 000010 000401 004150 000004 000000 000000 000000 000216 000000 000000 000000  116403:.......h..................
116417(000165): 000216 000000 000217 000401 000410 000217 000001 050125 041040 020040 020040 051531  116417:..............PUA     SY
116433(000201): 051440 020040 020040 000401 010314 100061 001400 000000 000014 000000 000014 000000  116433:S     .....1..............
116447(000215): 000014 004000 000767 000000 000001 000400 000200 006150 000000 000000 000000 037401  116447:..............h.......!.
116463(000231): 000001 002001 002421 000000 000001 010000 000000 043103 047520 054440 020040 000000  116463:..............FCOPY   ..
116477(000245): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 177777  116477:........................
116513(000261): 177777 000000 000000 000000 000000 000000                                            116513:............
1165211: 000000 000000 000000 000000 000000 000000 000000 000000  1165311: 000000 000000 000000 000000 000000 000001 140020
1165411: 000155 000001 002244 001700 000024 001412 000000 000000  1165511: 177773 022123 052104 046516 020040 000110 000044 140020
1165611: 000155 000002 002614 001701 000000 000000 000000 000000  1165711: 177762 022123 052104 046111 051524 000111 000045 000040
1166011: 000000 002001 177766 000010 000401 000156 000004 000000  1166111: 000000 000000 000310 000000 000000 000000 000020 000000
1166211: 000043 003001 000400 000043 000001 050125 041040 020040  1166311: 020040 051531 051440 020040 020040 000401 012512 000000
1166411: 000000 000000 000000 000000 000000 000000 000000 000001  1166511: 000000 000000 000000 000000 000000 000000 000000 000000
1166611: 000000 000000 000000 000000 000000 000000 000000 000000  1166711: 000000
------- AVAILABLE FILE TABLE:          FNUM   FTYPE  $NULL    PACB  V     LACB  V    IOQX
1166721: 000000 000153 000000 000000     6    FILE             0  153     0   0
1166761: 000000 000160 000000 000000     5    FILE             0  160     0   0
1167021: 000000 000157 000000 000000     4    FILE             0  157     0   0
1167121: 000000 000155 002150 000000     2    FILE             0  155     1  150
1167161: 000000 000155 000150 000000     1    FILE             0  155     0  150
**PXPOINTERS:
1167221: 000000 000514 000572 000602
****DL REGISTER: ************************************************************************************************************
****DB REGISTER: ************************************************************************************************************
```

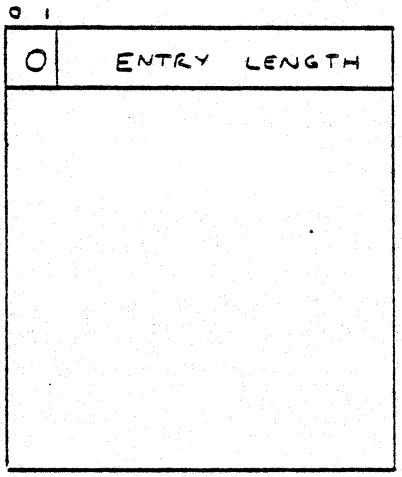
- WHAT DOES THE CONTROL BLOCK AREA OF A CBT LOOK LIKE?



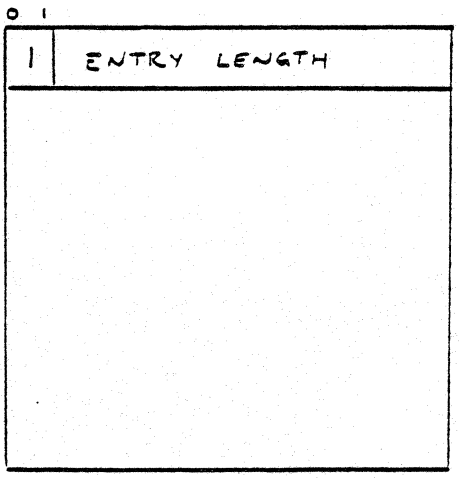
- THE CONTROL BLOCK AREA IS ALWAYS COMPLETELY FILLED CONTROL BLOCKS
- TO FACILITATE CBT MANAGEMENT, ALL CONTROL BLOCKS HAVE A COMMON FORMAT
- IN REALITY, THERE ARE FOUR TYPES OF CONTROL BLOCKS:
  - LACB
  - PACB
  - FCB
  - GARBAGE
- CONTROL BLOCKS CAN BE OF VARYING SIZES
- HOWEVER, ONCE CREATED, THE SIZE OF A CONTROL BLOCK CANNOT CHANGE, NOR CAN THE CONTROL BLOCK BE MOVED WITHIN THE CBT
- SPACE IS ALLOCATED BY SCANNING THE GARBAGE CONTROL BLOCKS USING A FIRST FIT ALGORITHM

```
HP 3000 II MEMORY DUMP B00.00 OF SYS VER B  UPDATE  00  FIX 02  DUMP TIME  4/29/79,  1:47PM                    BANK  2     PAGE  210
(C) HEWLETT-PACKARD CO. 1976

116174: 177777 000000 000000 000002 000000 000000 000000 000000  116204: 000000 000000 000000 000000 000000 000000

***PXFILE: (ZERO TABLE ENTRIES ARE NOT PRINTED)
116212: 000510 000000 000000 000000 000000 000030 000000 000000  116222: 000000 000000 000000 000000 000000 000000 000000 000000
116232: 000267 000150 000100 000000 000000
------- FILE VECTOR TABLE:            ENTRY  ADDRESS  LOCK  BRK  LOCK COUNT/PIN   HIPRI TAIL  HIPRI HEAD  LOPRI TAIL  LOPRI HEAD
116237: 000106 100426 000000 000000      0      106   LOCK              1  26
116243: 000126 100426 000000 000000      1      126   LOCK              1  26
------- CONTROL BLOCKS:
116137(000105): 000001 140020 000155 000001 002244 001700 000024 001412 000000 000000 177773 022123  116337:.....m.....................88
116353(000121): 052104 046111 051524 000111 000045 140070 000155 000002 002614 001701 000000 000000  116353:IDIN  .H.E...m............
116367(000135): 000000 000000 000000 022123 052104 046111 051524 000111 000045 000121 000000 002001  116367:......#SIDLIST.I.E.Q.....
116403(000153): 177766 000010 000401 000156 000004 000000 000000 000000 000216 000000 000000 000000  116403:........h.....................
116417(000165): 000216 000000 000217 000401 000400 000217 000001 050125 041040 020040 020040 051531  116417:..............PUB     SY
116433(000201): 051440 020040 020040 000401 010314 100061 001400 000000 000014 000000 000014 000000  116433:S     .....1.............
116447(000215): 000014 004000 000767 000000 000001 000400 000200 006150 000000 000000 000000 037401  116447:..............h.......I.
116463(000231): 000001 002001 002424 000000 000001 000000 000000 043103 047520 054440 020040 000000  116463:..............FCOPY   ..
116477(000245): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 177777  116477:........................
116513(000261): 177777 000000 000000 000000 000000 000000                                            116513:............
116521: 000000 000000 000000 000000 000000 000000 000000 000000  116531: 000000 000000 000000 000000 000000 000000 000000 140020
116541: 000155 000001 002244 001700 000024 001412 000000 000000  116551: 177773 022123 052104 046111 051524 000111 000045 000040
116561: 000155 000002 002614 001701 000000 000000 000000 000000  116571: 177762 022123 052104 046111 051524 000111 000045 000040
116601: 000000 002001 177766 000010 000401 000156 000004 000000  116611: 000000 000000 000310 000000 000000 000000 000020 000000
116621: 000013 003001 000400 000013 000001 050125 041040 020040  116631: 020040 051531 051440 020040 020040 000401 032512 000000
116641: 000000 000000 000000 000000 000000 000000 000000 000000  116651: 000000 000000 000000 000000 000000 000000 000000 000000
116661: 000000 000000 000000 000000 000000 000000 000000 000000  116671: 000000
------- AVAILABLE FILE TABLE:         FNUM   FTYPE  $NULL    PACB V     LACB V    IOQX
116672: 000000 000153 000000 000000      6    FILE            0  153     0    0
116676: 000000 000160 000000 000000      5    FILE            0  160     0    0
116702: 000000 000157 000000 000000      4    FILE            0  157     0    0
116712: 000000 000155 002150 000000      2    FILE            0  155     1  150
116716: 000000 000155 000150 000000      1    FILE            0  155     0  150
**PXPOINTERS:
116722: 000000 000514 000572 000602
****DL REGISTER: *-*****************************************************************************************************************
****DB REGISTER: ***************************************************************************************************************
116726(000000): 000002 042105 041125 043440 006520 054456 050125 041056 051531 051415 020040 020040  1167261..DEBUG .PY.PUB.SYS.
116742(000014): 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040  116742:
LINES 116756 - 116771 SAME AS ABOVE
```

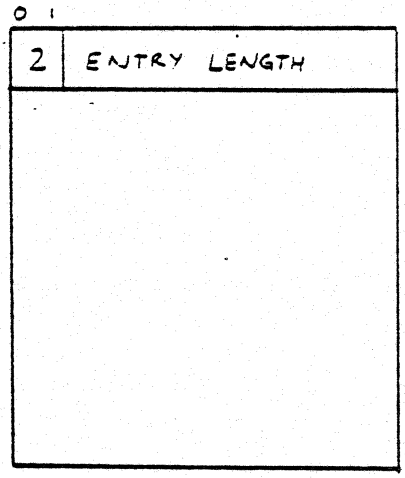
CONTROL BLOCKS



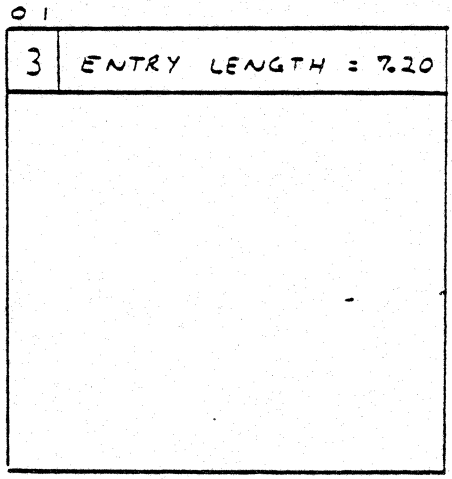
GARBAGE



FCB
(VARIABLE LENGTH FOR EXTENT MAP)

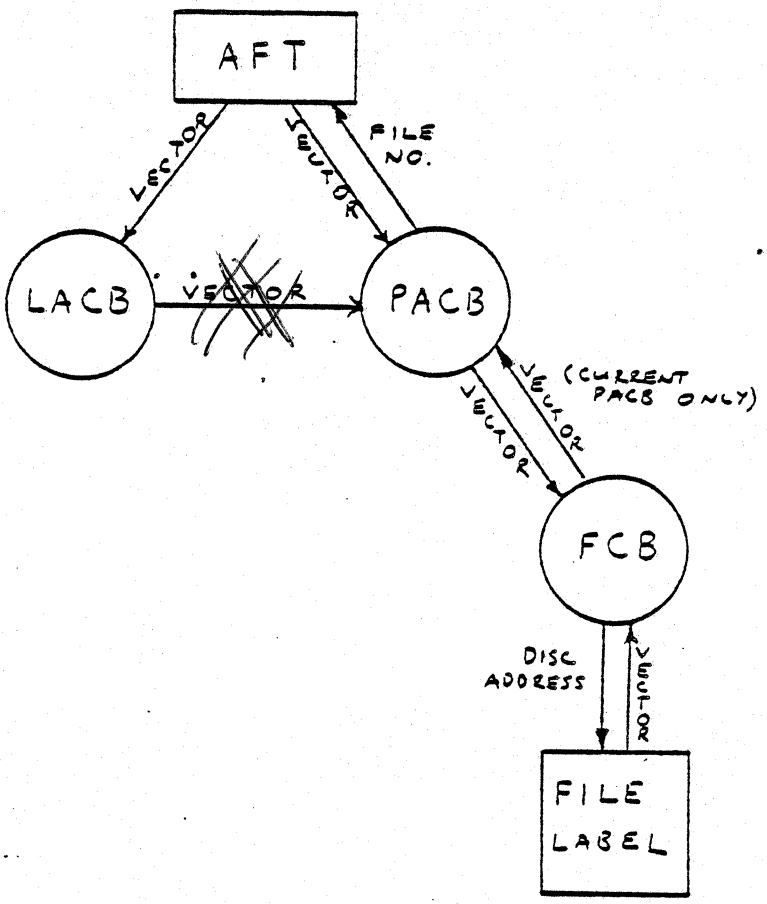


PACB
(VARIABLE LENGTH FOR BUFFER)



LACB

VECTORS



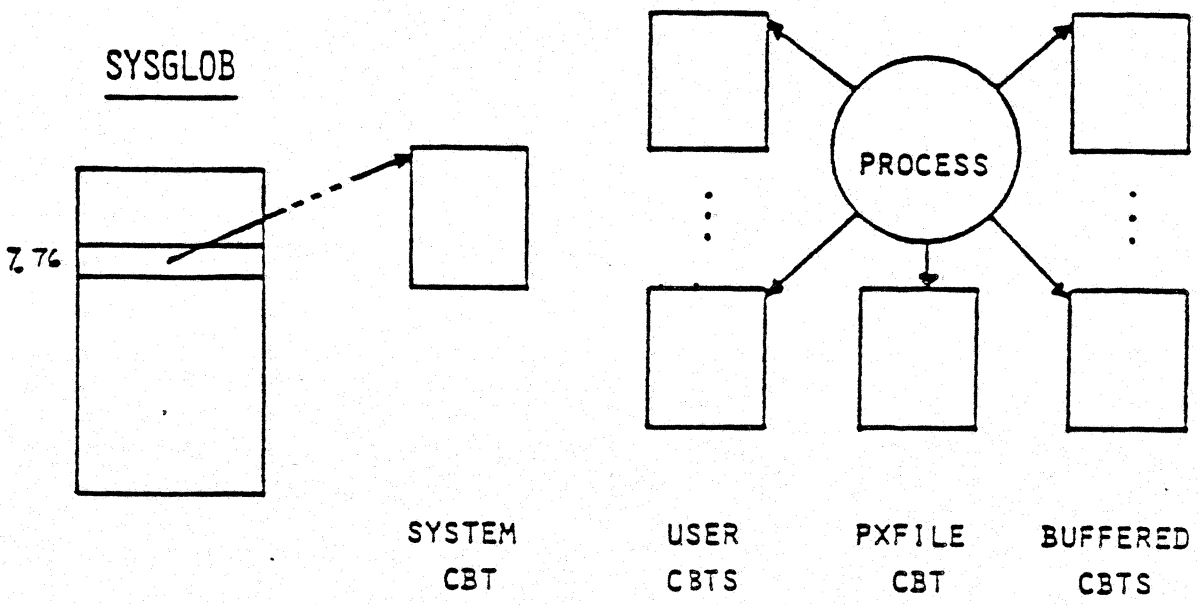
HOW WOULD THE FOLLOWING SITUATIONS AFFECT THIS DIAGRAM:

a) MULTI ?

b) SHARED ?

c) SHARED/MULTI ?

- HOW ARE CONTROL BLOCKS ASSIGNED TO CONTROL BLOCK TABLES?



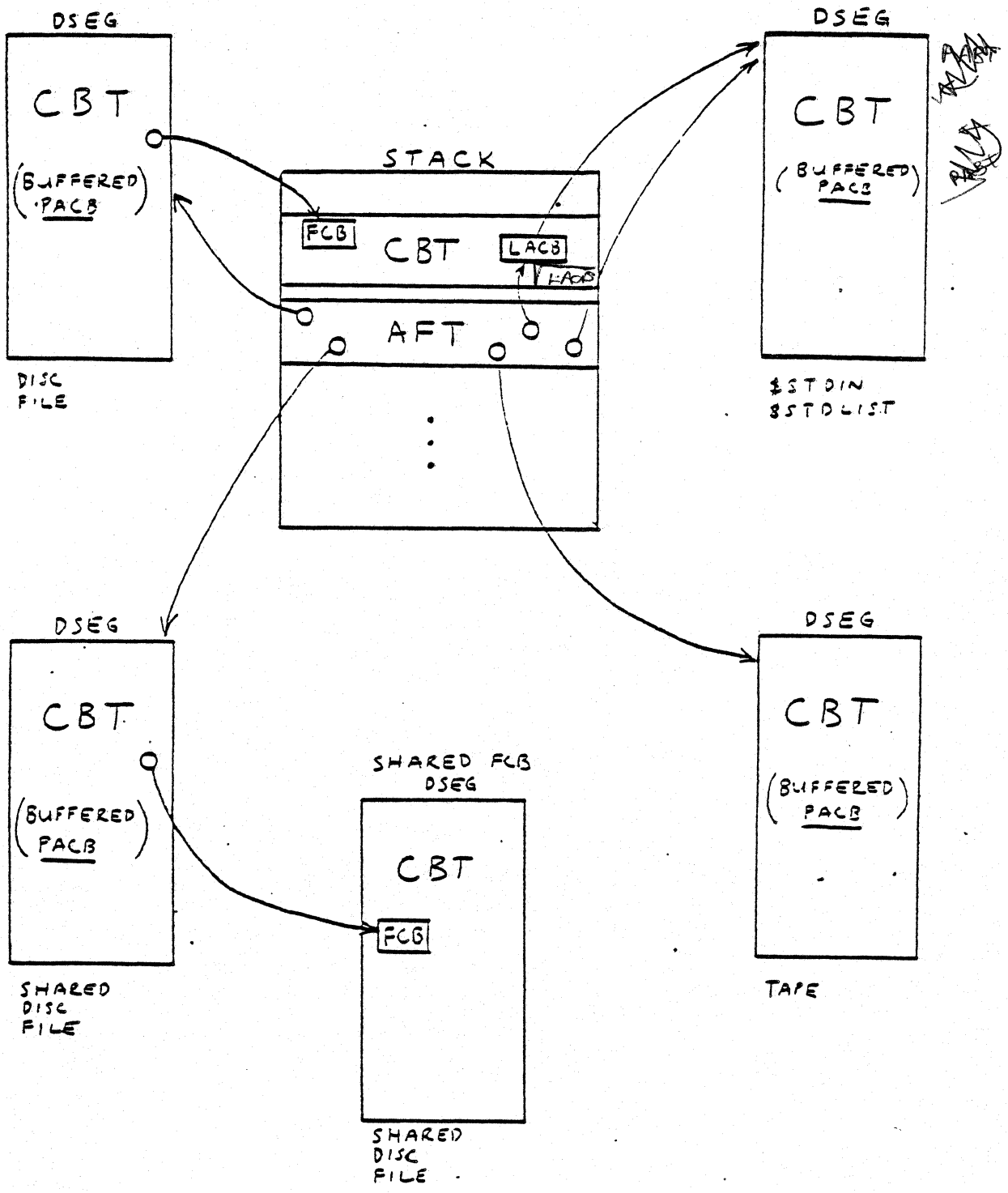
- THERE ARE FOUR CATEGORIES FOR CONTROL BLOCK TABLES:
  1. PXFILE CBT
  2. NOCB OR USER CBTs
  3. BUFFERED ACB CBTs
  4. SHARED FCB OR SYSTEM CBT

- IN ADDITION TO THE PXFILE CBT, A PROCESS MAY HAVE UP TO 8 USER OR NOCB CBTs. THESE HOLD: LACBs, UNBUFFERED PACBs, SYSTEM BUFFERED PACBs AND NON-SHARED FCBs.

- PACBs WITH THEIR OWN BUFFERS ARE PLACED IN SEPARATE, DEDICATED CBTs.

- SHARED FCBs ARE PLACED IN A SEPARATE, DEDICATED CBT CALLED THE SYSTEM OR SHARED FCB CBT

TYPICAL CBT's



* WHAT IS THE CB ASSIGNMENT STRATEGY?

* IF THE USER DID NOT SAY ";NOCB" THEN TRY TO PUT THE CB IN THE PXFILE AREA.

* IF IT WON'T FIT THERE, GO TO ONE OF THE USER CBT'S.

* IF IT WON'T FIT IN ANY CURRENTLY EXISTING USER CBT THEN CREATE ANOTHER ONE UP TO A MAXIMUM OF 8 (8 CBT GIVES ROOM FOR 253 FILES WHICH IS MAX).

* IN THE CASE OF ACB's WITH ACB BUFFERS, THEY ALWAYS GO INTO AN EXTRA DATA SEGMENT. BUFFERED ACB's ARE CREATED ONLY AS LONG AS NECESSARY WHEREAS A USER CBT IS CREATED WITH AN INITIAL SIZE OF 1024 AND MAXDATA OF 8K.

- WHERE ARE FILE LABELS AND WHEN ARE THEY UPDATED?

- FILE LABELS ONLY RESIDE ON THE FIRST SECTOR OF THE FILE.

- NO FILE LABEL EXISTS AT THE START OF EXTENTS REGARDLESS OF WHERE THEY RESIDE, i.e., ON THE SAME LDEV OR A DIFFERENT LDEV. (RECALL THAT IF AN LDEV IS SPECIFIED WHEN THE FILE IS CREATED, THEN ALL EXTENTS RESIDE ON THAT ONE LDEV. IF DEVICE CLASS IS SPECIFIED THEN ANY DISC OF THAT CLASS).

- THE FCB IS CREATED WHEN THE FILE IS OPENED BY THE FIRST ACCESSOR AND DESTROYED AFTER THE LAST FCLOSE AFTER BEING USED TO UPDATE THE FILE LABEL.

- WRITING THE FILE LABEL CAN BE FORCED BY ISSUING AN FCONTROL (6) (WRITE EOF).

FCB PLACEMENT

- FOPEN OF NEW DISC FILE:

  - IF MULTIACCESS, USE SYSTEM CONTROL BLOCK TABLE
  - IF NOT MULTI, USE PXFILE CBT

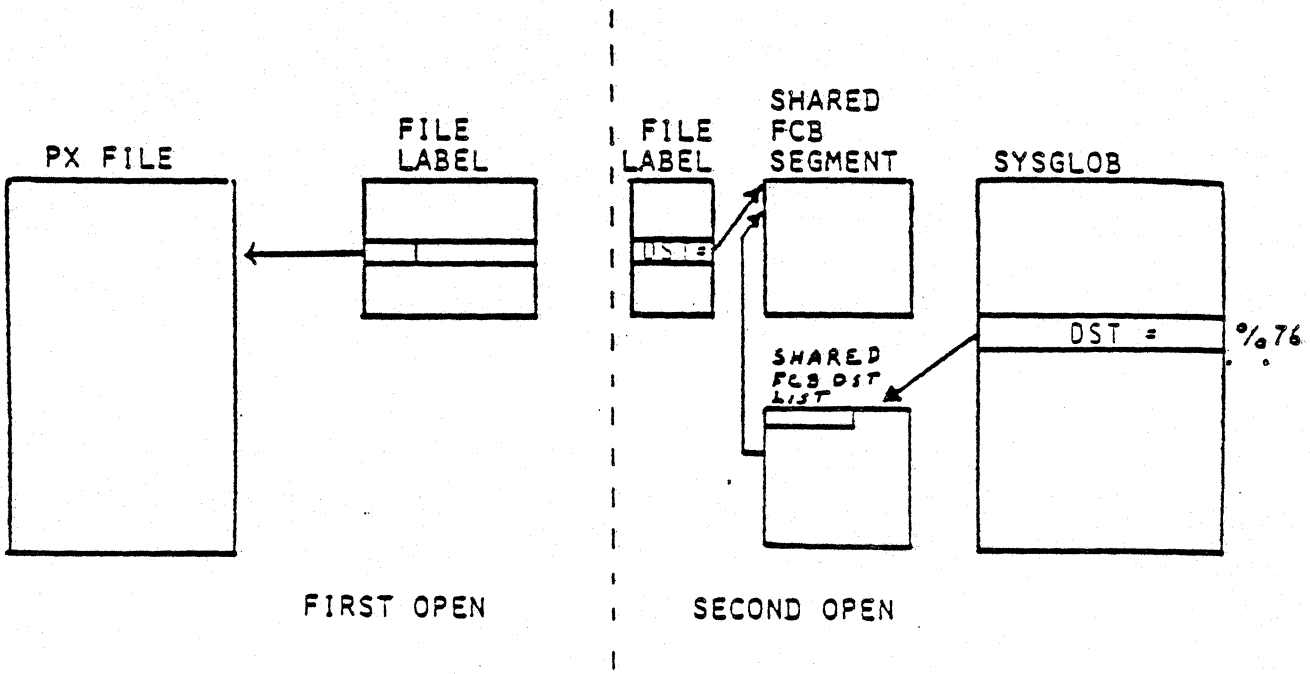
- FOPEN OF OLD DISC FILE:

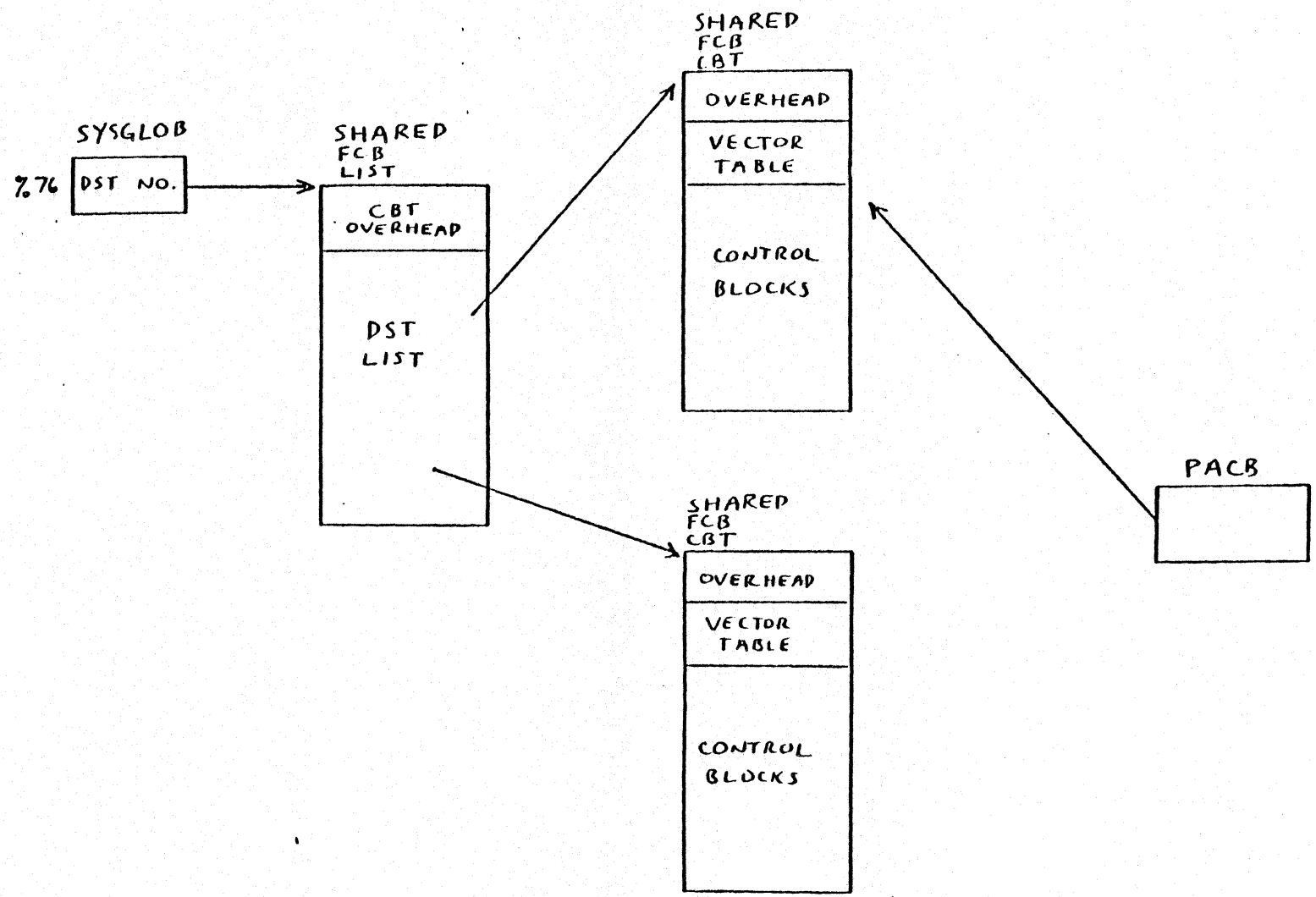
  - IF FCB DOES NOT ALREADY EXIST:

    IF EXCLUSIVE, PUT FCB IN PXFILE

    IF NOT EXCLUSIVE, PUT FCB IN SYSTEM CBT

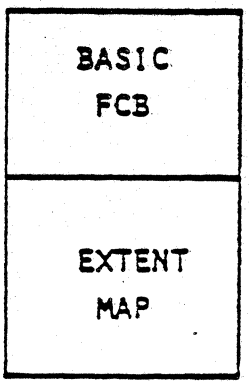
* HOW ARE FCB's FOR SHARED/MULTIACCESS FILES HANDLED?

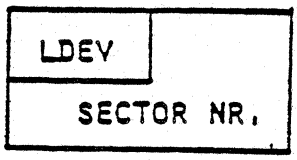



SYSGLOB
%76
DST NO.
SHARED
FCB
LIST
CBT
OVERHEAD
DST
LIST
SHARED
FCB
CBT
OVERHEAD
VECTOR
TABLE
CONTROL
BLOCKS
SHARED
FCB
CBT
OVERHEAD
VECTOR
TABLE
CONTROL
BLOCKS
PACB

SHARED FCB's

- WHAT DOES AN FCB LOOK LIKE?





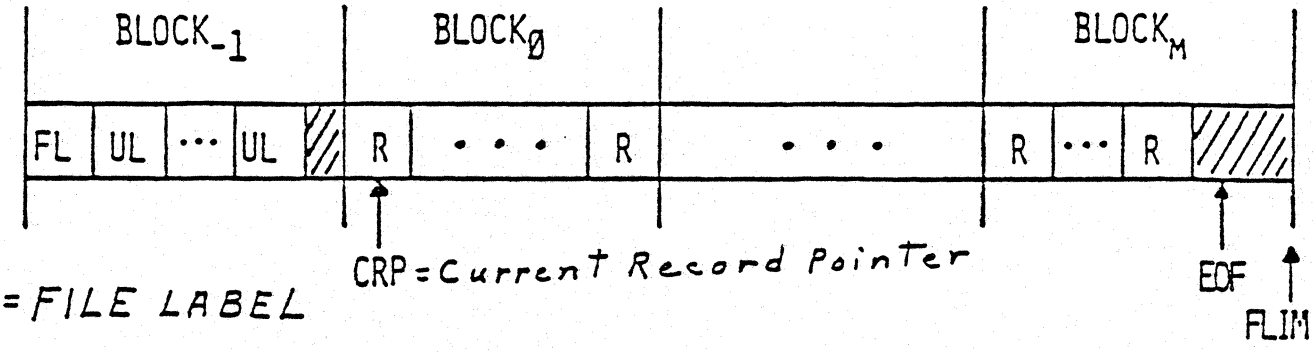
EXTENT DESCRIPTOR

- RECALL THAT AN FCB EXISTS FOR A DISC FILE ONLY
- THE FCB IS ESSENTIALLY A RUN-TIME DISTILLATION OF THE INFORMATION CONTAINED IN THE DISC FILE LABEL
- THE FCB EXISTS BECAUSE IT IS FASTER AND EASIER TO ACCESS THAN THE DISC FILE LABEL
- THE FCB CONTAINS INFORMATION SUCH AS:

GROUP NAME	USER LABLE INFO
ACCOUNT NAME	NR. OPENS INPUT MODE
NUMBER OF EXTENTS	NR. OPENS OUTPUT MODE
EXTENT SIZE	NR. OPENS ANY MODE
LAST EXTENT SIZE	END OF DATA POINTER
DEVICE TYPE	FILE LIMIT
FOPTIONS	RIN NUMBER
BLOCKING FACTOR	
SECTORS PER BLOCK	

- THE EXTENT MAP CONTAINS ONLY EXTENT DESCRIPTORS, EACH OF WHICH DESCRIBES THE LOCATION OF A SINGLE EXTENT

* WHAT IS THE FORMAT OF A TYPICAL DISC FILE?

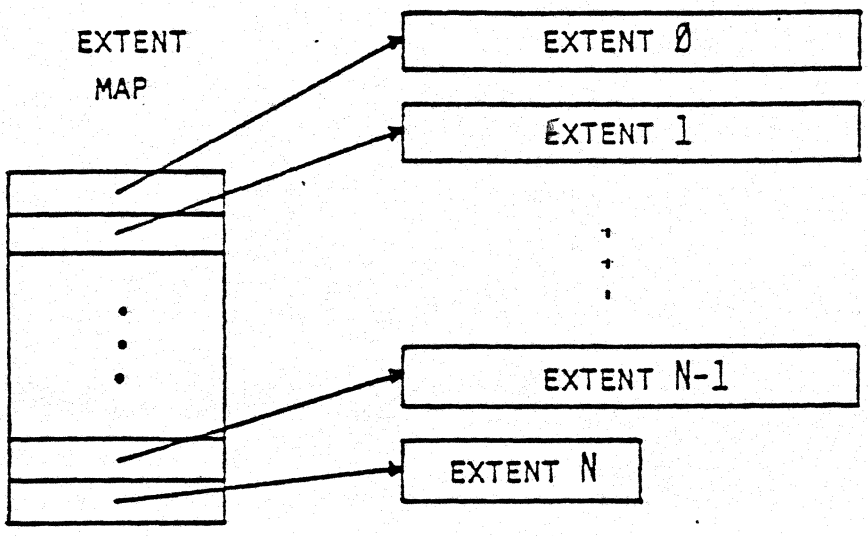


FL=FILE LABEL

UL=USER LABEL

LOGICAL VIEWPOINT

FLIM = FILE LIMIT



PHYSICAL VIEWPOINT

* THE LOGICAL FILE IS MAPPED INTO THE PHYSICAL FILE VIA THE CONSTRUCTS BLOCK AND EXTENT

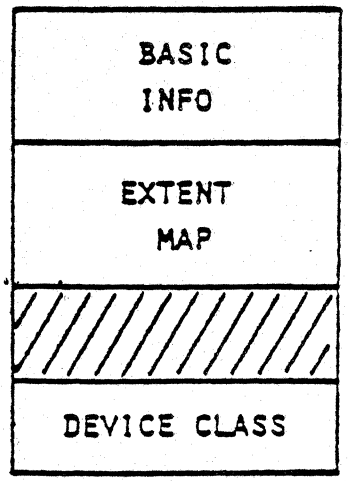
* FOR FIXED AND UNDEFINED FORMAT RECORDS A BLOCK CONSISTS OF AN INTEGRAL NUMBER OF RECORDS (BLOCKING FACTOR); FOR VARIABLE FORMAT RECORDS A BLOCK CONSISTS OF A VARIABLE NUMBER OF RECORDS

* EACH BLOCK OCCUPIES AN INTEGRAL NUMBER OF SECTORS (SECTORS/ BLOCK)

* EACH EXTENT CONSISTS OF AN INTEGRAL NUMBER OF BLOCKS

* EACH EXTENT HAS THE SAME LENGTH, EXCEPT (POSSIBLY) THE LAST

- WHAT DOES A FILE LABEL LOOK LIKE?



- THE PURPOSE OF A FILE LABEL IS TO MAKE A FILE SELF-IDENTIFYING

- AT PRESENT, ONLY DISC FILES HAVE LABELS

- THE INFORMATION CONTAINED IN A FILE LABEL IS ESSENTIALLY THE SAME AS THAT CONTAINED IN AN FCB

- SPECIFICALLY, A FILE LABEL CONTAINS:

LOCAL NAME	FOPTIONS
GROUP NAME	RECORD SIZE
ACCOUNT NAME	BLOCK SIZE
LOCKWORD	USER LABEL LIMIT
CREATION DATE	USER LABEL EOF
LAST ACCESS DATE	FILE LIMIT
LAST MODIFICATION DATE	DEVICE TYPE NR.
FILE CODE	DEVICE SUBTYPE NR.
SECURITY MATRIX	NUMBER OF EXTENS
DEVICE CLASS	EXTENT SIZE
	LAST EXTENT SIZE
	FCB VECTOR
	COLD LOAD ID
	CHECKSUM

- HOW ARE THE EXTENTS HANDLED?

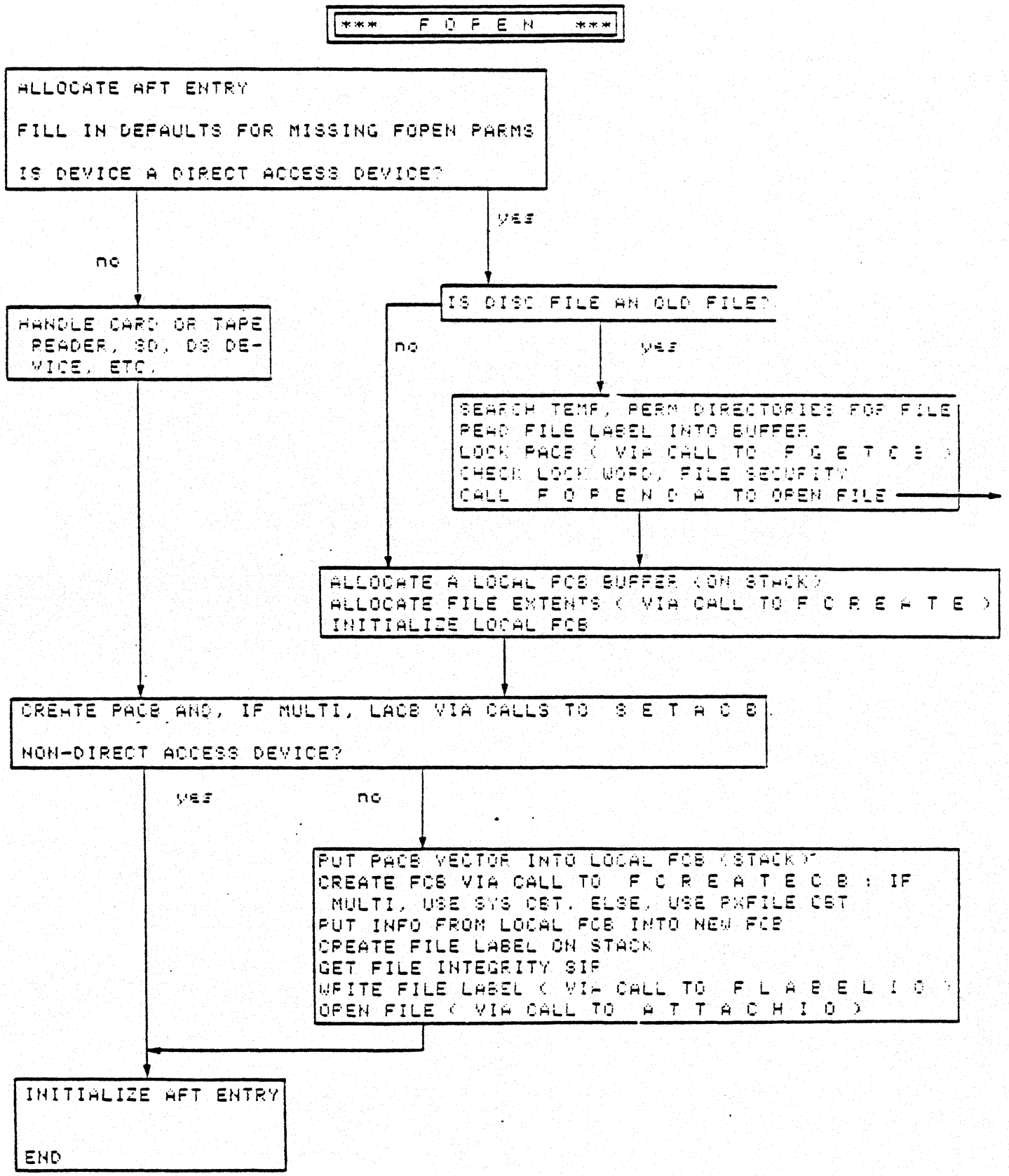
- THE MAXIMUM NUMBER OF EXTENTS FOR A FILE IS KEPT IN THE FLAB AND FCB.

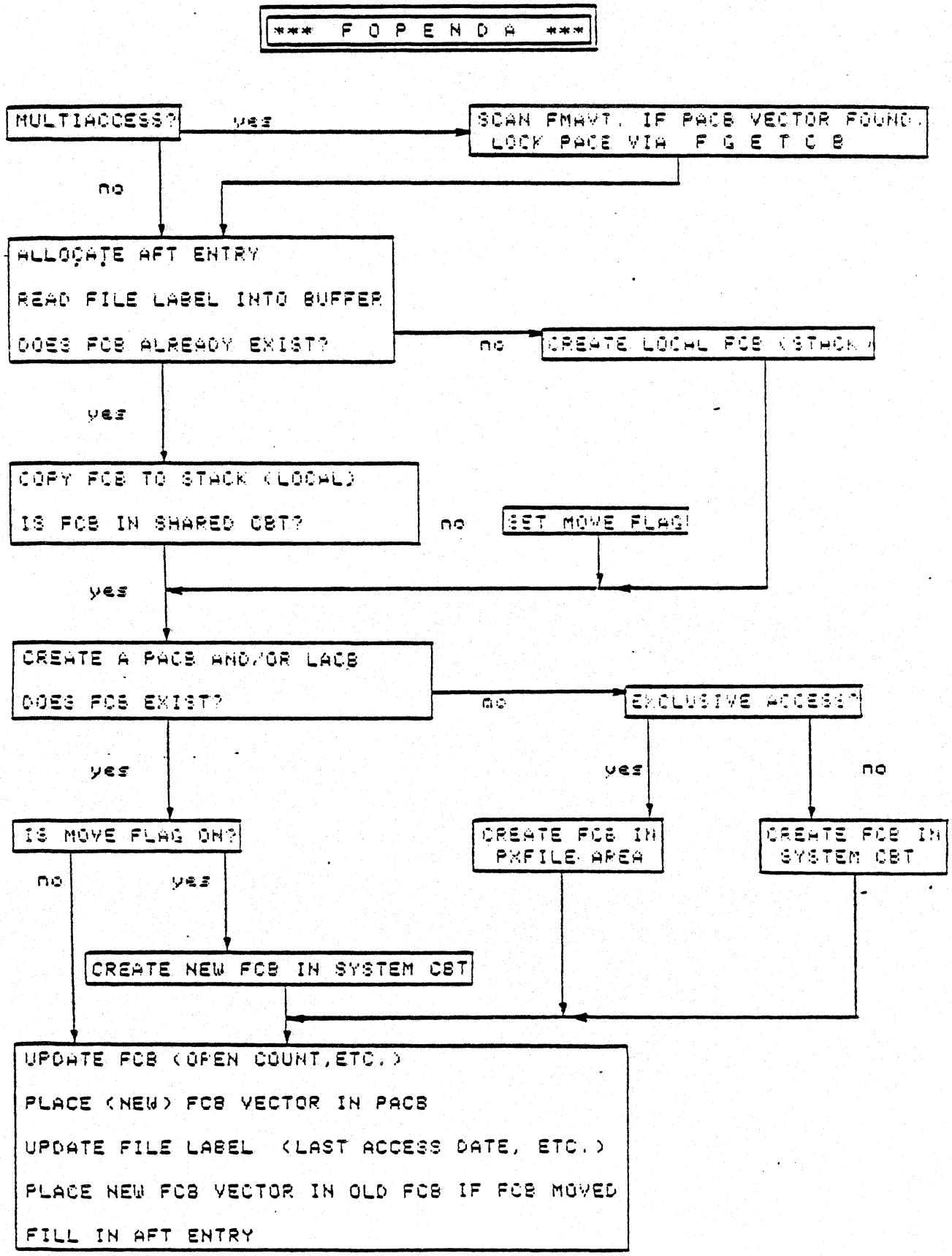
- THE ACTUAL NUMBER OF EXTENTS ALLOCATED IS FOUND BY FLAB.(39).(11:5)=NUMEXT-1 AND FCB.

- THE SIZE OF THE LAST EXTENT IS FOUND IN FLAB.(40) AND FCB

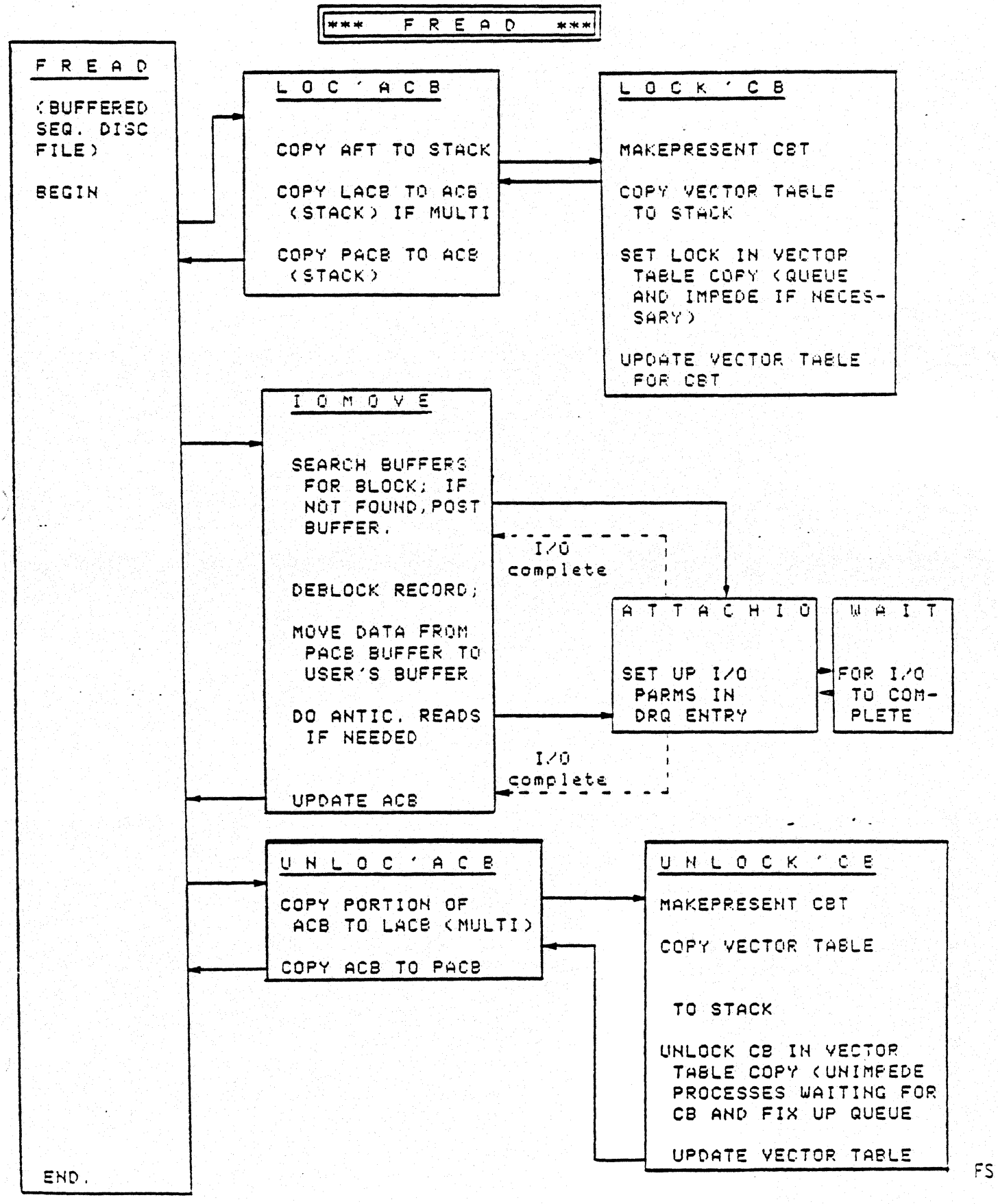
* WHAT ARE THE SYSGLOB LOCATIONS USED BY THE FILE SYSTEM?

* % 75 - COLDLOAD COUNT.
* % 76 - SHARED FCB DST LIST.
* % 77 - MONITORING FLAG WORD.
* % 100-101 - MAX NUMBER SPOOLFILE KILOSECTORS.
* % 102-103 - CURRENT NUMBER SPOOLFILE KILOSECTORS.
* % 104 - NUMBER SECTORS/SPOOLFILE EXTENT.
* % 132 - CLASS SPOOL INDEX (CS).
* % 135 - CSIOWAIT PLABEL (CS).
* % 140 - CS CCLOSE PLABEL (CS).
* % 323 - SDSLDEV PLABEL (DS).
* % 335 - DSCHECK PLABEL (DS).
* % 336 - DSOPEN PLABEL (DS).
* % 337 - DSCLOSE PLABEL (DS).
* % 340 - MANAGEWRITECONV PLABEL (DS).


*** F O P E N ***
ALLOCATE AFT ENTRY
FILL IN DEFAULTS FOR MISSING FOPEN PARMS
IS DEVICE A DIRECT ACCESS DEVICE?
yes
no
IS DISC FILE AN OLD FILE?
yes
no
HANDLE CARD OR TAPE READER, SD, DS DEVICE, ETC.
SEARCH TEMP, PERM DIRECTORIES FOR FILE
READ FILE LABEL INTO BUFFER
LOCK PACB (VIA CALL TO F G E T C B)
CHECK LOCK WORD, FILE SECURITY
CALL F O P E N C A TO OPEN FILE
ALLOCATE A LOCAL FCB BUFFER (ON STACK)
ALLOCATE FILE EXTENTS (VIA CALL TO F C R E A T E)
INITIALIZE LOCAL FCB
CREATE PACB AND, IF MULTI, LACB VIA CALLS TO S E T A C B
NON-DIRECT ACCESS DEVICE?
yes
no
PUT PACB VECTOR INTO LOCAL FCB (STACK)
CREATE FCB VIA CALL TO F C R E A T E C B ; IF MULTI, USE SYS CBT. ELSE, USE PXFILE CBT
PUT INFO FROM LOCAL FCB INTO NEW FCB
CREATE FILE LABEL ON STACK
GET FILE INTEGRITY SIP
WRITE FILE LABEL (VIA CALL TO F L A B E L I O)
OPEN FILE (VIA CALL TO A T T A C H I O)
INITIALIZE AFT ENTRY
END


*** F O P E N D A ***
MULTIACCESS?
yes
SCAN FMAVT, IF PACB VECTOR FOUND, LOCK PACE VIA F G E T C B
no
ALLOCATE AFT ENTRY
READ FILE LABEL INTO BUFFER
DOES FCB ALREADY EXIST?
no
CREATE LOCAL FCB (STACK)
yes
COPY FCB TO STACK (LOCAL)
IS FCB IN SHARED CBT?
no
SET MOVE FLAG!
yes
CREATE A PACB AND/OR LACB
DOES FCB EXIST?
no
EXCLUSIVE ACCESS?
yes
yes
no
IS MOVE FLAG ON?
CREATE FCB IN PXFILE AREA
CREATE FCB IN SYSTEM CBT
no
yes
CREATE NEW FCB IN SYSTEM CBT
UPDATE FCB (OPEN COUNT,ETC.)
PLACE (NEW) FCB VECTOR IN PACB
UPDATE FILE LABEL (LAST ACCESS DATE, ETC.)
PLACE NEW FCB VECTOR IN OLD FCB IF FCB MOVED
FILL IN AFT ENTRY

CALLABLE FILE SYSTEM INTRINSICS



FS

I O M O V E

- DOES RECORD I/O
- HOB IS ASSUMED TO LIE DIRECTLY BENEATH EXPLICIT PARMS TO I O M O V E
- MUST NOT STACK DATA BEFORE CALLING I O M O V E !

I O M O V E

READ

UNBUFFERED

F C O N V ' B L K (IF DISC)
(DETERMINES DISC ADDRESS OF SPECIFIED BLOCK. IF IN NON-EXISTING EXTENT, CREATE EXT.

A T T A C H I O
(SET UP REQUEST TO READ FROM DISC ADDRESS TO TARGET)

HANDLE NOWAIT I/O. (SAVE IOQX IN AFT)

HANDLE TAPE LABELS REEL SWITCH

CHECK FOR EOF

HANDLE VARIABLE RECORDS

HANDLE MULTIRECORD

BUFFERED

HANDLE VARIABLE RECORDS

SEARCH BUFFERS FOR BLOCK.
(IF FIRST FREAD, LAUNCH PREFEADS FOR ALL BUFFERS STARTING WITH NEEDED BLOCK. IF NOT FIRST FREAD AND THERE IS NOT AN EMPTY BUFFER, USE CURRENT BUFFER, WRITING OUT BUFFER IF DIRTY.

DEBLOCK RECORD

HANDLE UNDEFINED RECORDS

MOVE DATA FROM PAGE BUFFER TO USER BUFFER (MOS INST)

DO ANTICIPATORY READ, IF REQUIRED

UPDATE ACB

HANDLE MULTIRECORD

I O M O V E

WRITE

UNBUFFERED

F C O N V ' S B L K (IF DISC)
(DETERMINE DISC ADDRESS)

$ T T A C H I O
(SET UP WRITE REQUEST)

HANDLE NOWAIT I/O
(SAVE IOSB IN AFT)

HANDLE TAPE LABELS REEL SWITCH

CHECK FOR REEL SWITCH

CHECK FOR EOF

HANDLE VARIABLE RECORDS

BUFFERED

HANDLE VARIABLE RECORDS

SEARCH BUFFERS FOR BLOCK. IF THERE IS NO BUFFER, START A NEW ONE. IF BLOCK IS NOT FOUND AND THERE IS A CURRENT BUFFER, WRITE OUT IF DIRTY: A T T A C H I O
EITHER START A NEW BLOCK OR BRING IN BLOCK TO BE MODIFIED: A T T A C H I O

HANDLE VARIABLE RECORDS

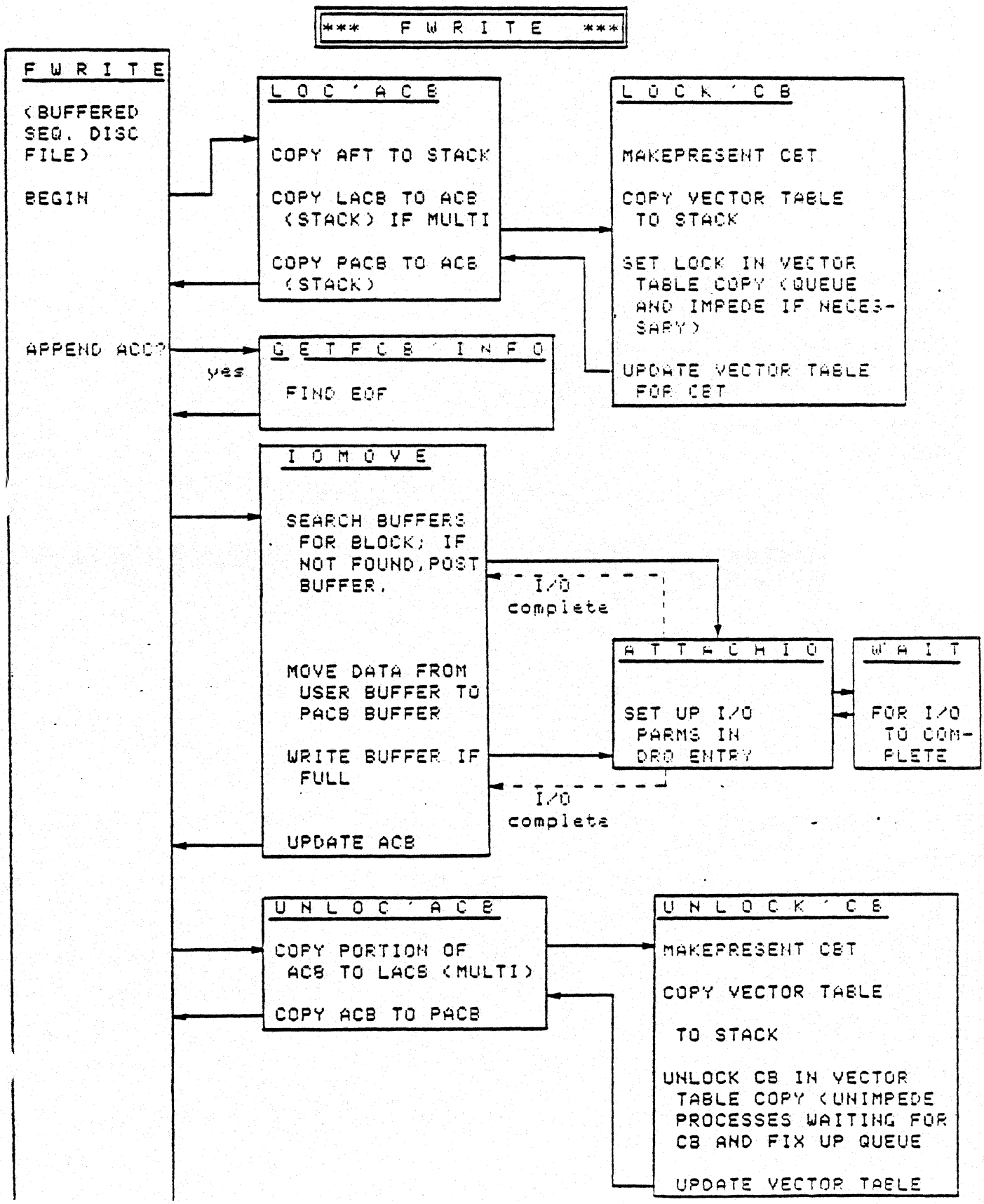
MOVE DATA FROM USER'S BUFFER TO PACB BUFFER (MOS INST.)

WRITE BUFFER. IF FULL: A T T A C H I O

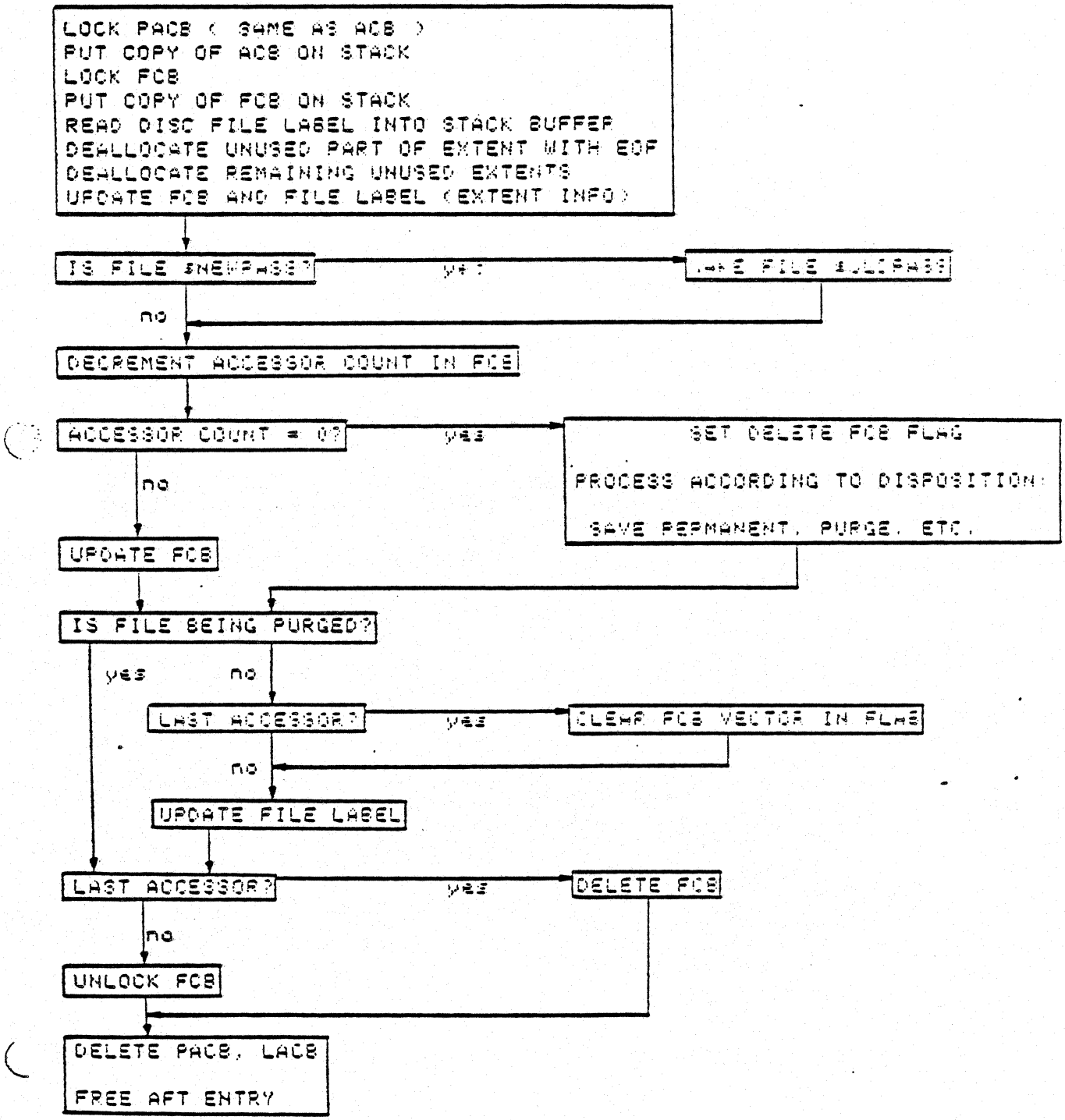
UPDATE ACB (ON STACK): FILE POINTER, TRANSFER COUNT, ETC

HANDLE MULTIRECORDS

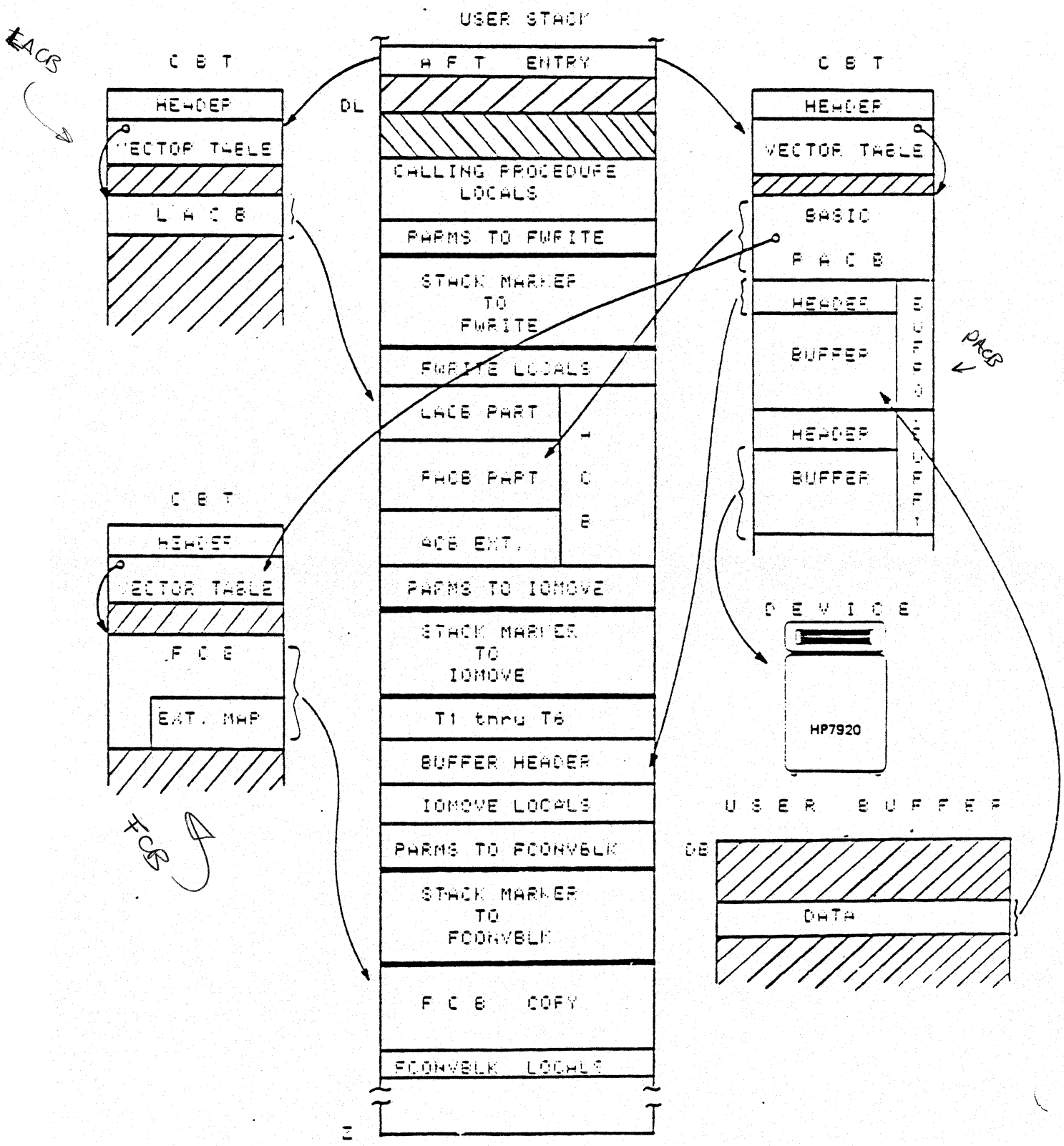
CALLABLE FILE SYSTEM INTRINSICS



F C L O S E


LOCK PACB (SAME AS ACB)
PUT COPY OF ACB ON STACK
LOCK FCB
PUT COPY OF FCB ON STACK
READ DISC FILE LABEL INTO STACK BUFFER
DEALLOCATE UNUSED PART OF EXTENT WITH EOF
DEALLOCATE REMAINING UNUSED EXTENTS
UPDATE FCB AND FILE LABEL (EXTENT INFO)
IS FILE SNEWFASE?
yes
no
DECREMENT ACCESSOR COUNT IN FCB
ACCESSOR COUNT = 0?
yes
SET DELETE FCB FLAG
PROCESS ACCORDING TO DISPOSITION:
SAVE PERMANENT, PURGE, ETC.
no
UPDATE FCB
IS FILE BEING PURGED?
yes
no
LAST ACCESSOR?
yes
CLEAR FCB VECTOR IN FLAGS
no
UPDATE FILE LABEL
LAST ACCESSOR?
yes
DELETE FCB
no
UNLOCK FCB
DELETE PACB, LACB
FREE AFT ENTRY

F I L E S Y S T E M C O N T R O L B L O C K S



FILE SYSTEM

LAB PROJECT

1. WRITE A SMALL PROGRAM WHICH EXECUTES THE FOLLOWING SEQUENCE:

   A. FOPEN AN OLD FILE WITH SHARED ACCESS.
   B. FOPEN THE SAME OLD FILE WITH SHARED ACCESS A SECOND TIME.
   C. CALL DEBUG.

2. RUN THE PROGRAM WITH NOCB.

3. WHEN THE PROGRAM HALTS WITH THE DEBUG CALL FIND ALL OF THE CONTROL BLOCKS CONNECTED WITH EACH OF THE TWO FILES YOU OPENED AND THE CONTROL BLOCKS FOR $STDIN AND $STDLIST. ON ANOTHER PAPER LIST THE STEPS YOU USED TO FIND THE CONTROL BLOCKS IN SUFFICIENT DETAIL THAT YOU CAN USE IT FOR FUTURE REFERENCE. YOU MAY WANT TO MAKE LINE PRINTER LISTINGS WITH DEBUG AND MARK THE POINTERS YOU FOLLOWED.

SUMMARY

A) CHECK FOR UNDERSTANDING:

1) HOW IS THE SYSTEM DIRECTORY ARRANGED?
ANS:_____
_____

2) EVERY DISC ADDRESS IN THE DIRECTORY IS DIRBASE RELATIVE EXCEPT TWO. THEY ARE:
A._____
B._____

3) WHAT IS THE DIVISION POINT BETWEEN THE FILE SYSTEM AND THE I/O SYSTEM?
ANS:_____
_____

4) THERE IS A SYSTEM AREA ABOVE DL IN EVERY PROCESS STACK. IT IS CALLED THE _____ AREA AND IS BROKEN INTO THREE MAIN AREAS CALLED:
A)_____
B)_____
C)_____

5) IF A PROGRAM IS RUN WITH THE PARAMETER ";NOCB", WHERE ARE THE CONTROL BLOCKS FOR $STDIN AND $STDLIST?
ANS:_____
_____

6) WHEN DOES THE FILE SYSTEM DO ANTICIPATORY READS FOR FILES ACCESSED BY THE INTRINSIC FREADDIR?
ANS:_____
_____
_____

7) WHEN ARE THE FILE LABELS UPDATED?
ANS:_____
_____
_____

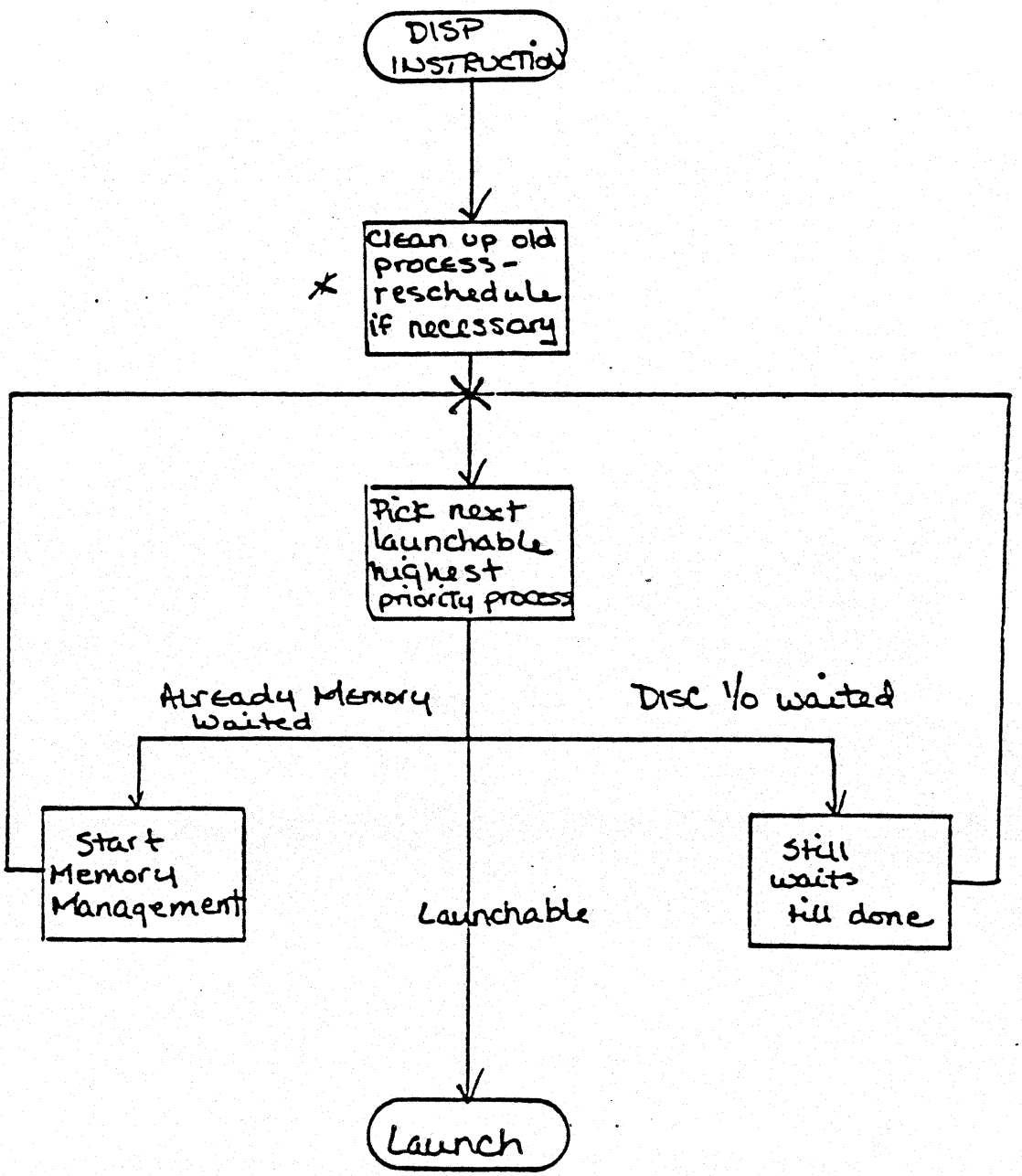
8) WHAT ARE THE TWO FORMATS OF THE FCB VECTOR WORD (27) IN THE FILE LABEL AND WHAT DO THEY MEAN?

A) PACB = DST / ety

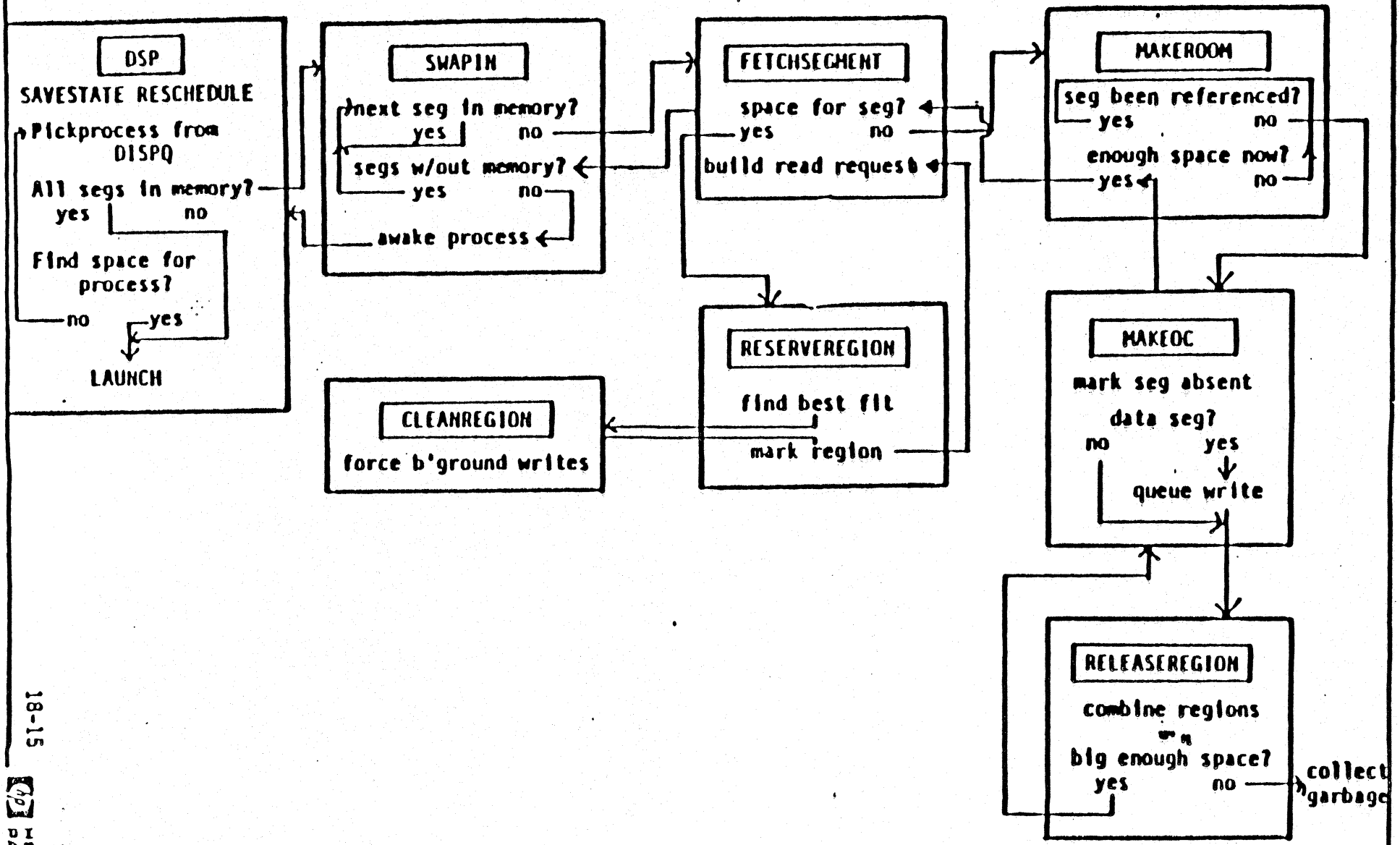
B)

MPE IV

DISPATCHER


DISP INSTRUCTION
Clean up old process - reschedule if necessary
Pick next launchable highest priority process
Already Memory Waited
DISC I/O waited
Start Memory Management
Still waits till done
Launchable
Launch

LAUNCHING A PROCESS



HEWLETT hp PACKARD

MPE IV

RESCHEDULING OF PROCESSES FOR THE CPU

WHEN DOES A PROCESS GIVE UP THE CPU?

PROCESS WILL RUN UNTIL IT:

- BLOCKS NATURALLY (FOR I/O, MEMORY, ETC.)
- IS PREEMPTED BY A HIGHER PRIORITY PROCESS
- DISPATCHER DECIDES IT HAS HAD THE CPU LONG ENOUGH

HOW DOES A

PROCESS

BLOCK NATURALLY?

MPE IV

WAIT

and

AWAKE

WAIT

* WAIT IS CALLED WHENEVER THE CALLING PROCESS MUST STOP UNTIL AN EVENT OCCURS

* WAKEMASK STORED IN PCB SPECIFIES EVENT TYPE PROCESS WANTS TO BE AWAKENED ON

* CALLER MAY REQUEST THAT CONTROL BE RETURNED IMMEDIATELY IF AN AWAKE ON THE EVENT TYPE HAS ALREADY OCCURRED (BY CHECKING EVENTMASK OF PCB)

AWAKE

* AWAKE IS CALLED TO AWAKEN A PROCESS FROM A WAIT

* IF PROCESS TO BE AWAKENED IS NOT WAITING UPON THE EVENT SPECIFIED IN THE EVENTMASK SPECIFIED BY AWAKE, ~~THE EVENT-MASK IS STORED IN THE PCB~~ (INSTEAD OF WWS OF MPE III)

* CALLER MAY REQUEST THAT SHE BE WAITED UPON SUCCESSFUL ACTIVATION OF THE SPECIFIED PROCESS

CONDITION CODES:

CCG - PROCESS TO BE AWAKENED IS ALREADY ACTIVE

CCL - PROCESS IS WAITING, BUT NOT ON THE EVENT SPECIFIED BY AWAKE

CCE - THE PROCESS IS WAITING ON THE SPECIFIED EVENT

WHAT EVENTS WILL A PROCESS BLOCK FOR?

HEWLETT PACKARD

WAIT FLAGS

M: MOURNING WAIT. SET THE FATHER PROCESS WHEN WAITING FOR A SON TO DIE.

RG: GLOBAL RIN WAIT.

RL: LOCAL RIN WAIT.

MA: MAIL WAIT.

BIO: BLOCKED FOR I/O. ~~NOWAIT~~ normal I/O.

IO: WAITFORIO. NOWAIT I/O.

UCP: UCOP WAIT. UCOP REQUEST IS POSTED, NOW WAIT FOR UCOP TO PROCESS IT.

JNK: JUNK WAIT. NONE OF THE ABOVE. OR BELOW. MOST OF THE SYSTEM PROCESSES USE THIS.

MSG: FILE SYSTEM BASIC IPC MESSAGE WAIT.

IMP: PROCESS WAITING TO BE UNIMPEDED.

SON: WAITING FOR THE SON.

FA: WAITING FOR THE FATHER.

TIM: PROCESS WAITING FOR A TIMEOUT.

MEM: PROCESS WAITING FOR MEMORY.

WHEN DOES A PROCESS GIVE UP THE CPU?

PROCESS WILL RUN UNTIL IT:

- BLOCKS NATURALLY (FOR I/O, MEMORY, ETC.)
- IS PREEMPTED BY A HIGHER PRIORITY PROCESS
- DISPATCHER DECIDES IT HAS HAD THE CPU LONG ENOUGH

HOW IS A PROCESS PREEMPTED?

- WHEN DISPATCHER LAUNCHES A PROCESS IT PLACES THE PRIORITY OF THE PROCESS IN A SYSGLOB CELL CALLED SCHEDULEDTOAWAKECOMM CELL.

- WHENEVER A PROCESS IS AWAKENED . . M AN EVENT ITS PRIORITY IS PLACED IN A SYSGLOB CELL CALLED AWAKETOSCHEDULEMESSAGE CELL.

- AWAKE CHECKS TO SEE IF THE PRIORITY OF THE AWAKENED PROCESS IS MORE URGENT THAN THAT OF THE CURRENTLY RUNNING PROCESS.

- IF SO, AWAKE ASSEMBLES A DISP WHICH WILL CAUSE DISPATCHER TO START OVER AND LOOK FOR THE HEAD OF THE DISPATCHING QUEUE.

- THEN THE MOST URGENT PROCESS WILL RUN.

WHEN DOES A PROCESS GIVE UP THE CPU?

PROCESS WILL RUN UNTIL IT:

- BLOCKS NATURALLY (FOR I/O, MEMORY, ETC.)
- IS PREEMPTED BY A HIGHER PRIORITY PROCES
- DISPATCHER DECIDES IT HAS HAD THE CPU LONG ENOUGH

WHEN DOES DISPATCHER DECIDE IF PROCESS HAS HAD THE CPU LONG ENOUGH?

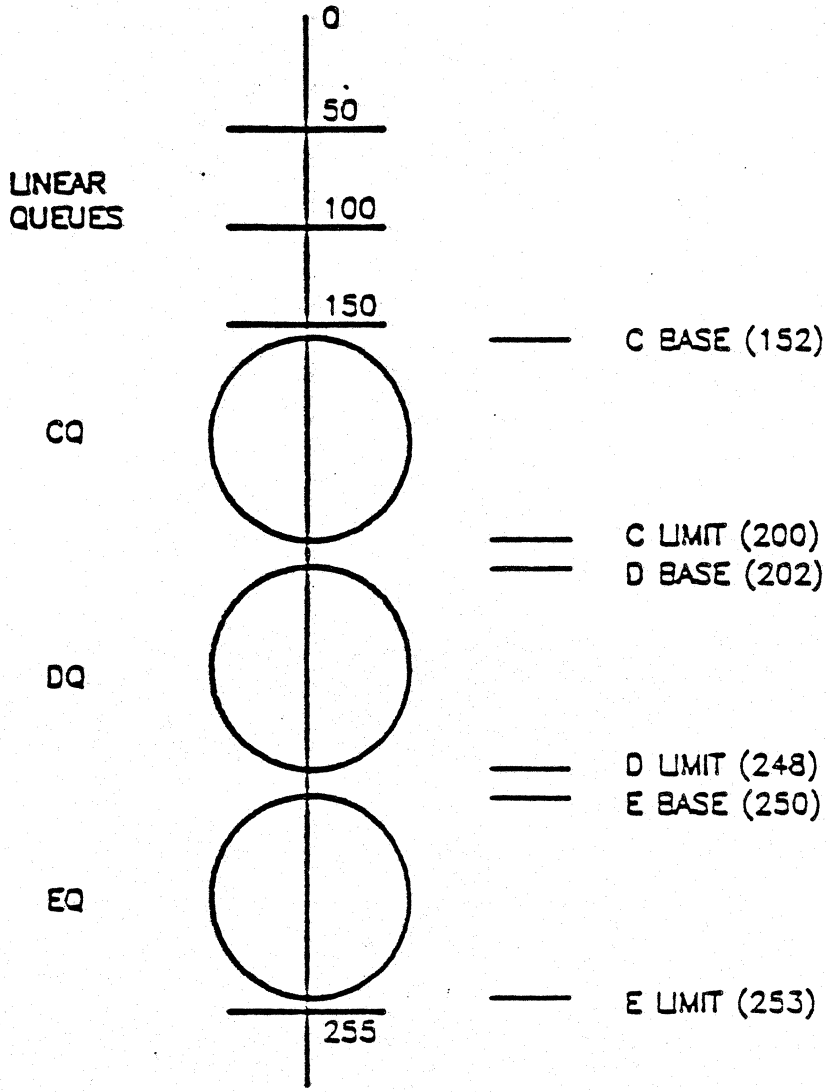
HOW DOES THE DISPATCHER DETERMINE IF A PROCESS WHICH DOES NOT BLOCK AND WHICH IS NOT PREEMPTED SHOULD GIVE UP THE CPU AND (POSSIBLY) HAVE ITS PRIORITY REDUCED?

BY MEANS OF PROCEDURE TICK

HOW DOES DISPATCHER DECIDE IF PROCESS HAS HAD THE CPU LONG ENOUGH?

* WHEN THE PROCESS IS LAUNCHED, QTIME IS SET TO 3
* EVERY TICK OF THE CLOCK (EXT. INTERRUPT), TICK RUNS
* TICK DECREMENTS QTIME
* IF QTIME FALLS TO 0, TICK ASSEMBLES A DISP, AND DISPATCHER RUNS
* DISPATCHER TREATS THE PROCESS JUST AS IF IT BLOCKED OR WAS PREEMPTED- IT SAVES THE STATE OF THE PROCESS, RESCHEDULES IT (PRIORITY MAY CHANGE), AND CHECKS DISPATCHING QUEUE FOR HIGHEST PRIORITY PROCESS (WHICH MAY STILL BE THAT SAME PROCESS)

MPE IV
SCHEDULING QUEUES



NOTE: Parameters are modifiable by :TUNE command only, not by SYSDUMP.

$$150 \leq \begin{pmatrix} \text{C BASE, C LIMIT} \\ \text{D BASE, D LIMIT} \\ \text{E BASE, E LIMIT} \end{pmatrix} \leq 255$$

5108

DISPATCHING QUEUES MPE IV

MPE IV

- PROCESS PRIORITY DOES NOT CHANGE WHEN PROCESS IS CUT IN FRONT BY ANOTHER PROCESS.

- PROCESS PRIORITY DECREASES IF:

  CS: PROCESS CPU TIME EXCEEDS A "AVERAGE SHORT TRANSACTION" SINCE IT LAST HAD ITS PRIORITY REDUCED.

  DS, ES: PROCESS CPU TIME EXCEEDS A "BACKGROUND QUANTUM" SINCE ITS PRIORITY WAS LAST REDUCED.

- PROCESS PRIORITY INCREASES TO "BASE" IF:
  CS,DS,ES: PROCESS ISSUES TERMINAL READ.

CS PROCESS RESCHEDULING

* WHEN PROCESS BEGINS, IT IS GIVEN A PRIORITY OF CBASE
* WHEN PROCESS STOPS (FOR DISC I/O, TERMINAL I/O, PREEMPTION,ETC.) ITS NEW PRIORITY IS DETERMINED SO THAT IT MAY BE RE-QUEUED FOR THE CPU
* DISPATCHER CHECKS TO SEE IF THE PROCESS STOPPED ON A TERMINAL READ.
* IF THE PROCESS STOPPED ON A TERMINAL READ (END OF TRANSACTION) THEN THF DISPATCHER WILL PLACE THE PROCESS PRIORITY AT CBASE.
* IF THE CS PROCESS HAS NOT COMPLETED A TRANSACTION (TERM READ) AND IF THE PROCESS HAS EXCEEDED "AVG. SHORT TRANSACTION" SINCE ITS PRIORITY WAS LAST REDUCED, DISPATCHER WILL PENALIZE THE PROCESS BY DECREASING THE PRIORITY BY 2 (PRI=PRI+2, BUT NOT > C LIMIT). IF IT DID NOT EXCEED THE AVERAGE, IT WILL NOT CHANGE THE PRIORITY UNLESS IT STOPPED ON A MEMORY TRAP, IN WHICH CASE THE PRIORITY WILL BE DECREASED BY 1 (PRI=PRI+1, BUT NOT > CLIMIT).
* IF PROCESS COMPLETED A TRANSACTION, DISPATCHER RECALCULATES THE "AVERAGE SHORT TRANSACTION"

$$\frac{99 * \text{OLD VALUE} + 1 * \text{TRANSACTION}}{100}$$

DS AND ES PROCESS RESCHEDULING

- WHEN PROCESS BEGINS, IT IS GIVEN A PRIORITY OF DBASE OR EBASE
- WHEN PROCESS STOPS ITS NEW PRIORITY IS DETERMINED
- DISPATCHER CHECKS TO SEE IF THE PROCESS STOPPED ON A TERMINAL READ. IF IT DID, PLACE PRIORITY AT D OR EBASE.
- IF THE PROCESS DID NOT STOP ON A TERMINAL READ, DISPATCHER CHECKS TO SEE IF THE PROCESS HAS COMPLETED A BACKGROUNDQUANTUM SINCE LAST RESCHEDULED.
- IF PROCESS HAS COMPLETED A BACKGROUNDQUANTUM SINCE THE LAST TIME THE PROCESS HAD ITS PRIORITY REDUCED, DISPATCHER WILL DECREASE ITS PRIORITY BY 2 (PRI=PRI+2, BUT NOT > DLIMIT OR ELIMIT)
- IF THE PROCESS TRAPPED ON MEMORY, THE PRIORITY WILL BE DECREASED BY 1 (PRI=PRI+1, BUT NOT $\geq$ DLIMIT OR ELIMIT)
- PROCESSES IN D OR E QUEUE MAY OR MAY NOT COMPETE WITH C QUEUE PROCESSES DEPENDING ON SETTING OF C,D,AND E BASE; C,D,AND ELIMIT; AND BACKGROUNDQUANTUM

TUNE COMMAND

* REPLACES THE QUANTUM COMMAND
* OP CAPABILITY REQUIRED
* SYNTAX:

:TUNE [MINCLOCKCYCLE] [(;CQ / ;DQ / ;EQ = [BASE][,[LIMIT][,[MIN][,[MAX]]]])]

MINCLOCKCYCLE – IF CLOCK ALGORITHM CYCLES THROUGH MEMORY IN LESS TIME THAN THIS VALUE (ms), IT WILL GIVE UP (THRASH PREVENTION)

BASE – PRIORITY AT WHICH C,D, OR E PROCESSES WILL BEGIN IN QUEUE

LIMIT – LOWEST (HIGHEST NUMBER) PRIORITY C,D, OR E PROCESSES CAN ATTAIN

MIN, MAX – FOR CS PROCESSES, THE MINIMUM AND MAXIMUM VALUES THE "AVERAGE SHORT TRANSACTION" CAN BECOME

FOR DS AND ES PROCESSES, THE VALUE OF "BACKGROUND QUANTUM". BOTH MUST BE SPECIFIED, BUT IF MIN AND MAX ARE DIFFERENT, MAX IS USED

150< BASE <255 BASE < LIMIT MIN < MAX
LIMIT

```
:SHOWQ
  DORMANT                    WAITING                    RUNNING
Q   PIN     JOBNUM      Q    PIN    JOBNUM      Q    PIN    JOBNUM
L      1                                         C    M36    #S44
L      2
L      3
L      4
L      5
L      6
L      7
L      8
L      9
L      10
C     U11   #S99
L      12
D     M16
L      17
L      18
C     M25   #S69
C     M26   #S76
C     M28   #S102
C     M30   #S85
L      32
C     U34   #S77
D     U35   #J80
C     U37   #S51
C     U39   #S96
C     M40   #S62

CQ MINQUANTUM=0, MAXQUANTUM=300, BASEPRI=152, LIMITPRI=200
DQ MINQUANTUM=1000, MAXQUANTUM=1000, BASEPRI=202, LIMITPRI=238
EQ MINQUANTUM=1000, MAXQUANTUM=1000, BASEPRI=240, LIMITPRI=253
MINIMUM CLOCK CYCLE=1000
```

SHOW Q

* RUNNING - ON THE DISPATCH LIST
  (LINKED THROUGH PCB ENTRIES)

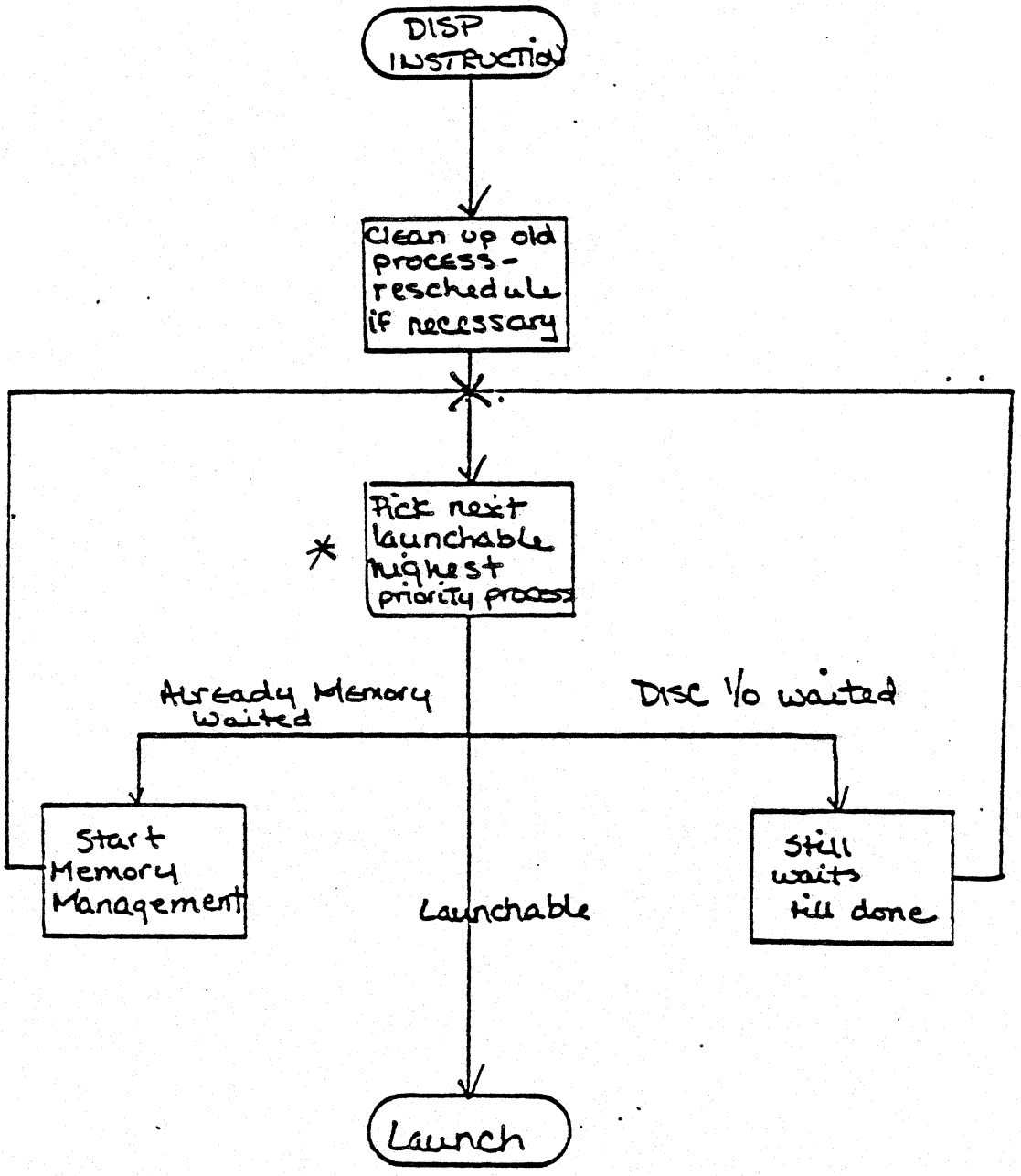
* DORMANT - NOT ON DISPATCH LIST
  (THROWN OFF BY DISPATCHER BECAUSE PROCESS IS WAITING FOR LONG-TERM EVENT).

EXERCISE FOR SHOWQ COMMAND

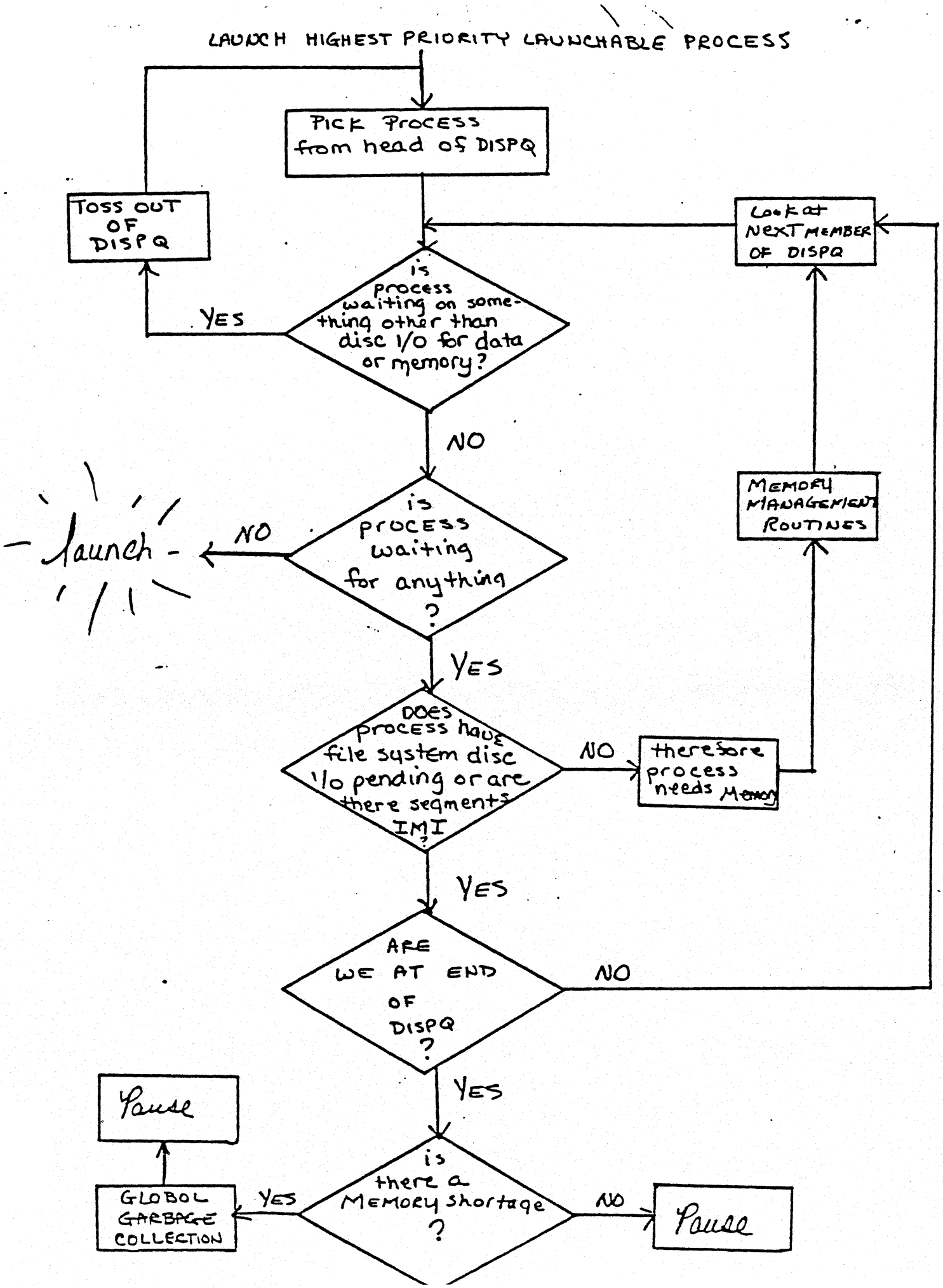
(Reference SHOWQ printout on previous page)

1. What is the value of CBASE?
2. What is the maximum value the average short transaction can become?
3. How many processes are waiting for the CPU?
4. How many command interpreter processes exist on the system?
5. How many processes are in the BS or AS subqueue?
6. How many user processes are there?

DP-12


DISP INSTRUCTION
Clean up old process - reschedule if necessary
Pick next launchable highest priority process
Already Memory Waited
Disc I/o waited
Start Memory Management
Launchable
Still waits till done
Launch

Dispatcher
picks
a process
to be
Launched


LAUNCH HIGHEST PRIORITY LAUNCHABLE PROCESS
PICK PROCESS from head of DISPQ
TOSS OUT OF DISPQ
LOOK AT NEXT MEMBER OF DISPQ
is process waiting on something other than disc I/O for data or memory?
YES
NO
is PROCESS waiting for anything ?
NO
launch
MEMORY MANAGEMENT ROUTINES
YES
DOES process have file system disc I/O pending or are there segments IN?
NO
therefore process needs Memory
YES
ARE WE AT END OF DISPQ ?
NO
YES
is there a MEMORY shortage ?
YES
GLOBAL GARBAGE COLLECTION
Pause
NO
Pause

HEWLETT hp PACKARD

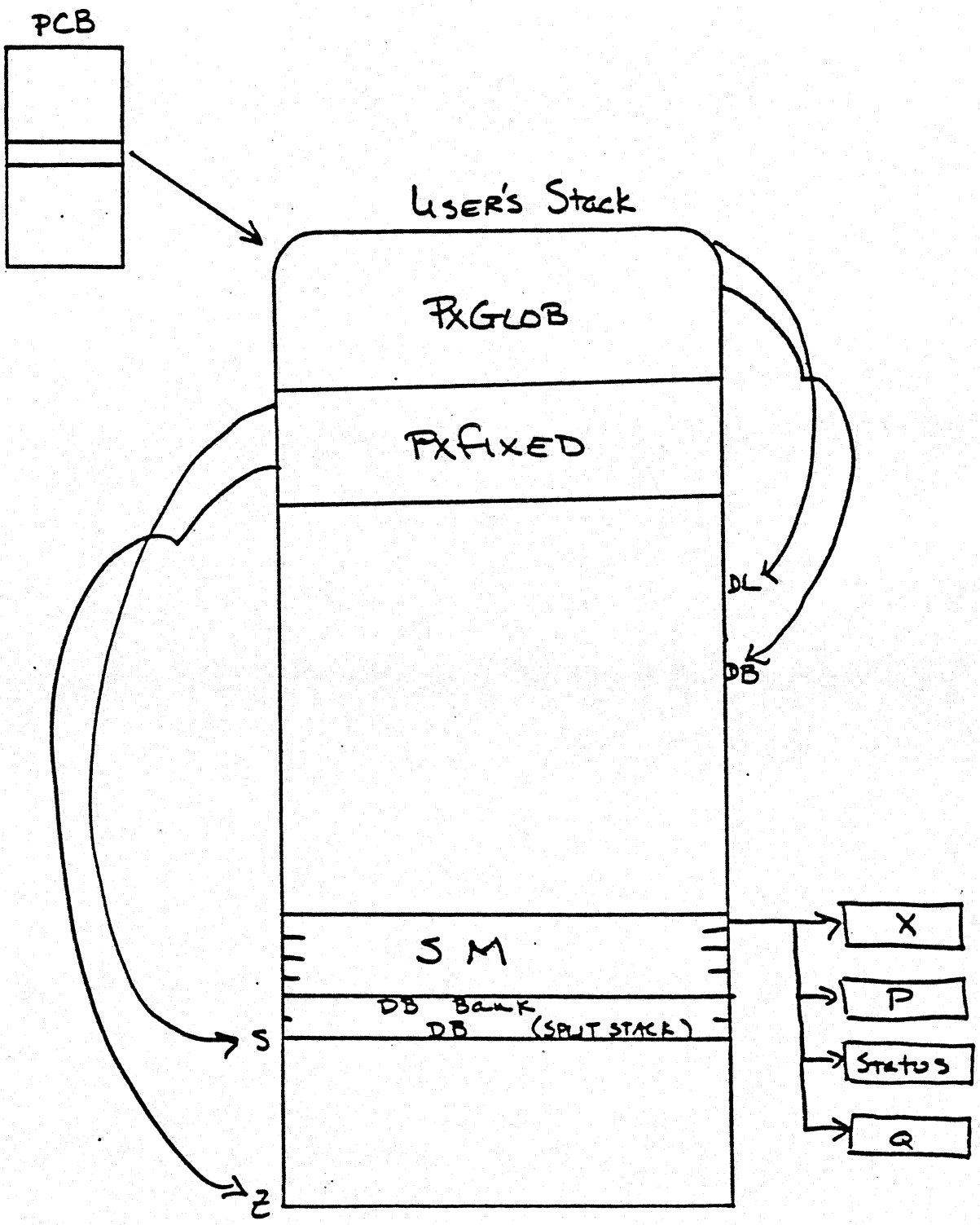
MPE IV

LAUNCHING

A

PROCESS

Process Launch



DISPATCHER

WHEN NO PROCESS IS RUNNING, YOU ARE EITHER EXECUTING AN INTERRUPT HANDLER OR DISPATCHER.

- EXECUTES ON THE ICS
- SCHEDULES PROCESSES IN CQ, DQ, AND EQ
- S ECTS NEXT PROCESS TO RUN
- KEEPS TRACK OF CPU TIME PER PROCESS

SAR FLAG

SCHEDULING ATTENTION REQUIRED
*WHO USES THE SAR FLAG OF THE PCB?

WHEN A PROCESS IS NOT LAUNCHING BECAUSE IT IS WAITING FOR SOMETHING, PROCEDURE DSP (DISPATCHER) CHECKS THE SAR FLAG.

IF IT IS 0 AND THE MEMORY WAIT FLAG IS SET, DSP KNOWS THAT SWAPIN HAS COMPLETED AND HAS CLEARED THE SAR BIT. THE SEGMENTS(S) ARE STILL ON THE WAY IN. IF THERE IS MEMORY PRESSURE, DSP WIL CALL ADJUSTLOCALITY TO REFERENCE THE SEGMENTS IN THE PROCESS'S LOCALITY SO THAT THEY WILL NOT BE EATEN BY THE MEMORY MANAGER BY A LESS URGENT PROCESS.

IF THE SAR BIT IS 1, DSP KNOWS THAT THE PROCESS IS STILL WAITING FOR ATTENTION. IT WILL SWAPIN THE PROCESS (UNLESS SWAPIN IS ALREADY IN PROGRESS (SET IN SWAPTABLE) DUE TO INTERRUPTED SWAPIN).

*WHO SETS THE SAR FLAG?

FLAGPROCABSENT SETS THE SAR FLAG WHEN SIODM TRIES TO IOFREEZE AN ABSENT SEGMENT FOR A FILE SYSTEM INITIATED DISC REQUEST.

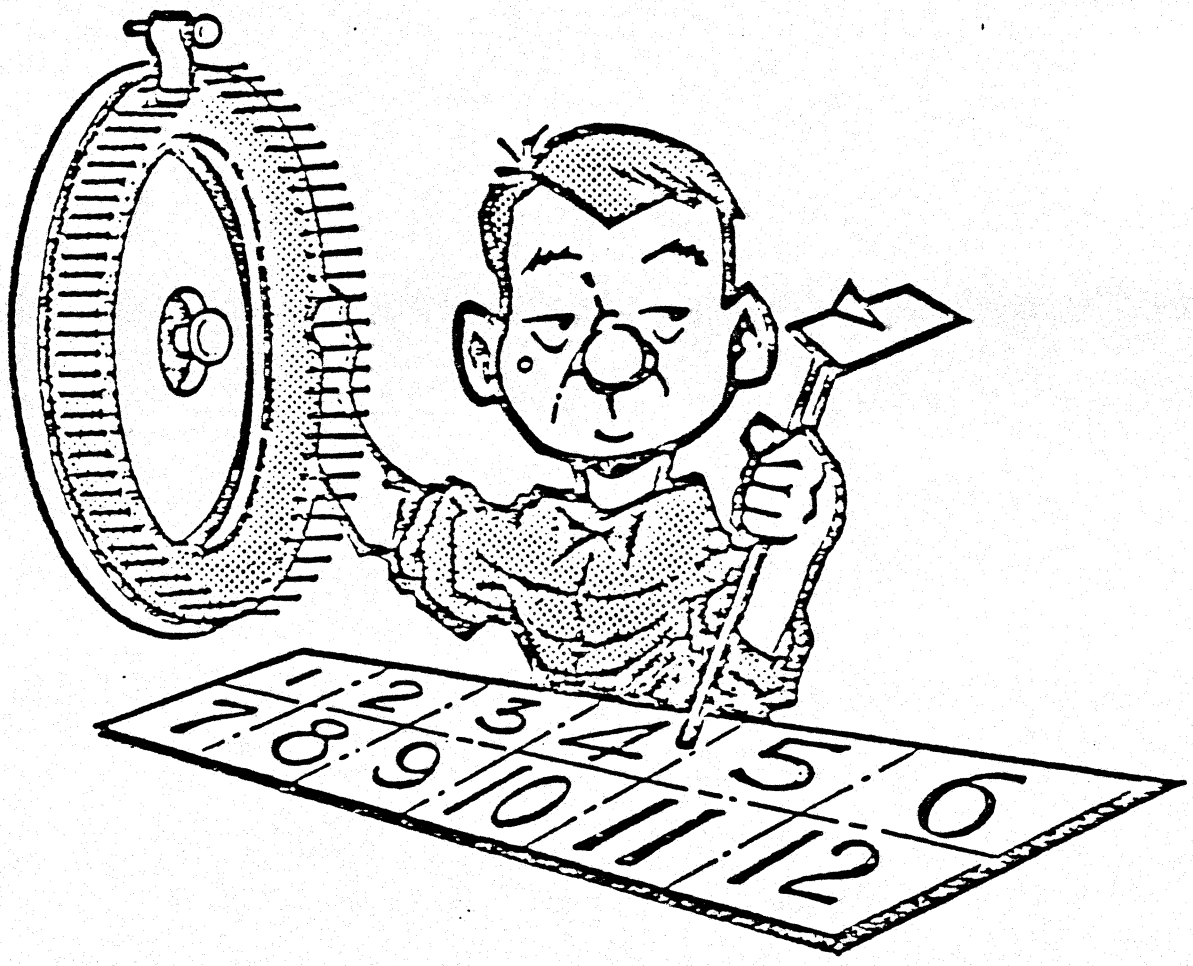
SAVESTATE SETS THE SAR FLAG IF THE IS TRAPPED ON MEMORY.

*WHO CLEARS THE SAR FLAG?

SWAPIN CLEARS THE SAR FLAG IF THE SWAP WAS SUCCESSFUL (BUT SEGMENTS ARE NOT NECESSARILY IN MEMORY YET).

HEWLETT hp PACKARD

MEMORY MANAGEMENT

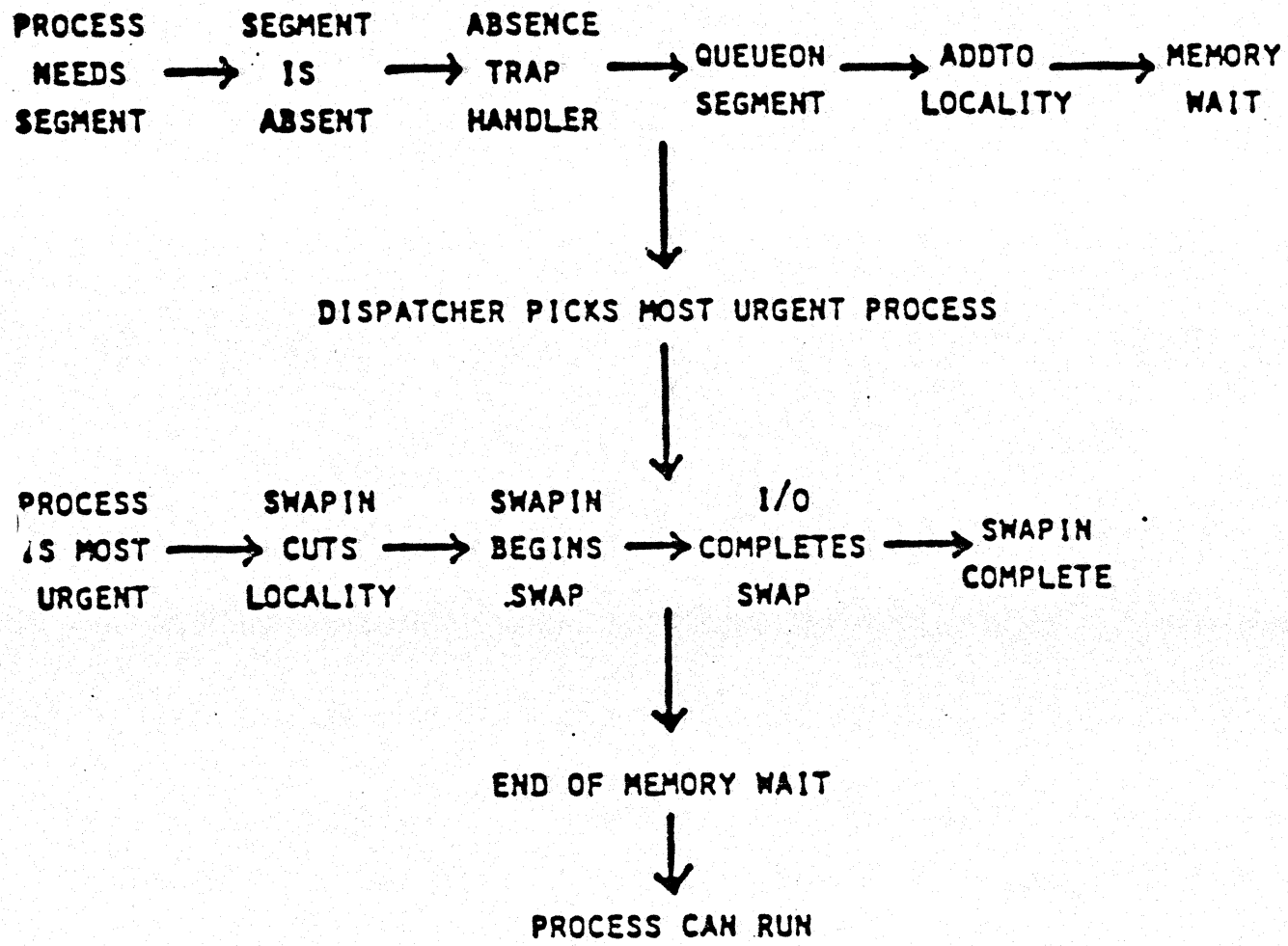


MEMORY MANAGER

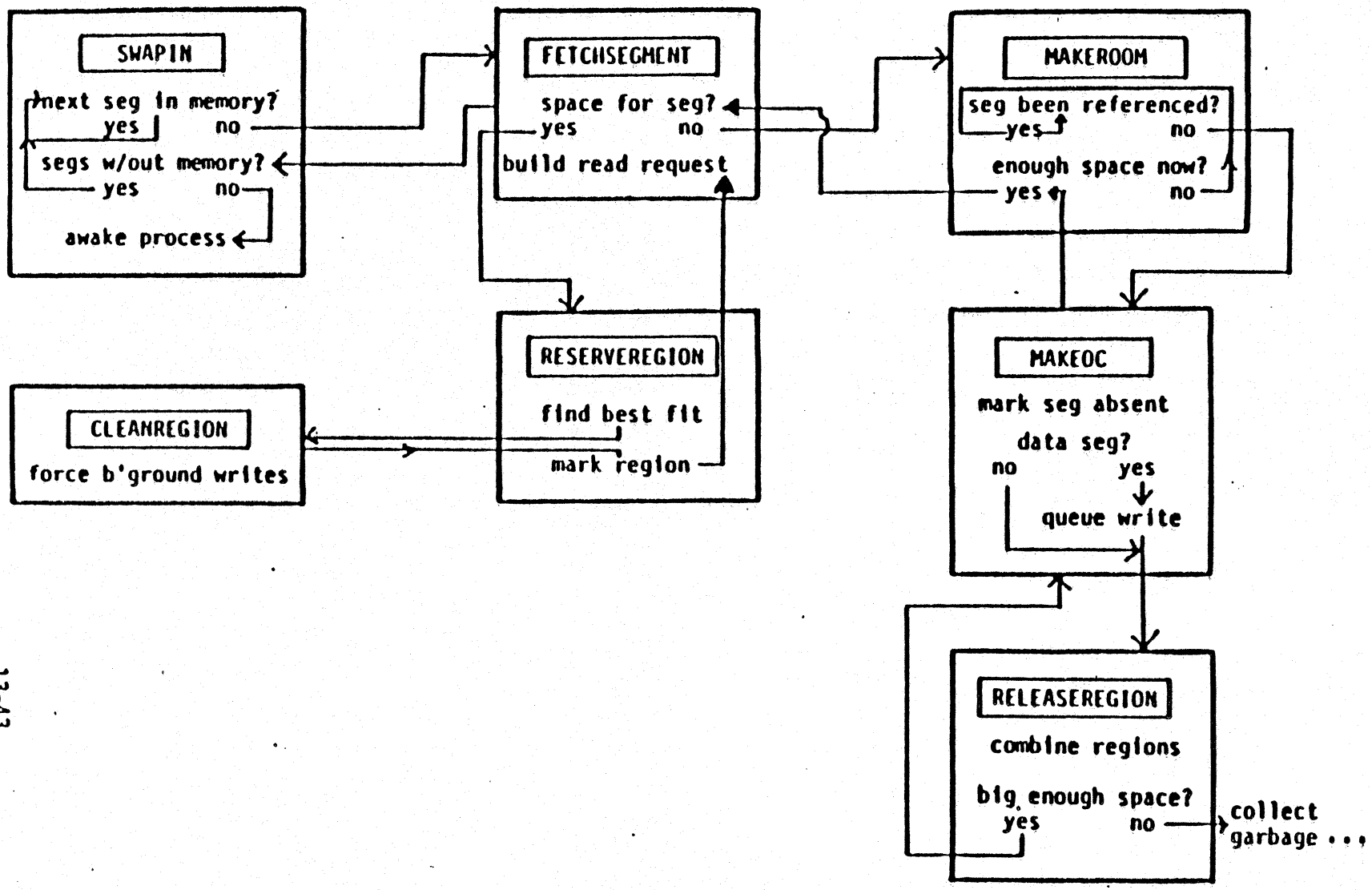
* USED TO LOCATE SPACE FOR, AND BRING INTO MAIN MEMORY, CODE AND DATA SEGMENTS NEEDED BY A PROCESS.

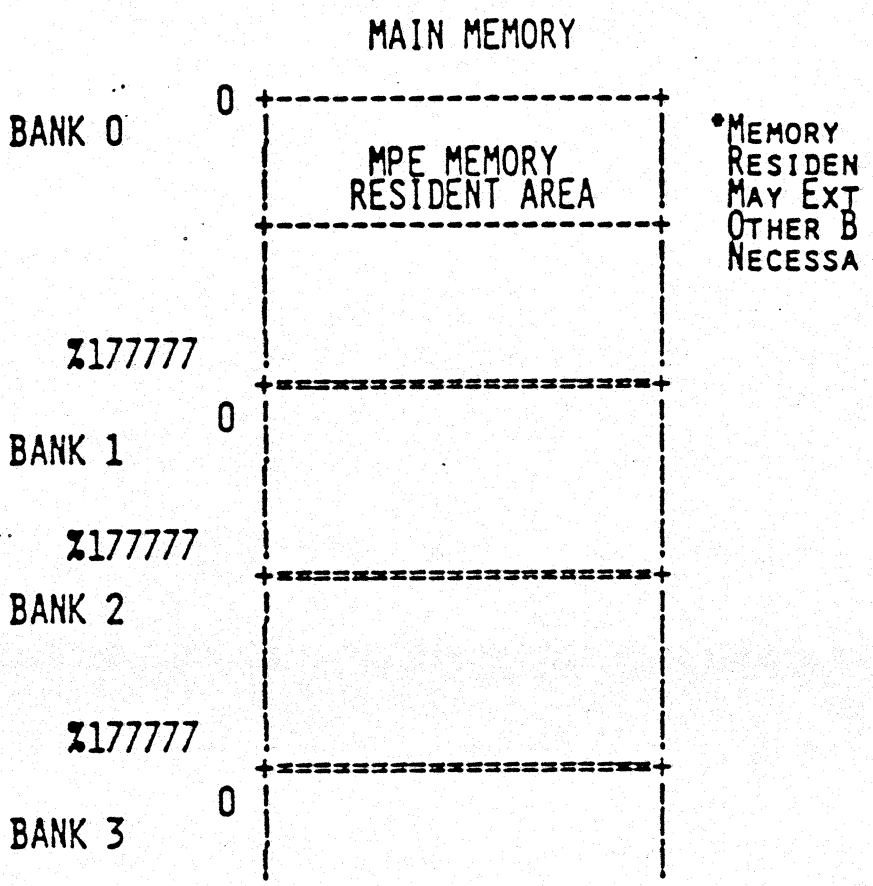
* CONSISTS OF A NUMBER OF PROCEDURES IN MODULE KERNELC WHICH RUN ON THE ICS, NOT AS A PROCESS.

REFERENCING AN ABSENT SEGMENT



PROCESS SWAPIN





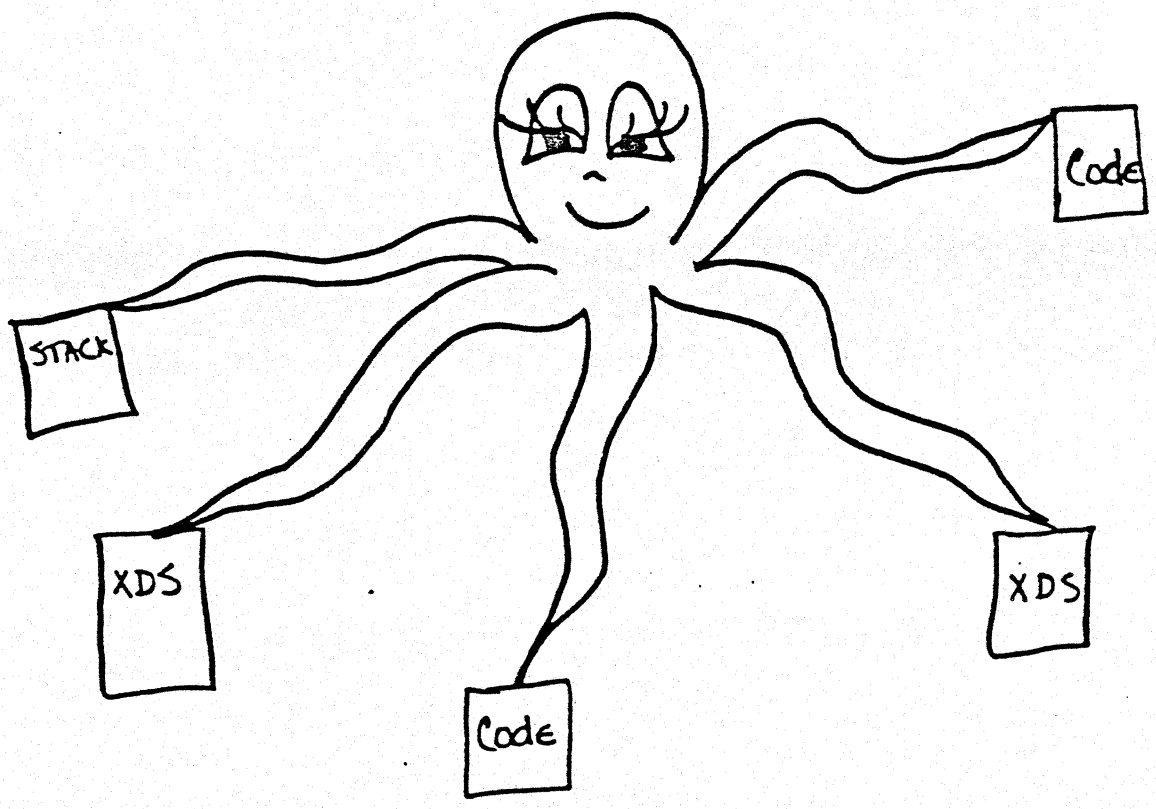
ADDRESSABLE UP TO 64K BANKS

MEMORY MANAGEMENT TABLES

* SWAPTABLE (SEGMENT LOCALITY LISTS)
* AVAILABLE REGION BITMAP
* AVAILABLE REGION LIST
* MAXAVAILREGION CELL OF SYSGLOB
* DATA SEGMENT TABLE
* CODE SEGMENT TABLES

LOCALITY OF A PROCESS

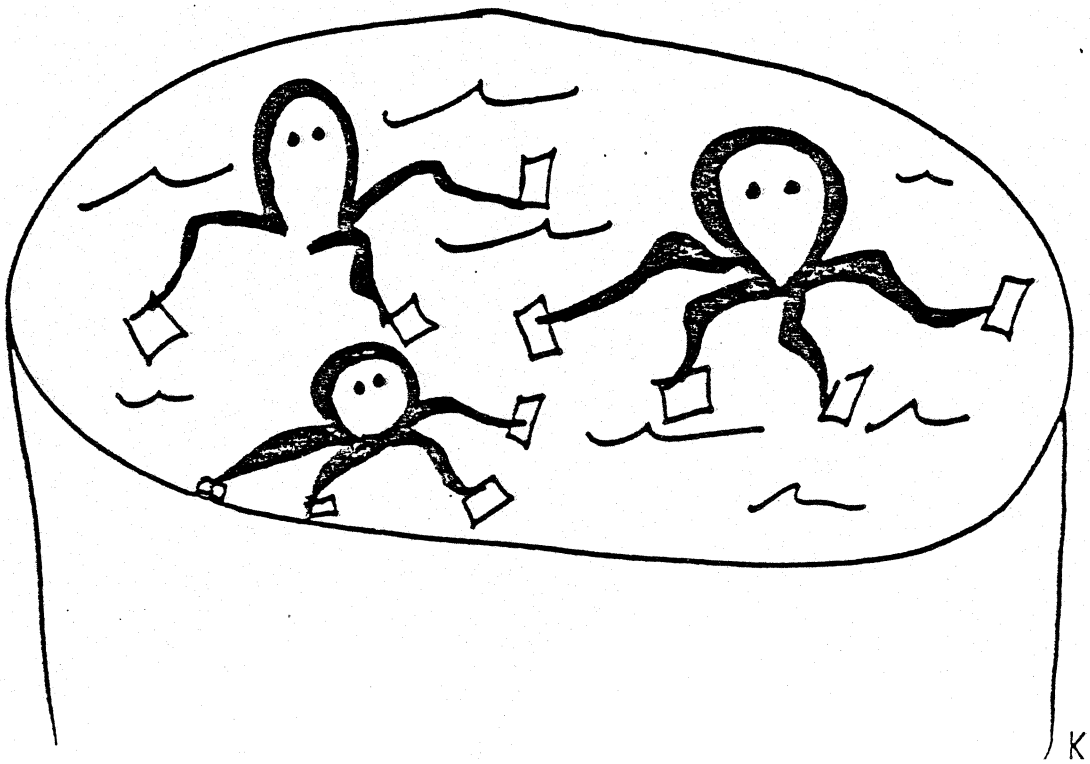
* THOSE SEGMENTS CURRENTLY BEING USED BY A PROCESS.



SWAPTABLE

* CONTAINS ALL SEGMENT LOCALITY LISTS FOR ALL PROCESSES.

* DST %27
* 5 WORD ENTRIES
* ENTRY 0 BEGINS AT SYSBASE RELATIVE ADDRESS FOUND IN %1025
* CORE RESIDENT
* CONTAINS LISTS USED TO KEEP TRACK OF LOCALITY OF EACH PROCESS ("SEGMENT LOCALITY LISTS")
* LIST FOR EACH PROCESS POINTED TO BY WORD 1 OF IT'S PCB ENTRY



MINIMAL LOCALITY OF A PROCESS

- STACK
- XDS (IF SET TO IT)
- DATA SEGMENT WITH OUTSTANDING DISC I/O
- MOST RECENTLY USED CODE SEGMENT
- FAULTING SEGMENT

SWAP TABLE

WHAT KEY INFORMATION IS KEPT IN THE SWAP TABLE?

- WHETHER THE SWAP-IN IS IN PROGRESS FOR THE PROCESS
- IF THE PROCESS' LOCALITY CONTAINS AT LEAST ONE SEGMENT
- IF THE PROCESS' SWAP-IN IS COMPLETE
- NUMBER OF SEGMENTS TO READ IN BEFORE SWAP-IN IS COMPLETE
- POINTERS TO QUEUES OF PROCESSES WAITING FOR A SEGMENT
- SEGMENT FREEZE REQUESTS
- WHETHER A PROCESS IS WAITING FOR DISC I/O AGAINST THE SEGMENT
- WHICH SEGMENT (IF ANY) IS BEING READ IN BY PROCEDURE SWAP-IN

SWAP TABLE (CONT.)

WHO USES THE SWAP TABLE?

- PROCEDURE SWAPIN
  READS INTO MEMORY THOSE SEGMENTS NEEDED BY A PROCESS THE DISPATCHER HAS CHOSEN TO RUN
- PROCEDURE DSP (DISPATCHER)
  MARKS AS REFERENCED THOSE SEGMENTS OF A PROCESS' MINIMAL LOCALITY SO THAT NEXT PROCESS CHOSEN IN INTERIM WILL NOT FORCE THOSE SEGMENTS TO OVERLAY CANDIDATES IF MEMORY PRESSURE OCCURS
- PROCEDURE UNDEFSEGSMPDQ
  DETERMINES WHICH PROCESSES ARE WAITING FOR A GIVEN SEGMENT TO COME INTO MEMORY SO THAT THEY CAN BE AWAKENED

SWAP TABLE (CONT.)

WHEN ARE SEGMENTS ADDED TO A PROCESS' LOCALITY LIST?

- CALLS TO PROCEDURE ADDTOLOCALITY

  LOCKSEG ADDS A SEGMENT REQUESTED TO BE LOCKED IN MEMORY TO THE LOCALITY LIST OF THE REQUESTING PROCESS

  QUEUEONSEGMENT ADDS A SEGMENT TO THE LOCALITY LIST OF A PROCESS WHEN THAT PROCESS HAS AN ABSENCE TRAP ON THE SEGMENT

  SAVESTATE SUBROUTINE OF DSP ADDS THE LAST REFERENCED CODE SEGMENT OF THE BLOCKING PROCESS TO ITS LOCALITY LIST IF MEMORY PRESSURE

WHEN ARE SEGMENTS REMOVED FROM A PROCESS' LOCALITY LIST?

- CALLS TO PROCEDURE ADJUSTLOCALITY

  SWAPIN REDUCES A PROCESS' LOCALITY TO MINIMAL BEFORE READING IN

  TERMINATE REMOVES ALL SEGMENTS FROM A PROCESS' LOCALITY LIST

SWAPTABLE

WHEN ARE SEGMENTS REMOVED FROM THE LOCALITY LIST OF A PROCESS?

CALLS TO PROCEDURE ADJUSTLOCALITY

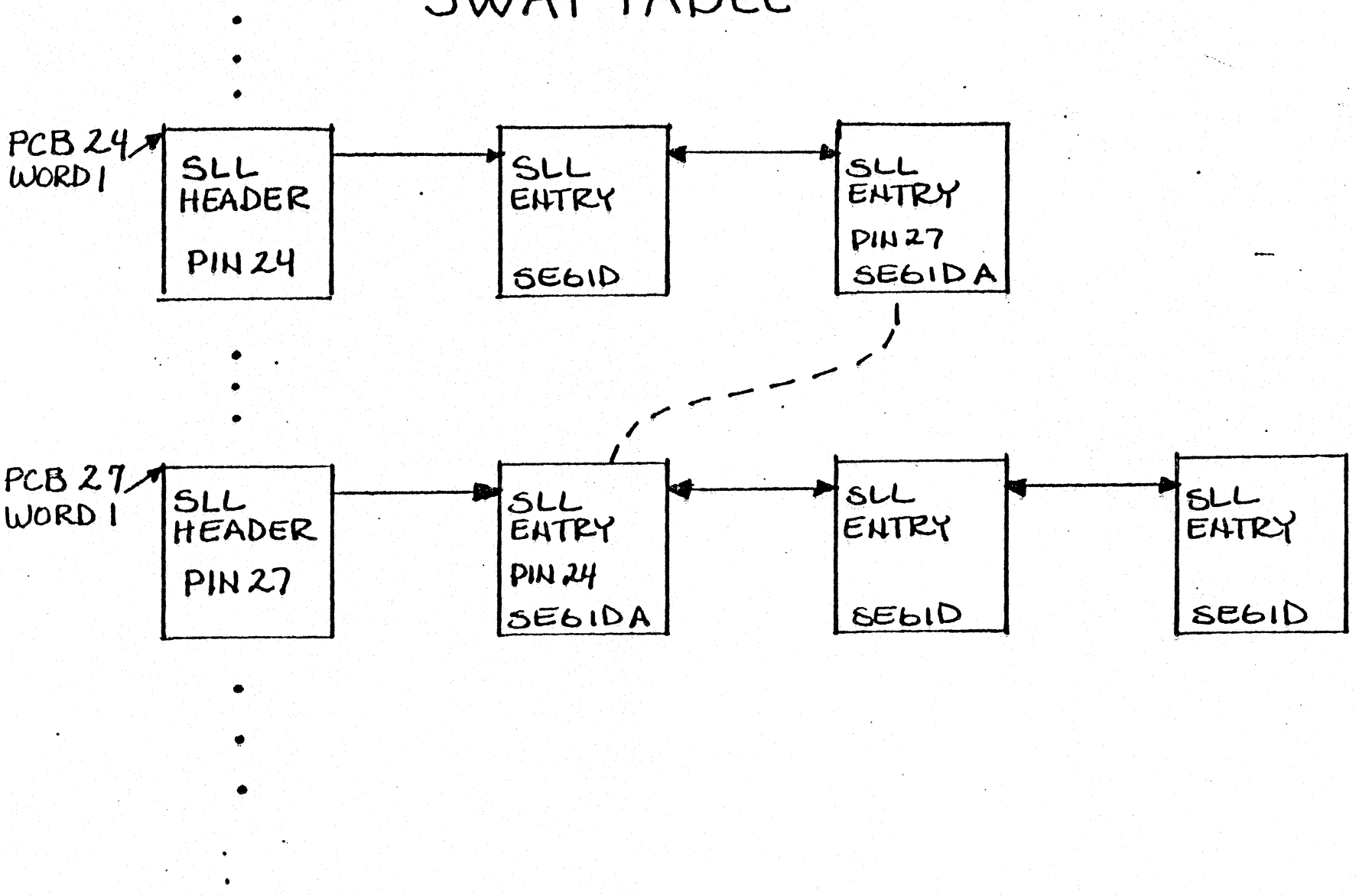
* SWAPIN CUTS LOCALITY TO MINIMAL BEFORE SWAPPING IN A PROCESS

* TERMINATE REMOVES ALL SEGMENTS FROM A PROCESS'S LOCALITY LIST

  UPON PROCESS TERMINATION

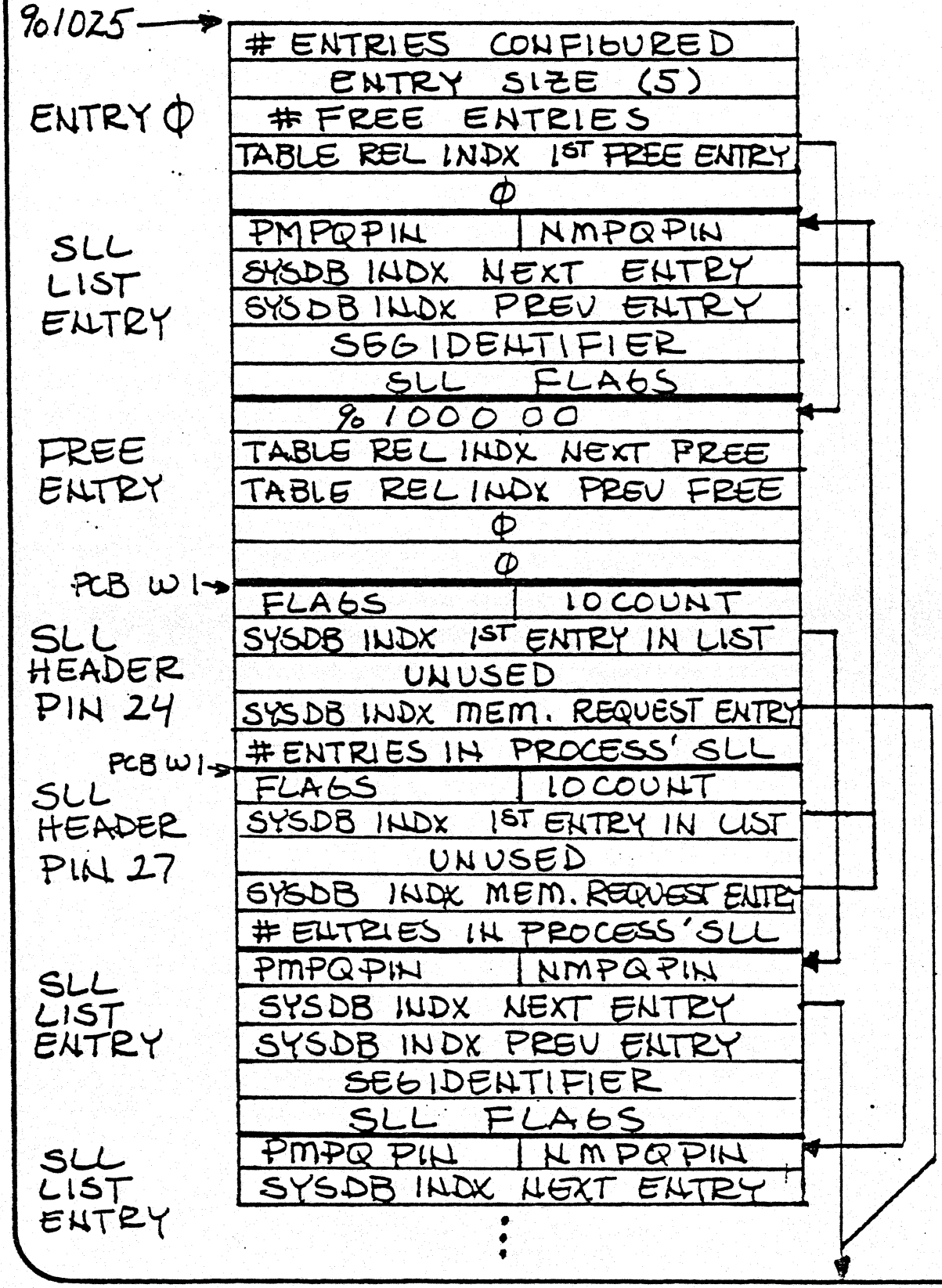
HEWLETT hp PACKARD

SWAPTABLE



HEWLETT hp PACKARD

SWAPTABLE DST 23$_{10}$



SEGMENT IDENTIFIER

FOUND IN PCB, REGION HEADERS, SPECIAL REQUEST TABLE, SWAP TABLE, AND SOME I/O TABLES

SEGIDENTIFIER. (0:1)=1

- SEGMENT IS PART OF A PROGRAM
- SEGIDENTIFIER. (1:7) IS PROGRAM INDEX INTO CSTXBLK
- SEGIDENTIFIER. (8:8) IS LOGICAL SEGMENT NUMBER

SEGIDENTIFIER. (0:2)=0

- SEGMENT IS A DATA SEGMENT
- SEGIDENTIFIER. (2:14) IS DST ENTRY NUMBER

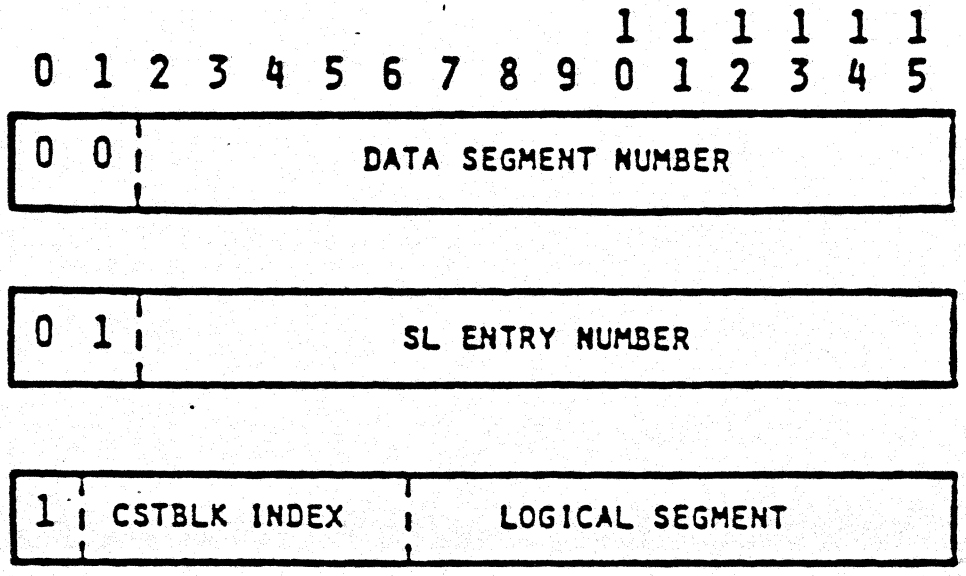
SEGIDENTIFIER. (0:2)=1

- SEGMENT IS AN SL SEGMENT
- SEGIDENTIFIER. (2:14) IS SL (CST) ENTRY NUMBER

STANDARD SEGMENT IDENTIFIER FORMAT ALLOWS FOR

- 2^{14} DATA SEGMENTS (16,384)
- 2^{14} SL SEGMENTS (16,384)
- 2^{7} CONCURRENTLY LOADED PROGRAMS (128)
- 2^{8} CODE SEGMENTS PER PROGRAM (256)

SEGMENT IDENTIFIER FORMAT



MAXIMUM AVAILABLE REGION CELL

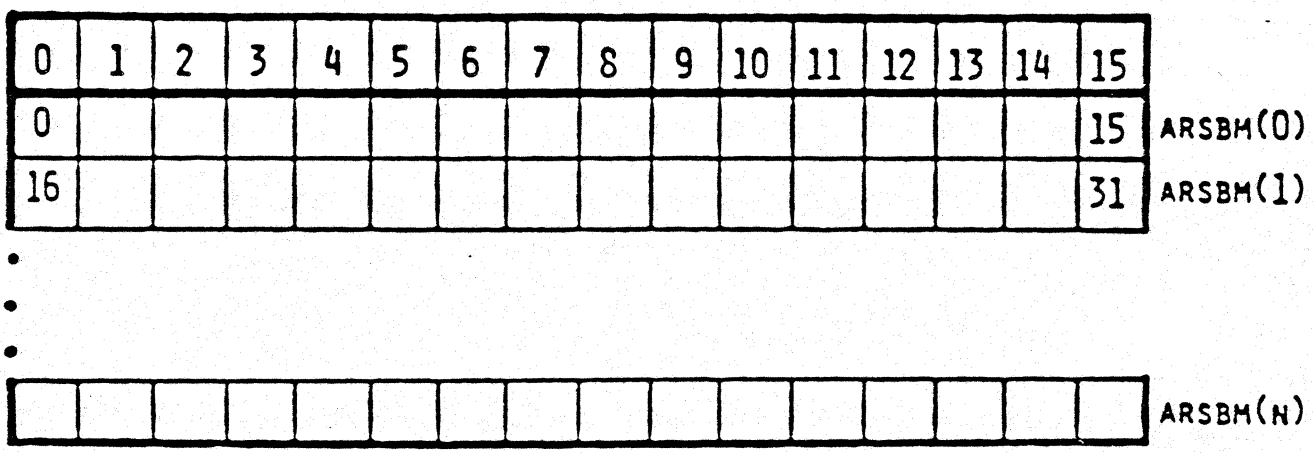
SYSGLOB % 45

CONTAINS THE # OF PAGES IN THE LARGEST CURRENTLY AVAILABLE REGION

AVAILABLE REGION BIT MAP (ARSBM)

%1044 SYSBASE INDEX OF BASE OF ARSBM

ARSBM DST # =58 (%71)



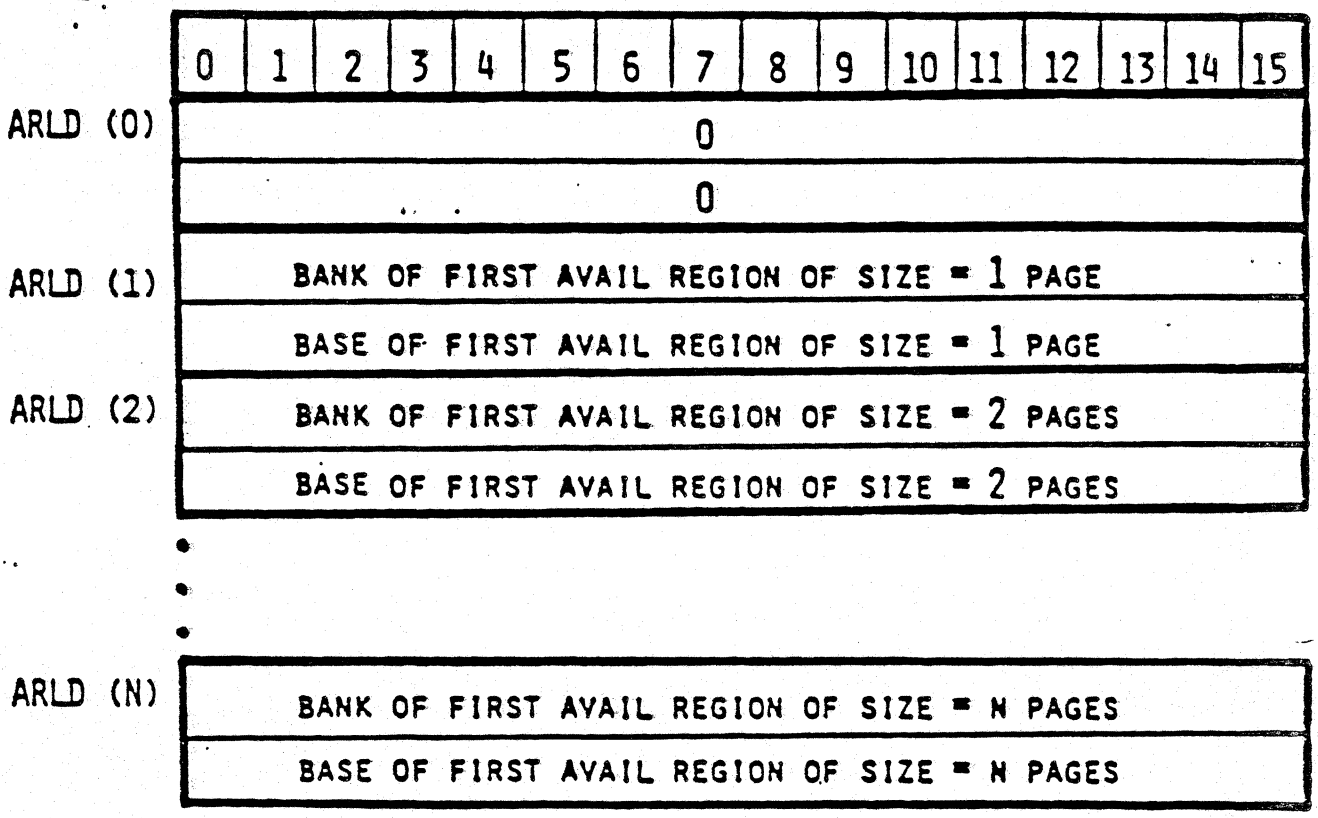
M= (# OF AVAILABLE REGION SIZES/16) +1

ARSBM (J) . (N:1) = 1 ==> THE AVAILABLE REGION LIST OF SIZE J*16+K PAGES IS NON-EMPTY.

AVAILABLE REGION LIST (ARL)

%1044 SYSBASE INDEX OF BASE OF ARL

ARL DST # = 55 (%67)



AVAILABLE REGIONS IN MAIN MEMORY ARE KEPT TRACK OF BY MULTIPLE FREE LISTS. ALL AVAILABLE REGIONS OF THE SAME SIZE ARE LINKED INTO THE SAME AVAILABLE REGION LIST (ARL). A BITMAP IS MAINTAINED TO INDICATE WHICH LISTS ARE NON-EMPTY (ARSBM). A SYSGLOB CELL IS MAINTAINED WHICH CONTAINS THE SIZE OF THE LARGEST CURRENTLY AVAILABLE REGION.

A SEGMENT MAY BE IN ONE OF THREE MEMORY STATES:

ABSENT

THE SEGMENT IS NOT IN MAIN MEMORY. IT IS IN VIRTUAL MEMORY (OUT ON DISC).

PRESENT

- THE SEGMENT IS IN MAIN MEMORY
- AN UP TO DATE COPY OF A CODE SEGMENT EXISTS IN VIRTUAL MEMORY •
- AN OBSOLETE COPY OF A DATA SEGMENT EXISTS IN VM •

RECOVERABLE OVERLAY CANDIDATE

- VALID COPY OF SEGMENT EXISTS BOTH IN MAIN MEMORY AND DISC.
- COPY IN MAIN MEMORY IS EXPENDABLE
- IF SEGMENT IS REFERENCED, SWAP IN IS NOT REQUIRED

SEGMENTS THAT ARE RECOVERABLE OVERLAY CANDIDATES ARE CONSIDERED AS "FREE AREAS" FOR THE MEMORY MANAGEMENT ALGORITHMS

MEMORY REGIONS

- AVAILABLE

  CONSISTS OF FREE AREAS OVERLAY CANDIDATES

- RESERVED

  A ONCE AVAILABLE REGION THAT HAS BEEN SELECTED BY MEMORY MANAGER FOR A SEGMENT; ALL ANTICIPATORY WRITES OF OVERLAY CANDIDATES AND READ OF THE NEW SEGMENT HAVE NOT TAKEN PLACE.

- ASSIGNED

  A ONCE RESERVED REGION THAT HAS BEEN CLEANED AND HAS HAD A NEW SEGMENT READ INTO IT.

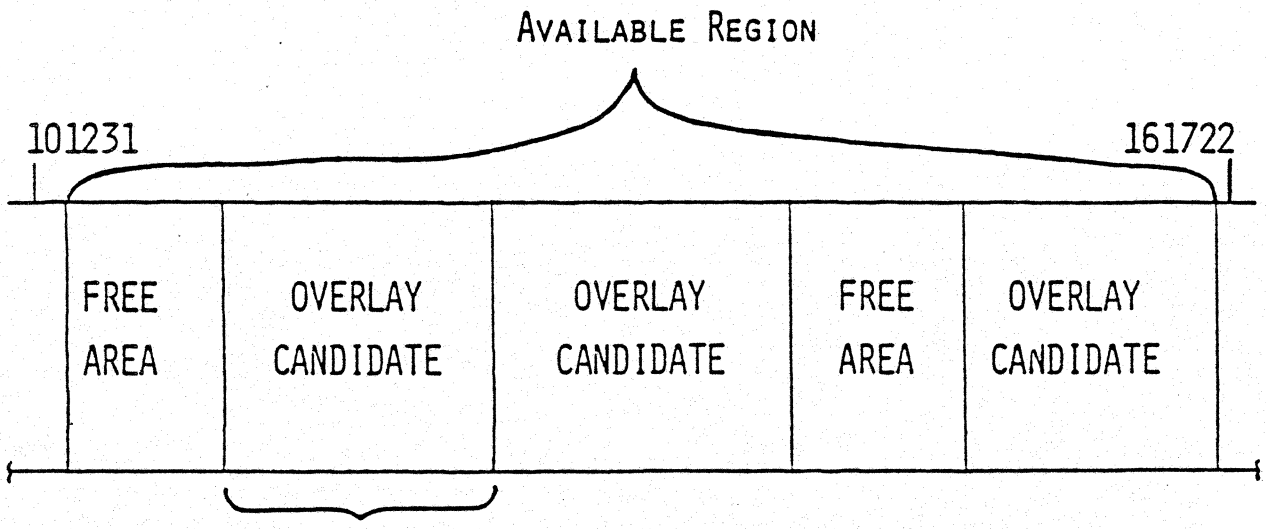
MAIN MEMORY REGIONS

1. MAIN MEMORY IS PARTITIONED INTO REGIONS. EACH REGION IS IN ONE OF THREE STATES: AVAILABLE, RESERVED, OR ASSIGNED. THE REGION HEADERS AND TRAILERS OF AVAILABLE, RESERVED, AND ASSIGNED REGIONS CONTAIN THE STATE AND CONTROL INFORMATION PERTAINING TO THE CURRENT OR PLANNED CONTENTS OF THE REGION.

2. AVAILABLE:

AN AVAILABLE REGION IS AVAILABLE FOR CONSUMPTION BY THE FREE SPACE ALLOCATION MECHANISM. AN AVAILABLE REGION CONSISTS OF NEIGHBORING SUBREGIONS, EACH OF WHICH IS EITHER A HOLE OR AN OVERLAY CANDIDATE. AN AVAILABLE REGION IS LINKED INTO THE AVAILABLE REGION LIST OF APPROPRIATE SIZE.

AVAILABLE REGION



WAS ONCE AN ASSIGNED REGION. MEMORY MANAGER FOUND THAT THIS SEGMENT WAS NOT BEING REFERENCED OFTEN AND, THEREFORE, MADE ITS SPACE AVAILABLE FOR USE. HOWEVER, IT MAY STILL BE REFERENCED BY A PROCESS AND THUS BECOME AN ASSIGNED SEGMENT. ONCE THIS AVAILABLE REGION HAS BEEN SELECTED FOR USE TO SATISFY A MEMORY REQUEST (RESERVED REGION) THE OVERLAY CANDIDATE MAY NOT BE RECOVERED.

3. RESERVED:

A RESERVED REGION IS A MAIN MEMORY REGION WHICH IS IN THE TRANSITION STATE FROM AVAILABLE TO ASSIGNED. A RESERVED REGION ~~HAS BEEN~~ is being CLEANED, AND THERE IS A PENDING DISC READ OF A SEGMENT INTO THE REGION.

4. ASSIGNED:

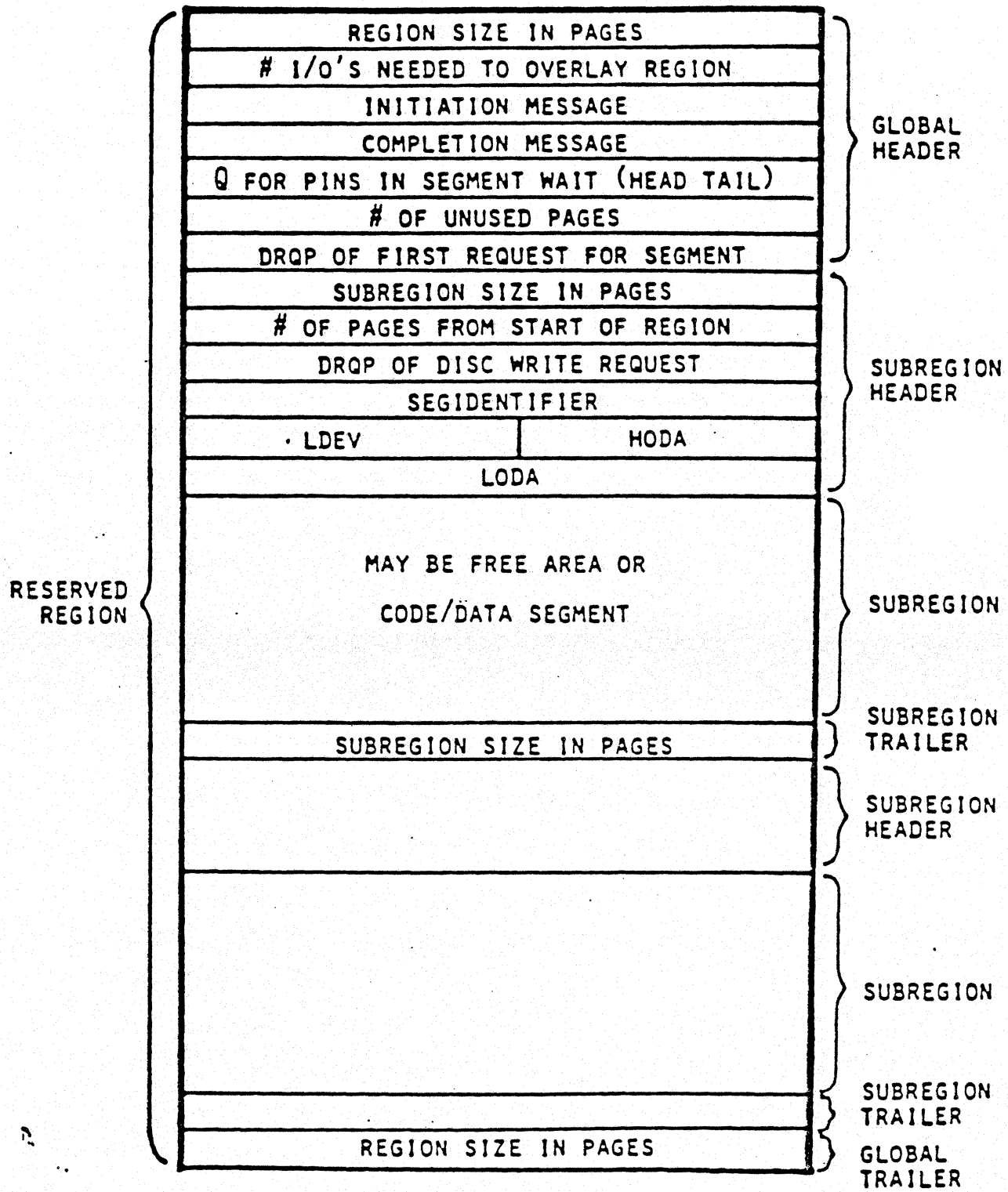
ASSIGNED REGIONS ARE OCCUPIED BY PRESENT SEGMENTS. AVAILABLE AND RESERVED REGIONS CONSIST OF ONE OR MORE ADJACENT SUBREGIONS. REGION HEADERS AND TREAILERS ARE PARTITIONED INTO GLOBAL AND LOCAL COMPONENTS. THE GLOBAL REGION HEADER/TRAILER IS ONLY VALID FOR THE FIRST/LAST SUBREGION IN REGIONS CONSISTING OF MORE THAN ONE SUBREGION.

HEWLETT PACKARD

MEMORY REGION HEADERS AND TRAILERS

- EACH REGION IN MEMORY HAS A HEADER AND TRAILER.
- USED INSTEAD OF THE MEMORY LINKS OF MPE III
- CONTAIN INFORMATION USED BY MPE TO ALLOCATE MEMORY SPACE.

REGION HEADER AND TRAILER



IN SUMMARY...

2.3 Main Memory Region Headers and Trailers

Main memory is partitioned into regions. Each region is in one of three states available, reserved, or assigned.

An available region is available for consumption by the free space allocation mechanism. An available region consists of neighboring subregions, each of which is either a hole or an overlay candidate. An available region is linked into the available region list of appropriate size.

A reserved region is a main memory region which is in the transition state from available to assigned. A reserved region has been cleaned, and there is a pending disc read of a segment into the region.

Assigned regions are occupied by present segments. Available and reserved regions consist of one or more adjacent subregions. Region headers and trailers are partitioned into global and local components. The global region header/trailer is only valid for the first/last subregion in regions consisting of more than one subregion.

The region headers and trailers of available, reserved, and assigned regions contain the state and control information pertaining to the current or planned contents of the region.

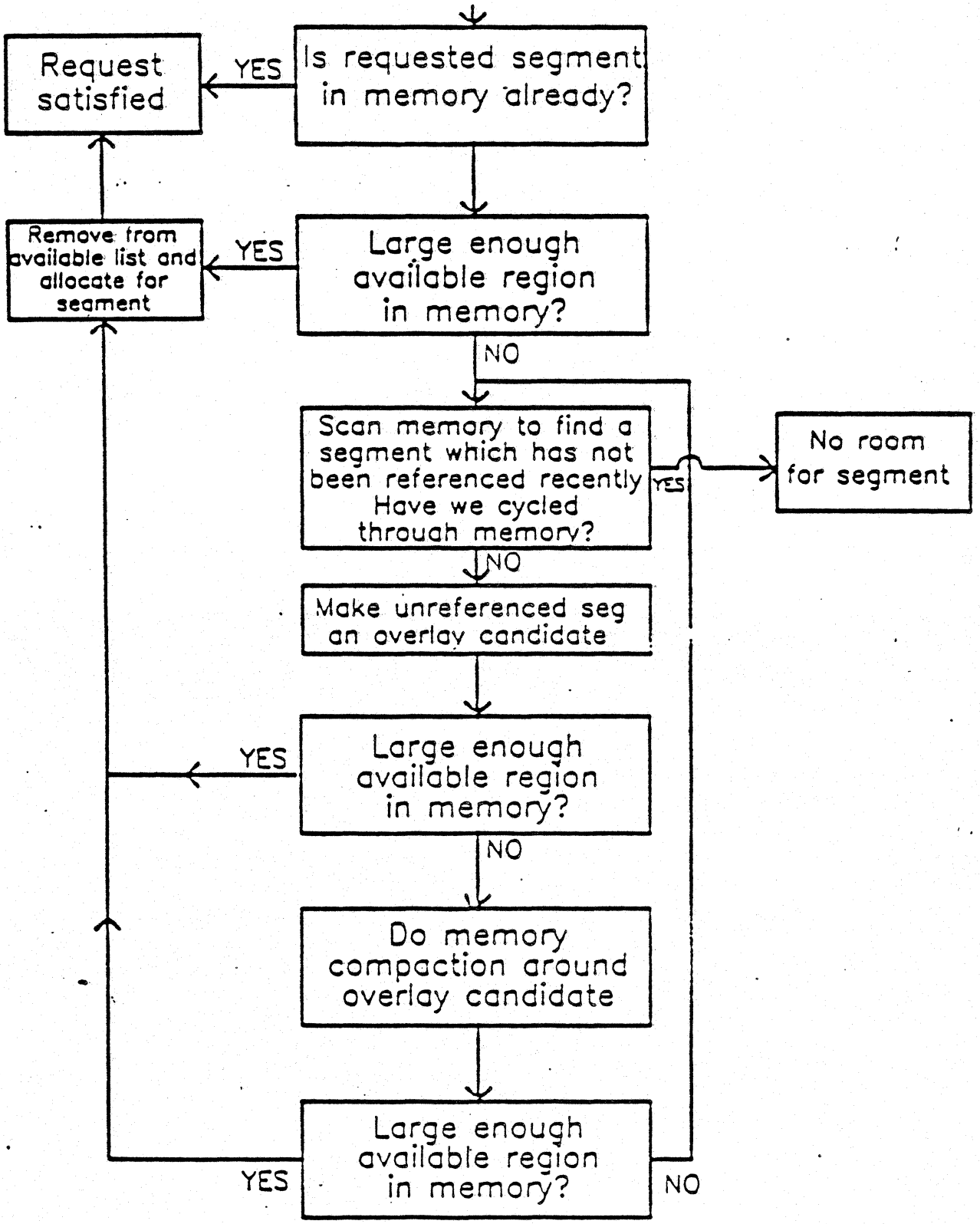
MPE IV MEMORY MANAGER

THE MEMORY MANAGER USED BY MPE IV CONSISTS OF A GROUP OF PROCEDURES (IN MODULE KERNELC) AND USES AN ALGORITHM FOR DETERMINING WHICH SEGMENTS ARE TO BECOME OVERLAY CANDIDATES WHEN MEMORY SPACE IS NEEDED. THIS ALGORITHM CHECKS TO SEE WHICH SEGMENTS HAVE NOT BEEN REFERENCED RECENTLY BY ANY PROCESS. THESE UNREFERENCED SEGMENTS WILL BECOME OVERLAY CANDIDATES WHEN SPACE IS NEEDED. THIS TENDS TO REDUCE THRASHING BY CONSIDERING THE MEMORY NEEDS OF THE SYSTEM AS A WHOLE. IN ADDITION, SINCE THE ALGORITHM CYCLES THROUGH MEMORY TO PRODUCE OVERLAY CANDIDATES, AVAILABLE AREAS ARE OPENED UP IN PROXIMITY TO EACH OTHER, INCREASING THE CHANCE (WITH THE HELP OF COMPACTION) THAT LARGE SPACES WILL BE CREATED. AS OVERLAY CANDIDATES ARE CREATED, BACKGROUND ANTICIPATORY WRITES ARE ISSUED (DATA SEGMENTS). THESE WRITES WILL NOT BE GIVEN A HIGH PRIORITY UNLESS THE SPACE CONTAINING THE OVERLAY CANDIDATE IS SELECTED AND RESERVED TO FILL A MEMORY REQUEST.

THE MEMORY MANAGER SELECTS THE SMALLEST AVAILABLE (FREE + O.C.) REGION WHICH WILL FULFILL THE MEMORY REQUEST. IT SETS UP THE WRITE AND READ INFORMATION IN A PART OF THE REGION CALLED A HEADER SO THAT THE I/O SYSTEM CAN COMPLETE NECESSARY WRITES AND READS INTO THE REGION. THE MEMORY MANAGER IS FREE TO HANDLE OTHER MEMORY REQUESTS WHILE THE I/O IS COMPLETING.

MEMORY COMPACTION IS DONE WHEN A SEGMENT BECOMES AN OVERLAY CANDIDATE AND DURING A DISPATCHER PAUSE.

MEMORY ALLOCATION



MEMORY MANAGEMENT REPLACEMENT

- USED TO FIND OVERLAY CANDIDATES WHICH MAY OR MAY NOT SATISFY REQUEST.

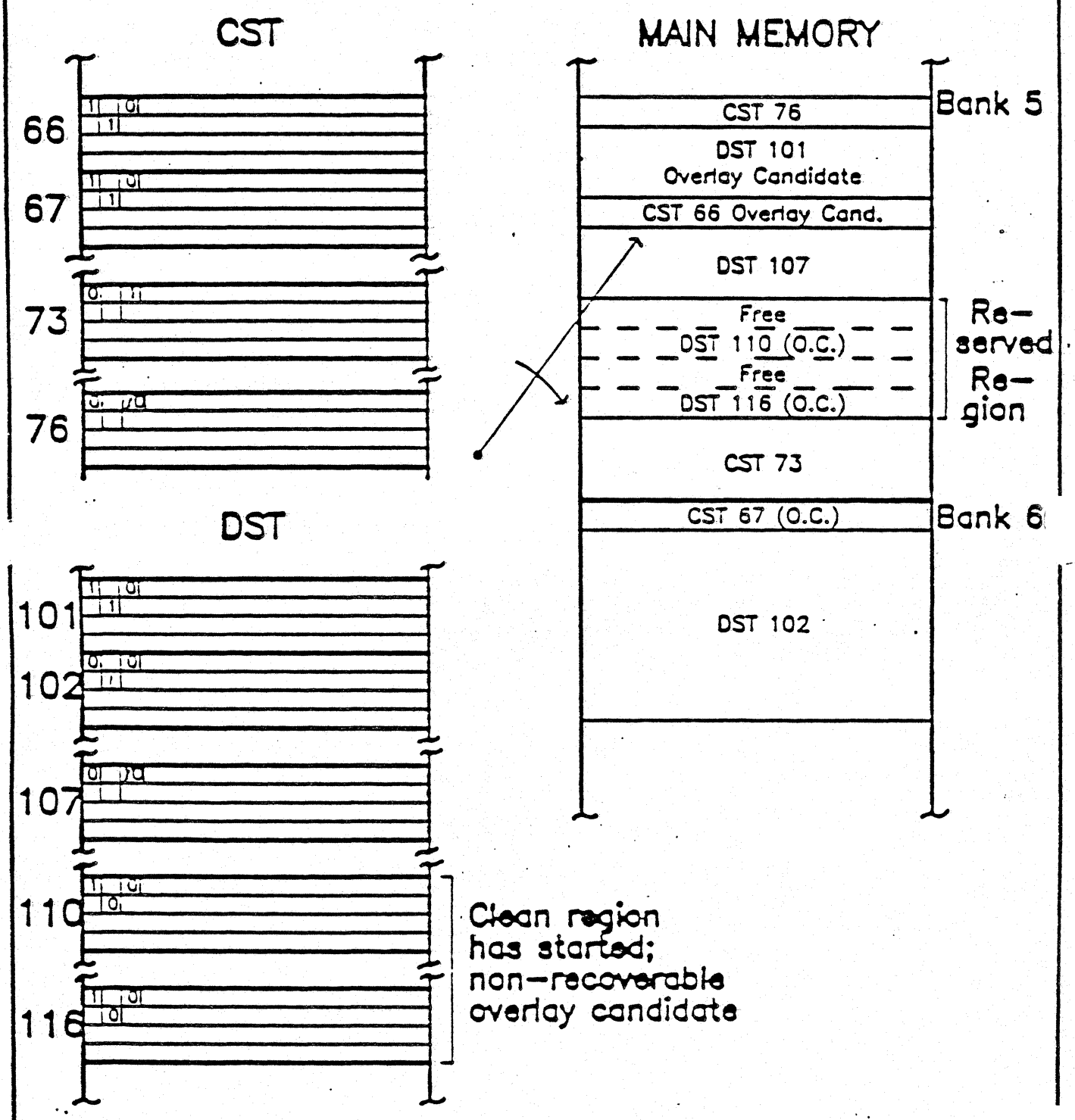
1. SCAN THROUGH MEMORY REGION BY REGION.

2. WHEN ASSIGNED REGION IS FOUND, CHECK REFERENCE BIT IN SEGMENT TABLE.

3. REFERENCE BIT IS ON, CLEAR BIT AND CONTINUE SCAN.

4. IF REFERENCE BIT IS OFF, MAKE SEGMENT AN OVERLAY CANDIDATE CHECK TO SEE IF BIG ENOUGH SPACE HAS BEEN CREATED. IF NOT, DO LOCAL GARBAGE COLLECTION AROUNG REGION. IF REQUEST STILL IS NOT SATISFIED, CONTINUE SCAN.

REPLACEMENT ALGORITHM



GARBAGE COLLECTION

(MEMORY COMPACTION)

* COMBINES AVAILABLE REGIONS INTO LARGER AVAILABLE REGIONS (HOLES)

* INCREASES THE PROBABILITY OF FINDING ROOM IN MEMORY FOR SEGMENTS

* 2 TYPES OF GARBAGE COLLECTION:

    1. LOCAL GARBAGE COLLECTION

    2. GARBAGE COLLECTION DURING DISPATCHER PAUSE

GARBAGE COLLECTION

Local Garbage Collection

- Called by RELEASEREGION when a new overlay candidate has been combined into neighboring available regions and there is still not a big enough space to satisfy request

- COLLECTGARBAGE scans down in memory until previously available region is found and marks all intervening segment absent

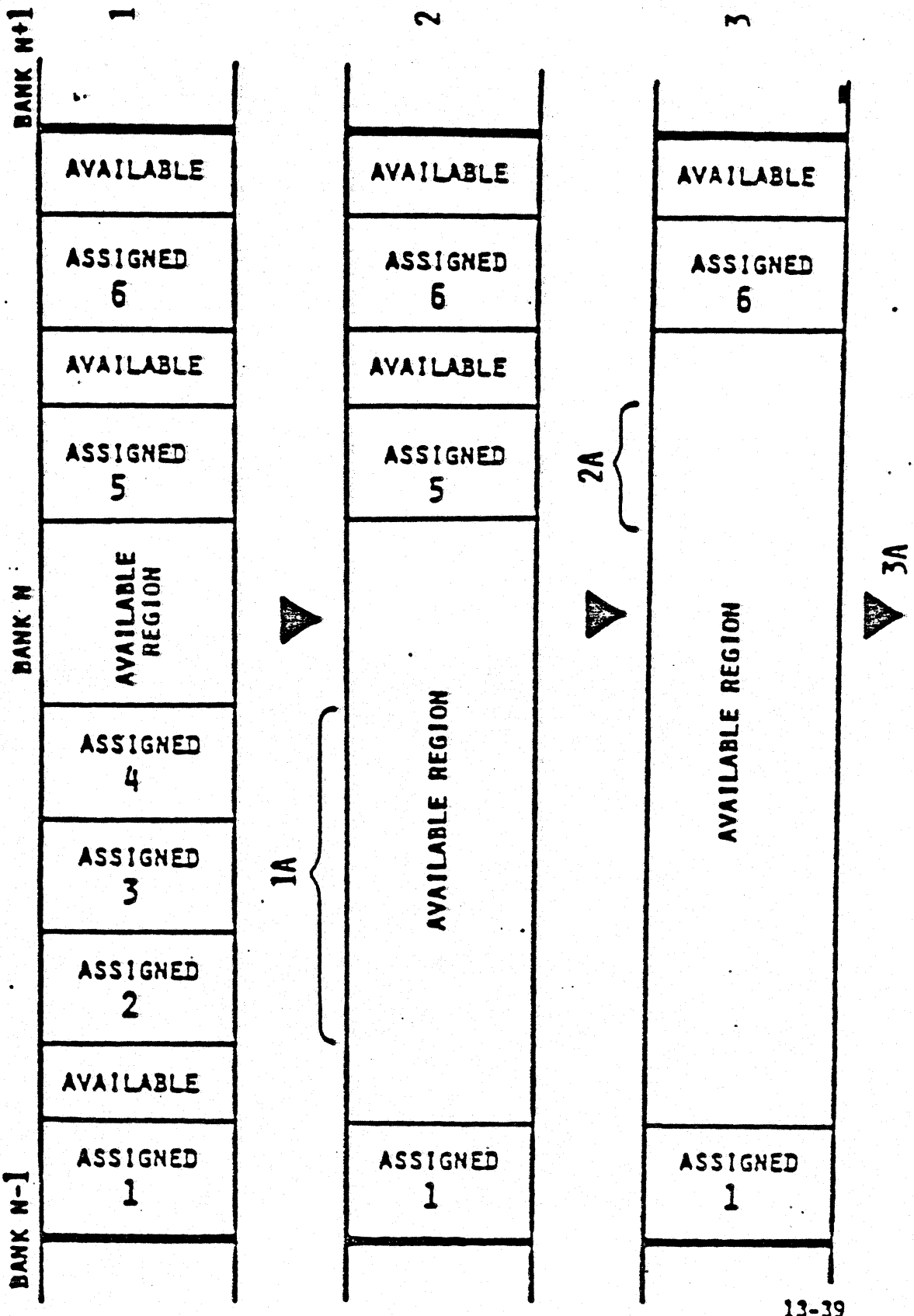
- Move intervening segments to another area of memory and combines original hole, space occupied by moved segments and previous available region

- If still not enough space, repeat process by scanning from original hole upwards in memory

GLOBAL GARBAGE COLLECTION

- ONLY HAPPENS WHEN DISPATCHER IS PAUSED, I.E. NO PROCESS IS READY OR ABLE TO RUN, POSSIBLY DUE TO MEMORY SHORTAGE
- DISPATCHER CHECKS TO SEE IF THERE IS MEMORY PRESSURE BY CHECKING TIME BETWEEN CALLS TO PROCEDURE MAKEROOM
- IF PRESSURE EXISTS, CALL COLLECTGARBAGE WITH A CODE INDICATING WE WANT TO INCREASE THE SIZE OF THE LARGEST AVAILABLE "HOLE"
- ALGORITHM THEN PROCEEDS AS IN LOCAL GARBAGE COLLECTION, BUT CHECKS BETWEEN STEPS TO SEE IF THERE IS SOMETHING MORE URGENT TO DO

MEMORY COMPRESSION / GARBAGE COLLECTION



FREEZING SEGMENTS

- FROM TIME TO TIME A SEGMENT (CODE OR DATA) NEEDS TO BE FROZEN. FREEZING INVOLVES MAKING A SEGMENT CORE RESIDENT AND KEEPING IT THERE.

- MPE WILL, UPON INITIATION OF I/O, FREEZE A DATA SEGMENT IN MEMORY FOR THE I/O SYSTEM TO DELIVER TO. THE STACK IS THEN SAID TO BE "I/O FROZEN".

- COMMUNICATIONS SUBSYSTEMS USE FREEZING FOR MAKING CS DRIVERS CORE RESIDENT AND FOR THE CS TRACE FACILITY. IT IS ALSO USED TO FREEZE A CORE RESIDENT TABLE FOR CS GLOBAL FLAGS.

- FREEZING IS USUALLY IMPLEMENTED BY LOCKING THE SEGMENT, THEN FREEZING TO REDUCE FRAGMENTATION.

LOCKING SEGMENTS

PRIVELEGED PROCESSES WITHIN MPE MAY REQUEST TO LOCK A SEGMENT. LOCKING INVOLVES MOVING A SEGMENT TO A BANK BOUNDARY. IT SHOULD BE DONE BEFORE A FREEZE TO INSURE MINIMAL FRAGMENTATION.*

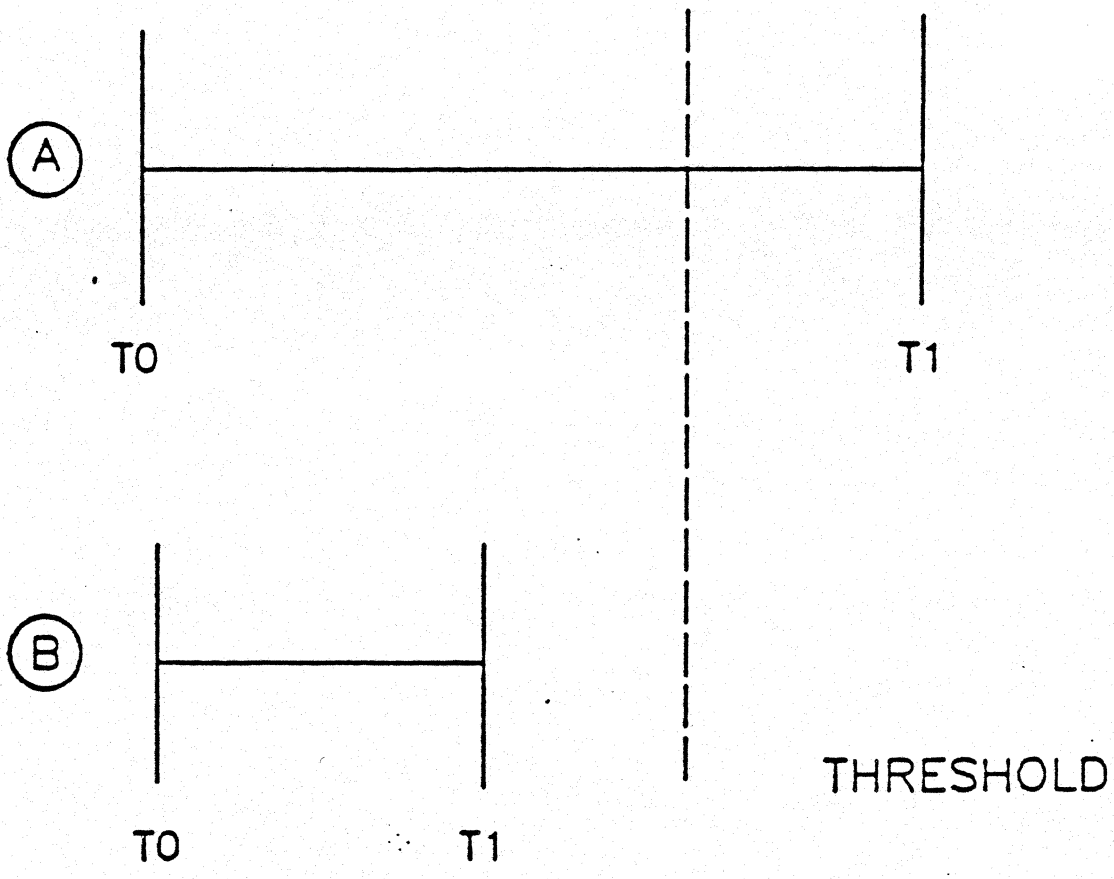
*THE I/O SYSTEM DOES NOT LOCK A SEGMENT BEFORE FREEZING FOR I/O.

MEMORY PRESSURE

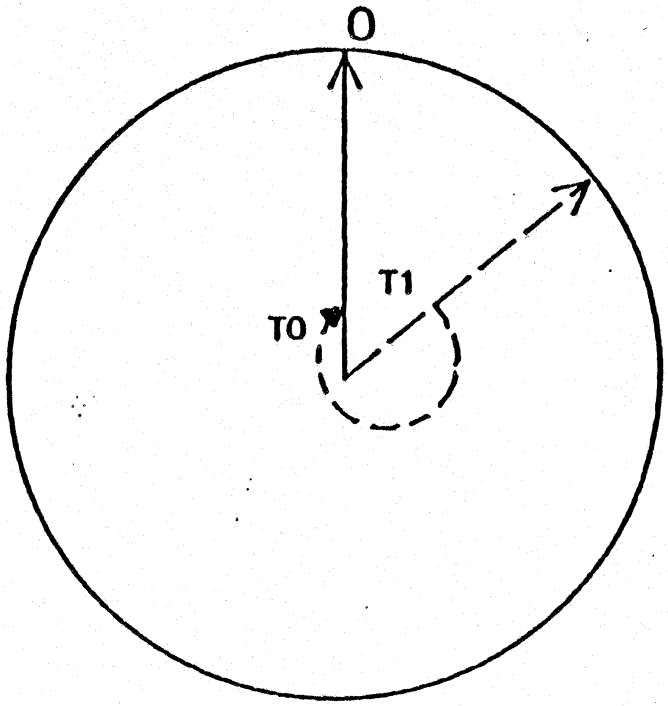
DETERMINED BY TIME BETWEEN SUCCESSIVE CALLS TO MAKE ROOM.

T0=TIME @ CALL 1

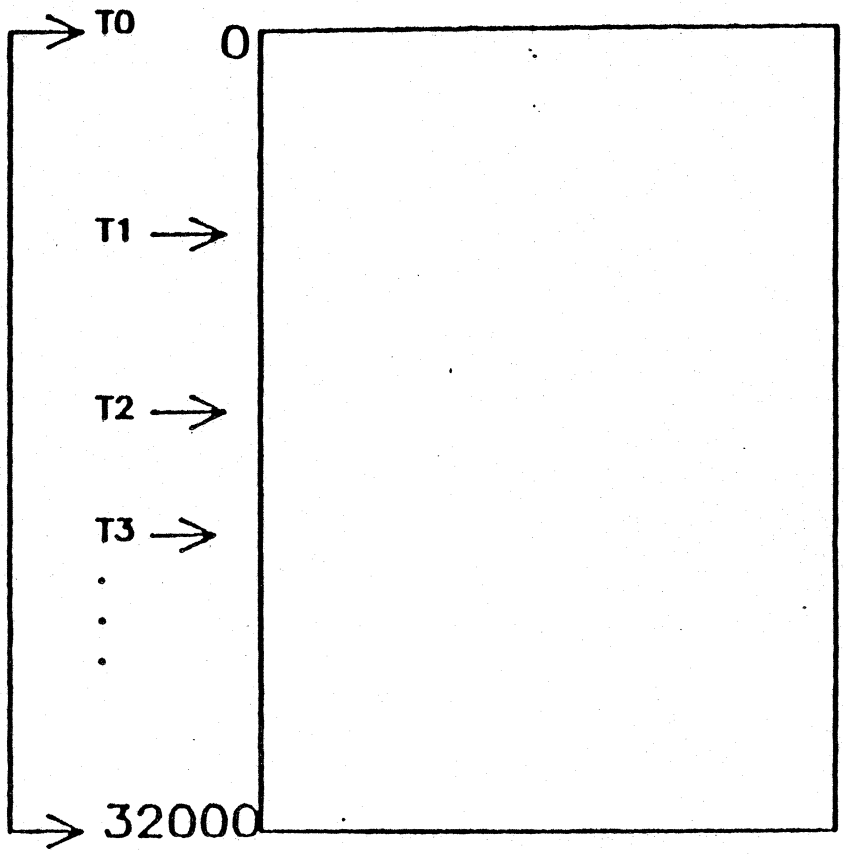
T1=TIME @ CALL 2



CLOCK ALGORITHM



MAINTAINS A POINTER
AT LAST STOPPING POINT!



BANK 1

MIN CLOCK CYCLE MEASURES HOW WELL THE SYSTEM IS DOING AT KEEPING MOST REFERENCED SEGMENTS IN MEMORY.

CLOCK ALGORITHM

CLOCK CYCLES THROUGH MEMORY SLOW?

MEMORY MANAGER WAS DEVELOPING O.C'S.

CLOCK CYCLES THROUGH MEMORY FAST?

EITHER;

- MANY SEGMENTS REFERENCED RECENTLY
- ALL SEGMENTS ALREADY O.C.
- LOTS OF FROZEN SEGMENTS

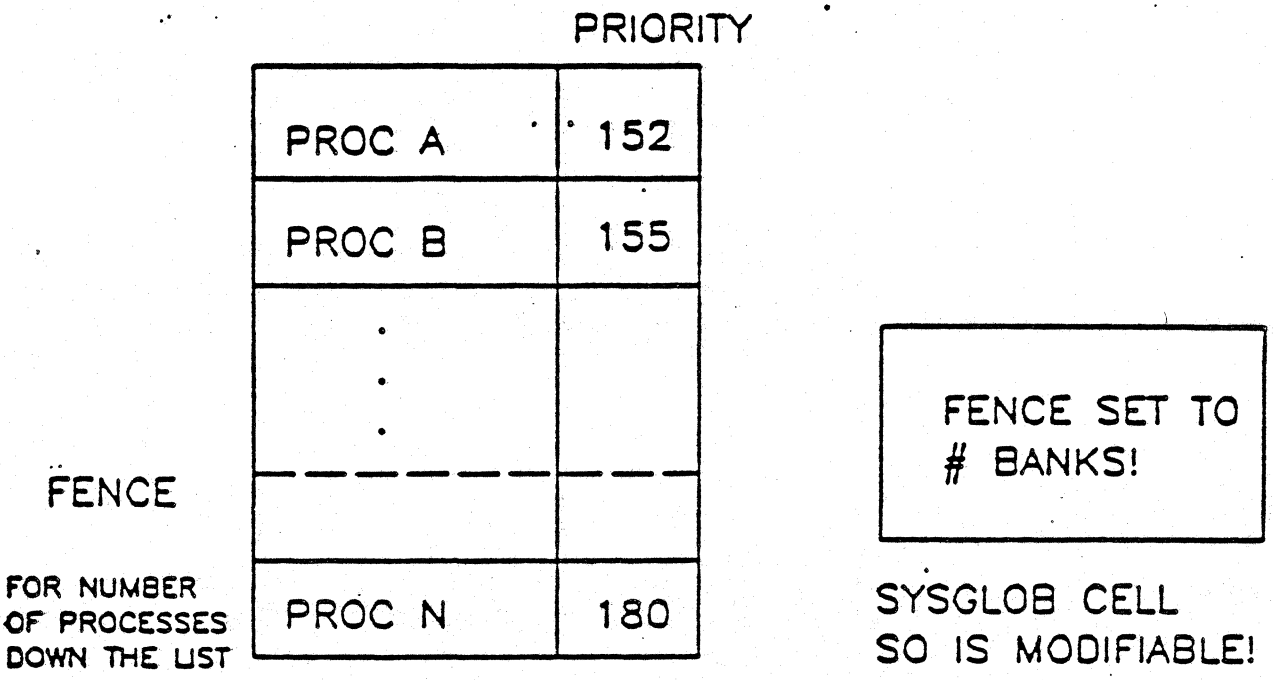
WHEN SHOULD THE ALGORITHM "GIVE UP"?

HOW CAN WE CONTROL THIS THRESHOLD?

MINCLOCKCYCLE [:TUNE COMMAND]

WHAT IS THE DEFAULT MINCLOCKCYCLE?

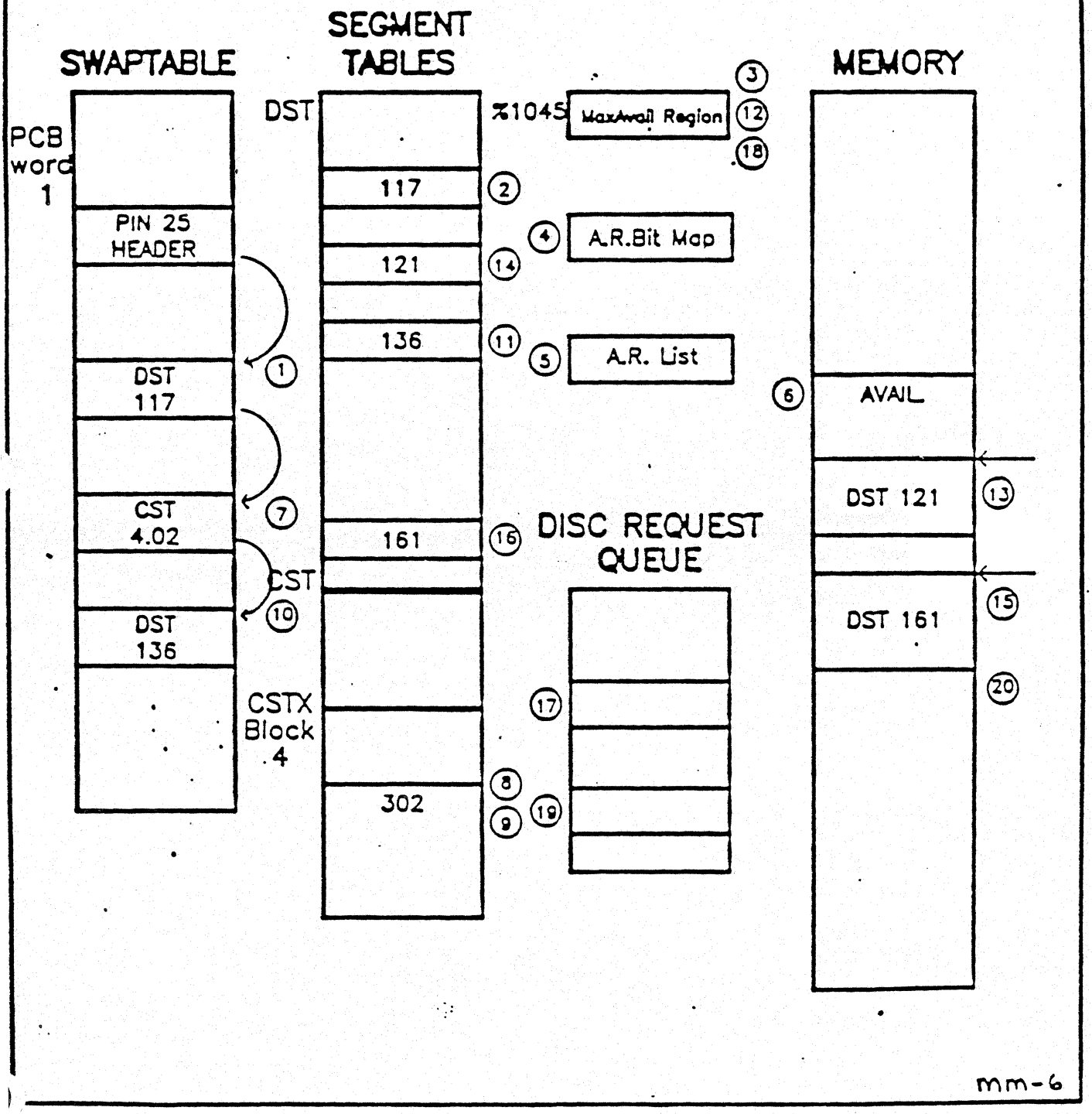
STATIC MULTIPROGRAMMING FENCE PREVENTS MEMORY MANAGER FROM TRYING TO DO "TOO MUCH"*



DISPATCH QUEUE

WHEN DISPATCHER FINDS PROCESS NEEDS MEMORY SCHEDULING IF THERE IS A MEMORY SHORTAGE A CHECK IS MADE TO SEE IF THE NUMBER OF SHORT WAITED PROCESSES ABOVE THE PROCESS IS EQUAL TO OR GREATER THAN THE STATIC MULTIPROGRAMMING FENCE. IF SO, DISPATCHER WILL SKIP TO THE NEXT PROCESS INSTEAD OF CALLING SWAPIN.

SWAPPING IN A PROCESS



mm-6

SWAPPING IN A PROCESS

1. SWAPIN USES PIN 25'S SLL HEADER TO FIND FIRST SEGMENT IN SLL TO BRING IN FOR PROCESS 25
2. SWAPIN CHECKS DATA SEGMENT TABLE ENTRY 117 AND FINDS THE SEGMENT IS ABSENT AND THAT THE SEGMENT SIZE IS 6 PAGES
3. FETCHSEGMENT CHECKS MAXAVAILREGION CELL AND FINDS MAXIMUM AVAILABLE SIZE IS 11 PAGES
4. FETCHSEGMENT CHECKS AVAILABLE REGION BITMAP TO FIND BEST FIT SIZE FOR THE SEGMENT, FINDS IT IS 10 PAGES
5. FETCHSEGMENT SELECTS FIRST AVAILABLE REGION OF SIZE = 10 PAGE
6. RESERVEREGION, CLEANREGION CLEAN OUT O.C.'S IN REGION AND SET UP REGION FOR DST 117; EXCESS PAGES ARE RETURNED
7. SWAPIN FINDS NEXT ENTRY (CST 4.02) IN SWAPTABLE
8. SWAPIN FINDS FROM CSTX THAT SEGMENT 4.02 IS AN OVERLAY CANDIDATE

SWAPPING IN A PROCESS (CONT.)

9. RECOVEROC TURNS OFF ROC BIT AND ABSENCE BIT IN CSTX AND FIXES UP REGION

10. SWAPIN FINDS NEXT ENTRY (DST 136) IN SWAPTABLE

11. SWAPIN FINDS ENTRY IN DST AND FINDS SEGMENT IS ABSENT AND NEEDS 13 PAGES

12. FETCHSEGMENT CHECKS MAXAVAILREGION CELL AND FINDS SIZE IS TO SMALL

13. MAKEROOM LOOKS AT ASSIGNED MEMORY REGION AT SCANPOINT

14. MAKEROOM LOOKS AT ENTRY IN DST FOR SEGMENT (121) IN MEMORY REGION; FINDS IT HAS BEEN REFERENCED SINCE LAST CYCLE; TURNS REFERENCE BIT OFF

15. MAKEROOM LOOKS AT NEXT ASSIGNED REGION IN MEMORY

SWAPPING IN A PROCESS (CONT.)

16. MAKEROOM LOOKS AT DST ENTRY 161 FOR THE SEGMENT IN THE REGION; FINDS IT HAS NOT BEEN REFERENCED SINCE LAST CYCLE; MAKEOC TURNS ON ROC AND ABSENCE BITS IN DST; MAKES AVAILABLE REGION TABLES CHANGES
17. MAKEOC SETS UP WRITE OF SEGMENT IN BACKGROUND DISC QUEUE
18. MAKEOC CHECKS MAXAVAILREGION CELL, FINDS SIZE IS BIG ENOUGH, BEST FIT REGION IS CLEANED AND RESERVED
19. PROCESSINITMSG QUEUES DISC READ REQUEST FOR DST 136 WHEN ALL WRITES FROM THE REGION HAVE COMPLETED
20. WHEN SIODM SIGNALS THAT READ OF DST 136 INTO MEMORY IS COMPLETE, SWAPIN IS COMPLETE

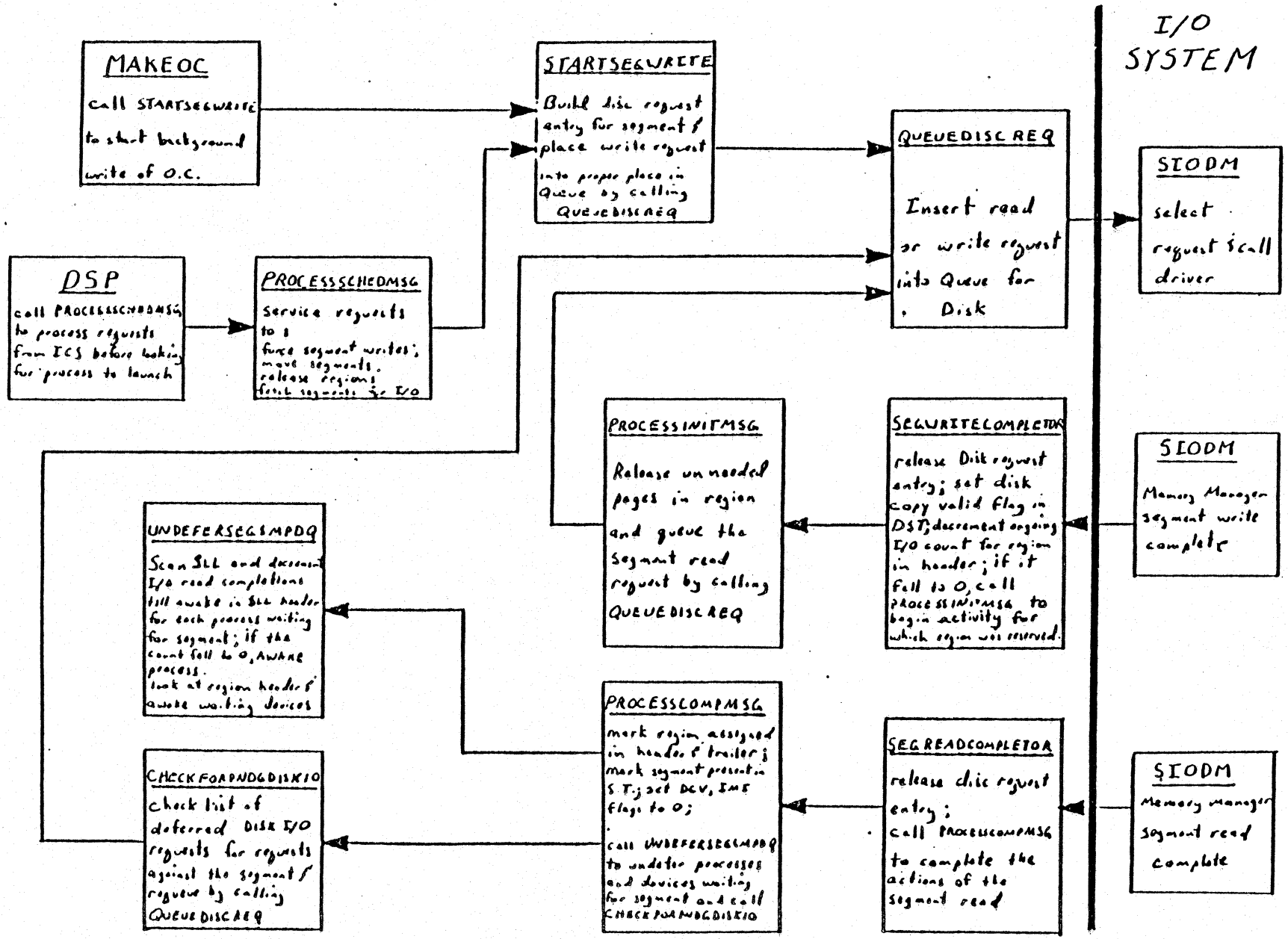
HEWLETT PACKARD

MPE IV

HOW MEMORY MANAGER INTERFACES WITH THE I/O SYSTEM

HEWLETT PACKARD

MEMORY MANAGER INTERFACE WITH I/O SYSTEM



REFERENCE SECTION

****** REGISTERS ******

DATA SEGMENT		CODE SEGMENT		MISCELLANEOUS		STATUS = 141077		ISR = 140017			
DB BANK	= 000001	PB	= .154423	X	= 001271	MODE	= PRIV	RUN/HALT	= HALT		
DB	= 053423	P	= 161203	CIR	= 051474	INTERRUPTS	= ON	IRQ	= OFF	TIMEOUT	= OFF
S BANK	= 000001	PL	= 171416	NIR	= 000000	TRAPS	= OFF	CSRQ	= OFF	NOT SS	= ON
DL	= 053267	PBBANK	= 000007			STACK OP	= LEFT	PARITY	= OFF	DISABLE ATN	= OFF
Q	= 060013	(P-PB)	= 004560			OVERFLOW	= OFF	POWERFAIL	= OFF		
S	= 060114					CARRY	= OFF	POWERON	= OFF		
Z	= 126711					COND CODE	= CCE	NOT DISP	= ON		
						SEGMENT #	= 77	NOT ICS	= ON		

****** FIXED LOW MEMORY ******

CODE SEGMENT TABLE POINTER	014650
EXTENDED CODE SEGMENT TABLE POINTER	016640
DATA SEGMENT TABLE POINTER	011550
PROCESS CONTROL BLOCK BASE	021350
CURRENT PCB POINTER	022730
INTERRUPT STACK BASE	023450
INTERRUPT STACK LIMIT	025066
INTERRUPT MASK	040120

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
1	ININ	PRIV	ON	OFF	4134	146500	0				S	C
2	FILESYS1 (0)	PRIV	ON	OFF	10764	123023	7				S	
3	FILESYS4 (1)	PRIV	ON	OFF	3550	150023	7				S	
4	FILESYS5 (2)	PRIV	ON	OFF	4240	106223	1				S	
5	FILESYS6 (3)	PRIV	ON	OFF	5154	054423	2				S	
6	FILESYS6A (4)	PRIV	ON	OFF	12170	135423	7				S	
7	FILESYS7 (5)	PRIV	ON	OFF	6220	005223	2				S	
10	CIALTORG (6)	PRIV	OFF	OFF	10224		1	54304			S	
11	CICOMSYS (7)	PRIV	OFF	OFF	4220		1	54354			S	
12	CIERR (10)	PRIV	ON	OFF	2400	104023	7				S	
13	CIFILEB (11)	PRIV	OFF	OFF	7704		1	54425			S	
14	CIFILEM (12)	PRIV	OFF	OFF	3304		1	54472			S	
15	CIINIT (13)	PRIV	ON	OFF	7244	004223	1				S	
16	CILISTF (14)	PRIV	OFF	OFF	6404		1	54571			S	
17	CIMISC (15)	PRIV	OFF	OFF	4504		1	54633			S	
20	CIORGMAN (16)	PRIV	OFF	OFF	6310		1	54661			S	
21	CIPREPRUN (17)	PRIV	ON	OFF	5570	004223	0				S	
22	CISUBS (20)	PRIV	ON	OFF	3724		1	54753			S	
23	CISYSMGR (21)	PRIV	ON	OFF	7334	144423	3				S	
24	CIUSERUTIL (22)	PRIV	ON	OFF	4444	012023	4				S	
25	CXSTOREST (23)	PRIV	OFF	OFF	5730		1	55072			S	
26	RESTORE (24)	PRIV	OFF	OFF	5574		1	55125			S	
27	STORE (25)	PRIV	OFF	OFF	10210		1	55162			S	
30	DIRC (26)	PRIV	ON	OFF	7444	134623	5				S	
31	ALLOCATE (27)	PRIV	OFF	OFF	6130		1	55270			S	
32	ALLOCUTIL (30)	PRIV	ON	OFF	7260	053623	6				S	
33	HARDRES (31)	PRIV	ON	OFF	34664	064654	0				S	C
34	ABORTDUMP (32)	PRIV	ON	OFF	6514	107223	2				S	
35	MESSAGE (33)	PRIV	ON	OFF	4230	154023	3				S	
36	PROCSEQ (34)	PRIV	ON	OFF	5330	101023	5				S	
37	NRIO (35)	PRIV	ON	OFF	7630	116023	2				S	
40	PCREATE (36)	PRIV	ON	OFF	10134	012223	6				S	
41	MORGUE (37)	PRIV	ON	OFF	4400	126023	2				S	
42	BIPC (40)	PRIV	OFF	OFF	3334		1	56034			S	
43	IPC (41)	PRIV	OFF	OFF	11174		1	56054			S	
44	CHECKER (42)	PRIV	ON	OFF	1764	171623	7				S	
45	UTILITY1 (43)	PRIV	ON	OFF	4544	107223	5				S	
46	UTILITY2 (44)	PRIV	OFF	OFF	6650		1	56164			S	
47	LOADER1 (45)	PRIV	ON	OFF	6030	126423	5				S	
50	RINS (46)	PRIV	OFF	OFF	3644		1	56275			S	
51	JOBTABLE (47)	PRIV	ON	OFF	5114	073623	5				S	
52	DEBUG (50)	PRIV	ON	OFF	20550	013623	2				S	
53	NURSERY (51)	PRIV	OFF	OFF	7310		1	56453			S	
54	SPOOLING (54)	PRIV	OFF	OFF	15640		1	56575			S	
55	SPOOLCOMS1 (55)	PRIV	OFF	OFF	6744		1	56670			S	
56	SPOOLCOMS2 (56)	PRIV	ON	OFF	12110	027623	6				S	
57	PVCOMSEG (57)	PRIV	OFF	OFF	3174		1	57003			S	
60	PVSYSD (60)	PRIV	OFF	OFF	5000		1	57022			S	

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH		ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
61	PVSYSM (61)	PRIV	OFF	OFF	7200	-		1	57050			S	
62	UDC (62)	USER	OFF	OFF	7644	-		1	57110			S	
63	USER (63)	USER	ON	OFF	3330	-	042023	6				S	
64	HELPUSER (64)	USER	OFF	OFF	2410			1	57172				
65	OPLOW (65)	PRIV	OFF	OFF	14020	-		1	57206			S	
66	OPMED (66)	PRIV	OFF	OFF	13540	-		1	57273			S	
67	OPHI (67)	PRIV	OFF	OFF	11340	-		1	57356			S	
70	LABSEG (70)	PRIV	OFF	OFF	13254	-		1	57427			S	
71	SDISC (71)	PRIV	OFF	OFF	12000	-		1	57507			S	
72	LOGSEG0 (73)	PRIV	OFF	OFF	12314	-		1	57570			S	
73	LOGSEG1 (74)	PRIV	OFF	OFF	13540	-		1	57645			S	
74	KERNELC (75)	PRIV	ON	OFF	23714	-	121540	0				S	C
75	KERNELD (76)	PRIV	ON	OFF	10260	-	104623	6				S	
76	MISCSEGC (77)	PRIV	ON	OFF	1024	-	145454	0				S	C
77	FILESYS1A (101)	PRIV	ON	OFF	14774	-	154423	7				S	
100	FILESYS2 (102)	PRIV	ON	OFF	7774	-	164023	0				S	
101	FILESYS3 (103)	PRIV	ON	OFF	10354	-	153023	1				S	
102	DEBUGUTL (104)	PRIV	OFF	OFF	4364	-		1	60365			S	
103	SEGUTIL (105)	PRIV	OFF	OFF	4424	-		1	60410			S	
104	KSAM01 (106)	PRIV	OFF	OFF	6324	-		1	60434			S	
105	KSAM02 (107)	PRIV	OFF	OFF	10764	-		1	60471			S	
106	KSAM03 (110)	PRIV	OFF	OFF	7724	-		1	60540			S	
107	KSAM04 (111)	PRIV	OFF	OFF	7004	-		1	60640			S	
110	KSAM05 (112)	PRIV	OFF	OFF	3070	-		1	60603			S	
111	FIRMWARESIM1 (52)	PRIV	OFF	OFF	5000	-		1	56514			S	
112	FIRMWARESIM2 (53)	PRIV	OFF	OFF	6330	-		1	56542			S	
113	KSAM06 (113)	USER	OFF	OFF	2410	-		1	60622			S	
114	KSAM07 (114)	USER	OFF	OFF	5044	-		1	60676			S	
115	COMSYS1 (135)	PRIV	OFF	OFF	10510	-		1	61567			S	
116	COMSYS3 (137)	PRIV	OFF	OFF	7274	-		1	61702			S	
117	COMSYS4 (140)	PRIV	OFF	OFF	7634	-		1	61745			S	
120	COMSYS5 (141)	PRIV	OFF	OFF	7504	-		1	62012			S	
121	CSUTILTY (142)	PRIV	OFF	OFF	12640	-		1	62057			S	
122	COMSYS2 (136)	PRIV	OFF	OFF	10274	-		1	61634			S	
123	BSCLCM (143)	PRIV	OFF	OFF	4310	-		1	62135			S	
124	BSCSLCP0 (144)	USER	OFF	OFF	1354	-		1	62163			S	
125	DVRSSLC (145)	PRIV	OFF	OFF	10500	-		1	62172			S	
126	DVRHSI (146)	PRIV	OFF	OFF	2154	-		1	62240			S	
127	DSSEG1 (147)	PRIV	OFF	OFF	4574	-		1	62253			S	
130	DSSEG2 (150)	PRIV	OFF	OFF	11234	-		1	62303			S	
131	DSSEG4 (152)	PRIV	OFF	OFF	7060	-		1	62405			S	
132	DSMISC (154)	PRIV	OFF	OFF	6004	-		1	62530			S	
133	DSIOM (155)	PRIV	ON	OFF	1550	-	156023	0				S	
134	DSSEG3 (151)	PRIV	OFF	OFF	5520	-		1	62354			S	
135	DSSEG5 (153)	PRIV	OFF	OFF	12540	-		1	62450			S	
136	CLIB'01 (200)	USER	OFF	OFF	6574			1	64227				
137	CLIB'03 (202)	USER	ON	OFF	7260		017023	4					
140	CLIB'04 (203)	USER	OFF	OFF	8530			1	64354				

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH		ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
141	CLIB'05 (204)	USER	OFF	OFF	5454			1	84412				
142	DSRTECALLS (158)	PRIV	OFF	OFF	7700	*		1	82577			S	
143	MRJEMISC1 (157)	PRIV	OFF	OFF	10740	*		1	82641			S	
144	MRJEMISC2 (160)	PRIV	OFF	OFF	8110	*		1	82711			S	
145	MPMONCMD (161)	PRIV	OFF	OFF	3470	*		1	82745			S	
146	IMAGE01 (210)	PRIV	OFF	OFF	6354			1	84553				
147	IMAGE02 (211)	PRIV	OFF	OFF	6244			1	84610				
150	IOMONITOR3270 (225)	PRIV	OFF	OFF	7114	*		1	85410			S	
151	HIOMDSC1	PRIV	ON	OFF	2430	*	152634	0				S	C
152	CSDUMMY	PRIV	OFF	OFF	70	*	155284	0				S	C
153	HIOTERM0	PRIV	ON	OFF	16764	*	156623	2				S	
154	HIOTAPE0	PRIV	ON	OFF	2380	*	007223	4				S	
155	HIOLPRT1	PRIV	OFF	OFF	1464	*		1	70150			S	
156	IOINP0	PRIV	OFF	OFF	10150	*		1	70230			S	
157	IOMPS0	PRIV	OFF	OFF	724	*		1	216457			S	
160	IOMPTRM0	PRIV	OFF	OFF	5820	*		1	216470			S	
161	IODS0	PRIV	OFF	OFF	1634	*		1	76704			S	
162	IODSTRM0	PRIV	OFF	OFF	2714	*		1	100016			S	
163		PRIV	ON	OFF	7004		000023	4					
164		USER	OFF	OFF	4104			1	84530				
165		USER	ON	OFF	7100		055423	5			I		

****** EXTENDED CST TABLE ******

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	R O C	I M I	S Y S	C R E S
301	1	PRIV	OFF	OFF	1614		1	70300			S	
301	2	PRIV	OFF	OFF	1604		1	72600			S	
301	3	PRIV	OFF	OFF	3060		1	74422			S	
301	4	PRIV	OFF	OFF	2264		1	70431			S	
301	5	PRIV	OFF	OFF	750		1	72466			S	
301	6	PRIV	ON	OFF	2350	175423	3			I	S	

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
301	7	PRIV	OFF	OFF	7054		1	70606			S	

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
301	10	PRIV	ON	OFF	5134	033423	1				S	

SEGMENT NUMBER	CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
301	11	USER	OFF	OFF	2404		1	225506				
302	11	USER	OFF	OFF	16200		1	225521				
303	11	USER	OFF	OFF	14250		1	225612				
304	11	USER	OFF	OFF	15210		1	225874				
305	11	USER	OFF	OFF	12260		1	225762				
306	11	USER	OFF	OFF	7144		1	226034				
307	11	USER	OFF	OFF	15324		1	226071				
310	11	USER	OFF	OFF	12564		1	226157				
311	11	USER	OFF	OFF	16020		1	226232				
312	11	USER	OFF	OFF	13604		1	226323				
313	11	USER	OFF	OFF	5674		1	226403				
314	11	USER	OFF	OFF	7774		1	226433				
315	11	USER	OFF	OFF	11250		1	226473				
316	11	USER	OFF	OFF	11304		1	226541				
317	11	USER	OFF	OFF	1664		1	226607				
320	11	USER	OFF	OFF	2270		1	226617				
321	11	USER	OFF	OFF	2304		1	226631				
322	11	USER	ON	OFF	3460	174023	7					
323	11	USER	OFF	OFF	3514	174223	0					
324	11	USER	OFF	OFF	2140		1	226701				
325	11	USER	OFF	OFF	4340	135423	1					
326	11	USER	ON	OFF	3764	131223	1					
327	11	USER	OFF	OFF	3524		1	226754				
330	11	USER	OFF	OFF	3320		1	226773				
331	11	USER	OFF	OFF	5504		1	227011				
332	11	USER	OFF	OFF	4574		1	227040				
333	11	USER	ON	OFF	5204	160423	3					
334	11	USER	OFF	OFF	2010		1	227111				
335	11	USER	ON	OFF	7254	144423	5					
336	11	USER	OFF	OFF	2330		1	227160				
337	11	USER	OFF	OFF	6024		1	227172				
340	11	USER	OFF	OFF	2344		1	227223				

341	11	USER	ON	OFF	5130	000023	2
342	11	USER	ON	OFF	4124	113023	7
343	11	USER	ON	OFF	4150	103023	2
344	11	USER	ON	OFF	2034	120223	7

SEGMENT NUMBER		CSTBLK/PROCESS INDX	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	ROC	IMI	SYS	CRES
301	*	12	USER	ON	OFF	1534	163623	1					
302		12	USER	ON	OFF	1484	176223	5					

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWIP	SYS	CRES	WD	VM ALLOC
1	(CODE SEGMENT TABLE)	OFF	1400	014650	0								S	C		0
2	(DATA SEGMENT TABLE)	ON	3100	011550	0								S	C		0
3	(PROCESS CONTROL BLOCK)	ON	2000	021350	0								S	C		0
4	(CST EXTENSION)	OFF	3100	016250	0								S	C		0
5	(SYSTEM GLOBAL AREA)	OFF	1120	001000	0								S	C		0
6	(FIXED LOW CORE)	ON	4000	000000	0								S	C		0
7	(INTERRUPT CONTROL STACK)	OFF	1520	023350	0								S	C		0
10	(SYSTEM BUFFERS)	ON	4434	045574	0								S	C		0
11	(UCOP REQUEST QUEUE)	OFF	104	187023	5			R					S			1
12	(PROCESS-PROCESS COMMUNICATION TABLE)	OFF	200		1	3372	D						S			1
13	(I/O QUEUE)	OFF	4240	025070	0								S	C		0
14	(TERMINAL BUFFERS)	OFF	6010	002120	0								S	C		0
15	(LOGICAL-PHYSICAL DEVICE TABLE)	ON	274	061530	0								S	C		0
16	(LOGICAL DEVICE AND CLASS TABLE)	ON	2020	174023	5								S			3
17	(DRIVER LINKAGE TABLE)	OFF	120	000600	0								S	C		0
20	(I/O RESOURCE TABLES)	OFF	20	000720	0								S	C		0
21	(DISK FREE SPACE)	ON	20000	064423	6								S			21
22	(LOADER SEGMENT TABLE)	OFF	2644	034423	2				I			F	S			14
23	(TIMER REQUEST LIST)	OFF	204	082024	0								S	C		0
24	(DIRECTORY)	ON	2000	181423	0								S			3
25	(DIRECTORY SPACE)	OFF	600		1	5116	D						S			1
26	(RIN TABLE)	OFF	1614		1	3136	D									0
27	(SWAPTABLE)	ON	3720	052230	0								S	C		0
30	(JOB PROCESS COUNT)	ON	30	000740	0								S	C		0
31	(JOB MASTER TABLE)	ON	600	115023	5								S			14
32	(TAPE LABEL TABLE)	OFF	1750		1	4142						F	S			2
33	(LOG TABLE)	OFF	170		1	3150	D						S			0
34	(REPLY INFORMATION TABLE)	OFF	2000		1	3352	D						S			3
35	(VOLUME TABLE)	ON	124	183623	0								S			1
36	(BREAKPOINT TABLE)	OFF	714		1	4246	D						S			1
37	(LOG BUFFER 1)	OFF	400		1	4252	D						S			1
40	(LOG BUFFER 2)	OFF	400		1	4256	D						S			1
41	(LOG ID TABLE)	OFF	150		1	3146	D						S			0
42	(ASSOCIATION TABLE)	OFF	1214		1	4172	D						S			2
43	(CST BLOCK)	OFF	50	056150	0								S	C		0
44	(JOB CUTOFF TABLE)	OFF	164	082230	0								S	C		0
45	(SYSTEM JIT)	OFF	100		1	3402	D						S			1
46	(SPECIAL REQUEST TABLE)	OFF	144	056220	0								S	C		0
47	(VIRTUAL DISK SPACE TABLE)	OFF	544	056714	0								S	C		0
51	(ARSBM TABLE)	OFF	44	057460	0								S	C		0
52	(ILT)	OFF	6024	037550	0								S	C		0
53	(SIR TABLE)	OFF	234	082414	0								S	C		0
54	(FILE MULTI-ACCESS VECTOR)	OFF	200	055023	3			R					S			2
55	(INPUT DEVICE DIRECTORY)	OFF	800		1	3472	D						S			40
56	(OUTPUT DEVICE DIRECTORY)	OFF	1600	112023	5			R					S			40
57	(WELCOME MESSAGE #1)	OFF	1750		1	4102	D						S			2

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWIP	SYS	CRES	WD	VM ALLOC
60	(WELCOME MESSAGE #2)	OFF	1750		1	4112	D						S			2
61	(CS SYSTEM SEGMENT)	OFF	340		1	3242	D						S			1
62	(JOB-PROCESS CROSS REFERENCE)	ON	100	027023	6								S			1
63	(SYSTEM JDT)	OFF	34		1	3406	D						S			1
64	(COMMAND INTERPRETER LOG-ON DST)	OFF	1000		1	4122	D						S			4
65	(MOUNTED VOLUME TAB.)	OFF	520		1	4202	D						S			1
66	(PRI. VOL. USER TABLE)	OFF	200		1	4206	D						S			10
67	(AVAILABLE REGION LIST)	OFF	2004	057524	0								S	C		0
70	(DISC REQUEST TABLE)	OFF	6220	031330	0								S	C		0
71	(MSG HBR TABLE)	OFF	10	056364	0								S	C		0
72	(PRIMARY MSG TABLE)	OFF	200	056374	0								S	C		0
73	(MEASUREMENT INFO TABLE)	OFF	120	056574	0								S	C		0
75		ON	3244	004223	7					S			S			7
76		OFF	3244		1	3206	D			S			S			7
77		OFF	3360		1	4262	D			S			S			7
100		OFF	12720		1	4316	D			S			S			16
101		OFF	2554		1	4406	D			S			S			6
102		OFF	2310		1	4436	D			S			S			6
103		OFF	2260		1	4466	D			S			S			6
104		ON	4764	137223	3					S			S			13
105		OFF	5364		1	4572	D			S			S			43
106		ON	5720	076023	7					S			S			17
107		ON	254	155423	0											1
110		ON	204	122423	7											1
111		OFF	3560	050023	2				I			F				12
112		ON	1404	013623	1											2
113		OFF	1324		1	10632	D									12
114		OFF	5324		1	5302	D			S						22
115		OFF	1110		1	11206	D									2
117		ON	4114	137023	2											5
120		OFF	1110		1	10332	D									2
121		OFF	4114		1	14326						F				5
122		OFF	1110		1	13342	D									2
123		OFF	50		1	10606	D									5
124		OFF	460		1	5762	D									1
125		OFF	7640		1	5766	D									10
126		OFF	104		1	12262	D									1
133		OFF	50		1	10166	D									5
134		ON	54674	000023	3					S						61
135		OFF	6774		1	5412	D			S						27
136		OFF	1110		1	11216	D									2
137		ON	55274	051623	1					S						61
140		ON	4114	172623	6											5
141		OFF	50		1	7242	D									5
142		ON	104	016623	4											1
143		OFF	1110	072423	5			R								2

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWIP	SYS	CRES	WD	VM ALLOC
144		OFF	4114	172623	1		D		I							5
145		ON	4114	026423	4											5
146		OFF	4114	000023	7											5
147		ON	4114	072423	2											5
151		OFF	1110		1	5666	D									2
152		OFF	104	177623	2			R								1
153		ON	104	152623	1											1
154		ON	50	177623	7											5
155		OFF	4114	000023	6											5
156		ON	100	071423	7											1
157		ON	4114	062023	2											5
160		OFF	6574		1	10026	D			S						27
161		OFF	130		1	5546	D									5
163		ON	6774	166223	3					S						27
164		OFF	104		1	11362	D									1
165		OFF	1110		1	7406	D									2
166		ON	4114	015423	1											5
170		OFF	104		1	7236	D									1
171		ON	4114	106423	5											5
173		OFF	1110		1	12072	D									2
174		ON	55274	007623	7					S						61
176		ON	12760	143423	2					S						54
177		ON	4114	043623	2											5
200		ON	4114	172023	4		D		I							5
201		ON	100	152423	1											1
202		OFF	1110		1	7416	D									2
203		OFF	6574		1	11226	D			S						27
204		ON	4114	122223	5											5
205		OFF	50		1	11366	D									5
206		OFF	104		1	10162	D									1
207		OFF	50		1	13212	D									5
210		ON	4114	066223	2											5
211		ON	4114	154023	5											5
212		OFF	1110		1	5676	D									2
213		OFF	1324		1	5712	D									12
215		OFF	6574		1	6442	D			S						27
216		ON	55274	000023	5					S						61
217		ON	4114	022423	6											5
221		OFF	1110		1	12036	D									2
222		ON	55274	032623	4					S						61
223		OFF	50		1	5616	D									5
224		OFF	6574		1	7102	D			S						27
225		OFF	1110		1	13016	D									2
226		ON	55274	055423	3					S						61
227		OFF	50		1	12266	D									5
230		OFF	104		1	5612	D									1

****** DST TABLE ******

SEGMENT NUMBER	SEGMENT DESCRIPTION	REFERENCE BIT	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	DCV	ROC	IMI	STK	MOD	FWIP	SYS	CRES	WD	VM ALLOC
231		OFF	1324		1	7286	D									12
232		OFF	1324		1	11412	D									12
233		OFF	1110		1	10342	D									2
234		ON	4114	041223	1											5
235		ON	4114	068223	5											5
237		ON	4114	045423	1											5
240		ON	4114	021623	1											5
241		OFF	104		1	10602	D									1
242		OFF	4114		1	13702	D									5
243		OFF	1324		1	10212	D									12
244		OFF	6574		1	13052	D			S						27
245		ON	4114	071623	7											5
246		ON	4114	027223	1											5
247		OFF	4114	146223	1											5
250		ON	4114	000023	1											5
251		OFF	4114	142023	1											5
252		OFF	1110		1	13332	D									2
253		OFF	6574		1	12126	D			S						27
254		OFF	6574		1	10446	D			S						27
255		ON	55274	110223	4					S						61
256		ON	4114	076623	2											5
257		OFF	1324		1	13236	D									12
260		OFF	4114		1	13656	D									5
261		OFF	1110		1	12762	D									2
262		ON	4114	116023	5											5
263		ON	4114	133023	3											5
264		OFF	1324		1	12312	D									12
265		ON	55274	115223	6					S						61
266		ON	4114	037423	2											5
267		ON	4114	165623	4											5
270		ON	4114	065223	7											5
271		ON	4114	106623	7											5
272		ON	4114	132623	2											5
274		ON	4114	046223	6											5
275		ON	4114	160223	5											5
276		OFF	6574		1	14636	D			S						27
277		ON	104	114623	5											1
300		ON	50	143223	2											1
302		OFF	1324		1	15022	D									12
303		OFF	1110	134223	7			R								2

****** PROCESS CONTROL BLOCK (1ST HALF) ******

WAIT STATE

		DATA -SEGMENTS--			-FAMILY TREE--				----------WAKEMASK----------														----------EVENTFLAGS----------												-PSEUDO INTERRUPTS--						--MISC---	
PIN	XDS	ADB	STK	OVALC	FTHR PIN	SON PIN	BRO PIN	OA	M	RGL	RLL	MAO	BIO	IOO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM	M	RGL	RLL	MAO	BIO	IOO	UCOP	JUNK	TIMER	MSG	SON	FATHR	IMP	SIR	TMOUT	MEM	PSIM	HK	SK	ST	HB	CY	BK	RITBK	PIOVR	PTYPE	CRIT	HSIR
1			108			2		S								J																									NORM									SYST		
2		A	75		1		3									J																									NORM									SYST		
3		A	78		1		4									J																									NORM									SYST		
4			77		1		5	F								J																									NORM									SYST		
5			100		1		6	F													I																			NORM									SYST	C	
6			101		1		7	F								J																									NORM									SYST		
7	11		102		1	61	10	F								J																									NORM									SYSTU		
10			103		1		11									J								M																	NORM									SYST		
11			104		1		12																	M																	NORM									SYST		
12			105		1		14									J																									NORM									SYST		
14		A	114		1			S					B																								S				M	NORM									SYST	C	
20			218		22			F																M																	NORM									USONM		
21	22		176		71			F																M																	NORM									USONM		H
22			215		7	20	31	F												S																					NORM									UMAIN		
30			228		33			F																M																	NORM									USONM		
31			224		7	57	33	F												S																					NORM									UMAIN		
32			254		7	58	55	F												S																					NORM									UMAIN		
33			160		7	30	32	F												S																					NORM									UMAIN		
41			255		53			F																M																	NORM									USONM		
44			222		54			F																M																	NORM									USONM		
47			174		55			F																M																	NORM									USONM	C	
50	144		134		51			F																M																	NORM									USONM	C	
51			278		7	50		F												S																					NORM									UMAIN		
52																																																		USER		
53			244		7	41	51	F												S																					NORM									UMAIN		
54			253		7	44	53	F												S																					NORM									UMAIN		
55			203		7	47	54	F												S																					NORM									UMAIN		
56			265		32			F																M																	NORM									USONM		
57			137		31			F																							I										NORM									USONM	C	
61	111		163		7		71	S																M																	NORM									UMAIN	C	H
71			135		7	21	22	S												S																					NORM									UMAIN		

****** PROCESS CONTROL BLOCK (2ND HALF) ******

			SCHEDULING INFORMATION																	RESOURCES				LIFE/DEATH			MISCELLANEOUS						
PIN		NQPIN	PQPIN	DISPQ	LQ	CQ	DQ	EQ	INTER	COR	PRI	HIPRI	USDQ	TRW	SW	LW	MP	PC	IPXP	HSPRI	SAR	SOV	CRIT	HSIR	PREV IMPD PIN	NEXT IMPD PIN	SC	LIVE	DEAD	FAC	BMS	PPC	PCST	PBXPTR	SLLPTR	BPT LNK	SYSTEM PROC NAME
1					L						61																	L			SNF	NUL		10	51367		PROGEN
2					L						62																	L			SNF	NUL			51235		SYSIO
3					L						175																	L			SNF	NUL			51247		IOMESS
4					L						62																	L			SNF	NUL		1	51261		LOG
5					L						175												C					L			SNF	NUL		2	51273		MEMLOG
6					L						175																	L			SNF	NUL		3	51305		
7					L						175																	L			SNF	NUL	CTX	4.001	4	51317		UCOP
10					L						12										S							L			SNF	NUL		5	51331		PFAIL
11		61		D	L						175				S													L			SNF	NUL		6	51343		DEVREC
12					L						218																	L			SNF	NUL		7	51355		LOAD
14					L						230												C					L			SNF	NUL			52120		
20		41	47	D			D				356				S													L		F	SNF	NUL		12	52341		
21		50	61	D		C			I		230				S									H						F	SNF	NUL		11	55011		
22							D				312					L											S	L			SNF	NUL			53534		
30		56	44	D			D				356				S													L		F	SNF	NUL		12	54114		
31							D				312					L											S	L			SNF	NUL			52144		
32							D				312					L											S	L			SNF	NUL			52813		
33							D				312					L											S	L			SNF	NUL			54684		
41		44	20	D			D				356				S													L		F	SNF	NUL		12	53243		
44		30	41	D			D				356				S													L		F	SNF	NUL		12	53123		
47		20	50	D			D				352				S								C				S	L		F	SNF	NUL	CST	77	12	52567		
50		47	21	D			D				312				S								C				S	L		F	SNF	NUL			12	52454		
51							D				312					L											S	L			SNF	NUL	CST	21		54121		
52																															SNF	NUL					
53							D				312					L											S	L			SNF	NUL			53224		
54							D				312					L											S	L			SNF	NUL			54821		
55							D				312					L											S	L			SNF	NUL			51545		
56		57	30	D			D				356				S													L		F	SNF	NUL		12	54802		
57	*		58	D			D				356				S								C				S	L		F	SNF	NUL		12	52303		
61		21	11	D		C			I		230				S								C	H			S	L			SNF	NUL			52517		
71						C			I		230					L											S	L			SNF	NUL			53510		

100 ENTRYS
40 UNASSIGNED ENTRYS
40 ASSIGNED ENTRYS

********** PROCESS SEGMENT LOCALITY LISTS **********

PIN: 1 FIRST SLL: 54373 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54373	CTX 10.001	51374	0								
51374	DST 108	0	54373			STK					

PIN: 2 FIRST SLL: 51242 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 1 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
51242	DST 75	0	0			STK					

PIN: 3 FIRST SLL: 53712 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 3 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53712	DST 37	51564	0								
51564	DST 112	51254	53712								
51254	DST 76	0	51564			STK					

PIN: 4 FIRST SLL: 54544 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54544	CTX 1.001	51200	0								
51200	DST 77	0	54544			STK					

PIN: 5 FIRST SLL: 53673 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53673	CTX 2.001	51300	0								
51300	DST 100	0	53673			STK					

PIN: 6 FIRST SLL: 51742 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
51742	CTX 3.001	51312	0								
51312	DST 101	0	51742			STK					

PIN: 7 FIRST SLL: 52461 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 11 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52461	DST 31	51704	0								
51704	DST 277	51463	52461								
51463	DST 276	53565	51704								
53565	DST 55	55100	51463								
55100	DST 58	54145	53565								
54145	DST 302	52365	55100								
52365	CTX 4.001	52721	54145								
52721	DST 11	51324	52365								
51324	DST 102	0	52721			STK					

PIN: 10 FIRST SLL: 51336 CURR SLL: 0 MEM REQ SLL: 51336 SLL COUNT: 1 IOCNT: 0 SWREQ

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
51336	DST 103	0	0			STK					

PIN: 11 FIRST SLL: 54128 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54128	CTX 6.001	51350	0							SLLIMI	
51350	DST 104	0	54128			STK					

PIN: 12 FIRST SLL: 53762 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 5 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53762	CST 7	54676	0								
54676	DST 125	52067	53762								
52067	DST 124	53010	54676								
53010	CTX 7.001	51362	52067								
51362	DST 105	0	53010			STK					

PIN: 14 FIRST SLL: 54234 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54234	DST 45	52125	0								
52125	DST 114	0	54234			STK					

PIN: 20 FIRST SLL: 52801 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER	NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52801	CST 165	53635	0		41					SLLIMI	
53635	DST 218	0	52801			STK					

PIN: 21 FIRST SLL: 54765 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54765	DST	22	53421	0							SLLIMI	
53421	DST	178	0	54765			STK					

PIN: 22 FIRST SLL: 52423 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52423	DST	216	55117	0								
55117	DST	112	55124	52423								
55124	DST	212	53231	55117								
53231	DST	151	52644	55124								
52644	DST	223	54506	53231								
54506	DST	215	0	52644			STK					

PIN: 30 FIRST SLL: 53736 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53736	CST	185	54277	0	44	56					SLLIMI	
54277	DST	228	0	53736			STK					

PIN: 31 FIRST SLL: 52276 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52276	DST	137	54417	0								
54417	DST	112	53445	52276								
53445	DST	202	52017	54417								
52017	DST	165	52466	53445								
52466	DST	141	53274	52017								
53274	DST	224	0	52466			STK					

PIN: 32 FIRST SLL: 52163 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52163	DST	265	51603	0								
51603	DST	112	54400	52163								
54400	DST	138	52555	51603								
52555	DST	115	54304	54400								
54304	DST	123	53267	52555								
53267	DST	254	0	54304			STK					

PIN: 33 FIRST SLL: 51552 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
51552	DST	226	55004	0								
55004	DST	112	51646	51552								
51646	DST	233	52214	55004								
52214	DST	120	53724	51646								
53724	DST	133	54431	52214								
54431	DST	160	0	53724			STK					

PIN: 41 FIRST SLL: 54753 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54753	CST	165	51470	0	20	44					SLLIMI	
51470	DST	255	0	54753			STK					

PIN: 44 FIRST SLL: 51627 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 10 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
51627	CST	165	55136	0	41	30					SLLIMI	
55136	DST	272	51413	51627								
51413	DST	271	52252	55136								
52252	DST	250	55042	51413								
55042	DST	274	53015	52252								
53015	CST	77	51711	55042								
51711	DST	267	54525	53015								
54525	DST	222	0	51711			STK					

PIN: 47 FIRST SLL: 53452 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 10 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53452	CST	3	54450	0								
54450	CST	77	52055	53452								
52055	DST	171	51475	54450								
51475	DST	242	53457	52055								
53457	DST	280	53320	51475								
53320	DST	121	53238	53457								
53238	DST	200	54272	53320							SLLIMI	
54272	DST	174	0	53238			STK					

PIN: 50 FIRST SLL: 53344 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 10 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53344	DST	144	51514	0							SLLIMI	
51514	DST	21	52745	53344								
52745	CST	32	52012	51514								
52012	CST	6	53717	52745								
53717	CST	3	52257	52012								
52257	DST	145	52264	53717								
52264	DST	148	54032	52257								
54032	DST	134	0	52264			STK					

PIN: 51 FIRST SLL: 55018 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 3 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
55018	CST	21	53147	0								
53147	DST	12	51677	55018								
51677	DST	276	0	53147			STK					

PIN: 52 FIRST SLL: 13650 CURR SLL: 10550 MEM REQ SLL: 20350 SLL COUNT: 58460 IOCNT: 0

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO

PIN: 53 FIRST SLL: 53755 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 10 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
53755	CST	7	54063	0								
54063	CST	77	53217	53755								
53217	DST	255	53471	54063								
53471	DST	112	51730	53217								
51730	DST	122	52271	53471								
52271	DST	252	53262	51730								
53262	DST	207	54253	52271								
54253	DST	244	0	53262			STK					

PIN: 54 FIRST SLL: 54424 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 11 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54424	CST	7	54501	0								
54501	CST	77	54532	54424								
54532	DST	222	54462	54501								
54462	DST	112	54215	54532								
54215	DST	225	54443	54462								
54443	DST	261	53077	54215								
53077	CST	3	52132	54443								
52132	DST	227	54102	53077								
54102	DST	253	0	52132			STK					

PIN: 55 FIRST SLL: 52778 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52778	DST	174	51521	0								
51521	DST	203	0	52778			STK					

PIN: 56 FIRST SLL: 52500 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 2 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
52500	CST	165	54607	0	30						SLLIMI	
54607	DST	265	0	52500			STK					

PIN: 57 FIRST SLL: 54176 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 0 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54176	DST	234	54203	0								
54203	DST	177	51723	54176								
51723	DST	266	54556	54203								
54556	DST	240	54575	51723								
54575	DST	235	53522	54556								
53522	DST	137	0	54575			STK					

PIN: 61 FIRST SLL: 54614 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 6 IOCNT: 1 HASMEM INTLC

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54614	DST	111	54140	0							SLLIMI	
54140	DST	110	54551	54614								
54551	DST	35	54107	54140								
54107	DST	24	54772	54551								
54772	CST	30	53154	54107								
53154	DST	163	0	54772			STK					

PIN: 71 FIRST SLL: 54703 CURR SLL: 0 MEM REQ SLL: 0 SLL COUNT: 5 IOCNT: 0 HASMEM

ENTRY INDEX	SEGMENT IDENTIFIER		NEXT ENTRY INDEX	PREV ENTRY INDEX	NEXT MAKE PRSNT DFRD QUEUE PIN	PREV MAKE PRSNT DFRD QUEUE PIN	STK	TOSS	FZREQ	LKREQ	SLLIMI	DISCIO
54703	DST	176	54227	0								
54227	DST	112	52043	54703								
52043	DST	158	54323	54227								
54323	DST	161	52137	52043								
52137	DST	135	0	54323			STK					

******** DISC REQUEST TABLE ******** (SUMMARY INFO)

TOTAL ENTRIES IN TABLE:	310
ENTRY SIZE:	20
ENTRIES IN PRIMARY AREA:	271
IMPEDED PROCESS PCB:	
TABLE INDEX OF FIRST AVAIL ENTRY:	2760
TABLE INDEX OF LAST AVAIL ENTRY:	3540
MAXIMUM NUMBER OF ENTRIES IN USE:	218
CURRENT NUMBER OF ENTRIES IN USE:	44
OVERFLOWS:	
TOTAL REQUESTS:	6044233
SYSBASE INDEX OF DISABLED Q HEAD:	34330
SYSBASE INDEX OF DISABLED Q TAIL:	31750

******** DISC REQUEST TABLE ******** (ACTIVE LISTS)

LDEV 1

STATUS: 0.XX -> PENDING
1.XX -> SUCCESSFUL
2.XX -> END OF FILE
3.XX -> UNUSUAL CONDITION
4.XX -> IRRECOVERABLE ERROR

TABLE INDEX	LDEV	UNIT	PCB	S S	DST/ BANK	OFFSET/ ADDRESS	FUNC	XFER CNT	PARM1	PARM2	MISC	SEQ	IDENT	SEGDSP	URGCLS	- F L A G S - MAIN	AUX	STATUS
001060*	1	0	57		235	002111	READ	1777	000003	057060	000000				356	000410	005320	0. 1
003740	1	0	57		240	002111	READ	1777	000003	060060	000000				356	000100	010200	0. 0
004640	1	0	57		266	000102	READ	1777	000003	032020	000000				356	000100	011100	0. 0
005700	1	0	57		177	000102	READ	1777	000003	041020	000000				356	000100	012140	0. 0
001760	1	0	57		234	000102	READ	1777	000003	050020	000000				356	000100	008220	0. 0
001240	1	0	57		235	000102	READ	1777	000003	057070	000000				356	000100	005500	0. 0
002200	1	0	57		240	000102	READ	1777	000003	060070	000000				356	000100	008440	0. 0
003400	1	0	57		266	002111	READ	1777	000003	032030	000000				356	000100	007640	0. 0
004660	1	0	57		177	002111	READ	1777	000003	041030	000000				356	000100	011120	0. 0
002040	1	0	57		234	002111	READ	1777	000003	050030	000000				356	000100	008300	0. 0
002420	1	0	41		4	175023	WRITE	2644	000000	005122	000000	DST	22	0	376	040100	006660	1. 0
004160	1	0	44		5	055623	WRITE	4114	000000	014326	000000	DST	121	0	376	040100	010420	1. 0
001360	1	0	56		1	172623	WRITE	3560	000000	005222	000000	DST	111	0	376	040100	005620	1. 0
000220	1	0	47		3	175423	WRITE	1750	000000	004142	000000	DST	32	0	376	040100	004460	1. 0
004440	1	0	47		2	177623	WRITE	104	000000	013206	000000	DST	152	0	377	040100	010700	1. 0
003300	1	0	44		5	167023	WRITE	104	000000	003366	000000	DST	11	0	377	040100	007540	1. 0
000120	1	0	44		5	112623	WRITE	1600	000000	003672	000000	DST	56	0	377	040100	004360	1. 0
005620	1	0	50		5	072423	WRITE	1110	000000	015072	000000	DST	143	0	377	040100	012060	1. 0
002060	1	0	56		7	134223	WRITE	1110	000000	015102	000000	DST	303	0	377	040100	008320	1. 0

******** DISC REQUEST TABLE ******** (ACTIVE LISTS)

STATUS: 0.XX -> PENDING
1.XX -> SUCCESSFUL
2.XX -> END OF FILE
3.XX -> UNUSUAL CONDITION
4.XX -> IRRECOVERABLE ERROR

TABLE INDEX	LDEV	UNIT	PCB	S S	DST/ BANK	OFFSET/ ADDRESS	FUNC	XFER CNT	PARM1	PARM2	MISC	SEQ IDENT	SEGDSP	URGCLS	- F L A G S - MAIN	AUX	STATUS
001200	1	0	56		3	055023	WRITE	200	000000	004072	000000	DST 54	0	377	040100	005440	1. 0

LDEV 2: NO CURRENT REQUEST.

```
030110: 100176 000404 000001 177656 000320 000004 010401 007000  030120: 000242 000025 000000 100176 002744 000001 000035 000080
030130: 000004 010401 017002 003055 000024 000000 000010 000000  030140: 000037 000000 000000 000000 000401 017004 003116 000024
030150: 000000 000010 000000 000037 000000 000001 000000 000401  030160: 007000 002766 000025 000000 100176 000333 000001 177656
030170: 000320 000004 010401 007000 001720 000025 000000 100176  030200: 000455 000001 177716 000320 000004 010401 011004 003131
030210: 000024 000000 000010 000212 000001 177777 000320 000000  030220: 000001 007004 003144 000024 000043 100106 000060 000000
030230: 177767 000005 000000 000401 011003 003157 000024 000000  030240: 000010 000413 000001 177735 000000 000000 000001 017002
030250: 000000 000024 000000 000010 000000 000037 000000 000000  030260: 000000 000401 007000 003205 000025 000000 100176 000455

030270: 000001 177656 000320 000004 010401 007000 003070 000025  030300: 000000 100176 000526 000001 177662 000320 000004 010401
030310: 007000 003350 000025 000000 100176 000333 000001 177656  030320: 000320 000004 010401 007000 002664 000025 000000 100176
030330: 000455 000001 177656 000320 000004 010401 007000 003274  030340: 000025 000032 100176 000333 000001 177656 000320 000004
030350: 010401 007000 003363 000025 000000 100176 000455 000001  030360: 177656 000320 000004 010401 007000 000741 000025 000000
030370: 100176 000404 000001 177656 000320 000004 010401 007000  030400: 003220 000025 000000 100176 000526 000001 177662 000320
030410: 000004 010401 007000 000754 000025 000032 100176 000455  030420: 000001 177656 000320 000004 010401 007000 003261 000025
030430: 000000 100176 000404 000001 177656 000320 000004 010401  030440: 007000 003322 000025 000000 100176 000404 000001 177656
030450: 000320 000004 010401 007000 003554 000025 000000 100176  030460: 000526 000001 177656 000320 000004 010401 007000 003704

030470: 000025 000000 100176 000333 000001 177656 000320 000004  030500: 010401 007000 003424 000025 000000 100176 000577 000001
030510: 177777 000320 000004 010401 007000 003437 000025 000000  030520: 100176 000333 000001 177656 000320 000004 010401 007000
030530: 003500 000025 000000 100176 000404 000001 177656 000320  030540: 000004 010401 007000 003465 000025 000000 100176 000526
030550: 000001 177667 000320 000004 010401 007000 003335 000025  030560: 000032 100176 000333 000001 177656 000320 000004 010401
030570: 007000 003452 000025 000000 100176 000455 000001 177656  030600: 000320 000004 010401 007000 004151 000025 000000 100176
030610: 003106 000001 000000 000000 000004 010401 007000 004136  030620: 000025 000000 100176 003106 000001 177706 000000 000004
030630: 010401 007000 000637 000025 000000 100176 002744 000001  030640: 000010 000000 000004 010401 007000 003567 000025 000000
030650: 100176 000577 000001 177772 000320 000004 010401 007000  030660: 003615 000025 000000 100176 000333 000001 177773 000320

030670: 000004 010401 007000 000064 000025 000000 100176 003106  030700: 000001 000000 000000 000004 010401 007000 003630 000025
030710: 000000 100176 000333 000001 177656 000320 000004 010401  030720: 007000 003643 000025 000000 100176 000404 000001 177722
030730: 000320 000004 010401 007000 002651 000025 000032 100176  030740: 000333 000001 177656 000320 000004 010401 007000 004006
030750: 000025 000000 100176 000000 000021 000000 000000 000000  030760: 010401 007000 002137 000025 000000 100176 000000 000014
030770: 000000 000000 000000 010401 007000 003760 000025 000000  031000: 100176 000404 000001 177725 000320 000004 010401 007000
031010: 004021 000025 000000 100176 000000 000011 000000 000000  031020: 000000 010401 007000 004110 000025 000000 100176 002744
031030: 000001 000000 000000 000004 010401 011003 002725 000024  031040: 000000 000010 004232 000001 177735 000000 000000 000001
031050: 007000 003773 000025 000000 100176 000333 000001 177765  031060: 000320 000004 010401 007000 003656 000025 000000 100176

031070: 000000 000015 000000 000000 000000 010401 017000 003717  031100: 000025 000000 000010 000000 000005 000000 000017 000000
031110: 010401 007000 004047 000025 000043 100176 000333 000000  031120: 177777 100001 000000 010401 007000 003671 000025 000000
031130: 100176 000000 000010 000001 000000 000000 010401 007000  031140: 004034 000025 000000 100176 000000 000020 000000 000000
031150: 000000 010401 007000 003513 000025 000043 100176 000333  031160: 000000 177771 000001 000000 010401 007000 004225 000025
031170: 000000 100176 003106 000001 177706 000000 000004 010401  031200: 007000 003027 000025 000000 100176 002744 000001 000035
031210: 000000 000004 010401 007000 004212 000025 000000 100176  031220: 003106 000001 177710 000000 000004 010401 007000 003602
031230: 000025 000000 100176 003106 000001 177706 000000 000004  031240: 010401 007000 004164 000025 000000 100176 003106 000001
031250: 177713 000060 000004 010401 007000 004123 000025 000000  031260: 100176 003106 000001 177707 000000 000004 010401 007000

031270: 003526 000025 000033 100176 003106 000001 177706 000000  031300: 000004 010401 007000 004075 000025 000000 100176 003106
031310: 000001 177706 000000 000004 010401 007000 000010 000025  031320: 000000 100176 003106 000001 177706 000000 000004 010401

$$$$$$$$ DST   70 (DISC REQUEST TABLE) $$$$$$$$
031330: 144271 000020 002780 003540 107044 000000 000030 044233  031340: 034330 031750 000000 000000 000000 000000 000000 000000
031350: 005010 002500 000001 000000 000235 000102 000000 001777  031360: 000003 056870 027401 000000 000000 000217 000000 004280
031370: 040000 000231 000001 000000 000002 034423 000000 002644  031400: 000000 005122 010000 000000 000000 000022 000000 004300
031410: 005010 004120 000001 000000 000235 002111 000000 001777  031420: 000003 056460 027401 000000 000000 000058 000000 004320
031430: 005010 003580 000001 000000 000235 000102 000000 001777  031440: 000003 056570 027401 000000 000000 040007 000000 004340
```

```
031450: 040100 000377 000001 000000 000005 112623 000001 001600  031460: 000000 003672 022001 033630 036150 000058 000000 004360
031470: 005010 002620 000001 000000 000177 000102 000000 001777  031500: 000003 040600 027401 000000 000000 000240 000000 004400
031510: 005010 005660 000001 000000 000240 002111 000000 001777  031520: 000003 065560 027401 000000 000000 000112 000000 004420

031530: 005010 001440 000001 000000 000266 002111 000000 001777  031540: 000003 031650 027401 000000 000000 000145 000000 004440
031550: 040100 000376 000001 000000 000003 175423 000001 001750  031560: 000000 004142 023401 031710 034770 000032 000000 004460
031570: 005010 001700 000001 000000 000240 000102 000000 001777  031600: 000003 065510 027401 000000 000000 000235 000000 004500
031610: 005010 002340 000001 000000 000177 000102 000000 001777  031620: 000003 040660 027401 000000 000000 000153 000000 004520
031630: 005010 005540 000001 000000 000234 002111 000000 001777  031640: 000003 047610 027401 000000 000000 000068 000000 004540
031650: 005010 006200 000001 000000 000240 002111 000000 001777  031660: 000003 065660 027401 000000 000000 000303 000000 004560
031670: 005010 002020 000001 000000 000177 002111 000000 001777  031700: 000003 040730 027401 000000 000000 000022 000000 004600
031710: 005010 001740 000001 000000 000177 000102 000000 001777  031720: 000003 040740 027401 000000 000000 000163 000000 004620

031730: 005010 005400 000001 000000 000240 002111 000000 001777  031740: 000003 068000 027401 000000 000000 000035 000000 004640
031750: 005010 004060 000001 000000 000234 002111 000000 001777  031760: 000003 047650 027401 000000 000000 000142 000000 004660
031770: 005010 004600 000001 000000 000266 000102 000000 001777  032000: 000003 031440 027401 000000 000000 000147 000000 004700
032010: 005010 000540 000001 000000 000240 002111 000000 001777  032020: 000003 065520 027401 000000 000000 040030 000000 004720
032030: 005010 006140 000001 000000 000234 002111 000000 001777  032040: 000003 047430 027401 000000 000000 000251 000000 004740
032050: 005010 004500 000001 000000 000240 000102 000000 001777  032060: 000003 065730 027401 000000 000000 000177 000000 004760
032070: 005010 002600 000001 000000 000266 000102 000000 001777  032100: 000003 031460 027401 000000 000000 000166 000000 005000
032110: 005010 003720 000001 000000 000234 000102 000000 001777  032120: 000003 047540 027401 000000 000000 102001 000000 005020

032130: 005010 000260 000001 000000 000266 000102 000000 001777  032140: 000003 031660 027401 000000 000000 000276 000000 005040
032150: 005010 000240 000001 000000 000235 000102 000000 001777  032160: 000003 056510 027401 000000 000000 000226 000000 005060
032170: 005010 005120 000001 000000 000235 002111 000000 001777  032200: 000003 056740 027401 000000 000000 000260 000000 005100
032210: 000004 000352 000001 000000 000200 002111 000000 001777  032220: 000001 146537 023400 033370 032550 104426 000000 005120
032230: 005010 001600 000001 000000 000177 002111 000000 001777  032240: 000003 040550 027401 000000 000000 000055 000000 005140
032250: 000004 000352 000001 000000 000260 000102 000000 001777  032260: 000001 142547 023400 032550 031750 104425 000000 005160
032270: 005010 005340 000001 000000 000234 002111 000000 001777  032300: 000003 047510 027401 000000 000000 000268 000000 005200
032310: 005010 004140 000001 000000 000240 000102 000000 001777  032320: 000003 066010 027401 000000 000000 104405 000000 005220

032330: 005010 004560 000001 000000 000235 000102 000000 001777  032340: 000003 056470 027401 000000 000000 000153 000000 005240
032350: 005010 003060 000001 000000 000177 000102 000000 001777  032360: 000003 040760 027401 000000 000000 000242 000000 005260
032370: 005010 003000 000001 000000 000235 002111 000000 001777  032400: 000003 057040 027401 000000 000000 104402 000000 005300
032410: 000410 000356 000001 000000 000235 002111 000000 001777  032420: 000003 057060 027410 000000 000000 000112 000000 005320
032430: 005010 003600 000001 000000 000177 000102 000000 001777  032440: 000003 040500 027401 000000 000000 000163 000000 005340
032450: 005010 000060 000001 000000 000234 002111 000000 001777  032460: 000003 047410 027401 000000 000000 040163 000000 005360
032470: 040000 000313 000001 000000 000001 172623 000000 004114  032500: 000000 015466 027000 000000 000000 000144 000000 005400
032510: 005010 005300 000001 000000 000177 000102 000000 001777  032520: 000003 040520 027401 000000 000000 000021 000000 005420

032530: 040100 000377 000001 000000 000003 055023 000001 000200  032540: 000000 004072 027001 032410 000000 000054 000000 005440
032550: 005010 005020 000001 000000 000235 002111 000000 001777  032560: 000003 056540 027401 000000 000000 000035 000000 005460
032570: 000100 000356 000001 000000 000235 000102 000000 001777  032600: 000003 057070 027400 032310 032530 000102 000000 005500
032610: 005010 004260 000001 000000 000240 002111 000000 001777  032620: 000003 065840 027401 000000 000000 000247 000000 005520
032630: 005010 001040 000001 000000 000234 002111 000000 001777  032640: 000003 047770 027401 000000 000000 000275 000000 005540
032650: 001010 000352 000001 000000 000171 002111 000000 001777  032660: 000001 144537 023401 000000 000000 000205 000000 005560
032670: 005010 003660 000001 000000 000266 000102 000000 001777  032700: 000003 031420 027401 000000 000000 000026 000000 005600
032710: 040100 000376 000001 000000 000001 172623 000001 003560  032720: 000000 005222 027001 034510 030550 000111 000000 005620

032730: 040000 000356 000001 000000 000005 055423 000000 007100  032740: 000000 064443 027000 000000 000000 040165 000000 005640
032750: 000004 000352 000001 000000 000242 000102 000000 001777  032760: 000001 143547 023400 031250 000000 040054 000000 005660
032770: 005010 000420 000001 000000 000177 002111 000000 001777  033000: 000003 040650 027401 000000 000000 000177 000000 005700
033010: 005010 001100 000001 000000 000266 000102 000000 001777  033020: 000003 031500 027401 000000 000000 000216 000000 005720
033030: 005010 003120 000001 000000 000266 002111 000000 001777  033040: 000003 031530 027401 000000 000000 040052 000000 005740
033050: 005010 004100 000001 000000 000234 002111 000000 001777  033060: 000003 047710 027401 000000 000000 000203 000000 005760
033070: 005010 006000 000001 000000 000177 002111 000000 001777  033100: 000003 040750 027401 000000 000000 000266 000000 006000
```

```
033110: 005010 004340 000001 000000 000235 000102 000000 001777  033120: 000003 057050 027401 000000 000000 000244 000000 006020

033130: 005010 003240 000001 000000 000234 002111 000000 001777  033140: 000003 047550 027401 000000 000000 104423 000000 006040
033150: 005010 002440 000001 000000 000177 000102 000000 001777  033160: 000003 040720 027401 000000 000000 000210 000000 006060
033170: 005010 004620 000001 000000 000235 000102 000000 001777  033200: 000003 058750 027401 000000 000000 000147 000000 006100
033210: 005010 002000 000001 000000 000266 002111 000000 001777  033220: 000003 031410 027401 000000 000000 000176 000000 006120
033230: 005010 006060 000001 000000 000266 002111 000000 001777  033240: 000003 031450 027401 000000 000000 040041 000000 006140
033250: 005010 003640 000001 000000 000235 000102 000000 001777  033260: 000003 057030 027401 000000 000000 000210 000000 006160
033270: 005010 002400 000001 000000 000234 000102 000000 001777  033300: 000003 047740 027401 000000 000000 000201 000000 006200
033310: 000100 000358 000001 000000 000234 000102 000000 001777  033320: 000003 050020 027400 036230 031570 000260 000000 006220

033330: 005010 001120 000001 000000 000177 002111 000000 001777  033340: 000003 040410 027401 000000 000000 000256 000000 006240
033350: 005010 005160 000001 000000 000234 002111 000000 001777  033360: 000003 047730 027401 000000 000000 000031 000000 006260
033370: 000100 000358 000001 000000 000234 002111 000000 001777  033400: 000003 050030 027400 035210 032750 000120 000000 006300
033410: 040100 000377 000001 000000 000007 134223 000001 001110  033420: 000000 015102 027001 036150 031530 000303 000000 006320
033430: 005010 000440 000001 000000 000240 002111 000000 001777  033440: 000003 065500 027401 000000 000000 000255 000200 006340
033450: 000004 000352 000001 000000 000260 002111 000000 001777  033460: 000001 142537 023400 033650 033370 000155 000000 006360
033470: 005010 001300 000001 000000 000177 002111 000000 001777  033500: 000003 040770 027401 000000 000000 000154 000000 006400
033510: 005010 005040 000001 000000 000240 002111 000000 001777  033520: 000003 066020 027401 000000 000000 000012 000000 006420

033530: 000100 000358 000001 000000 000240 000102 000000 001777  033540: 000003 066070 027400 031570 033730 000042 000000 006440
033550: 000004 000352 000001 000000 000121 002111 000000 001777  033560: 000001 151537 023400 031210 031250 000154 000000 006460
033570: 005010 002140 000001 000000 000266 002111 000000 001777  033600: 000003 031770 027401 000000 000000 000174 000200 006500
033610: 005010 004200 000001 000000 000266 002111 000000 001777  033620: 000003 032010 027401 000000 000000 000066 000000 006520
033630: 005010 005560 000001 000000 000177 002111 000000 001777  033640: 000003 040670 027401 000000 000000 000026 000000 006540
033650: 005010 001560 000001 000000 000234 000102 000000 001777  033660: 000003 050000 027401 000000 000000 040030 000000 006560
033670: 005010 004460 000001 000000 000234 000102 000000 001777  033700: 000003 047660 027401 000000 000000 000077 000000 006600
033710: 005010 006020 000001 000000 000234 000102 000000 001777  033720: 000003 047580 027401 000000 000000 000075 000000 006620

033730: 005010 000760 000001 000000 000235 000102 000000 001777  033740: 000003 057010 027401 000000 000000 040021 000000 006640
033750: 040100 000378 000001 000000 000004 175023 000001 002844  033760: 000000 005122 020401 032370 034510 000022 000000 006660
033770: 005010 003820 000001 000000 000234 000102 000000 001777  034000: 000003 047720 027401 000000 000000 000177 000000 006700
034010: 005010 002740 000001 000000 000266 000102 000000 001777  034020: 000003 032000 027401 000000 000000 000024 000000 006720
034030: 005010 003100 000001 000000 000240 000102 000000 001777  034040: 000003 065870 027401 000000 000000 000062 000000 006740
034050: 005010 000200 000001 000000 000240 000102 000000 001777  034060: 000003 065710 027401 000000 000000 000154 000000 006760
034070: 005010 001520 000001 000000 000177 002111 000000 001777  034100: 000003 040710 027401 000000 000000 000266 000000 007000
034110: 005010 001000 000001 000000 000234 000102 000000 001777  034120: 000003 047420 027401 000000 000000 000246 000000 007020

034130: 005010 005600 000001 000000 000177 000102 000000 001777  034140: 000003 040460 027401 000000 000000 000271 000000 007040
034150: 005010 004720 000001 000000 000234 000102 000000 001777  034160: 000003 047600 027401 000000 000000 000055 000000 007060
034170: 005010 000020 000001 000000 000234 000102 000000 001777  034200: 000003 047620 027401 000000 000000 000140 000000 007100
034210: 005010 005460 000001 000000 000240 000102 000000 001777  034220: 000003 065850 027401 000000 000000 000302 000000 007120
034230: 005010 005360 000001 000000 000240 000102 000000 001777  034240: 000003 065770 027401 000000 000000 000034 000000 007140
034250: 005010 003220 000001 000000 000266 002111 000000 001777  034260: 000003 031430 027401 000000 000000 000126 000000 007160
034270: 005010 002320 000001 000000 000177 000102 000000 001777  034300: 000003 041000 027401 000000 000000 040041 000000 007200
034310: 005010 001660 000001 000000 000240 000102 000000 001777  034320: 000003 065450 027401 000000 000000 000257 000000 007220

034330: 005010 002460 000001 000000 000240 002111 000000 001777  034340: 000003 066040 027401 000000 000000 000256 000000 007240
034350: 005010 004220 000001 000000 000240 002111 000000 001777  034360: 000003 065760 027401 000000 000000 000253 000000 007260
034370: 000004 000352 000001 000000 000242 002111 000000 001777  034400: 000001 143537 023400 032450 031210 000143 000000 007300
034410: 005010 001720 000001 000000 000234 000102 000000 001777  034420: 000003 047760 027401 000000 000000 000276 000000 007320
034430: 005010 005640 000001 000000 000266 002111 000000 001777  034440: 000003 031630 027401 000000 000000 000276 000000 007340
034450: 005010 005100 000001 000000 000177 002111 000000 001777  034460: 000003 040530 027401 000000 000000 040183 000000 007360
034470: 005010 003460 000001 000000 000234 002111 000000 001777  034500: 000003 047450 027401 000000 000000 040024 000000 007400
034510: 005010 002360 000001 000000 000177 000102 000000 001777  034520: 000003 040560 027401 000000 000000 000075 000000 007420
```

```
034530: 001010 000352 000001 000000 000171 000102 000000 001777  034540: 000001 144547 023401 000000 000000 000265 000000 007440
034550: 005010 000500 000001 000000 000177 002111 000000 001777  034560: 000003 040430 027401 000000 000000 104416 000000 007460
034570: 005010 003260 000001 000000 000235 002111 000000 001777  034600: 000003 056620 027401 000000 000000 000266 000000 007500
034610: 005010 006160 000001 000000 000240 002111 000000 001777  034620: 000003 065620 027401 000000 000000 040047 000000 007520
034630: 040100 000377 000001 000000 000005 187023 000001 000104  034640: 000000 003368 022001 034770 030450 000011 000000 007540
034650: 000004 000352 000001 000000 000121 000102 000000 001777  034660: 000001 151527 023400 034330 032450 000075 000000 007560
034670: 005010 004520 000001 000000 000177 002111 000000 001777  034700: 000003 040470 027401 000000 000000 000261 000000 007600
034710: 040000 000353 000001 000000 000004 172023 000000 004114  034720: 000000 013752 020400 000000 000000 000200 000000 007620

034730: 000100 000358 000001 000000 000266 002111 000000 001777  034740: 000003 032030 027400 032530 035210 000255 000200 007640
034750: 005010 004700 000001 000000 000268 000102 000000 001777  034760: 000003 031700 027401 000000 000000 000240 000000 007680
034770: 005010 006040 000001 000000 000235 002111 000000 001777  035000: 000003 056800 027401 000000 000000 000146 000000 007700
035010: 005010 000460 000001 000000 000235 002111 000000 001777  035020: 000003 056520 027401 000000 000000 000062 000000 007720
035030: 005010 003780 000001 000000 000240 000102 000000 001777  035040: 000003 065830 027401 000000 000000 040034 000000 007740
035050: 005010 002160 000001 000000 000235 002111 000000 001777  035060: 000003 057020 027401 000000 000000 000134 000200 007760
035070: 005010 000000 000001 000000 000234 002111 000000 001777  035100: 000003 050010 027401 000000 000000 040154 000000 010000
035110: 005010 001500 000001 000000 000240 000102 000000 001777  035120: 000003 065570 027401 000000 000000 000134 000200 010020

035130: 005010 005420 000001 000000 000234 000102 000000 001777  035140: 000003 047500 027401 000000 000000 000166 000000 010040
035150: 005010 002700 000001 000000 000235 000102 000000 001777  035160: 000003 056770 027401 000000 000000 000115 000000 010060
035170: 005010 002240 000001 000000 000240 000102 000000 001777  035200: 000003 066030 027401 000000 000000 000056 000000 010100
035210: 005010 002560 000001 000000 000177 000102 000000 001777  035220: 000003 040420 027401 000000 000000 000104 000000 010120
035230: 005010 000600 000001 000000 000240 002111 000000 001777  035240: 000003 065720 027401 000000 000000 040024 000000 010140
035250: 005010 006100 000001 000000 000235 000102 000000 001777  035260: 000003 056610 027401 000000 000000 040063 000000 010160
035270: 000100 000356 000001 000000 000240 002111 000000 001777  035300: 000003 066060 027400 000000 035170 000224 000000 010200
035310: 005010 005140 000001 000000 000266 002111 000000 001777  035320: 000003 031570 027401 000000 000000 040037 000000 010220

035330: 000004 000352 000001 000000 000200 000102 000000 001777  035340: 000001 146527 023400 000000 033650 000255 000000 010240
035350: 005010 002640 000001 000000 000177 000102 000000 001777  035360: 000003 040620 027401 000000 000000 000075 000000 010260
035370: 005010 002540 000001 000000 000266 002111 000000 001777  035400: 000003 031710 027401 000000 000000 000125 000000 010300
035410: 005010 003700 000001 000000 000235 002111 000000 001777  035420: 000003 056720 027401 000000 000000 000062 000000 010320
035430: 005010 003020 000001 000000 000235 002111 000000 001777  035440: 000003 056760 027401 000000 000000 040034 000000 010340
035450: 005010 001340 000001 000000 000240 002111 000000 001777  035460: 000003 065460 027401 000000 000000 000157 000000 010360
035470: 005010 001540 000001 000000 000266 002111 000000 001777  035500: 000003 031750 027401 000000 000000 000235 000000 010400
035510: 040100 000376 000001 000000 000005 055623 000001 004114  035520: 000000 014328 022001 032750 031710 000121 000000 010420

035530: 005010 003540 000001 000000 000177 002111 000000 001777  035540: 000003 041010 027401 000000 000000 000102 000000 010440
035550: 005010 001620 000001 000000 000266 000102 000000 001777  035560: 000003 031720 027401 000000 000000 000176 021000 010460
035570: 005010 001260 000001 000000 000235 002111 000000 001777  035600: 000003 056640 027401 000000 000000 000250 000000 010500
035610: 005010 000140 000001 000000 000266 000102 000000 001777  035620: 000003 031600 027401 000000 000000 000216 000000 010520
035630: 005010 002520 000001 000000 000235 000102 000000 001777  035640: 000003 056710 027401 000000 000000 000140 000000 010540
035650: 005010 004400 000001 000000 000240 000102 000000 001777  035660: 000003 065550 027401 000000 000000 000147 000000 010560
035670: 005010 002260 000001 000000 000240 000102 000000 001777  035700: 000003 066050 027401 000000 000000 000040 000000 010600
035710: 005010 004300 000001 000000 000234 000102 000000 001777  035720: 000003 047640 027401 000000 000000 040023 000000 010620

035730: 005010 005720 000001 000000 000266 002111 000000 001777  035740: 000003 031510 027401 000000 000000 000201 000000 010640
035750: 005010 005260 000001 000000 000235 000102 000000 001777  035760: 000003 056530 027401 000000 000000 104413 000000 010660
035770: 040100 000377 000001 000000 000002 177623 000001 000104  036000: 000000 013208 023401 030550 033630 000152 000000 010700
036010: 005010 000520 000001 000000 000235 000102 000000 001777  036020: 000003 056730 027401 000000 000000 040035 000000 010720
036030: 005010 002300 000001 000000 000266 002111 000000 001777  036040: 000003 031670 027401 000000 000000 104001 000000 010740
036050: 005010 001220 000001 000000 000234 002111 000000 001777  036060: 000003 047470 027401 000000 000000 000176 000000 010760
036070: 005010 004240 000001 000000 000234 002111 000000 001777  036100: 000003 047570 027401 000000 000000 000250 000000 011000
036110: 005010 002720 000001 000000 000240 000102 000000 001777  036120: 000003 065470 027401 000000 000000 000110 000000 011020

036130: 005010 005440 000001 000000 000177 000102 000000 001777  036140: 000003 040440 027401 000000 000000 000256 000000 011040
036150: 005010 004040 000001 000000 000240 000102 000000 001777  036160: 000003 065750 027401 000000 000000 000226 000000 011060
```

```
036170: 000100 000358 000001 000000 000266 000102 000000 001777  036200: 000003 032020 027400 034270 036230 000247 000000 011100
036210: 000100 000358 000001 000000 000177 002111 000000 001777  036220: 000003 041030 027400 033730 032370 000062 000000 011120
036230: 005010 006120 000001 000000 000177 000102 000000 001777  036240: 000003 040700 027401 000000 000000 104401 000000 011140
036250: 005010 002660 000001 000000 000235 000102 000000 001777  036260: 000003 056650 027401 000000 000000 000271 000000 011160
036270: 005010 005320 000001 000000 000235 002111 000000 001777  036300: 000003 058700 027401 000000 000000 000222 000000 011200
036310: 040000 000231 000001 000000 000002 050023 000000 003560  036320: 000000 005222 020400 000000 000000 000111 000000 011220)

036330: 005010 003340 000001 000000 000266 002111 000000 001777  036340: 000003 031470 027401 000000 000000 000255 000000 011240
036350: 005010 001460 000001 000000 000240 002111 000000 001777  036360: 000003 085540 027401 000000 000000 000024 000000 011260
036370: 005010 001020 000001 000000 000266 000102 000000 001777  036400: 000003 031760 027401 000000 000000 000110 000000 011300
036410: 005010 000560 000001 000000 000177 000102 000000 001777  036420: 000003 040540 027401 000000 000000 000222 000200 011320
036430: 005010 003440 000001 000000 000234 002111 000000 001777  036440: 000003 047530 027401 000000 000000 000130 000000 011340
036450: 005010 003420 000001 000000 000240 002111 000000 001777  036460: 000003 065740 027401 000000 000000 000122 000000 011360
036470: 005010 004540 000001 000000 000177 002111 000000 001777  036500: 000003 040570 027401 000000 000000 104441 000000 011400
036510: 005010 000400 000001 000000 000235 002111 000000 001777  036520: 000003 057000 027401 000000 000000 000105 000000 011420

036530: 005010 000300 000001 000000 000177 002111 000000 001777  036540: 000003 040810 027401 000000 000000 000117 000000 011440
036550: 005010 004740 000001 000000 000234 002111 000000 001777  036560: 000003 047830 027401 000000 000000 000227 000000 011460
036570: 005010 005060 000001 000000 000266 000102 000000 001777  036600: 000003 031540 027401 000000 000000 000112 000000 011500
036610: 005010 005000 000001 000000 000240 000102 000000 001777  036620: 000003 085530 027401 000000 000000 000204 000000 011520
036630: 005010 000100 000001 000000 000234 000102 000000 001777  036640: 000003 047520 027401 000000 000000 000267 000000 011540
036650: 005010 005760 000001 000000 000240 002111 000000 001777  036660: 000003 065700 027401 000000 000000 000235 000000 011560
036670: 005010 000160 000001 000000 000235 002111 000000 001777  036700: 000003 056580 027401 000000 000000 000105 000000 011600
036710: 005010 000340 000001 000000 000266 002111 000000 001777  036720: 000003 031730 027401 000000 000000 104424 000000 011620

036730: 005010 000360 000001 000000 000266 000102 000000 001777  036740: 000003 031740 027401 000000 000000 104430 000000 011640
036750: 005010 004320 000001 000000 000235 000102 000000 001777  036760: 000003 056550 027401 000000 000000 000042 000000 011660
036770: 005010 000620 000001 000000 000234 000102 000000 001777  037000: 000003 047440 027401 000000 000000 000012 000000 011700
037010: 005010 005200 000001 000000 000266 002111 000000 001777  037020: 000003 031610 027401 000000 000000 104403 000000 011720
037030: 005010 000700 000001 000000 000266 002111 000000 001777  037040: 000003 031550 027401 000000 000000 000216 000000 011740
037050: 005010 004360 000001 000000 000177 000102 000000 001777  037060: 000003 040640 027401 000000 000000 104435 000000 011760
037070: 005010 000320 000001 000000 000235 002111 000000 001777  037100: 000003 056660 027401 000000 000000 000106 000000 012000
037110: 005010 000640 000001 000000 000234 002111 000000 001777  037120: 000003 047670 027401 000000 000000 000024 000000 012020

037130: 005010 004420 000001 000000 000234 000102 000000 001777  037140: 000003 047460 027401 000000 000000 000270 000000 012040
037150: 040100 000377 000001 000000 000005 072423 000001 001110  037160: 000000 015072 024001 030450 032410 000143 000000 012060
037170: 005010 005220 000001 000000 000177 002111 000000 001777  037200: 000003 040630 027401 000000 000000 000126 000000 012100
037210: 005010 001160 000001 000000 000266 000102 000000 001777  037220: 000003 031520 027401 000000 000000 000031 000000 012120
037230: 000100 000358 000001 000000 000177 000102 000000 001777  037240: 000003 041020 027400 035170 032310 000155 000000 012140
037250: 005010 000740 000001 000000 000177 002111 000000 001777  037260: 000003 040510 027401 000000 000000 000263 000000 012160
037270: 040000 000175 000001 000000 000003 175423 000000 002350  037300: 000000 070354 022000 000000 000000 103001 000000 012200
037310: 005010 005520 000001 000000 000266 000102 000000 001777  037320: 000003 031640 027401 000000 000000 000200 000000 012220

037330: 005010 003520 000001 000000 000234 002111 000000 001777  037340: 000003 047750 027401 000000 000000 000143 000000 012240
037350: 005010 003500 000001 000000 000235 000102 000000 001777  037360: 000003 056630 027401 000000 000000 000130 000000 012260
037370: 005010 005240 000001 000000 000240 002111 000000 001777  037400: 000003 065600 027401 000000 000000 000147 000000 012300
037410: 005010 003140 000001 000000 000177 002111 000000 001777  037420: 000003 040450 027401 000000 000000 000126 000000 012320
037430: 005010 005500 000001 000000 000240 000102 000000 001777  037440: 000003 065610 027401 000000 000000 105002 000000 012340
037450: 005010 001640 000001 000000 000234 000102 000000 001777  037460: 000003 047700 027401 000000 000000 040070 000000 012360
037470: 005010 002100 000001 000000 000235 002111 000000 001777  037500: 000003 058500 027401 000000 000000 104427 000000 012400
037510: 005010 003160 000001 000000 000266 000102 000000 001777  037520: 000003 031560 027401 000000 000000 040063 000000 012420

037530: 005010 004020 000001 000000 000266 000102 000000 001777  037540: 000003 031620 027401 000000 000000 000181 000000 012440
```

$$$$$$$$ DST 52 (ILT) $$$$$$$$

```
037550: 000000 000000 000000 000000 000000 000000 000000 000010  037560: 038567 000000 000000 000000 105000 000000 007130 011415
037570: 005012 011415 006005 014400 006014 002401 003002 003427  037600: 000021 000010 020000 020441 020400 000000 020000 000031
037610: 002002 000001 000001 122000 037574 002003 000001 000001  037620: 122000 037575 002003 000001 000001 122000 037576 001000
037630: 000000 002403 000001 000002 000013 000117 000011 000600  037640: 000001 001400 000001 170401 142000 037601 000000 000252
037650: 000000 177755 002000 000000 000001 160000 004554 000601  037660: 000001 000000 177744 002005 000001 000001 162000 037604
037670: 000000 000000 002003 000001 000001 122000 037575 001000  037700: 000000 002000 000000 000001 120000 037601 000000 000002
037710: 000000 177765 002002 000001 000001 162000 037574 002003  037720: 000001 000001 162000 037575 002003 000001 000001 122000
037730: 037576 001000 000000 002403 000000 000002 000013 000015  037740: 000011 000600 000002 001400 000001 170401 142000 037601

037750: 000602 000022 000000 177755 000600 000002 001401 000001  037760: 001001 102000 037601 000000 000032 001402 000001 000001
037770: 102000 037600 002003 000001 000001 162000 037576 000602  040000: 000006 002003 000001 000001 162000 037575 000000 000000
040010: 002003 000001 000001 162000 037576 000601 000002 001400  040020: 000001 000001 142000 037601 002003 000001 000001 162000
040030: 037576 000602 000004 002003 000001 000001 162000 037576  040040: 000602 000000 002002 000001 000001 162000 037605 002007
040050: 000001 000001 162000 037606 002007 000001 000001 122000  040060: 037606 000602 000032 002001 000001 000001 122000 037573
040070: 000602 000024 002001 000001 000001 122000 037573 002006  040100: 000001 000001 162000 037577 000000 177604 002006 000001
040110: 000001 122000 037577 002006 000001 000001 162000 037577  040120: 000000 177741 002003 000001 000001 162000 037576 000602
040130: 000003 002003 000001 000001 162000 037576 002003 000001  040140: 000001 162000 037575 000000 177564 000403 000001 001000

040150: 000000 002403 000000 000002 177575 177577 177770 000600  040160: 000002 001400 000012 111001 100000 005032 000602 000002
040170: 000000 177755 000403 000002 000000 177533 002002 000001  040200: 000001 162000 037574 002003 000001 000001 122000 037576
040210: 000000 177735 177777 030360 030360 030360 030360 030360  040220: 030360 030360 030360 030360 030360 030360 030360 030360
040230: 030360 030360 030360 030360 030360 030360 030360 030360  040240: 030360 030360 030360 030360 030360 030360 030360 030360
LINES 040250 - 040267 SAME AS ABOVE

040270: 030360 002001 000001 000001 122000 040401 002007 000001  040300: 000001 162000 040402 002003 000001 000001 162000 040401
040310: 002004 000000 000001 120000 040615 000000 000004 001000  040320: 000000 000000 177765 002005 000000 000001 120000 040411
040330: 000000 000002 000000 177767 002006 000001 000001 122000  040340: 040412 002006 000001 000001 162000 040412 002003 000001
040350: 000001 122000 040402 000000 000017 002002 000001 000001  040360: 162000 000000 002007 000001 000001 122000 000000 002007
040370: 000001 000001 162000 000000 001000 000000 000000 177774  040400: 177777 114404 000040 000410 006406 010430 014433 011023
040410: 005001 000000 003427 000007 000050 000001 001003 002005  040420: 003007 004011 005013 006015 007017 010021 011023 012025
040430: 013027 014031 015033 016035 017037 020041 021043 022045  040440: 023047 024051 025053 026055 027057 030061 031063 032065
040450: 033067 034071 035073 036075 037077 040101 041103 042105  040460: 043107 044111 045113 046115 047117 050121 051123 052125

040470: 053127 054131 055133 056135 057137 060141 061143 062145  040500: 063147 064151 065153 066155 067157 070161 071163 072165
040510: 073167 074171 075173 076175 077177 100201 101203 102205  040520: 103207 104211 105213 106215 107217 110221 111223 112225
040530: 113227 114231 115233 116235 117237 120241 121243 122245  040540: 123247 124251 125253 126255 127257 130261 131263 132265
040550: 133267 134271 135273 136275 137277 140301 141303 142305  040560: 143307 144311 145313 146315 147317 150321 151323 152325
040570: 153327 154331 155333 156335 157337 160341 161343 162345  040600: 163347 164351 165353 166355 167357 170361 171363 172365
040610: 173367 174371 175373 176375 177377 000000 000000 000000  040620: 000000 000000 000000 000000 000011 037634 000000 000000
040630: 000000 105000 000000 007175 011415 005012 011415 006005  040640: 014400 006014 002401 003002 003427 000021 000000 020000
040650: 000000 000000 000000 000000 000031 002002 000001 000001  040660: 122000 040641 002003 000001 000001 122000 040642 002003

040670: 000001 000001 122000 040643 001000 000000 002403 000001  040700: 000002 000013 000117 000011 000600 000001 001400 000001
040710: 170401 142000 040646 000000 000252 000000 177755 002000  040720: 000000 000001 120000 002445 000601 000001 000000 177744
040730: 002005 000001 000001 162000 040651 000000 000000 002003  040740: 000001 000001 122000 040642 001000 000000 002000 000000
040750: 000001 160000 040635 000601 000003 000000 177765 002002  040760: 000001 000001 162000 040641 002003 000001 000001 162000
040770: 040642 002003 000001 000001 122000 040643 001000 000000  041000: 002403 000000 000002 000013 000015 000011 000600 000002
041010: 001400 000001 170401 142000 040646 000602 000010 000000  041020: 177755 000600 000002 001401 000001 001001 102000 040646
041030: 000000 000032 001402 000001 000001 102000 040653 002003  041040: 000001 000001 162000 040643 000602 000006 002003 000001
041050: 000001 162000 040642 000000 000000 002003 000001 000001  041060: 162000 040643 000601 000002 001400 000001 000001 142000

041070: 040646 002003 000001 000001 162000 040643 000602 000004  041100: 002003 000001 000001 162000 040643 000602 000000 002002
041110: 000001 000001 162000 040652 002007 000001 000001 162000  041120: 040653 002007 000001 000001 122000 040653 000602 000032
041130: 002001 000001 000001 122000 040640 000602 000024 002001  041140: 000001 000001 122000 040640 002006 000001 000001 162000
041150: 040644 000000 177604 002006 000001 000001 122000 040644  041160: 002006 000001 000001 162000 040644 000000 177741 002003
```

```
041170: 000001 000001 162000 040643 000602 000003 002003 000001  041200: 000001 162000 040643 002003 000001 000001 162000 040642
041210: 000000 177564 000403 000001 001000 000000 002403 000000  041220: 000002 177575 177577 177770 000600 000002 001400 000074
041230: 002001 100000 005472 000602 000002 000000 177755 000403  041240: 000002 000000 177533 002002 000001 000001 162000 040641
041250: 002003 000001 000001 122000 040643 000000 177735 177777  041260: 000000 000000 000000 000000 000000 000000 000000 000012

041270: 040277 000000 000000 000000 105000 000000 007242 011415  041300: 005012 011415 006005 114403 006014 002401 003002 000027
041310: 000021 000000 020000 000000 000000 000000 000000 000031  041320: 002002 000001 000001 122000 041304 002003 000001 000001
041330: 122000 041305 002003 000001 000001 122000 041306 001000  041340: 000000 002403 000000 000002 000013 000117 000011 000600
041350: 000001 001400 000001 170401 142000 041311 000000 000252  041360: 000000 177755 000000 000000 000000 000000 000000 000000
041370: 000000 000000 177744 002005 000001 000001 162000 041314  041400: 000000 000000 002003 000001 000001 122000 041305 001000
041410: 000000 000000 000000 000000 000000 000000 000000 000000  041420: 000000 177765 002002 000001 000001 162000 041304 002003
041430: 000001 000001 162000 041305 002003 000001 000001 122000  041440: 041306 001000 000000 002403 000000 000002 000013 000015
041450: 000011 000600 000002 001400 000001 000001 142000 041311  041460: 000602 000010 000000 177755 000600 000002 001401 000001

041470: 001001 102000 041311 000000 000032 001402 000001 000001  041500: 102000 041316 002003 000001 000001 162000 041306 000602
041510: 000006 002003 000001 000001 162000 041305 000000 000000  041520: 002003 000001 000001 162000 041306 000601 000002 001400
041530: 000001 000001 142000 041311 002003 000001 000001 162000  041540: 041306 000602 000004 002003 000001 000001 162000 041306
041550: 000602 000000 002002 000001 000001 162000 041315 002007  041560: 000001 000001 162000 041316 002007 000001 000001 122000
041570: 041316 000602 000032 002001 000001 000001 122000 041303  041600: 000602 000024 002001 000001 000001 122000 041303 002006
041610: 000001 000001 162000 041307 000000 177604 002006 000001  041620: 000001 122000 041307 002006 000001 000001 162000 041307
041630: 000000 177741 002003 000001 000001 162000 041306 000602  041640: 000003 002003 000001 000001 162000 041306 002003 000001
041650: 000001 162000 041305 000000 177564 000403 000001 001000  041660: 000000 002403 000000 000002 177575 177577 177770 000600

041670: 000002 001400 000000 000000 000000 000000 000602 000002  041700: 000000 177755 000403 000002 000000 177533 002002 000001
041710: 000001 162000 041304 002003 000001 000001 122000 041306  041720: 000000 177735 177777 000000 000000 000000 000000 000000
041730: 000000 000000 000013 040742 000000 000000 000000 105000  041740: 000000 007307 011415 005012 011415 006005 114403 006014
041750: 002401 003002 000027 000021 000000 020000 000000 000000  041760: 000000 000000 000031 002002 000001 000001 122000 041747
041770: 002003 000001 000001 122000 041750 002003 000001 000001  042000: 122000 041751 001000 000000 002403 000000 000002 000013
042010: 000117 000011 000600 000001 001400 000001 170401 142000  042020: 041754 000000 000252 000000 177755 000000 000000 000000
042030: 000000 000000 000000 000000 000000 177744 002005 000001  042040: 000001 162000 041757 000000 000000 002003 000001 000001
042050: 122000 041750 001000 000000 000000 000000 000000 000000  042060: 000000 000000 000000 000000 177765 002002 000001 000001

042070: 162000 041747 002003 000001 000001 162000 041750 002003  042100: 000001 000001 122000 041751 001000 000000 002403 000000
042110: 000002 000013 000015 000011 000600 000002 001400 000001  042120: 000001 142000 041754 000602 000010 000000 177755 000600
042130: 000002 001401 000001 001001 102000 041754 000000 000032  042140: 001402 000001 000001 102000 041761 002003 000001 000001
042150: 162000 041751 000602 000006 002003 000001 000001 162000  042160: 041750 000000 000000 002003 000001 000001 162000 041751
042170: 000601 000002 001400 000001 000001 142000 041754 002003  042200: 000001 000001 162000 041751 000602 000004 002003 000001
042210: 000001 162000 041751 000602 000000 002002 000001 000001  042220: 162000 041760 002007 000001 000001 162000 041761 002007
042230: 000001 000001 122000 041761 000602 000032 002001 000001  042240: 000001 122000 041746 000602 000024 002001 000001 000001
042250: 122000 041746 002006 000001 000001 162000 041752 000000  042260: 177604 002006 000001 000001 122000 041752 002006 000001

042270: 000001 162000 041752 000000 177741 002003 000001 000001  042300: 162000 041751 000602 000003 002003 000001 000001 162000
042310: 041751 002003 000001 000001 162000 041750 000000 177564  042320: 000403 000001 001000 000000 002403 000000 000002 177575
042330: 177577 177770 000600 000002 001400 000000 000000 000000  042340: 000000 000602 000002 000000 177755 000403 000002 000000
042350: 177533 002002 000001 000001 162000 041747 002003 000001  042360: 000001 122000 041751 000000 177735 177777 000000 000000
042370: 000000 000000 000000 000000 000000 000014 041405 000000  042400: 000000 000000 105000 000000 007354 011415 005012 011415
042410: 006005 114403 006014 002401 003002 000027 000021 000000  042420: 020000 000000 000000 000000 000000 000031 002002 000001
042430: 000001 122000 042412 002003 000001 000001 122000 042413  042440: 002003 000001 000001 122000 042414 001000 000000 002403
042450: 000000 000002 000013 000117 000011 000600 000001 001400  042460: 000001 170401 142000 042417 000000 000252 000000 177755

042470: 000000 000000 000000 000000 000000 000000 000000 000000  042500: 177744 002005 000001 000001 162000 042422 000000 000000
042510: 002003 000001 000001 122000 042413 001000 000000 000000  042520: 000000 000000 000000 000000 000000 000000 000000 177765
042530: 002002 000001 000001 162000 042412 002003 000001 000001  042540: 162000 042413 002003 000001 000001 122000 042414 001000
042550: 000000 002403 000000 000002 000013 000015 000011 000600  042560: 000002 001400 000001 000001 142000 042417 000602 000010
042570: 000000 177755 000600 000002 001401 000001 001001 102000  042600: 042417 000000 000032 001402 000001 000001 102000 042424
042610: 002003 000001 000001 162000 042414 000602 000006 002003  042620: 000001 000001 162000 042413 000000 000000 002003 000001
```

```
042630: 000001 162000 042414 000601 000002 001400 000001 000001  042640: 142000 042417 002003 000001 000001 162000 042414 000602
042650: 000004 002003 000001 000001 162000 042414 000602 000000  042660: 002002 000001 000001 162000 042423 002007 000001 000001

042670: 162000 042424 002007 000001 000001 122000 042424 000602  042700: 000032 002001 000001 000001 122000 042411 000602 000024
042710: 002001 000001 000001 122000 042411 002006 000001 000001  042720: 162000 042415 000000 177604 002006 000001 000001 122000
042730: 042415 002006 000001 000001 162000 042415 000000 177741  042740: 002003 000001 000001 162000 042414 000602 000003 002003
042750: 000001 000001 162000 042414 002003 000001 000001 162000  042760: 042413 000000 177564 000403 000001 001000 000000 002403
042770: 000000 000002 177575 177577 177770 000600 000002 001400  043000: 000000 000000 000000 000000 000602 000002 000000 177755
043010: 000403 000002 000000 177533 002002 000001 000001 162000  043020: 042412 002003 000001 000001 122000 042414 000000 177735
043030: 177777 000000 000000 000000 000000 000000 000000 000000  043040: 000015 042050 000000 000000 000000 105000 000000 007421
043050: 011415 005012 011415 006005 114403 006014 002401 003002  043060: 000027 000021 000000 020000 000000 000000 000000 000000

043070: 000031 002002 000001 000001 122000 043055 002003 000001  043100: 000001 122000 043056 002003 000001 000001 122000 043057
043110: 001000 000000 002403 000000 000002 000013 000117 000011  043120: 000600 000001 001400 000001 170401 142000 043062 000000
043130: 000252 000000 177755 000000 000000 000000 000000 000000  043140: 000000 000000 000000 177744 002005 000001 000001 162000
043150: 043065 000000 000000 002003 000001 000001 122000 043056  043160: 001000 000000 000000 000000 000000 000000 000000 000000
043170: 000000 000000 177765 002002 000001 000001 162000 043055  043200: 002003 000001 000001 162000 043056 002003 000001 000001
043210: 122000 043057 001000 000000 002403 000000 000002 000013  043220: 000015 000011 000600 000002 001400 000001 000001 142000
043230: 043062 000602 000010 000000 177755 000600 000002 001401  043240: 000001 001001 102000 043062 000000 000032 001402 000001
043250: 000001 102000 043067 002003 000001 000001 162000 043057  043260: 000602 000006 002003 000001 000001 162000 043056 000000

043270: 000000 002003 000001 000001 162000 043057 000601 000002  043300: 001400 000001 000001 142000 043062 002003 000001 000001
043310: 162000 043057 000602 000004 002003 000001 000001 162000  043320: 043057 000602 000000 002002 000001 000001 162000 043066
043330: 002007 000001 000001 162000 043067 002007 000001 000001  043340: 122000 043067 000602 000032 002001 000001 000001 122000
043350: 043054 000602 000024 002001 000001 000001 122000 043054  043360: 002006 000001 000001 162000 043060 000000 177604 002006
043370: 000001 000001 122000 043060 002006 000001 000001 162000  043400: 043060 000000 177741 002003 000001 000001 162000 043057
043410: 000602 000003 002003 000001 000001 162000 043057 002003  043420: 000001 000001 162000 043056 000000 177564 000403 000001
043430: 001000 000000 002403 000000 000002 177575 177577 177770  043440: 000600 000002 001400 000000 000000 000000 000000 000602
043450: 000002 000000 177755 000403 000002 000000 177533 002002  043460: 000001 000001 162000 043055 002003 000001 000001 122000

043470: 043057 000000 177735 177777 000000 000000 000000 000000  043500: 000000 000000 000000 000016 042513 000000 000000 000000
043510: 105000 000000 007466 011415 005012 011415 006005 114403  043520: 006014 002401 003002 000027 000021 000000 020000 000000
043530: 000000 000000 000000 000031 002002 000001 000001 122000  043540: 043520 002003 000001 000001 122000 043521 002003 000001
043550: 000001 122000 043522 001000 000000 002403 000000 000002  043560: 000013 000117 000011 000600 000001 001400 000001 170401
043570: 142000 043525 000000 000252 000000 177755 000000 000000  043600: 000000 000000 000000 000000 000000 000000 177744 002005
043610: 000001 000001 162000 043530 000000 000000 002003 000001  043620: 000001 122000 043521 001000 000000 000000 000000 000000
043630: 000000 000000 000000 000000 000000 177765 002002 000001  043640: 000001 162000 043520 002003 000001 000001 162000 043521
043650: 002003 000001 000001 122000 043522 001000 000000 002403  043660: 000000 000002 000013 000015 000011 000600 000002 001400

043670: 000001 000001 142000 043525 000602 000010 000000 177755  043700: 000600 000002 001401 000001 001001 102000 043525 000000
043710: 000032 001402 000001 000001 102000 043532 002003 000001  043720: 000001 162000 043522 000602 000006 002003 000001 000001
043730: 162000 043521 000000 000000 002003 000001 000001 162000  043740: 043522 000601 000002 001400 000001 000001 142000 043525
043750: 002003 000001 000001 162000 043522 000602 000004 002003  043760: 000001 000001 162000 043522 000602 000000 002002 000001
043770: 000001 162000 043531 002007 000001 000001 162000 043532  044000: 002007 000001 000001 122000 043532 000602 000032 002001
044010: 000001 000001 122000 043517 000602 000024 002001 000001  044020: 000001 122000 043517 002006 000001 000001 162000 043523
044030: 000000 177604 002006 000001 000001 122000 043523 002006  044040: 000001 000001 162000 043523 000000 177741 002003 000001
044050: 000001 162000 043522 000602 000003 002003 000001 000001  044060: 162000 043522 002003 000001 000001 162000 043521 000000

044070: 177564 000403 000001 001000 000000 002403 000000 000002  044100: 177575 177577 177770 000600 000002 001400 000000 000000
044110: 000000 000000 000602 000002 000000 177755 000403 000002  044120: 000000 177533 002002 000001 000001 162000 043520 002003
044130: 000001 000001 122000 043522 000000 177735 177777 000000  044140: 000000 000000 000000 000000 000000 000000 000017 043156
044150: 000000 000000 000000 105000 000000 007533 011415 005012  044160: 011415 006005 114403 006014 002401 003002 000027 000021
044170: 000000 020000 000000 000000 000000 000000 000031 002002  044200: 000001 000001 122000 044163 002003 000001 000001 122000
044210: 044164 002003 000001 000001 122000 044165 001000 000000  044220: 002403 000000 000002 000013 000117 000011 000600 000001
044230: 001400 000001 170401 142000 044170 000000 000252 000000  044240: 177755 000000 000000 000000 000000 000000 000000 000000
044250: 000000 177744 002005 000001 000001 162000 044173 000000  044260: 000000 002003 000001 000001 122000 044164 001000 000000
```

```
044270: 000000 000000 000000 000000 000000 000000 000000 000000  044300: 177765 002002 000001 000001 162000 044163 002003 000001
044310: 000001 162000 044164 002003 000001 000001 122000 044165  044320: 001000 000000 002403 000000 000002 000013 000015 000011
044330: 000600 000002 001400 000001 000001 142000 044170 000602  044340: 000010 000000 177755 000600 000002 001401 000001 001001
044350: 102000 044170 000000 000032 001402 000001 000001 102000  044360: 044175 002003 000001 000001 162000 044165 000602 000006
044370: 002003 000001 000001 162000 044164 000000 000000 002003  044400: 000001 000001 162000 044165 000601 000002 001400 000001
044410: 000001 142000 044170 002003 000001 000001 162000 044165  044420: 000602 000004 002003 000001 000001 162000 044165 000602
044430: 000000 002002 000001 000001 162000 044174 002007 000001  044440: 000001 162000 044175 002007 000001 000001 122000 044175
044450: 000602 000032 002001 000001 000001 122000 044162 000602  044460: 000024 002001 000001 000001 122000 044162 002006 000001

044470: 000001 162000 044166 000000 177604 002006 000001 000001  044500: 122000 044166 002006 000001 000001 162000 044166 000000
044510: 177741 002003 000001 000001 162000 044165 000602 000003  044520: 002003 000001 000001 162000 044165 002003 000001 000001
044530: 162000 044164 000000 177564 000403 000001 001000 000000  044540: 002403 000000 000002 177575 177577 177770 000800 000002
044550: 001400 000000 000000 000000 000000 000802 000002 000000  044560: 177755 000403 000002 000000 177533 002002 000001 000001
044570: 162000 044163 002003 000001 000001 122000 044165 000000  044600: 177735 177777 000000 100000 000000 000000 000000 000000
044610: 000000 000111 043651 043624 026622 000000 036003 140003  044620: 007600 007614 007630 007644 040400 000000 000000 000000
044630: 000000 000000 000000 000000 000000 000000 000000 000000  044640: 000000 000000 000000 000000 000000 000000 000000 000000
044650: 000000 002001 000001 000000 042000 045003 001000 000000  044660: 002401 000000 000000 000060 002001 000001 000000 042000

044670: 045004 001000 000000 002401 000000 000038 000045 001400  044700: 020000 004500 100005 010137 000000 000007 000000 177761
044710: 000000 000000 000002 000000 177754 002007 000001 000000  044720: 042000 045005 001402 000002 000000 002000 045014 001000
044730: 000000 002401 000000 000000 000007 001401 000003 000000  044740: 002000 045007 000601 000000 001401 000003 000000 002000
044750: 045007 000601 000001 001402 000002 000000 002000 000000  044760: 000000 177747 002007 000001 000000 042000 045006 001000
044770: 000000 002400 000001 000000 001401 000003 000000 002000  045000: 044624 000601 000000 000001 000018 000023 000025 000002
045010: 000000 000000 000000 000000 000120 002007 000001 000000  045020: 042000 045030 002401 000000 000000 000000 000601 000000
045030: 000063 000000 000000 000000 036025 000000 000000 000000  045040: 040000 000000 000000 100002 000000 000000 000000 000000
045050: 000131 044071 044061 037407 007680 060004 100001 007660  045060: 007724 000000 000000 000000 000000 000000 000000 000000

045070: 000000 000000 000030 002010 000002 000000 002000 045347  045100: 001000 000000 002010 000002 000000 002000 045357 001410
045110: 000004 000000 002000 045061 002402 000000 000000 000000  045120: 000213 000602 000001 002010 000002 000000 002000 045343
045130: 001000 000000 002010 000006 000000 002000 045344 001000  045140: 000000 002010 000002 000000 002000 045366 001410 000004
045150: 000000 002000 010721 001000 000000 002010 000002 000000  045160: 002000 045361 001000 000000 002010 000006 000000 002000
045170: 045350 001000 000000 000000 000014 002010 000004 000000  045200: 002000 045362 002010 000006 000000 002000 045354 001000
045210: 000000 002010 000002 000000 002000 045364 001400 000000  045220: 000000 000005 072333 001000 000000 002402 000000 000023
045230: 000025 000102 002010 000008 000000 002000 045344 001000  045240: 000000 002010 000004 000000 002000 045362 001000 000000
045250: 002402 000000 000000 000002 000057 000602 000002 002010  045260: 000002 000000 002000 045357 001410 000004 000000 002000

045270: 010702 002010 000002 000000 002000 045360 001410 000004  045300: 000000 002000 010704 001000 000000 000601 000001 002010
045310: 000002 000000 002000 045353 001410 000016 000000 002000  045320: 010706 001000 000000 000602 000003 002010 000002 000000
045330: 002000 045364 000602 000004 004400 000000 001000 000000  045340: 000603 000001 000000 013000 001200 000577 000460 012400
045350: 006000 000577 000460 006400 006000 000000 000000 001400  045360: 012000 007407 000000 000000 002400 000000 001400 177777
045370: 000000 000000 000000 000000 000000 000000 000000 000000  045400: 000132 044410 000000 000000 000000 030400 040000 007770
045410: 001000 000000 002403 000007 000000 000070 000070 000070  045420: 000000 000007 002010 000001 000000 042000 045544 001000
045430: 000000 000000 000000 002403 000000 000000 000047 000047  045440: 000000 002000 000104 000007 100008 036276 000000 000002
045450: 000000 177755 001000 000000 002403 000000 000000 000026  045460: 000026 000000 000000 000000 002010 000001 000000 044000

045470: 045545 001000 000000 002403 000000 000000 000007 000007  045500: 000000 001416 000001 000000 002000 045546 000601 000001
045510: 001416 000001 000000 002000 045546 000601 000002 002000  045520: 000002 000000 002000 045551 001000 000000 004401 000000
045530: 002001 000001 000000 042000 045547 002002 000001 000000  045540: 042000 045550 000801 000000 000000 000001 000000 000330
045550: 000202 015556 000000 000000 000000 000000 000000 000000  045560: 000000 000135 010128 000000 000000 000000 000000 000000
045570: 010003 000000 000000 000000
```

$$$$$$$$ DST   10 (SYSTEM BUFFERS) $$$$$$$$

```
146267(000044): 010285 000000 000000 000001 010001 000001 001777 002007 000000 000000 000001 001000  146267:........................
146303(000060): 037001 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  146303:>........................
146317(000074): 000001 001777 177777 177777 000403 075130 155555 133333 066666 155555 133333 066666  146317:..........zX.m..m..m..m.
146333(000110): 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666  146333:.m..m..m..m..m..m..m..m..
LINES 146347 - 150312 SAME AS ABOVE
150313(002070): 155555 133333 066666 155555 133333 066666 155555 133333 066666 000000 000000 000001  150313:.m..m..m..m..m..m.......
150327(002104): 001777 177777 177777 000403 075140 155555 133333 066666 155555 133333 066666 155555  150327:.........z .m..m..m..m..m
150343(002120): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  150343:..m..m..m..m..m..m..m..m
LINES 150357 - 152322 SAME AS ABOVE
152323(004100): 133333 066666 155555 133333 066666 155555 133333 066666 133333 066666 155555 133333  152323:..m..m..m..m..m...m..m..
152337(004114): 041404 022000 141203 041404 051025 000800 051021 030040 041000 026401 013705 000800  152337:C.$...C.R...R.O B.-.....
152353(004130): 051604 140124 177704 000700 043025 031101 141206 000200 021101 000800 031102 140420  152353:S..T....F.2A....^A..2B..
152367(004144): 030041 000600 053025 002400 051023 037777 000021 100000 000021 100000 000001 000000  152367:0!..V...R.7.............
152403(004160): 000000 034410 100000 030461 000000 000000 000001 000000 000000 000201 000000 000000  152403:...9...ii................
152417(004174): 000400 005706 000000 000000                                                          152417:........

$$$$$$$$ DST  201 $$$$$$$$
152423(000000): 000074 000201 140004 000000 000000 000014 000000 000000 000000 000003 155555 133333  152423:.<.....................m..
152437(000014): 100060 000004 047525 052040 020040 020040 002617 001501 000121 000051 000000 000060  152437:.0..OUT      ...A.Q.)...0
152453(000030): 000000 000000 000000 177735 000000 000561 000000 000000 000000 000561 000000 000561  152453:............q.......q...q
152467(000044): 177777 177777 000000 000000 000402 010001 010001 000000 000000 000000 000000 000014  152467:........................
152503(000060): 000061 001400 037024 000000 000000 000000 000000 000000 000000 000000 000000 000000  152503:.1..>....................
152517(000074): 066666 155555 133333 066666 000000 000000 000000 000000 000000 000000 000000 000000  152517:m..m..m..................
152533(000110): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  152533:........................
LINES 152547 - 152562 SAME AS ABOVE
152563(000140): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000001 100000  152563:........................
152577(000154): 000001 100000 000001 000000 110001 030610 100000 030461 000000 000000 000001 100000  152577:.........i...ii........
152613(000170): 000000 000153 000000 000000 000400 005212 000000 000000                              152613:...k.............

$$$$$$$$ DST  153 $$$$$$$$
152623(000000): 000000 014153 000010 000080 000073 000000 000000 000000 000007 040005 113081 000000  152623:...k...0.;.......@..1..
152637(000014): 000000 004531 002041 004102 051531 051440 020040 020040 051531 051517 050105 051040  152637:...Y.!.BSYS     SYSOPER
152653(000030): 051531 051517 050105 051040 047520 042522 040524 047522 000065 000067 000000 000000  152653:SYSOPER OPERATOR.5.7....
152667(000044): 000000 000000 002003 000800 000000 000000 000000 000000 020040 020040 020040 020040  152667:.....................
152703(000060): 000003 000001 000000 000220 000000 000000 000006 000000 000030 000000 000000 000001  152703:........................
152717(000074): 000000 066667 155556 133331 133331 133331 133331 133331 066666 155555 133333 066666  152717:..m..n..........m..m..m.
152733(000110): 155556 133331 066667 155556 066666 155555 133333 066666 155555 133333 066666 155555  152733:.n..m..nm..m..m..m..m..m
152747(000124): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  152747:..m..m..m..m..m..m..m..m
152763(000140): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 000001 100000  152763:..m..m..m..m..m..m......
152777(000154): 000001 100000 000043 000000 000000 033330 100000 024451 000000 000000 000043 000000  152777:.....#....6...))....#..
153013(000170): 000000 040101 000000 000000 000400 060315 000000 000000                              153013:..@A.....  .....

$$$$$$$$ CST 101   (FILESYS3               %010354    070316) $$$$$$$$    SEGMENT IS STANDARD.
 STT AREA:
163305: 106477 105477 106507 103402 108043 104107 145433 107032  163315: 126033 146033 146433 100504 101104 107477 111070 111470
163325: 103470 103070 111077 111477 131033 101107 107050 100507  163335: 107450 103403 105450 105077 110044 125474 123074 110077
163345: 124474 107036 107077 125074 100433 110477 126074 111444  163355: 101403 102003 050131 050060 050003 047754 007522 007104
```

163365: 005357 045112 002577 001314 001306 000777 000771 000524 163375: 000035 040071
**** (163377 TO 163622 NOT PRINTED) ****

$$$$$$$$ CST 301.CST.BLOCK INDEX =   12 $$$$$$$$
**** (163623 TO 165422 NOT PRINTED) ****

$$$$$$$$ AVAILABLE AREA $$$$$$$$
**** (165423 TO 166222 NOT PRINTED) ****

$$$$$$$$ CST   4   (FILESYS5   X004240   174506) $$$$$$$$ SEGMENT IS STANDARD.
 STT AREA:
172425: 107475 113075 106002 114475 106402 124474 105002 106007 172435: 102443 103043 105402 125074 123074 111475 104003 131033
172445: 103403 127033 107451 044034 043101 042474 041511 041254 172455: 040703 040453 040445 040267 040036 040035
**** (172463 TO 177022 NOT PRINTED) ****

$$$$$$$$ AVAILABLE AREA $$$$$$$$
**** (177023 TO   22 NOT PRINTED) ****

```
000000: 100000 000025 000000 110001 035470 100000 010421 000000  000010: 000000 000025 100000 000000 104441 000000 000000 000401
000020: 027235 000000 000000



$$$$$$$$ CST 341 CST BLOCK INDEX =     11 $$$$$$$$
**** (    23 TO   5222 NOT PRINTED) ****



$$$$$$$$ CST   7   (FILESYS7                %008220    147481) $$$$$$$$    SEGMENT IS STANDARD.
 STT AREA:
013338: 103004 127033 105003 114475 142433 125474 104104 105101  013348: 104002 105002 104003 106445 111032 101432 104004 104450
013358: 131033 105070 106070 105470 103070 111070 111470 107038  013368: 104043 104030 105051 111430 112430 107450 105450 106003
013378: 103030 101032 104403 106451 107032 125074 102404 100402  013408: 124474 102437 104402 100433 126074 100460 123074 105402
013418: 103403 103432 101403 102003 110044 103435 107501 111444  013428: 045722 045672 045531 045353 044760 000442 000452 000464
013438: 000477 000514 000433 000000 040104
**** ( 13443 TO  13622 NOT PRINTED) ****



$$$$$$$$ CST  52   (DEBUG                   %020550    125236) $$$$$$$$    SEGMENT IS STANDARD.
 STT AREA:
034270: 137474 137074 142433 136474 135074 122074 102041 114475  034300: 124474 113075 125074 101007 100406 110075 110475 100433
034310: 144474 144074 115075 120075 103047 100502 101477 131033  034320: 102447 123474 124074 123074 106434 125474 126074 107044
034330: 111444 110044 020354 060344 060326 060224 060143 060000  034340: 057506 057457 057451 057367 057303 017146 057111 057031
034350: 056712 056637 056521 056317 056156 055730 054674 053237  034360: 007012 007015 007007 041146 041104 041061 041020 040832
034370: 000315 000023 040102
**** ( 34373 TO  37422 NOT PRINTED) ****



$$$$$$$$ DST  266 $$$$$$$$
037423(000000): 004110 000266 140004 000000 000000 000012 000000 000000 000000 000001 104076 000005  037423:.H..................>..
037437(000014): 053522 045517 052524 030061 000000 000005 003776 001777 000000 000401 000042 000000  037437:WRKOUT01.............".."
037453(000030): 000000 001777 000000 000671 000000 000000 000000 001670 000000 001670 000000 000672  037453:........................
037467(000044): 010137 000000 000000 000001 010001 000001 001777 002007 000000 000000 000001 001000  037467:_.......................I...
037503(000060): 037001 000000 000000 000000 000000 000000 000000 000000 000000 000000 044640 000001  037503:>.....................I...
037517(000074): 000001 001777 000000 000671 000403 032020 155555 133333 066666 155555 133333 066666  037517:..........4..m..m..m..m.
037533(000110): 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666  037533:.m..m..m..m..m..m..m..m.
LINES 037547 - 041512 SAME AS ABOVE
041513(002070): 155555 133333 066666 155555 133333 066666 155555 133333 066666 043400 000001 000001  041513:.m..m..m..m..m..m.G.....
041527(002104): 001777 000000 000672 000403 032030 155555 133333 066666 155555 133333 066666 155555  041527:.......4..m..m..m..m..m
041543(002120): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  041543:..m..m..m..m..m..m..m..m
LINES 041557 - 043522 SAME AS ABOVE
043523(004100): 133333 066666 155555 133333 066666 155555 133333 066666 066666 155555 133333 066666  043523:..m..m..m..m..m.m..m..m.
043537(004114): 155556 133331 066667 155556 133331 066667 155556 133331 066667 155556 133331 066667  043537:.n..m..n..m..n..m..n..m.
LINES 043553 - 043566 SAME AS ABOVE
043567(004144): 155556 133331 066667 155556 133331 066667 000021 100000 000021 100000 000021 000000  043567:.n..m..n..m.............
043603(004160): 110001 032770 100000 027457 000000 000000 000021 100000 000000 000177 000000 000000  043603:..5.../////.............
043617(004174): 000400 010306 000000 000000                                                        043617:........
```

```
$$$$$$$ DST  177 $$$$$$$$
43623(000000): 004110 000177 140004 000000 000000 000012 000000 000000 000000 000001 104078 000008  043623:.H.....>..
43637(000014): 053522 045517 052524 030082 000000 000005 003778 001777 000000 000401 000042 000000  043637:WRKOUT02.....".."
43653(000030): 000000 001777 000000 000871 000000 000000 000000 001670 000000 001870 000000 000872  043653:.....
43667(000044): 012137 000000 000000 000001 010001 000001 001777 002007 000000 000000 000001 001000  043667:._.....
43703(000060): 037001 000000 000000 000000 000000 000000 000000 000000 000000 000000 045700 000001  043703:>.....K...
43717(000074): 000001 001777 000000 000871 000403 041020 155555 133333 066666 155555 133333 066666  043717:.....B..m..m..m..m.
43733(000110): 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666  043733:.m:.m..m..m..m..m..m..m.
INES 043747 - 045712 SAME AS ABOVE
45713(002070): 155555 133333 066666 155555 133333 066666 155555 133333 066666 044660 000001 000001  045713:.m..m..m..m..m..m.I.....
45727(002104): 001777 000000 000872 000403 041030 155555 133333 066666 155555 133333 066666 155555  045727:.....B..m..m..m..m..m
45743(002120): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  045743:..m..m..m..m..m..m..m..m
INES 045757 - 047722 SAME AS ABOVE
47723(004100): 133333 066666 155555 133333 066666 155555 133333 066666 133333 066666 155555 133333  047723:..m..m..m..m..m...m..m..
47737(004114): 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333  047737:m..m..m..m..m..m..m..m..
INES 047753 - 047766 SAME AS ABOVE
47767(004144): 066666 155555 133333 066666 155555 133333 000021 100000 000021 040000 000020 000000  047767:m..m..m..m.....@.....
50003(004160): 050001 035310 000000 030461 000000 000000 000020 100000 000000 000111 000000 000000  050003:P.:...II.....I....
50017(004174): 000400 005222 000000 000000 155555 133333 066666 155555 133333 066666 155555 133333  050017:.....m..m..m..m..m..
50033(004210): 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333  050033:m..m..m..m..m..m..m..m..
INES 050047 - 050126 SAME AS ABOVE
50127(004304): 066666 155555 133333 066666 066667 155556 133331 066667 066667 066667 066667 066667  050127:m..m..m.m..n..m.m.m.m.m.
50143(004320): 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667  050143:m.m.m.m.m.m.m.m.m.m.m.m.
INES 050157 - 050222 SAME AS ABOVE
50223(004400): 133331 066667 155556 133331 066667 155556 133331 066667 155556 133331 066667 155556  050223:..m..n..m..n..m..n..m..n
INES 050237 - 051022 SAME AS ABOVE
51023(005200): 004110 000266 140004 000000 000000 000012 000000 000000 000000 000001 104078 000011  051023:.H.....>..
51037(005214): 053522 045517 052524 030085 000000 000005 003778 001777 000000 000000 000042 000000  051037:WRKOUT05.....".."
51053(005230): 000000 174002 000000 000777 000000 000000 000000 002775 000000 002773 000000 000778  051053:.....
51067(005244): 020243 000000 000000 000001 010001 000401 001777 002007 000000 000000 000001 001000  051067: .....
51103(005260): 037001 000000 000000 000000 000000 000000 000000 000000 000000 000000 045520 000005  051103:>.....KP...
51117(005274): 000001 001777 000000 000778 000404 000770 155555 133333 066666 155555 133333 066666  051117:.....m..m..m..m.
51133(005310): 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666  051133:.m..m..m..m..m..m..m..m.
INES 051147 - 053112 SAME AS ABOVE
53113(007270): 155555 133333 066666 155555 133333 066666 155555 133333 066666 041000 000005 000001  053113:.m..m..m..m..m..m.B.....
53127(007304): 001777 000000 000775 000404 000780 155555 133333 066666 155555 133333 066666 155555  053127:.....m..m..m..m..m
53143(007320): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  053143:..m..m..m..m..m..m..m..m
INES 053157 - 053772 SAME AS ABOVE
53773(010150): 133333 066666 000020 040000 000020 020000 000002 066666 000000 000001 040023 000000  053773:..m...@... ...m.....A...
54007(010164): 000000 066666 000002 000000 066666 000000 133333 066666 155555 133333 066666 155555  054007:..m.....m.....m..m..m..m
```

```
$$$$$$$ AVAILABLE AREA $$$$$$$$
*** ( 54023 TO  54422 NOT PRINTED) ****
```

```
$$$$$$$ CST   5   (FILESYS0                X005154    054330) $$$$$$$$    SEGMENT IS STANDARD.
STT AREA:
61533: 112445 111445 107451 125474 103432 104004 106007 124474  061543: 105002 107036 104002 105050 104404 103404 103403 103032
61553: 105003 123074 105402 104032 103004 125074 102004 107032  061563: 126074 100433 101403 102003 044377 044320 044243 043677
```

```
174041(004752): 000000 001000 000061 000000 001000                                                    174041:.....1....

174046(MARKER): 177758 017537 103074 000011 KERNELC (75)     240 WAIT
***S REGISTER: ********************************************************************************************************
174052(004763): 000000 001000 000013 000000 001700 000000 000005 004163 177774 000060 000002 177775  174052:..............s...0....
174066(004777): 001775 141004 000044 177264 177274 000000 177304 000000 000005 000000 177777 177322  174066:.....$.................
174102(005013): 177307 000000 000003 003736 140404 000017 000020 000061 001360 003633 002446 056000  174102:.............1.....&\.
174116(005027): 037435 123317 001360 003633 000000 000000 000000 000000 000000 000000 000000 000000  174116:?.....................
174132(005043): 000001 177777 000001 000000 000000 000000 000000 000467 000400 100000 020000 000000  174132:.............7.... ....
174146(005057): 000000 000000 000000 000001 000000 000000 000000 000000 000000 000000 000000 000000  174146:.....................
174162(005073): 000000 000000 000000 000000 000012 000001 000004 003446 142032 000173 000024 140032  174162:.............&...{....
174176(005107): 000176 000000 000005 000061 001420 000000 000000 000000 000000 000000 000000 000000  174176:.~.....1.............
174212(005123): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  174212:.....................
LINES 174226 - 175215 SAME AS ABOVE
175216(006127): 000000                                                                                175216:..

175217: 057605 003243 004500 057605 021540 057605 000600 051607  175227: 177605 040052 002000 051606 021006 051404 021406 047610
175237: 013402 057610 040042 021426 057605 021002 021441 057605  175247: 021007 021434 057605 040032 071607 021431 057605 021010
175257: 021533 057605 041606 021432 057605 043611 026461 017717  175267: 040013 071006 101606 041006 031010 041404 040005 001700
175277: 141310 021004 142606 001000 001400 100000 000076 000056  175307: 041404 040052 001700 145344 021426 047605 021001 027204
175317: 057605 040042 021427 057605 040040 021430 057605 041404  175327: 040033 002100 021434 057605 040030 021435 057605 021431
175337: 047605 021436 057605 041606 003443 020320 041006 031010  175347: 020320 071006 040447 002000 021437 057605 021406 047610
175357: 013404 057610 140017 000400 002100 041404 021427 057605  175367: 040012 021430 057605 000600 021435 057605 000035 100000
175377: 000035 040000 000012 000001 030001 036270 000000 004411  175407: 000000 000000 000010 100000 030550 103001 000000 000000

175417: 000400 070354 000000 000000 000000 000000 000000 000000  175427: 000000 000000 000000 000000 000000 000000 000000 000000
175437: 000000 000000 000000 000000 000000 000000 000000 000000  175447: 053117 046040 052516 046101 041105 046114 042504 020040
175457: 020115 047525 047124 042504 020117 047040 046104 042526  175467: 021440 020040 000000 000000 000000 000000 000000 000000
175477: 000000 000000 000000 000000 000000 000000 000000 000000  175507: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 175517 - 175556 SAME AS ABOVE

175557: 000000 000000 000000 000000 000000 000000 000000 000000  175567: 002012 000000 000226 001604 000000 000000 000000 000000
175577: 000000 000000 003400 000000 000000 000000 000000 000000  175607: 000000 000000 000000 000000 004000 000000 000000 000000
175617: 000000 000000 000000 000000 000000 000000 004400 000000  175627: 000000 000000 000000 000000 000000 000000 000000 000000
175637: 005000 000000 000000 000000 000000 000000 000000 000000  175647: 000000 000000 000000 000000 000000 000000 000000 000000
175657: 000000 000000 000000 000000 000000 000000 000000 000000  175667: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 175677 - 175776 SAME AS ABOVE

175777: 000000 000000 000000 000000 000000 000000 000000 000000  176007: 000000 000000 000000 000000 000000 000000 000260 000000
176017: 000000 000000 000000 000000 000000 000000 000000 000000  176027: 000000 000000 000000 000000 000000 000000 000000 000000
176037: 000000 000000 000000 000000 000000 000000 000000 000000  176047: 000312 000000 000000 000000 000000 000000 000000 000000
176057: 000000 000000 000000 000000 000000 000000 000000 000000  176067: 000000 000000 000000 000000 000000 000000 000000 000000
176077: 000000 000000 000344 000000 000000 000000 000000 000000  176107: 000000 000000 000000 000000 000000 000000 000000 000000
176117: 000000 000000 000000 000000 000000 000000 000000 000000  176127: 000000 000000 000000 000000 000376 000000 000000 000000
176137: 000000 000000 000000 000000 000000 000000 000000 000000  176147: 000000 000000 000000 000000 000000 000000 000000 000000
176157: 000000 000000 000000 000000 000000 000000 000430 000000  176167: 000000 000000 000000 000000 000000 000000 000000 000000

176177: 000000 000000 000000 000000 000000 000000 000000 000000  176207: 000000 000000 000000 000000 000000 000000 000000 000000
176217: 000462 000000 000000 000000 000000 000000 000000 000000  176227: 000000 000000 000000 000000 000000 000000 000000 000000
176237: 000000 000000 000000 000000 000000 000000 000000 000000  176247: 000000 000000 000514 000000 000000 000000 000000 000000
176257: 000000 000000 000000 000000 000000 000000 000000 000000  176267: 000000 000000 000000 000000 000000 000000 000000 000000
176277: 000000 000000 000000 000000 000546 000000 000000 000000  176307: 000000 000000 000000 000000 000000 000000 000000 000000
176317: 000000 000000 000000 000000 000000 000000 000000 000000  176327: 000000 000000 000000 000000 000000 000000 000600 000000
176337: 000000 000000 000000 000000 000000 000000 000000 000000  176347: 000000 000000 000000 000000 000000 000000 000000 000000
176357: 000000 000000 000000 000000 000000 000000 000000 000000  176367: 000632 000000 000000 000000 000000 000000 000000 000000
```

```
176377: 000000 000000 000000 000000 000000 000000 000000 000000  176407: 000000 000000 000000 000000 000000 000000 000000 000000
176417: 000000 000000 000684 000000 000000 000000 000000 000000  176427: 000000 000000 000000 000000 000000 000000 000000 000000
176437: 000000 000000 000000 000000 000000 000000 000000 000000  176447: 000000 000000 000000 000000 000716 000000 000000 000000
176457: 000000 000000 000000 000000 000000 000000 000000 000000  176467: 000000 000000 000000 000000 000000 000000 000000 000000
176477: 000000 000000 000000 000000 000000 000000 000750 000000  176507: 000000 000000 000000 000000 000000 000000 000000 000000
176517: 000000 000000 000000 000000 000000 000000 000000 000000  176527: 000000 000000 000000 000000 000000 000000 000000 000000
176537: 001002 000000 000000 000000 000000 000000 000000 000000  176547: 000000 000000 000000 000000 000000 000000 000000 000000
176557: 000000 000000 000000 000000 000000 000000 000000 000000  176567: 000000 000000 001034 000000 000000 000000 000000 000000

176577: 000000 000000 000000 000000 000000 000000 000000 000000  176607: 000000 000000 000000 000000 000000 000000 000000 000000
176617: 000000 000000 000000 000000 001066 000000 000000 000000  176627: 000000 000000 000000 000000 000000 000000 000000 000000
176637: 000000 000000 000000 000000 000000 000000 000000 000000  176647: 000000 000000 000000 000000 000000 000000 001120 000000
176657: 000000 000000 000000 000000 000000 000000 000000 000000  176667: 000000 000000 000000 000000 000000 000000 000000 000000
176677: 000000 000000 000000 000000 000000 000000 000000 000000  176707: 001152 000000 000000 000000 000000 000000 000000 000000
176717: 000000 000000 000000 000000 000000 000000 000000 000000  176727: 000000 000000 000000 000000 000000 000000 000000 000000
176737: 000000 000000 001204 000000 000000 000000 000000 000000  176747: 000000 000000 000000 000000 000000 000000 000000 000000
176757: 000000 000000 000000 000000 000000 000000 000000 000000  176767: 000000 000000 000000 000000 001236 000000 000000 000000

176777: 000000 000000 000000 000000 000000 000000 000000 000000  177007: 000000 000000 000000 000000 000000 000000 000000 000000
177017: 000000 000000 000000 000000 000000 000000 001270 000000  177027: 000000 000000 000000 000000 000000 000000 000000 000000
177037: 000000 000000 000000 000000 000000 000000 000000 000000  177047: 000000 000000 000000 000000 000000 000000 000000 000000
177057: 001322 000000 000000 000000 000000 000000 000000 000000  177067: 000000 000000 000000 000000 000000 000000 000000 000000
177077: 000000 000000 000000 000000 000000 000000 000000 000000  177107: 000000 000000 001354 000000 000000 000000 000000 000000
177117: 000000 000000 000000 000000 000000 000000 000000 000000  177127: 000000 000000 000000 000000 000000 000000 000000 000000
177137: 000000 000000 000000 000000 001406 000000 000000 000000  177147: 000000 000000 000000 000000 000000 000000 000000 000000
177157: 000000 000000 000000 000000 000000 000000 000000 000000  177167: 000000 000000 000000 000000 000000 000000 001440 000000

177177: 000000 000000 000000 000000 000000 000000 000000 000000  177207: 000000 000000 000000 000000 000000 000000 000000 000000
177217: 000000 000000 000000 000000 000000 000000 000000 000000  177227: 001472 000000 000000 000000 000000 000000 000000 000000
177237: 000000 000000 000000 000000 000000 000000 000000 000000  177247: 000000 000000 000000 000000 000000 000000 000000 000000
177257: 000000 000000 001524 000000 000000 000000 000000 000000  177267: 000000 000000 000000 000000 000000 000000 000000 000000
177277: 000000 000000 000000 000000 000000 000000 000000 000000  177307: 000000 000000 000000 000000 001558 000000 000000 000000
177317: 000000 000000 000000 000000 000000 000000 000000 000000  177327: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 177337 - 177356 SAME AS ABOVE

177357: 000000 000000 000000 000000 000000 000000 000000 000000  177367: 000000 000000 000000 000000 104433 102441 000010 020000
177377: 000010 020000 000001 000000 000000 034410 100000 030461  177407: 000000 000000 000001 100010 000000 000000 000000 000000
177417: 000400 005708 000000 000000 000074 000201 140004 000000  177427: 000000 000014 000000 000000 000000 000003 155555 133333
177437: 100060 000004 047525 052040 020040 020040 002617 001501  177447: 000121 000051 000000 001412 000024 000000 000000 000000
177457: 000000 000463 000000 000000 000000 000463 000000 000463  177467: 177777 177777 000000 000000 000402 010001 010001 000000
177477: 000000 000000 000000 000014 000061 001400 037024 000000  177507: 000000 000000 000000 000000 000000 000000 000000 000000
177517: 066666 155555 133333 066666 000000 000000 000000 000000  177527: 000000 000000 000000 000000 000000 000000 000000 000000
177537: 000000 000000 000000 000000 000000 000000 000000 000000  177547: 000000 000000 000000 000000 000000 000000 000000 000000

177557: 000000 000000 000000 000000 000000 000000 000000 000000  177567: 000000 000000 000000 000000 000000 000000 000001 020000
177577: 000011 020000 000001 000000 000000 000001 152623 000000  177607: 000000 000000 000001 100011 000000 000000 000000 000000
177617: 000400 005212 000000 000000 000000 014153 000010 000060  177627: 000073 000000 000000 000000 000007 040005 113081 000000
177637: 000000 004531 002041 004102 051531 051440 020040 020040  177647: 051531 051517 050105 051040 051531 051517 050105 051040
177657: 047520 042522 040524 047522 000065 000067 000000 000000  177667: 000000 000000 002003 000600 000000 000000 000000 000000
177677: 020040 020040 020040 020040 000003 000001 000000 000220  177707: 000000 000000 000008 000000 000030 000000 000000 000001
177717: 000000 066667 155558 133331 133331 133331 133331 133331  177727: 030360 030360 030360 030360 030360 030360 030360 030360
177737: 030360 030360 030360 030360 030360 030360 030360 030360  177747: 030360 030360 030360 030360 030360 030360 030360 030360

177757: 030360 030360 030360 030360 030360 030360 030360 030360  177767: 030360 030360 030360 030360 030360 030360 000001 040000
177777: 000012 100000 000035 000000 110001 031450 100000 010421  000007: 000000 000000 000035 100000 000000 040163 000000 000000
000017: 000400 060123 000000 000000
```

```
LINES 167757 - 171722 SAME AS ABOVE
171723(004100): 133333 066666 155555 133333 066666 155555 133333 066666 133333 066666 155555 133333  171723:..m..m..m..m..m...m..m..
171737(004114): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  171737:..m..m..m..m..m..m..m..n
LINES 171753 - 171766 SAME AS ABOVE
171767(004144): 133333 066666 155555 133333 066666 155555 000021 100000 000021 040000 000030 000001  171767:..m..m..m..m.....@.....
172003(004160): 030201 033710 000000 023447 000007 000000 000014 100000 000000 000200 000000 000000  172003:0.7...  ;.....
172017(004174): 000400 013752 000000 000000 021010 020023 043010 026503 071010 003300 040013 000600  172017:.....".F.-Cr...@...
172033(004210): 163702 022402 051011 000800 031013 031008 141104 021002 140003 040000 021001 041801  172033:..x.R...2.2..D"...@.".C.
172047(004224): 003200 027142 051601 031400 151023 161106 000700 151023 150701 051703 000600 160702  172047:...bS.3....F.....S.....
172063(004240): 040020 035005 021425 045022 035001 040014 031045 051105 141612 000800 171050 035003  172063:@...#.J.:.@.2xRE.....(:.
172077(004254): 021030 031048 021065 140177 000767 001602 041121 041105 031043 041105 025001 031107  172077:".2&"5.....BQBE2#BE".2G
172113(004270): 041105 041003 041000 000700 031077 145512 021401 047003 051101 021001 021077 131101  172113:BEB.B...2?.J#.N.RA"."?.A
172127(004304): 012604 021045 140150 000144 040007 004300 020320 061101 141604 021046 140140 001106  172127:..."x.h.d@...  bA.."&.`.F
172143(004320): 041101 022471 010301 071101 004500 031023 145123 041007 101701 051110 041007 101701  172143:BAx9..rA.@2..SB...RHB...
172157(004334): 051007 041110 173003 041702 020023 031031 145110 000600 041105 173016 170011 010201  172157:R.BH..C. .2..H..BE.....
172173(004350): 140016 000106 000105 000104 000103 000102 000101 050122 047507 051101 046440 043111  172173:...F.E.D.C.B.APROGRAM FI
172207(004364): 046105 020040 021015 020042 171702 035021 034050 031051 004500 031012 145163 031013  172207:LE  "  .  ":.:.8(2).@2..s2.
172223(004400): 145164 031027 145165 031031 145166 021410 047110 021415 107110 010207 004500 031023  172223:.t2..u2..v#.NH#..H...@2.
172237(004414): 145130 041105 041010 041702 000600 047110 031077 141530 041010 051053 043053 022000  172237:.XBEB.C...NH2?.XB.R+F+$.
172253(004430): 141215 031011 043054 021003 027004 053054 041053 071058 051053 140413 000016 000000  172253:..2.F."...V,B+r.R+.....
172267(004444): 000013 041053 051011 031006 141107 021002 140008 041105 031035 140002 051051 021001  172267:..B+R.2..G"...BE2...R)".
172303(004460): 041601 003200 027142 051601 031400 043053 026504 051057 043053 026503 071053 003300  172303:C....bS.3.F+-DR/F+-Cr+..
172317(004474): 051054 043054 037777 071054 003300 051055 041055 101053 000600 041055 031022 002000  172317:R.F.?.r...R-B-.+..B-2...
172333(004510): 051056 031400 035051 043110 026552 000600 027162 022000 145202 140002 000060 171401  172333:R.3.:)FH-j...r$.....0..
172347(004524): 040036 053701 004573 021007 020023 041604 013734 171405 010201 171401 010201 173016  172347:@.W..("..C.............
172363(004540): 021056 020161 003300 020230 000133 020232 000700 021010 000700 171405 171401 171401  172363:". q.. ..[ ..  .  .  .  .
172377(004554): 171401 031111 000200 141604 021112 140023 020040 021427 045401 140003 021414 045022  172377:..2I....J..  #.K..#.J.
172413(004570): 004500 026621 051006 043110 026552 004556 006717 141203 021047 140003 021002 140003  172413:.@-.R.FH-j.n..  "  "  "
172427(004604): 051051 021001 041601 003200 027142 051601 031401 051514 027040 050125 041056 051531  172427:R)".C....bS.3.SL. PUB.SY
172443(004620): 051440 041401 010301 040020 053701 004573 021011 020023 041701 041402 041401 005600  172443:S C...@.W..("..C.C.C...
172457(004634): 021056 020161 003300 020230 000133 020232 032001 020040 035005 171700 010201 051401  172457:". q....[ .4.  :....S.
172473(004650): 035005 171700 010201 051402 035005 171700 051404 035050 173113 170446 010201 021003  172473:.....S.:...S.:(.K,&..".
172507(004664): 020042 170450 010201 021004 020042 170452 010201 021004 020043 173114 170463 010201  172507:". .(..". .". .". #.L.3..
172523(004700): 021003 020042 170465 010201 021004 020042 173115 170474 010201 021003 020042 041604  172523:". .  5.". ".M.<.". ."C.
172537(004714): 013733 173016 021056 020161 003300 020230 021001 020061 004543 020231 021040 162701  172537:....". q.. ." 1.c ." ..
172553(004730): 003500 020231 021040 162701 041016 170002 140515 173401 040006 020161 021056 162701  172553:.@ ." ..B....M..@.q"...
172567(004744): 004000 140042 020040 021040 021410 166401 021040 166402 173401 021432 171022 010201  172567:...".  ." #..".  #.....
172603(004760): 003700 021010 020063 173402 021442 171022 010201 003700 021010 020063 173401 020231  172603:..". 3..#".....". 3....
172617(004774): 021056 162701 003300 173402 020230 021001 020063 173402 020230 021001 020063 041022  172617:".....". 3.. ." 3B.
172633(005010): 026442 041401 170575 010201 021004 020243 141525 041402 170601 010201 021004 020243  172633:-"C..)..". ..UC.....
172647(005024): 141507 022000 141204 021024 140056 140001 140010 022002 141506 041022 021001 027042  172647:.G$..  ."....  $..FB."..
172663(005040): 051022 140001 140002 004000 021001 051130 000645 051132 051131 041022 026442 051403  172663:R.....".RX..RZRYB.-"S.
172677(005054): 141330 131403 045113 170002 140637 000700 021010 000700 041402 010101 041401 010101  172677:....JK.....".C..AC..A
172713(005070): 041404 041404 031111 000200 141506 021427 047404 026621 131403 055130 121403 140427  172713:C.C.2I...F#.O.-...ZX....
172727(005104): 021002 140003 051051 021001 041601 003200 027142 051601 031401 041143 141210 131112  172727:"...R)".C...bS.3.Bc...J
172743(005120): 045118 041004 021200 151141 031076 141505 000600 051143 021002 140004 045116 031035  172743:JNB."..a2>.E..Rc"...JN2.
172757(005134): 021001 041601 003200 027142 051601 031400 041604 021004 002300 010205 071004 051140  172757:"tO...bS.3.C."....r.R
172773(005150): 131112 075133 043700 004500 061142 141212 031014 141114 051142 045116 041004 021200  172773:.Jz[G..@bb..2..LRbJNB."..
173007(005164): 151141 031077 141503 021002 140004 045116 031035 021001 041601 003200 027142 051601  173007:.a2?.C"...JN2."..C....bS.
173023(005200): 031401 035002 041604 010201 043604 026514 043604 026504 003443 158403 010410 000400  173023:3.:.C...G.-LG.-D.#.....
173037(005214): 156403 002005 141502 026514 021137 006341 051401 000600 041604 041401 031020 013715  173037:.....B-L"_.S...C.C.2...
173053(005230): 151164 034005 001100 161164 120605 140107 000000 000001 140104 000103 000104 000078  173053:.t#..@.t...G.....D.C.D.>
173067(005244): 041401 022441 004300 047003 145247 131112 045116 041004 021200 151404 031077 145555  173067:C.x!..N...JJNB."...2?.n
173103(005260): 021402 177004 051136 041136 021401 047004 004300 177004 001700 145606 043136 026503  173103:#...R^B^@.N.........F^-C
173117(005274): 071136 003300 051137 000600 041604 041136 031037 013726 041112 022000 141521 151166  173117:r^..R_.C.B^2...BJ$..Q.v
173133(005310): 034452 001100 161166 041137 101136 000600 021401 177137 031022 002033 051402 171136  173133:9".@.vB_.^..#.._2...S..^
```

```
173147(005324): 041402 041401 031017 120605 140014 000800 021401 177137 031022 071137 003300 051136
173163(005340): 140455 043004 051405 140473 021002 140004 045116 031035 021001 041601 003200 027142
173177(005354): 051601 031401 035003 040083 004300 020320 040081 002043 020320 022000 145202 140003
173213(005370): 000051 000050 041605 022052 145643 041605 003300 051403 041604 023452 022404 051401
173227(005404): 041401 022452 003400 051402 040032 004300 020320 040030 002043 020320 041402 040023
173243(005420): 004300 020320 040021 002043 020320 041402 101403 041403 023052 020154 041401 171403
173257(005434): 021001 020132 043606 041605 020134 031403 001377 001072 035054 041112 022000 145502
173273(005450): 140004 000142 000141 000140 040130 004300 020320 040126 002043 020320 022000 145251
173307(005464): 041163 040117 004300 020320 040115 002043 020320 041604 023452 022404 021052 020174
173323(005500): 000600 051401 131401 047163 004500 051402 071401 022052 145475 041401 022052 145602
173337(005514): 140004 000076 000071 000052 000600 041605 041401 003343 177163 031037 017750 041401
173353(005530): 022000 145252 171403 131401 177163 041402 020023 041401 071402 003443 177163 041401
173367(005544): 003443 177163 041401 002400 020023 173163 171403 041402 020023 040027 004300 020320
173403(005560): 040025 002043 020320 041604 023452 022404 041163 021052 020134 041163 003300 051136
173417(005574): 043136 026503 071136 003300 051137 025001 051606 031402 001377 001072 140004 041401
173433(005810): 071402 051401 140510 031402 000600 053604 021401 177604 173604 021004 020023 043606
173447(005624): 026402 043605 026402 004617 141302 003200 004025 145222 021401 177604 047606 141217
173463(005640): 047605 141215 001700 141507 043606 037777 043605 037777 001700 141205 021001 053604
173477(005654): 031403 000132 041401 022001 145212 043606 026446 043605 026446 004617 141211 021002
173513(005670): 053604 031403 000115 000114 000113 000107 000105 041401 022002 145246 004025 145247
173527(005704): 000400 047606 145251 047605 145252 047606 026404 047605 026404 004617 141215 004500
173543(005720): 022004 141212 041701 022004 141207 021003 053604 041402 004400 023002 031043 047606
173557(005734): 026506 047605 026506 004617 141220 041701 022000 141507 004500 022001 141212 004500
173573(005750): 022002 141207 021003 053604 041402 004400 023002 031043 047606 037477 047605 037477
173607(005764): 004617 141215 004500 022013 141212 041701 022013 141207 021003 053604 041402 004400
173623(006000): 023002 031043 035406 004000 004000 012702 140502 031403 043604 026402 141503 021001
173637(006014): 140011 041401 022003 141505 043604 026446 022402 140002 021002 051605 031401 041007
173653(006030): 101011 041604 081401 145313 000600 041010 021200 041604 041701 071401 001700 141304
173667(006044): 022600 140406 000032 002100 031071 041700 041010 041011 101010 020023 101010 041052
173703(006060): 071701 051052 041053 071701 051053 041054 071701 051054 041055 071701 051055 041010
173717(006074): 071701 051010 071011 051011 041604 041007 101011 001700 141303 021107 140003 021002
173733(006110): 140003 051051 021001 041601 003200 027142 051601 031401 000700 041606 041605 041604
173747(006124): 031025 051607 031403 035006 021414 000600 041607 041605 006300 151607 000100 041605
173763(006140): 006300 022460 000500 164401 161607 151607 000710 141556 173604 171401 010201 003700
173777(006154): 021014 004421 004500 051810 020063 031404 021060 000300 166604 041606 022000 141212
174013(006170): 000600 041606 041605 006300 022460 166604 051606 000500 140413 031403 041022 026541
174027(006204): 013714 041013 041015 000600 031032 000600 031112 141203 021100 140005 025001 051017
174043(006220): 021002 140003 051051 021001 031033 041601 003200 027142 051601 031400 021507 177016
174057(006234): 152700 022040 141504 004574 025107 020262 101016 003324 051604 031400 041015 040006
174073(006250): 053701 004573 021043 020023 031400 020040 035002 025001 000600 031032 002433 051401
174107(006264): 000600 051402 171402 021001 021077 041401 022102 141307 031030 145502 140002 000050
174123(006300): 000600 051401 000600 041604 041402 031044 013725 041401 022000 141207 041403 013605
174137(006314): 021054 131401 166016 120401 000600 051403 000600 041402 003300 021012 131401 177016
174153(006330): 031024 071401 051401 052442 041401 022000 141203 031030 141507 031033 041601 021002
174167(006344): 027142 051601 031401 041601 021001 027142 051601 031401 041604 004545 000300 081116
174203(006360): 141210 021401 081117 141205 021402 081120 141202 021403 021074 003700 051051 041604
174217(006374): 171050 035003 021030 031046 031401 043604 022000 141222 000600 041121 043604 031044
174233(006410): 013710 041121 043604 031042 043604 000700 031113 140004 043604 000700 031050 000600
174247(006424): 053604 031401 041605 010201 041604 010201 003373 043605 026504 043604 026504 081701
174263(006440): 141504 020263 141502 120606 031402 041604 000600 053701 004573 021177 020023 031401
174277(006454): 041604 000600 053701 004573 021017 020023 031401 041604 026414 071605 043700 131604
174313(006470): 017300 053701 031402 041604 026414 071605 043700 131604 017400 053701 031402 041604
174327(006504): 026414 071605 043700 041604 037417 004300 014401 051606 031402 050060 032520 030060
174343(006520): 031103 046117 040504 020040 020040 020040 020040 020040 020040 000000 000000 121032
174357(006534): 013006 000000 000000 030376 030376 030376 030376 030376 030376 030376 030376 030376
174373(006550): 030376 030376 030376 030376 030376 030376 030376 030376 030376 030376 030376 030376
```

```
LINES 174407 - 174546 SAME AS ABOVE
174547(006724): 030376 030376 030376 030376 030376 030376 030376 030376 030376 030376 000200  174547:0.0.0.0.0.0.0.0.0.0.0...
174563(006740): 101407 101477 112030 102447 101501 113447 113047 116475 117475 112475 115475 107101  174563:...?...'.A.'.'.=.=.=.=.A
174577(006754): 100477 101077 110447 114447 102441 100433 110475 131074 102100 101101 124474 123074  174577:.?.?.'.'.I...=.<.@.A.<.<
174613(006770): 114047 115047 127074 107447 116075 117075 115075 125074 112447 113075 104501 101007  174613:.'.'.'.<.'.=.=.=.<.'.=.A..
174627(007004): 100406 105101 100405 046503 046473 046463 046454 046445 046428 008401 046354 046258  174627:...A..MCM;M3M.MxM...L.L.
174643(007020): 046246 046232 046202 046205 046202 046162 046127 046120 046027 046010 045814 045444  174643:L.L.L.L.L.LrLWLPL.L.K.K$
174657(007034): 045356 045201 045142 045115 044644 044512 044465 044230 044173 042742 042280 041701  174657:J.J.JbJMI.IJI5H.H(E.D.C.
174673(007050): 041365 041033 040247 040113 177777 177777 177777 177777 177777 177777 177777 177777  174673:B.B.@.@K..............
174707(007064): 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777  174707:..........................
LINES 174723 - 174752 SAME AS ABOVE
174753(007130): 177777 177777 177777 177777 177777 177777 177760 000000 002001 000001 000000 042000  174753:......................D.
174767(007144): 027244 001000 000000 002401 000000 000000 000014 020000 000035 020000 000014 000000  174767:.. ..... .. .....
175003(007160): 000000 030670 100000 030481 000000 000000 000014 100014 032750 000022 000000 000000  175003:...i...ii.....5.....
175017(007174): 000400 005122 000000 000000 000000 000311 002331 000031 000331 000631 002564 002554  175017:...R..............t.i
175033(007210): 002557 001131 001331 001531 001731 002131 177777 030360 177777 030360 030360 000000  175033:.o.Y...Y...Y..0...0.0...
175047(007224): 000000 000000 030360 030360 000000 177777 000001 000003 000003 000003 000003 000003  175047:....0.0...............
175063(007240): 000003 000003 000003 000003 000003 000003 000003 000003 000003 000003 000003 000003  175063:..........................
LINES 175077 - 175156 SAME AS ABOVE
175157(007334): 000003 000003 000003 000003 000003 000003 000001 000001 000001 000001 000001 000001  175157:..........................
175173(007350): 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001  175173:..........................
175207(007364): 000001 000001 000001 000003 000003 000003 000003 000001 000001 000001 000001 000001  175207:..........................
175223(007400): 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001 000001  175223:..........................
175237(007414): 000001 000001 000001 177777 177777 177777 177777 177777 177777 177777 177777 177777  175237:..........................
175253(007430): 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777 177777  175253:..........................
LINES 175267 - 175346 SAME AS ABOVE
175347(007524): 177777 177777 177777 177777 177777 177400 000330 000010 000410 001010 001410 002010  175347:..........................
175363(007540): 002410 003010 003410 004010 004410 005010 005410 006010 006410 007010 007410 010010  175363:..........................
175377(007554): 010410 011010 011410 012010 012410 013010 013410 014010 014430 015010 015410 016010  175377:..........................
175413(007570): 016410 017010 017410 020010 020410 021010 021410 022010 022410 023010 023410 024010  175413:..... .!.".#.$.%.&.'.(.
175427(007604): 024410 026010 026410 027010 027410 030010 030410 031010 031410 032040 032410 033010  175427:).*.+.-.../.0.1.2.3.4.5.6.
175443(007620): 033410 034010 034410 035410 036010 036430 037010 037430 040410 041010 041410 042010  175443:7.8.9.:.;.<.=.>.?.A.B.C.D.
175457(007634): 042410 043010 043410 044010 044410 045010 025010 025410 045410 046010 056410 057410  175457:E.F.G.H.I.J.^.+.K.L.].-.
175473(007650): 060010 060410 061010 057010 061410 062010 062410 063010 063410 064010 065010 066010  175473:`.a.b.^.c.d.e.f.g.h.i.I.
175507(007664): 066410 064410 065410 100000 101000 101400 102000 067010 067410 070010 070410 104040  175507:m.l.k.........n.o.p.q..
175523(007700): 104440 112410 000330 000330 000310 000310 000310 000310 000310 000310 000310 000310  175523:. .......................
175537(007714): 040000 103400 102400 177400 177400 177400 177400 177400 177400 177400 177400 177400  175537:@.......................
175553(007730): 177400 177400 177400 177400 177400 177400 177400 177400 177400 177400 177400 177400  175553:..........................
LINES 175567 - 175646 SAME AS ABOVE
175647(010024): 177400 177400 177400 177400 177400 030360 030360 000000 000000 000000 000000 000000  175647:..........0.0.....0.0.....
175663(010040): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000  175663:..........................
LINES 175677 - 176022 SAME AS ABOVE
176023(010200): 000000 000000 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360  176023:....0.0.0.0.0.0.0.0.0.0.
176037(010214): 000000 000000 000000 030360 030360 030360 030360 030360 030360 030360 030360 030360  176037:......0.0.0.0.0.0.0.0.0.
176053(010230): 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360  176053:0.0.0.0.0.0.0.0.0.0.0.0.
LINES 176067 - 176146 SAME AS ABOVE
176147(010324): 030360 030360 030360 030360 030360 000600 000002 004228 000001 000023 047040 000000  176147:0!0.0.0.0.............N ..
176163(010340): 177777 000043 000000 000000 177777 177777 000041 000001 177777 000000 000000 000000  176163:...#.........!.........
176177(010354): 000000 000000 000000 000000 000000 000000 000000 000000 000000 140702 001534 001464  176177:.....................\.4
176213(010370): 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000400 000600  176213:..........................
176227(010404): 001000 000171 000371 000000 000001 177222 177000 177000 177000 000002 001382 000576  176227:...y.....................
176243(010420): 001374 000000 000401 045427 000420 000400 111437 051113 047525 052062 002001 000767  176243:......K.......RKOUT2....
176257(010434): 000400 000200 000000 000440 000713 044120 031462 031063 030040 051525 050120 047522  176257:..... ..HP32230 SUPPOR
176273(010450): 052040 000000 000000 000000 177000 177000 177003 177006 000007 000005 177426 000101  176273:T .....................A
176307(010464): 177222 000004 000012 000001 000000 000001 000842 000362 177222 177224 177224 000002  176307:..........................
176323(010500): 000000 000000 177224 000002 000113 000000 000000 000003 000400 111437 177727 000000  176323:.........K................
```

```
176337(010514): 000001 001544 001560 001600 000001 000000 000000 000713 000400 053731 000000 000000  176337:...d.p.........W....
176353(010530): 000000 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360  176353:..0.0.0.0.0.0.0.0.0.0.
176367(010544): 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360  176367:0.0.0.0.0.0.0.0.0.0.0.
LINES 176403 - 177152 SAME AS ABOVE
177153(011330): 030360 044104 042506 040525 046124 051400 000251 000002 063523 054523 042125 046520  177153:0.HDEFAULTS....qSYSDUMP
177167(011344): 000247 000001 000000 013002 015403 012404 000000 004106 051105 040504 042111 051000  177167:.............FREADDIR.
177203(011360): 000003 003000 015001 016404 000000 002506 046117 041513 000003 002400 007402 015004  177203:...........FLOCK.....
177217(011374): 000000 002506 047520 042516 000005 002000 010001 010402 014004 024405 000000 003106  177217:...FOPEN.....|....F
177233(011410): 041510 042503 045400 000003 001400 033004 023005 000000 004106 043505 052111 047106  177233:CHECK.....0.&....FGETINF
177247(011424): 047400 000004 001000 005001 010002 011404 000000 000000 044111 051503 050125 000001  177247:O.............HISCPU..
177263(011440): 006401 000000 001527 044117 000001 004401 000000 002520 051111 047124 000005 002401  177263:.....WHO.....PRINT....
177277(011454): 017402 014403 035004 020005 000000 002121 052511 052000 000006 007000 010401 016002  177277:.....:....QUIT........
177313(011470): 016403 012004 026005 000000 003503 047515 046501 047104 000004 006400 006001 025002  177313:.....COMMAND.....^.
177327(011504): 027004 000000 003106 041514 047523 042400 000004 005000 012402 013004 025005 000000  177327:.....FCLOSE.....^....
177343(011520): 003506 052516 046117 041513 000001 004400 000000 003106 053522 000001 000400 053731  177343:.FUNLOCK.....FWR....W.
177357(011534): 000000 037777 177777 177777 177777 177777 177777 177600 016000 000000 000000 000000  177357:..?.....................
177373(011550): 000000 000000 000147 030360 030360 030360 030360 030360 030360 030360 030360 030360  177373:.....g0.0.0.0.0.0.0.0.
177407(011564): 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360 030360  177407:0.0.0.0.0.0.0.0.0.0.0.
LINES 177423 - 177502 SAME AS ABOVE
177503(011660): 030360 030360 030360 030360 030360 030360 030360 030360 000002 000401 017027         177503:0.0.0.0.0.0.0.0.0.....
177517(011674): 030360 000013 000000 000004 000000 000000 000000 000000 000000 000000 000000 000000  177517:0.......................
177533(011710): 000000 000000 000000 000000 000000 000000 000007 017027 000006 000050 000400 111437  177533:..................|....
177547(011724): 000006 000041 000400 111437 000006 000044 000400 111437 000006 000047 000400 111437  177547:....!.....$...../....
177563(011740): 000006 000056 000400 111437 000006 000030 000400 111437 000006 000057 000400 111437  177563:.....................?....
177577(011754): 000002 000400 111437 030360 000012 000010 000002 000000 037777 177777 177777 177777  177577:......0.........?.......
177613(011770): 177600 000003 140000 006000 000000 000000 000000 000000 000000 000003 111437 000006  177613:........................
177627(012004): 000020 000400 111437 000002 000401 025471 030360 000011 000001 000044 000000 037777  177627:..........+80.....$..?.
177643(012020): 177777 177777 177777 177600 000003 140000 010000 000000 000000 000000 000000 000000  177643:........................
177657(012034): 000003 025471 000006 000021 000401 025471 000052 000001 117045 102041 100451 100406  177657:..+9.....+9.^...%.I.)..
177673(012050): 001406 001304 001223 001155 001061 000757 000705 000643 000530 000231 000000 040024  177673:.......m.1.....X....@.
177707(012064): 000305 001300 002000 001000 000000 002006 000054 000000 000000 177670 000100 000002  177707:.....................@..
177723(012100): 022123 052104 046111 051524 000704 001301 002000 001000 000000 000000 000140 000000  177723:$$TDLIST.................
177737(012114): 000000 177701 000060 000003 053517 051113 047525 052062 002001 000767 000400 000200  177737:.....0..WORKOUT2.........
177753(012130): 000000 000000 004000 000000 000000 004226 000000 000023 000000 000000 000000 000023  177753:........................
177767(012144): 000000 000023 177777 177777 044111 000000 000014 040000 000030 100000 000266 000000  177767:.........HI....@.........
000003(012160): 110001 032010 100000 010020 000000 000000 000266 100000 000000 000216 000000 000000  000003:..4_....................
000017(012174): 000400 006576 000000 000000                                                          000017:....____
```

```
051137(047314): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  051137:..m..m..m..m..m..m..m..m
LINES 051153 - 053118 SAME AS ABOVE
053117(051274): 133333 066666 155555 133333 066666 042300 000005 000001 001777 000000 000775 000403  053117:..m..m..m.D.............
053133(051310): 046060 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333  053133:LO.m..m..m..m..m..m..m..
053147(051324): 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333  053147:m..m..m..m..m..m..m..m..
LINES 053163 - 055126 SAME AS ABOVE
055127(053304): 066666 155555 133333 066666 066667 155556 133331 066667 066667 066667 066667 066667  055127:m..m..m.m..n..m.m.m.m.m.
055143(053320): 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667 066667  055143:m.m.m.m.m.m.m.m.m.m.m.m.
LINES 055157 - 055222 SAME AS ABOVE
055223(053400): 133331 066667 155556 133331 066667 155556 133331 066667 155556 133331 066667 155556  055223:..m..n..m..n..m..n..m..n
LINES 055237 - 055302 SAME AS ABOVE
055303(053460): 133331 066667 155556 133331 066667 155556 133331 066667 155556 133331 066667 155556  055303:..m..n..m..n..m..n..m..n

055317: 000000 000000 000000 000000 000000 000000 000000 000000  055327: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 055337 - 055356 SAME AS ABOVE

055357: 000000 000000 000000 000000 000000 000000 000000 000000  055367: 000000 000000 000000 000000 000000 000000 000266 100000
055377: 000266 040000 000043 000001 030201 031730 000000 027020  055407: 000008 000000 000001 100000 000000 040165 000000 000000
055417: 000400 064443 000000 000000 004174 000030 000030 000030  055427: 000030 000030 000040 000004 153522 045517 052524 030061
055437: 144120 031462 031063 030323 052520 050117 051124 177777  055447: 177777 177777 100027 000051 001512 041527 000000 003503
055457: 044505 051122 047522 000000 000000 001776 177777 177777  055467: 014240 000000 000000 000001 000000 000000 000000 000000
055477: 020040 020040 020040 020040 000003 000001 000000 143566  055507: 000000 000000 000036 000000 000247 000000 000000 000001
055517: 000000 066667 155556 133331 066667 155556 133331 066667  055527: 000000 000000 000000 000000 000000 000000 000000 000000
055537: 000000 000000 000000 000000 000000 000000 000000 000000  055547: 000000 000000 000000 000000 000000 000000 000000 000000

055557: 000000 000000 000000 000000 000000 000000 000000 000000  055567: 000000 000000 000000 000000 000000 000000 000001 020000
055577: 000001 020000 000042 000000 000000 000000 000000 000000  055607: 000000 000000 000021 100001 034510 000121 000000 000000
055617: 000400 014326 000000 000000 004110 000121 140004 000000  055627: 000000 000012 000000 000000 000000 000001 104076 000011
055637: 053522 045517 052524 030065 000000 000005 003776 001777  055647: 000000 000401 000042 000000 000000 001777 000000 000011
055657: 000000 000000 000000 001010 000000 001010 000000 000012  055667: 020174 000000 000000 000001 010001 000001 001777 002007
055677: 000000 000000 000001 001000 037001 000000 000000 000000  055707: 000000 000000 000000 000000 000000 000000 043320 000001
055717: 000001 001777 000000 000011 000401 151527 155555 133333  055727: 066666 155555 133333 066666 155555 133333 066666 155555
055737: 133333 066666 155555 133333 066666 155555 133333 066666  055747: 155555 133333 066666 155555 133333 066666 155555 133333

055757: 066666 155555 133333 066666 155555 133333 066666 155555  055767: 133333 066666 155555 133333 066666 155555 133333 066666
055777: 155555 133333 066666 155555 133333 066666 155555 133333  056007: 066666 155555 133333 066666 155555 133333 066666 155555
056017: 133333 066666 155555 133333 066666 155555 133333 066666  056027: 155555 133333 066666 155555 133333 066666 155555 133333
056037: 066666 155555 133333 066666 155555 133333 066666 155555  056047: 133333 066666 155555 133333 066666 155555 133333 066666
056057: 155555 133333 066666 155555 133333 066666 155555 133333  056067: 066666 155555 133333 066666 155555 133333 066666 155555
056077: 133333 066666 155555 133333 066666 155555 133333 066666  056107: 155555 133333 066666 155555 133333 066666 155555 133333
056117: 066666 155555 133333 066666 155555 133333 066666 155555  056127: 133333 066666 155555 133333 066666 155555 133333 066666
056137: 155555 133333 066666 155555 133333 066666 155555 133333  056147: 066666 155555 133333 066666 155555 133333 066666 155555

056157: 133333 066666 155555 133333 066666 155555 133333 066666  056167: 155555 133333 066666 155555 133333 066666 155555 133333
056177: 066666 155555 133333 066666 155555 133333 066666 155555  056207: 133333 066666 155555 133333 066666 155555 133333 066666
056217: 155555 133333 066666 155555 133333 066666 155555 133333  056227: 066666 155555 133333 066666 155555 133333 066666 155555
056237: 133333 066666 155555 133333 066666 155555 133333 066666  056247: 155555 133333 066666 155555 133333 066666 155555 133333
056257: 066666 155555 133333 066666 155555 133333 066666 155555  056267: 133333 066666 155555 133333 066666 155555 133333 066666
056277: 155555 133333 066666 155555 133333 066666 155555 133333  056307: 066666 155555 133333 066666 155555 133333 066666 155555
056317: 133333 066666 155555 133333 066666 155555 133333 066666  056327: 155555 133333 066666 155555 133333 066666 155555 133333
056337: 066666 155555 133333 066666 155555 133333 066666 155555  056347: 133333 066666 155555 133333 066666 155555 133333 066666

056357: 155555 133333 066666 155555 133333 066666 155555 133333  056367: 066666 155555 133333 066666 155555 133333 066666 155555
056377: 133333 066666 155555 133333 066666 155555 133333 066666  056407: 155555 133333 066666 155555 133333 066666 155555 133333
056417: 066666 155555 133333 066666 155555 133333 066666 155555  056427: 133333 066666 155555 133333 066666 155555 133333 066666
056437: 155555 133333 066666 155555 133333 066666 155555 133333  056447: 066666 155555 133333 066666 155555 133333 066666 155555
```

```
056457: 133333 066666 155555 133333 066666 155555 133333 066666  056467: 155555 133333 066666 155555 133333 066666 155555 133333
056477: 066666 155555 133333 066666 155555 133333 066666 155555  056507: 133333 066666 155555 133333 066666 155555 133333 066666
056517: 155555 133333 066666 155555 133333 066666 155555 133333  056527: 066666 155555 133333 066666 155555 133333 066666 155555
056537: 133333 066666 155555 133333 066666 155555 133333 066666  056547: 155555 133333 066666 155555 133333 066666 155555 133333

056557: 066666 155555 133333 066666 155555 133333 066666 155555  056567: 133333 066666 155555 133333 066666 155555 133333 066666
056577: 155555 133333 066666 155555 133333 066666 155555 133333  056607: 066666 155555 133333 066666 155555 133333 066666 155555
056617: 133333 066666 155555 133333 066666 155555 133333 066666  056627: 155555 133333 066666 155555 133333 066666 155555 133333
056637: 066666 155555 133333 066666 155555 133333 066666 155555  056647: 133333 066666 155555 133333 066666 155555 133333 066666
056657: 155555 133333 066666 155555 133333 066666 155555 133333  056667: 066666 155555 133333 066666 155555 133333 066666 155555
056677: 133333 066666 155555 133333 066666 155555 133333 066666  056707: 155555 133333 066666 155555 133333 066666 155555 133333
056717: 066666 155555 133333 066666 155555 133333 066666 155555  056727: 133333 066666 155555 133333 066666 155555 133333 066666
056737: 155555 133333 066666 155555 133333 066666 155555 133333  056747: 066666 155555 133333 066666 155555 133333 066666 155555

056757: 133333 066666 155555 133333 066666 155555 133333 066666  056767: 155555 133333 066666 155555 133333 066666 155555 133333
056777: 066666 155555 133333 066666 155555 133333 066666 155555  057007: 133333 066666 155555 133333 066666 155555 133333 066666
057017: 155555 133333 066666 155555 133333 066666 155555 133333  057027: 066666 155555 133333 066666 155555 133333 066666 155555
057037: 133333 066666 155555 133333 066666 155555 133333 066666  057047: 155555 133333 066666 155555 133333 066666 155555 133333
057057: 066666 155555 133333 066666 155555 133333 066666 155555  057067: 133333 066666 155555 133333 066666 155555 133333 066666
057077: 155555 133333 066666 155555 133333 066666 155555 133333  057107: 066666 155555 133333 066666 155555 133333 066666 155555
057117: 133333 066666 155555 133333 066666 155555 133333 066666  057127: 155555 133333 066666 155555 133333 066666 155555 133333
057137: 066666 155555 133333 066666 155555 133333 066666 155555  057147: 133333 066666 155555 133333 066666 155555 133333 066666

057157: 155555 133333 066666 155555 133333 066666 155555 133333  057167: 066666 155555 133333 066666 155555 133333 066666 155555
057177: 133333 066666 155555 133333 066666 155555 133333 066666  057207: 155555 133333 066666 155555 133333 066666 155555 133333
057217: 066666 155555 133333 066666 155555 133333 066666 155555  057227: 133333 066666 155555 133333 066666 155555 133333 066666
057237: 155555 133333 066666 155555 133333 066666 155555 133333  057247: 066666 155555 133333 066666 155555 133333 066666 155555
057257: 133333 066666 155555 133333 066666 155555 133333 066666  057267: 155555 133333 066666 155555 133333 066666 155555 133333
057277: 066666 155555 133333 066666 155555 133333 066666 155555  057307: 133333 066666 155555 133333 066666 155555 133333 066666
057317: 155555 133333 066666 155555 133333 066666 155555 133333  057327: 066666 155555 133333 066666 155555 133333 066666 155555
057337: 133333 066666 155555 133333 066666 155555 133333 066666  057347: 155555 133333 066666 155555 133333 066666 155555 133333

057357: 066666 155555 133333 066666 155555 133333 066666 155555  057367: 133333 066666 155555 133333 066666 155555 133333 066666
057377: 155555 133333 066666 155555 133333 066666 155555 133333  057407: 066666 155555 133333 066666 155555 133333 066666 155555
057417: 133333 066666 155555 133333 066666 155555 133333 066666  057427: 155555 133333 066666 155555 133333 066666 155555 133333
057437: 066666 155555 133333 066666 155555 133333 066666 155555  057447: 133333 066666 155555 133333 066666 155555 133333 066666
057457: 155555 133333 066666 155555 133333 066666 155555 133333  057467: 066666 155555 133333 066666 155555 133333 066666 155555
057477: 133333 066666 155555 133333 066666 155555 133333 066666  057507: 155555 133333 066666 155555 133333 066666 155555 133333
057517: 066666 155555 133333 066666 155555 133333 066666 155555  057527: 133333 066666 155555 133333 066666 155555 133333 066666
057537: 155555 133333 066666 155555 133333 066666 155555 133333  057547: 066666 155555 133333 066666 155555 133333 066666 155555

057557: 133333 066666 155555 133333 066666 155555 133333 066666  057567: 155555 133333 066666 155555 133333 066666 155555 133333
057577: 066666 155555 133333 066666 155555 133333 066666 155555  057607: 133333 066666 155555 133333 066666 155555 133333 066666
057617: 155555 133333 066666 155555 133333 066666 155555 133333  057627: 066666 155555 133333 066666 155555 133333 066666 155555
057637: 133333 066666 155555 133333 066666 155555 133333 066666  057647: 155555 133333 066666 155555 133333 066666 155555 133333
057657: 066666 155555 133333 066666 155555 133333 066666 155555  057667: 133333 066666 155555 133333 066666 155555 133333 066666
057677: 155555 133333 066666 155555 133333 066666 155555 133333  057707: 066666 155555 133333 066666 155555 133333 066666 155555
057717: 133333 066666 155555 133333 066666 042220 000001 000001  057727: 001777 000000 000012 000401 151537 155555 133333 066666
057737: 155555 133333 066666 155555 133333 066666 155555 133333  057747: 066666 155555 133333 066666 155555 133333 066666 155555

057757: 133333 066666 155555 133333 066666 155555 133333 066666  057767: 155555 133333 066666 155555 133333 066666 155555 133333
057777: 066666 155555 133333 066666 155555 133333 066666 155555  060007: 133333 066666 155555 133333 066666 155555 133333 066666
060017: 155555 133333 066666 155555 133333 066666 155555 133333  060027: 066666 155555 133333 066666 155555 133333 066666 155555
060037: 133333 066666 155555 133333 066666 155555 133333 066666  060047: 155555 133333 066666 155555 133333 066666 155555 133333
060057: 066666 155555 133333 066666 155555 133333 066666 155555  060067: 133333 066666 155555 133333 066666 155555 133333 066666
060077: 155555 133333 066666 155555 133333 066666 155555 133333  060107: 066666 155555 133333 066666 155555 133333 066666 155555
```

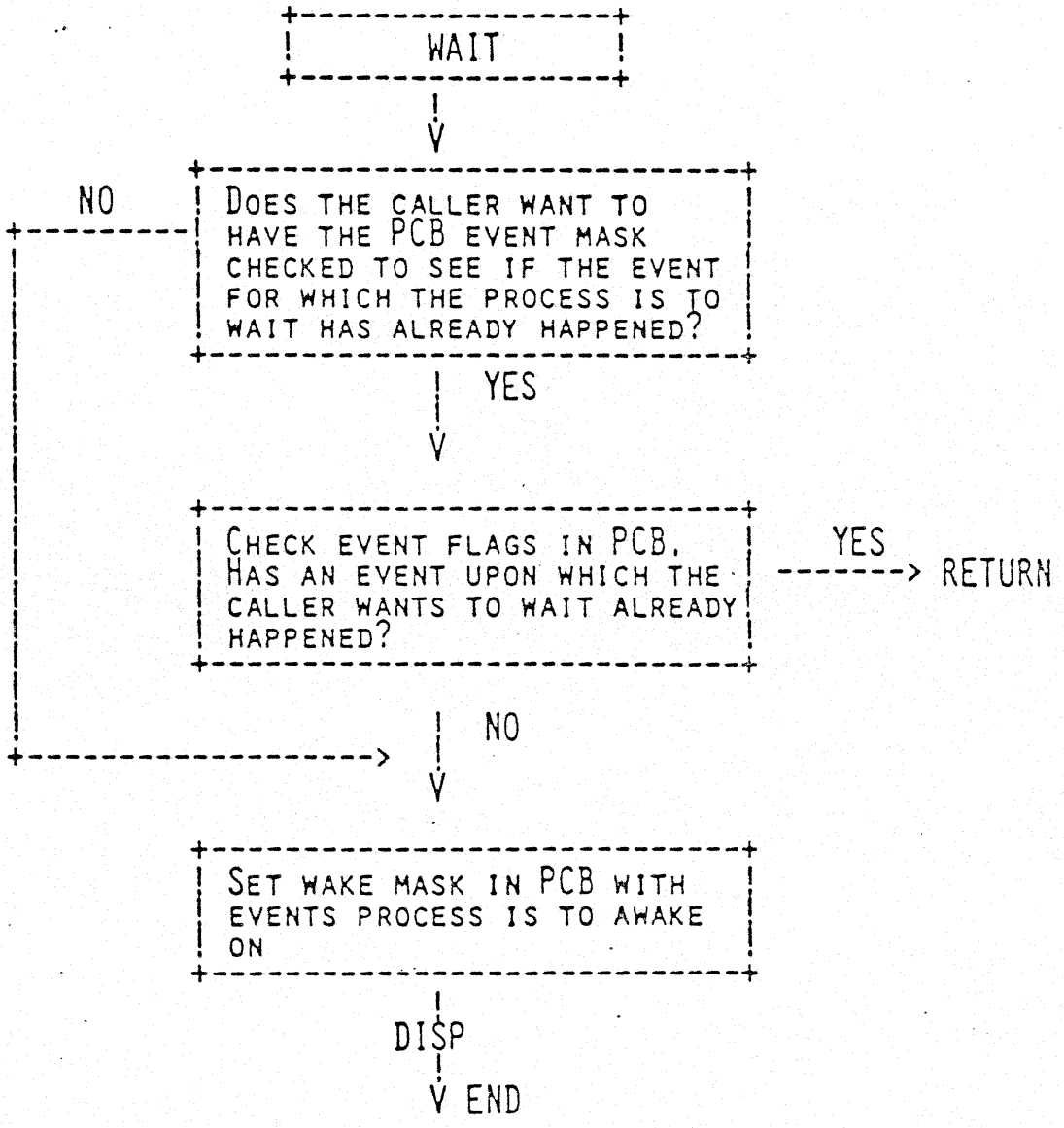
```
060117: 133333 066666 155555 133333 066666 155555 133333 066666  060127: 155555 133333 066666 155555 133333 066666 155555 133333
060137: 066666 155555 133333 066666 155555 133333 066666 155555  060147: 133333 066666 155555 133333 066666 155555 133333 066666

060157: 155555 133333 066666 155555 133333 066666 155555 133333  060167: 066666 155555 133333 066666 155555 133333 066666 155555
060177: 133333 066666 155555 133333 066666 155555 133333 066666  060207: 155555 133333 066666 155555 133333 066666 155555 133333
060217: 066666 155555 133333 066666 155555 133333 066666 155555  060227: 133333 066666 155555 133333 066666 155555 133333 066666
060237: 155555 133333 066666 155555 133333 066666 155555 133333  060247: 066666 155555 133333 066666 155555 133333 066666 155555
060257: 133333 066666 155555 133333 066666 155555 133333 066666  060267: 155555 133333 066666 155555 133333 066666 155555 133333
060277: 066666 155555 133333 066666 155555 133333 066666 155555  060307: 133333 066666 155555 133333 066666 155555 133333 066666
060317: 155555 133333 066666 155555 133333 066666 155555 133333  060327: 066666 155555 133333 066666 155555 133333 066666 155555
060337: 133333 066666 155555 133333 066666 155555 133333 066666  060347: 155555 133333 066666 155555 133333 066666 155555 133333

060357: 066666 155555 133333 066666 155555 133333 066666 155555  060367: 133333 066666 155555 133333 066666 155555 133333 066666
060377: 155555 133333 066666 155555 133333 066666 155555 133333  060407: 066666 155555 133333 066666 155555 133333 066666 155555
060417: 133333 066666 155555 133333 066666 155555 133333 066666  060427: 155555 133333 066666 155555 133333 066666 155555 133333
060437: 066666 155555 133333 066666 155555 133333 066666 155555  060447: 133333 066666 155555 133333 066666 155555 133333 066666
060457: 155555 133333 066666 155555 133333 066666 155555 133333  060467: 066666 155555 133333 066666 155555 133333 066666 155555
060477: 133333 066666 155555 133333 066666 155555 133333 066666  060507: 155555 133333 066666 155555 133333 066666 155555 133333
060517: 066666 155555 133333 066666 155555 133333 066666 155555  060527: 133333 066666 155555 133333 066666 155555 133333 066666
060537: 155555 133333 066666 155555 133333 066666 155555 133333  060547: 066666 155555 133333 066666 155555 133333 066666 155555

060557: 133333 066666 155555 133333 066666 155555 133333 066666  060567: 155555 133333 066666 155555 133333 066666 155555 133333
060577: 066666 155555 133333 066666 155555 133333 066666 155555  060607: 133333 066666 155555 133333 066666 155555 133333 066666
060617: 155555 133333 066666 155555 133333 066666 155555 133333  060627: 066666 155555 133333 066666 155555 133333 066666 155555
060637: 133333 066666 155555 133333 066666 155555 133333 066666  060647: 155555 133333 066666 155555 133333 066666 155555 133333
060657: 066666 155555 133333 066666 155555 133333 066666 155555  060667: 133333 066666 155555 133333 066666 155555 133333 066666
060677: 155555 133333 066666 155555 133333 066666 155555 133333  060707: 066666 155555 133333 066666 155555 133333 066666 155555
060717: 133333 066666 155555 133333 066666 155555 133333 066666  060727: 155555 133333 066666 155555 133333 066666 155555 133333
060737: 066666 155555 133333 066666 155555 133333 066666 155555  060747: 133333 066666 155555 133333 066666 155555 133333 066666

060757: 155555 133333 066666 155555 133333 066666 155555 133333  060767: 066666 155555 133333 066666 155555 133333 066666 155555
060777: 133333 066666 155555 133333 066666 155555 133333 066666  061007: 155555 133333 066666 155555 133333 066666 155555 133333
061017: 066666 155555 133333 066666 155555 133333 066666 155555  061027: 133333 066666 155555 133333 066666 155555 133333 066666
061037: 155555 133333 066666 155555 133333 066666 155555 133333  061047: 066666 155555 133333 066666 155555 133333 066666 155555
061057: 133333 066666 155555 133333 066666 155555 133333 066666  061067: 155555 133333 066666 155555 133333 066666 155555 133333
061077: 066666 155555 133333 066666 155555 133333 066666 155555  061107: 133333 066666 155555 133333 066666 155555 133333 066666
061117: 155555 133333 066666 155555 133333 066666 155555 133333  061127: 066666 155555 133333 066666 155555 133333 066666 155555
061137: 133333 066666 155555 133333 066666 155555 133333 066666  061147: 155555 133333 066666 155555 133333 066666 155555 133333

061157: 066666 155555 133333 066666 155555 133333 066666 155555  061167: 133333 066666 155555 133333 066666 155555 133333 066666
061177: 155555 133333 066666 155555 133333 066666 155555 133333  061207: 066666 155555 133333 066666 155555 133333 066666 155555
061217: 133333 066666 155555 133333 066666 155555 133333 066666  061227: 155555 133333 066666 155555 133333 066666 155555 133333
061237: 066666 155555 133333 066666 155555 133333 066666 155555  061247: 133333 066666 155555 133333 066666 155555 133333 066666
061257: 155555 133333 066666 155555 133333 066666 155555 133333  061267: 066666 155555 133333 066666 155555 133333 066666 155555
061277: 133333 066666 155555 133333 066666 155555 133333 066666  061307: 155555 133333 066666 155555 133333 066666 155555 133333
061317: 066666 155555 133333 066666 155555 133333 066666 155555  061327: 133333 066666 155555 133333 066666 155555 133333 066666
061337: 155555 133333 066666 155555 133333 066666 155555 133333  061347: 066666 155555 133333 066666 155555 133033 066666 155555

061357: 133333 066666 155555 133333 066666 155555 133333 066666  061367: 155555 133333 066666 155555 133333 066666 155555 133333
061377: 066666 155555 133333 066666 155555 133333 066666 155555  061407: 133333 066666 155555 133333 066666 155555 133333 066666
061417: 155555 133333 066666 155555 133333 066666 155555 133333  061427: 066666 155555 133333 066666 155555 133333 066666 155555
061437: 133333 066666 155555 133333 066666 155555 133333 066666  061447: 155555 133333 066666 155555 133333 066666 155555 133333
061457: 066666 155555 133333 066666 155555 133333 066666 155555  061467: 133333 066666 155555 133333 066666 155555 133333 066666
061477: 155555 133333 066666 155555 133333 066666 155555 133333  061507: 066666 155555 133333 066666 155555 133333 066666 155555
061517: 133333 066666 155555 133333 066666 155555 133333 066666  061527: 155555 133333 066666 155555 133333 066666 155555 133333
061537: 066666 155555 133333 066666 155555 133333 066666 155555  061547: 133333 066666 155555 133333 066666 155555 133333 066666
```

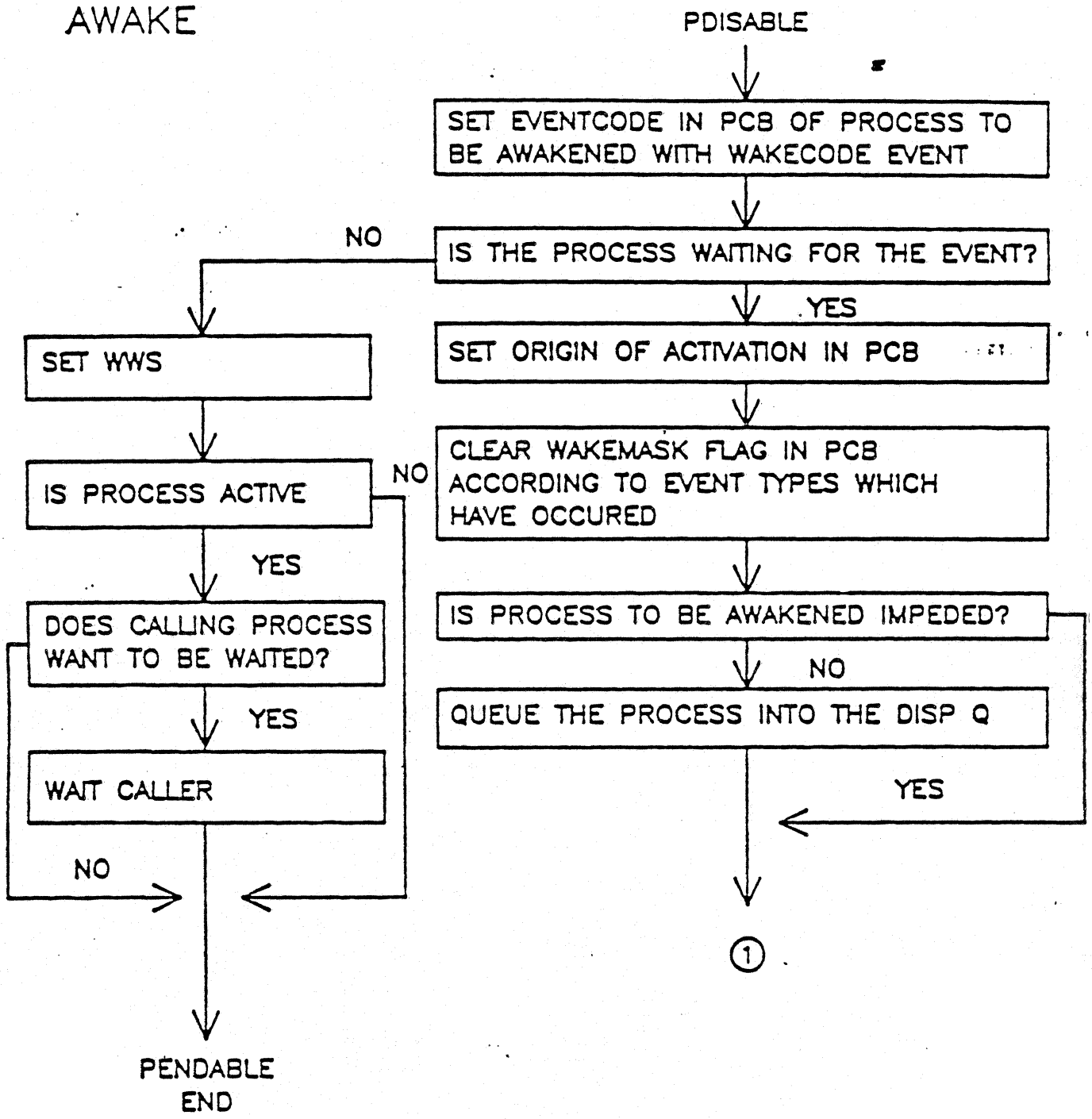
```
061557: 155555 133333 066666 155555 133333 066666 155555 133333  061567: 066666 155555 133333 066666 155555 133333 066666 155555
061577: 133333 066666 155555 133333 066666 155555 133333 066666  061607: 155555 133333 066666 155555 133333 066666 155555 133333
061617: 066666 155555 133333 066666 155555 133333 066666 155555  061627: 133333 066666 155555 133333 066666 155555 133333 066666
061637: 155555 133333 066666 155555 133333 066666 155555 133333  061647: 066666 155555 133333 066666 155555 133333 066666 155555
061657: 133333 066666 155555 133333 066666 155555 133333 066666  061667: 155555 133333 066666 155555 133333 066666 155555 133333
061677: 066666 155555 133333 066666 155555 133333 066666 155555  061707: 133333 066666 155555 133333 066666 155555 133333 066666
061717: 155555 133333 066666 155555 133333 066666 155555 133333  061727: 066666 155555 133333 066666 155558 133331 066667 155558
061737: 155555 133333 066666 155555 133333 066666 155555 133333  061747: 066666 155555 133333 066666 155555 133333 066666 155555

061757: 133333 066666 155555 133333 066666 155555 133333 066666  061767: 155555 133333 066666 155555 133333 066666 000021 020000
061777: 000022 020000 000047 000000 000000 000000 000000 000000  062007: 000000 000000 000021 100022 000000 000000 000000 000000
062017: 000400 010422 000000 000000 004110 000240 140004 000000  062027: 000000 000012 000000 000000 000000 000001 104076 000011
062037: 053522 045517 052524 030065 000000 000005 003776 001777  062047: 000000 000401 000042 000000 000000 001777 000000 000020
062057: 000000 000000 000000 001017 000000 001017 000000 000021  062067: 020137 000000 000000 000001 010001 000401 001777 002007
062077: 000000 000000 000001 001000 037001 000000 000000 000000  062107: 000000 000000 000000 000000 000000 000000 042020 000001
062117: 000001 001777 000000 000021 000401 102877 155555 133333  062127: 066666 155555 133333 066666 155555 133333 066666 155555
062137: 133333 066666 155555 133333 066666 155555 133333 066666  062147: 155555 133333 066666 155555 133333 066666 155555 133333

062157: 066666 155555 133333 066666 155555 133333 066666 155555  062167: 133333 066666 155555 133333 066666 155555 133333 066666
062177: 155555 133333 066666 155555 133333 066666 155555 133333  062207: 066666 155555 133333 066666 155555 133333 066666 155555
062217: 133333 066666 155555 133333 066666 155555 133333 066666  062227: 155555 133333 066666 155555 133333 066666 155555 133333
062237: 066666 155555 133333 066666 155555 133333 066666 155555  062247: 133333 066666 155555 133333 066666 155555 133333 066666
062257: 155555 133333 066666 155555 133333 066666 155555 133333  062267: 066666 155555 133333 066666 155555 133333 066666 155555
062277: 133333 066666 155555 133333 066666 155555 133333 066666  062307: 155555 133333 066666 155555 133333 066666 155555 133333
062317: 066666 155555 133333 066666 155555 133333 066666 155555  062327: 133333 066666 155555 133333 066666 155555 133333 066666
062337: 155555 133333 066666 155555 133333 066666 155555 133333  062347: 066666 155555 133333 066666 155555 133333 066666 155555

062357: 133333 066666 155555 133333 066666 155555 133333 066666  062367: 155555 133333 066666 155555 133333 066666 155555 133333
062377: 066666 155555 133333 066666 155555 133333 066666 155555  062407: 133333 066666 155555 133333 066666 155555 133333 066666
062417: 155555 133333 066666 155555 133333 066666 155555 133333  062427: 066666 155555 133333 066666 155555 133333 066666 155555
062437: 133333 066666 155555 133333 066666 155555 133333 066666  062447: 155555 133333 066666 155555 133333 066666 155555 133333
062457: 066666 155555 133333 066666 155555 133333 066666 155555  062467: 133333 066666 155555 133333 066666 155555 133333 066666
062477: 155555 133333 066666 155555 133333 066666 155555 133333  062507: 066666 155555 133333 066666 155555 133333 066666 155555
062517: 133333 066666 155555 133333 066666 155555 133333 066666  062527: 155555 133333 066666 155555 133333 066666 155555 133333
062537: 066666 155555 133333 066666 155555 133333 066666 155555  062547: 133333 066666 155555 133333 066666 155555 133333 066666

062557: 155555 133333 066666 155555 133333 066666 155555 133333  062567: 066666 155555 133333 066666 155555 133333 066666 155555
062577: 133333 066666 155555 133333 066666 155555 133333 066666  062607: 155555 133333 066666 155555 133333 066666 155555 133333
062617: 066666 155555 133333 066666 155555 133333 066666 155555  062627: 133333 066666 155555 133333 066666 155555 133333 066666
062637: 155555 133333 066666 155555 133333 066666 155555 133333  062647: 066666 155555 133333 066666 155555 133333 066666 155555
062657: 133333 066666 155555 133333 066666 155555 133333 066666  062667: 155555 133333 066666 155555 133333 066666 155555 133333
062677: 066666 155555 133333 066666 155555 133333 066666 155555  062707: 133333 066666 155555 133333 066666 155555 133333 066666
062717: 155555 133333 066666 155555 133333 066666 155555 133333  062727: 066666 155555 133333 066666 155555 133333 066666 155555
062737: 133333 066666 155555 133333 066666 155555 133333 066666  062747: 155555 133333 066666 155555 133333 066666 155555 133333

062757: 066666 155555 133333 066666 155555 133333 066666 155555  062767: 133333 066666 155555 133333 066666 155555 133333 066666
062777: 155555 133333 066666 155555 133333 066666 155555 133333  063007: 066666 155555 133333 066666 155555 133333 066666 155555
063017: 133333 066666 155555 133333 066666 155555 133333 066666  063027: 155555 133333 066666 155555 133333 066666 155555 133333
063037: 066666 155555 133333 066666 155555 133333 066666 155555  063047: 133333 066666 155555 133333 066666 155555 133333 066666
063057: 155555 133333 066666 155555 133333 066666 155555 133333  063067: 066666 155555 133333 066666 155555 133333 066666 155555
063077: 133333 066666 155555 133333 066666 155555 133333 066666  063107: 155555 133333 066666 155555 133333 066666 155555 133333
063117: 066666 155555 133333 066666 155555 133333 066666 155555  063127: 133333 066666 155555 133333 066666 155555 133333 066666
063137: 155555 133333 066666 155555 133333 066666 155555 133333  063147: 066666 155555 133333 066666 155555 133333 066666 155555

063157: 133333 066666 155555 133333 066666 155555 133333 066666  063167: 155555 133333 066666 155555 133333 066666 155555 133333
```

```
063177: 066666 155555 133333 066666 155555 133333 066666 155555  063207: 133333 066666 155555 133333 066666 155555 133333 066666
063217: 155555 133333 066666 155555 133333 066666 155555 133333  063227: 066666 155555 133333 066666 155555 133333 066666 155555
063237: 133333 066666 155555 133333 066666 155555 133333 066666  063247: 155555 133333 066666 155555 133333 066666 155555 133333
063257: 066666 155555 133333 066666 155555 133333 066666 155555  063267: 133333 066666 155555 133333 066666 155555 133333 066666
063277: 155555 133333 066666 155555 133333 066666 155555 133333  063307: 066666 155555 133333 066666 155555 133333 066666 155555
063317: 133333 066666 155555 133333 066666 155555 133333 066666  063327: 155555 133333 066666 155555 133333 066666 155555 133333
063337: 066666 155555 133333 066666 155555 133333 066666 155555  063347: 133333 066666 155555 133333 066666 155555 133333 066666

063357: 155555 133333 066666 155555 133333 066666 155555 133333  063367: 066666 155555 133333 066666 155555 133333 066666 155555
063377: 133333 066666 155555 133333 066666 155555 133333 066666  063407: 155555 133333 066666 155555 133333 066666 155555 133333
063417: 066666 155555 133333 066666 155555 133333 066666 155555  063427: 133333 066666 155555 133333 066666 155555 133333 066666
063437: 155555 133333 066666 155555 133333 066666 155555 133333  063447: 066666 155555 133333 066666 155555 133333 066666 155555
063457: 133333 066666 155555 133333 066666 155555 133333 066666  063467: 155555 133333 066666 155555 133333 066666 155555 133333
063477: 066666 155555 133333 066666 155555 133333 066666 155555  063507: 133333 066666 155555 133333 066666 155555 133333 066666
063517: 155555 133333 066666 155555 133333 066666 155555 133333  063527: 066666 155555 133333 066666 155555 133333 066666 155555
063537: 133333 066666 155555 133333 066666 155555 133333 066666  063547: 155555 133333 066666 155555 133333 066666 155555 133333

063557: 066666 155555 133333 066666 155555 133333 066666 155555  063567: 133333 066666 155555 133333 066666 155555 133333 066666
063577: 155555 133333 066666 155555 133333 066666 155555 133333  063607: 066666 155555 133333 066666 155555 133333 066666 155555
063617: 133333 066666 155555 133333 066666 155555 133333 066666  063627: 155555 133333 066666 155555 133333 066666 155555 133333
063637: 066666 155555 133333 066666 155555 133333 066666 155555  063647: 133333 066666 155555 133333 066666 155555 133333 066666
063657: 155555 133333 066666 155555 133333 066666 155555 133333  063667: 066666 155555 133333 066666 155555 133333 066666 155555
063677: 133333 066666 155555 133333 066666 155555 133333 066666  063707: 155555 133333 066666 155555 133333 066666 155555 133333
063717: 066666 155555 133333 066666 155555 133333 066666 155555  063727: 133333 066666 155555 133333 066666 155555 133333 066666
063737: 155555 133333 066666 155555 133333 066666 155555 133333  063747: 066666 155555 133333 066666 155555 133333 066666 155555

063757: 133333 066666 155555 133333 066666 155555 133333 066666  063767: 155555 133333 066666 155555 133333 066666 155555 133333
063777: 066666 155555 133333 066666 155555 133333 066666 155555  064007: 133333 066666 155555 133333 066666 155555 133333 066666
064017: 155555 133333 066666 155555 133333 066666 155555 133333  064027: 066666 155555 133333 066666 155555 133333 066666 155555
064037: 133333 066666 155555 133333 066666 155555 133333 066666  064047: 155555 133333 066666 155555 133333 066666 155555 133333
064057: 066666 155555 133333 066666 155555 133333 066666 155555  064067: 133333 066666 155555 133333 066666 155555 133333 066666
064077: 155555 133333 066666 155555 133333 066666 155555 133333  064107: 066666 155555 133333 066666 155555 133333 066666 155555
064117: 133333 066666 155555 133333 066666 046120 000001 000001  064127: 001777 000000 000020 000401 102667 155555 133333 066666
064137: 155555 133333 066666 155555 133333 066666 155555 133333  064147: 066666 155555 133333 066666 155555 133333 066666 155555

064157: 133333 066666 155555 133333 066666 155555 133333 066666  064167: 155555 133333 066666 155555 133333 066666 155555 133333
064177: 066666 155555 133333 066666 155555 133333 066666 155555  064207: 133333 066666 155555 133333 066666 155555 133333 066666
064217: 155555 133333 066666 155555 133333 066666 155555 133333  064227: 066666 155555 133333 066666 155555 133333 066666 155555
064237: 133333 066666 155555 133333 066666 155555 133333 066666  064247: 155555 133333 066666 155555 133333 066666 155555 133333
064257: 066666 155555 133333 066666 155555 133333 066666 155555  064267: 133333 066666 155555 133333 066666 155555 133333 066666
064277: 155555 133333 066666 155555 133333 066666 155555 133333  064307: 066666 155555 133333 066666 155555 133333 066666 155555
064317: 133333 066666 155555 133333 066666 155555 133333 066666  064327: 155555 133333 066666 155555 133333 066666 155555 133333
064337: 066666 155555 133333 066666 155555 133333 066666 155555  064347: 133333 066666 155555 133333 066666 155555 133333 066666

064357: 155555 133333 066666 155555 133333 066666 155555 133333  064367: 066666 155555 133333 066666 155555 133333 066666 155555
064377: 133333 066666 155555 133333 066666 155555 133333 066666  064407: 155555 133333 066666 155555 133333 066666 155555 133333
064417: 066666 155555 133333 066666 155555 133333 066666 155555  064427: 133333 066666 155555 133333 066666 155555 133333 066666
064437: 155555 133333 066666 155555 133333 066666 155555 133333  064447: 066666 155555 133333 066666 155555 133333 066666 155555
064457: 133333 066666 155555 133333 066666 155555 133333 066666  064467: 155555 133333 066666 155555 133333 066666 155555 133333
064477: 066666 155555 133333 066666 155555 133333 066666 155555  064507: 133333 066666 155555 133333 066666 155555 133333 066666
064517: 155555 133333 066666 155555 133333 066666 155555 133333  064527: 066666 155555 133333 066666 155555 133333 066666 155555
064537: 133333 066666 155555 133333 066666 155555 133333 066666  064547: 155555 133333 066666 155555 133333 066666 155555 133333

064557: 066666 155555 133333 066666 155555 133333 066666 155555  064567: 133333 066666 155555 133333 066666 155555 133333 066666
064577: 155555 133333 066666 155555 133333 066666 155555 133333  064607: 066666 155555 133333 066666 155555 133333 066666 155555
064617: 133333 066666 155555 133333 066666 155555 133333 066666  064627: 155555 133333 066666 155555 133333 066666 155555 133333
```

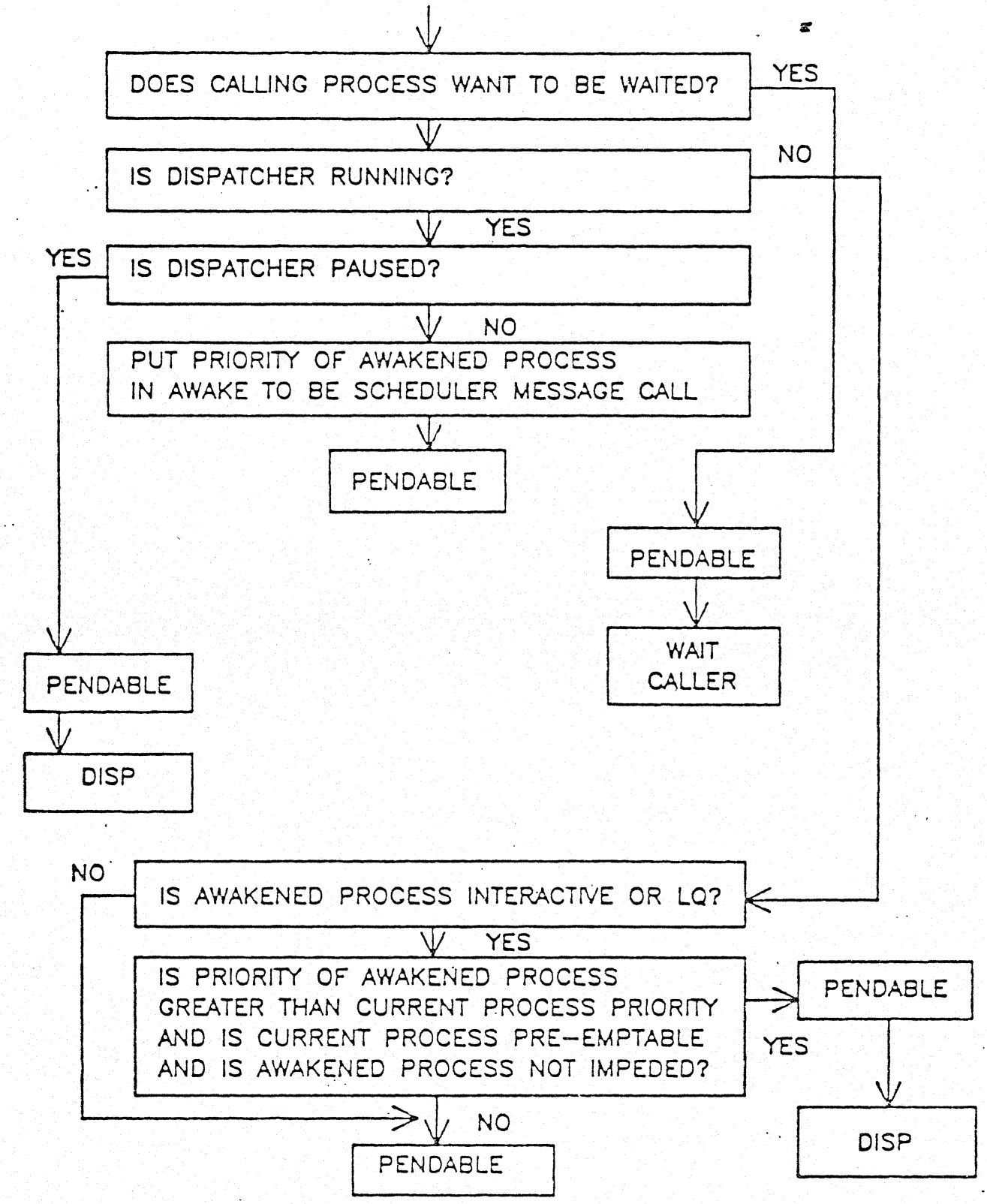
```
064637: 066666 155555 133333 066666 155555 133333 066666 155555  064647: 133333 066666 155555 133333 066666 155555 133333 066666
064657: 155555 133333 066666 155555 133333 066666 155555 133333  064667: 066666 155555 133333 066666 155555 133333 066666 155555
064677: 133333 066666 155555 133333 066666 155555 133333 066666  064707: 155555 133333 066666 155555 133333 066666 155555 133333
064717: 066666 155555 133333 066666 155555 133333 066666 155555  064727: 133333 066666 155555 133333 066666 155555 133333 066666
064737: 155555 133333 066666 155555 133333 066666 155555 133333  064747: 066666 155555 133333 066666 155555 133333 066666 155555

064757: 133333 066666 155555 133333 066666 155555 133333 066666  064767: 155555 133333 066666 155555 133333 066666 155555 133333
064777: 066666 155555 133333 066666 155555 133333 066666 155555  065007: 133333 066666 155555 133333 066666 155555 133333 066666
065017: 155555 133333 066666 155555 133333 066666 155555 133333  065027: 066666 155555 133333 066666 155555 133333 066666 155555
065037: 133333 066666 155555 133333 066666 155555 133333 066666  065047: 155555 133333 066666 155555 133333 066666 155555 133333
065057: 066666 155555 133333 066666 155555 133333 066666 155555  065067: 133333 066666 155555 133333 066666 155555 133333 066666
065077: 155555 133333 066666 155555 133333 066666 155555 133333  065107: 066666 155555 133333 066666 155555 133333 066666 155555
065117: 133333 066666 155555 133333 066666 155555 133333 066666  065127: 155555 133333 066666 155555 133333 066666 155555 133333
065137: 066666 155555 133333 066666 155555 133333 066666 155555  065147: 133333 066666 155555 133333 066666 155555 133333 066666

065157: 155555 133333 066666 155555 133333 066666 155555 133333  065167: 066666 155555 133333 066666 155555 133333 066666 155555
065177: 133333 066666 155555 133333 066666 155555 133333 066666  065207: 155555 133333 066666 155555 133333 066666 155555 133333
065217: 066666 155555 133333 066666 155555 133333 066666 155555  065227: 133333 066666 155555 133333 066666 155555 133333 066666
065237: 155555 133333 066666 155555 133333 066666 155555 133333  065247: 066666 155555 133333 066666 155555 133333 066666 155555
065257: 133333 066666 155555 133333 066666 155555 133333 066666  065267: 155555 133333 066666 155555 133333 066666 155555 133333
065277: 066666 155555 133333 066666 155555 133333 066666 155555  065307: 133333 066666 155555 133333 066666 155555 133333 066666
065317: 155555 133333 066666 155555 133333 066666 155555 133333  065327: 066666 155555 133333 066666 155555 133333 066666 155555
065337: 133333 066666 155555 133333 066666 155555 133333 066666  065347: 155555 133333 066666 155555 133333 066666 155555 133333

065357: 066666 155555 133333 066666 155555 133333 066666 155555  065367: 133333 066666 155555 133333 066666 155555 133333 066666
065377: 155555 133333 066666 155555 133333 066666 155555 133333  065407: 066666 155555 133333 066666 155555 133333 066666 155555
065417: 133333 066666 155555 133333 066666 155555 133333 066666  065427: 155555 133333 066666 155555 133333 066666 155555 133333
065437: 066666 155555 133333 066666 155555 133333 066666 155555  065447: 133333 066666 155555 133333 066666 155555 133333 066666
065457: 155555 133333 066666 155555 133333 066666 155555 133333  065467: 066666 155555 133333 066666 155555 133333 066666 155555
065477: 133333 066666 155555 133333 066666 155555 133333 066666  065507: 155555 133333 066666 155555 133333 066666 155555 133333
065517: 066666 155555 133333 066666 155555 133333 066666 155555  065527: 133333 066666 155555 133333 066666 155555 133333 066666
065537: 155555 133333 066666 155555 133333 066666 155555 133333  065547: 066666 155555 133333 066666 155555 133333 066666 155555

065557: 133333 066666 155555 133333 066666 155555 133333 066666  065567: 155555 133333 066666 155555 133333 066666 155555 133333
065577: 066666 155555 133333 066666 155555 133333 066666 155555  065607: 133333 066666 155555 133333 066666 155555 133333 066666
065617: 155555 133333 066666 155555 133333 066666 155555 133333  065627: 066666 155555 133333 066666 155555 133333 066666 155555
065637: 133333 066666 155555 133333 066666 155555 133333 066666  065647: 155555 133333 066666 155555 133333 066666 155555 133333
065657: 066666 155555 133333 066666 155555 133333 066666 155555  065667: 133333 066666 155555 133333 066666 155555 133333 066666
065677: 155555 133333 066666 155555 133333 066666 155555 133333  065707: 066666 155555 133333 066666 155555 133333 066666 155555
065717: 133333 066666 155555 133333 066666 155555 133333 066666  065727: 155555 133333 066666 155555 133333 066666 155555 133333
065737: 066666 155555 133333 066666 155555 133333 066666 155555  065747: 133333 066666 155555 133333 066666 155555 133333 066666

065757: 155555 133333 066666 155555 133333 066666 155555 133333  065767: 066666 155555 133333 066666 155555 133333 066666 155555
065777: 133333 066666 155555 133333 066666 155555 133333 066666  066007: 155555 133333 066666 155555 133333 066666 155555 133333
066017: 066666 155555 133333 066666 155555 133333 066666 155555  066027: 133333 066666 155555 133333 066666 155555 133333 066666
066037: 155555 133333 066666 155555 133333 066666 155555 133333  066047: 066666 155555 133333 066666 155555 133333 066666 155555
066057: 133333 066666 155555 133333 066666 155555 133333 066666  066067: 155555 133333 066666 155555 133333 066666 155555 133333
066077: 066666 155555 133333 066666 155555 133333 066666 155555  066107: 133333 066666 155555 133333 066666 155555 133333 066666
066117: 155555 133333 066666 155555 133333 066666 155555 133333  066127: 066666 155555 133333 066666 000201 000000 044120 031462
066137: 155555 133333 066666 155555 133333 066666 155555 133333  066147: 066666 155555 133333 066666 155555 133333 066666 155555

066157: 133333 066666 155555 133333 066666 155555 133333 066666  066167: 155555 133333 066666 155555 133333 066666 000021 040000
066177: 000043 100400 000021 000000 000000 000000 000000 000000  066207: 000000 000000 000021 000000 000000 000235 000000 000001
066217: 000400 010376 000000 000000
```

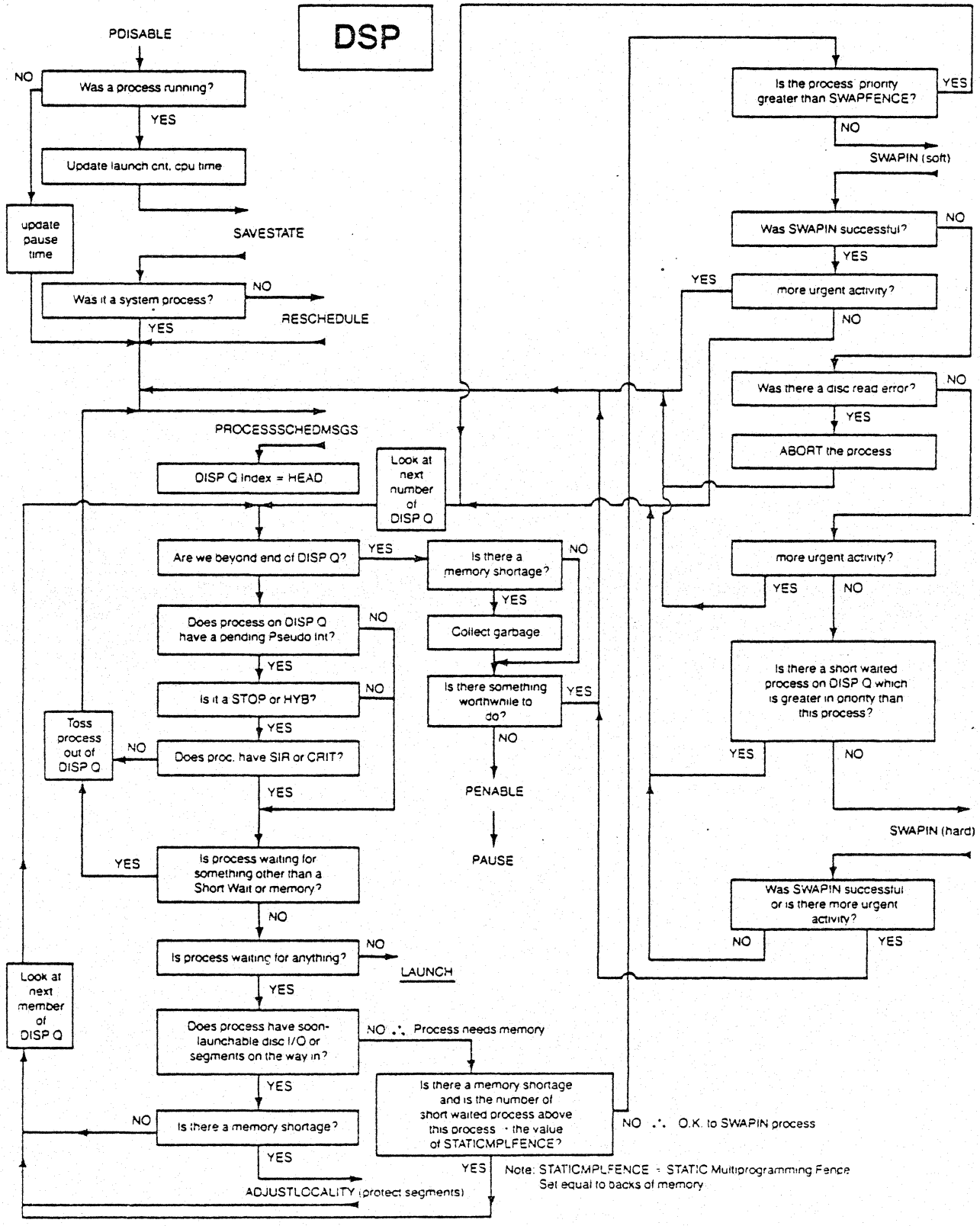
```
$$$$$$$$ DST  235 $$$$$$$$
066223(000000): 004110 000235 140004 000000 000000 000012 000000 000000 000000 000001 104076 000010  066223:.H.....>..
066237(000014): 053522 045517 052524 030064 000000 000005 003776 001777 000000 000401 000042 000000  066237:WRKOUT04.....".
066253(000030): 000000 001777 000000 000670 000000 000000 000000 001667 000000 001667 000000 000671  066253:.....
066267(000044): 016137 000000 000000 000001 010001 000401 001777 002007 000000 000000 000001 001000  066267:._.....
066303(000060): 037001 000000 000000 000000 000000 000000 000000 000000 000000 000000 041240 000001  066303:>.....B...
066317(000074): 000001 001777 000000 000671 000403 057070 155555 133333 086666 155555 133333 086666  066317:.....^8.m..m..m..m.
066333(000110): 155555 133333 086666 155555 133333 086666 155555 133333 086666 155555 133333 086666  066333:.m..m..m..m..m..m..m..m.
LINES 066347 - 070312 SAME AS ABOVE
070313(002070): 155555 133333 086666 155555 133333 086666 155555 133333 086666 041060 000001 000001  070313:.m..m..m..m..m..m.BO....
070327(002104): 001777 000000 000670 000403 057060 155555 133333 086666 155555 133333 086666 155555  070327:.....^0.m..m..m..m..m.
070343(002120): 133333 066666 155555 133333 066666 155555 133333 066666 155555 133333 066666 155555  070343:..m..m..m..m..m..m..m..m
LINES 070357 - 072336 SAME AS ABOVE
072337(004114): 020124 047440 041517 047124 044516 052505 021054 020043 041120 025054 021320 031013  072337: TO CONTINUE", #BP",".2.
072353(004130): 000600 041120 025001 031037 004000 025001 051604 140003 025001 051604 140003 025001  072353:..BP".2...".S...".S...".
072367(004144): 051604 031400 020040 020040 020040 020103 000021 100000 000021 020000 000005 000000  072367:S.3.       C.....  .....
072403(004160): 000000 000006 052423 000006 083223 000000 000005 000000 036150 000143 000000 000000  072403:....U...f.....<h.c....
072417(004174): 000400 015072 000000 000000                                                          072417:...:....


$$$$$$$$ AVAILABLE AREA $$$$$$$$


$$$$$$$$ DST  143 $$$$$$$$
072423(000000): 001104 000143 140004 000000 000000 000014 000000 000000 000000 000003 104076 000005  072423:.D.c.....>..
072437(000014): 101070 000004 022123 052104 044516 020040 000305 001300 002000 001000 000000 002024  072437:.8..$STDIN  .....,..
072453(000030): 000050 000000 000000 177772 000000 000012 000000 000000 000000 000012 000000 000001  072453:.(.....
072467(000044): 000000 000000 040111 000000 001403 000001 000001 000000 000077 001010 000000 000104  072467:....@I.....?.....D
072503(000060): 000001 001400 054401 100040 060400 002245 001300 000542 000000 000000 000000 000000  072503:....Y.. a.....b.....
072517(000074): 000000 000000 000001 001000 000000 000000 000403 136154 000025 000110 000001 000000  072517:.....l...H....
072533(000110): 000000 035122 052516 020127 047522 045517 052524 031040 000011 000110 000001 000000  072533:...:RUN WORKOUT2  ...H....
072547(000124): 000000 032440 000010 000110 000001 000000 000000 000010 000110 000001 000000 000000  072547:..5  ...H.....H.....
072563(000140): 000010 000110 000001 000000 000000 000010 000110 000001 000000 000000 000010 000110  072563:...H.....H.....H
072577(000154): 000001 000000 000000 000013 000110 000001 000000 000000 054505 051440 000012 000110  072577:.....H.....YES  ...H
072613(000170): 000001 000000 000000 047117 000013 000110 000001 000000 000000 031060 030000 000014  072613:.....NO...H.....200...
072627(000204): 000110 000001 000000 000000 035105 047512 177777 002105 000003 007137 007140 007141  072627:.H.....:EOJ...E.....`.a
072643(000220): 007142 007143 007144 007150 007157 007171 015152 014623 014551 014117 012360 012124  072643:b.c.d.h.o.y.j...i.0...T
072657(000234): 012071 000001 012032 001420 000200 000000 000001 000720 140153 000121 021770 004000  072657:.9.............k.Q$...
072673(000250): 000000 001000 001120 004310 004300 001120 004310 032351 100033 000011 000040 000001  072673:.....P.....P..4.....  ..
072707(000264): 000001 177600 024742 177777 000163 000000 177777 000007 032041 142033 000016 000000  072707:.....)....s.....&!......
072723(000300): 000020 024070 024071 000652 000642 024070 000004 032245 102033 000011 000013 000001  072723:.>(8(9.....(8..4.....
072737(000314): 140153 000121 021770 004000 000000 001000 007130 000000 000145 000000 023003 140033  072737:.k.Q$.....X...e..&...
072753(000330): 000015 000024 000010 000014 000001 177735 000060 000004 000001 000043 000001 140077  072753:.............0.....$...?
072767(000344): 000200 000061 177777 000000 000400 000003 000310 000400 100000 020000 000000 000000  072767:....1.....
073003(000360): 000163 002214 000006 037567 000024 000000 177777 000000 000000 036656 000012 000000  073003:s....?w.....=.....
073017(000374): 000003 000007 000010 007136 007137 007140 007141 007142 007143 007144 007150 007157  073017:.....^.`.a.b.c.d.h.o
073033(000410): 007171 015152 014623 014551 014117 012360 012124 012071 000001 012032 001420 000200  073033:.y.j...i.0...T.9.....
073047(000424): 000000 000001 000720 140153 000121 021770 004000 000000 001000 001120 004750 004740  073047:.....k.Q$.....P....
073063(000440): 001120 004750 032351 100033 000011 000040 000001 037567 000024 000000 177777 000000  073063:.P..4.....  ..?w......
073077(000454): 000000 002542 000012 000000 000000 000007 000010 000000 007137 007140 007141 007142  073077:...b.....:.`.a.b
```


WAIT
DOES THE CALLER WANT TO HAVE THE PCB EVENT MASK CHECKED TO SEE IF THE EVENT FOR WHICH THE PROCESS IS TO WAIT HAS ALREADY HAPPENED?
NO
YES
CHECK EVENT FLAGS IN PCB. HAS AN EVENT UPON WHICH THE CALLER WANTS TO WAIT ALREADY HAPPENED?
YES
RETURN
NO
SET WAKE MASK IN PCB WITH EVENTS PROCESS IS TO AWAKE ON
DISP
END


AWAKE
PDISABLE
SET EVENTCODE IN PCB OF PROCESS TO BE AWAKENED WITH WAKECODE EVENT
IS THE PROCESS WAITING FOR THE EVENT?
NO
YES
SET WWS
SET ORIGIN OF ACTIVATION IN PCB
CLEAR WAKEMASK FLAG IN PCB ACCORDING TO EVENT TYPES WHICH HAVE OCCURED
IS PROCESS ACTIVE
NO
YES
DOES CALLING PROCESS WANT TO BE WAITED?
YES
WAIT CALLER
NO
IS PROCESS TO BE AWAKENED IMPEDED?
NO
QUEUE THE PROCESS INTO THE DISP Q
YES
1
PENDABLE END

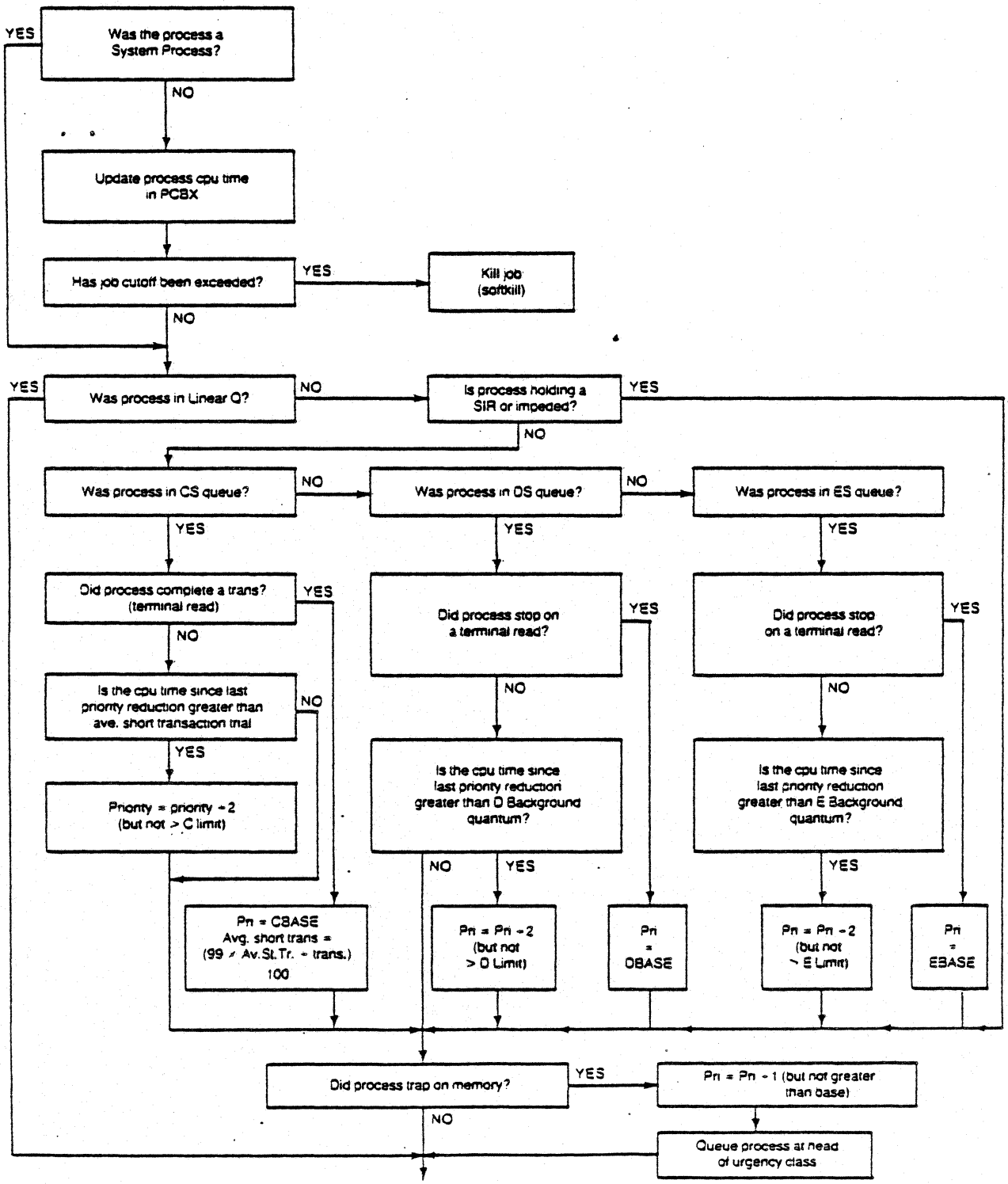
AWAKE (CONTINUED) ①



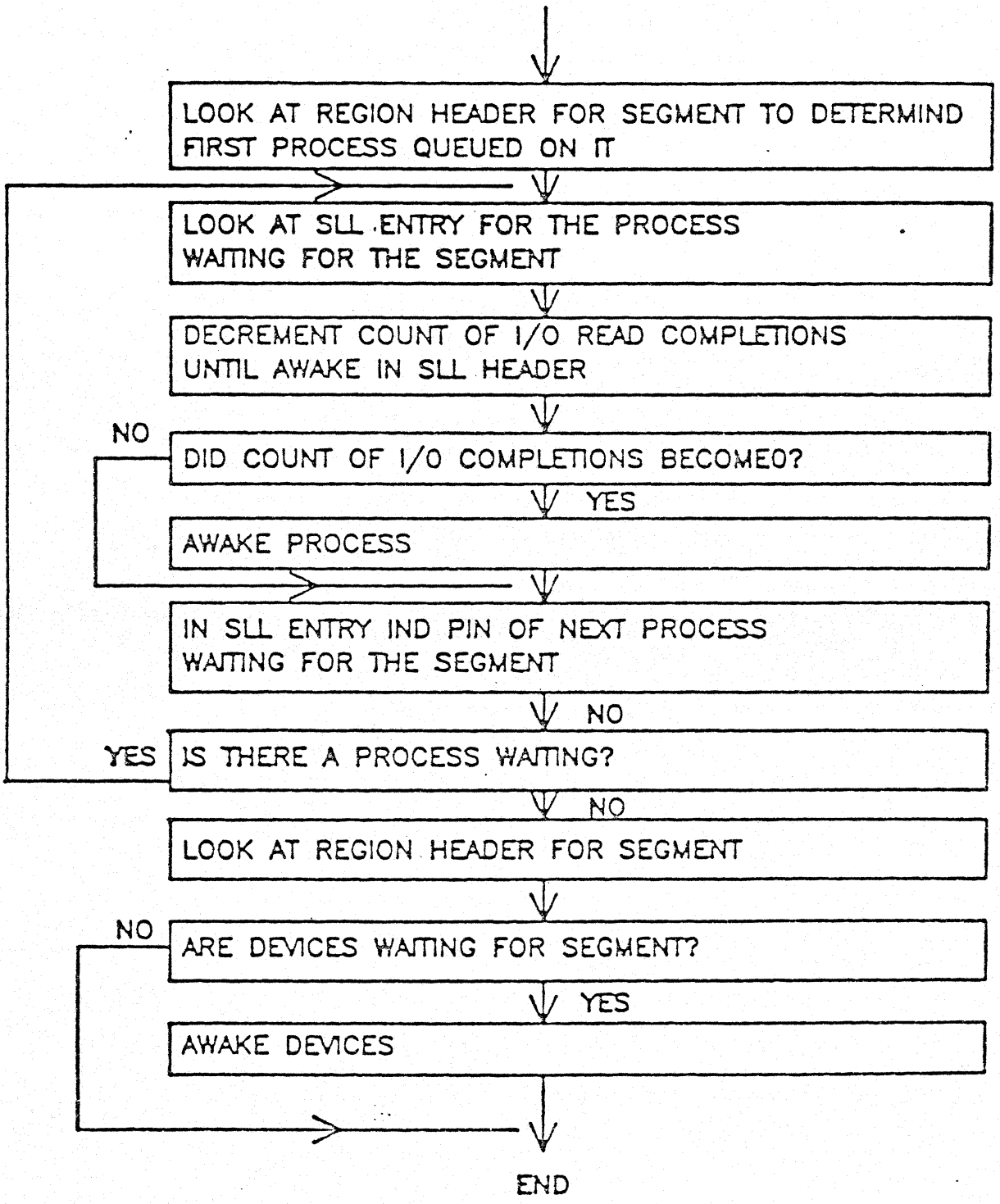

DSP
PDISABLE
Was a process running?
NO
YES
Update launch cnt, cpu time
update pause time
SAVESTATE
Was it a system process?
NO
RESCHEDULE
YES
PROCESSSCHEDMSGS
DISP Q index = HEAD
Look at next number of DISP Q
Are we beyond end of DISP Q?
YES
Is there a memory shortage?
NO
YES
Collect garbage
Does process on DISP Q have a pending Pseudo Int?
NO
YES
Is it a STOP or HYB?
NO
YES
Is there something worthwhile to do?
YES
NO
Toss process out of DISP Q
NO
Does proc. have SIR or CRIT?
YES
PENABLE
PAUSE
YES
Is process waiting for something other than a Short Wait or memory?
NO
Is process waiting for anything?
NO
LAUNCH
YES
Look at next member of DISP Q
Does process have soon-launchable disc I/O or segments on the way in?
NO ∴ Process needs memory
YES
NO
Is there a memory shortage?
YES
Is there a memory shortage and is the number of short waited process above this process ≥ the value of STATICMPLFENCE?
NO ∴ O.K. to SWAPIN process
YES
ADJUSTLOCALITY (protect segments)
Note: STATICMPLFENCE = STATIC Multiprogramming Fence
Set equal to backs of memory
Is the process' priority greater than SWAPFENCE?
YES
NO
SWAPIN (soft)
Was SWAPIN successful?
NO
YES
more urgent activity?
YES
NO
Was there a disc read error?
NO
YES
ABORT the process
more urgent activity?
YES
NO
Is there a short waited process on DISP Q which is greater in priority than this process?
YES
NO
SWAPIN (hard)
Was SWAPIN successful or is there more urgent activity?
NO
YES

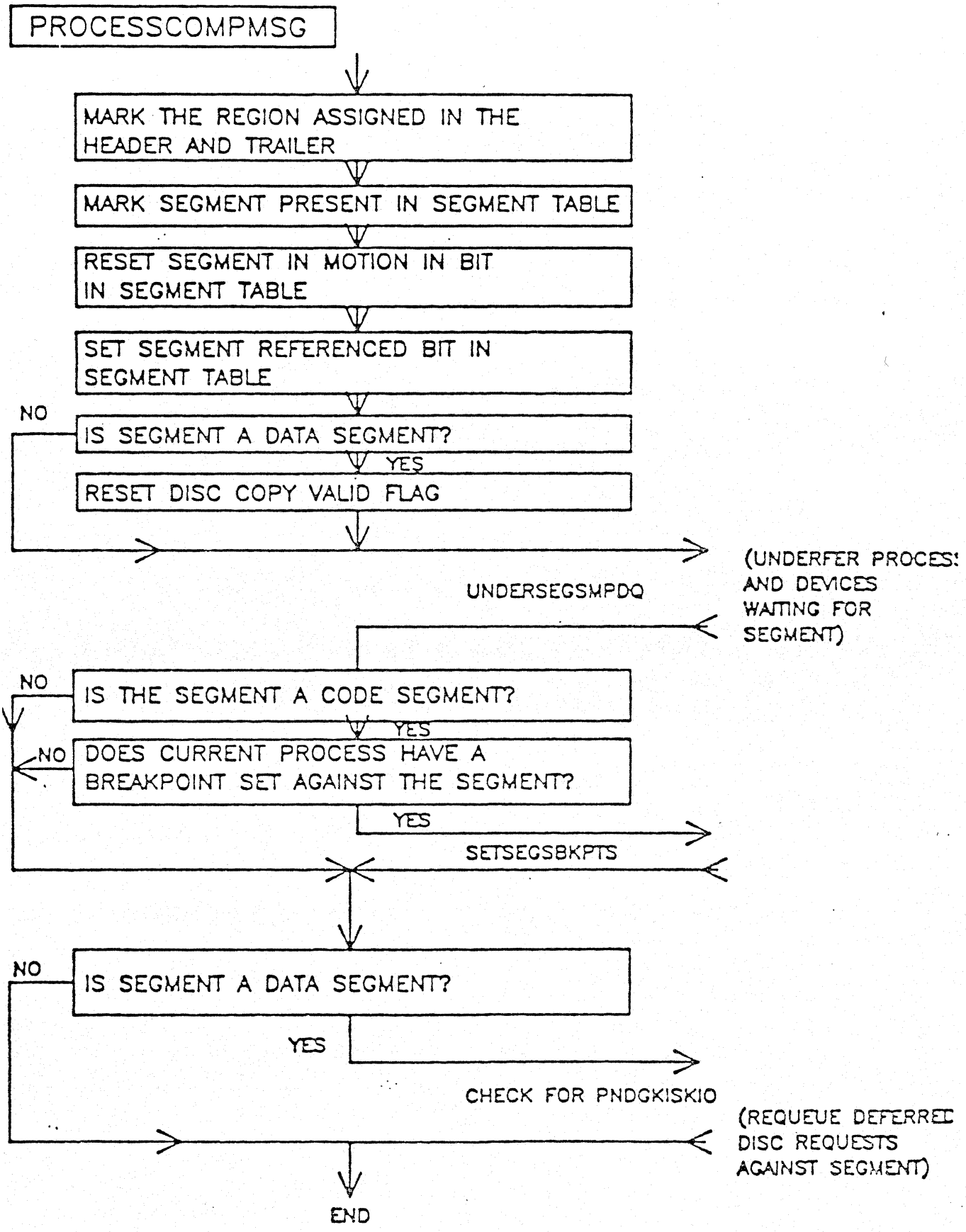
RESCHEDULE

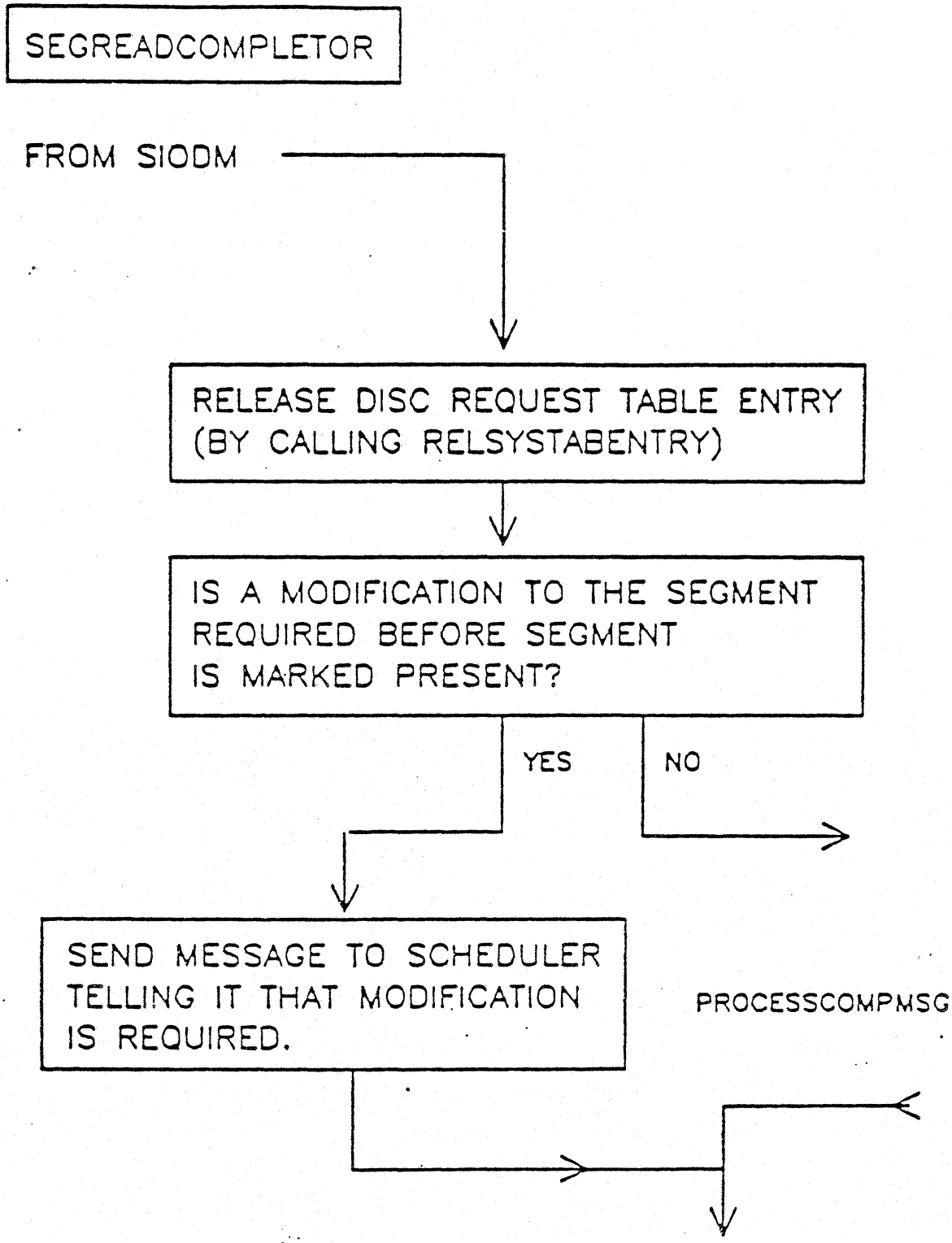
(subroutine of DSP)

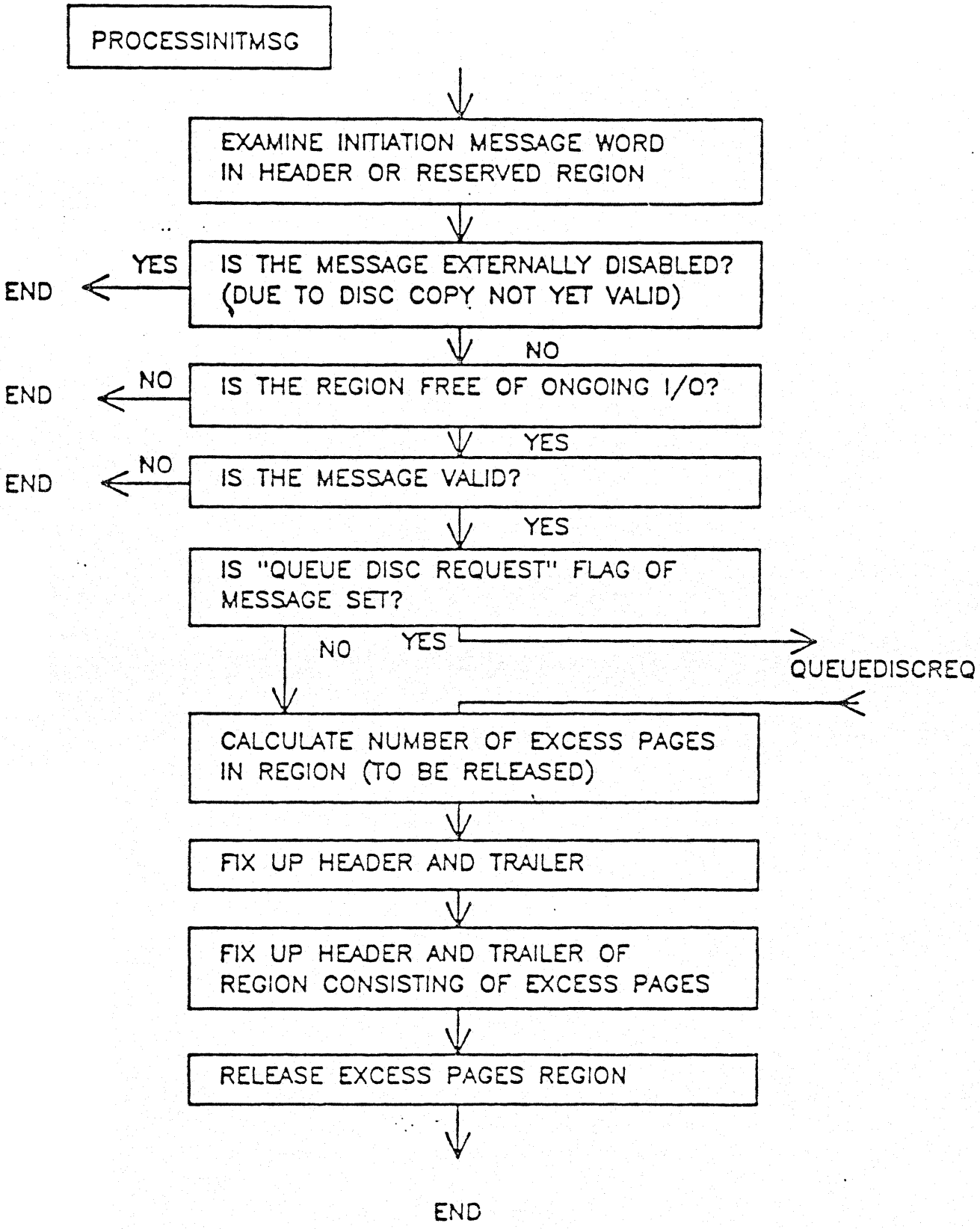


UNDEFERSEGSMPDQ

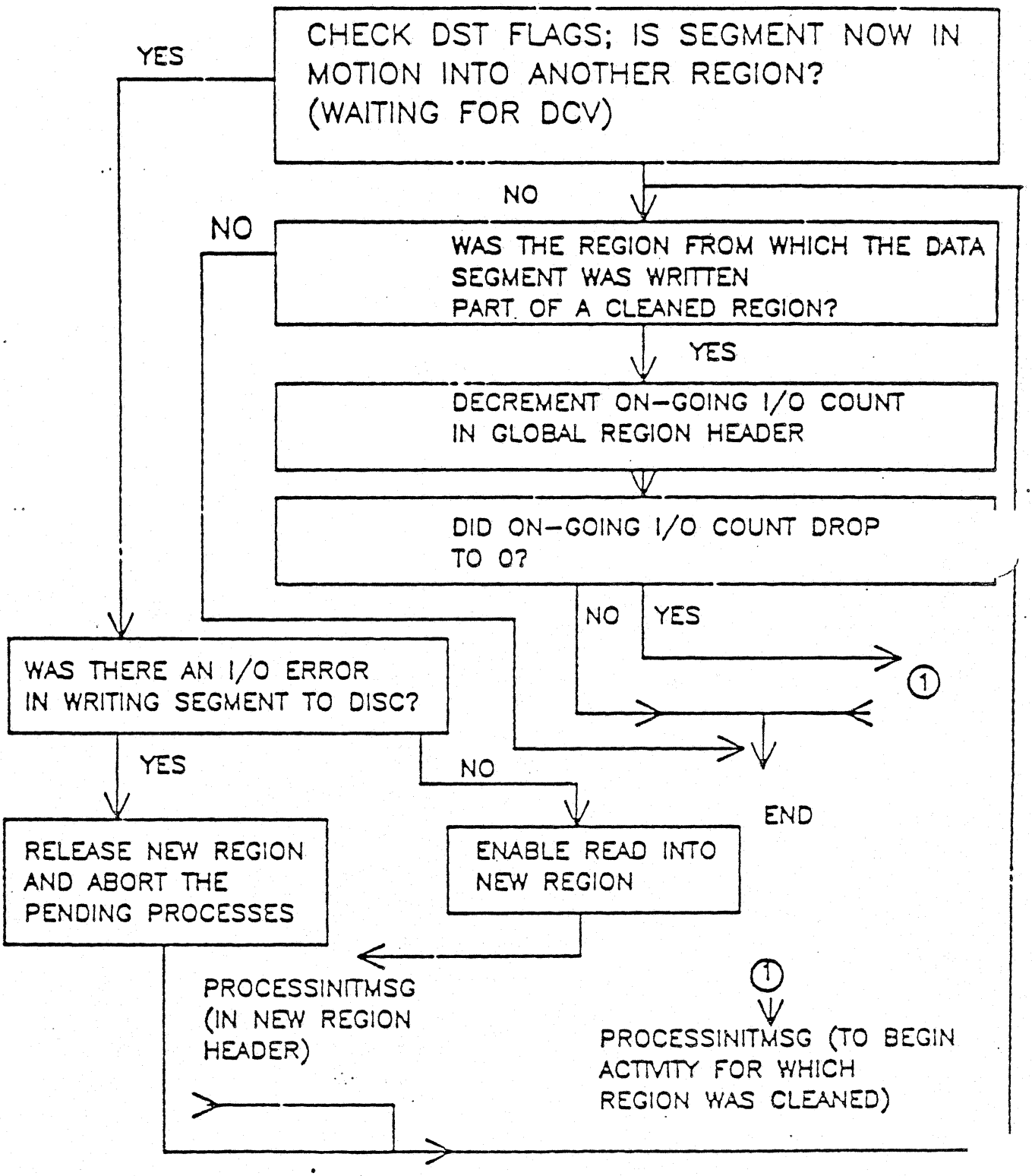


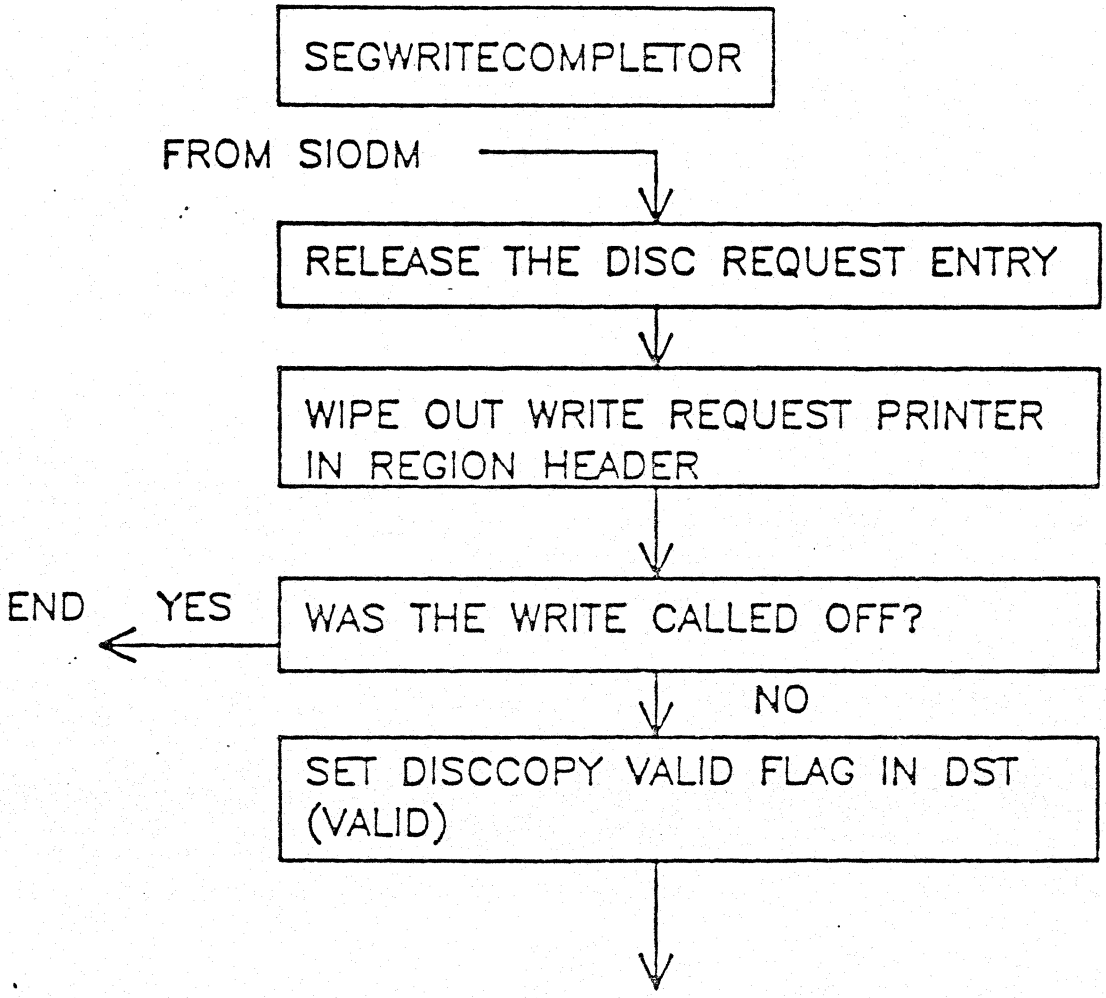

PROCESSCOMPMSG
MARK THE REGION ASSIGNED IN THE HEADER AND TRAILER
MARK SEGMENT PRESENT IN SEGMENT TABLE
RESET SEGMENT IN MOTION IN BIT IN SEGMENT TABLE
SET SEGMENT REFERENCED BIT IN SEGMENT TABLE
NO
IS SEGMENT A DATA SEGMENT?
YES
RESET DISC COPY VALID FLAG
(UNDERFER PROCESS AND DEVICES WAITING FOR SEGMENT)
UNDERSEGSMPDQ
NO
IS THE SEGMENT A CODE SEGMENT?
YES
NO
DOES CURRENT PROCESS HAVE A BREAKPOINT SET AGAINST THE SEGMENT?
YES
SETSEGSBKPTS
NO
IS SEGMENT A DATA SEGMENT?
YES
CHECK FOR PNDGKISKIO
(REQUEUE DEFERRED DISC REQUESTS AGAINST SEGMENT)
END


SEGREADCOMPLETOR
FROM SIODM
RELEASE DISC REQUEST TABLE ENTRY (BY CALLING RELSYSTABENTRY)
IS A MODIFICATION TO THE SEGMENT REQUIRED BEFORE SEGMENT IS MARKED PRESENT?
YES
NO
SEND MESSAGE TO SCHEDULER TELLING IT THAT MODIFICATION IS REQUIRED.
PROCESSCOMPMSG


PROCESSINITMSG
EXAMINE INITIATION MESSAGE WORD IN HEADER OR RESERVED REGION
IS THE MESSAGE EXTERNALLY DISABLED? (DUE TO DISC COPY NOT YET VALID)
YES
END
NO
IS THE REGION FREE OF ONGOING I/O?
NO
END
YES
IS THE MESSAGE VALID?
NO
END
YES
IS "QUEUE DISC REQUEST" FLAG OF MESSAGE SET?
YES
QUEUEDISCREQ
NO
CALCULATE NUMBER OF EXCESS PAGES IN REGION (TO BE RELEASED)
FIX UP HEADER AND TRAILER
FIX UP HEADER AND TRAILER OF REGION CONSISTING OF EXCESS PAGES
RELEASE EXCESS PAGES REGION
END

SEGWRITECOMPLETER (CONTINUED)




SEGWRITECOMPLETOR
FROM SIODM
RELEASE THE DISC REQUEST ENTRY
WIPE OUT WRITE REQUEST PRINTER IN REGION HEADER
WAS THE WRITE CALLED OFF?
END
YES
NO
SET DISCCOPY VALID FLAG IN DST (VALID)

MEMORY MANAGER
I/O INTERFACE

(1) AFTER DELETING A SEGMENT WHICH HAS NOT BEEN REFERENCED (SINCE LAST TRIP THROUGH MEMORY) MAKEROOM CALLS MAKEOC TO CONVERT THE SEGMENT TO AN OVERLAY CANDIDATE. MAKEOC WILL THEN CALL STARTSEGWRITE TO WRITE OUT THE CONTENTS OF THE OVERLAY AREA (IF REQUIRED).

(2) STARTSEGWRITE BUILDS THE DISC REQUEST ENTRY AND THEN CALLS QUEUEDISCREQ TO INSERT THE DISC REQUEST ENTRY INTO ITS PLACE IN THE DISC REQUEST QUEUE BY URGENCY CLASS (PROCESS PRIORITY).

(3) QUEUEDISCREQ INSERTS THE REQUEST INTO THE DISC QUEUE AND CALLS THE I/O MONITOR SIODM. SIODM SELECTS A REQUEST AND CALLS THE DRIVER.

(4) WHEN THE MEMORY MANAGEMENT WRITE REQUEST IS COMPLETE SIODM WILL CALL SEGWRITECOMPLETOR.

(5) SEGWRITECOMPLETOR RELEASES THE ENTRY IN THE DISC REQUEST QUEUE AND SETS THE DCY FLAG IN THE DST ENTRY. IT THEN DECREMENTS THE ONGOING I/O COUNT FOR THE REGION IN THE HEADER. IF THIS COUNT (IN THE HEADER!!!) FALLS TO ZERO THEN SEGWRITECOMPLETOR WILL CALL PROCESSINITMSG TO BEGIN THE ACTIVITY FOR WHICH THE REGION WAS RESERVED.

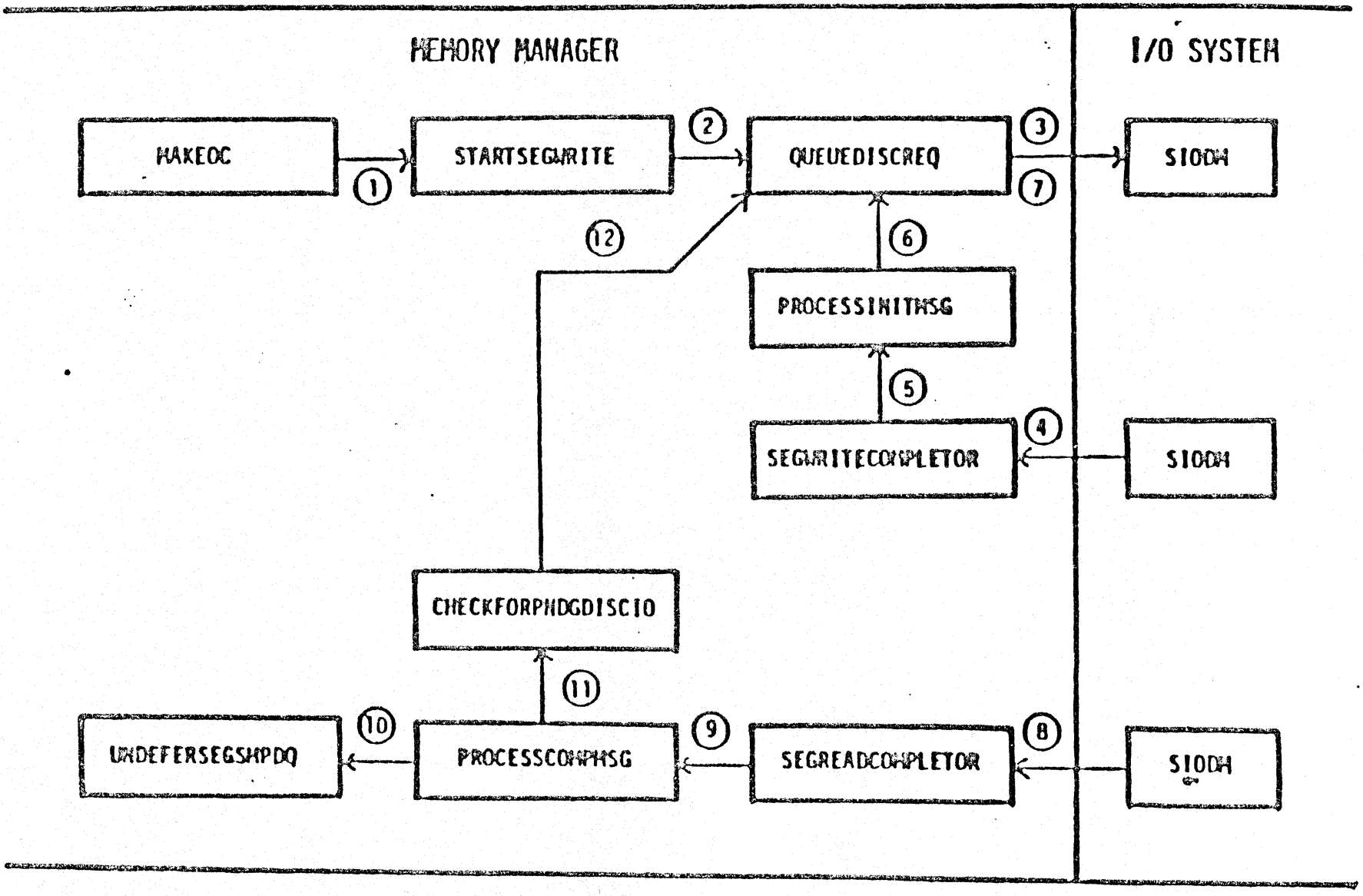
(6) PROCESSINITMSG WILL RELEASE ANY UNNEEDED PAGES IN THE REGION AND CALL QUEUEDISCREQ TO QUEUE THE READ REQUEST FOR THE SEGMENT DESTINED FOR THE RESERVED REGION.

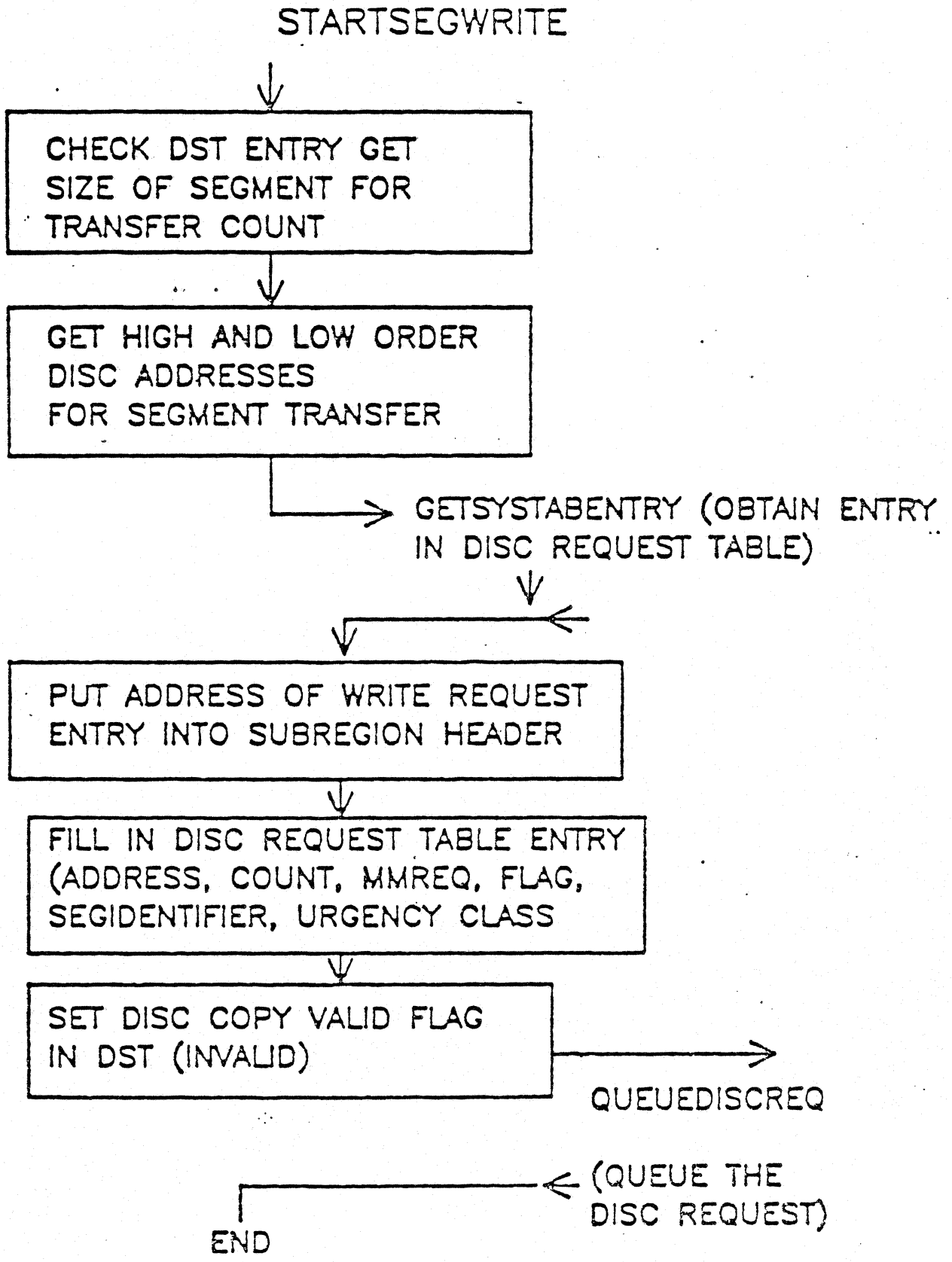
(7) QUEUEDISCREQ WILL INSERT THE READ REQUEST INTO THE DISC QUEUE AND THE CALL SIODM. SIODM WILL SELECT THE REQUEST AND CALL THE DRIVER.

(8) AFTER THE MEMORY MANAGEMENT READ REQUEST IS COMPLETE SIODM WILL CALL SEGREADCOMPLETOR.

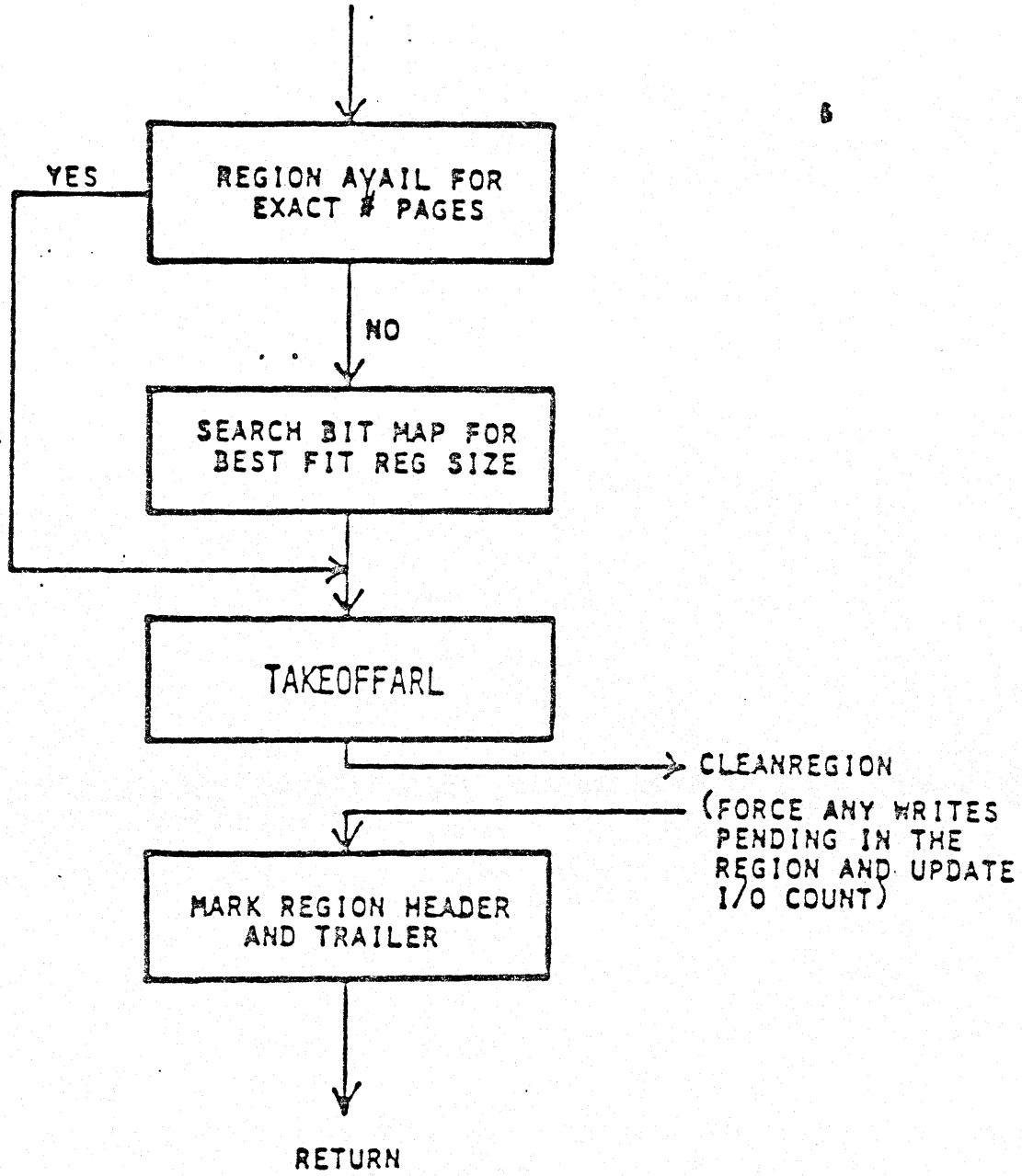
(9) SEGREADCOMPLETOR RELEASES THE DISC REQUEST ENTRY AND THEN CALL PROCESSCOMPMSG TO COMPLETE THE ACTIVITY ASSOCIATED WITH THE SEGMENT READ.

I/O INTERFACE

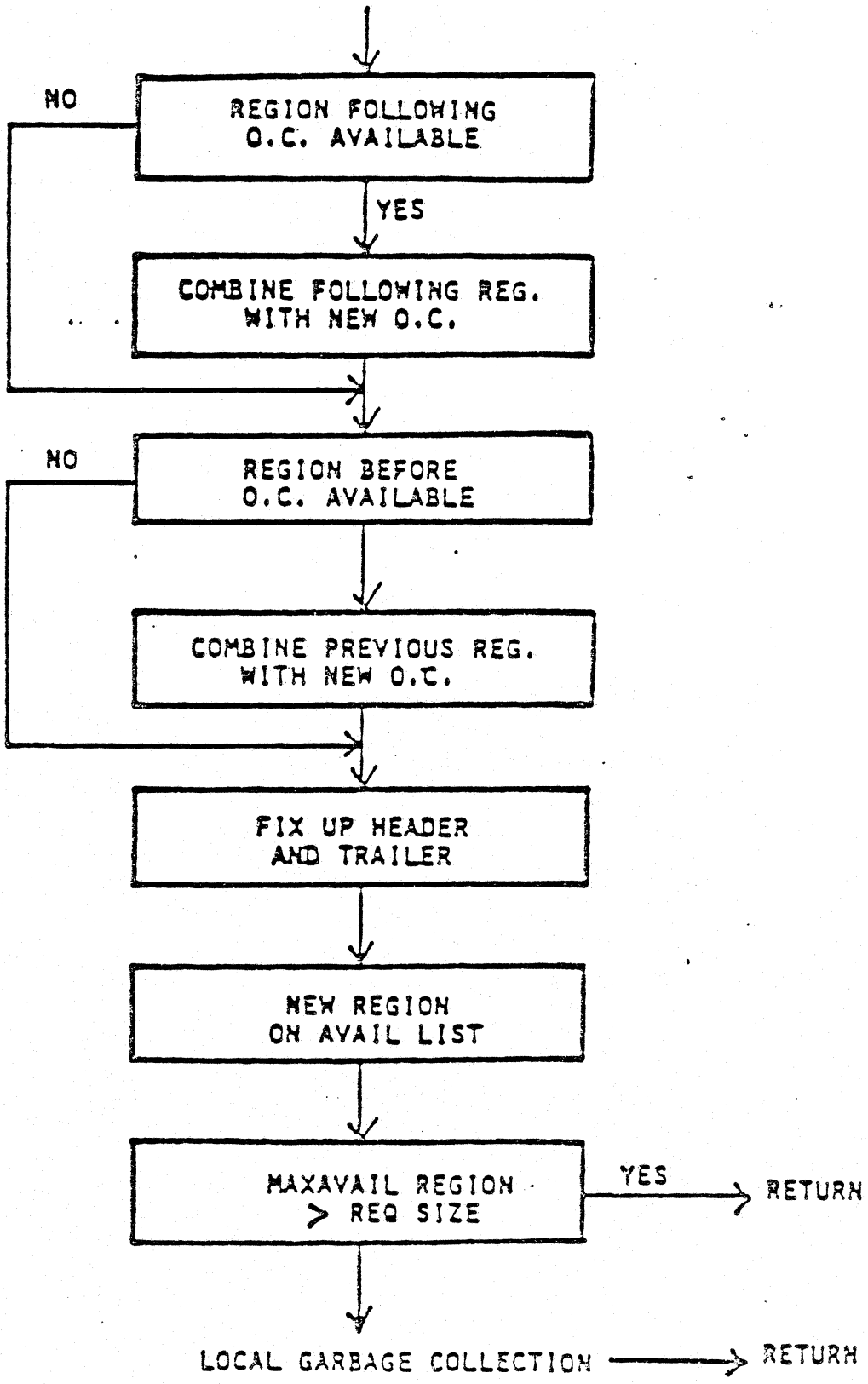



STARTSEGWRITE
CHECK DST ENTRY GET SIZE OF SEGMENT FOR TRANSFER COUNT
GET HIGH AND LOW ORDER DISC ADDRESSES FOR SEGMENT TRANSFER
GETSYSTABENTRY (OBTAIN ENTRY IN DISC REQUEST TABLE)
PUT ADDRESS OF WRITE REQUEST ENTRY INTO SUBREGION HEADER
FILL IN DISC REQUEST TABLE ENTRY (ADDRESS, COUNT, MMREQ, FLAG, SEGIDENTIFIER, URGENCY CLASS
SET DISC COPY VALID FLAG IN DST (INVALID)
QUEUEDISCREQ
(QUEUE THE DISC REQUEST)
END

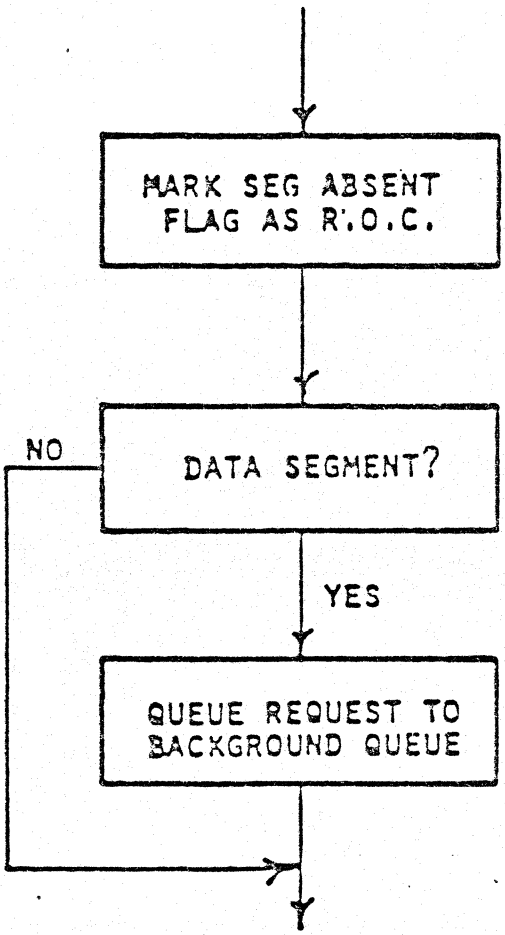
RESERVEREGION



RELEASEREGION

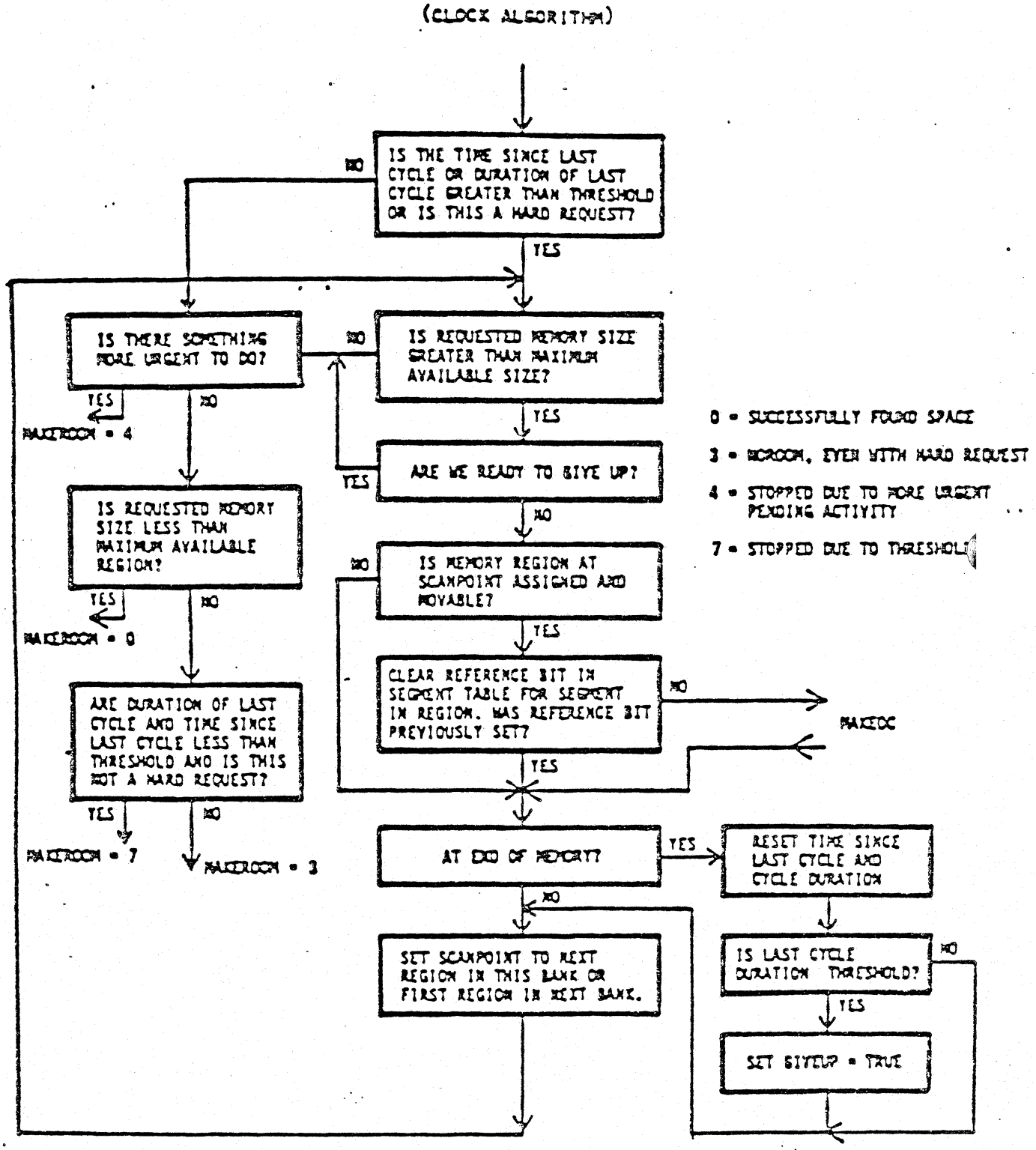


MAKEOC

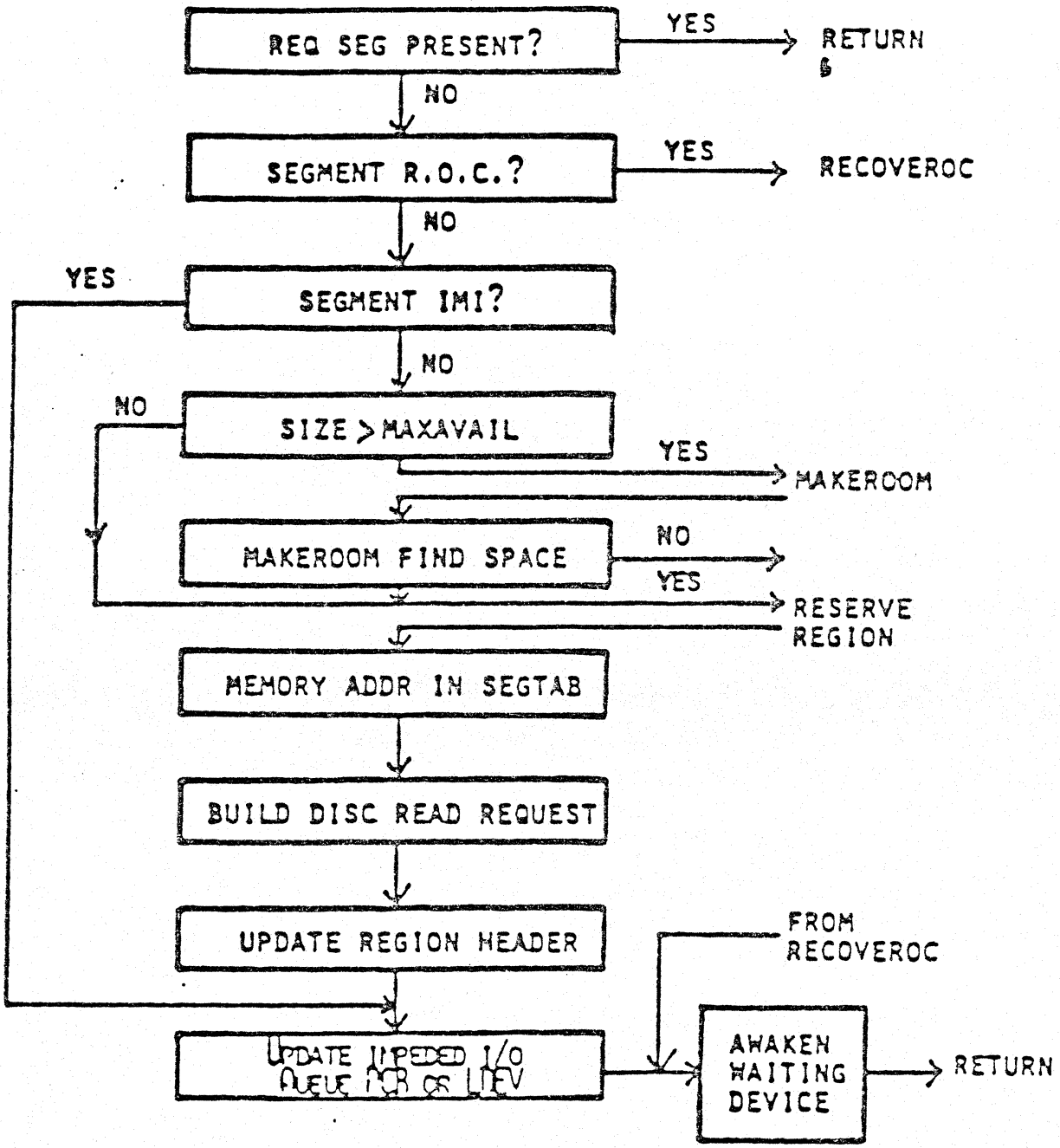


RELEASE REGION

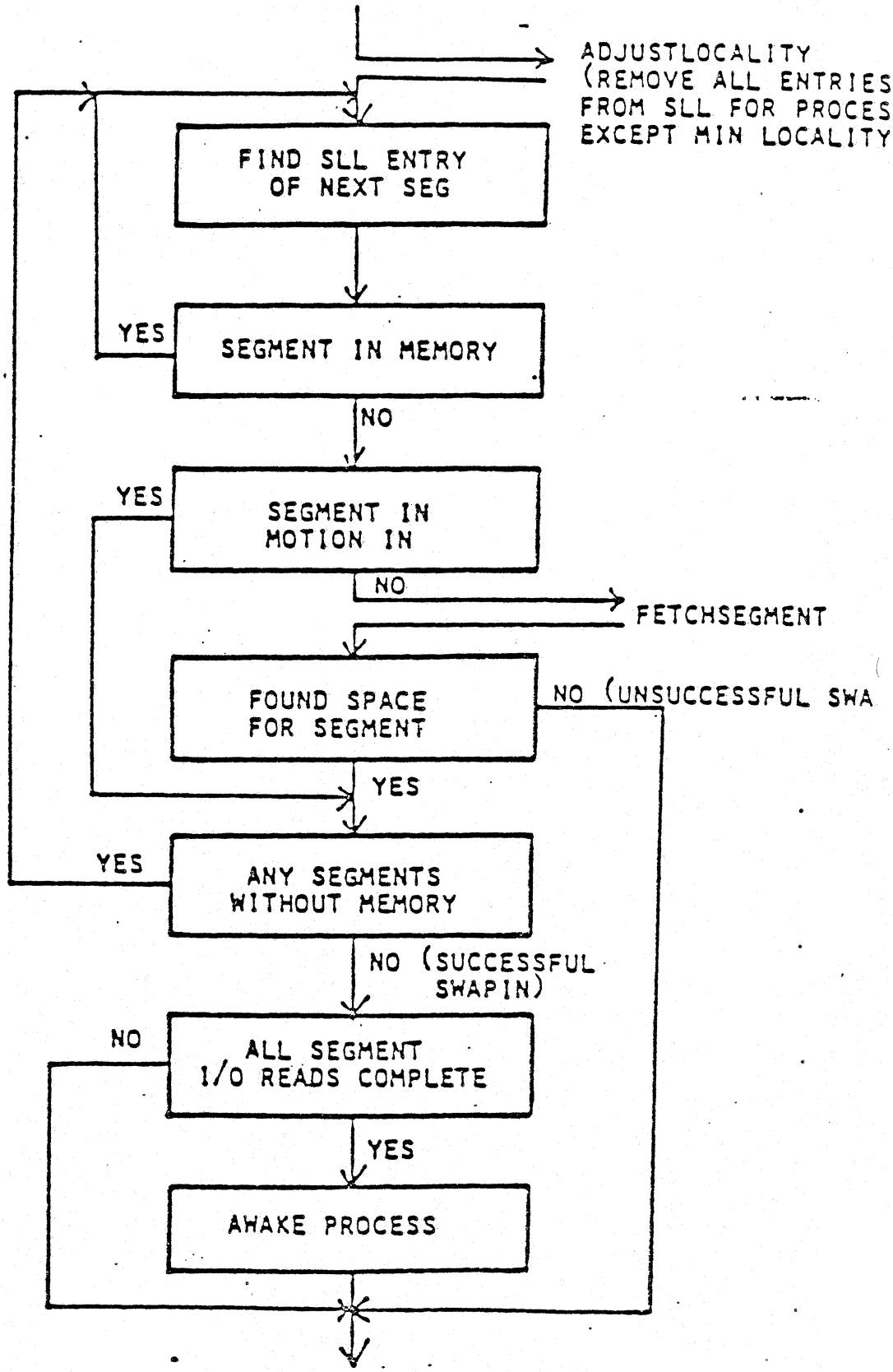
MAKEROOM



FETCHSEGMENT

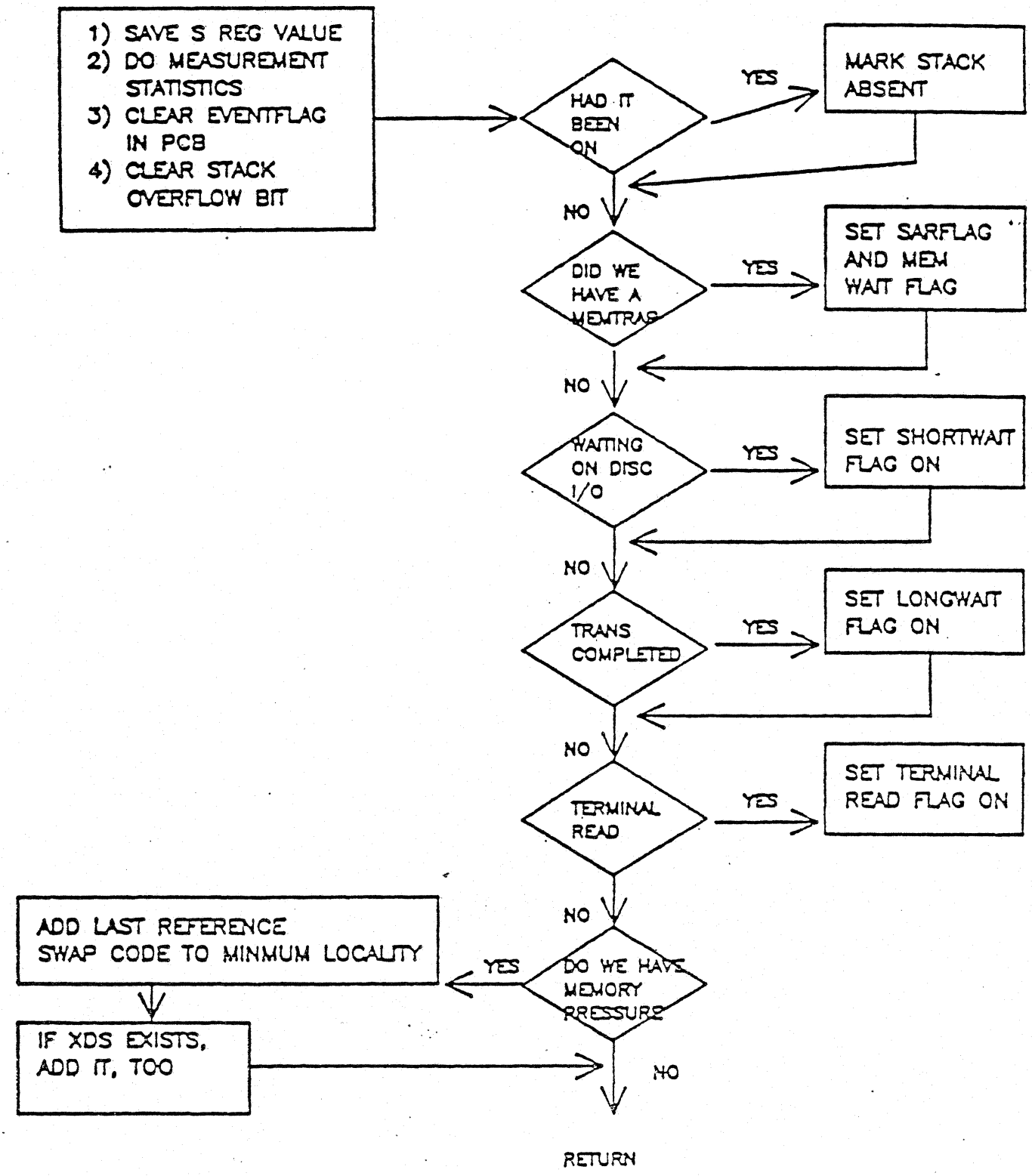


SWAPIN



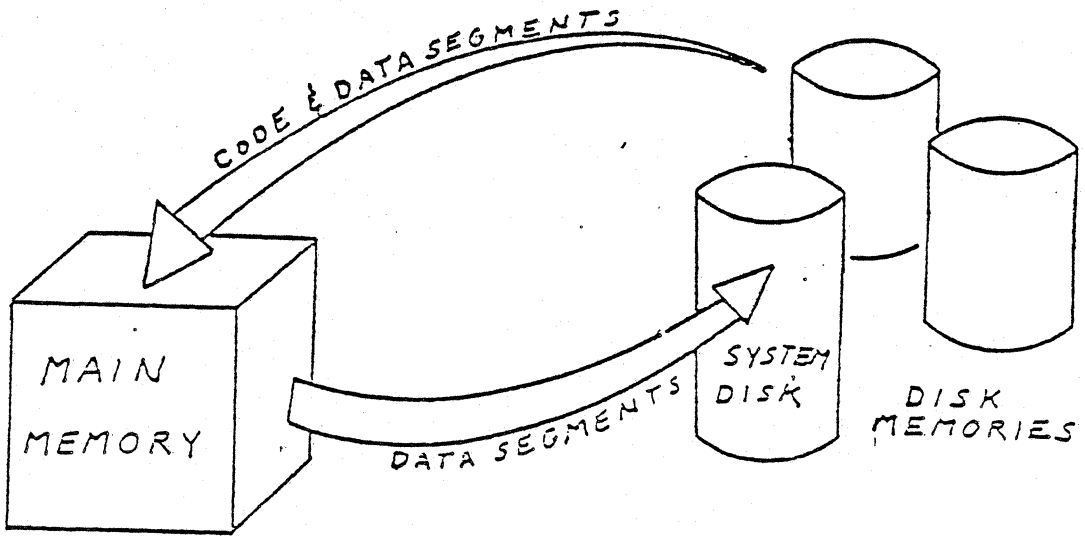
SAVESTATE

SUBROUTINE SAVESTATE



VIRTUAL MEMORY

VIRTUAL MEMORY



DATA SEGMENTS

1. SWAPPED OUT TO SYSTEM DISC ONLY (VIRTUAL MEMORY).
2. VIRTUAL MEMORY IS A SPECIALLY RESERVED AREA ON THE SYSTEM DISC FOR DATA SEGMENT SWAPPING. IT IS USED EXCLUSIVELY FOR THIS PURPOSE.

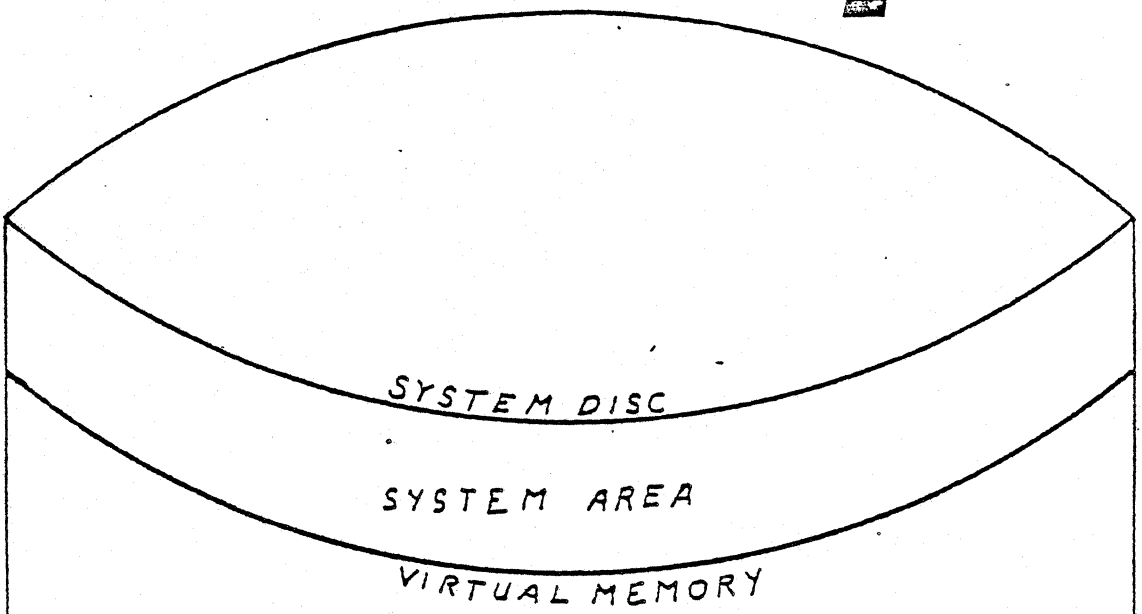
CODE SEGMENTS

1. PROGRAM FILES THEMSELVES ACT AS VIRTUAL MEMORY FOR CODE. SINCE THE CODE IS NON-MODIFIABLE IT NEED ONLY BE READ INTO MEMORY NOT OUT.
2. BECAUSE THEY ARE FILES THEY MAY RESIDE ON ANY RANDOM ACCESS DISC.

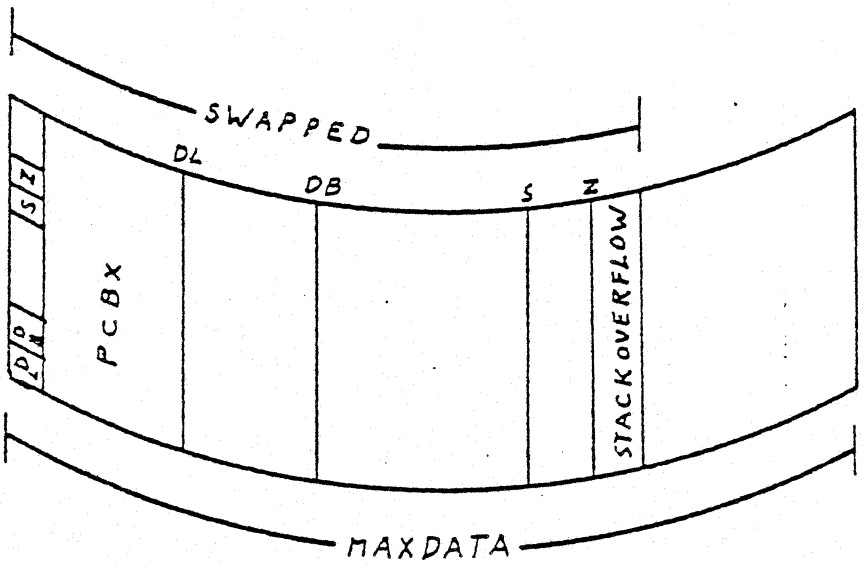
LIBRARIES

* SYSTEM LIBRARY (SL) RESIDES ON THE SYSTEM DISC

* USER SL'S RESIDE ON ANY DISC - WHICH EVER DISC THE LIBRARY RESIDES ON IS WHERE THE CODE SEGMENTS ARE LOADED FROM.



A DATA STACK IN VIRTUAL MEMORY



VIRTUAL MEMORY

- FOR DATA SEGMENTS ONLY
- CAN EXIST ON ANY AND ALL SYSTEM DOMAIN DISCS
- SIZE IS CONFIGURABLE
- SIZE OF VIRTUAL MEMORY CAN VARY BETWEEN VOLUMES (MINIMUM TOTAL VIRTUAL MEMORY IS 0)
- SPREADING VIRTUAL MEMORY OVER SEVERAL DISCS:
  1) REDUCES DELAYS ON DISC REQUESTS DIRECTED TOWARD LDEV 1.
  2) ELIMINATES CONSTRAINTS ON VMEM SIZE.
  3) BALANCES DISC REQUEST QUEUE LENGTHS AMONG VOLUMES, THUS MAXIMIZING THE EFFECTIVENESS OF OVERLAPPED SEEKS.

ASSIGNING VIRTUAL MEMORY

- USE THE BOOT DIALOGUE FOR COOLSTARTS, COLDSTARTS, AND UPDATES.
- USE SYSDUMP DIALOGUE FOR RELOAD. THE BOOT DIALOGUE CAN OVERRIDE VALUES ON TAPE OR SERIAL DISC.
- EXAMPLE DIALOGUE:

```
VIRTUAL MEMORY CHANGES?
Y
LIST VIRTUAL MEMORY DEVICE ALLOCATION?
Y
VOLUME NAME    LDEV #   VM ALLOCATION
FHD0001          1          5
MUTHA            2          0

ENTER VOLUME NAME, SIZE IN KILOSECTORS
MUTHA, 7

LIST VIRTUAL MEMORY DEVICE ALLOCATION?
N
```

AFTER VIRTUAL MEMORY HAS BEEN ASSIGNED:

1) THE DIRECTORY AND RECOVERABLE DATA SEGMENTS (IDD,ODD,JMAT, AND WELCOME MESSAGE) REMAIN ON LDEV 1.

2) VIRTUAL MEMORY SPACE FOR DATA SEGMENTS IS ASSIGNED CYCLICALLY AS THE SEGMENTS ARE CREATED, STARTING WITH LDEV 1.

RESTRICTIONS AND CAUTIONS:

1) VMEM SPACE ON LDEV1 CAN ONLY BE ALTERED ON A RELOAD.

2) DEFAULT VIRTUAL MEMORY SIZES ARE,
   A) FOR LDEV1, THE VALUE STORED IN THE COLD LOAD INFO TABLE.
   B) FOR ALL OTHER VMS DEVICES, 0.

3) MAXIMUM VIRTUAL MEMORY SIZES ARE,
   A) FOR LDEV 1, 64K SECTORS
   B) FOR ALL OTHER DISCS, 2 BILLION SECTORS.

INTERNALS OF VIRTUAL MEMORY MANAGEMENT

CODE:

- EIGHT PROCEDURES SPECIFICALLY FOR VIRTUAL MEMORY MANAGEMENT.
- PARTS OF MAINSEG1 IN INITIAL.

DATA

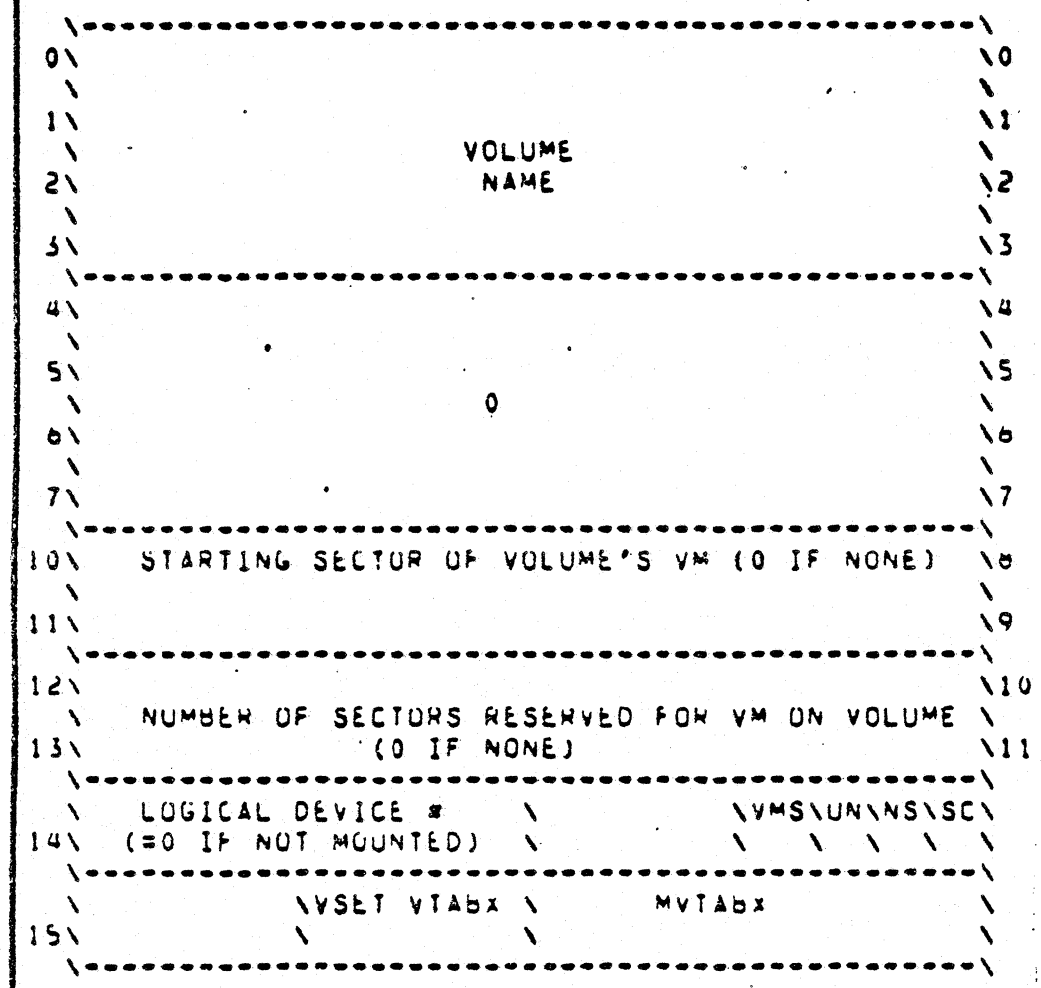
- VIRTUAL DISC SPACE MANAGEMENT TABLE. ONE OVERHEAD ENTRY AND ONE ENTRY FOR EACH DISC WHICH HAS CONFIGURED VIRTUAL MEMORY.
- VOLUME TABLE INDICATES WHICH VOLUMES HAVE VIRTUAL MEMORY, SECTOR ADDRESS OF WHERE VIRTUAL MEMORY BEGINS, AND THE NUMBER OF SECTORS RESERVED FOR VIRTUAL MEMORY ON EACH DISC.

THE VOLUME TABLE

* INDICATES WHICH VOLUMES HAVE VIRTUAL MEMORY

- DESCRIBED BY DST %35.
- ALL ENTRIES ARE %16 WORDS LONG
- FOR EACH VIRTUAL MEMORY SUPPORTING VOLUME:
  - WORD %14 .(12:1) IS THE VMS FLAG
  - WORDS %10 AND %11 CONTAIN SECTOR ADDRESS OF WHERE VIRTUAL MEMORY STARTS ON THIS VOLUME. IF 0 THEN NO VIRTUAL MEMORY HAS BEEN ASSIGNED TO THIS VOLUME.
  - WORDS %12 AND %13 CONTAIN THE NUMBER OF SECTORS RESERVED FOR VIRTUAL MEMORY ON THIS VOLUME. IF 0 THEN NO VIRTUAL MEMORY AREA HAS BEEN ASSIGNED.

TYPICAL SYSTEM VOLUME ENTRY

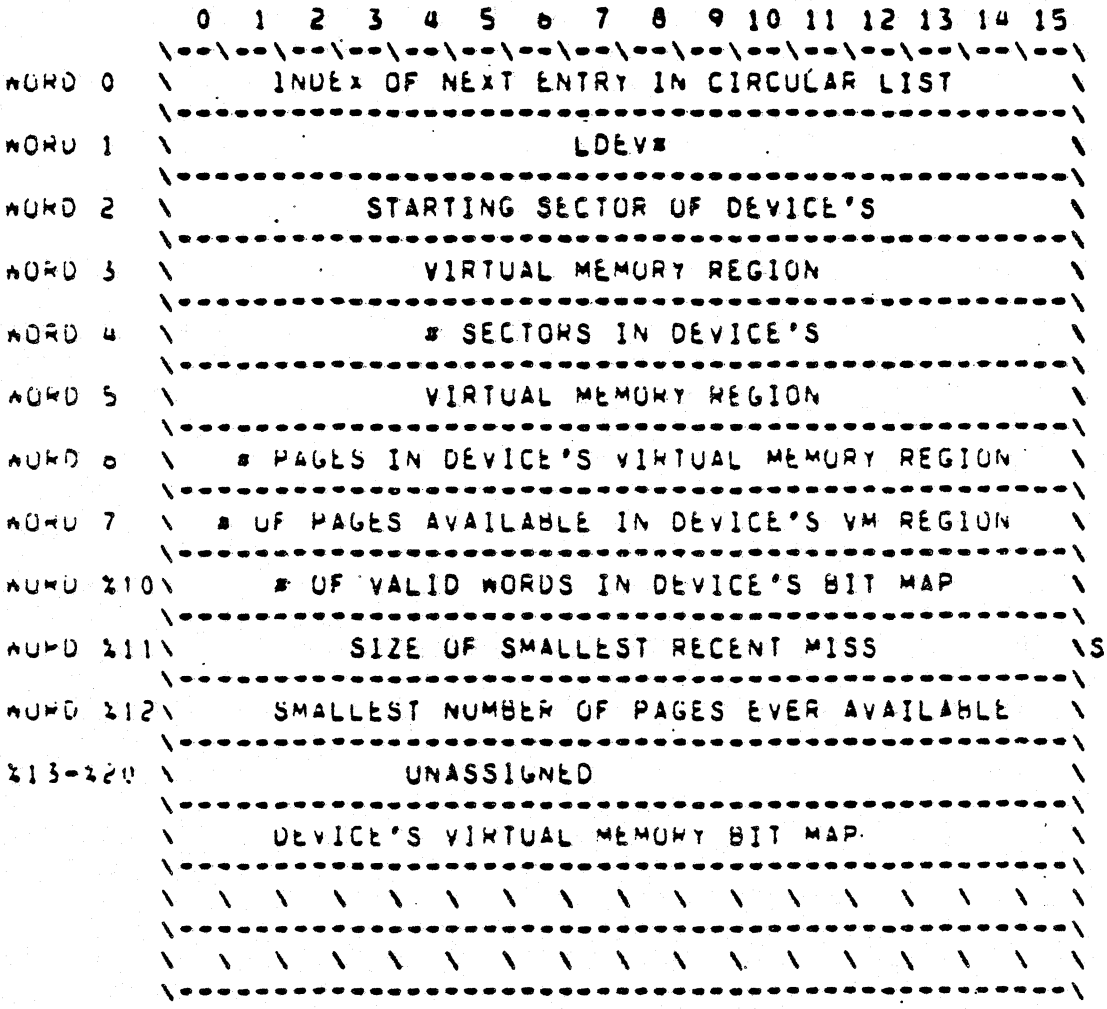


NS - NON-SYSTEM DOMAIN
SC - SCRATCH
UN - UNREADABLE/ UNFORMATTED
VMS - VIRTUAL MEMORY SUPPORTING

VIRTUAL DISC SPACE MANAGEMENT TABLE

- TABLE IS CORE RESIDENT.
- DESCRIBED BY DST %47.
- EACH ENTRY IS %20 WORDS LONG.
- ONE OVERHEAD ENTRY, AND ONE ENTRY FOR EACH VMS DEVICE.
- TYPICAL ENTRY HAS LDEV NUMBER FOR DEVICE, BITMAP OF VIRTUAL MEMORY ON THE DEVICE, AND A LINK TO THE NEXT ENTRY.
- IS USED WHEN VIRTUAL MEMORY IS BEING SELECTED FROM, OR RETURNED TO A VMS DISC.

VDSMTAB GENERAL ENTRY FORMAT



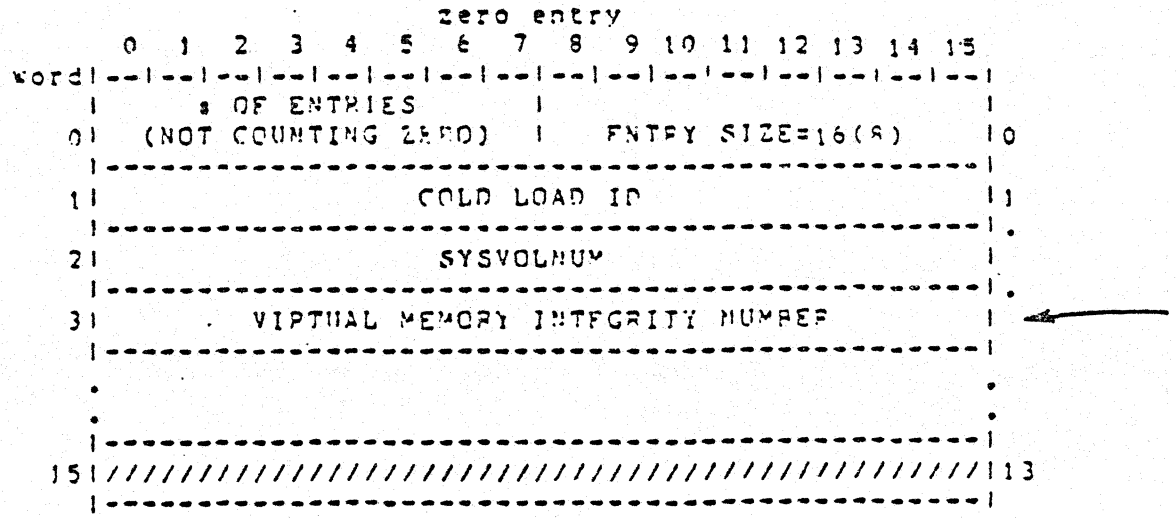
***COMMENT: A BIT ON IN A DEVICE'S VMBIT MAP
==> CORRESPONDING VM PAGE IS FREE.

HEWLETT PACKARD

MPE IV VM

VOLUME TABLE

SIR #22=#26
DST #29=#35



Virtual memory integrity number (copy of cold load ID) put in Volume Table by MPE IV. Used to when going from MPE IV to MPE III and back to MPE IV. Assumes default VM - VM on LDEV 1 only - when returning to MPE IV. (Procedure VERIFY VM)

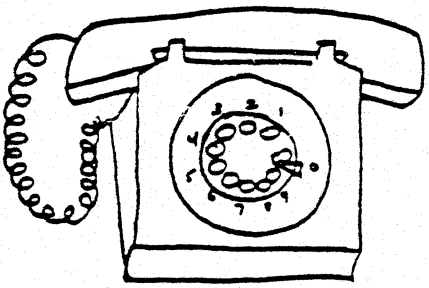
VIRTUAL MEMORY INTEGRITY NUMBER

* WORD 3 of VOLUME TABLE, ENTRY 0.
* NUMBER IS COPY OF COLD LOAD ID.
* PUT HERE BY MPE III AND USED WHEN GOING FROM MPE IV TO MPE III AND BACK, AGAIN, TO MPE IV.
* ASSURES DEFAULT VM — ON LDEV 1 ONLY — WHEN RETURNING TO MPE III.

VIRTUAL MEMORY INTEGRITY NUMBER

* WORD 3 of VOLUME TABLE, ENTRY 0.
* NUMBER IS COPY OF COLD LOAD ID.
* PUT HERE BY MPE III AND USED WHEN GOING FROM MPE IV TO MPE III AND BACK, AGAIN, TO MPE IV.
* ASSURES DEFAULT VM — ON LDEV 1 ONLY — WHEN RETURNING TO MPE III.

INTERRUPT SYSTEM



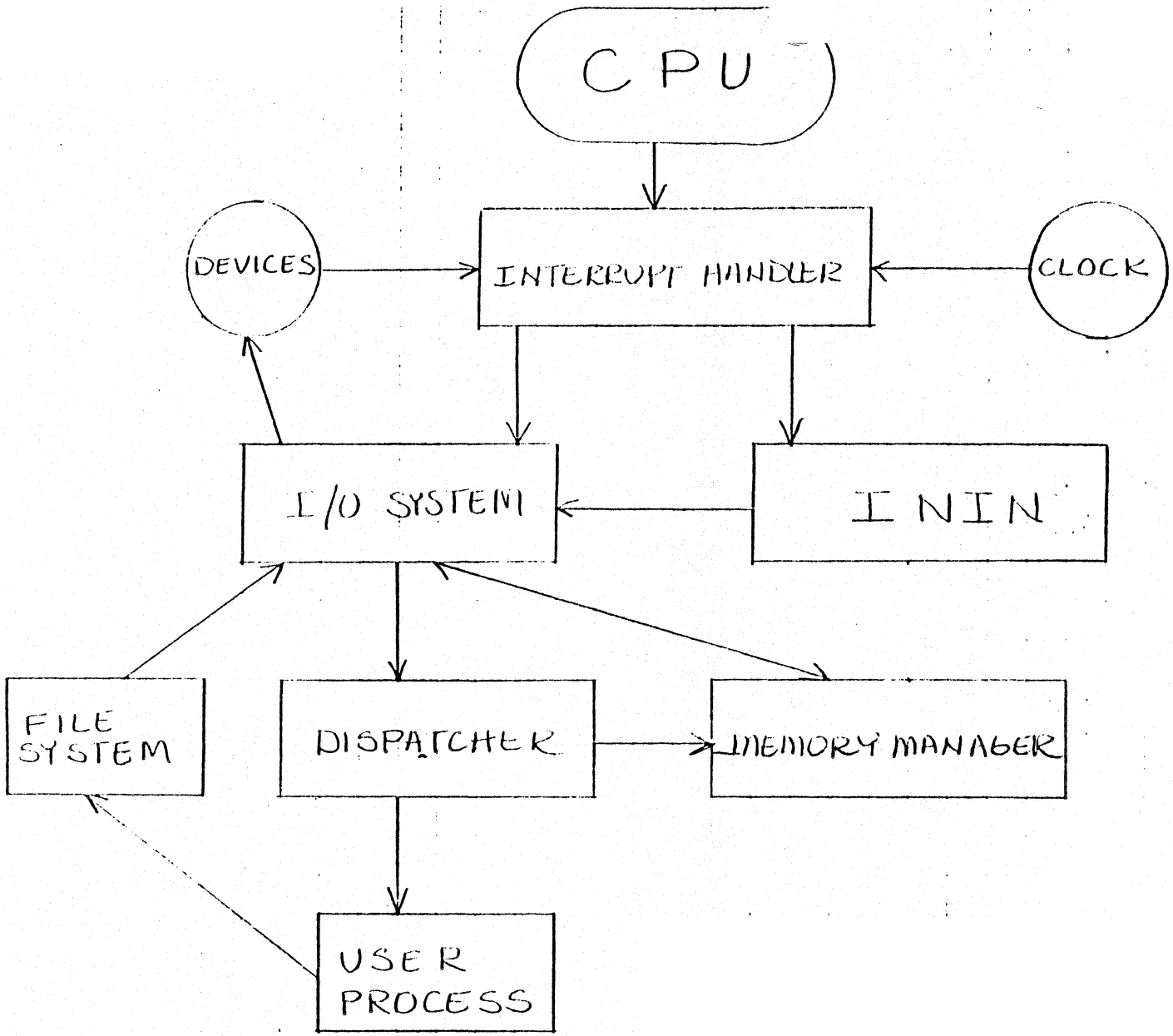
SE 335 INTERRUPT SECTION

OBJECTIVE:

- KNOW THE 3 TYPES OF INTERRUPTS AND HOW THEY ARE HANDLED BY MICROCODE.

- BE ABLE TO TRACE THROUGH STACK MARKERS ON THE ICS AND DETERMINE THE STATUS OR THE SYSTEM.

- UNDERSTAND HOW THE DISPATCHER INTERACTS WITH AND USES THE INTERRUPT SYSTEM.



INTERRUPT SYSTEM OVERVIEW (INTERACTIONS)

EXTERNAL INTERRUPTS

- A SIGNAL SENT TO THE CPU THAT REPRESENTS A SPECIFIC PHYSICAL CONDITION

EXAMPLE:

DISC SWITCHED ON-LINE

INTERNAL INTERRUPTS

- TYPICALLY ABNORMAL CONDITIONS RESULTING FROM PROGRAM EXECUTION. DETECTED BY THE CPU HARDWARE OR MICROCODE.

  * NON ICS TYPE INTERNAL INTERRUPTS HANDLED ON THE EXECUTING STACK.

  * ICS TYPE INTERNAL INTERRUPTS HANDLED ON THE ICS.

TRAPS

TRAPS ARE A SPECIAL TYPE OF INTERNAL INTERRUPT THAT ARE DETECTED BY THE CPU MICROCODE. THEY ARE HANDLED ON THE USER'S STACK.

EXAMPLE:

ARITHMETIC DIVIDE BY ZERO

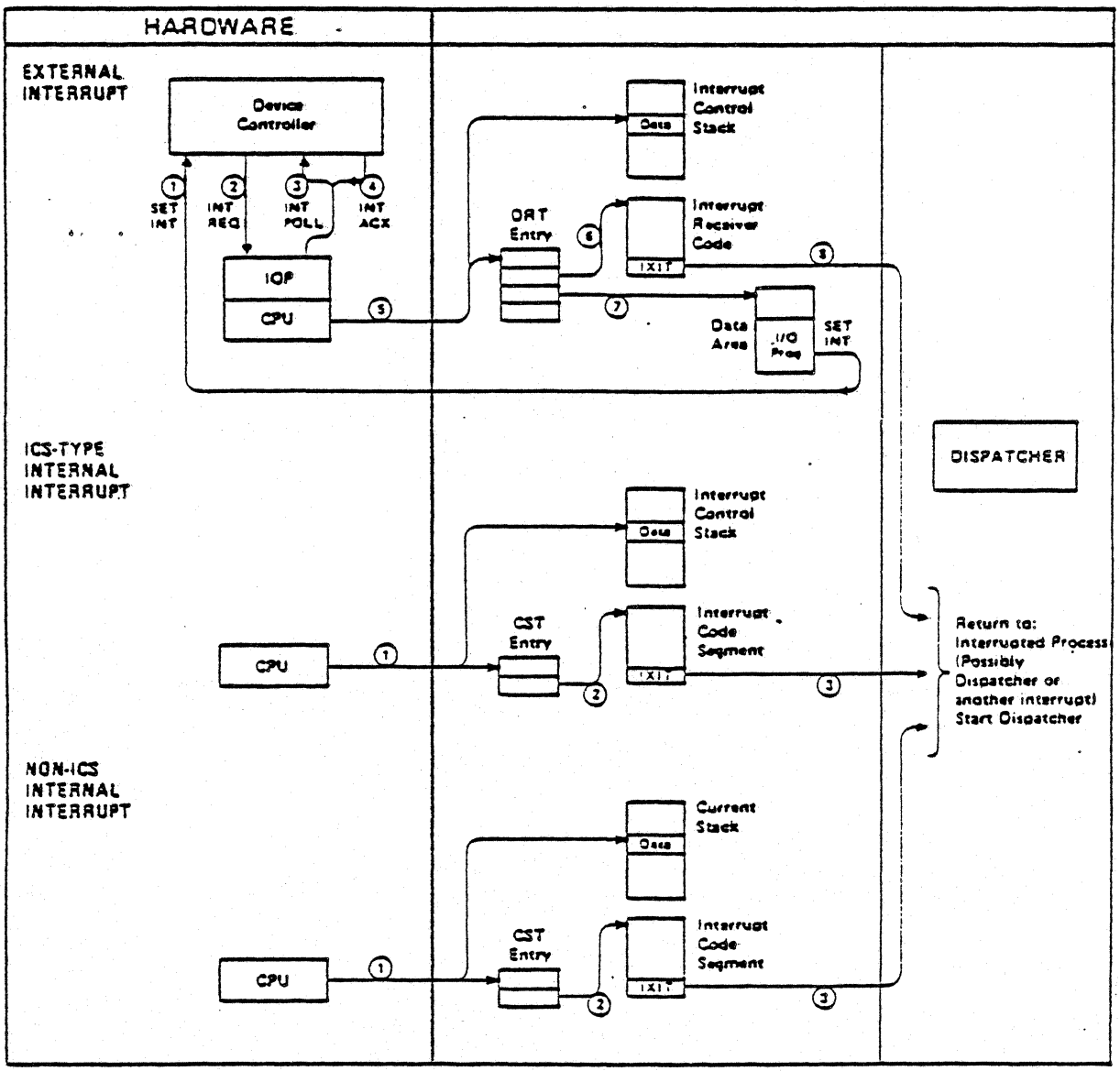


Figure 8-2. Interrupt System Overview

WHEN AN INTERNAL INTERRUPT OCCURS:

- IF INTERNAL INTERRUPT, MICROCODE WILL CHECK CPX1 REGISTER TO SEE WHAT TYPE OF INTERRUPT OCCURRED.

CPX1 REGISTER. The CPX1 register provides 16 bits that are used to monitor the system run mode interrupt status. When a run mode interrupt is experienced, the CPU reads the CPX1 register and checks the content for the cause of the interrupt. The S-bus field code CPX1 reads the CPX1 register and the special field code CCPX affects the CPX1 register as described in the field code definitions. The following is a list of the CPX1 register contents with the significance of each bit:

Bit 0: Integer overflow
Bit 1: Bounds violation
Bit 2: Illegal address
Bit 3: CPU timer
Bit 4: System parity error
Bit 5: Address parity error
Bit 6: Data parity error
Bit 7: Module interrupt
Bit 8: External interrupt
Bit 9: Power fail interrupt
Bit 10: 0
Bit 11: ICS flag
Bit 12: DISP flag
Bit 13: Emulator
Bit 14: LO timer
Bit 15: Option present

- IF TRAP, MICROCODE ALREADY KNOWS WHAT TYPE (BY DEFINITION OF TRAP).

WHEN AN INTERNAL INTERRUPT OCCURS (Cont.)

- MICROCODE DOES AN "IMPLICIT PCAL" TO THE APPROPRIATE INTERRUPT PROCEDURE OF ININ.

EXT. PROG. LABEL (%)	STT NO. (%)	INTERRUPT TYPE	PARAMETER*	EXECUTING STACK**
100401	1	Bounds Violation		
101001	2	Illegal Memory Address		
101401	3	Non-Responding Module		
102001	4	System Parity Error		ICS
102401	5	Address Parity Error		ICS
103001	6	Data Parity Error		ICS
103401	7	Module Interrupt	Module No.	ICS
104001	10	(Unused)		
104401	11	Power Fail		ICS
105001	12	(Unused)		
105401	13	(Unused)		
106001	14	(Unused)		
106401	15	(Unused)		
107001	16	(Unused)		
107401	17	(Unused)		
110001	20	Unimplemented Instruction		
110401	21	STT Violation		
111001	22	CST Violation		

EXT. PROG. LABEL (%)	STT NO. (%)	INTERRUPT TYPE	PARAMETER*	EXECUTING STACK**
111401	23	DST Violation		
112001	24	Stack Underflow		
112401	25	Privileged Mode Violation		
113001	26	(Unused)		
113401	27	(Unused)		
114001	30	Stack Overflow		ICS
114401	31	User Traps		
		a. Integer Overflow	%000001	
		b. Floating-Point Over.	%000002	
		c. Floating-Point Under.	%000003	
		d. Integer Divide by 0	%000004	
		e. Floating-Point Divide by 0	%000005	
		f. Ext. Prec. Floating-Point Overflow	%000010	
		g. Ext. Prec. Floating-Point Underflow	%000011	
		h. Ext. Prec. Floating-Point Divide by 0	%000012	
		i. Decimal Overflow	%000013	
		j. Invalid ASCII digit	%000014	
		k. Invalid Dec. digit	%000015	
		l. Invalid Source Word Count	%000016	
		m. Result Word Count Overflow	%000017	
		n. Decimal Divide by 0	%000020	
115001	32	(Unused)		
115401	33	(Unused)		
116001	34	(Unused)		
116401	35	(Unused)		
117001	36	(Unused)		
117401	37	Absent Code Segment		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120001	40	Trace		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120401	41	STT Entry Uncallable	P-Label	
121001	42	Absent Data Segment	DST No.	
121401	43	Power On		ICS
122001	44	Cold Load		ICS
		a. System I/O (SIO)	0	
		b. Direct I/O (DIO)	Label	

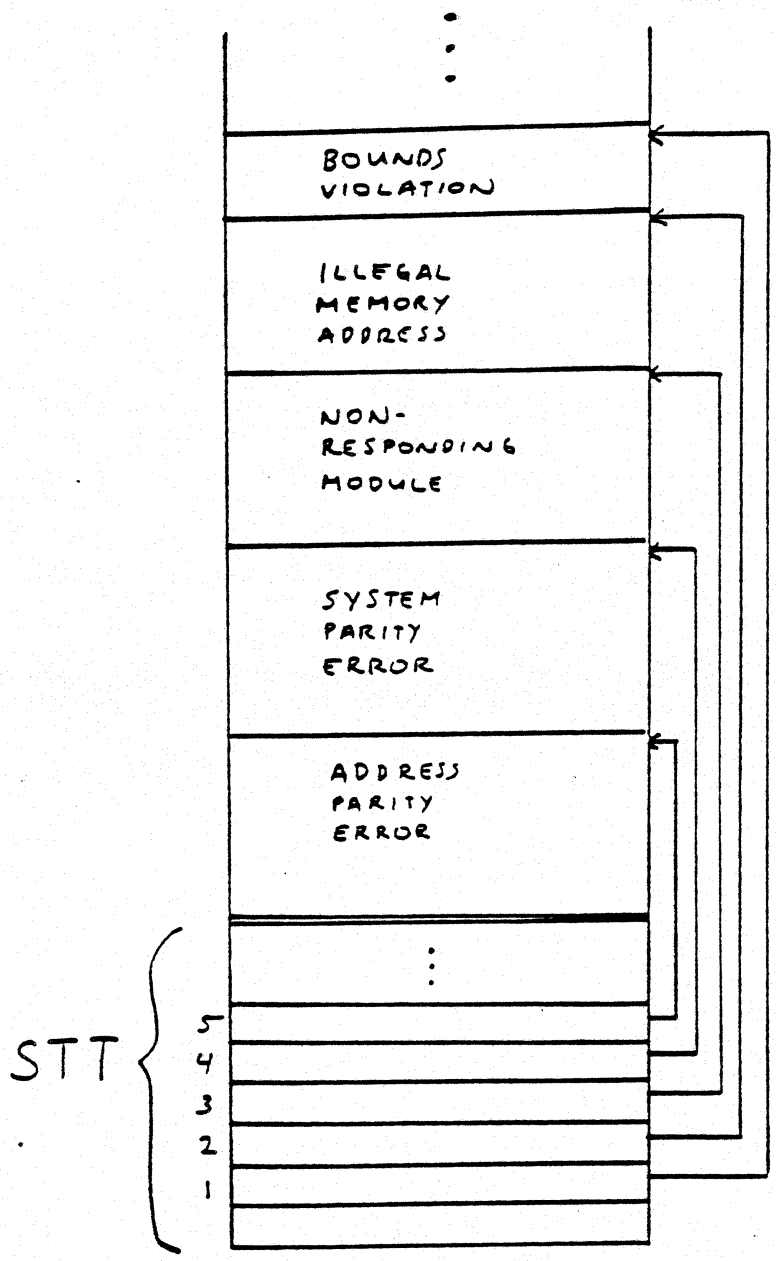
*Unless noted, the parameter is the External Program Label.
**Unless noted, interrupts are serviced on the User Stack.

All User Traps (STT No. %31) are enabled by the User Traps bit in the Status register.

- STATUS REGISTER IS SET TO %100001, WHICH IS WHY ININ IS ALWAYS SEGMENT 1.

* ININ (MODULE 10)

* MUST BE HANDLED SPECIALLY AND EXACTALY BECAUSE THE MICROCODE MUST BE ABLE TO FIND THE PROCEDURES.
* TO HANDLE THIS, INITIAL CONTAINS SPECIAL CODE.
* THE MODULE IS WRITTEN THE SAME AS OTHER SYSTEM MODULES EXCEPT FOR THE MAIN OR OUTER BLOCK.
* THE INTERNAL STT LABELS ARE DEFINED IN THE OUTER BLOCK SO THAT INITIAL CAN SPECIAL BUILD THE STT.
* THE PMAP OF THE COMPILED MODULE DOES NOT REFLECT THE STT.


BOUNDS VIOLATION
ILLEGAL MEMORY ADDRESS
NON-RESPONDING MODULE
SYSTEM PARITY ERROR
ADDRESS PARITY ERROR
STT
5
4
3
2
1

ININ

WHY ARE THE PMAP AND STT FOR ININ DIFFERENT FROM THE LISTING?

- THE OUTER BLOCK OF ININ IS NOT REALLY CODE. IT IS A TABLE USED TO CONSTRUCT THE ORDER IN WHICH THE INTERRUPT ROUTINES ARE EXPECTED (INCLUDING CALLS TO GHOST FOR UNUSED INTERRUPTS).

- THE PMAP IS A RESULT OF PREPING ALL OF ININ, INCLUDING THE OUTER BLOCK.

- DURING INITIAL,
  A. THE OLD STT FOR ININ IS NOT LOADED AND A NEW ONE IS CREATED TO MATCH THE SEQUENCE EXPECTED BY THE CPU MICROCODE.
  B. THE OUTER BLOCK OF ININ IS THROWN AWAY.

THEREFORE,

- 1. THE LENGTH OF AN EXECUTING ININ IS SHORTER THAN THE LENGTH OF THE SAME COMPILED ININ.
  2. IN ORDER TO USE THE PMAP, YOU MUST ADD THE LENGTH OF THE OUTER BLOCK TO ΔP BEFORE COMPARING TO THE PMAP.

USING THE ACCOMPANYING PMAP FOR ININ, TRANSLATE THE FOLLOWING ΔP's :

ΔP	=	ADJUSTED DP	PROCEDURE NAME	BEGINNING OFFSET
1534	-	111	=	1645
3417				
1776				
1501				
2675				

'ROGRAM FILE P10P033C.HP32033.SUPPORT

:EG' 0

NAME	STT	CODE	ENTRY	SEG
ININ	1	0	0	
TERMINATE'	47			?
CALLHELP	2	111	111	
HELP	50			?
POWERON	3	141	141	
GET'DSDEVICE	51			?
DEQUEUE	52			?
AWAKE	53			?
MASTERCLEARHPIB	54			?
CHECKLDEV	55			?
AWAKEIO	56			?
DATAABSENCE	4	450	450	
SUDDENDEATH	57			?
ABORT	60			?
RECOVEROC	61			?
QUEUEONSEGMENT	62			?
STTUNCALLABLE	5	663	663	
TRACE	6	672	672	
CY'	63			?
CODEABSENCE	7	1034	1034	
BUILDSEGID	64			?
CONVSEGIDTOSTIN	65			?
USERTRAP	10	1532	1532	
PRIVILEGEDMODEV	11	1610	1610	
STACKUNDERFLOW	12	1617	1617	
STUNSIM	66			?
STACKOVERFLOW	13	1730	1730	
MMSTAT	67			?
GETDATASEGCHANG	70			?
GENSPECREQ	71			?
SENDMSG	72			?
DSTVIOLATION	14	2674	2674	
CSTVIOLATION	15	2677	2677	
STTVIOLATION	16	2706	2706	
UNIMPLEMENTEDIN	17	2715	2731	
EADD	73			?
ESUB	74			?
EMPY	75			?
EDIV	76			?
ENEG	77			?
ECMP	100			?
QADD	101			?
QSUB	102			?
QMPY	103			?
QDIV	104			?
QNEG	105			?
QCMP	106			?
QASR	107			?
QASL	110			?
DIDIV	111			?
DIMPY	112			?
DMUL	113			?
CVAD	114			?
CVDA	115			?
CVBD	116			?

```
    CVDB                117                     ?
    SLD                 120                     ?
    NSLD                121                     ?
    SRD                 122                     ?
    ADDD                123                     ?
    CMPD                124                     ?
    SUBD                125                     ?
    MPYDSIM             126                     ?
    SEG'ID'TYPE         127                     ?
    TESTSTOP            130                     ?
    DEBUG               131                     ?
    BREAKPOINT           20   3223   3223
    SYSTEMCLOCK          21   3225   3225
    TICK                132                     ?
    POWERFAIL            22   3246   3246
    EXTGHOST             23   3344   3344
    GHOST                24   3347   3347
    GHOST36              25   3347   3416
    GHOST30              26   3347   3414
    GHOST29              27   3347   3412
    GHOST28              30   3347   3410
    GHOST27              31   3347   3406
    GHOST26              32   3347   3404
    GHOST23              33   3347   3402
    GHOST22              34   3347   3400
    GHOST15              35   3347   3376
    GHOST14              36   3347   3374
    GHOST10              37   3347   3372
    GHOST7               40   3347   3370
    GHOST5               41   3347   3366
    GHOST4               42   3347   3364
    DATAPARITY           43   3421   3436
    DCONVERT            133                     ?
    BCONVERT            134                     ?
    WRITE2              135                     ?
    ILLEGALADDRESS       44   3517   3517
    BOUNDSVIOLATION      45   3531   3531
    TESTCRUNCH           46   3543   3543
    SEGMENT LENGTH            4134

PRIMARY DB            0     INITIAL STACK     2260     CAPABILITY         600
SECONDARY DB          0     INITIAL DL           0     TOTAL CODE        4134
TOTAL DB              0     MAXIMUM DATA         ?     TOTAL RECORDS       26
ELAPSED TIME    00:00:06.552                          PROCESSOR TIME    00:01.182
```

THE INTERRUPT CONTROL STACK (ICS)

* IT IS THE ONLY MEMORY RESIDENT DATA STACK FOR MPE.

* IT'S LOCATION IS DEFINED BY FIXED LOW MEMORY LOCATIONS:

  QI=@5

  ZI=@6

* IT IS MEMORY RESIDENT BY VIRTUE OF THE FACT THAT IT RESIDES IN UNLINKED MEMORY.

* IT IS ONLY USED WHEN THE MICROCODE POSITIONS THE STACK DELIMIT REGISTERS TO IT.

* IT IS USED TO HANDLE SOME INTERNAL INTERRUPTS, ALL EXTERNAL INTERRUPTS AND BY THE DISPATCHER.

* [illegible] BY DST 7.

ICS FORMAT

QI		
-18	DISAP	PSEB,PSDB COUNTER
-17	XXXX	UNUSED
-16	SDST	PROCESS STACK DST
-15	PSTA	PSEUDO-INTERRUPT STATUS
-14	PADDR	PSEUDO-INTERRUPT ADDRESS
-13	TRACE FLAG	FLAG SET NON-ZERO IN EXIT FROM ICS
-12	PFAIL	POINTER TO PFAIL PCB
-11	JCUT	ABSOLUTE JCUT ADDRESS
-10	XP	POINTER TO EXECUTING PROCESS PCB
-9	PCBX	ABSOLUTE EXECUTING STACK ADDRESS
-8	Z	STACK DB RELATIVE Z(REL TO QI-4)
-7	DL	STACK RELATIVE DL (REL TO QI-4)
-6	S	STACK DB RELATIVE S (REL TO QI-4)
-5	S BANK	(NOT NECESSARILY EQUAL TO RETURN DB BANK)
-4	STDB	ABL STACK DB (NOT NECESSARILY EQUAL TO
-3	X	!DISPATCHER RETURN DB)
-2	DELTA P	!STARTING
-1	STATUS	!STACK
QI	DELTA Q	!MARKER
+1	DB BANK	DISPATCHER RETURN DB BANK
+2	DB	DISPATCHER RETURN DB
+3	PARM	DEVNO IF EXT INT, ELSE PARM

*INITIAL SETS THE FOLLOWING AS DESCRIBED:

CPCB - ABL. ADDR 4 IS AN ABSOLUTE VERSION OF XP. IF CPCB IS ZERO, THEN THE ABOVE CELLS ARE INVALID. THIS WILL NEVER BE THE CASE WHEN A PROCESS IS RUNNING.

SDST - DST NUMBER FOR THE RUNNING PROCESS STACK.

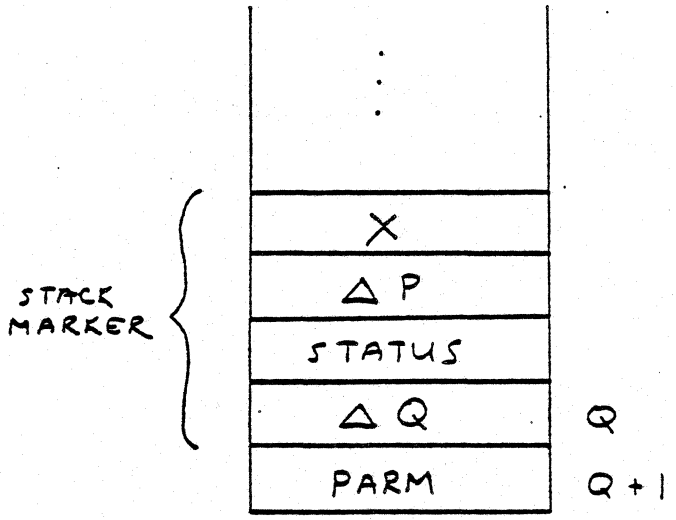
JCUT - THE BANK ZERO ABSOLUTE ADDRESS OF THE JCUT TABLE.

PADDR - PB RELATIVE ADDRESS FOR THE PROCEDURE PSEUDOINT.

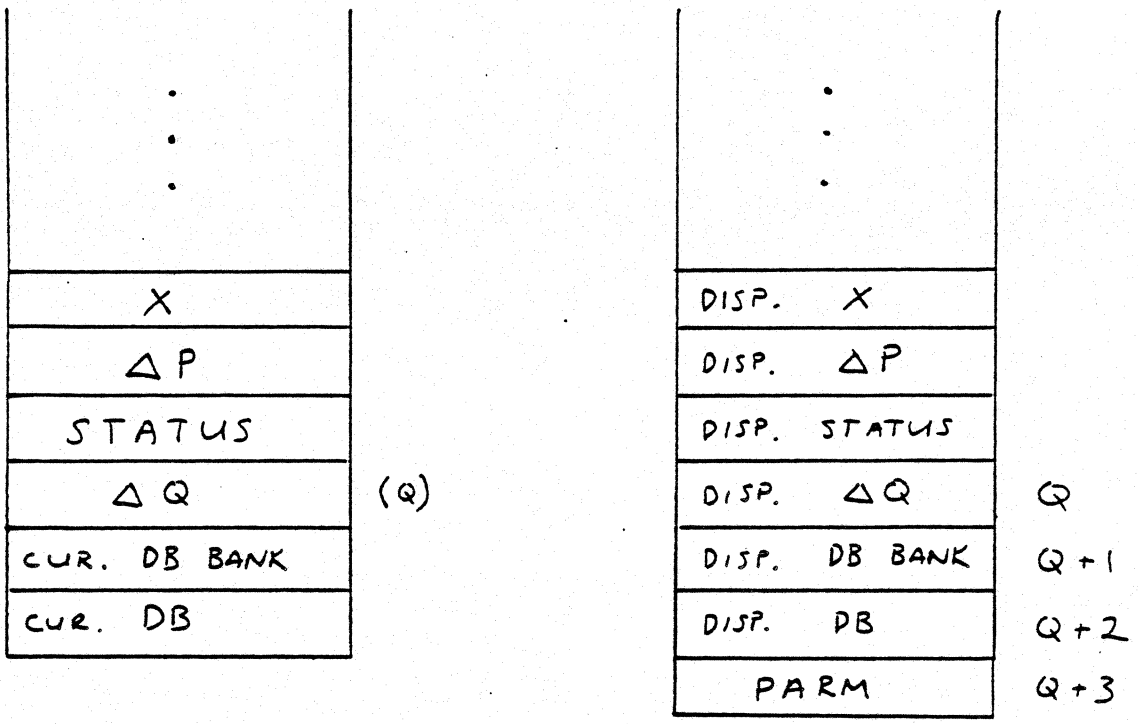
PSTA - STATUS VALUE FOR PSEUDOINT = %140000 + CST#.

DISAP - PSDB COUNTER, INITIALLY ZERO.

MOST INTERNAL INTERRUPTS ARE HANDLED ON THE USER'S STACK, FOLLOWING A REGULAR 4-WORD STACK MARKER. A PARAMETER MAY BE FOUND AT Q+1; THIS IS USUALLY THE PLABEL INTO ININ (SEE LIST OF INTERNAL INTERRUPTS).



SOME INTERNAL INTERRUPTS MUST BE HANDLED ON THE INTERRUPT CONTROL STACK (ICS). IF SO, A 6-WORD MARKER IS PUT ON THE USER'S STACK. THE PARM WILL BE FOUND ON THE ICS AT Q+3.



USER STACK

ICS

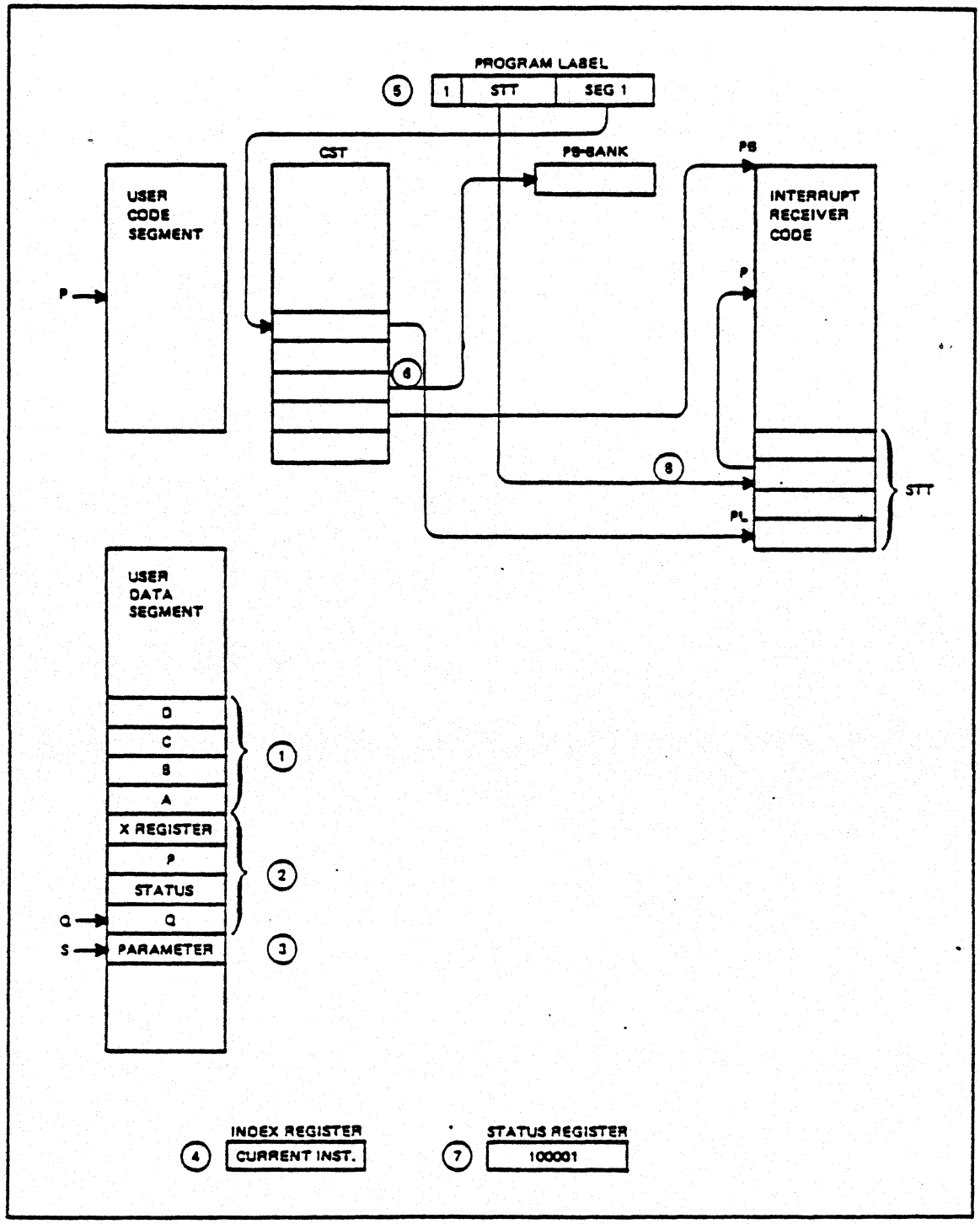
WHEN AN INTERNAL INTERRUPT OCCURS . . .

- ALL EXTERNAL INTERRUPTS ARE INITIALLY DISABLED; IE: INTERNAL INTERRUPTS HAVE HIGHER PRIORITY.

- ANY INTERNAL INTERRUPT MAY INTERRUPT THE PROCESSING OF ANOTHER.

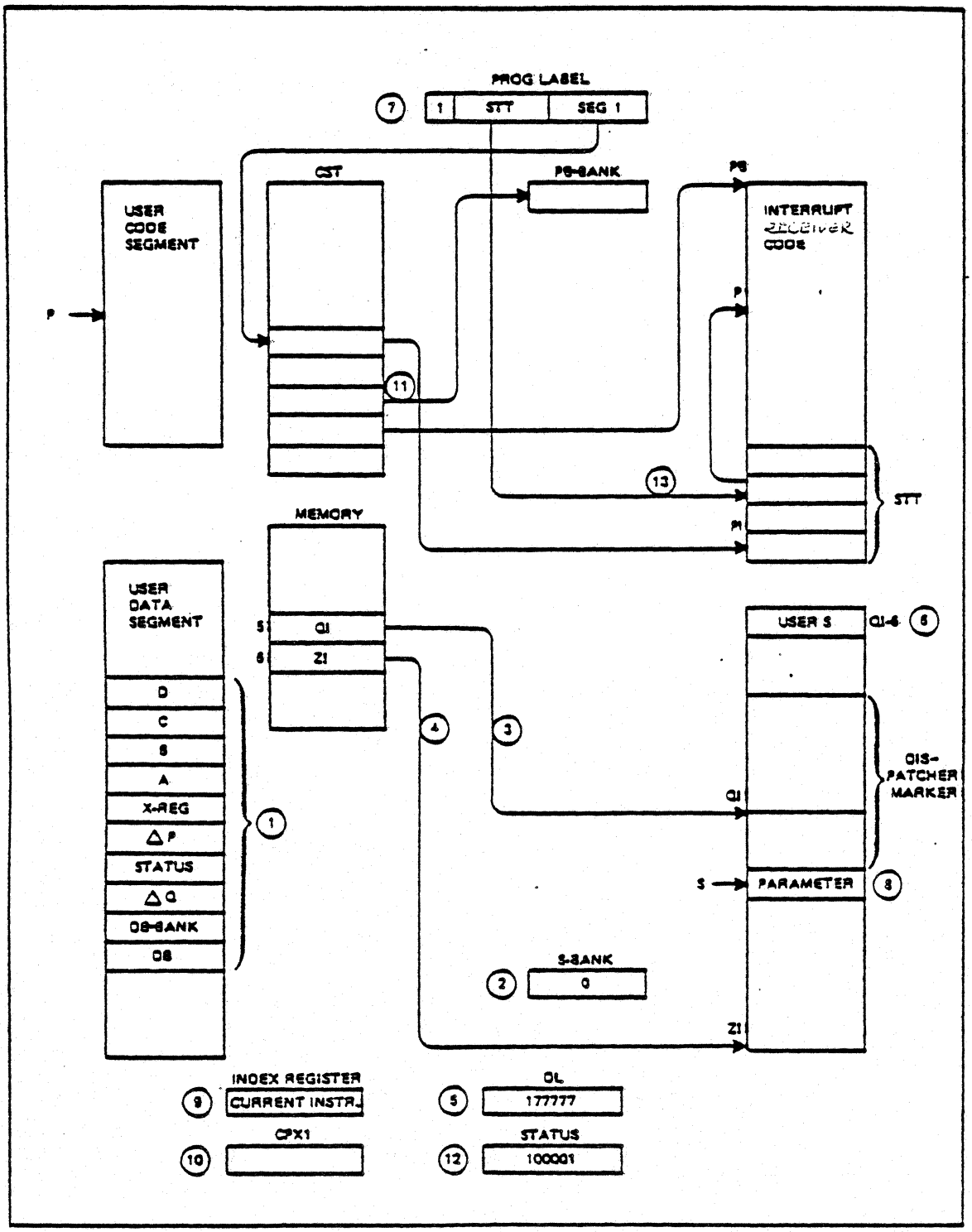
- AFTER PROCESSING IS COMPLETED RETURN TO THE INTERRUPTED PROCESS VIA THE "IXIT" INSTRUCTION.

Interrupt System



Non-ICS Type Internal Interrupts

Interrupt System

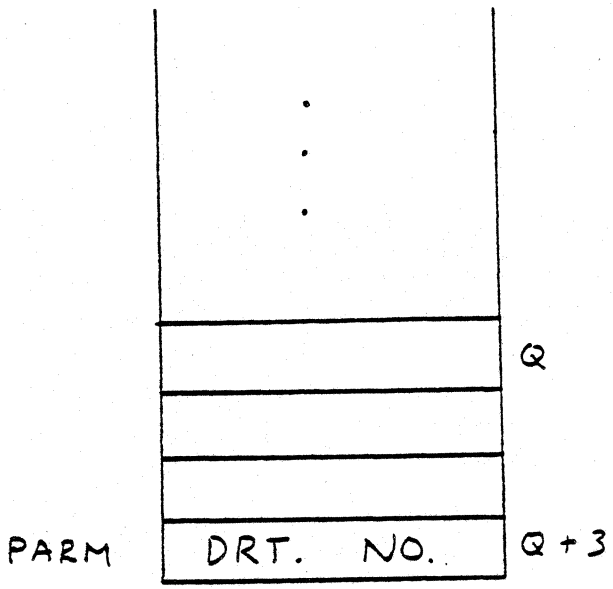


ICS-Type Internal Interrupt

EXTERNAL INTERRUPT CAUSES

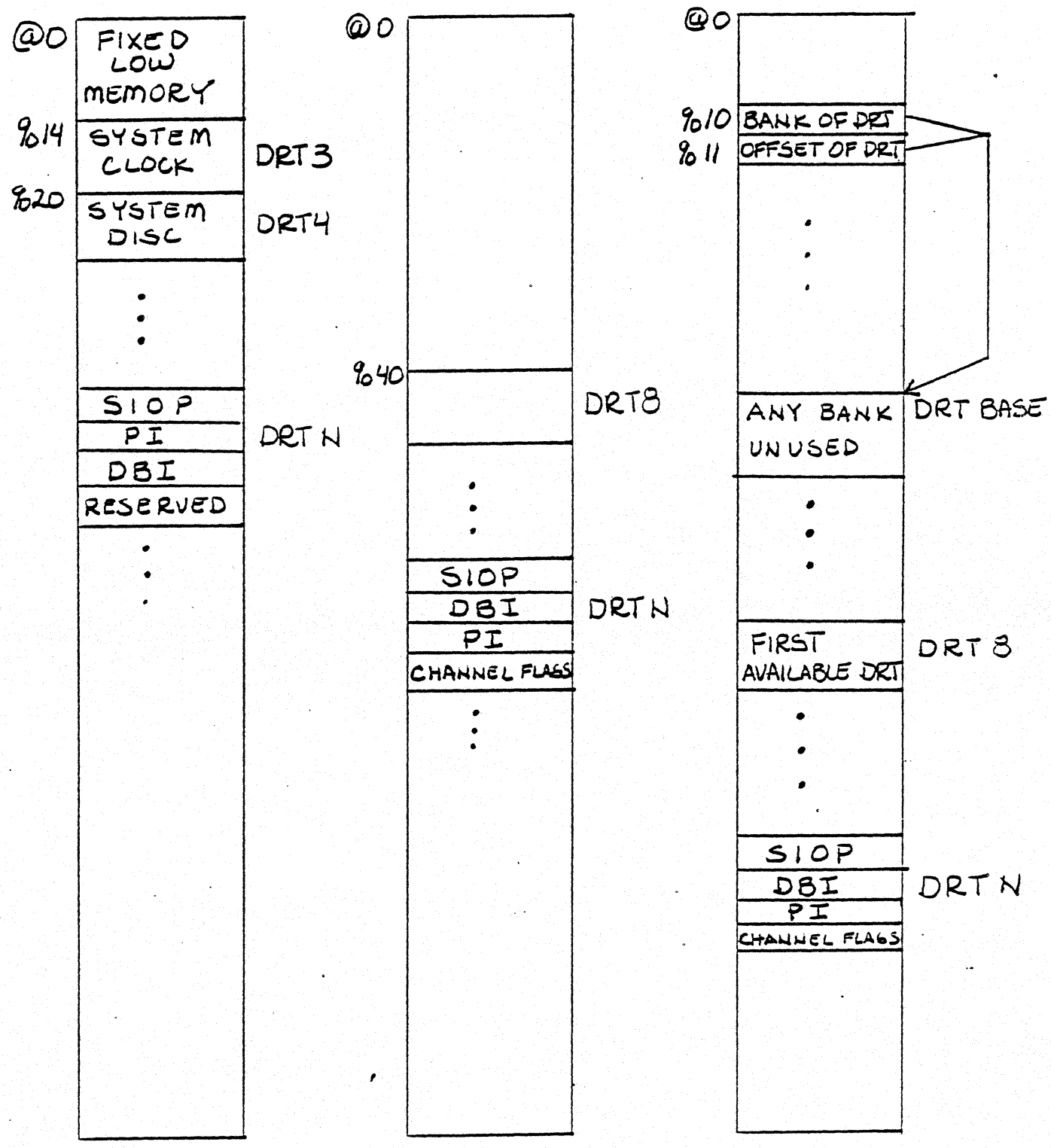
* EXTERNAL INTERRUPTS CAN BE CAUSED BY:
  1. HARDWARE - THE DEVICE CONTROLLER DETECTS A CONDITION THAT WAS DESIGNED INTO THE CONTROLLER WHICH CAUSES THE HARDWARE TO ISSUE AN INTERRUPT. THIS IS USUALLY AN ERROR CONDITION, BUT NOT ALWAYS (COMMUNICATIONS CONTROLLERS AND TERMINALS). MOST OF THE TIME AN INTERRUPT OCCURS WHEN AN I/O DEVICE GOES READY.
  2. SIO ORDER PAIRS
     A. INTERRUPT ORDER PAIR - USED MOSTLY BY COMMUNICATIONS CONTROLLERS.
     B. END WITH INTERRUPT - THE ORDER PAIR PROGRAMS DEVELOPED BY MPE ALWAYS END WITH INTERRUPT.
  3. CPU INSTRUCTION SIN - USED FOR STACK OVERFLOW WITH INTERRUPTS DISABLED.

WHEN THE EXTERNAL INTERRUPT OCCURS, THE CPU KNOWS WHICH DEVICE CONTROLLER INTERRUPTED. THE DRT NO. IS THE PARAMETER AT Q+3 ON THE ICS.



ICS

DEVICE REFERENCE TABLE LAYOUT

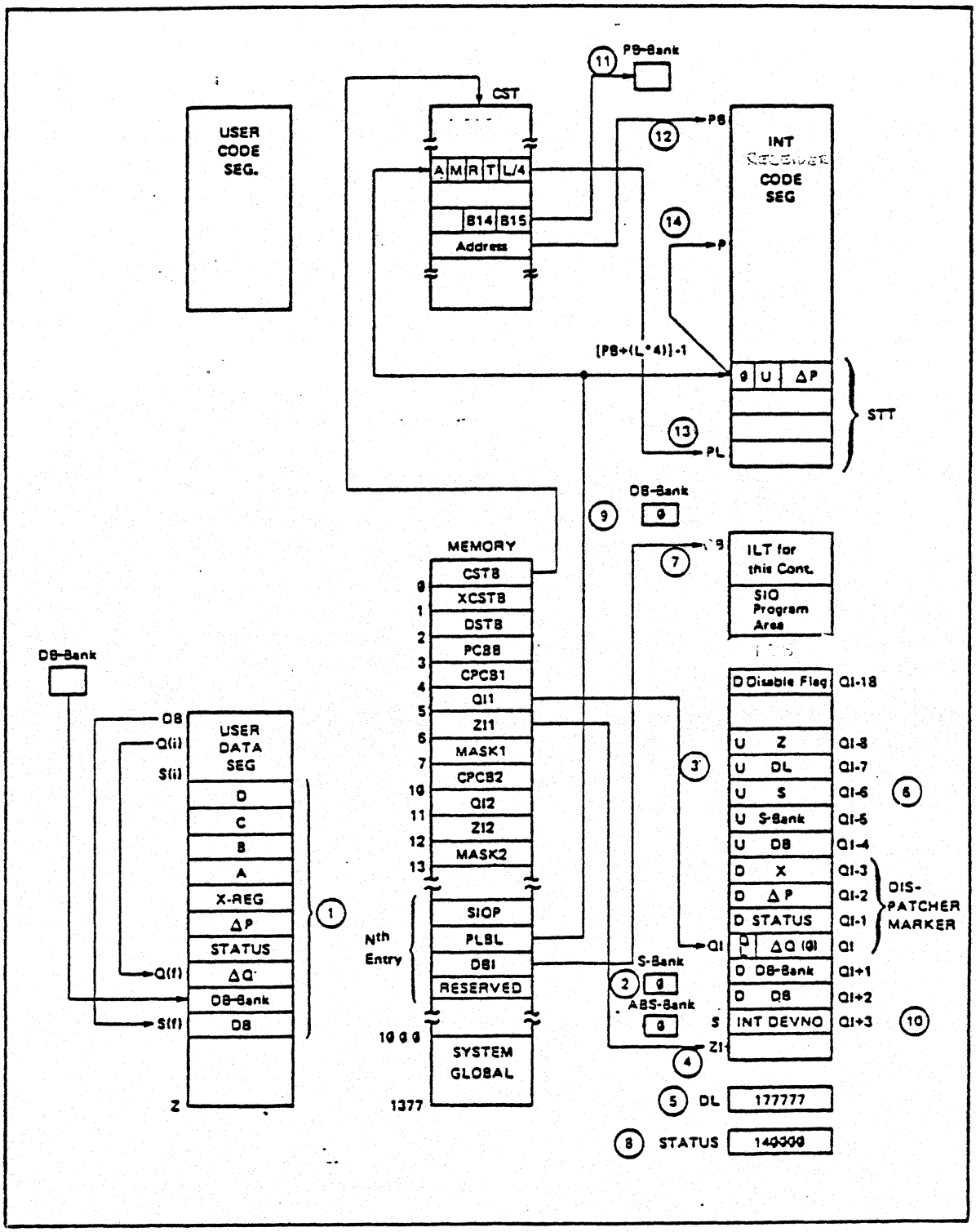


WHEN AN EXTERNAL INTERRUPT OCCURS . . .

- ONLY A HIGHER PRIORITY INTERRUPT CAN SUSPEND THE EXTERNAL INTERRUPT.

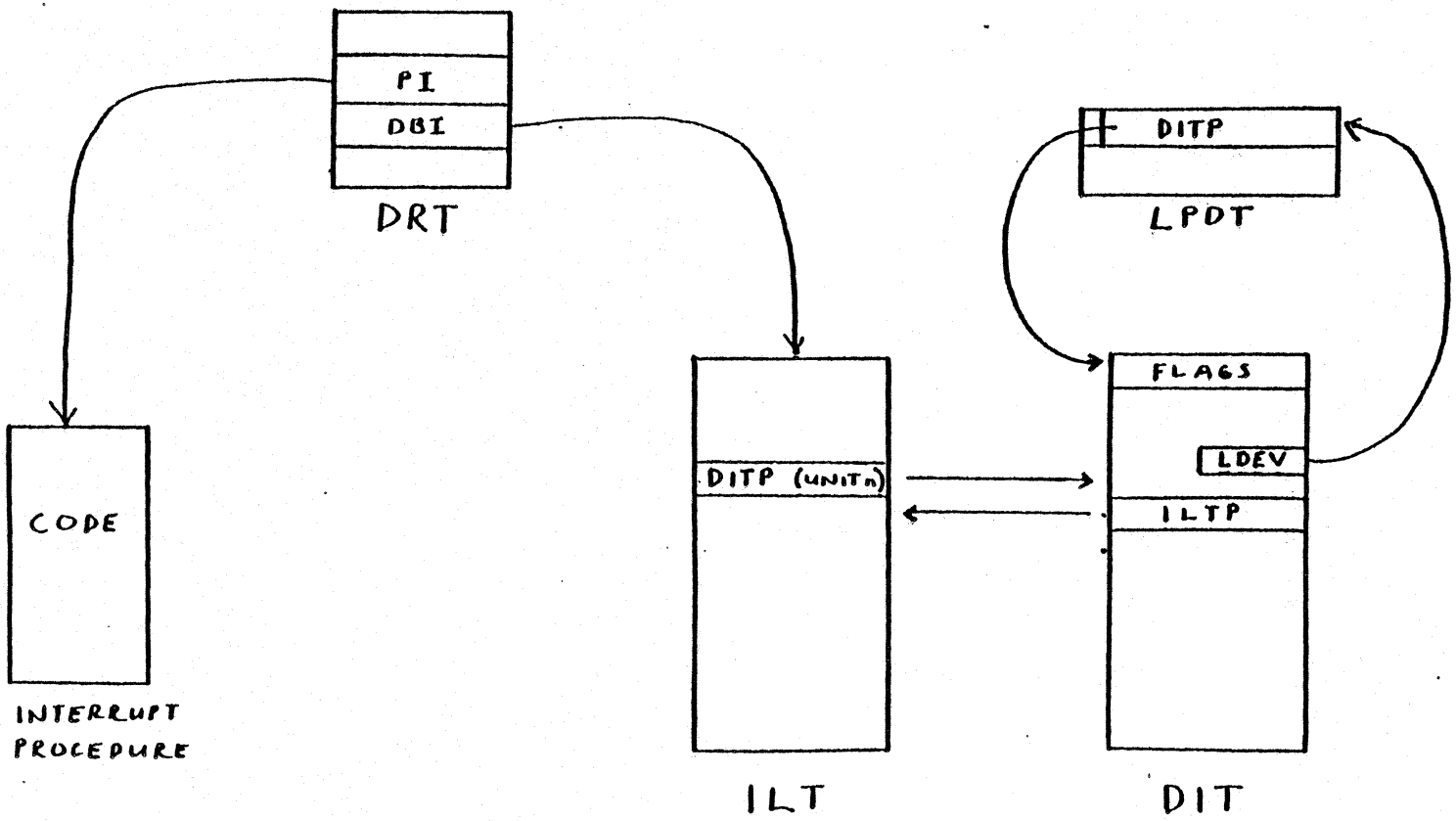
- AFTER PROCESSING IS COMPLETED, RETURN IS VIA THE "IXIT" INSTRUCTION (TO THE INTERRUPTED PROCESS) OR VIA "DISP" INSTRUCTION (TO THE DISPATCHER).

Interrupt System



First Level External Interrupt

EXTERNAL INTERRUPT TABLES



DEVICE REFERENCE TABLE (DRT)

* LOCATED IN FIXED LOW MEMORY STARTING AT ADDRESS %14.
* EACH ENTRY AT LOCATION 4*DRTNUM.
* ONE EACH 4 WORD ENTRY IS INITIATED FOR EACH DRT NUMBER LISTED IN THE I/O CONFIGURATION TABLE.
* THE SIZE OF THE TABLE IS CONFIGURED BY THE QUESTION "HIGHEST DRT#" IN SYSDUMP AND INITIAL.
* IF ALL OF THE AREA THAT COULD BE USED FOR THE DRT (UP TO ADDRESS %777) IS NOT USED THEN THAT AREA IS USED FOR OTHER SYSTEM TABLES.
* WORD 0 - SIOP - ABSOLUTE ADDRESS OF SIO PROGRAM.
* WORD 1 - PLABEL FOR THE INTERRUPT [illegible] FOR THIS DEVICE. INITIALLY SET BY INITIAL.
* WORD 2 - ILTP - THE ABSOLUTE ADDRESS OF WHERE THE ILT FOR THIS CONTROLLER RESIDES. THE VALUE FROM THIS LOCATION IS PUT IN DB BY THE MICROCODE HANDLING AN EXTERNAL INTERRUPT.
* WORD 3 - UNUSED BY MPE. USED BY INITIAL - .(0:8)= NUMBER INTERRUPT HANDLERS THIS DEVICE; .(8:8)= NUMBER UNITS THIS CONTROLLER.

INTERRUPT LINKAGE TABLE

* THERE IS ONE ILT FOR EACH CONTROLLER CONFIGURED IN THE SYSTEM.
* CONTAINS A UNIT EXTRACT INSTRUCTION TO BE USED TO EXTRACT THE UNIT NUMBER FROM THE STATUS RETURNED BY THE CONTROLLER AS THE RESULT OF A TIO INSTRUCTION. THE CONTENTS OF THIS MEMORY LOCATION IS PLACED ON TOS AND EXECUTED.
* CONTAINS ONE DIT POINTER FOR EACH UNIT CONFIGURED IN THE SYSTEM.
* CONTAINS AN SIO AREA WHERE THE ORDER PAIRS RESIDE WHILE THEY ARE ACTUALLY BEING EXECUTED BY THE CHANNEL.
* POINTED TO BY WORD 2 OF THE DRT ENTRY FOR THAT CONTROLLER AND WORD 5 OF THE DIT ENTRY FOR EACH UNIT ON THAT CONTROLLER.

DEVICE INFORMATION TABLE (DIT)

- ONE ENTRY FOR EACH UNIT CONFIGURED IN THE SYSTEM.
- FORMAT AND LENGTH IS DEVICE DEPENDENT AFTER WORD 7.
- GENERALLY DESIGNED TO CONTAIN INFORMATION THAT IS UNIQUE TO THAT TYPE OF UNIT (ESPECIALLY FOR MULTIUNIT CONTROLLERS) AND INFORMATION REGUARDING THE CURRENT STATE OF THAT UNIT.
- CONTAINS A POINTER (WORD 2) TO THE STRING OF IOQ'S LINKED TO THAT UNIT.
- DIT ENTRIES THEMSELVES ARE LINKED TO FORM REQUEST QUEUES FOR I/O RESOURCES (WORD 1).
- POINTED TO BY WORD (%16 + UNIT #) IN THE ILT, WORD 1 OF THE LPDT ENTRY AND IOQ(2).(8:8) OF ANY IOQ ENTRY LINKED TO THAT DIT (THROUGH THE LPDT).

LOGICAL TO PHYSICAL DEVICE TABLE (LPDT)

- THE TABLE IS POINTED TO BY DST %15 AND EACH ENTRY IS POINTED TO BY THE LOGICAL DEVICE NUMBER FROM THE I/O CONFIGURATION TABLE.
- EACH ENTRY IS TWO WORDS LONG.
- THE ADDRESS OF THE NTH ENTRY IS TABLE ADDRESS + (2 * N).
- THERE ARE TWO TYPES OF ENTRIES:
  - REAL ENTRIES (BIT 0 OF WORD 0 = 0) - THIS ENTRY IS ONE THAT REFLECTS INFORMATION ABOUT A DEVICE THAT IS LISTED IN THE I/O CONFIGURATION TABLE.
  - VIRTUAL ENTRIES (BIT 0 OF WORD 0 = 1) - THIS ENTRY IS ONE THAT EXISTS TO KEEP THE ABOVE LISTED FORMULA VALID WITHOUT HAVING TO PLACE THE REQUIREMENT ON THE USER THAT THERE CANNOT BE ANY UNUSED LDEV NUMBERS. THESE ENTRIES ARE USED BY SPOOLER TO HOLD DISC ADDRESSES OF ACTIVE INPUT STREAMED JOBS.
- EACH REAL ENTRY CONTAINS A POINTER TO THE DIT FOR THIS UNIT AND SYSTEM USAGE INFORMATION FROM THE I/O CONFIGURATION TABLE SUCH AS J, A, I, AND D PARAMETERS.
- IT IS NAMED THE LPDT BECAUSE IT MAPS A LOGICAL DEVICE INTO A PHYSICAL DEVICE THROUGH THE DIT FOR THE DEVICE.

HEWLETT PACKARD

EXTERNAL INTERRUPT PROCEDURES

SIO DEVICES	-	GIP (GENERAL INTERRUPT PROCEDURE)
TERMINALS	-	TIP (TERMINAL INTERRUPT PROCEDURE)
CLOCK BOARD	-	TICK (DB POINTS TO TRL)
CS DEVICES	-	VARIOUS ROUTINES

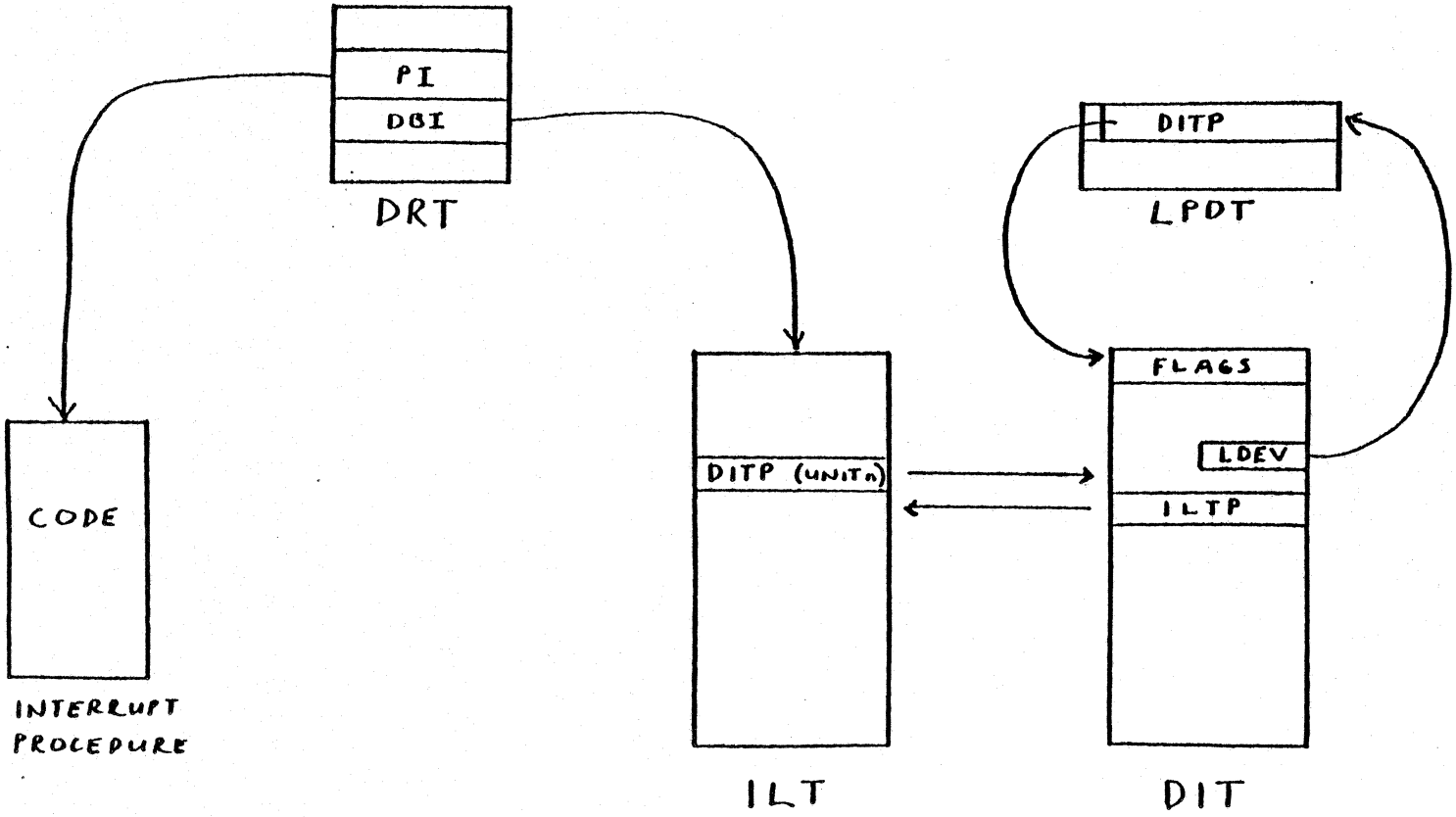
HEWLETT PACKARD

USUALLY, EXTERNAL INTERRUPTS SIGNAL SOME I/O COMPLETION.

ATTENTION INTERRUPTS RESULT IN THE AWAKENING OF DEVREC AND SETTING LPDT.

CLOCK BOARD INTERRUPTS RESULT IN A DISP.

EXTERNAL INTERRUPT TABLES



HEWLETT PACKARD

THE DANGERS OF DISABLING INTERRUPTS:

1. IF A STACK OVERFLOWS WHILE PSEUDO DISABLED, YOU HAVE 128 WORDS GRACE. SHOULD YOU OVERWRITE THAT, THE SYSTEM WILL CRASH.

2. IF MULTIPLE INTERRUPTS OCCUR WHILE DISABLED, IT IS POSSIBLE TO LOSE AN INTERRUPT. THEREFORE, DISABLE FOR AS SHORT A TIME AS POSSIBLE.

HEWLETT PACKARD

DISPATCHER

WHEN NO PROCESS IS RUNNING, YOU ARE EITHER EXECUTING AN INTERRUPT HANDLER OR DISPATCHER.

- EXECUTES ON THE ICS.
- SCHEDULES PROCESSES IN CO, DQ & EQ.
- SELECTS NEXT PROCESS TO RUN.
- KEEPS TRACK OF CPU TIME PER PROCESS.

HOW DO YOU GET INTO THE DISPATCHER?

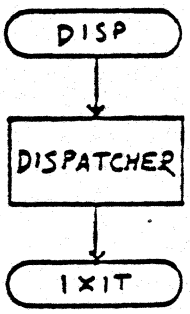
- THROUGH THE DISP INSTRUCTION.

EXECUTED:

- AT THE END OF SOME INTERNAL INTERRUPTS (eg. ABSENCE TRAPS).
- FOR UNEXPECTED INTERRUPTS (eg. LINE PRINTER NOT READY).
- AT THE END OF WAIT PROCEDURE, CALLED BY A PROCESS WHENEVER MPE WANTS TO SUSPEND IT (eg. I/O, PAUSE, etc.).
- FROM PROCEDURE TICK.

HOW DO YOU GET OUT OF THE DISPATCHER?

- DISPATCHER DOES AN IXIT.



HEWLETT PACKARD

WHAT DISP INST. DOES NORMALLY (UNLESS PSEUDODISABLED OR ON THE ICS)

- LAYS DOWN A 6-WORD MARKER ON THE EXECUTING PROCESS' STACK.
- SETS THE DISP & ICS FLAGS IN CPX1 REGISTER.
- CAUSES IMMEDIATE TRANSFER TO QI ON THE ICS AND DISPATCHER CODE.

HEWLETT PACKARD

WHAT HAPPENS IF DISP IS DONE WHILE PSEUDODISABLED OR ON THE ICS?

- SET QI(0:1) = 1 (DISP REQUEST FLAG).
- CONTINUE PROCESSING.

WHAT HAPPENS IF DISPATCHER GETS AN INTERRUPT?

- SET Q(0:1) = 1 (DISPATCHER INTERRUPT FLAG).
- HANDLE THE INTERRUPT.

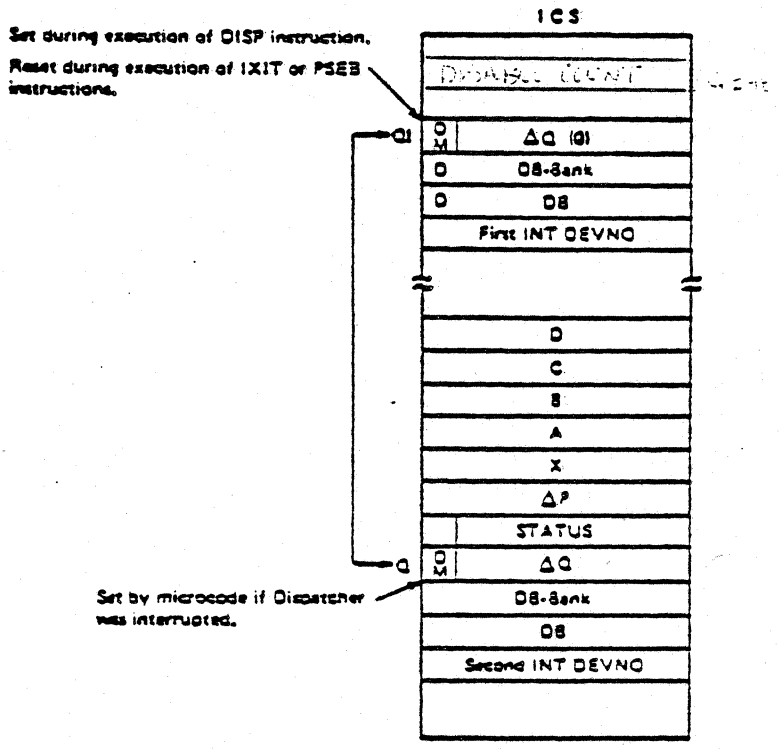
PSDB / PSEB INSTRUCTIONS

- PSDB PSEUDODISABLES THE DISPATCHER.
- PSDB INCREMENTS QI-18.

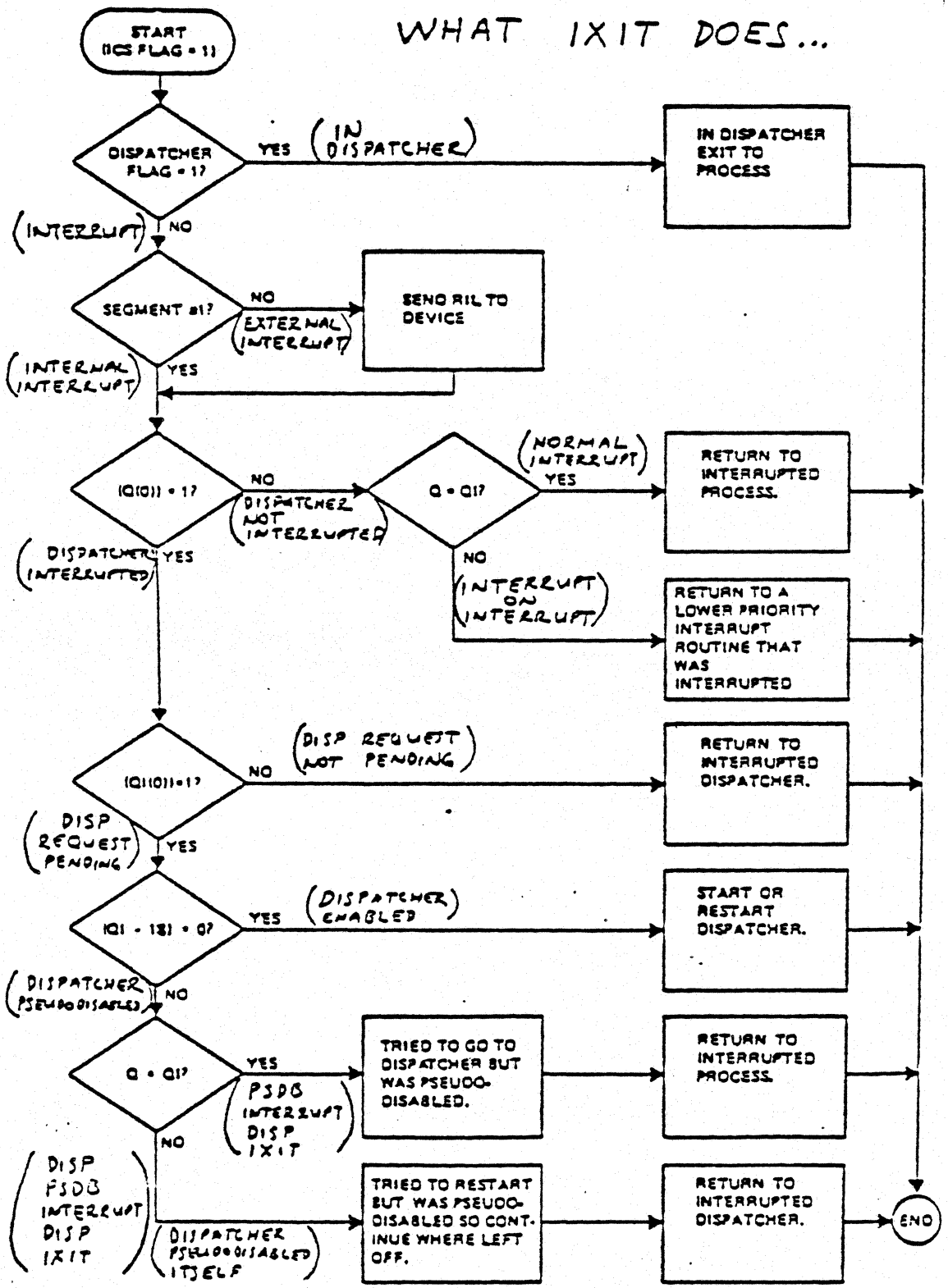
- PSEB DECREMENTS QI-18.
- DISPATCHER IS NOT ENABLED UNTIL QI-18 = 0.

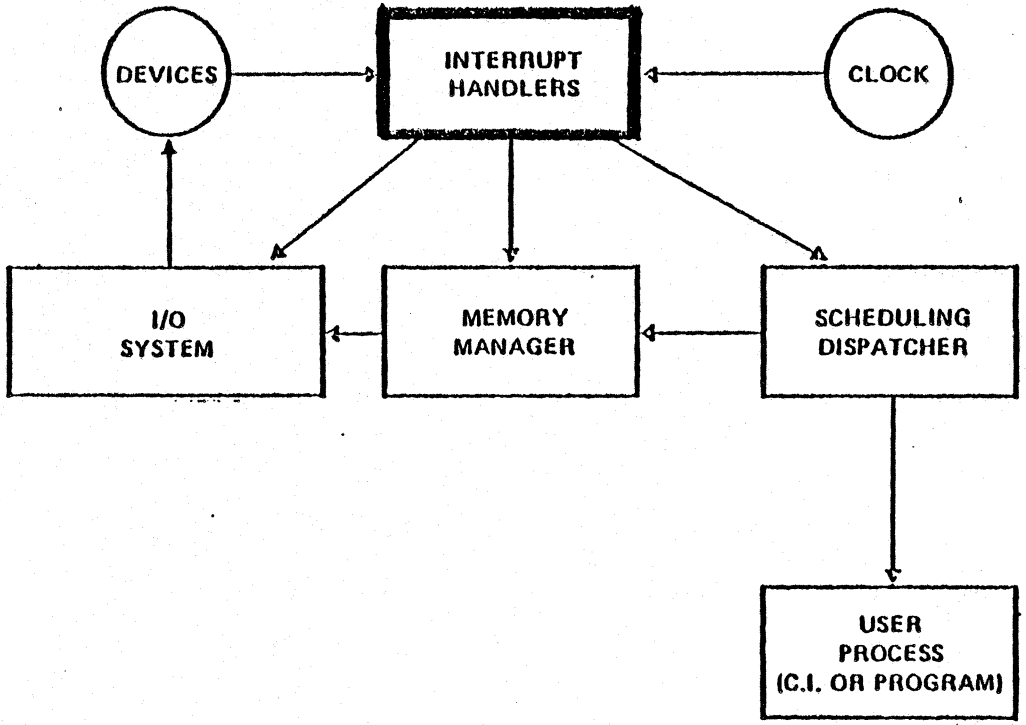
- IF NOT IN DISPATCHER AND
  QI(0:1) = 1 (DISP REQUEST PENDING),
  A PSEB WHICH SETS QI-18 = 0
  WILL EFFECT A DISP IMMEDIATELY.

- IF IN DISPATCHER,
  ANY PSEB WHICH SET QI-18 = 0
  WILL ALSO SET QI(0:1) = 0
  (ELIMINATE DISP REQUEST SINCE YOU ARE
  ALREADY IN THE DISPATCHER).


ICS
Set during execution of DISP instruction.
Reset during execution of IXIT or PSEB instructions.
DISPATCH COUNT
Q1
O M
ΔQ (0)
D
DB-Bank
D
DB
First INT DEVNO
D
C
B
A
X
ΔP
STATUS
Q
O M
ΔQ
Set by microcode if Dispatcher was interrupted.
DB-Bank
DB
Second INT DEVNO

WHAT IXIT DOES...




DEVICES
INTERRUPT HANDLERS
CLOCK
I/O SYSTEM
MEMORY MANAGER
SCHEDULING DISPATCHER
USER PROCESS (C.I. OR PROGRAM)

LAB

USING THE FOLLOWING PAGES FROM A MEMORY DUMP, ANSWER THESE QUESTIONS:

1. WAS THE SYSTEM CURRENTLY ON THE ICS ?
   HOW CAN YOU TELL ?
2. LIST ALL OF THE STACK MARKERS THAT YOU CAN FIND ON THIS STACK, MOST RECENT TO LEAST RECENT.,

3. NEXT TO EACH OF THE ABOVE STACK MARKERS, WRITE THE NAME OF THE CODE SEGMENT AND PROCEDURE NAME.
4. IN THE DUMP, CIRCLE THE PARAMETER PASSED TO SUDDENDEATH.
5. WHAT DRT NO. INTERRUPTED ?
6. WHAT DRT NO. CAUSED THE FAILURE ?

****** REGISTERS ******

DATA SEGMENT	CODE SEGMENT	MISCELLANEOUS	STATUS = 102033	CPX2 = 000001
DB BANK = 0	PB = 040664	X = 000005	MODE = PRIV	RUN/HALT = RUN	EXEC SW = OFF
DB = 001000	P = 063003	CTR = 030377	INTERRUPTS = OFF	SYS DUMP = ON	INC ADDR = OFF
S BANK = 0	PL = 064067	LPX1 = 000021	TRAPS = OFF	COLD LOAD = OFF	DEC ADDR = OFF
DL = 177777	PBBANK= 0	SP1 = 000033	STACK OP = LEFT	LOAD REG = OFF	INHIBIT
					AUTO RES = OFF
Q = 017566	(P-PB)= 022117	SP2 = 000000	OVERFLOW = OFF	LOAD ADDR = OFF	
S = 017570			CARRY = ON	LOAD MEM = OFF	
Z = 020206			COND CODE = CCG	DISP MEM = OFF	
Z BANK = 0			SEGMENT # = 33	SNGL INST = OFF	

********* HALT 17

****SYSTEM FAILURE # 201; STATUS 100033; DELTA P 022323 (I/O SYSTEM)

****** FIXED LOW MEMORY ******

CODE SEGMENT TABLE POINTER	010750
EXTENDED CODE SEGMENT TABLE POINTER	012424
DATA SEGMENT TABLE POINTER	007310
PROCESS CONTROL BLOCK BASE	014010
CURRENT PCB POINTER	000000
INTERRUPT STACK BASE	017210
INTERRUPT STACK LIMIT	020206
INTERRUPT MASK	000000

```
                                ******    PROCESS CONTROL BLOCK (1ST HALF)    ******


                                                         W A I T   S T A T E
         DATA
      -SEGMENTS--  -FAMILY TREF--  -----------WAKEMASK------------  ----------EVENTFLAGS----------- -PSEUDO INTERRUPTS--   --MISC---

                O                                  T     F     T                    T     F     T                     R P
                V                              U J T     A     M                U J T     A     M                     I I        C H
          A     A                          B   C U M M S T I S O M          B   C U M M S T I S O M                   T O        R S
          D     L  FTHR SON BRO O    R R M I I O N E S O H M I U E    R R M I I O N E S O H M T U E       H S S H C B B V        I I
PIN   XDS B SIK C  PIN  PIN PIN A  M G L A O O P K R G N R P R T M  M G L A O O P K R G N R P R T M  PSIM K K T H Y K K R  PTYPE T R
---   --- - --- -  ---- --- --- -  - - - - - - - - - - - - - - - -  - - - - - - - - - - - - - - - -  ---- - - - - - - - -  ----- - -

  1         106           2     S                J                                                   NORM                  SYST
  2       A  75       1       3                  J                                                   NORM                  SYST
  3       A  76       1       4                  J                                                   NORM                  SYST
  4          77       1       5 F                J                                                   NORM                  SYST
  5         100       1       6 F                          I                                         NORM                  SYST  C
  6         101       1       7 F                J                                                   NORM                  SYST
  7    11   102       1  15  10 F                J                                                   NORM                  SYSTU
 10         103       1      11                  J                                                   NORM                  SYST
 11         104       1      12              1   J                                                   NORM                  SYST
 12         105       1      14                  J               M                                   NORM                  SYST
 14       A 113       1         S          B                                                         NORM                  SYST  C
 15         114       7         F                                M                                   NORM                  UMAIN


                                ******    PROCESS CONTROL BLOCK (2ND HALF)    ******


      ----- S C H E D U L I N G   I N F O R M A T I O N -----   ---RESOURCES---  LIFE/  --------------- MISCELLANEOUS ---------------
                                                                                 DEATH
                      D         I C     H U           I H
                      T         N O     I S           P S       C H              L D
                      S         T R     P E T         F P S S   R S PREV NEXT    I E F                                         SYSTEM
                      P L C D F F F     R D R S L M P X R A U   T T IMPD IMPD S  V A A                                    BPT  PROC
PIN    NOPIN  PUPIN   Q Q Q Q Q R R PRI I Q W W W P C P I R V   T R PIN  PIN  C  E D C  BMS PPC    PCST     PBXPTR SLLPTR LNK  NAME
---    -----  -----   - - - - - - - --- - - - - - - - - - - -   - - ---- ---- -  - - -  --- --- ----------- ------ ------ --- ------

  1                     L            61                                          L      SNF NUL CIX  10.001     10  26617     PROGEN
  2                     L            62                                          L      SNF NUL                     26465     SYSIO
  3                     L           175                                          L      SNF NUL CST      37         26477     IOMESS
  4                     L            62                                          L      SNF NUL                  1  26511     LOG
  5                     L           175                         C                L      SNF NUL CIX   2.001      2  26523     MEMLOG
  6                     L           175                                          L      SNF NUL CIX   3.001      3  26535
  7                     L           175                                          L      SNF NUL CIX   4.001      4  26547     UCOP
 10                     L            12                                          L      SNF NUL                  5  26561     PFAIL
 11                     L           175                                          L      SNF NUL CIX   6.001      6  26573     DEVREC
 12                     L           216                   S                      L      SNF NUL                  7  26605     LOAD
 14                     L           230                         C                L      SNF NUL CST      54         27350
 15                   D   C     1   242   U   S                               S  L      SNF NUL CST      15         27576

   144  ENTRYS
```

```
017030: 100000 003040 003000 000000 000000 000000 000000 000000  017040: 000000 000000 000000 000000 000000 000000 000000 177777
017050: 100000 003060 003020 000000 000000 000000 000000 000000  017060: 000000 000000 000000 000000 000000 000000 000000 177777
017070: 100000 000200 003040 000000 000000 000000 000000 000000  017100: 000000 000000 000000 000000 000000 000000 000000 177777


$$$$$$$$ DSI   7 (INTERRUPT CONTROL STACK) $$$$$$$$
017110: 000000 000000 000000 000000 000000 000000 000000 000000  017120: 000000 000000 000000 000000 000000 000000 000000 000015
017130: 000001 000000 000000 100000 001750 001750 000143 000144  017140: 000347 000454 000000 000360 000312 000230 000375 000356
017150: 000310 000000 000000 000000 000000 000000 000000 000000  017160: 000000 000000 000000 000000 000000 000000 000002 000000
017170: 000114 100076 000131 177777 000000 030374 000320 046023  017200: 003724 000000 000001 046667 000000 001517 100074 000000
017210: 000000 000000 001000 000003 013330 000770 000245 000430  017220: 013330 000000 013330 101033 000001 000012 015374 000000
017230: 000000 000001 046223 000470 000460 000440 000004 000600  017240: 000444 013330 040015 000400 000000 000010 177777 001075
017250: 071274 001075 071274 000000 000764 000242 024040 000603  017260: 000000 000157 000000 055033 000000 055212 000576 100433
017270: 000031 000014 001262 002410 141074 100005 000000 001000  017300: 000004 100000 000000 023360 023400 002250 000004 002250

017310: 000000 000007 020463 142033 000011 177174 000000 002250  017320: 000000 000001 020022 140433 000010 000001 042200 000026
017330: 000000 000001 172425 000000 177174 023360 000000 000000  017340: 023020 000026 030507 000000 000000 023020 000006 001732
017350: 140033 000025 000000 155423 000161 002344 000000 000000  017360: 000000 021000 000000 155405 004000 000000 155423 040161
017370: 002344 010654 012473 102473 000014 102033 000020 017400: 017400: 000000 155405 000000 155423 040161 000000 010654 012027
017410: 100074 000015 000000 032316 000000 000000 155400 000000  017420: 000000 155410 000006 040161 000000 032316 014132 101074
017430: 000017 003510 003510 000006 003510 000000 177224 014225  017440: 100074 000011 177224 000000 003510 000000 000001 022022
017450: 100433 000010 000161 071353 177777 023223 101074 000013  017460: 000000 177224 024200 000000 000002 017643 000000 023360
017470: 000000 002250 000002 017643 003510 000000 172300 024217  017500: 000016 000002 001141 140433 000033 000001 000201 000001

017510: 000000 000053 177741 177777 025217 000002 000000 000000  017520: 003510 000000 003510 017643 000016 177777 000053 000742
017530: 140561 000025 000000 000002 000035 000000 003510 000016  017540: 000000 000016 000006 002245 140161 000014 003510 000016
017550: 000000 000016 003510 024207 022242 140033 000011 000006  017560: 000000 001000 000311 000005 022323 100033 000010 000000
017570: 001000 000412 000005 022115 102033 000007 000412 000005  017600: 022412 102033 000005 002370 073000 000007 000001 000007
017610: 000012 022520 100033 000011 100033 000011 102433 000031  017620: 000000 000000 000303 000000 016623 100000 177623 020240
017630: 020241 000001 002100 177777 000004 000043 000011 000001  017640: 000000 000000 023360 000000 002250 023400 177777 000001
017650: 002345 141155 000031 023360 000004 000004 024400 000000  017660: 000004 000007 021230 100033 000012 100033 000012 000012
017670: 000001 000000 001000 102074 000000 000000 000000 000000  017700: 000000 000000 000000 000000 000000 000000 000000 000000

017710: 000000 000000 000000 000000 000000 000000 000000 000000  017720: 000000 000000 000000 000000 000000 000000 000000 000000
LINES 017730 - 020167 SAME AS ABOVE

020170: 000000 000000 000000 000000 000000 000000 000000 000000  020200: 000000 000000 000000 000000 000000 000000 000000 000000


$$$$$$$$ DSI  13 (I/O OUEUE) $$$$$$$$
020210: 030052 000013 000440 000420 010401 000000 000000 000405  020220: 007000 000023 000006 000100 100113 002005 000001 000000
020230: 000220 000000 000001 007000 000036 000006 000100 100113  020240: 002033 000001 000052 000201 000000 000006 007000 000051
020250: 000000 000100 100113 002033 000001 000052 000201 000000  020260: 000001 007000 000004 000000 000100 100113 002033 000001
020270: 000052 000201 000000 000001 007000 000077 000000 000100  020300: 100113 002005 000001 000000 000220 000000 000001 007000
020310: 000012 000006 000100 100113 002046 000001 000037 000201  020320: 000000 000401 007000 000125 000006 000100 100113 002046
020330: 000001 000037 000201 000000 000001 007000 000140 000006  020340: 000100 100113 002046 000001 000037 000001 000000 000001
020350: 001000 000153 000000 000000 100113 002000 000004 000000  020360: 000000 000000 000001 005004 000227 000024 000000 100103
020370: 000002 000001 000001 000000 000000 000401 007000 000576  020400: 000024 000000 100114 000000 000034 000000 000000 000000

020410: 000401 007000 000255 000024 000000 100114 000033 000001  020420: 177770 000000 000004 000401 007000 000351 000024 000000
020430: 100114 001000 000001 177703 000000 000000 000401 007000  020440: 000100 000024 000043 100114 000001 000000 000000 000003
020450: 000000 000001 007000 000201 000024 000043 100114 000001  020460: 000000 177774 000003 000000 000401 007000 000270 000024
```

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISL ADDRESS	R O C	I M I	M O D	S Y S	C R E S
1	INTN	PRIV	ON	OFF	3530 ?	110760	0					S	C
2	FILESYS1 (0)	PRIV	OFF	OFF	10764 ?	161223	1					S	
3	FILESYS4 (1)	PRIV	OFF	OFF	3550 ?	155023	1					S	
4	FILESYS5 (2)	PRIV	ON	OFF	4224 ?	111623	1					S	
5	FILESYS6 (3)	PRIV	OFF	OFF	5144 ?	126223	0					S	
6	FILESYS6A (4)	PRIV	OFF	OFF	12164 ?	142423	1					S	
7	FILESYS7 (5)	PRIV	ON	OFF	6220 ?	063023	1					S	
10	CIALTORG (6)	PRIV	OFF	OFF	10224 ?			1 30277				S	
11	CICOMSYS (7)	PRIV	OFF	OFF	4220 ?			1 30347				S	
12	CIERR (10)	PRIV	OFF	OFF	2400 ?			1 30403				S	
13	CIFILEB (11)	PRIV	OFF	OFF	7704 ?			1 30420				S	
14	CIFILEM (12)	PRIV	OFF	OFF	3304 ?			1 30465				S	
15	CIINIT (13)	PRIV	ON	OFF	7244 ?	025223	1			I		S	
16	CILIBIF (14)	PRIV	ON	OFF	6404 ?	133423	0					S	
17	CIMISC (15)	PRIV	OFF	OFF	4504 ?			1 30626				S	
20	CIORGMAN (16)	PRIV	OFF	OFF	6310 ?			1 30654				S	
21	CIPREPRUN (17)	PRIV	OFF	OFF	5570 ?			1 30712				S	
22	CISUBS (20)	PRIV	OFF	OFF	3724 ?			1 30746				S	
23	CISYSMGR (21)	PRIV	OFF	OFF	7334 ?			1 30773				S	
24	CIUSERUTIL (22)	PRIV	OFF	OFF	4444 ?			1 31040				S	
25	CXSIORFST (23)	PRIV	OFF	OFF	5730 ?			1 31065				S	
26	RESIORF (24)	PRIV	OFF	OFF	5574 ?			1 31120				S	
27	SIORF (25)	PRIV	OFF	OFF	10210 ?			1 31155				S	
30	DIRC (26)	PRIV	OFF	OFF	7444 ?	053223	1					S	
31	ALLOCATE (27)	PRIV	OFF	OFF	6130 ?			1 31263				S	
32	ALLOCUTIL (30)	PRIV	OFF	OFF	7260 ?	015623	1					S	
33	HARDRES (31)	PRIV	ON	OFF	23204 ?	040664	0					S	C
34	ABORTDUMP (32)	PRIV	OFF	OFF	6514 ?			1 31504				S	
35	MESSAGE (33)	PRIV	OFF	OFF	4224 ?			1 31543				S	
36	PROCSEG (34)	PRIV	OFF	OFF	5330 ?	150023	0					S	
37	NRIO (35)	PRIV	OFF	OFF	2544 ?			1 31621				S	
40	PCREATE (36)	PRIV	OFF	OFF	10130 ?			1 31636				S	
41	MORGUE (37)	PRIV	OFF	OFF	4400 ?			1 31704				S	
42	BIPC (40)	PRIV	OFF	OFF	3334 ?			1 31734				S	
43	IPC (41)	PRIV	OFF	OFF	11174 ?			1 31754				S	
44	CHECKER (42)	PRIV	OFF	OFF	1764 ?	145623	0					S	
45	UTILITY1 (43)	PRIV	OFF	OFF	4544 ?	010423	1					S	
46	UTILITY2 (44)	PRIV	OFF	OFF	6650 ?			1 32064				S	
47	LOADFR1 (45)	PRIV	OFF	OFF	6030 ?			1 32122				S	
50	RINS (46)	PRIV	OFF	OFF	3644 ?			1 32156				S	
51	JURIABLE (47)	PRIV	OFF	OFF	5114 ?			1 32177				S	
52	DEBUG (50)	PRIV	OFF	OFF	20544 ?			1 32270				S	
53	NURSERY (51)	PRIV	OFF	OFF	7310 ?			1 32376				S	
54	SPOOLING (54)	PRIV	ON	OFF	15640 ?	117423	1					S	
55	SPOOLCOMS1 (55)	PRIV	OFF	OFF	6744 ?			1 32565				S	
56	SPOOLCOMS2 (56)	PRIV	OFF	OFF	12110 ?			1 32625				S	
57	PVCOMSEG (57)	PRIV	OFF	OFF	3174 ?			1 32700				S	
60	PVSYSD (60)	PRIV	OFF	OFF	5000 ?			1 32717				S	

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISC ADDRESS	R O C	I M I	M O D	S Y S	C K E S
61	PVSYSM (61)	PRIV	OFF	OFF	7200 ?		1	$2745				S	
62	UDC (62)	USER	OFF	OFF	7044 ?		1	$3005				S	
63	USER (63)	USER	OFF	OFF	3330 ?	160423	0					S	
64	HELPUSER (64)	USER	OFF	OFF	2410		1	$3067					
65	OPLOW (65)	PRIV	OFF	OFF	14020 ?		1	$3103				S	
66	OPMED (66)	PRIV	OFF	OFF	13540 ?		1	$3170				S	
67	OPHI (67)	PRIV	OFF	OFF	11360 ?		1	$3253				S	
70	LABSEG (70)	PRIV	OFF	OFF	13254 ?		1	$3324				S	
71	SDISC (71)	PRIV	OFF	OFF	11764 ?		1	$3404				S	
72	LOGSEG0 (73)	PRIV	OFF	OFF	12314 ?		1	$3465				S	
73	LOGSEG1 (74)	PRIV	OFF	OFF	13540 ?		1	$3542				S	
74	KERNELC (75)	PRIV	ON	OFF	23644 ?	064070	0					S	C
75	KERNELD (76)	PRIV	OFF	OFF	10264 ?	000023	1					S	
76	MISCSEGL (77)	PRIV	ON	OFF	1024 ?	107734	0					S	C
77	FILESYS1A (101)	PRIV	OFF	OFF	14734 ?		1	$4057				S	
100	FILESYS2 (102)	PRIV	OFF	OFF	7774 ?		1	$4147				S	
101	FILESYS3 (103)	PRIV	OFF	OFF	14354 ?		1	$4214				S	
102	DEBUGUTL (104)	PRIV	OFF	OFF	4364 ?		1	$4264				S	
103	SEGUTIL (105)	PRIV	OFF	OFF	4424 ?		1	$4307				S	
104	KSAM01 (106)	PRIV	OFF	OFF	6524 ?		1	$4333				S	
105	KSAM02 (107)	PRIV	OFF	OFF	10764 ?		1	$4370				S	
106	KSAM03 (110)	PRIV	OFF	OFF	7724 ?		1	$4437				S	
107	KSAM04 (111)	PRIV	OFF	OFF	7004 ?		1	$4501				S	
110	KSAM05 (112)	PRIV	OFF	OFF	3070 ?		1	$4537				S	
111	FIRMWARESIM1 (52)	PRIV	OFF	OFF	5000 ?		1	$2227				S	
112	FIRMWARESIM2 (53)	PRIV	OFF	OFF	6330 ?		1	$2437				S	
113	KSAM06 (113)	USER	OFF	OFF	2410 ?		1	$4556				S	
114	KSAM07 (114)	USER	OFF	OFF	5044 ?		1	$4574				S	
115	COMSYS1 (116)	PRIV	OFF	OFF	10510		1	$4646				S	
116	COMSYS3 (120)	PRIV	OFF	OFF	7274		1	$4760				S	
117	COMSYS4 (121)	PRIV	OFF	OFF	7034		1	$5022				S	
120	COMSYS5 (122)	PRIV	OFF	OFF	7504		1	$5064				S	
121	CSUTILTY (123)	PRIV	OFF	OFF	12010		1	$5126				S	
122	COMSYS2 (117)	PRIV	OFF	OFF	10274		1	$4713				S	
123	BSCLCM (124)	PRIV	OFF	OFF	4310		1	$5203				S	
124	BSCSLCPU (125)	USER	OFF	OFF	1354		1	$5230				S	
125	DVRSSLC (126)	PRIV	OFF	OFF	10500		1	$5240				S	
126	DVRHSI (127)	PRIV	OFF	OFF	2154		1	$5506				S	
127	DSSEG1 (151)	PRIV	OFF	OFF	4574		1	$6222				S	
130	DSSEG2 (152)	PRIV	OFF	OFF	11234		1	$6251				S	
131	DSSEG4 (154)	PRIV	OFF	OFF	7060		1	$6354				S	
132	DSMISC (156)	PRIV	OFF	OFF	6004		1	$6477				S	
133	DSIOM (157)	PRIV	OFF	OFF	1014		1	$6534				S	
134	DSSEG5 (153)	PRIV	OFF	OFF	5500		1	$6322				S	
135	DSSEG5 (155)	PRIV	OFF	OFF	12530		1	$6417				S	
136	CLIB'01 (204)	USER	OFF	OFF	6574		1	40222					
137	CLIB'03 (206)	USER	OFF	OFF	7260		1	40307					
140	CLIB'04 (207)	USER	OFF	OFF	6530		1	40347					

****** CST TABLE ******

SEGMENT NUMBER	SEGMENT NAME	MODE	REFERENCE BIT	TRACE	SEGMENT LENGTH	ABSOLUTE ADDRESS	BANK/ /LDEV	DISL ADDRESS	R U C	I M I	M O D	S Y S	C R E S
141	CLIB*05 (210)	USER	OFF	OFF	5454		1	40404					
142	DSRTFCALLS (100)	PRIV	OFF	OFF	7700		1	36550				S	
143	MRJEMISC1 (161)	PRIV	OFF	OFF	10710		1	36012				S	
144	MRJEMISC2 (162)	USER	OFF	OFF	6114		1	36061				S	
145	MRJESLCP (163)	USER	OFF	OFF	574		1	36715				S	
146	RSCSLCP1 (164)	USER	OFF	OFF	1374		1	36722				S	
147	MPMONCMD (165)	PRIV	OFF	OFF	3470		1	36731				S	
150	IMAGE01 (214)	PRIV	OFF	OFF	6354		1	40543					
151	IMAGE02 (215)	PRIV	OFF	OFF	6244		1	40600					
152	IOMONITOR3270 (231)	PRIV	OFF	OFF	7114		1	41402				S	
153	TRACE0* (232)	USER	OFF	OFF	6330		1	41442					
154	TRACE1* (233)	USER	OFF	OFF	6444		1	41476					
155	IOMDISC1	PRIV	ON	OFF	2720 ?	116050	0					S	C
156	IOTEPL0	PRIV	ON	OFF	5074	120770	0					S	C
157	IOTAPE0	PRIV	OFF	OFF	1620 ?		1	52705				S	
160	IOTERM0	PRIV	OFF	OFF	6060 ?		1	52743				S	
161	IOLPRT0	PRIV	ON	OFF	2730 ?	155423	0					S	

CHECK FOR UNDERSTANDING

1. Why is the Segment Transfer Table (STT) area of ININ different than indicated by the PMAP for ININ?

   ANS: _____

2. What is the ICS and what is it used for?

   ANS: _____

3. How can you tell if you are on the ICS?

   ANS: 1777>

4. What CPU instruction is used to leave the ICS?

   ANS: IXIT

5. On what stack(s) might an internal interrupt be serviced?

   ANS: ICS / User stack

6. On what stack(s) might an external interrupt be serviced?

   ANS: ICS

7. Where is the "parameter" placed for the following situations?

   A. Interrupt/Trap handled on the ICS?

      ANS: 973

   B. Interrupt/Trap handled on the user's stack?

      ANS: 971

8. On an external interrupt, what does the value of the PARM represent?

   ANS: dnt

9. Can you have an interrupt on top of another interrupt? If so, what kinds?

   ANS: Ext or Int, Int or Ext.

10. If QI(2:1) = 1 what does this indicate?

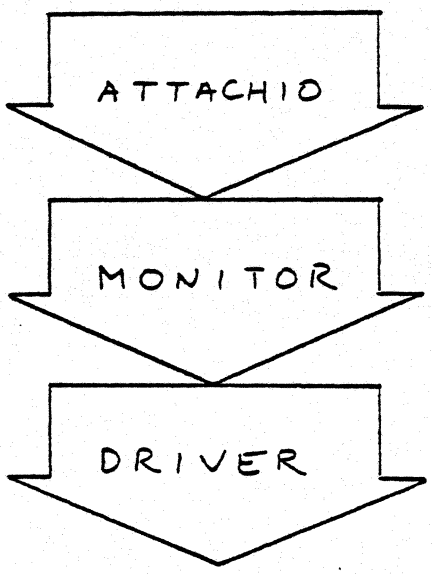
ANS: Dispatcher was pseudo disabled.

11. If QI (0:1)=1, Q=QI and the next CPU instruction to be executed is IXIT, what code segment will begin execution (i.e., Interrupt handler, interrupted user process, next CPU instruction after IXIT, ect.)? Note QI-18≠0.

ANS: Return to user process.

X	ΔP	STATUS	ΔQ	MODULE /PROCEDU
3	022323	100033	10	HARDRES/IOFAILURE
24207	022232	140033	11	HR / DOCIO
6	002245	140161	14	IOLPRTO/CHECK'STA
53	000742	140561	25	IOLPRTO/LPDRIVER
2	001141	140433	33	HR/ SIODM
1	022022	100433	10	HR/AWAKEIO
177224	014225	100074	11	KC/AWAKEDEVIC
?316	014132	101074	17	KC/UNDEFERSEGSM
10654	012627	100074	15	KC/PROCESSCOMPMS
10654	012473	102074	23	KC/SEG READ COMPLE
6	001732	140033	25	HR/SIODM
1	022022	140433	10	HR/AWAKEIO
7	020463	142033	17	HR/GIP
1262	002616	141074	100065	KC/DSP
00	001517	100074	00	KC/DSP

MPE I/O SYSTEM



SE 335 I/O SECTION

OBJECTIVES:

-LEARN THE BASIC 3 ELEMENTS OF THE I/O SYSTEM AND BE ABLE TO EXPLAIN HOW EACH PART WORKS TOGETHER TO HANDLE AN I/O REQUEST.

- BE ABLE TO DESCRIBE THE FLOW OF THE I/O SYSTEM THROUGH THE MPE PROCEDURES, FOR A TERMINAL REQUEST AND A SIO DEVICE REQUEST.

- DETERMINE THE CAUSE OF A SIMPLE I/O PROBLEM BY ANALYZING THE DUMP.

SIO - (PROGRAMMED I/O)

- USES A PROGRAM TO CONTROL THE DEVICE CONTROLLER.
- ONE PROGRAM FOR A COMPLETE TRANSFER OF MULTIPLE WORDS
- THE I/O PROGRAM REQUIRES A DIRECT INSTRUCTION TO BEGIN ITS EXECUTION (START I/O)
- USED FOR MOST DEVICES (DISC, TAPE...)

DIO - (DIRECT I/O)

- MULTIPLE NUMBER OF DIO INSTRUCTIONS FOR ONE COMPLETE TRANSFER OF MULTIPLE WORDS
- EACH INSTRUCTION CAN TRANSFER AT MOST ONE CHARACTER
- USED PRIMARILY BY TERMINALS
- EXAMPLES
  - READ I/O
  - WRITE I/O
  - TEST I/O

I/O

TERMINOLOGY:

IOQ - I/O QUEUE ENTRY. THIS TABLE ENTRY REPRESENTS A SINGLE NON-DISC I/O REQUEST.

DISC REQUEST QUEUE ENTRY - THIS TABLE ENTRY REPRESENTS A SINGLE DISC I/O REQUEST.

ATTACHIO - THE PROCEDURE WHICH CREATES AND LINKS AN IOQ or DRQ ENTRY FOR A FILE SYSTEM I/O REQUEST.

DRIVER - A PROGRAM WHICH DOES THE LOGICAL I/O.

DRIVER INITIATOR - SECTION OF THE DRIVER WHICH STARTS THE PHYSICAL I/O.

DRIVER COMPLETOR - SECTION OF THE DRIVER WHICH IS ACTIVATED BY THE INTERRUPT AT THE END OF THE PHYSICAL I/O.

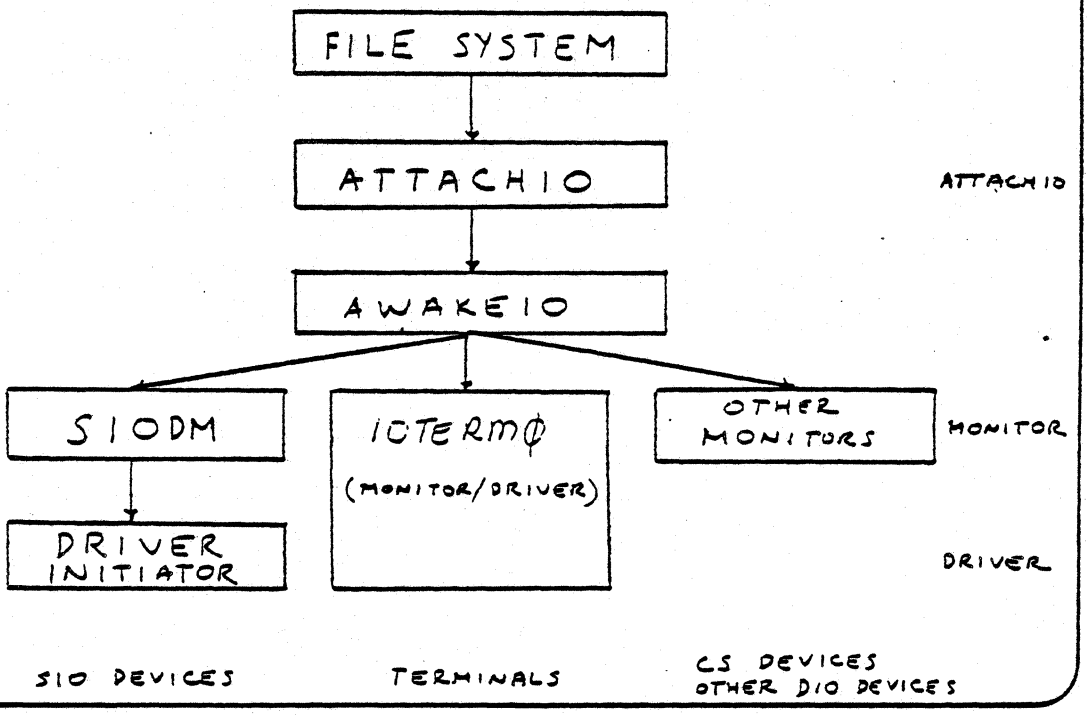
INTERRUPT PROCEDURE - I/O COMPLETION INTERRUPT SERVICE ROUTINE.

AWAKEIO - THE PROCEDURE THAT DETERMINES WHICH MONITOR TO CALL FOR THE PARTICULAR DEVICE AND THEN CALLS THE CORRECT MONITOR PROCEDURE.

I/O MONITOR - A PROGRAM THAT SELECTS WHICH IOQ OR DRQ WILL EXECUTE (ANALOGOUS TO DISPATCHER SELECTING WHICH PROCESS WILL EXECUTE).

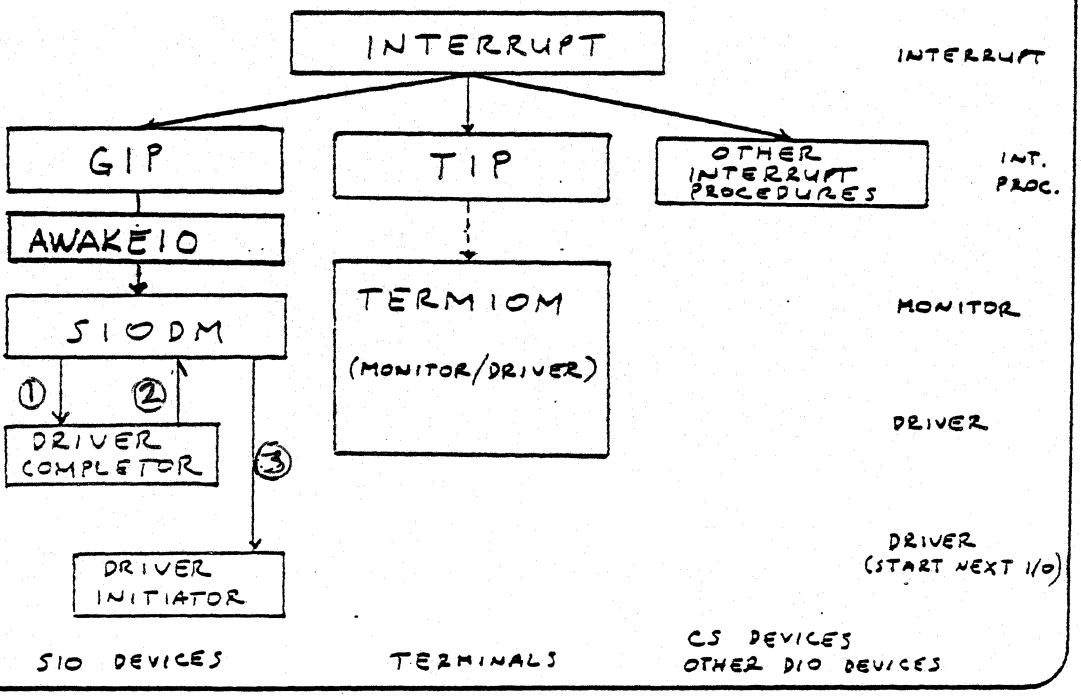
HEWLETT PACKARD

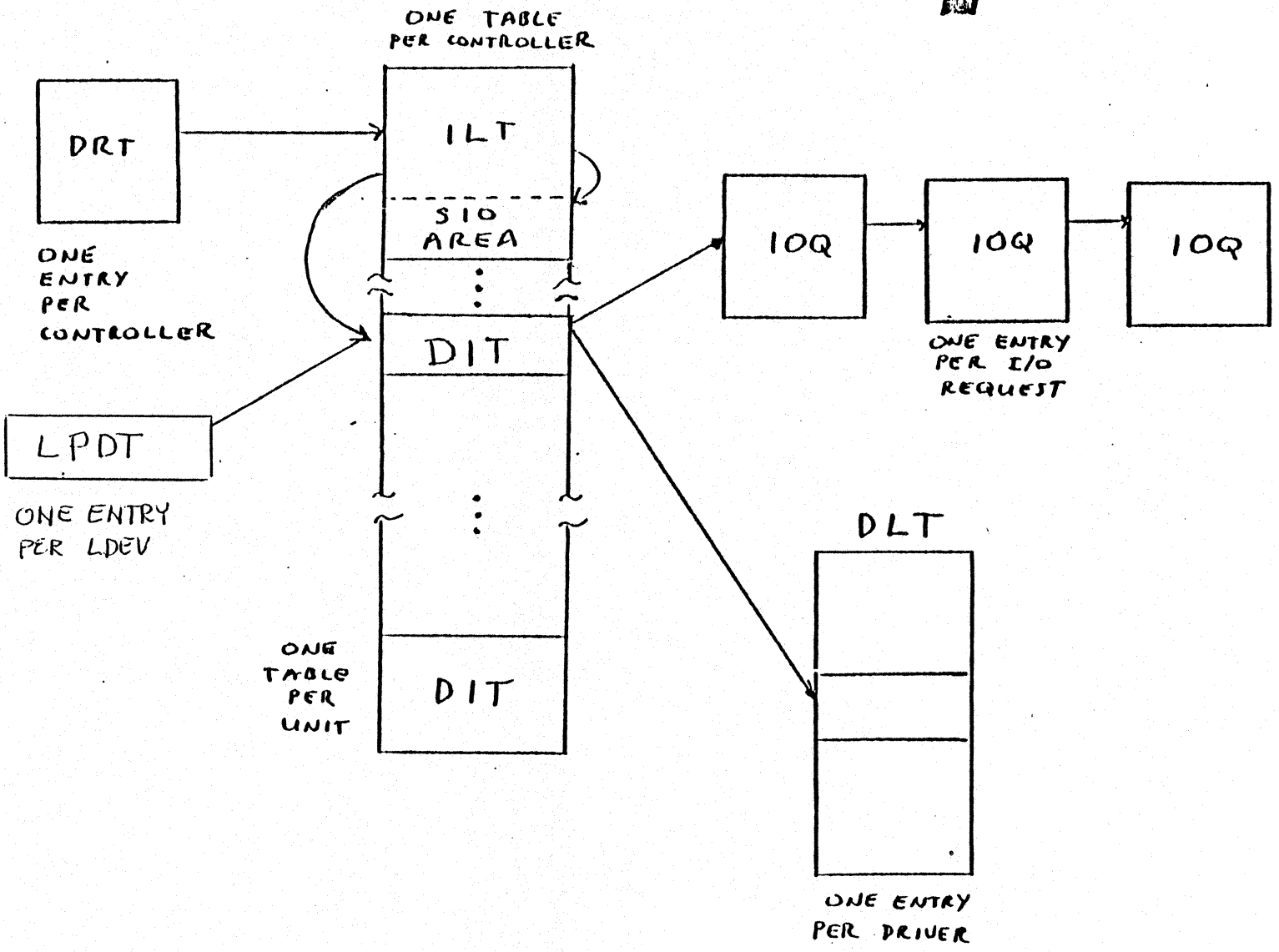
START OF LOGICAL I/O

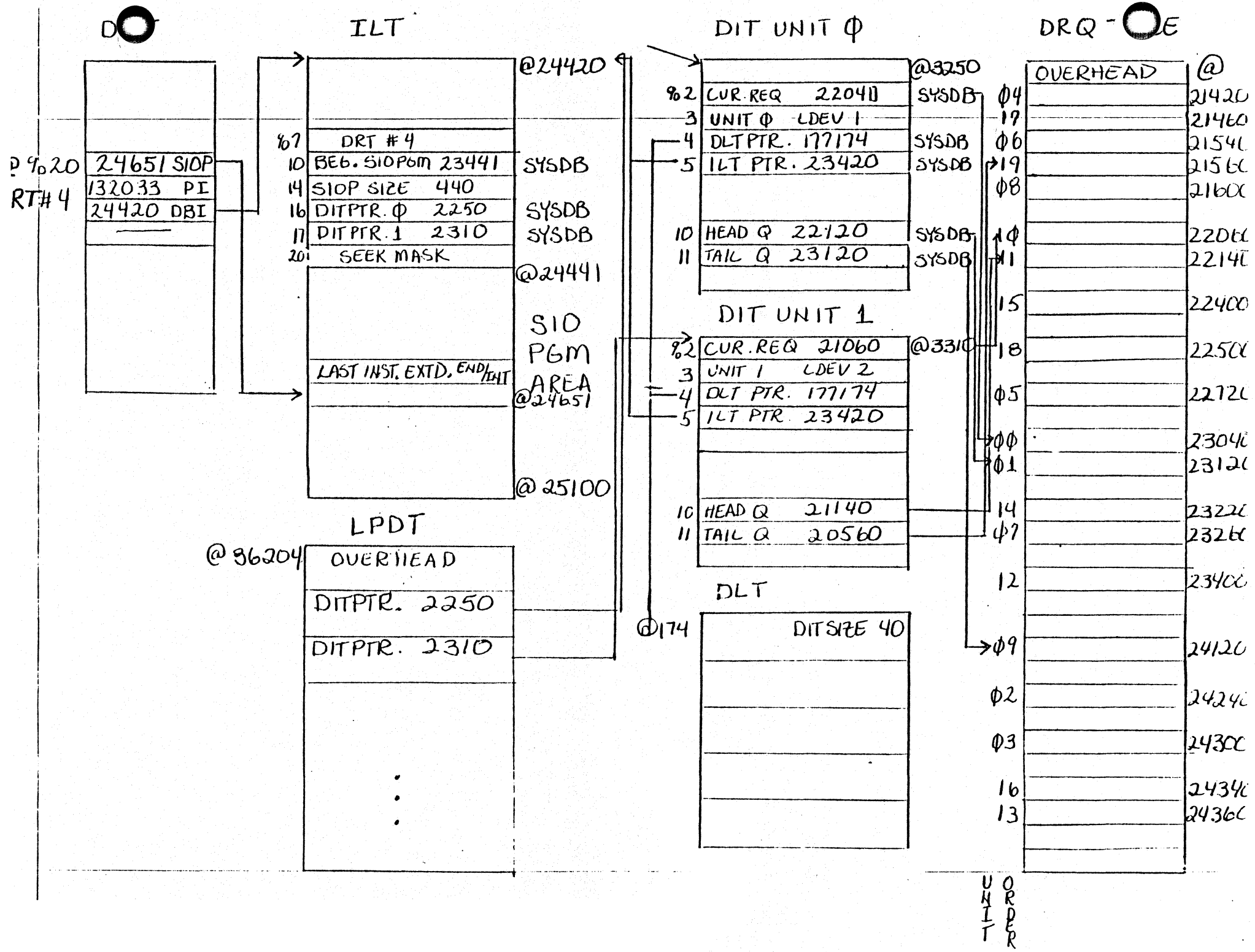


HEWLETT PACKARD

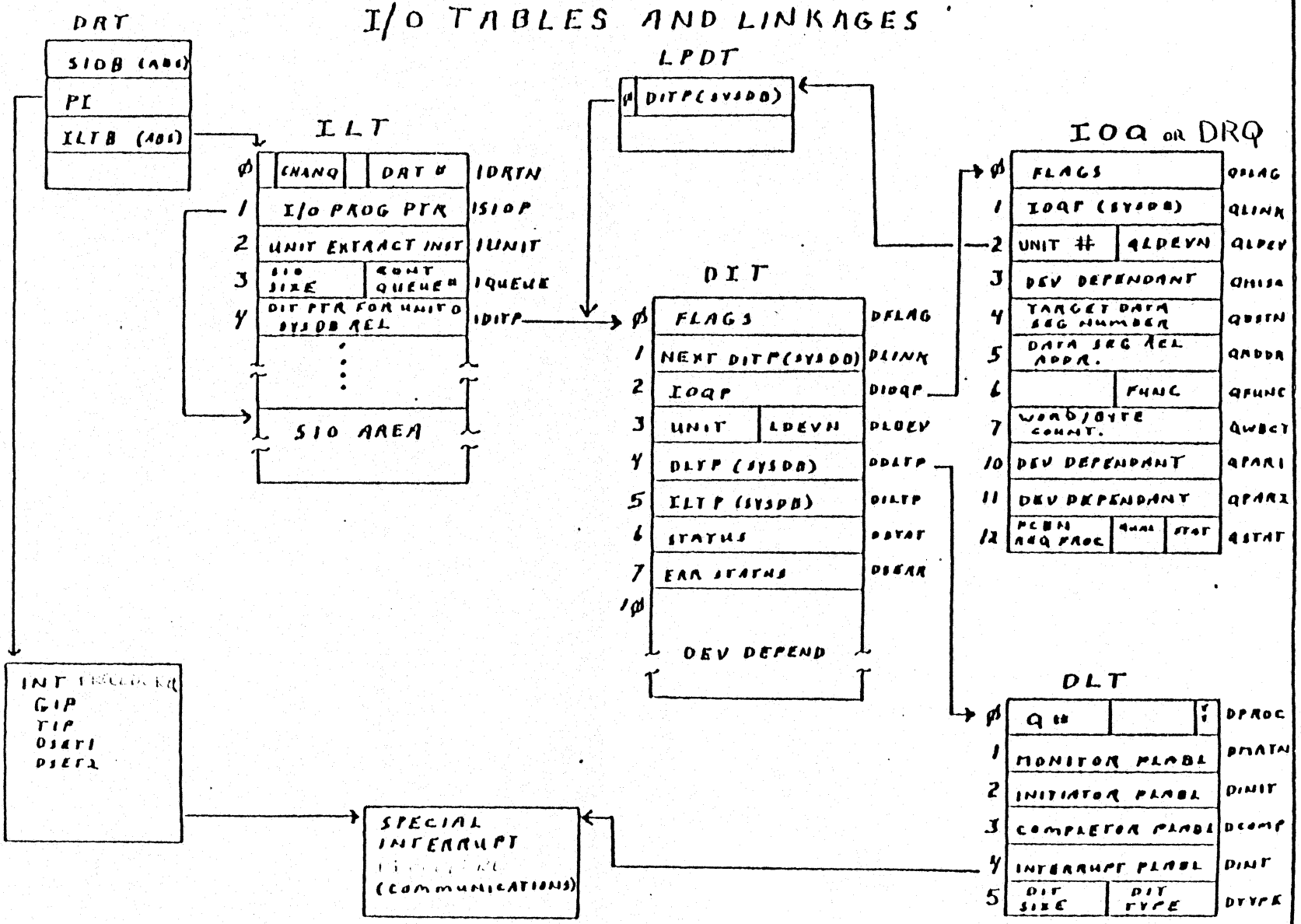
COMPLETION OF LOGICAL I/O




ONE TABLE PER CONTROLLER
DRT
ILT
SIO AREA
ONE ENTRY PER CONTROLLER
IOQ
IOQ
IOQ
ONE ENTRY PER I/O REQUEST
DIT
LPDT
ONE ENTRY PER LDEV
DLT
ONE TABLE PER UNIT
DIT
ONE ENTRY PER DRIVER


DCT
ILT
DIT UNIT Φ
DRQ - DE
@24420
@3250
OVERHEAD
@
%20 24651 SIOP
RT#4 132033 PI
24420 DBI
%7 DRT #4
10 BEG. SIOPGM 23441 SYSDB
14 SIOP SIZE 440
16 DITPTR. Φ 2250 SYSDB
17 DITPTR. 1 2310 SYSDB
20 SEEK MASK
@24441
SIO PGM AREA
@24651
LAST INST. EXTD. END/HALT
@25100
%2 CUR. REQ 22041 SYSDB
3 UNIT Φ LDEV 1
4 DLT PTR. 177174 SYSDB
5 ILT PTR. 23420 SYSDB
10 HEAD Q 22120 SYSDB
11 TAIL Q 23120 SYSDB
DIT UNIT 1
%2 CUR. REQ 21060
@3310
3 UNIT 1 LDEV 2
4 DLT PTR. 177174
5 ILT PTR 23420
10 HEAD Q 21140
11 TAIL Q 20560
LPDT
@96204
OVERHEAD
DITPTR. 2250
DITPTR. 2310
DLT
@174
DITSIZE 40
Φ4 21420
17 21460
Φ6 21540
19 21560
Φ8 21600
Φ 22060
1 22140
15 22400
18 22500
Φ5 22720
ΦΦ 23040
Φ1 23120
14 23220
Φ7 23260
12 23400
Φ9 24120
Φ2 24240
Φ3 24300
16 24340
13 24360
UNIT ORDER

I/O TABLES AND LINKAGES



DEVICE REFERENCE TABLE (DRT)

* LOCATED IN FIXED LOW MEMORY STARTING AT ADDRESS %14.

* EACH ENTRY AT LOCATION 4*DRTNUM.

* ONE EACH 4 WORD ENTRY IS INITIATED FOR EACH DRT NUMBER LISTED IN THE I/O CONFIGURATION TABLE.

* THE SIZE OF THE TABLE IS CONFIGURED BY THE QUESTION "HIGHEST DRT#" IN SYSDUMP AND INITIAL.

* IF ALL OF THE AREA THAT COULD BE USED FOR THE DRT (UP TO ADDRESS %777) IS NOT USED THEN THAT AREA IS USED FOR OTHER SYSTEM TABLES.

* WORD 0 - SIOP

* WORD 1 - PLABEL FOR THE INTERRUPT PROCESS FOR THIS DEVICE. INITIALLY SET BY INITIAL.

* WORD 2 - ILTP - THE ABSOLUTE ADDRESS OF WHERE THE ILT FOR THIS CONTROLLER RESIDES. THE VALUE FROM THIS LOCATION IS PUT IN DB BY THE MICROCODE HANDLING AN EXTERNAL INTERRUPT.

* WORD 3 - UNUSED BY MPE. USED BY INITIAL - .(0:8)= NUMBER INTERRUPT HANDLERS THIS DEVICE; .(8:8)= NUMBER UNITS THIS CONTROLLER.

INTERRUPT LINKAGE TABLE

* THERE IS ONE ILT FOR EACH CONTROLLER CONFIGURED IN THE SYSTEM.

* CONTAINS A UNIT EXTRACT INSTRUCTION TO BE USED TO EXTRACT THE UNIT NUMBER FROM THE STATUS RETURNED BY THE CONTROLLER AS THE RESULT OF A TIO INSTRUCTION. THE CONTENTS OF THIS MEMORY LOCATION IS PLACED ON TOS AND EXECUTED.

* CONTAINS ONE DIT POINTER FOR EACH UNIT CONFIGURED IN THE SYSTEM.

* CONTAINS AN SIO AREA WHERE THE ORDER PAIRS RESIDE WHILE THEY ARE ACTUALLY BEING EXECUTED BY THE CHANNEL.

* POINTED TO BY WORD 2 OF THE DRT ENTRY FOR THAT CONTROLLER AND WORD 5 OF THE DIT ENTRY FOR EACH UNIT ON THAT CONTROLLER.

DEVICE INFORMATION TABLE (DIT)

* ONE ENTRY FOR EACH UNIT CONFIGURED IN THE SYSTEM.

* FORMAT AND LENGTH IS DEVICE DEPENDENT AFTER WORD % 7

* GENERALLY DESIGNED TO CONTAIN INFORMATION THAT IS UNIQUE TO THAT TYPE OF UNIT (ESPECIALLY FOR MULTIUNIT CONTROLLERS) AND INFORMATION REGUARDING THE CURRENT STATE OF THAT UNIT.

* CONTAINS A POINTER (WORD 2) TO THE STRING OF IOQ'S LINKED TO THAT UNIT.

* DIT ENTRIES THEMSELVES ARE LINKED TO FORM REQUEST QUEUES FOR I/O RESOURCES (WORD 1).

* POINTED TO BY WORD (8:8-UNIT NUMBER) IN THE ILT, WORD 1 OF THE LPDT ENTRY AND IOQ(2) .(8:8) OF ANY IOQ ENTRY LINKED TO THAT DIT (THROUGH THE LPDT).

LOGICAL TO PHYSICAL DEVICE TABLE (LPDT)

* THE TABLE IS POINTED TO BY DST %15 AND EACH ENTRY IS POINTED TO BY THE LOGICAL DEVICE NUMBER FROM THE I/O CONFIGURATION TABLE.

* EACH ENTRY IS TWO WORDS LONG.

* THE ADDRESS OF THE NTH ENTRY IS TABLE ADDRESS + (2 * N).

* THERE ARE TWO TYPES OF ENTRIES:
  - REAL ENTRIES (BIT 0 OF WORD 0 = 0) - THIS ENTRY IS ONE THAT REFLECTS INFORMATION ABOUT A DEVICE THAT IS LISTED IN THE I/O CONFIGURATION TABLE.
  - VIRTUAL ENTRIES (BIT 0 OF WORD 0 = 1) - THIS ENTRY IS ONE THAT EXISTS TO KEEP THE ABOVE LISTED FORMULA VALID WITHOUT HAVING TO PLACE THE REQUIREMENT ON THE USER THAT THERE CANNOT BE ANY UNUSED LDEV NUMBERS. THESE ENTRIES ARE USED BY SPOOLER TO HOLD DISC ADDRESSES OF ACTIVE INPUT STREAMED JOBS.

* EACH REAL ENTRY CONTAINS A POINTER TO THE DIT FOR THIS UNIT AND SYSTEM USAGE INFORMATION FROM THE I/O CONFIGURATION TABLE SUCH AS J, A, I, AND D PARAMETERS.

* IT IS NAMED THE LPDT BECAUSE IT MAPS A LOGICAL DEVICE INTO A PHYSICAL DEVICE THROUGH THE DIT FOR THE DEVICE.

DRIVER LINKAGE TABLE (DLT)

* POINTED TO BY DST %17.

* AN ENTRY IS POINTED TO BY DIT(4) FOR THAT UNIT.

* AN ENTRY CONTAINS:
  - MONITOR PLABEL
  - DRIVER INITIATOR PLABEL
  - DRIVER COMPLETOR PLABEL
  - INTERRUPT [illegible] PLABEL
  - DIT SIZE
  - DEVICE TYPE FROM I/O CONFIGURATION TABLE

* ONE ENTRY PER DRIVER CONFIGURED IN THE SYSTEM.

* IF MORE THAN ONE UNIT USES THE SAME DRIVER THE DIT FOR EACH UNIT USING THAT ONE DRIVER WILL POINT TO THE SAME DLT ENTRY.

* IF DIFFERENT UNITS ON THE SAME CONTROLLER USE DIFFERENT DRIVERS THE RESPECTIVE DIT'S WILL POINT TO DIFFERENT DLT ENTRIES TO USE THE CORRECT MONITOR, DRIVER AND INTERRUPT HANDLER.

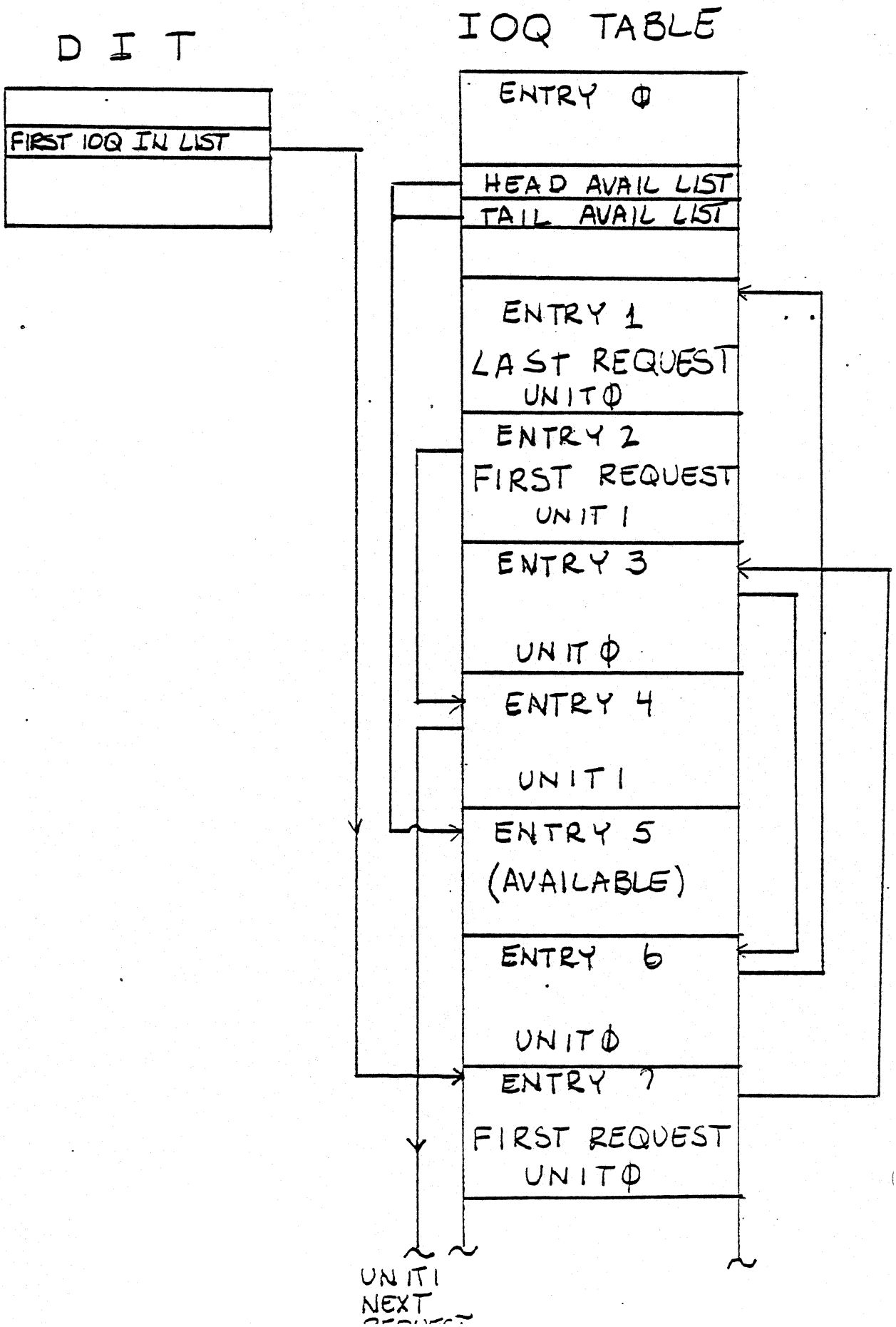
I/O REQUEST QUEUE (IOQ)

* THE TABLE IS POINTED TO BY DST %13 AND THE ENTRY TO ONE QUEUE LIST IS POINTED TO BY WORD 2 OF A DIT ENTRY.

* ALL ENTRIES ARE %13 WORDS LONG.

* THERE IS ONE IOQ TABLE IN THE SYSTEM. ONE ENTRY IS DYNAMICALLY ALLOCATED FOR EACH NON-DISC I/O REQUEST POSTED FOR THE FILE SYSTEM. THAT ENTRY IS THEN LINKED TO ALL OTHER REQUESTS FOR THAT UNIT. THE START OF A REQUEST LIST, I.E., THE HIGHEST PRIORITY REQUEST, IS LINKED TO BY WORD 2 OF THE DIT OF THE UNIT THE REQUEST WAS POSTED FOR. ALL OTHER REQUESTS FOR THAT UNIT ARE THEN LINKED BY PRIORITY BY WORD 1 OF THE HIGHER PRIORITY IOQ ENTRY. THE END OF THE LIST IS FOUND BY HAVING A ZERO IN WORD 1 OF THE LOWEST PRIORITY IOQ ENTRY.

* EACH IOQ ENTRY CONTAINS ALL OF THE PARAMETERS NEEDED TO PERFORM THE I/O FOR THAT REQUEST AND INFORMATION DEFINING THE CURRENT STATE OF THAT REQUEST.


DIT
FIRST IOQ IN LIST
IOQ TABLE
ENTRY Φ
HEAD AVAIL LIST
TAIL AVAIL LIST
ENTRY 1
LAST REQUEST
UNIT Φ
ENTRY 2
FIRST REQUEST
UNIT 1
ENTRY 3
UNIT Φ
ENTRY 4
UNIT 1
ENTRY 5
(AVAILABLE)
ENTRY 6
UNIT Φ
ENTRY 7
FIRST REQUEST
UNIT Φ
UNIT 1
NEXT

DISC REQUEST QUEUE

- THE TABLE IS POINTED TO BY DST X'7B'. THE CURRENT REQUEST IS POINTED TO BY WORD 2 OF THE DIT AND THE HEAD AND TAIL OF THE QUEUE ARE POINTED TO BY WORD 10 AND 11 OF THE DIT.

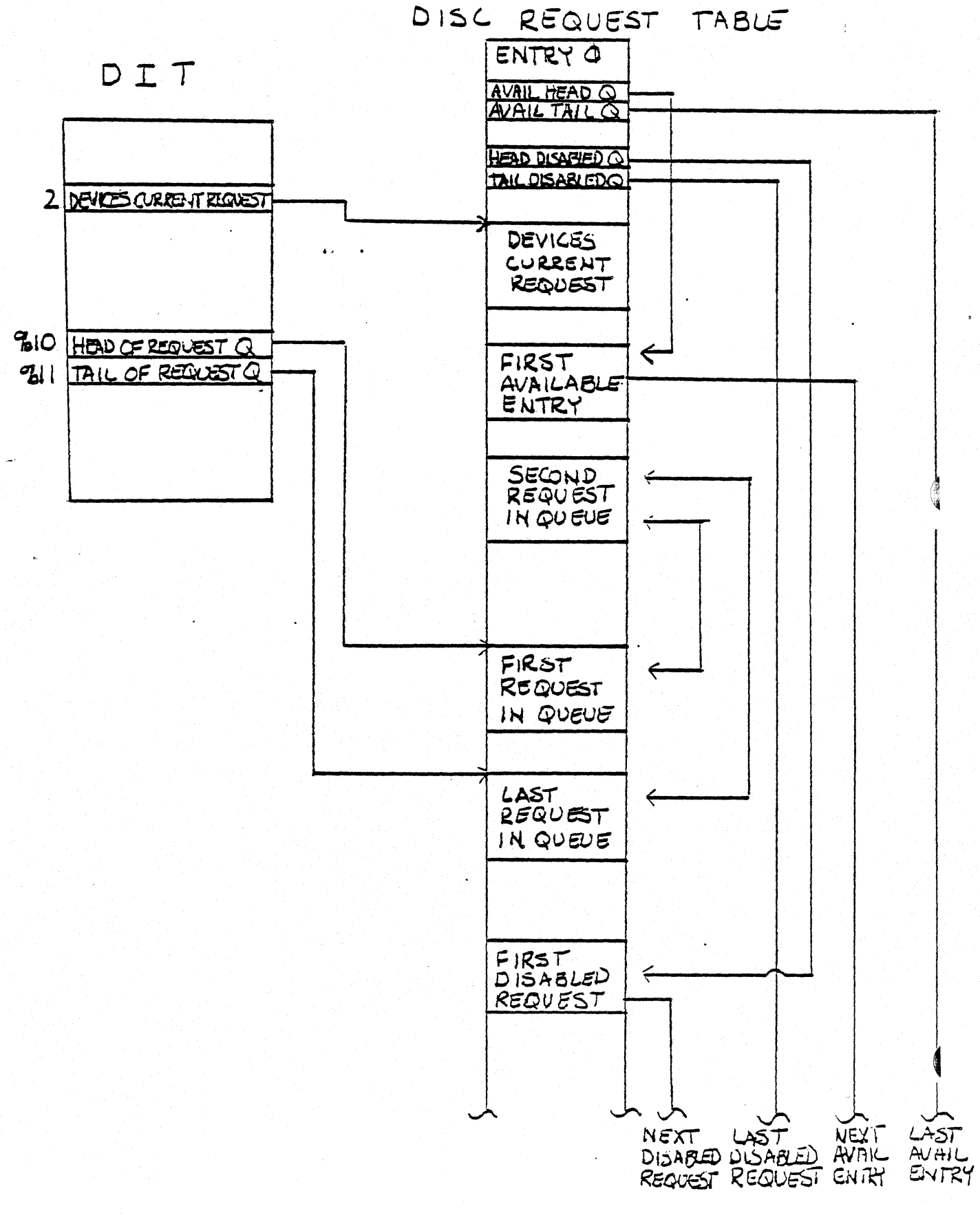
- ALL ENTRIES ARE X'10' WORDS LONG.

- PRIORITY OF REQUEST DETERMINES PLACEMENT IN QUEUE.

- PRIORITY OF DISC REQUESTS ARE SET TO THE PRIORITY OF THE REQUESTING PROCESS.

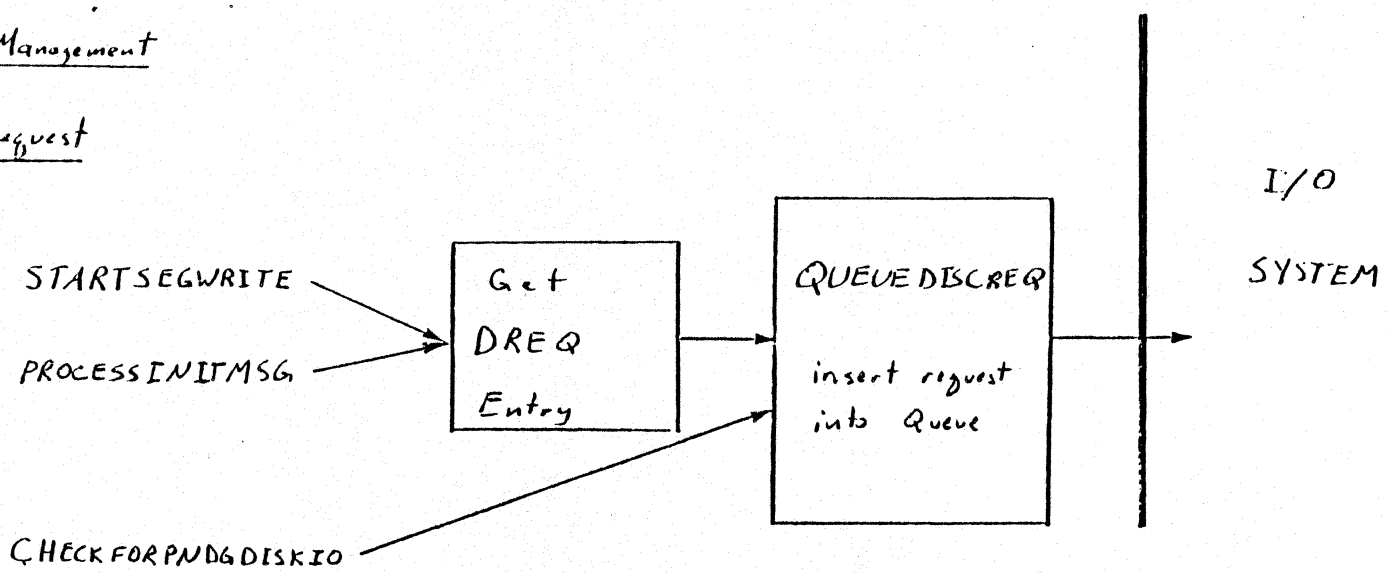
- PRIORITY OF BACKGROUND WRITE REQUESTS ARE SET VERY LOW (250) UNTIL THE AREA IN MEMORY IS ACTUALLY SELECTED BY THE MEMORY MANAGER. THEN THE PRIORITY BECOMES THE PRIORITY OF THE PROCESS REQUESTING THE MEMORY SPACE.

- THERE IS ONE DRQ TABLE FOR THE WHOLE SYSTEM.


DISC REQUEST TABLE
DIT
ENTRY 0
AVAIL HEAD Q
AVAIL TAIL Q
HEAD DISABLED Q
TAIL DISABLED Q
2
DEVICES CURRENT REQUEST
DEVICES CURRENT REQUEST
%10
HEAD OF REQUEST Q
%11
TAIL OF REQUEST Q
FIRST AVAILABLE ENTRY
SECOND REQUEST IN QUEUE
FIRST REQUEST IN QUEUE
LAST REQUEST IN QUEUE
FIRST DISABLED REQUEST
NEXT DISABLED REQUEST
LAST DISABLED REQUEST
NEXT AVAIL ENTRY
LAST AVAIL ENTRY

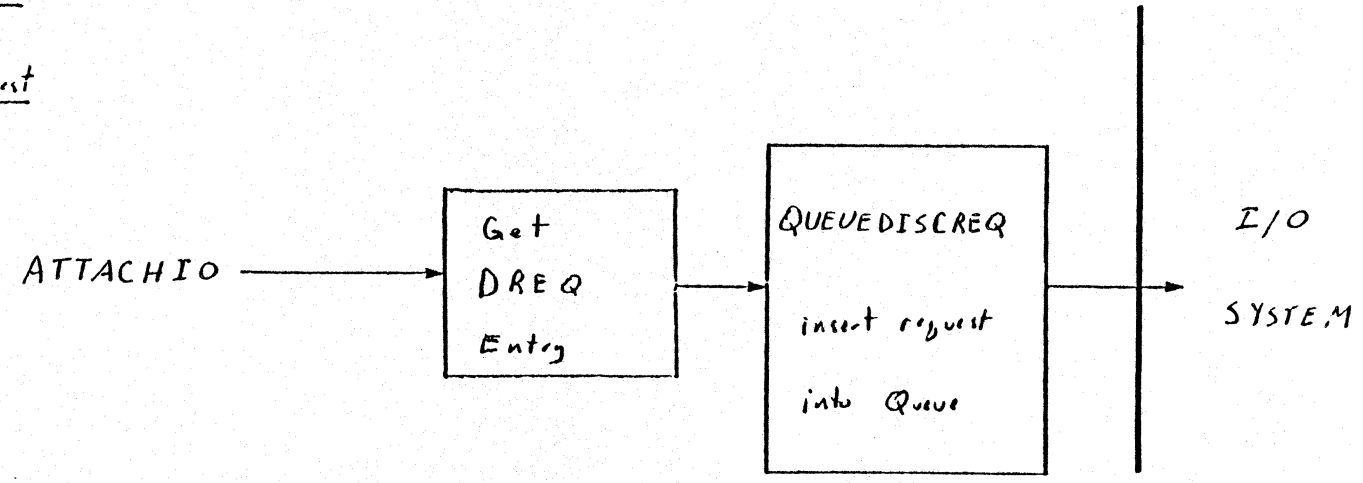
Memory Management

Disk Request

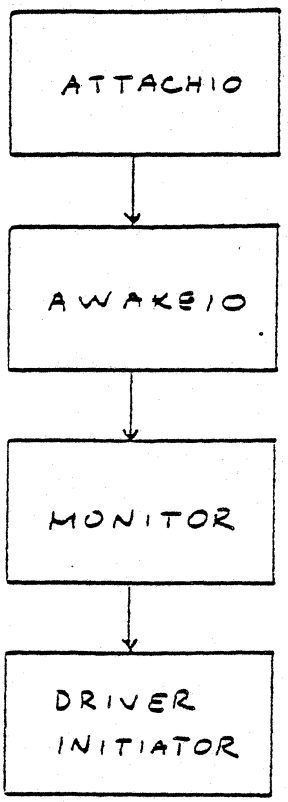


File System

Disk Request



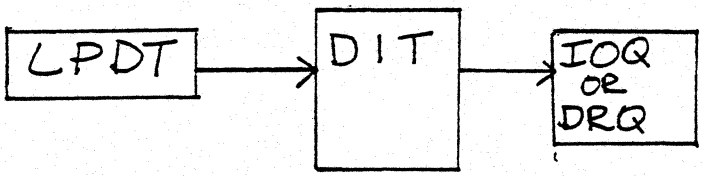
START OF LOGICAL I/O



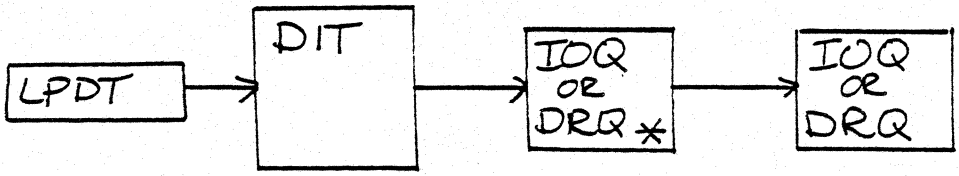
ATTACHIO

ATTACHIO'S PRIMARY FUNCTION IS TO LINK AN IOQ OR DRQ TO THE DIT FOR EACH I/O REQUEST

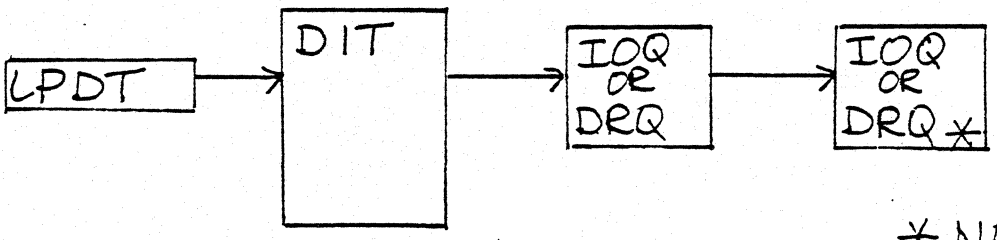
-IF THE REQUEST IS THE FIRST ELEMENT TO BE LINKED TO THE DIT. ATTACHIO WILL CALL AWAKEIO WHO WILL FIND THE CORRECT MONITOR



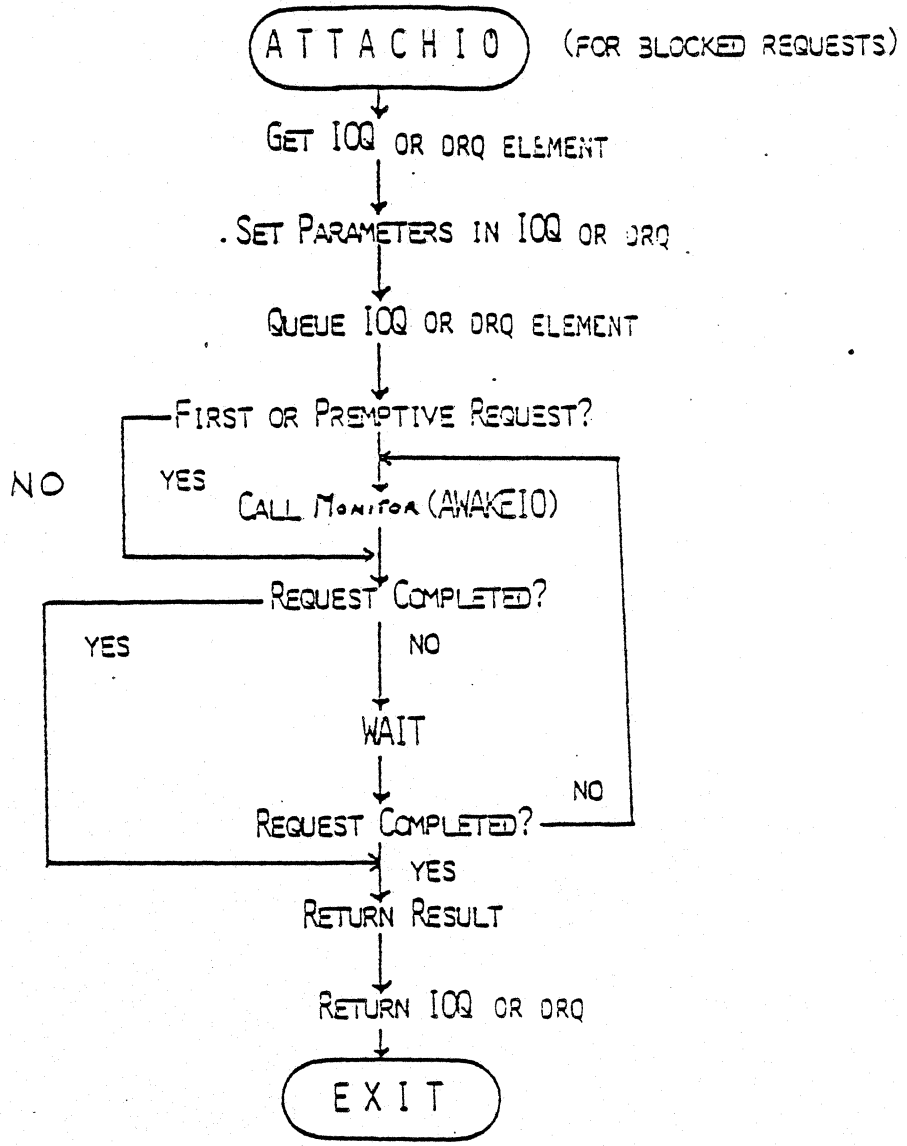
-IF THE REQUEST IS A PREEMPTIVE IOQ OR A HIGHER PRIORITY DRQ IT IS LINKED AT THE BEGINNING OF THE QUEUE. ATTACHIO CALLS AWAKEIO WHO FINDS THE MONITOR



-IF ONE OF THE ABOVE CONDITIONS IS NOT MET, ATTACHIO LINKS THE IOQ OR DRQ AT THE END OF THE LIST AND CALLS WAIT.

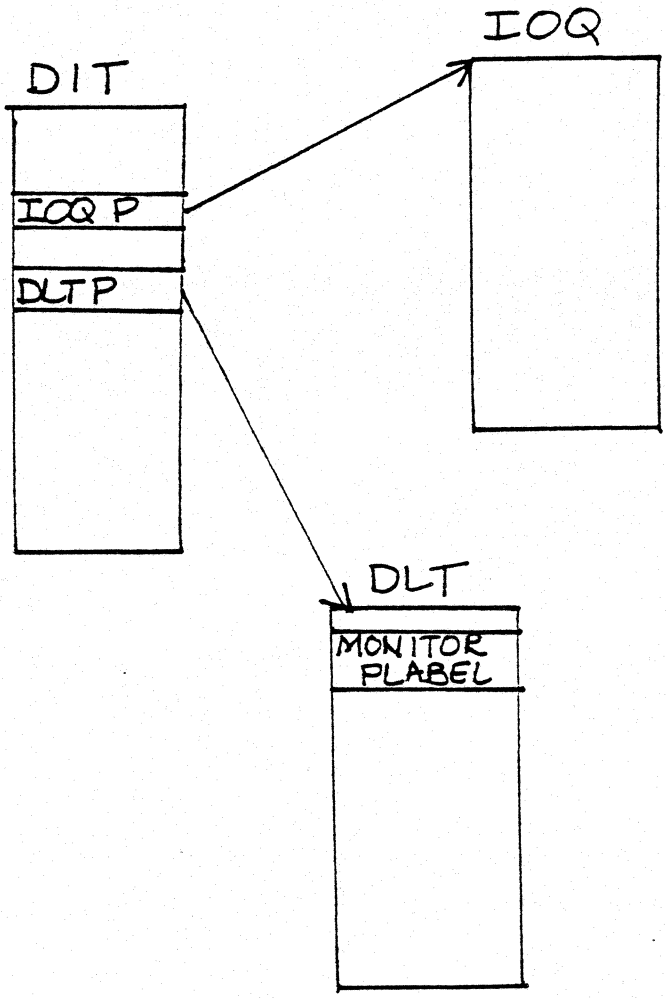


* NEW REQUEST


ATTACHIO
(FOR BLOCKED REQUESTS)
GET IOQ OR DRQ ELEMENT
SET PARAMETERS IN IOQ OR DRQ
QUEUE IOQ OR DRQ ELEMENT
FIRST OR PREMPTIVE REQUEST?
NO
YES
CALL MONITOR (AWAKEIO)
REQUEST COMPLETED?
YES
NO
WAIT
NO
REQUEST COMPLETED?
YES
RETURN RESULT
RETURN IOQ OR DRQ
EXIT

AWAKE IO

* DETERMINES THE CORRECT MONITOR FOR THE DEVICE
* GET THE MONITOR PLABEL
* PCAL THE I/O MONITOR PROCEDURE

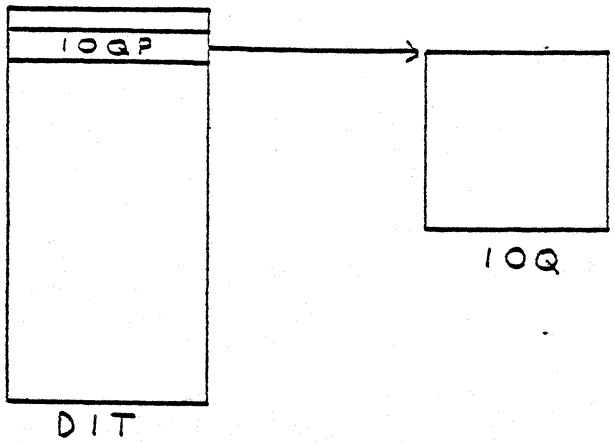


SIODM IS THE MONITOR FOR ALL SIO DEVICES

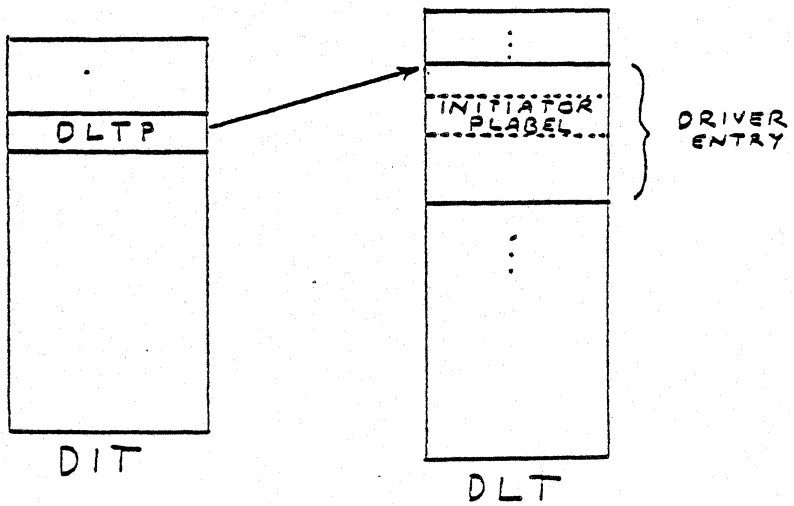
MONITOR

THE MONITOR IS RESPONSIBLE FOR:

1. SELECTING THE NEXT IOQ or DRQ



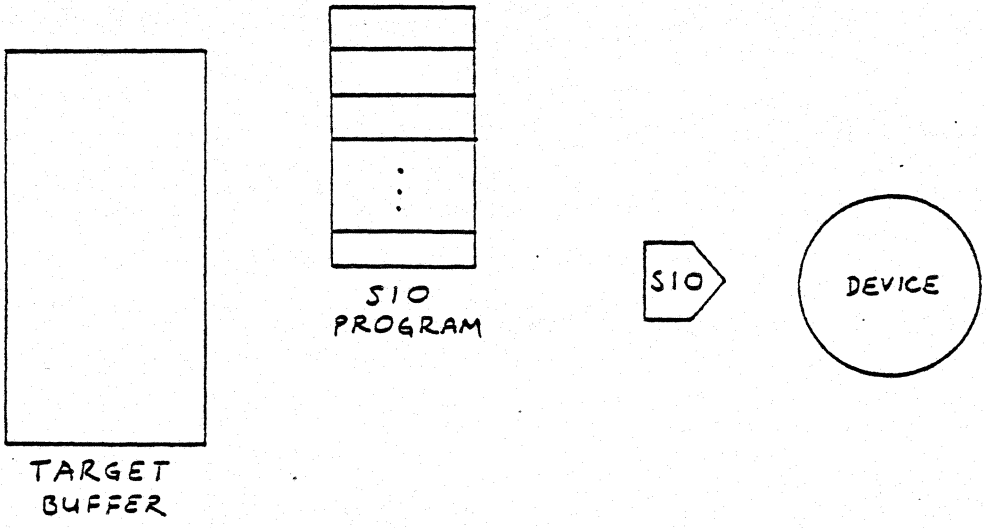
2. FINDING THE CORRECT DRIVER INITIATOR AND PCALING IT.



DRIVER INITIATOR

THE DRIVER INITIATOR IS RESPONSIBLE FOR INITIALIZING AND STARTING THE PHYSICAL I/O TRANSFER.

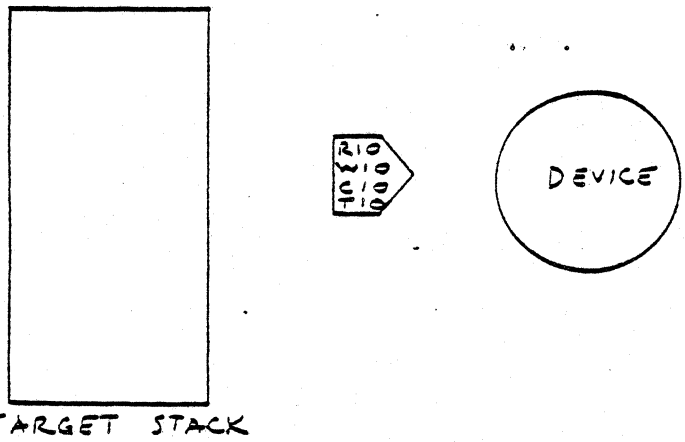
FOR S/III SIO DEVICES, THIS INVOLVES FILLING IN AN SIO PROGRAM, THEN CALLING STARTIO WHICH DOES AN SIO CPU INSTRUCTION.



HEWLETT PACKARD

DRIVER INITIATOR

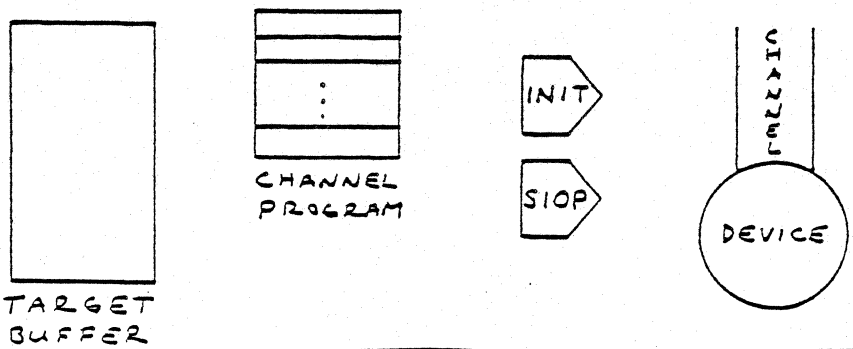
FOR S/III DIRECT I/O DEVICES, THIS INVOLVES EXECUTING RIO, WIO, CIO, TIO CPU INSTRUCTIONS.



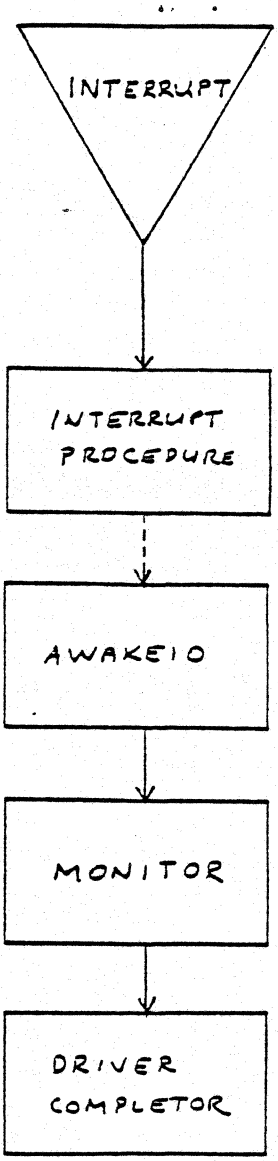
HEWLETT PACKARD

DRIVER INITIATOR

FOR S/33 DEVICES, THIS INVOLVES FILLING IN THE CHANNEL PROGRAM, THEN CALLING STARTIO WHICH DOES AN INIT (CHANNEL) AND SIOP CPU INSTRUCTIONS.

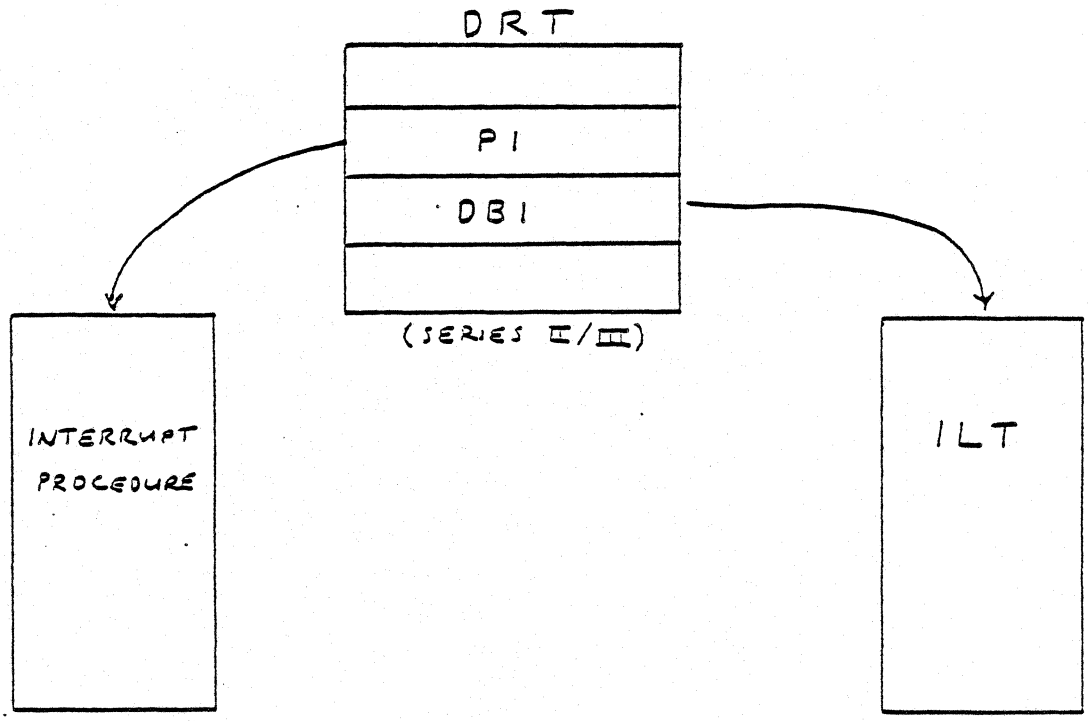


COMPLETION OF LOGICAL I/O



INTERRUPT PROCEDURE

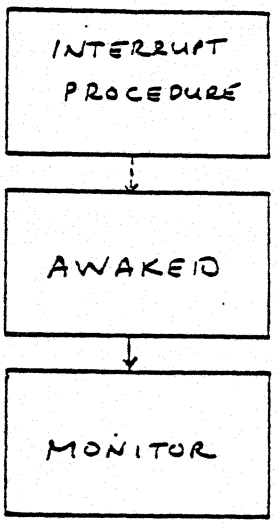
FOR SIO DEVICES AND TERMINALS, AN INTERRUPT WILL OCCUR UPON AN I/O TRANSFER COMPLETION. THIS CAUSES THE CPU MICROCODE TO FETCH THE DRT ENTRY, IN ORDER TO FIND THE CORRECT MPE INTERRUPT PROCEDURE AND DB LOCATION.



GIP : SIO DEVICES
TIP : TERMINALS

INTERRUPT PROCEDURE

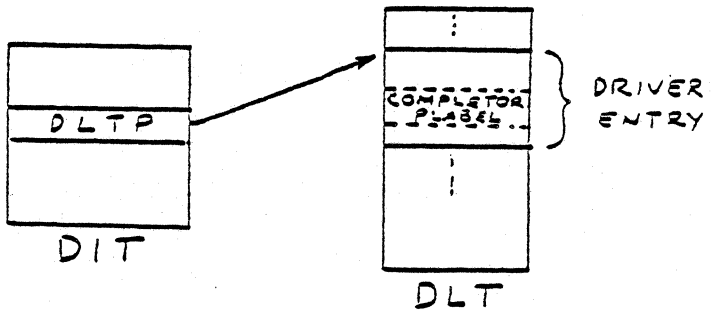
IF THE ENTIRE I/O IS DONE, THE EXECUTION OF THE INTERRUPT PROCEDURE WILL RESULT IN AWAKEIO PCALING THE MONITOR. THIS MAY OCCUR ON A SYSTEM PROCESS STACK OR THE ICS, DEPENDING ON THE MONITOR TYPE.



HEWLETT PACKARD

MONITOR

UPON I/O COMPLETION, THE MONITOR MUST FIND THE COMPLETOR SECTION OF THE DRIVER AND PCAL IT.



HEWLETT PACKARD

DRIVER COMPLETOR

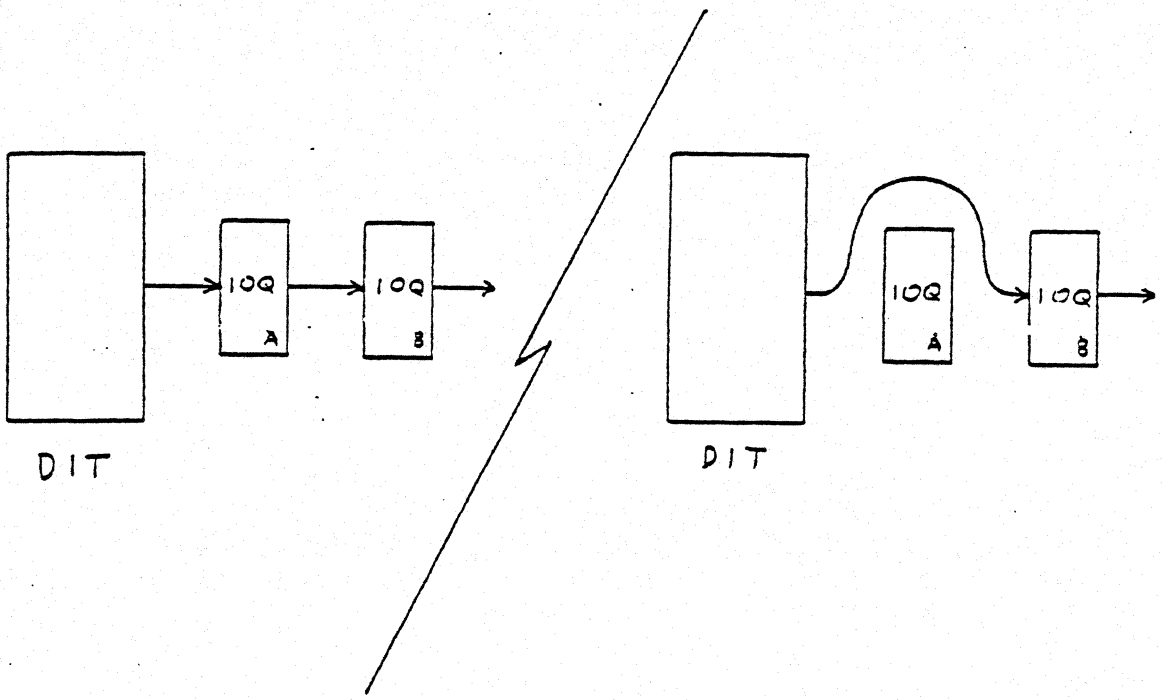
THE DRIVER COMPLETOR WILL CHECK THE STATUS OF THE I/O COMPLETION (DIT) AND DO ANY ERROR PROCESSING NECESSARY.

WHEN DONE, IT RETURNS TO THE MONITOR.

MONITOR

THE MONITOR RUNS AGAIN, THIS TIME TO:

1. AWAKE THE WAITING PROCESS (IF NECESSARY).
2. RELEASE THE IOQ FROM THE DIT.
3. IF ANOTHER IOQ IS WAITING FOR THIS [illegible], START IT BY CALLING AWAKEIO (WHICH WILL CALL THE APPROPRIATE MONITOR).
   THIS IS A <u>RIPPLING EFFECT</u> FOR IOQ's.
4. RETURN TO ATTACHIO.



ATTACHIO

THE USER PROCESS AWAKES IN ATTACHIO AND WILL:

1. RETURN THE IOQ&DRQ TO THE FREE LIST.
2. RETURN CORRECT STATUS TO CALLER.

FOR SIO DEVICES

USER STACK:

1. THE USER PROGRAM CALLS A FILE SYSTEM INTRINSIC WHICH TRANSFERS CONTROL TO THE FILE SYSTEM.

2. THE FILE SYSTEM SETS UP THE ENVIROMENT FOR THE IO SYSTEM AND CALLS ATTACHIO.

3. ATTACHIO FILLS IN THE IOQ OR DRQ ENTRY FOR THIS REQUEST, USING THE PARAMETERS PASSED FROM THE FILE SYSTEM, AND LINKS IT INTO THE IOQ OR DRQ LIST FOR THE APPROPRIATE DIT. IF THE MONITOR IS BUSY ATTACHIO CALLS WAIT, OTHERWISE IT CALLS AWAKEIO.

4. AWAKEIO FINDS THE CORRECT MONITOR FOR THE DRIVER AND THEN CALLS IT. IN THIS CASE THE MONITOR IS SIODM.

5. SIODM CHECKS IF THE CONTROLLER IS BUSY, IF NOT IT CALLS THE DRIVER INITIATOR. OTHERWISE IT RETURNS TO ATTACHIO.

6. THE DRIVER INITIATOR FILLS IN THE CHANNEL PROGRAM SKELETON IN THE ILT AND CALLS STARTIO, TO START THE I/O REQUEST. THE DRIVER INITIATOR THEN RETURNS TO ATTACHIO.

7. ATTACHIO CALLS WAIT

AT THIS POINT THE CHANNEL PROGRAM EXECUTES AND INTERRUPTS UPON COMPLETION.

FOR SIO DEVICES

ICS:

8. GIP SAVES THE HARDWARE INTERRUPT STATUS, AND CALLS AWAKEIO.

9. AWAKEIO FINDS THE PLABEL FOR SIODM AND THEN CALLS SIODM.

10. SIODM CALLS THE DRIVER COMPLETOR.

11. THE DRIVER COMPLETOR CHECKS FOR ERROR CONDITIONS AND UPDATES THE IOQ OR DRQ ENTRY WITH THE COMPLETION STATUS AND RETURNS TO SIODM.

12. SIODM UNFREEZES THE DATA BUFFER AND CALLS AWAKE FOR THE USER WHO'S I/O HAS COMPLETED.

13. AWAKE SETS THE USER ACTIVE, CLEARS THE WAIT CONDITIONS, AND RETURNS TO SIODM.

14. SIODM MANAGES THE CONTROLLER QUEUE AND STARTS ANOTHER REQUEST IF ONE IS READY BY CALLING AWAKEIO, IF A REQUEST IS NOT AVAILABLE SIODM RETURNS TO AWAKEIO WHO RETURNS TO GIP.

15. GIP EXECUTES AN 'IXIT' WHICH EXITS OFFTHE ICS AND RETURNS CONTROL TO THE INTERRUPTED PROCESS OR THE DISPATCHER.

USERS STACK:

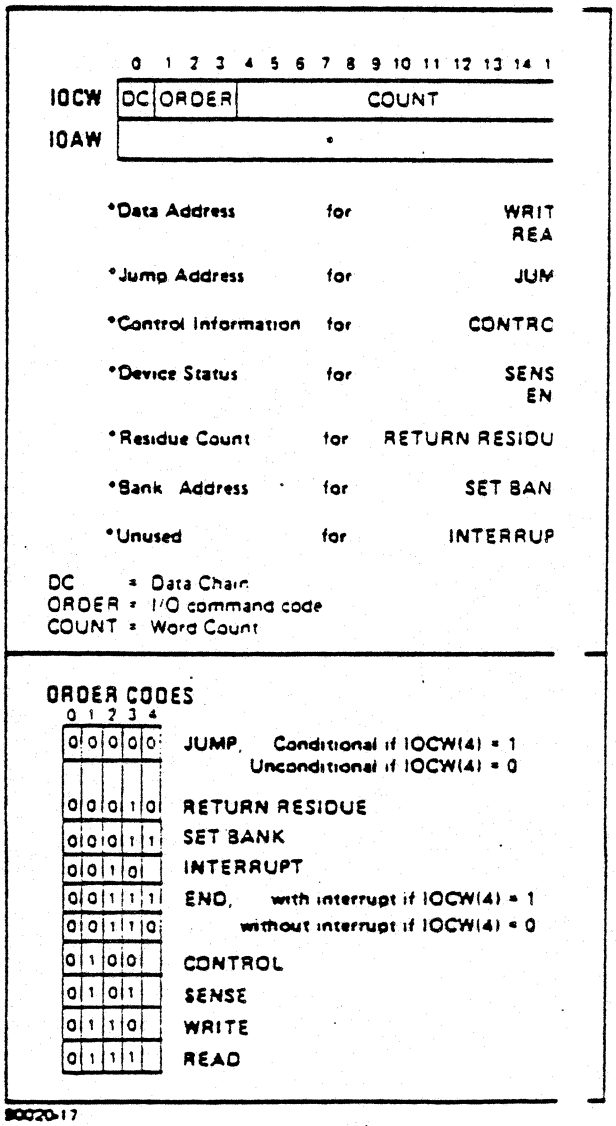
16. THE USER PROCESS IS LAUNCHED BY THE DISPATCHER WHEN IT IS THE HIGHEST PRIORITY PROCESS.

17. ATTACHIO DELINKS THE IOQ OR DRQ AND RETURNS TO THE FILE SYSTEM, PASSING THE I/O REQUEST STATUS.

18. THE FILE SYSTEM SETS THE CONDITION CODES REFLECTING THE I/O STATUS AND CONTROL IS TRANSFERRED BACK TO THE USER PROGRAM.

WHAT IS AN SIO PROGRAM AND WHAT DOES IT DO?

AN SIO PROGRAM IS A SET OF ORDER PAIRS USED TO CONTROL MANY PERIPHERALS (DISC, TAPE, LINE PRINTER, ETC.).



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IOCW	DC	ORDER							COUNT							
IOAW	*															

*Data Address	for	WRIT REA
*Jump Address	for	JUM
*Control Information	for	CONTRC
*Device Status	for	SENS EN
*Residue Count	for	RETURN RESIDU
*Bank Address	for	SET BAN
*Unused	for	INTERRUP

DC = Data Chain
ORDER = I/O command code
COUNT = Word Count

ORDER CODES

0	1	2	3	4	
0	0	0	0	0	JUMP, Conditional if IOCW(4) = 1 Unconditional if IOCW(4) = 0
0	0	0	1	0	RETURN RESIDUE
0	0	0	1	1	SET BANK
0	0	1	0	0	INTERRUPT
0	0	1	1	1	END, with interrupt if IOCW(4) = 1
0	0	1	1	0	without interrupt if IOCW(4) = 0
0	1	0	0		CONTROL
0	1	0	1		SENSE
0	1	1	0		WRITE
0	1	1	1		READ

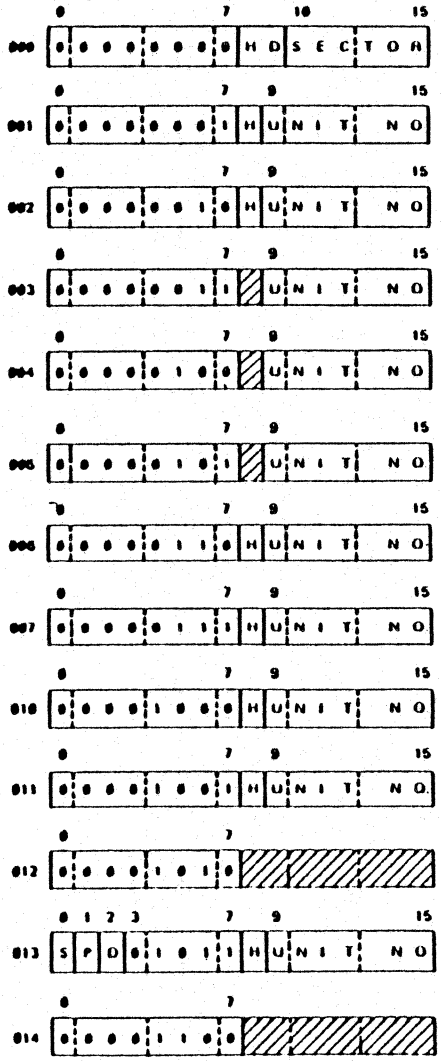
90020-17

I/O Program Word Format

HEWLETT hp PACKARD

7905/7920 Disc Drives

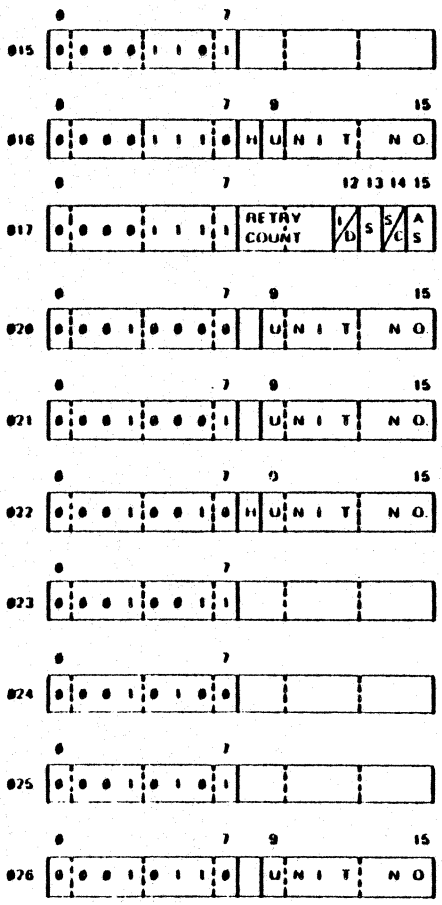
(SIO)



*13037A Will Disconnect After Execution.

7905/7920 Disc Drives

(SIO)



*13037A Will Disconnect After Execution.

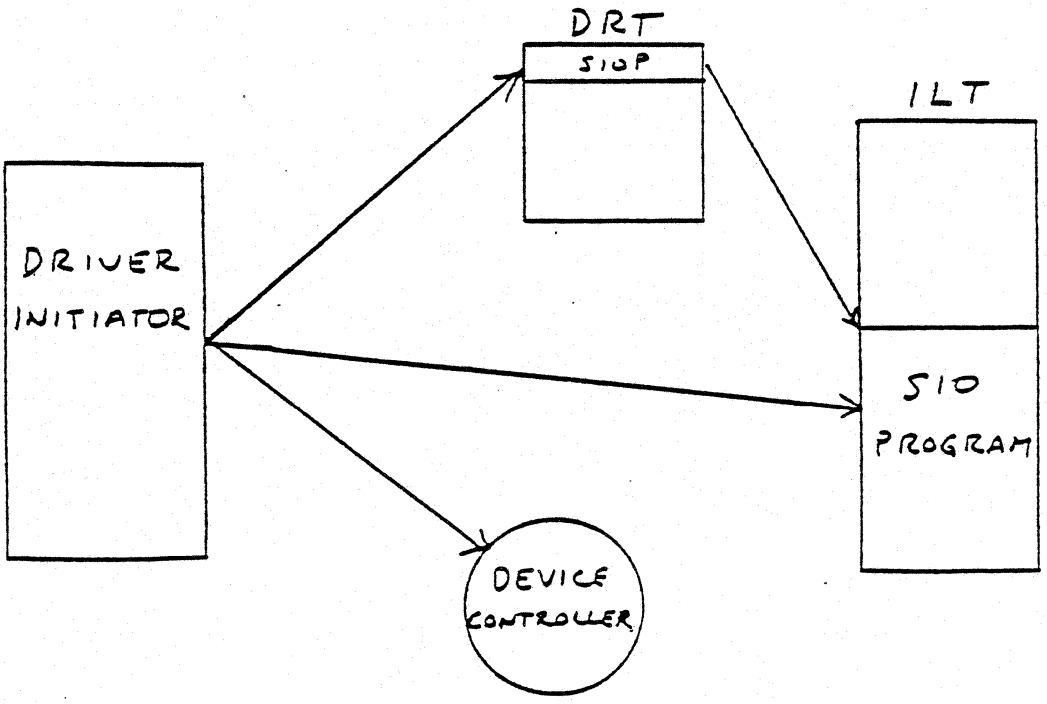
SIO CONTROL WORDS FOR 7905/06/20/25

TO 21

HOW ARE SIO PROGRAMS EXECUTED?

THE DRIVER INITIATOR BUILDS THE SIO PROGRAM IN THE ILT (MEMORY RESIDENT). NOTE THAT THERE IS ONLY ONE ILT PER CONTROLLER; THUS, ONLY ONE UNIT PER CONTROLLER MAY BE ACTIVE.

THE DRIVER INITIATOR DOES AN 'SIO' CPU INSTRUCTION TO THE DEVICE CONTROLLER, THEN EXITS BACK TO SIODM.
THE SIO INSTRUCTION ALSO INITIALIZES THE DRT WORD Ø (SIOP) TO POINT TO THE START OF THE SIO PROGRAM.



HEWLETT hp PACKARD

HOW SIO PROGRAMS ARE EXECUTED (cont.)

THE SIO INSTRUCTION ALERTS THE IOP TO TELL THE DEVICE CONTROLLER TO START EXECUTING THE PROGRAM.

FROM THIS POINT, THE HARDWARE EXECUTES THE SIO PROGRAM.

HEWLETT hp PACKARD

HOW SIO PROGRAMS ARE EXECUTED (cont.)

WHEN THE SIO PROGRAM IS FINISHED, THE LAST ORDER PAIR IS USUALLY AN "END WITH INTERRUPT".

THE INTERRUPT TRIGGERS GIP, WHO CALLS THE DRIVER COMPLETOR.

LOOK-AHEAD SEEKS

* AVAILABLE ONLY ON SERIES II/III SYSTEMS

* SEEKS FOR REQUESTS ON NEXT THREE DISC UNITS WRITTEN INTO SIO PROGRAM FOR CONTROLLER'S CURRENT REQUEST (IF THE SEEKS HAVE NOT ALREADY BEEN ISSUED)

* ALLOWS THREE UNITS TO BE "SEEKING" WHILE ONE IS TRANSFERRING DATA

WHAT HAPPENS IF A CONTROLLER SERVES MORE THAN ONE UNIT?

THERE IS ONLY ONE ILT PER CONTROLLER; HENCE, ONLY ONE DIT (UNIT) PER CONTROLLER MAY BE ACTIVE FOR I/O.

SO HOW DO WE CHOOSE WHICH DIT GOES NEXT?

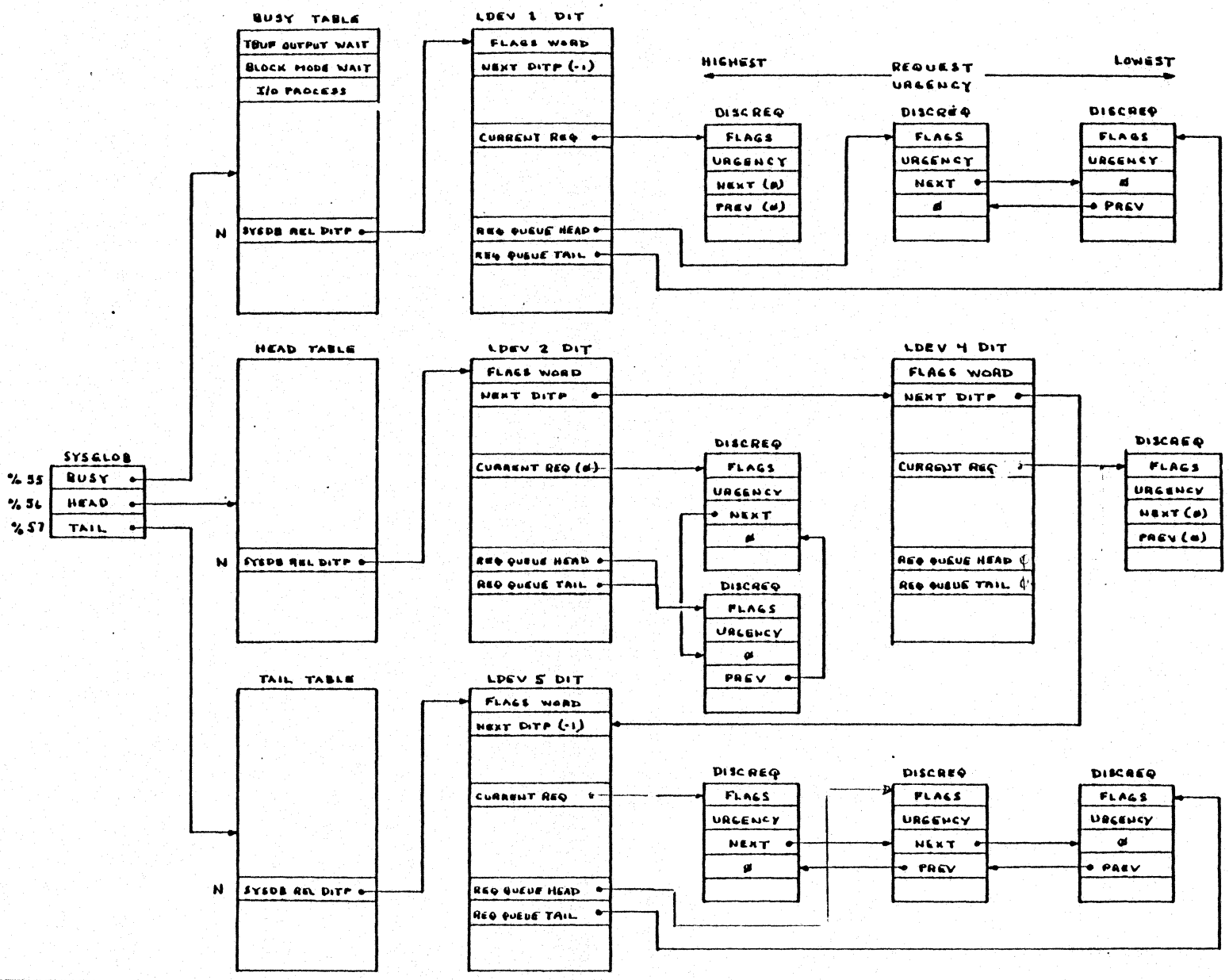
FIRST, EVERY MULTIUNIT CONTROLLER IS GIVEN AN I/O RESOURCE NUMBER, KEPT IN THE ILT.

NEXT, THERE ARE THREE LISTS,
BUSY, HEAD AND TAIL LISTS,
POINTED TO BY SYSGLOB.

SYSGLOB %55 = SYSGLOB REL. ADDR. TO BUSY LIST
56 = " " " " HEAD "
57 = " " " " TAIL "

EACH LIST CONSISTS OF ONE WORD ENTRIES INDEXED BY I/O RESOURCE NUMBER, CONTAINING THE ABSOLUTE ADDRESS OF A DIT BELONGING TO RESOURCE n.

WHEN THE IO COMPLETES, SIODM WILL ROTATE THE BUSY DIT TO THE TAIL AND MOVE THE HEAD DIT TO BUSY.


BUSY TABLE
TBUF OUTPUT WAIT
BLOCK MODE WAIT
I/O PROCESS
N
SYSDB REL DITP
LDEV 1 DIT
FLAGS WORD
NEXT DITP (-1)
CURRENT REQ
REQ QUEUE HEAD
REQ QUEUE TAIL
HIGHEST
REQUEST URGENCY
LOWEST
DISCREQ
FLAGS
URGENCY
NEXT (Ø)
PREV (Ø)
NEXT
Ø
PREV
HEAD TABLE
N
SYSDB REL DITP
LDEV 2 DIT
FLAGS WORD
NEXT DITP
CURRENT REQ (Ø)
REQ QUEUE HEAD
REQ QUEUE TAIL
LDEV 4 DIT
FLAGS WORD
NEXT DITP
CURRENT REQ
REQ QUEUE HEAD
REQ QUEUE TAIL
SYSGLOB
%55
%56
%57
BUSY
HEAD
TAIL
TAIL TABLE
N
SYSDB REL DITP
LDEV 5 DIT
FLAGS WORD
NEXT DITP (-1)
CURRENT REQ
REQ QUEUE HEAD
REQ QUEUE TAIL

DIFFERENCES BETWEEN HP-IB AND SERIES III I/O

1. THE DRT STARTING VALUES IN F.L.M. ARE DIFFERENT FOR EACH SYSTEM.

   - SERIES III, FIRST AVAILABLE ENTRY AT %20.

   - SERIES 30/33/40/44, FIRST AVAILABLE ENTRY AT %40.

   - SERIES 64, FIRST AVAILABLE ENTRY AT %40. THE DRT CAN START IN ANY AVAILABLE LOCATION AS SPECIFIED BY WORD %10 AND %11 IN F.L.M.

2. THE DRT ENTRY FORMAT IS DIFFERENT.

   - SERIES III: WORD 0 = SIOP
     WORD 1 = PI
     WORD 2 = DBI
     WORD 3 = RESERVED

   - HP-IB: WORD 0 = SIOP
     WORD 1 = DBI
     WORD 2 = PI
     WORD 3 = CHANNEL FLAGS

3. THE DITS HAVE DIFFERENT FORMATS AND LENGTHS DEPENDING ON WHICH KIND OF SYSTEM. LOOK IN THE TABLES MANUAL FOR SPECIFIC CASES.

4. SERIES III USES A SIO PROGRAM TO HANDLE PROGRAMMED I/O, WHILE THE HP-IB SYSTEMS USE A CHANNEL PROGRAM. THE MAIN DIFFERENCE BETWEEN A SIO PROGRAM AND A CHANNEL PROGRAM IS THE LENGTH OF THE INSTRUCTION. A SIO PROGRAM INSTRUCTION IS 2 WORDS LONG WHILE A CHANNEL PROGRAM INSTRUCTION IS VARIABLE IN LENGTH, UP TO 5 WORDS.

5. THE DRIVERS FOR THE SERIES III AND THE HP-IB ARE DIFFERENT. THE SERIES III DRIVERS BUILD SIO PROGRAMS, WHILE THE HP-IB DRIVERS BUILD CHANNEL PROGRAMS.

6. SERIES III HAS LOOK AHEAD SEEKS TO DISC, SERIES 30/33 CAN ONLY PERFORM SERIAL DATA TRANSFERS TO DISC, AND THE SERIES 40/44/64 HAS OVERLAPPING SEEKS WHEN THERE IS MORE THAN ONE MASTER ON A GIC.

IOCDPNØ

IOCDPNØ IS AN UNSUPPORTED HP UTILITY/DIAGNOSTIC. ITS PURPOSE IS TO CALL ATTACHIO.

SOME OF THE REGULARLY USED COMMANDS ARE:

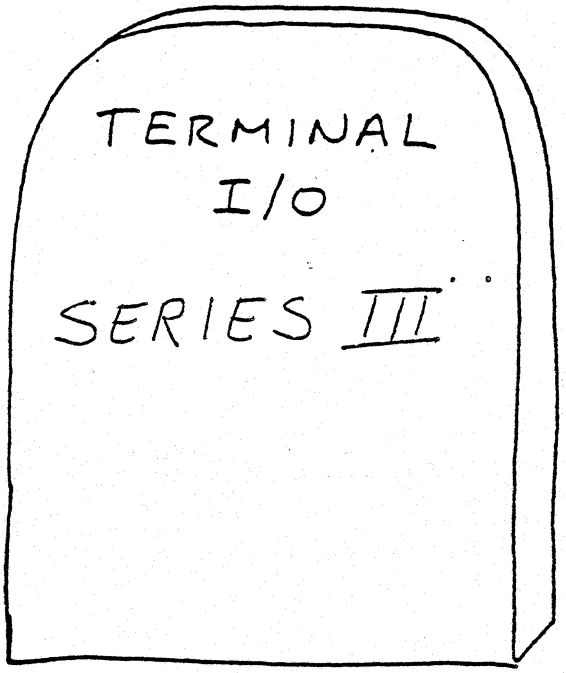
F __	FUNCTION : 0=READ, 1=WRITE, ETC.	
C __	COUNT : >0=WORD COUNT, <0=BYTE COUNT	
L __	LDEV	
LI __	LIST PROGRAM	
A "___"	ASCII-FILL DATA BUFFER	
B "___"	BINARY-FILL DATA BUFFER	(OCTAL)
DA	DISPLAY BUFFER IN ASCII	
DB	DISPLAY BUFFER IN BINARY	(OCTAL)
PA ____	PARM1	} DEVICE DEPENDENT eg. DISC ADDRESS
PB ____	PARM2	
E ____	CALL ATTACHIO __ TIMES (DEFAULT = 1)	
X	EXPLAIN COMMANDS	
EX	EXIT	

WARNING - DO <u>NOT</u> TYPE H (HELP TERMINAL) !!!

IOCDPNØ EXERCISE

USING IOCDPNØ, DO THE FOLLOWING:

1. WRITE A MESSAGE TO YOUR TERMINAL.
2. WRITE A MESSAGE TO SOMEONE ELSE'S TERMINAL IN THIS CLASSROOM.
3. READ SECTOR ZERO FROM THE SYSTEM DISC.
4. WRITE A MESSAGE TO THE LINE PRINTER.


TERMINAL
I/O
SERIES III

SE 335 TERMINAL I/O SECTION

OBJECTIVES:

TO BECOME ACQUAINTED WITH THE METHOD THE I/O SYSTEM USES TO HANDLE REQUESTS FOR TERMINALS.

TO RECOGNIZE THE DIFFERENCES BETWEEN TERMINAL I/O AND NON-TERMINAL I/O (I.E. DISC).

TO LEARN THE SPECIAL MEMORY STRUCTURES AND PROCESSES NECESSARY FOR TERMINAL I/O.

TERMINAL SOFTWARE

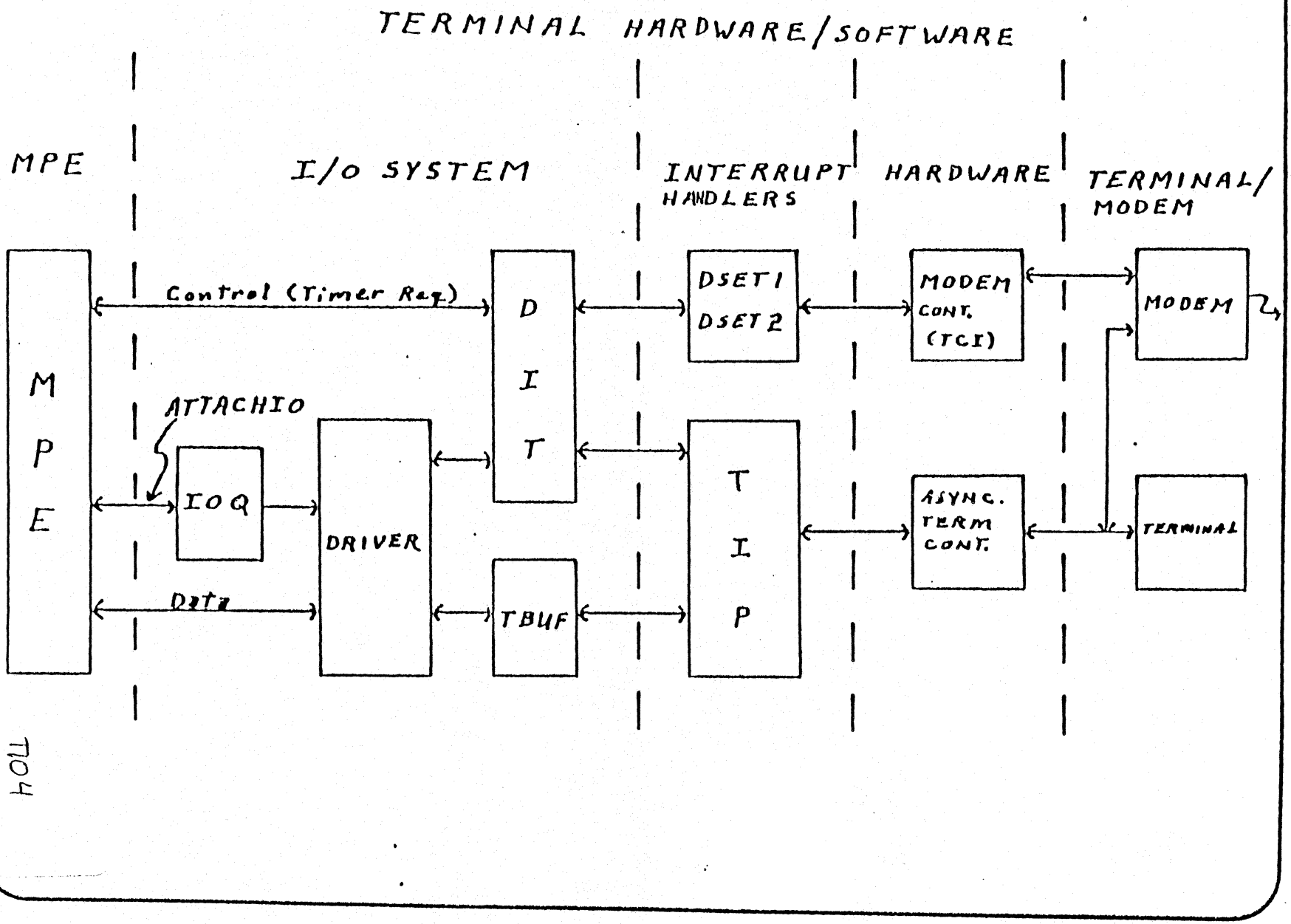
- DRIVER MOVES DATA TO/FROM TBUFS.

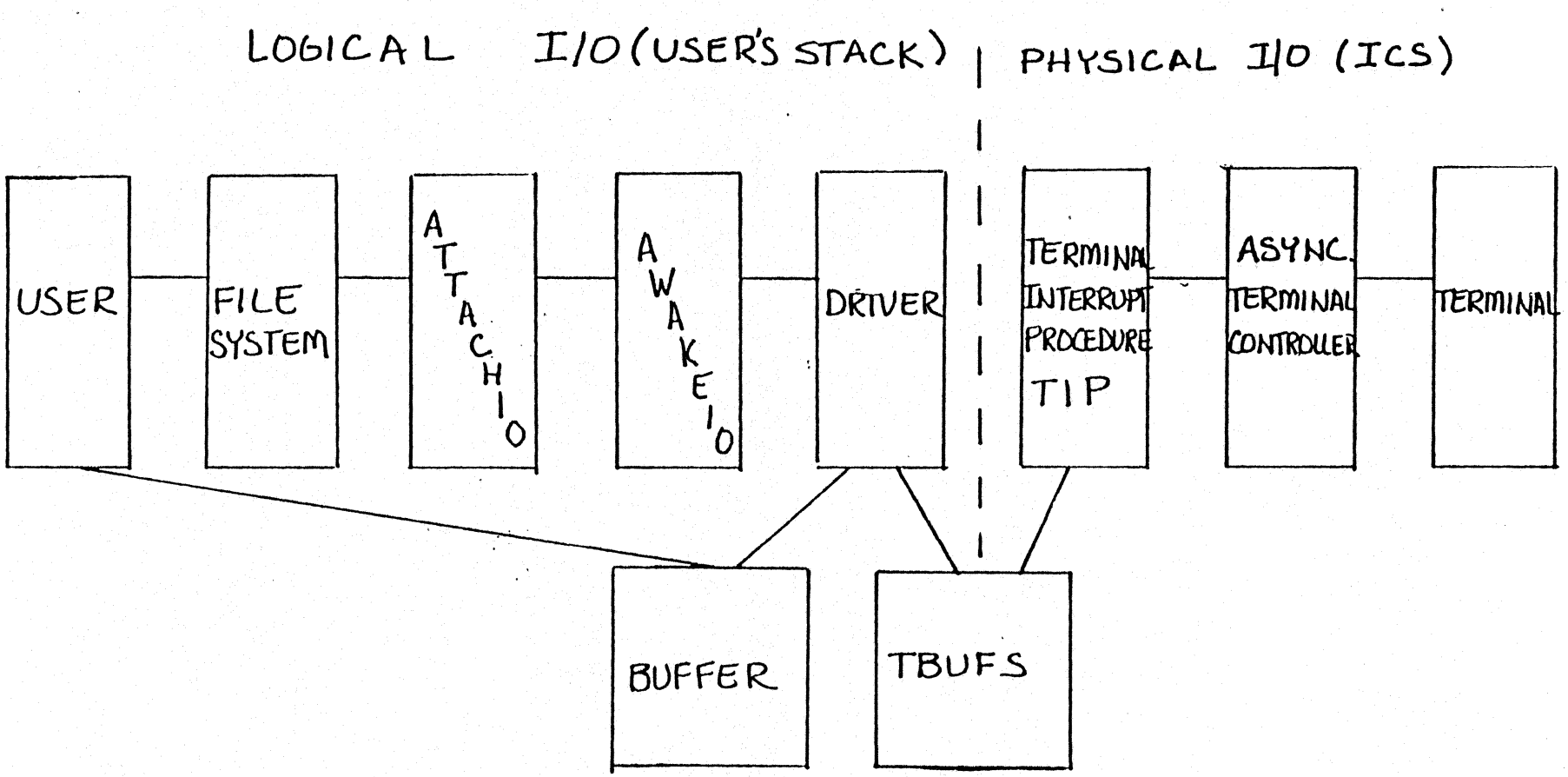
- READS/WRITES LOGICALLY DONE WHEN DATA IS MOVED TO/FROM THE USER STACK.

- INTERRUPT PROCESSOR (TIP) DOES THE PHYSICAL I/O.

- THE WRITE DATA IS CONCATENATED TO A MAX OF 270 BYTES. WHEN ABOUT 30 CHARACTERS ARE LEFT, MORE DATA IS MOVED TO THE TBUFS TO MAKE THE TERMINAL LOOK AS IF IT IS RUNNING CONTINUOUSLY.

- READ DATA IS FIRST PUT ON THE ICS, THEN IT IS MOVED TO THE TBUF.


TERMINAL HARDWARE/SOFTWARE
MPE
I/O SYSTEM
INTERRUPT HANDLERS
HARDWARE
TERMINAL/ MODEM
MPE
Control (Timer Req)
ATTACHIO
IOQ
DRIVER
Data
DIT
TBUF
DSET1
DSET2
TIP
MODEM CONT. (TCI)
ASYNC. TERM CONT.
MODEM
TERMINAL


LOGICAL I/O (USER'S STACK)
PHYSICAL I/O (ICS)
USER
FILE SYSTEM
ATTACHIO
AWAKEIO
DRIVER
TERMINAL INTERRUPT PROCEDURE TIP
ASYNC. TERMINAL CONTROLLER
TERMINAL
BUFFER
TBUFS

TERMINAL READ

USER STACK:

1. ATTACHIO FILLS IN AN IOQ ENTRY AND LINKS IT TO THE APPROPRIATE DIT. IF THIS IS THE FIRST OR A PREEMPTIVE REQUEST, ATTACHIO CALLS AWAKEIO. IF NOT, ATTACHIO CALLS WAIT.

2. AWAKEIO FINDS THE TERMINAL MONITOR PLABEL AND THEN CALLS TERMIOM.

3. TERMIOM DETERMINES IF THERE MAY BE A PREEMPTIVE REQUEST (DIT WORD %10(13:3)), TERMIOM SCANS THE LIST OF IOQ'S FOR THE HIGHEST PREEMPTIVE PRIORITY REQUEST (COMPARING IOQ "RPLEVEL" WORD 0 (13:3) AND DIT "LPLEVEL" WORD %10(0:1), IF THE CURRENT REQUEST IS OF LOWER PRIORITY THAN THE NEW PREEMPTIVE REQUEST, TERMIOM UPDATES THE LAST PREEMPTIVE LEVEL FIELD IN THE DIT TO THE NEW PREEMPTIVE LEVEL. TERMIOM THEN SETS THE DIT TO REFLECT A READ. TERMIOM CALLS THE PROCEDURE SENDSYNC, WHICH CAUSES THE ATC TO GENERATE AN INTERRUPT. TERMIOM THEN EXITS BACK TO ATTACHIO.

4. ATTACHIO CALLS WAIT FOR A BLOCKED REQUEST.

TERMINAL READ

ICS:

5. TIP IS STARTED, ON BEHALF OF THIS REQUEST, BY THE INTERRUPT GENERATED BY THE ATC. TIP CHECKS THE STATE OF THE DEVICE (DSTATE OF THE DIT), INITIALIZES THE DEVICE FOR THIS READ REQUEST AND SENDS A DC1 TO THE TERMINAL, TO LET IT KNOW THAT THE ATC IS READY TO RECEIVE DATA. TIP THEN IXITS FROM THE ICS.

6. WHEN A CHARACTER IS TYPED ON THE TERMINAL, THE ATC GENERATES AN INTERRUPT.

7. TIP FETCHS THE CHARACTER FROM THE ATC USING AN RIO DIRECT I/O COMMAND. TIP THEN SENDS A CIO TO THE ATC WHICH ACKNOWLEDGES THE INTERRUPT. TIP NEXT DETERMINES IF THE CHARACTER JUST FETCHED IS A SPECIAL CHARACTER, IF IT IS TIP PERFORMS SOME SPECIAL PROCESSING. TIP NOW PLACES THE CHARACTER INTO THE TBUF (IF REQUIRED TIP WILL GET A TBUF). FINALLY TIP IXITS FROM THE ICS UNLESS THE CONDITIONS OUTLINED BELOW ARE MET.

8. IF THE READ IS COMPLETED (I.E., THE BYTE COUNT IS REACHED OR A TERMINATION CHARACTER, "CR", IS TYPED), TIP MODIFIES THE IOQ TO REFLECT THE COMPLETION AND THEN CHECKS IF TERMIOM IS CURRENTLY ACTIVE. IF IT IS, TIP SETS THE REQUEST BIT IN THE DIT WORD 0(3:1) OTHERWISE, TIP CALLS AWAKETERMINAL, WHO AWAKENS A PROCESS TO RUN TERMIOM ON BEHALF OF THE REQUESTING PROCESS. TIP THEN IXITS FROM THE ICS.

EACH SUCCESSIVE CHARACTER CAUSES ANOTHER INTERRUPT. REPEAT AT STEP 6.

TERMINAL READ

SOME PROCESS' STACK:

9. ATTACHIO CALLS AWAKEIO, WHO CALLS TERMIOM.

10. TERMIOM SETS THE ACTIVE BIT IN THE DIT. TERMIOM DETERMINES WHICH REQUEST TO HANDLE BY LOOKING IN THE "COMMUNICATIONS TO MONITOR FROM TIP" CELL (DRQST, WORD 6 OF THE DIT). SOME CHECKING IS PERFORMED IN THE IOQ "REQUEST STATE" FIELD. AND TERMIOM SEES THAT A READ HAS COMPLETED. THE IOQ CONTAINS THE ADDRESS OF THE DATA SEGMENT WHERE THE DATA IS TO BE PLACED. TERMIOM THEN TRANSFERS THE DATA FROM THE TBUFS TO THE USER'S STACK OR AN XDS. IF THERE WERE NOT ANY READ ERRORS, TERMIOM UPDATES THE IOQ COMPLETION BIT AND EXITS BACK TO ATTACHIO.

11. ATTACHIO RETURNS THE IOQ AND PASSES THE COMPLETION STATUS TO THE FILE SYSTEM, WHO DELIVERS IT TO THE USER.

TERMINAL WRITE

USER STACK:

1. ATTACHIO FILLS IN AND LINKS THE IOQ ENTRY TO THE DIT. IF THIS IS THE FIRST/PREEMPTIVE REQUEST, ATTACHIO CALLS AWAKEIO. OTHERWISE, ATTACHIO CALLS WAIT FOR A BOCKED I/O REQUEST.

2. AWAKEIO FINDS THE MONITOR PLABEL IN THE DLT AND CALLS TERMIOM (THE MONITOR/DRIVER IN THE MODULE IOTERMO FOR SERIES III OR HIOTERMO FOR HP-IB).

3. TERMIOM GETS THE TBUFS (UP TO 9 AT A TIME, 30 CHARACTERS EACH), FILLS THEM WITH CHARACTERS AND LINKS THEM TO THE DIT. WHEN THE TBUFS ARE FULL, TERMIOM UPDATES THE STATE FIELD OF THE DIT AND CALLS THE PROCEDURE SENDSYNC, WHICH SENDS A SYNC CHARACTER AND A CIO COMMAND TO THE ATC, ALLOWING IT TO GENERATE AN INTERRUPT TO START TIP AND TO GENERATE AN INTERRUPT AFTER WRITING EACH CHARACTER TO THE TERMINAL.

4. IF ALL OF THE CHARACTERS TO BE WRITTEN ARE PLACED INTO THE TBUFS, TERMIOM SETS THE IOQ FLAGS TO REFLECT THE COMPLETION AND EXITS TO ATTACHIO, WHO RETURNS THE IOQ AND EXITS. IF THERE ARE STILL MORE CHARACTERS TO BE PLACED INTO THE TBUFS, TERMIOM EXITS TO ATTACHIO, WHO CALLS WAIT.

TERMINAL WRITE

ICS:

5. TIP RECEIVES THE INTERRUPT AND TAKES ONE CHARACTER FROM THE TBUF, TIP CHECKS IF IT IS A SPECIAL CHARACTER, IF NOT TIP SENDS IT TO THE ATC (USING A WIO COMMAND) AND IXITS (UNTIL CONDITIONS OCCUR THAT ARE OUTLINED BELOW).

6. THE ATC INTERRUPTS THE CPU FOR THE NEXT CHARACTER. REPEAT AT STEP 5.

7. TIP RELEASES THE TBUFS AS THEY ARE EXHAUSTED. WHEN THERE ARE ONLY 2 TBUFS LEFT, TIP CHECKS IF THERE IS ANY MORE DATA TO BE PUT INTO THE TBUFS BY THE MONITOR. IF THERE IS SOME MORE DATA, TIP CALLS AWAKETERMINAL. IF NOT, REPEAT AT STEP 5.

8. TO COMPLETE THIS REQUEST, AWAKETERMINAL FINDS THE APPROPRIATE PROCESS TO RUN TERMIOM. AWAKETERMINAL REQUESTS THE AWAKENING OF THE WAITING PROCESS AND RETURNS TO TIP.

9. TIP CONTINUES TO TRANSFER THE REMAINING CHARACTERS TO THE ATC. REPEAT AT STEP 5.

10. WHEN THE BYTE COUNT REACHES ZERO, TIP CHECKS IF THERE ARE ANY PENDING REQUESTS FOR THIS PORT. IF THERE ARE, TIP CALLS AWAKETERMINAL, WHO REQUESTS THE AWAKENING OF THE PROCESS TO HANDLE THE NEXT REQUEST. IF THERE ARE NOT ANY PENDING REQUESTS, TIP IXITS FROM THE ICS.

TERMINAL WRITE

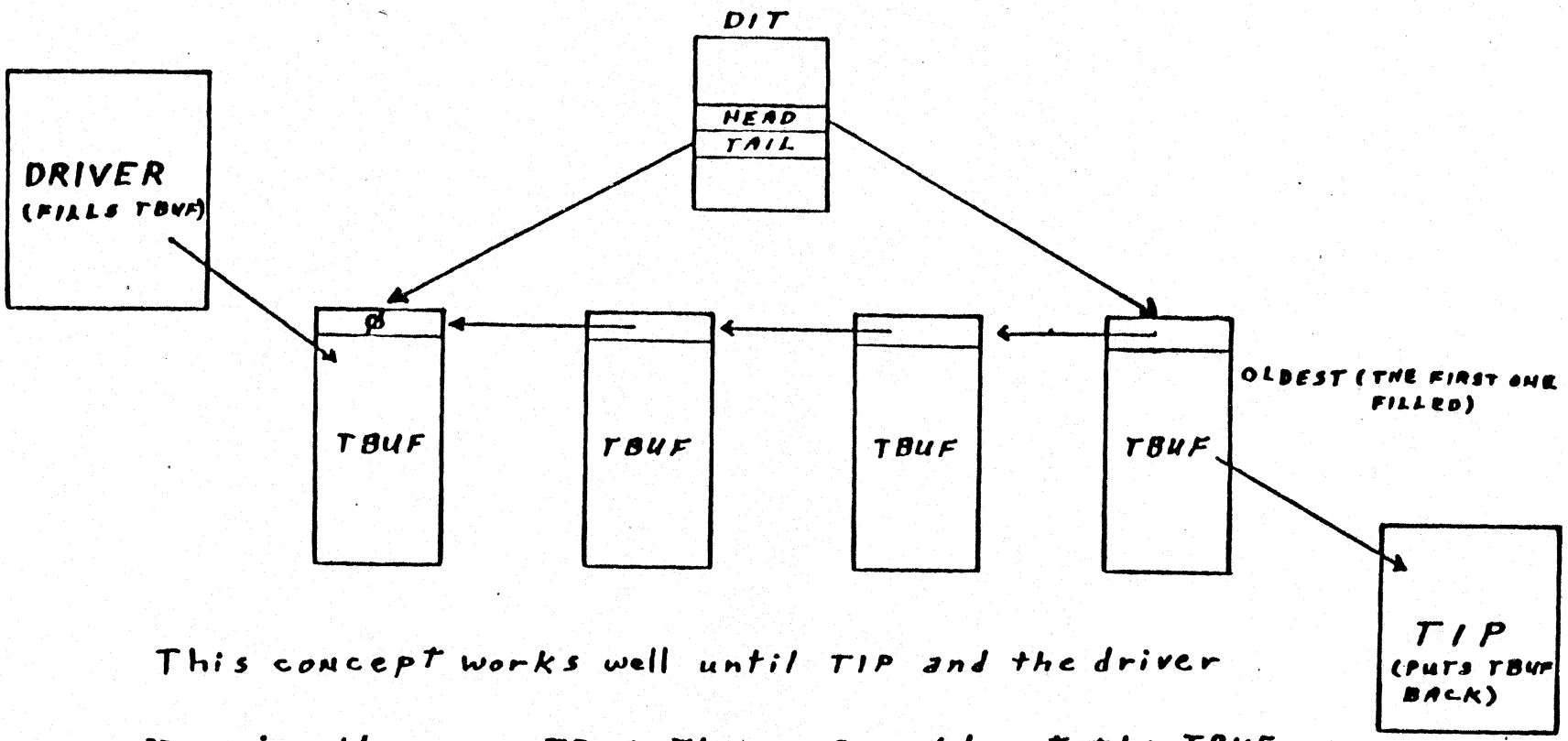
SOME PROCESS STACK:

11. IF THERE ARE ANY REMAINING CHARACTERS TO BE TRANSFERED, ATTACHIO CALLS CALLS AWAKEIO, WHO IN TURN CALLS THE TERMINAL MONITOR, TERMIOM.

12. TERMIOM INSERTS THE REMAINING CHARACTERS INTO THE TBUFS , SETS THE IOQ FLAGS TO REFLECT THE COMPLETION OF THE REQUEST AND EXITS BACK TO ATTACHIO.

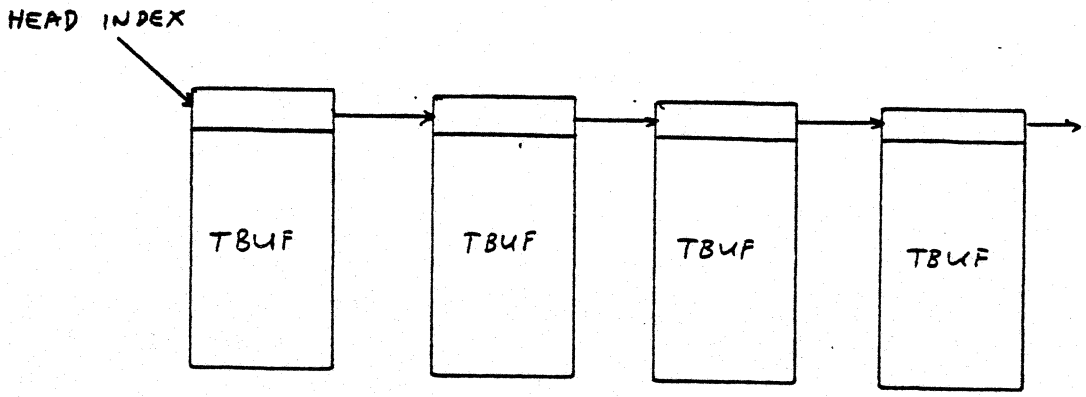
13. ATTACHIO RETURNS THE IOQ TO THE FREE LIST AND EXITS BACK TO THE FILE SYSTEM, WITH COMPLETION STATUS RETURNED.

TERMINAL BUFFER HANDLING (WRITING)



This concept works well until TIP and the driver are using the same TBUF. Then, TIP could put the TBUF away while the driver is trying to fill it. (The driver did not keep ahead because DISP did not run it.) The locking mechanism to stop this is DIT(9).(10:1) (filling) for TIP and DIT(18)=-2 for IOTERMØ.

WHEN TBUF'S ARE FREED, THEY ARE RETURNED TO THE FREE LIST.



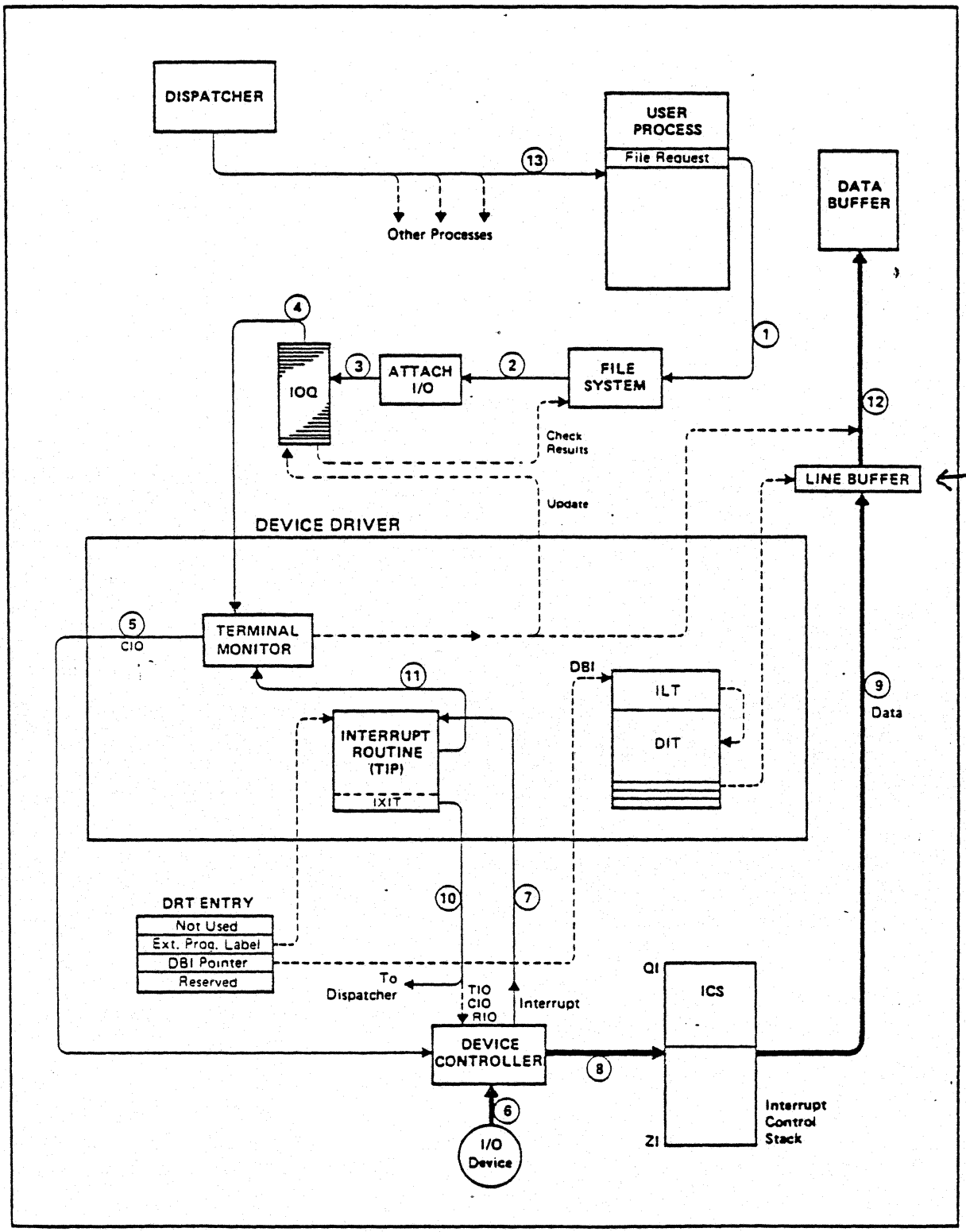
ARE BLOCK MODES REALLY DIFFERENT?

NO.

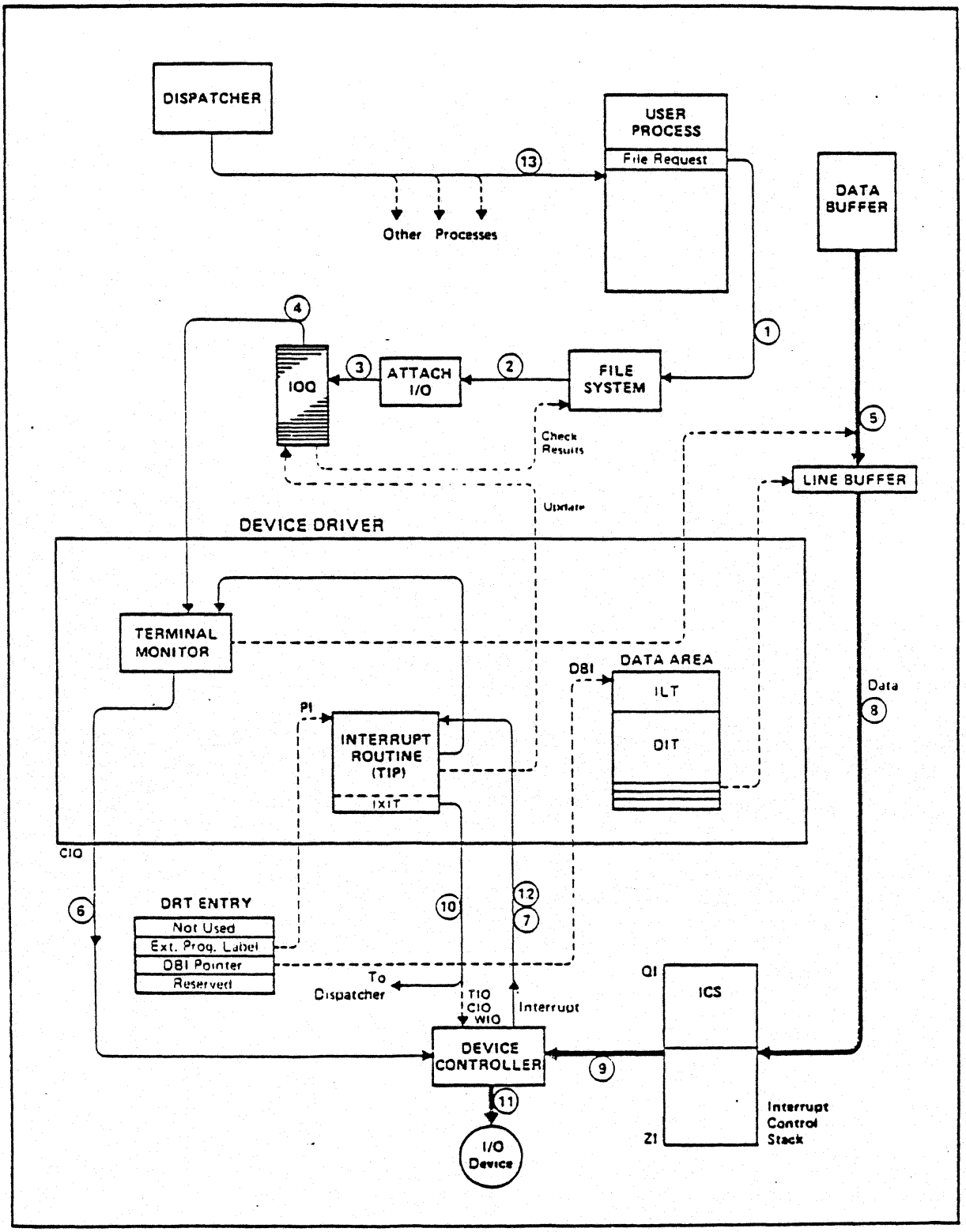
- THE TRANSMISSION OF CHARACTERS DOES NOT BEGIN UNTIL THE "ENTER" KEY IS DEPRESSED.

- EACH CHARACTER STILL INTERRUPTS THE CPU.

- THE NUMBER OF TERMINALS RUNNING IN BLOCK MODE CONCURRENTLY IS LIMITED BY THE BANDWIDTH OF THE CHARACTER BY CHARACTER INTERRUPT PROCESSING.



Direct Read For Terminal Devices



Direct Write For Terminal Devices

SYSTEMS WITH ADCC'S

THERE ARE A FEW MINOR DIFFERENCES TO THE SEQUENCE DESCRIBED PREVIOUSLY FOR THE SERIES III.

1. TERMIOM IS LOCATED IN THE MODULE HIOTERM0.

2. THE TBUFS HOLD 60 CHARACTERS INSTEAD OF 30, FOR THE SERIES 30/33 AND 40/44.

3. THE EXECUTION OF THE TRANSFER ON A SERIES III IS DONE BY DIRECT I/O COMMANDS ("CIO", "WIO" AND "RIO"). IN A SYSTEM WITH ADCC'S, THE TRANSFER OF CHARACTERS IS DONE BY THE MICRO-CODE IN THE CHANNEL PROGRAM PROCESSOR ON THE CPU, BY EXECUTING A CHANNEL PROGRAM. THE CHANNEL PROGRAM WILL TRANSFER ONE CHARACTER PER EXECUTION.

SYSTEMS WITH ATP'S

THERE ARE MAJOR DIFFERENCES BETWEEN TERMINAL I/O ON SYSTEMS WITH ATC'S OR ADCC'S VERSUS SYSTEMS WITH ATP'S (44 OR 64) .

1. THE TERMINAL I/O IS DONE IN GENERAL BY DMA TRANSFER, WITHOUT INTERRUPTING THE CPU AFTER EACH CHARACTER. THE CPU IS ONLY INTERRUPTED WHEN THE TBUF IS FILLED, OR AT THE END OF A TRANSACTION.

2. THE TERMINAL MONITOR, TERMMON, IS A STAND-ALONE MODULE. IT PERFORMS SIMILAR FUNCTIONS AS TERMIOM. THERE ARE TWO NEW SOFTWARE PROCEDURES TO HANDLE TERMINAL I/O FOR ATP SYSTEMS: PDMANAGER, WHO SETS THE PHYSICAL TABLES UP; AND ATPDRIVER, WHO BUILDS THE CONTROL PROGRAMS, EXECUTES A START CONTROL PROGRAM AND HANDLES THE COMPLETION. TIP IS NO LONGER USED.

3. THERE ARE A MAXIMUM OF 5 TBUFS AVAILABLE FOR EACH TERMINAL WRITE, WITH 134 CHARACTERS PER TBUF.

HEWLETT hp PACKARD

I/O EXERCISE

THE INSTRUCTOR HAS AN I/O REQUEST POSTED TO A CERTAIN TERMINAL. GIVEN THE LDEV, YOU ARE TO USE DEBUG AND FIND THE FOLLOWING:

LDEV = _____

1. WHAT IS THE SYSGLOB-RELATIVE ADDRESS OF THE DIT FOR THIS LDEV? _____

2. WHAT FUNCTION-STATE IS THE DEVICE CURRENTLY IN ? (READ, WRITE, NULL, ETC.) _____

3. WHAT IS THE SYSGLOB-RELATIVE ADDRESS OF THE CURRENT IOQ? _____

4. FOR THIS IOQ, WHAT IS THE FUNCTION? _____
   REQUESTED COUNT? _____ PCB NO.? _____

5. ARE THERE ANY OTHER IOQ'S FOR THIS DEVICE? _____
   IF SO, WHAT ARE THEIR FUNCTIONS, COUNTS, PCB'S ?

I/O EXERCISE

5. (Cont.) _____

_____

_____

_____

6. WHAT WAS ENTERED ON LDEV ____ .

_____

_____

I/O SYSTEM

CHECK FOR UNDERSTANDING

1. WHAT IS THE SOFTWARE LINKAGE BETWEEN THE FILE SYSTEM AND THE I/O SYSTEM?

   attachio

2. IN GENERAL TERMS, WHAT IS THE MAIN WORK DONE BY ATTACHIO?

3. WHAT KIND OF REQUEST QUEUE WILL THE FOLLOWING RESOURCE REQUESTS BELONG TO (DRQ OR IOQ)?

   [illegible] ____
   [illegible] ____
   [illegible] ____

4. WHAT IS THE MAJOR FUNCTION OF THE MONITOR?

5. WHAT ARE THE MAJOR FUNCTIONS OF THE DRIVER?

   handshake

6. WHAT ARE THE TWO PARTS OF THE DRIVER CALLED?

   initiator, capture.

7. WHAT IS THE NAME OF THE INTERRUPT PROCEDURES FOR TERMINALS AND SIO DEVICES?

   [illegible] Tip Gnp

8. WHAT IS AN SIO/CHANNEL PROGRAM?

9. HOW MANY COPIES OF EACH TABLE DO YOU GET ON A SYSTEM? IF THERE IS ONLY 1 COPY PER SYSTEM? HOW MANY ENTRIES PER SYSTEM?

   DRT one per IDev
   ILT
   DIT
   IOQ one per driver
   LPDT
   DLT

10. WHEN THE PHYSICAL I/O TRANSFER COMPLETES, WHAT IS THE EVENT THAT CAUSES THE INTERRUPT FROM THE DEVICE.

11. HOW CAN YOU TELL IF A PROCESS IS WAITING FOR I/O?

wake mask.

TERMINAL I/O

REFERENCE SECTION

TERMINAL I/O OVERVIEW

INITIAL AWAKENS PROGENITOR

PROGEN

Check terminal DIT to determine one-time initialization required. Calls TERMINIT.

TERMINIT

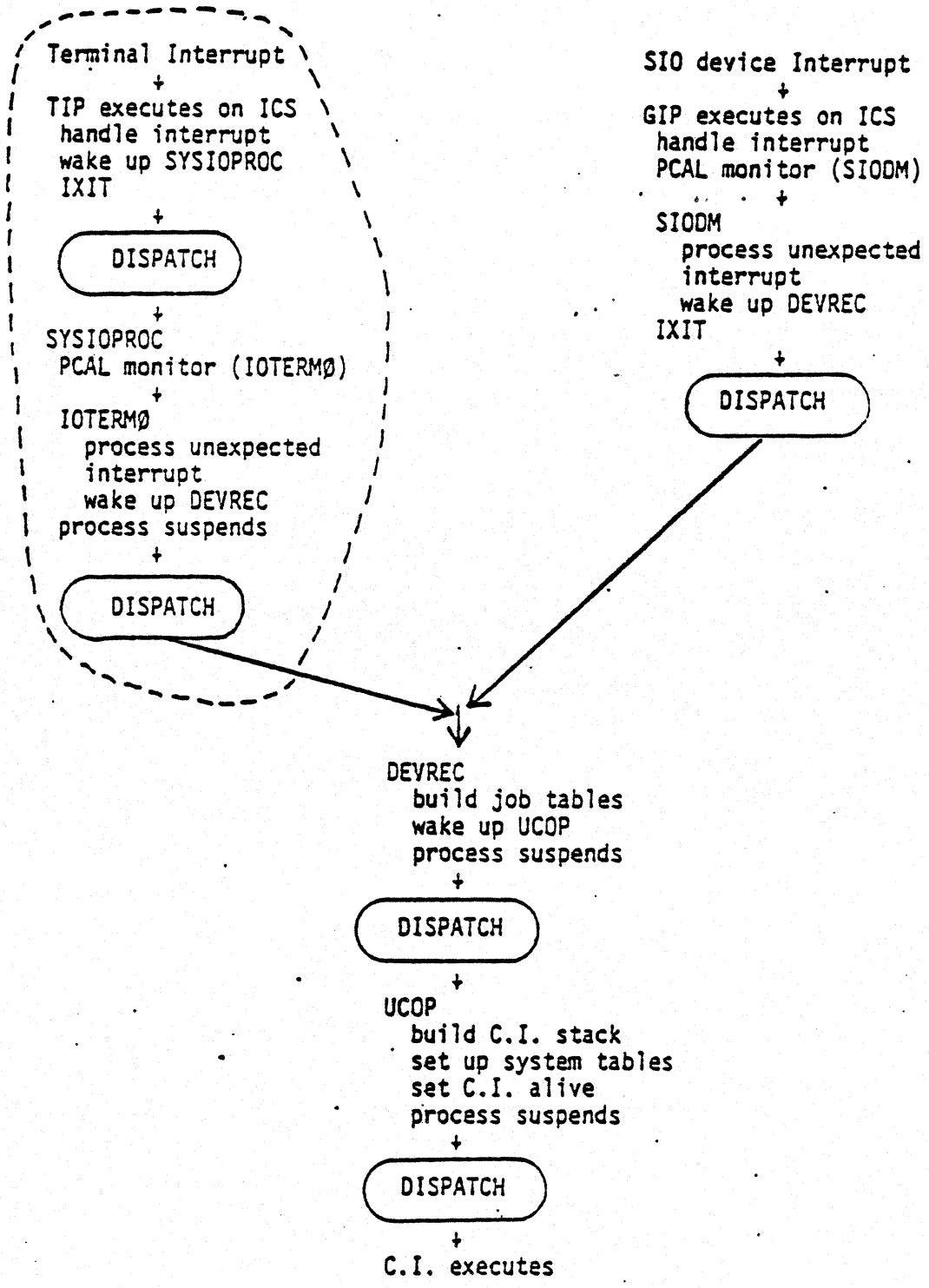
Iterates through the DITs and initializes devices to:

- 240 CPS initially
- Sets READ state at the device
- Sets diagnos mode at controller (for device).
- Sets configuration defaults in DIT
- Sets DIT to null activity state and sets diagnose flag
- Any errors while initializing device causes an eventual SUDDENDEATH 201

PROGEN then completes initialization of other devices. Terminal devices are now ready for use.

A user must now depress the "CR" key at a terminal which will cause an interrupt. On interrupt, microcode sets stack to ICS and code to TIP/TIPX.

Interrupt-Command Interpreter Overview



TERMINAL I/O OVERVIEW (PAGE 2)

TIP

SETS DB TO SYSDB, READS CHARACTER FROM DEVICE AND SAVES IT IN STACK. SETS UP LOCAL STACK VARIABLES, CHECKS FOR SPECIAL INTERRUPT PROCESSOR AND CALLS IT IF REQUIRED. ACKNOWLEDGE INTERRUPT TO CONTROLLER (SCANNING ON CONTROLLER MAY NOW RESUME). CHECKS FOR OUTPUTTING TO DEVICE. IF NOT OUTPUT, BRANCHES TO INPUT SECTION WHERE CHECKS ARE MADE FOR NUMEROUS SPECIAL CASES INCLUDING "TERMINAL UP". (NOTE: IN THE CASE OF THE CONSOLE DEVICE THE "UP" FLAG IS ALWAYS SET.) A CHECK IS MADE FOR A "MAIN" UNIT (0-15). TERMINALS IN DIAGNOSE MODE ARE SECONDARY UNITS (16-20). A TEST IS THEN MADE AGAINST THE INPUT CHARACTER TO DETECT "CR". IF FAIL THEN IXIT ELSE CONTINUE. ON "CR" THE "SPEED" (SECONDARY UNIT CALCULATED INDEX) NUMBER IS SAVED IN THE DIT, PARITY CHECK FLAG IS SET, AND THE "REQUEST" FOR SERVICE FLAG IS SET. TIP THEN CALLS AWAKETERMINAL. ON RETURN, AN IXIT IS DONE.

AWAKETERMINAL

CHECKS FOR AN IOQ ASSOCIATED WITH THE REQUEST. IF NO IOQ, IT ASSUMES THE CALL IS FOR UNBLOCKED I/O REQUEST.

AWAKEIO

CHECKS MONITOR TYPE. TERMINAL MONITOR IS TYPE 2. (IT MUST NEVER EXECUTE ON ICS.) AN AWAKE CALL IS MADE. CERTAIN CHECKS ARE ALSO MADE TO DETERMINE IF MONITOR CALL COULD BE IMPEDED. IN THIS PARTICULAR CASE, NO IMPEDING IS REQUESTED.

AWAKE

SETS PCB ACTIVE BIT FOR SYSIOPROC. PLACES PCB IN READY LIST. NOTIFIES DISPATCH IF SYSIOPROC IS HIGHER PRIORITY THAN CPCB. RETURNS ARE THEN MADE ALL THE WAY BACK TO TIP WHICH WILL IXIT.

TERMINAL I/O OVERVIEW (PAGE 3)

DISPATCH

Launches SYSIOPROC in its own process.

SYSIOPROC

Calls AWAKEIO to start the monitor.

AWAKEIO

Again checks monitor type. Finds it is on "user" stack and calls the monitor.

IOTERMO/TERMIOM

Checks for another copy of monitor running on another stack. If there is a copy active elsewhere, sets "REQUEST" flag and EXIT else continue. Checks "REQUEST" flag for something to do. If not "REQUEST" then EXIT. TIP has set "REQUEST" so monitor clears it and continues. Sets local stack variables, checks for IOQ with this request. If no IOQ, sets temporary "NULL" state and invalid function in local variables. Checks for device "UP". Device not "UP", so some additional housekeeping of the DIT is performed. Then, "SERVICE REQUESTS" are checked. TIP has set a request for device to be put "ONLINE". Monitor executes this request. Tests to determine if device is already "OWNED" by system and to ensure device is job/session or :data accepting. Monitor then sets DEVREC service request flag in LPDT, increments global devrec request count, sets mode in DIT to "LOGGING_ ON". A LON-ON timeout is initiated for the system configured value.

TERMINAL I/O OVERVIEW (PAGE 4)

SETTERMTYPE

Validates correct terminal type against the actual speed saved in the DIT. Returns error condition if the speed is not capable for term type specified. Monitor checks for error. If none, then sets terminal "UP" flag in DIT, resets "SPEED SENSING" flag which indicates device is now a "MAIN" unit then initializes both send and receive channels on the device. This is necessary in order to get the device out of "diagnose" mode. The monitor then is almost complete except that now DEVREC must now run.

AWAKE

Sets PCB ACTIVE bit for devrec. Places PCB in Ready list. Notifies DISPATCHER if DEVREC is higher priority than CPCB. Returns are then made all the way back to SYSIOPROC which then suspends.

DISPATCH

Launches DEVREC in its own process.

DEVREC

DEVREC calls ATTACHIO to output the prompt on the terminal.

ATTACHIO

Sets SYSDB. Validates logical device requested. Allocates IOQ entry. Sets passed paremeters from user into IOQ. Sets PCB of requestor in IOQ. Links IOQ to DIT.

AWAKEIO

Checks monitor type. Terminal monitor runs on user stack or in own process. This is a user stack, so PCAL of monitor is done.

TERMINAL I/O OVERVIEW (PAGE 5)

IOTERMO

IF ANOTHER COPY OF MONITOR IS RUNNING, THE "REQUEST" FLAG IS SET AND THEN EXIT. MONITOR THEN PERFORMS HOUSEKEEPING. CHECKS FOR VALID IOQ. CHECKS FOR VALID FUNCTION CODE (WRITE IN THIS CASE) IN IOQ. TESTS FOR DEVICE "UP" AND ANY "SERVICE REQUESTS" PENDING. TESTS FOR REQUEST TO BE ABORTED, PREEMPT LEVELS IN EFFECT OR PREEMPT LEVELS BEING REQUESTED (IOQ PARAMETER). CHECKS DIT STATE AND EXECUTES SYNDROME SPECIFIED BY "STATE" (NEW OR NULL). FURTHER CHECKS ARE MADE FOR CONSOLE DEVICE. CERTAIN SPECIAL FLAGS ARE CHECKED. IF A NEW LINE IS AT THE DEVICE, MONITOR THEN SETS UP APPROPRIATE PRE/POST SPACING. A NEW SYNDROME CALLED "NOTNEW" IS SET IN STATE AND PERFORMED. THIS CAUSES THE IOQ FUNCTION SYNDROME (WRITE) TO BE PERFORMED. A TERMINAL BUFFER IS ALLOCATED AND THE USER DATA MOVED TO THE TANK (MAX 270 BYTES). THE BANDWIDTH IS THEN CHECKED TO SEE IF THE MAXIMUM CONCURRENT TERMINAL I/O HAS BEEN EXCEEDED. IF IT HAS, THE REQUEST IS QUEUED AND THE MONITOR EXITs ELSE CONTINUE. AN INTERRUPT IS PROVOKED (BY SENDING A "NULL" CHARACTER TO THE TERMINAL) THROUGH MPXCONTROL TO GET TIP TO START THE OUTPUT. MONITOR THEN RETURNS ALL THE WAY BACK WITH A GOOD COMPLETION. NOTE: ACTUAL PHYSICAL OUTPUT HAS NOT BEEN ACCOMPLISHED AT THIS TIME, BUT IN ALL RESPECTS FOR THE USER OUTPUT IS DONE.

TERMINAL I/O OVERVIEW (PAGE 6)

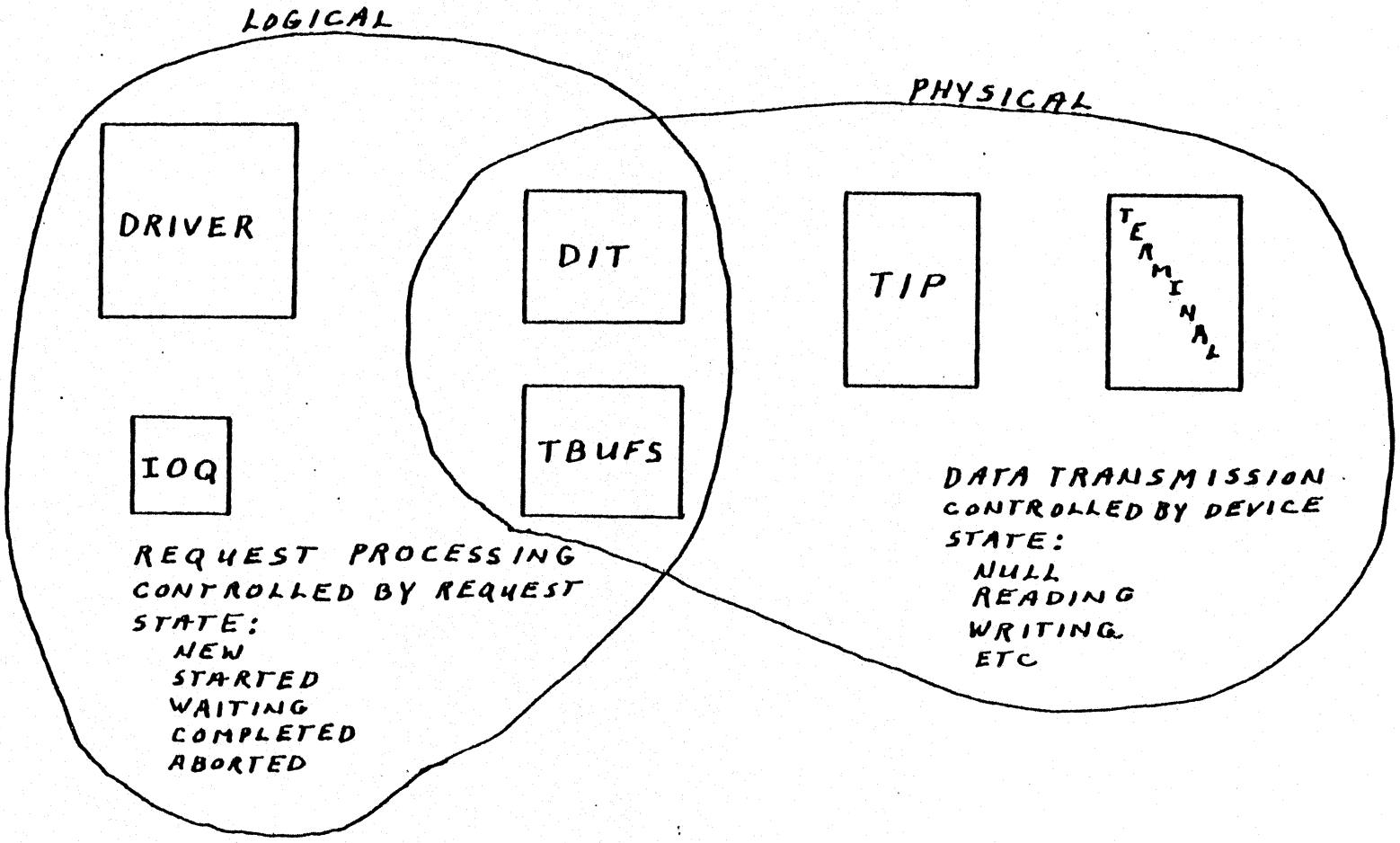
INTERRUPT

TIP

SETS DB TO SYSDB. READS CHARACTER FROM DEVICE AND SAVES IN ICS (MAY BE INVALID OR SPECIAL CHARACTER). SETS UP LOCAL VARIABLES, CHECKS FOR SPECIAL INTERRUPT PROCESSOR AS BEFORE. ACKNOWLEDGES INTERRUPT. CHECKS FOR OUTPUTTING TO DEVICE. THEN EXECUTES THE OUTPUT SECTION OF TIP. TESTS ARE MADE FOR END-OF-WRITE, END TBUF AND CERTAIN SPECIAL FLAGS. THE COUNT IS DECREMENTED FOR # CHARACTERS, THE CHARACTER TO BE OUTPUT IS OBTAINED AND LOGICALLY OR'D WITH AN OUTPUT CONTROL WORD. THE CHARACTER IS THEN OUTPUT TO THE DEVICE AND AN IXIT IS DONE.

WHEN THE OUTPUT IS COMPLETED, AN INTERRUPT AGAIN OCCURS AND TIP IS ENTERED. TIP DOES ALL OF THE CHECKS OUTLINED ABOVE AND THEN GOES TO THE OUTPUT SECTION TO OUTPUT ANOTHER CHARACTER. EVENTUALLY THE END-OF-WRITE CHECK PROVES TRUE (COUNT GOES TO 0) AND DIT STATE IS SET TO "NULL". TERMINAL BUFFERS ARE RELEASED AND A CHECK IS MADE TO SEE IF ANY MORE REQUESTS ARE QUEUED FOR THE DEVICE. IF THERE ARE REQUESTS, THEN THE NEXT REQUEST IN THE QUEUE IS PULLED UP (BY AN EXTERNAL PROCEDURE). TIP IS THEN RE-EXECUTED. IF NO MORE REQUESTS, THE INTERRUPT PROCESSOR DOES AN IXIT. DISPATCH AWAKENS THE REQUESTING PROCESS AND EXITS ARE THEN EXECUTED ALL THE WAY BACK TO ATTACHIO. ATTACHIO KNOWS THAT COMPLETION FOR "WRITE" WAS DONE, SO IT GIVES BACK THE IOQ AND EXITS ARE DONE BACK TO THE USER CODE.

TERMINAL I/O PROCESSING



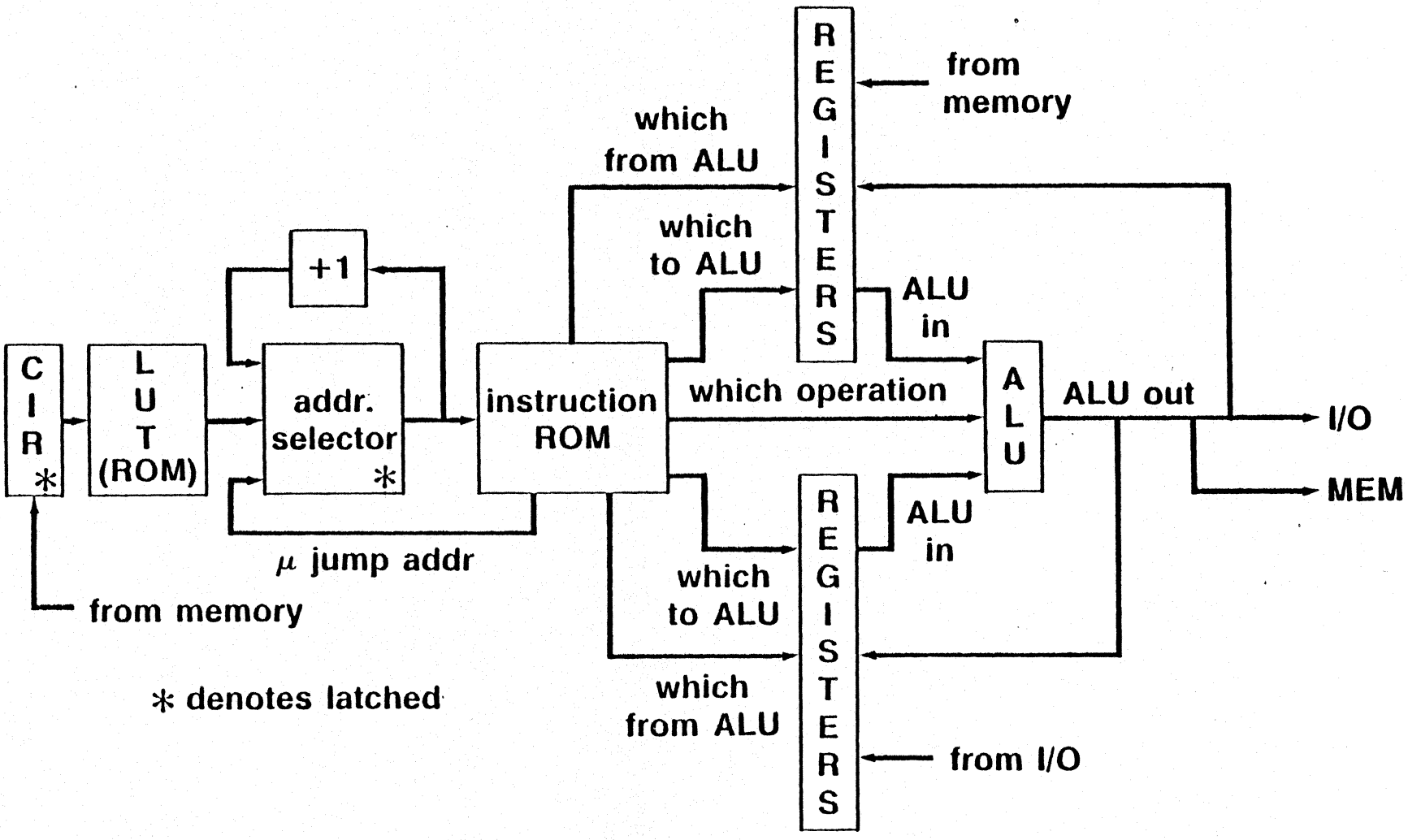
HEWLETT PACKARD

HARDWARE OVERVIEW

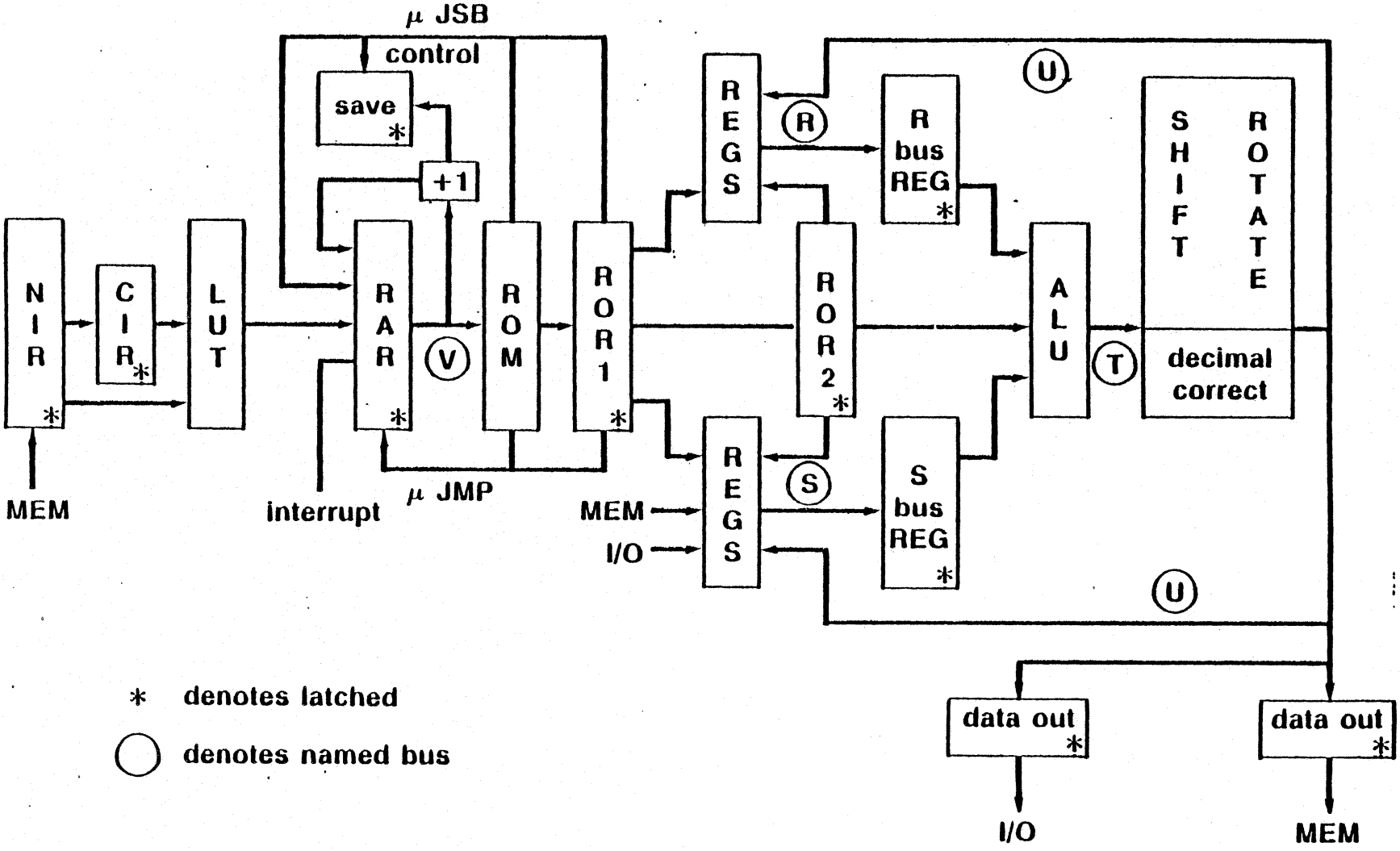
SERIES II/III

SERIES 30/33/44

hypothetical microprogrammed CPU

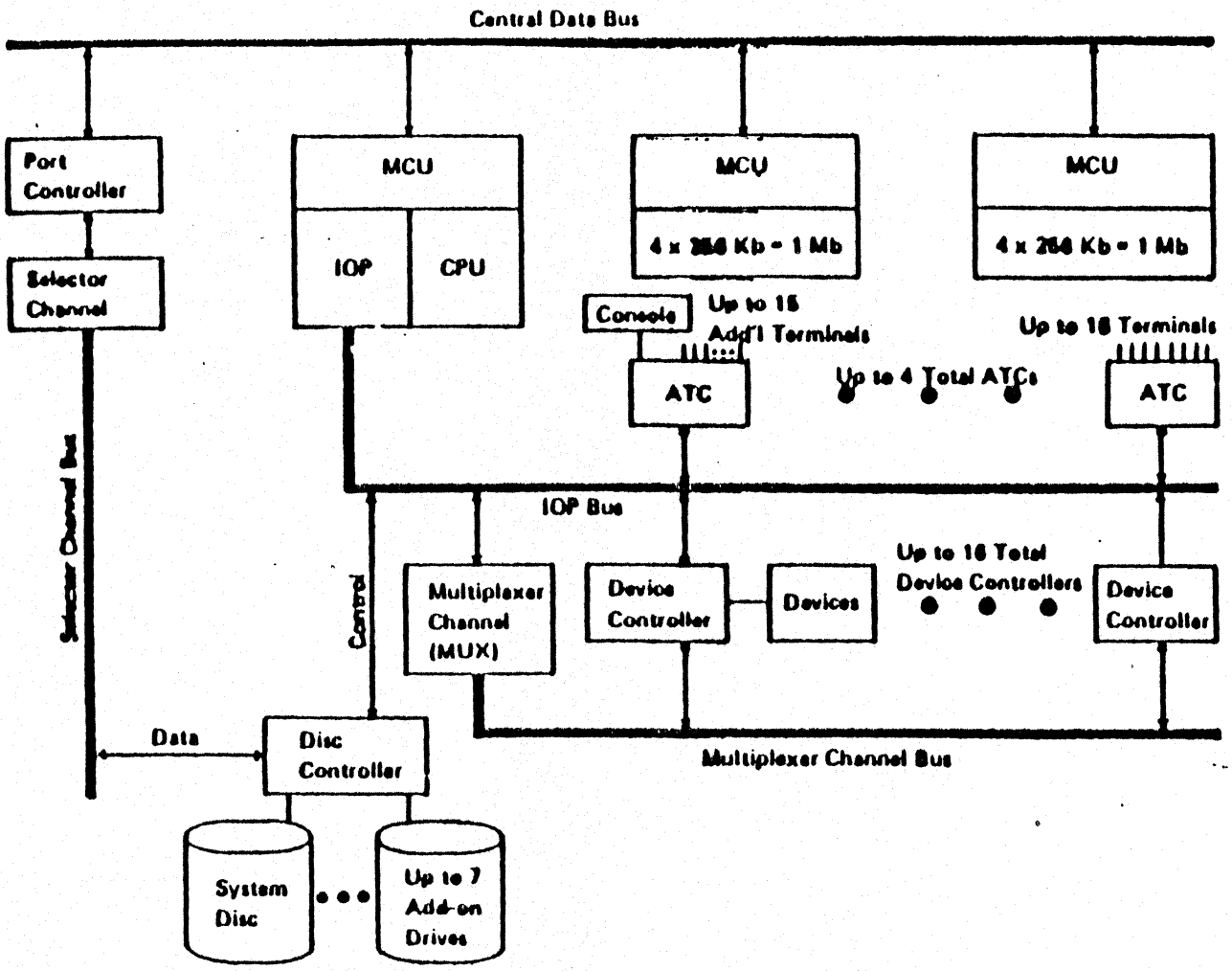


supercharged CPU



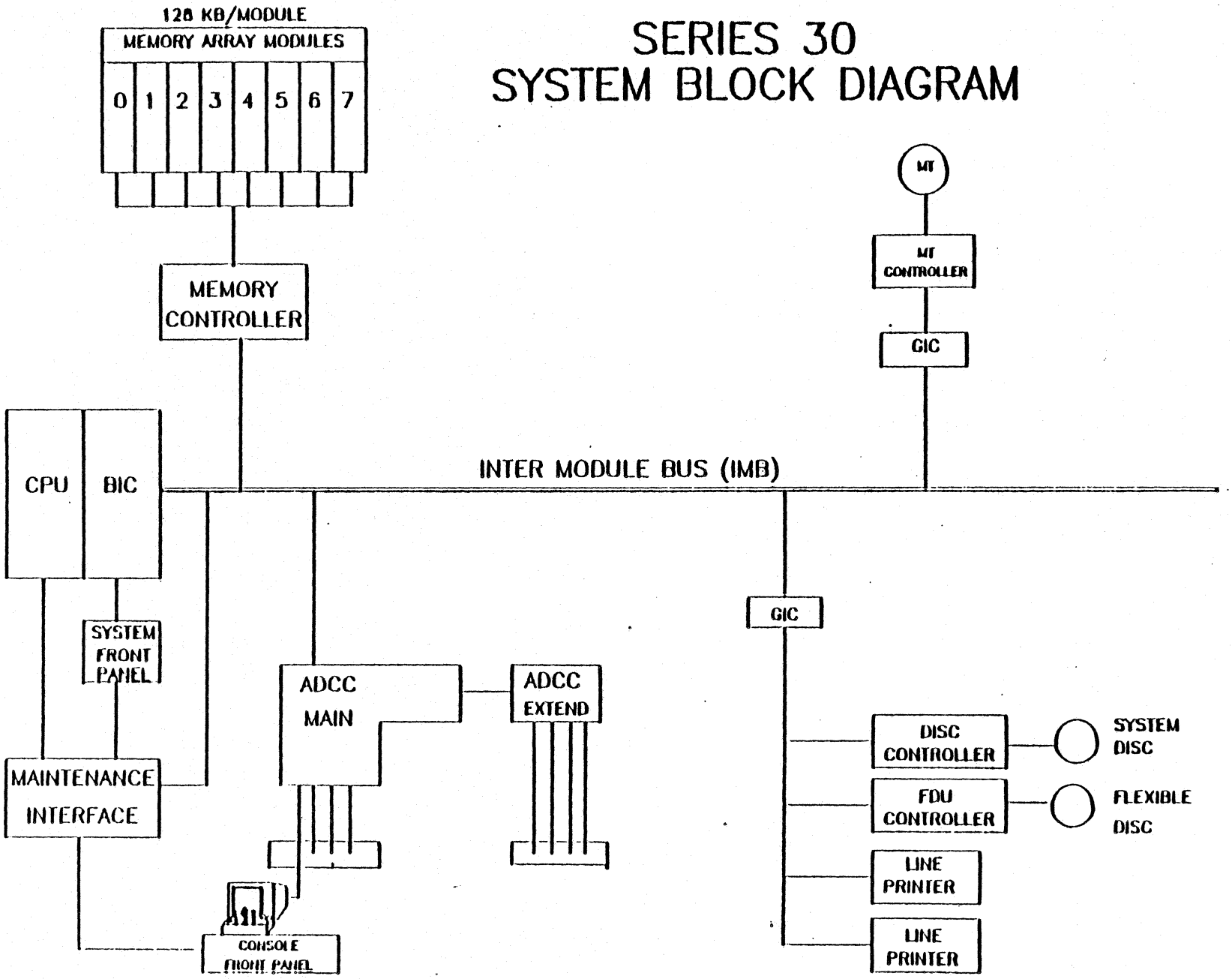
HEWLETT PACKARD

HP 3000 SERIES III ARCHITECTURE

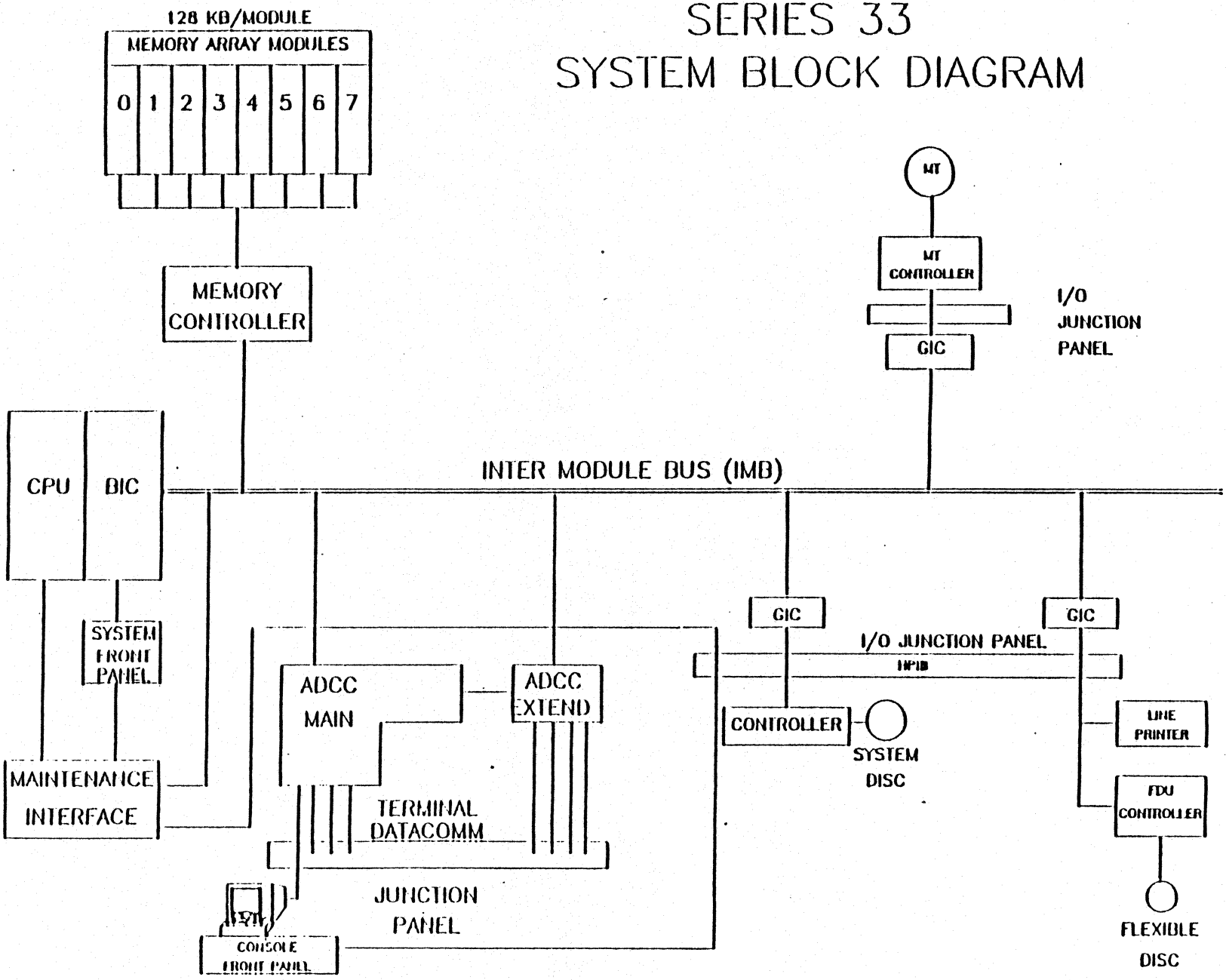


SYSTEM TERMINOLOGY: ASYNCHRONOUS TERMINAL CONTROLLER

SERIES 30
SYSTEM BLOCK DIAGRAM



SERIES 33
SYSTEM BLOCK DIAGRAM



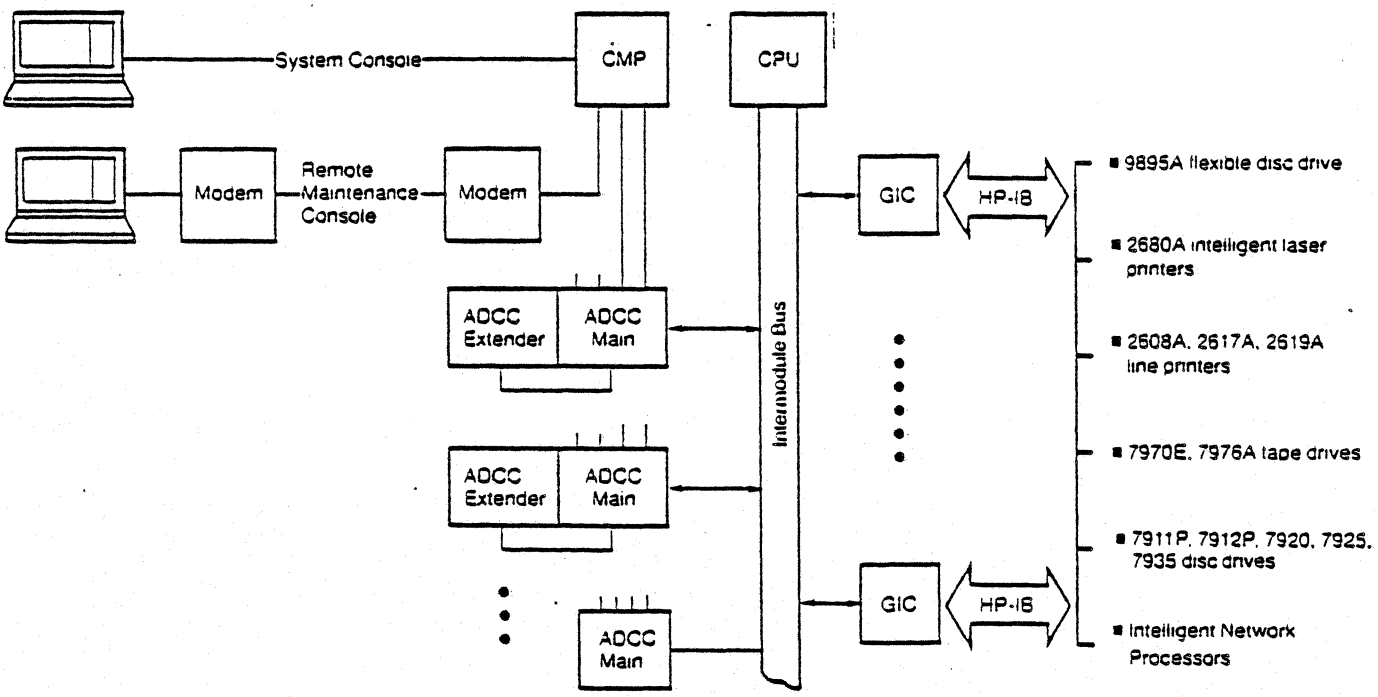
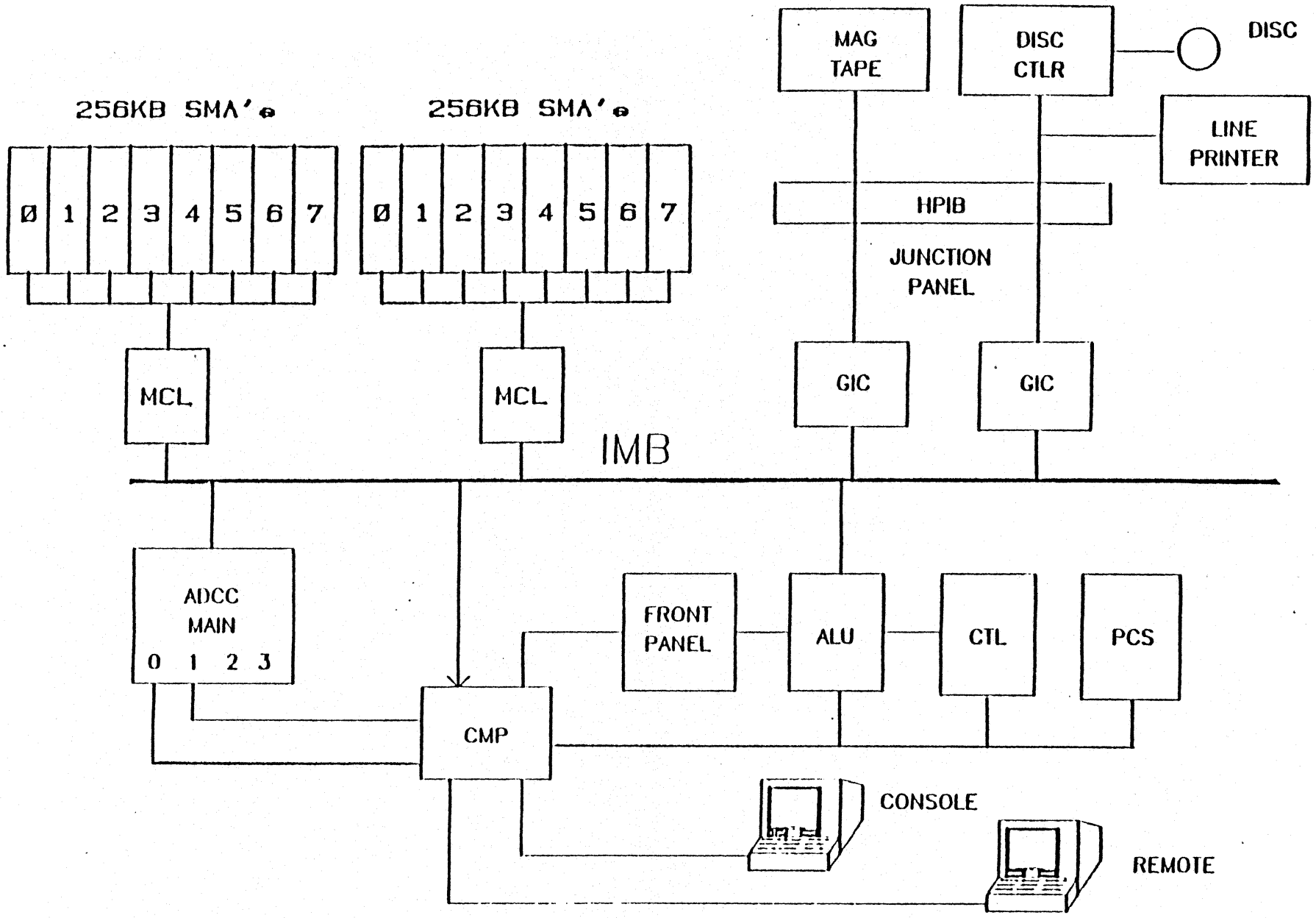
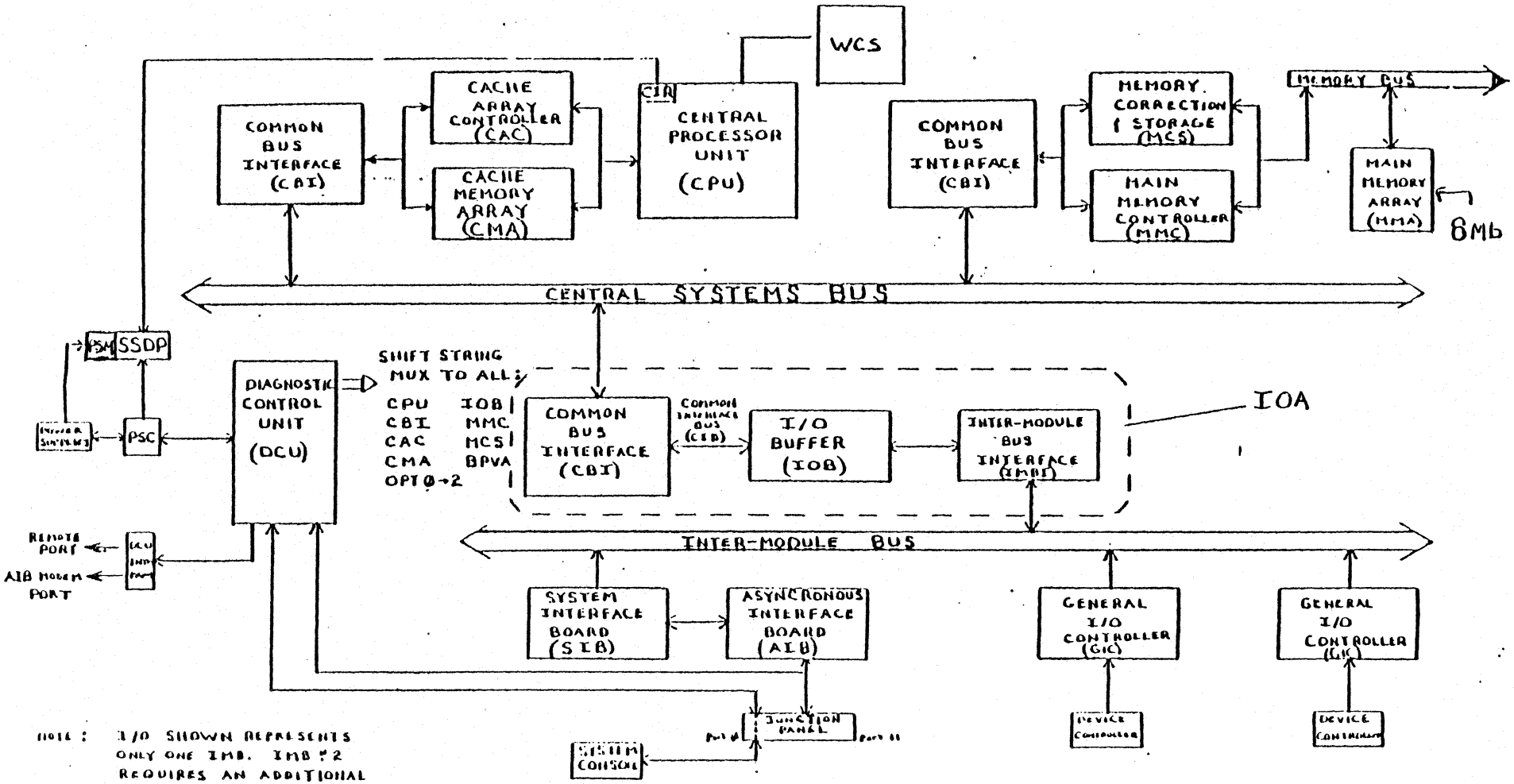


Figure G-1 HP 3000 Series 40 Hardware Organization



SERIES 44 SYSTEM BLOCK DIAGRAM

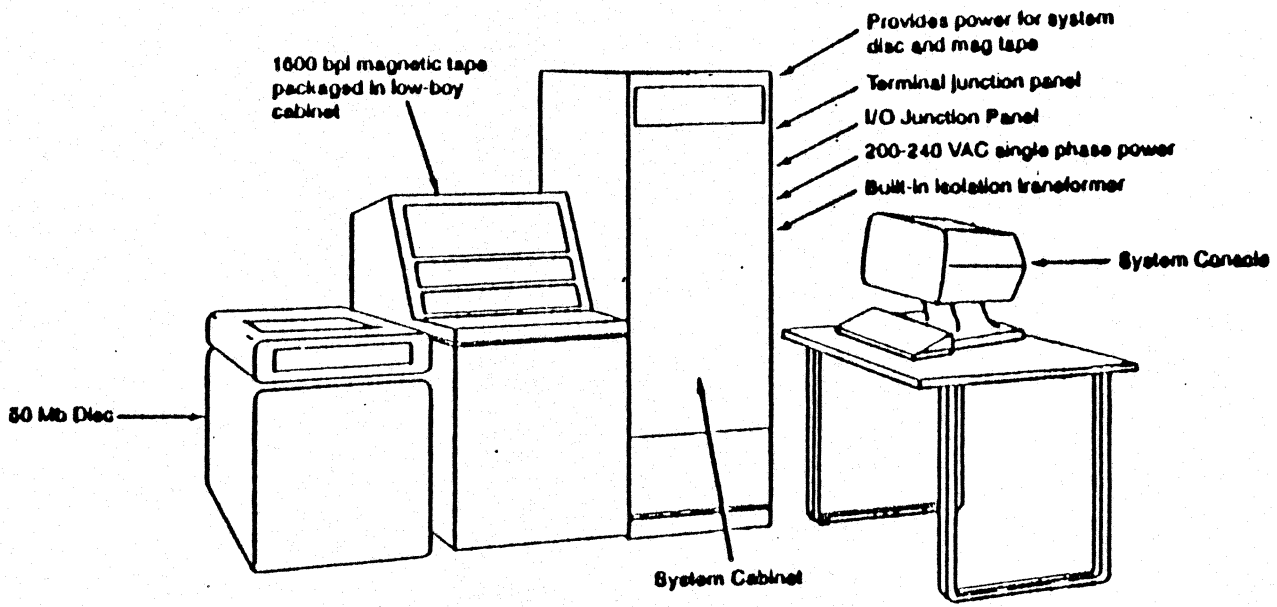
SERIES 64 SYSTEM BLOCK DIAGRAM



NOTE: I/O SHOWN REPRESENTS ONLY ONE IMB. IMB #2 REQUIRES AN ADDITIONAL IOA.

HEWLETT hp PACKARD

HP 3000 SERIES III MINIMUM CONFIGURATION



Note: Under certain power line conditions at the system site, an isolation transformer or other line conditioning equipment may be required for add-on disc and magnetic tape drives.

STANDARD (NOT ILLUSTRATED/LABELED):

256KB FAULT CONTROL MEMORY

16 TERMINAL PORTS WITH CAPACITY FOR 48 PORTS

CONSOLE TABLE

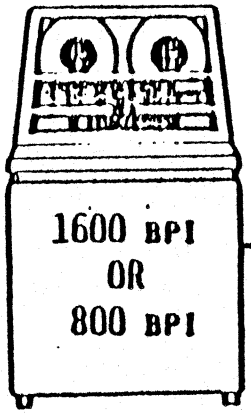
9 I/O SLOTS

MPE AND FOS

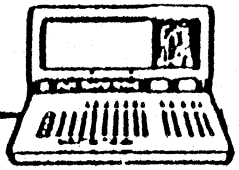
INSTALLATION AND MANUAL SET

HEWLETT PACKARD

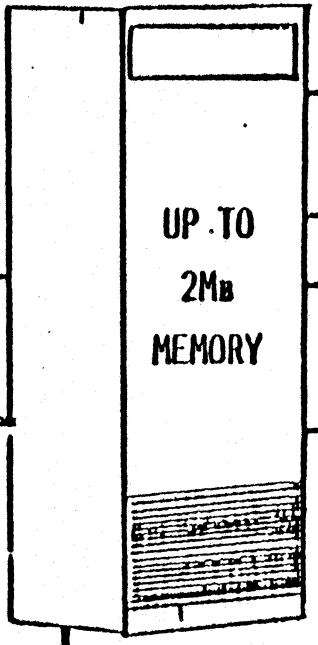
HP 3000 SERIES III
MAXIMUM CONFIGURATION



UP TO 8 MAG TAPES
(2 CONTROLLERS, EACH WITH 4 DRIVES)

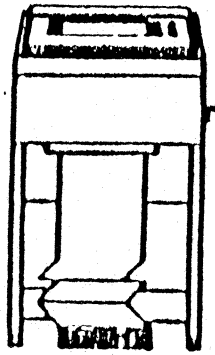


64 HARDWIRED TERMINALS
(INCLUDING CONSOLE)



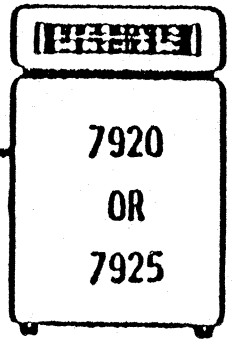
2 HSI BOARDS

7 SSLC AND /OR INP BOARDS



4 LINE PRINTERS
MIX AND MATCH: 2608, 2613, 2617, 2619

CARD READER
CARD READER/PUNCH
PAPER TAPE READER
PAPER TAPE PUNCH
CALCOMP PLOTTER

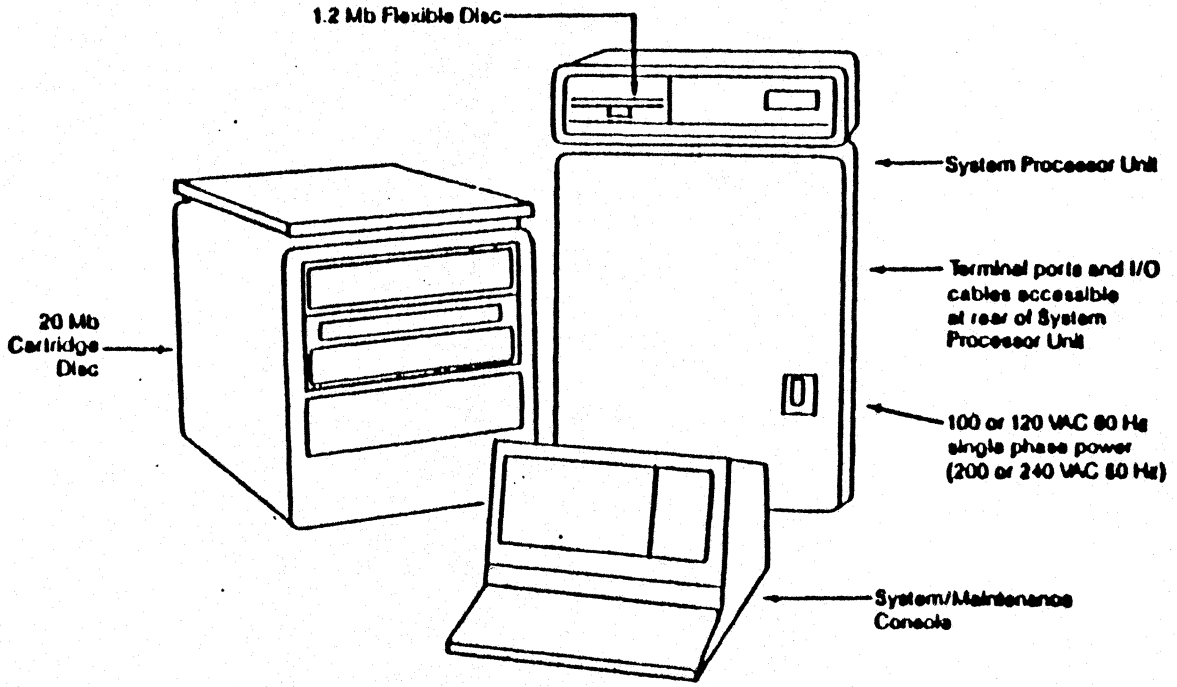


1-8 DISCS
(960MB MAX)

HEWLETT PACKARD

HP 3000 SERIES 30 MINIMUM CONFIGURATION

HP 3000 Series 30 Minimum Configuration



Note: Power outlets for disc and terminal not provided in System Processor Unit. Under certain power line conditions at the system site, an isolation transformer or other line conditioning equipment may be required. Refer to Site Preparation section of this guide.

STANDARD (NOT ILLUSTRATED/LABELED)

256KB FAULT CONTROL MEMORY

4 TERMINAL PORTS

1GENERAL I/O CHANNEL

8 I/O SLOTS

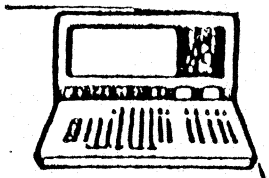
MPE AND FOS

INSTALLATION AND MANUAL SET

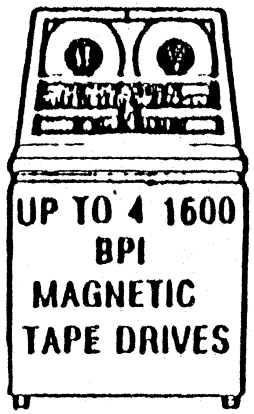
3000 SERIES 30

HEWLETT • PACKARD

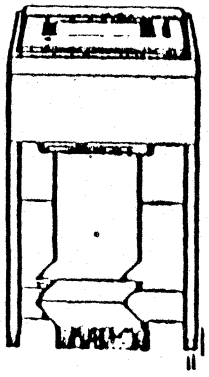
EXPANDABILITY



UP TO 32
ASYNCHRONOUS
TERMINALS*



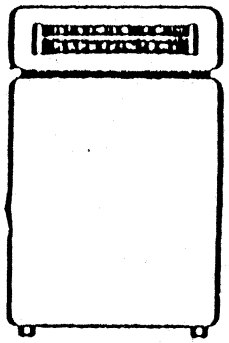
UP TO 4 1600
BPI
MAGNETIC
TAPE DRIVES



2 LINE PRINTERS
400 lpm
OR
180 cps

* NOTE TRADCOFFS

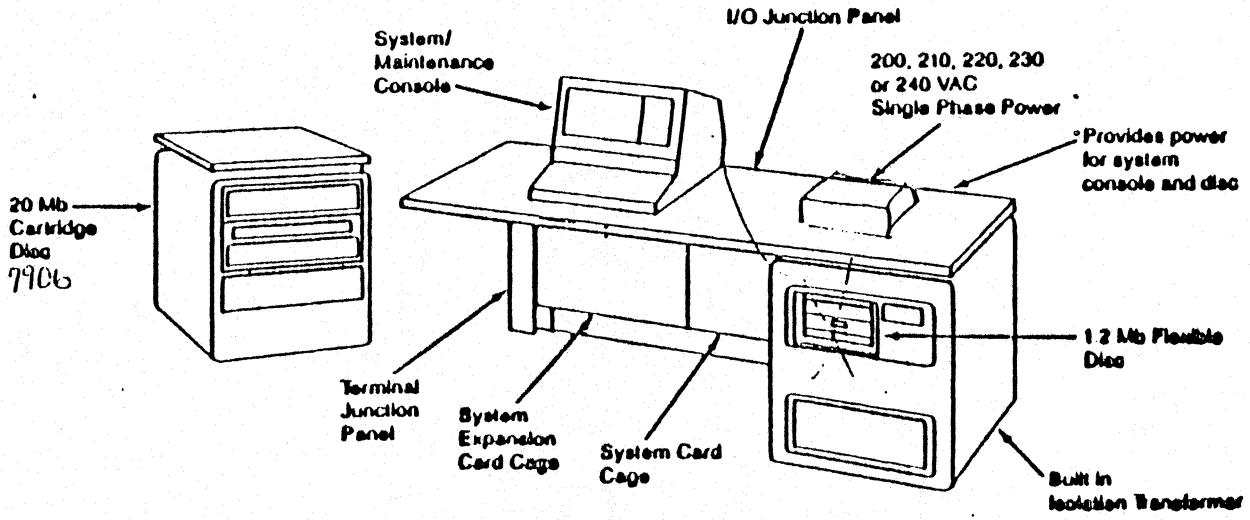
UP TO
2 INP's



UP TO 8
79XX DISC DRIVES
(960Mb maximum)

HEWLETT PACKARD

HP 3000 SERIES 33 MINIMUM CONFIGURATION



Note: Under certain power line conditions at the system site, an isolation transformer or other line conditioning equipment may be required for add-on disc and magnetic tape drives.

STANDARD (NOT ILLUSTRATED/LABELED)

256Kw FAULT CONTROL MEMORY

8 TERMINAL PORTS WITH CAPACITY FOR 32

2 GENERAL I/O CHANNELS

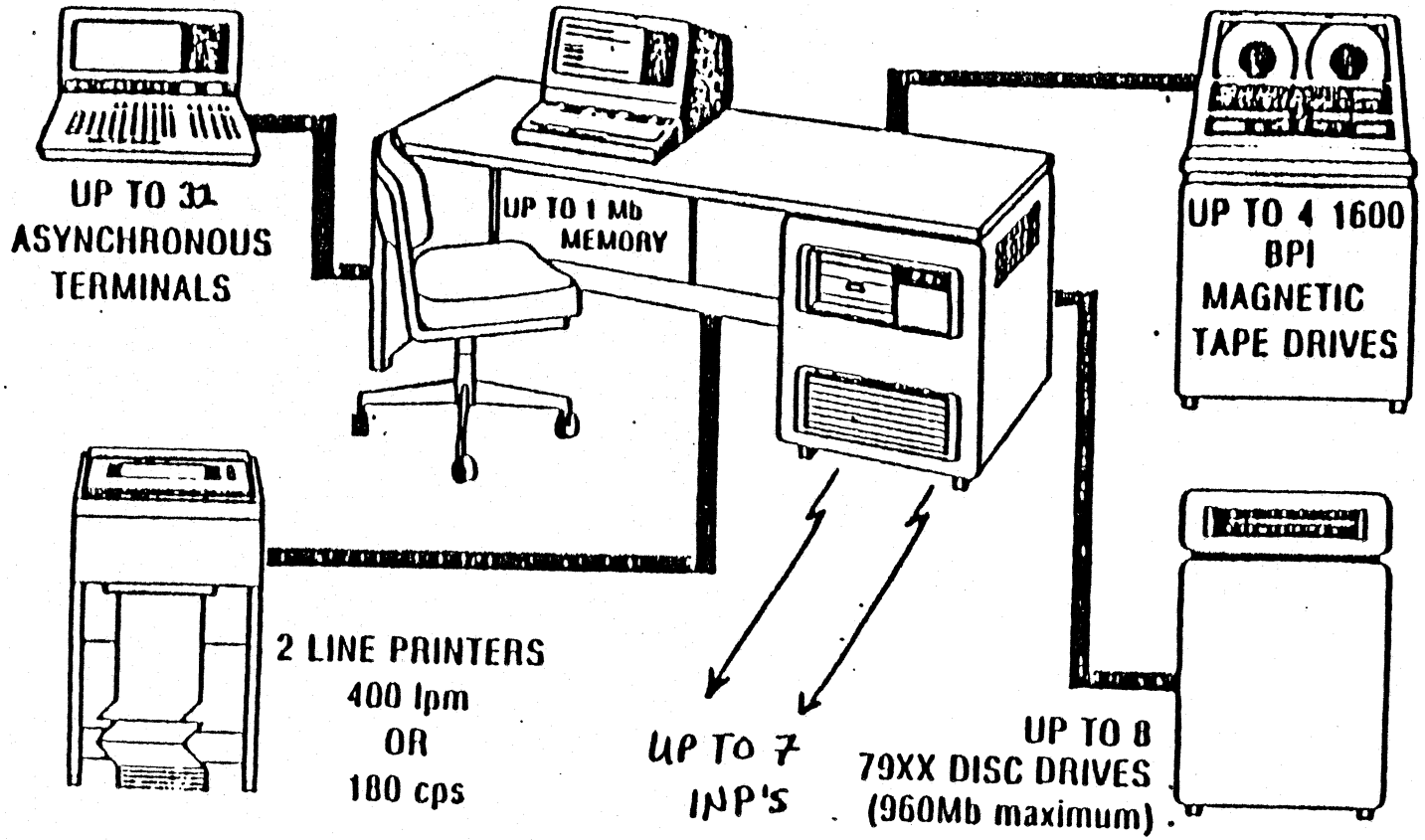
6 I/O SLOTS

SYSTEM DESK

INSTALLATION AND MANUAL SET

MPE AND FOS

HP 3000/33 EXPANDABILITY



HP 3000 Series 40 and 40SX Minimum System Configuration

Supplied Hardware:

- Central Processing Unit.
- System Clock.
- Control and Maintenance Processor (CMP).
- 2 General I/O Channels (GIC's) for System Disc and Backup Tape Drive.
- 6 Kb (Series 40SX); 512 Kb (Series 40) Fault Control Memory with Controller.
- System Mainframe Cabinet including Card Cage and Power Supplies.
- 13 I/O Card Slots and expansion capability to support up to 2 Mb of memory.

Series 40SX Only:

- One (1) 7911P or 7912P (Option 012) Integrated Storage Unit with Cartridge Tape Drive.

Required Hardware Ordered Separately:

- 1 System Console: Any HP 262X, 264x, 2382A, or 2635B terminal.
- 1 System Console Cable: if console is terminal type 262x, 264x, or 2382A.
- 1 Asynchronous Data Communications Controller (ADCC-Main) to interface system console.
- 1 Magnetic Tape Drive for system backup: 7970E or 7976A required for systems with more than 128 Mb disc storage. A 7911P or 7912P with integrated cartridge tape, which is supplied with the Series 40SX, may be used for systems having 128 Mb or less disc storage.

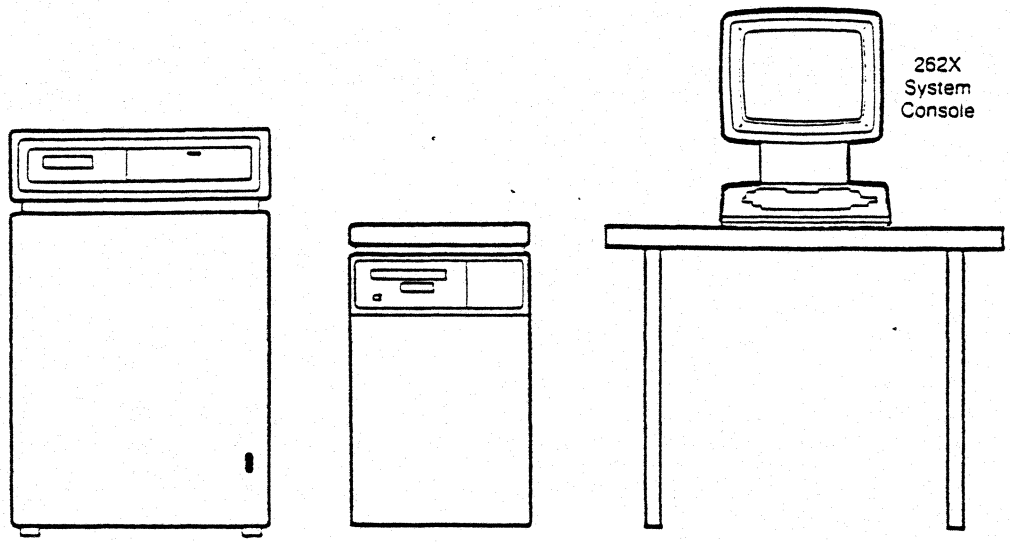
Series 40 only:

- 1 System Disc: 7920M, 7925M, or 7935H Master Disc Driver or 7911P or 7912P Integrated Storage Unit.

Supplied Software:

Standard on each HP 3000 system is the Fundamental Operating Software which includes:

- Multiprogramming Executive (MPE) Operating System.
- File Copying Utility (FCOPY/3000).
- Text Editor (EDIT/3000).
- Sort and Merge Package (SORT-MERGE/3000).
- Data Base Management System (IMAGE/3000).
- Data Base Inquiry Language (QUERY/3000).
- Data Entry and Forms Management Software (HP VPLUS/3000).
- Keyed Sequential Access Method Software (KSAM/3000).
- Complete User Manual Set.

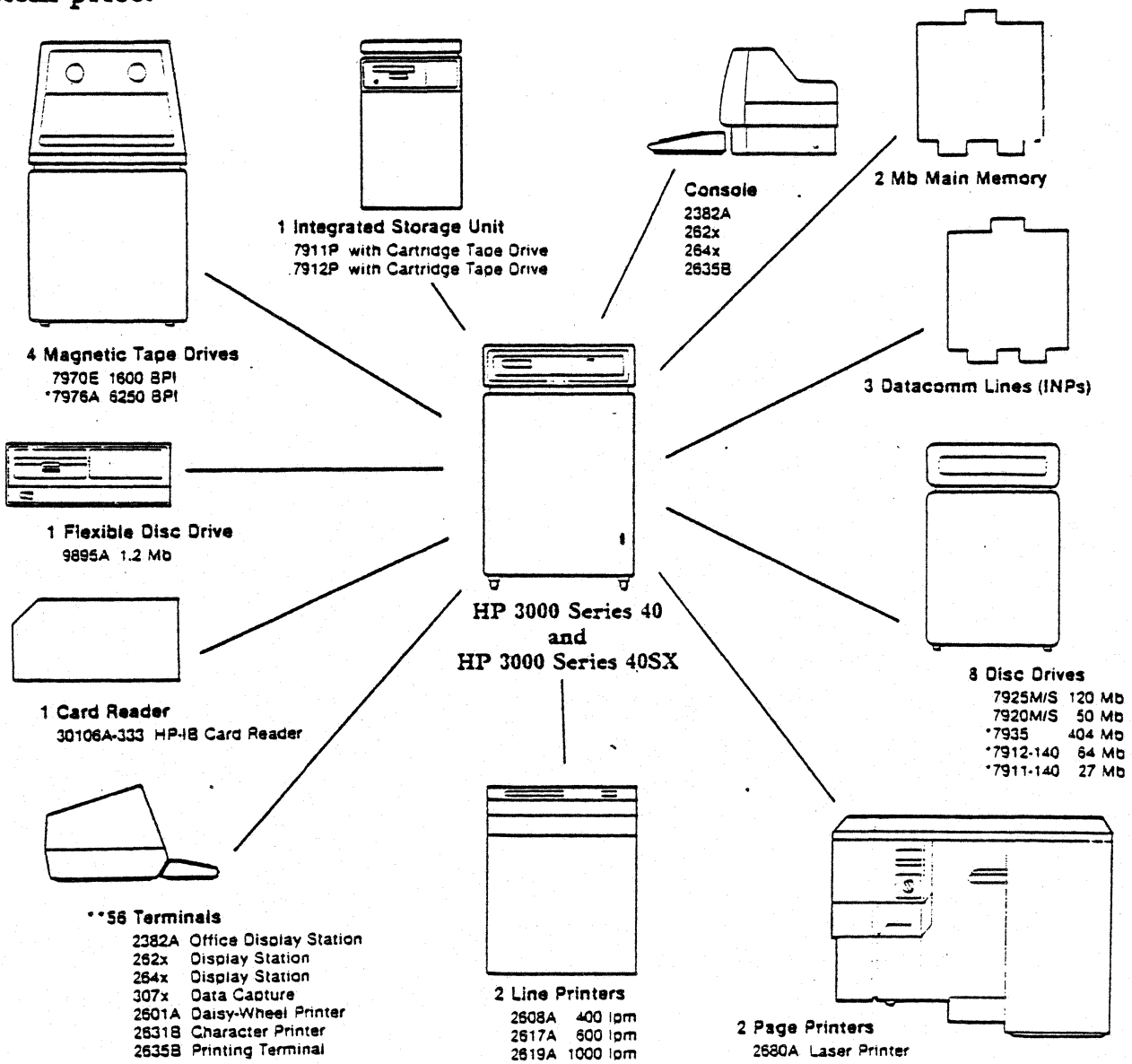


HP 3000 Series 40 Minimum System Configuration

Expandability

The HP 3000 Series 40 and Series 40SX offer the expandability of a mid-sized computer at a small system price!

It is not possible to configure a system to include all these maximums simultaneously. For additional information and guidelines please refer to the *HP 3000 Configuration Guide*.



*See the Series 40 Addendum for maximum specifications of this peripheral.

**Including point-to-point, multipoint, DSN/DS virtual terminals, and system console (maximum point-to-point terminals = 32).

HP 3000 Series 44 Minimum System Configuration

Supplied Hardware:

- Central Processing Unit.
- System Clock.
- Control and Maintenance Processor (CMP).
- 2 General I/O Channels (GICs): for System Disc and Backup Tape Drive.
- 1 Megabyte Fault Control Memory with Controller.
- System Mainframe Cabinet including Card Cages and Power Supplies.
- 26 I/O Slots and expansion capacity for 4 Megabytes of Memory.
- Built-in Isolation Transformer.

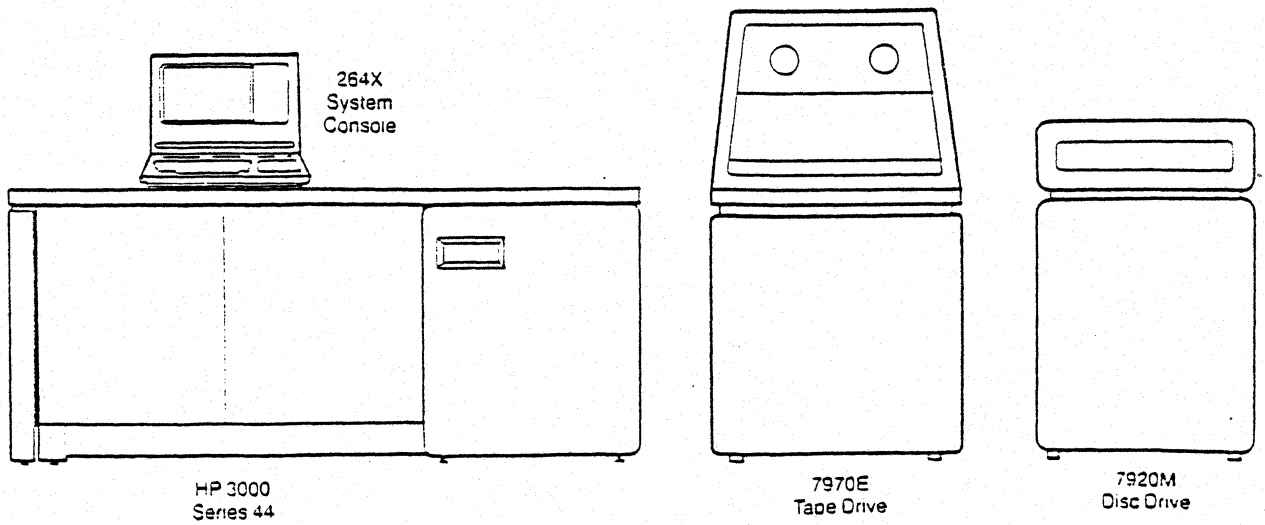
Required Hardware Ordered Separately:

- 1 System Console: Any HP 262x, 264x, 2382A, or 2635B terminal.
- 1 System Console Cable: if console is terminal type 262x, 264x, or 2382A.
- 1 System Disc: 7920M, 7925M or 7935H Master Disc Drive.
- 1 Asynchronous Data Communications Controller (ADCC-Main) to interface system console.
- 1 Magnetic Tape Drive for system backup: 7970E or 7976A.

Supplied Software:

Standard on each HP 3000 system is the Fundamental Operating Software which includes:

- Multiprogramming Executive (MPE) Operating System.
- Text Editor (EDIT/3000).
- File Copying Utility (FCOPY/3000).
- Sort and Merge Package (SORT-MERGE/3000).
- Data Base Management System (IMAGE/3000).
- Data Base Inquiry Language (QUERY/3000).
- Data Entry and Forms Management Software (HP VPLUS/3000).
- Keyed Sequential Access Method Software (KSAM/3000).
- Complete User Manual Set.



HP 3000 Series 44 Minimum System Configuration

HP 3000 Series 64 Minimum System Configuration

Supplied Hardware:

Central Processing Unit.

System Clock.

Diagnostic Control Unit (DCU).

2 General I/O Channels (GIC's): for System Disc and Backup Tape Drive.

2 Megabytes Fault Control Memory with Controller.

8 Kb Cache Memory.

System Mainframe Cabinet including Card Cage and Power Supplies.

24 I/O Slots and expansion capacity for 8 Mbytes of Memory.

40 Kb of Writeable Control Store (WCS).

Built-in Isolation Transformers.

Required Hardware Ordered Separately:

1 System Console: HP 2642A Terminal with Option 964 (includes cable).

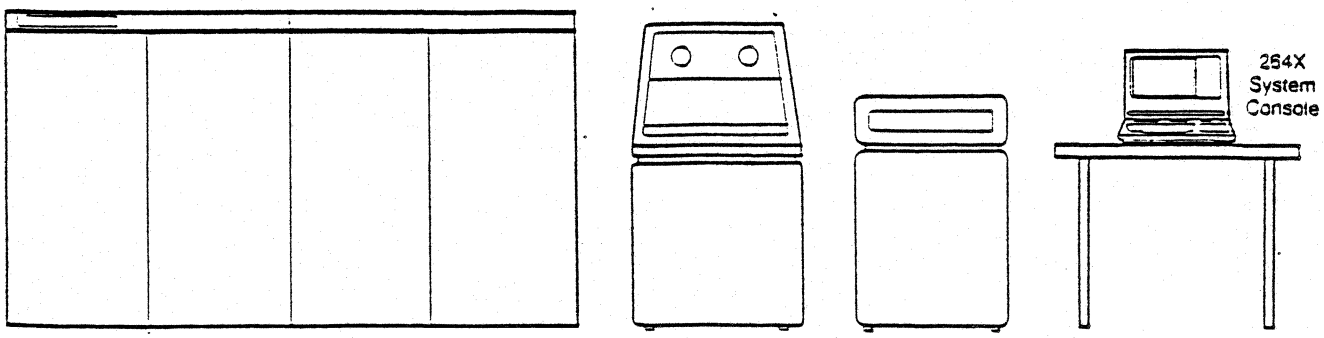
1 System Disc: 7920M, 7925M, or 7935H Master Disc Drive.

- 1 DSN/Advanced Terminal Processor (1 System Interface Board *AND* 1 Port Controller) to interface the system console.
- 1 Magnetic Tape Drive for system backup: 7970E or 7976A.

Supplied Software:

Standard on each HP 3000 system is the Fundamental Operating Software which includes:

- Multiprogramming Executive (MPE) Operating System.
- Text Editor (EDIT/3000).
- File Copying Utility (FCOPY/3000).
- Sort and Merge Package (SORT-MERGE/3000).
- Data Base Management System (IMAGE/3000).
- Data Base Inquiry Language (QUERY/3000).
- Data Entry and Forms Management Software (HP VPLUS/3000).
- Keyed Sequential Access Method Software (KSAM/3000).
- Complete User Manual Set.

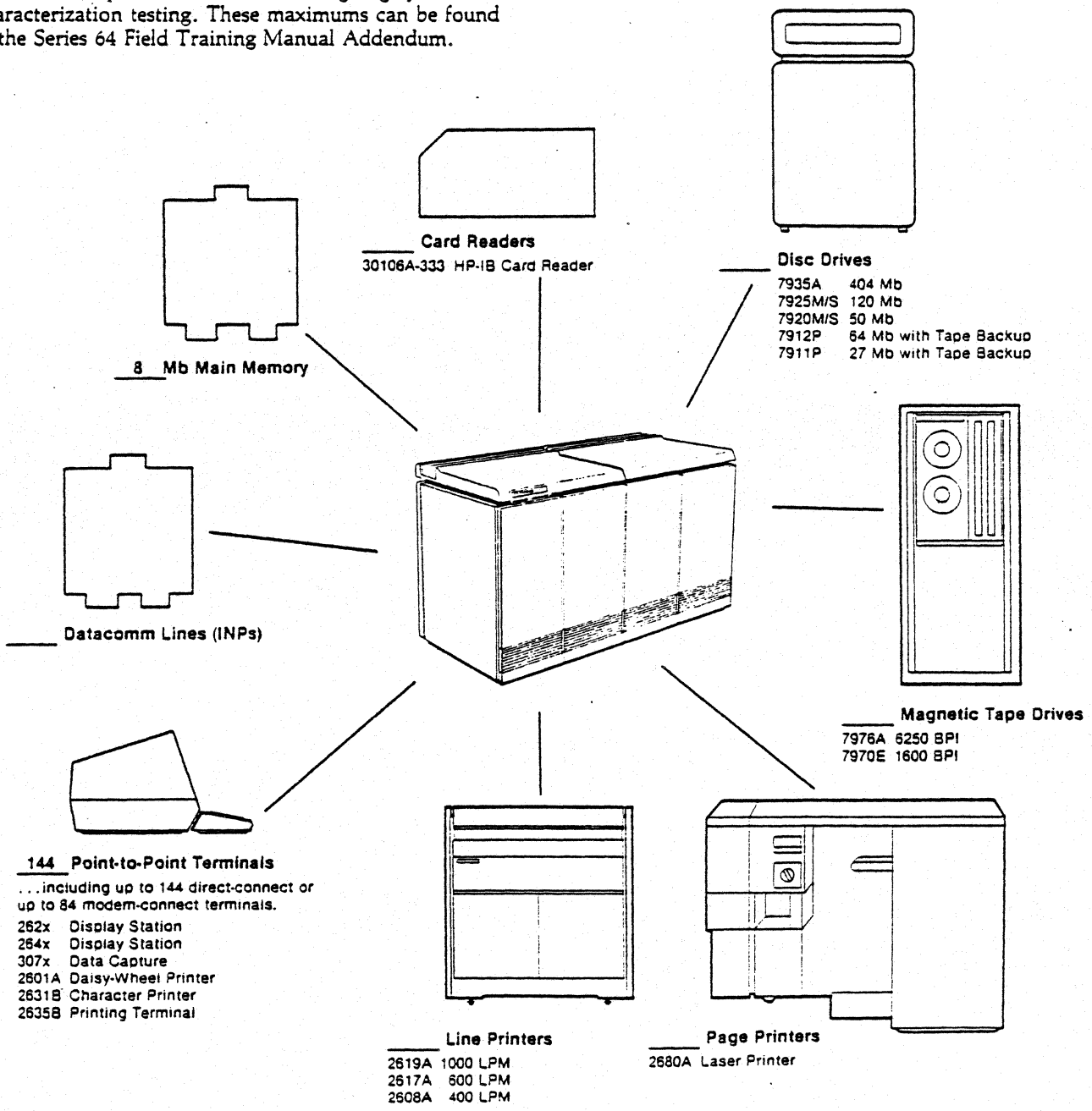


HP 3000 Series 64 Minimum System Configuration

Expandability

The Series 64 offers the best overall expandability of any HP 3000. Shown below are the individual maximums for each system feature.

It was not possible to include the individual device maximums at press time because of ongoing system characterization testing. These maximums can be found in the Series 64 Field Training Manual Addendum.



____ Card Readers
30106A-333 HP-IB Card Reader

____ Disc Drives
7935A 404 Mb
7925M/S 120 Mb
7920M/S 50 Mb
7912P 64 Mb with Tape Backup
7911P 27 Mb with Tape Backup

8 Mb Main Memory

____ Datacomm Lines (INPs)

____ Magnetic Tape Drives
7976A 6250 BPI
7970E 1600 BPI

144 Point-to-Point Terminals
. . . including up to 144 direct-connect or up to 84 modem-connect terminals.
262x Display Station
264x Display Station
307x Data Capture
2601A Daisy-Wheel Printer
2631B Character Printer
2635B Printing Terminal

____ Line Printers
2619A 1000 LPM
2617A 600 LPM
2608A 400 LPM

____ Page Printers
2680A Laser Printer

SUPPORTED CONFIGURATIONS

DEVICE	ELECTRICAL LOADS PER DEVICE	S/30	S/33	S/40	S/44	S/64	STARFISH	GIC
MAX HIGH SPEED GICS***		1	1	2	2	4*****	1	6 DEVICES EACH MAX
MAX GICS		3	4	4	5	10	1	
# 7911/12 DISC****	1?	0	0	4	1	1	0	HIGH SPEED ONLY
# 7935H	1	1	1	8	8	16	1	HIGH SPEED ONLY
7920/25H	1	1	1	2	2	16	N/A	HIGH SPEED ONLY
7920/25S	-	7	7	7	14	14	N/A	-
MAX DISC	-	8	8	8	16	16	1	-
LINUS	1?	0	0	1	1	1	0	DEDICATED
7970E-H	1	1	1	1	2	6	N/A	DEDICATED
7970E-S	-	3	3	3	6	6	N/A	-
7976A	1-4	1	1	1	1	2	1	HIGH SPEED ONLY
MAX TAPES	-	4###	4###	4	8	8	1	-
2617A/19A	1	1	1	2	4	4	N/A	ANY
2608A	1	1	1	2	2	4	N/A	NOT ON HI SPEED GIC
MAX LPS	-	2	2	2	4	8	N/A	-
9895A	1	1	1	1	1	1	0	ANY
2680A	3	1####	1####	2	2	2	2	HIGH SPEED ONLY
INP	1	2	7	3	7	16**	N/A	ANY
30196A	1	1	1	1	1	1	N/A	DEDICATED
2631B				4	4	16		
DIRECT CONNECT TERMINALS				32*	60	144		(incl. 2631Bs)
MODEM DIRECT CONNECT TERMS				31*	59	84		
MULTIPOINT TERMINALS				95##	95	143		
DS PSEUDO TERMINALS				95##	95	143		
MAX TERMINALS				96##	96	144		

####	dedicated slow speed GIG
###	1 master max
?	currently specified as 1, but may increase to 2 due to EMC changes
#	not tested
##	not tested on 256Kb memory systems
*	less on 256Kb memory systems
**	10 @ 7000 CPS; 16 @ 2400 CPS
***	a high speed GIC is a GIC with high speed devices on it - no more than six devices (not loads) may be on a high speed GIC
****	if ordered with Linus, see Linus specification
*****	max of two per IMB
N/A	peripherials available via selector channel or MUX, and not supported via HPIB.

HP 3000 Family
Hardware Capabilities

The chart below shows how the Series 64 hardware features compare to those of the other HP 3000 family members.

Area/Feature	S30	SIII	S40	S44	S64
General					
Performance Index	0.7-1	1.5-2	2.5-3	2.5-3.5	7-9
Technology	Silicon On Saphire	TTL	Schottky TTL	Schottky TTL	Emitter Coupled Logic
CPU and Diagnostic Boards	4	9	4	4	14
Number of ALUs	1	1	1	1	2
Microcode Cycle Time	450 ns	175 ns	105 ns	105 ns	75 ns
Writeable Control Storage	—	—	—	—	40 Kb
32-bit Operations	Microcode loop	Microcode loop	Microcode loop	Microcode loop	Single Operation
Cache Memory Size	—	—	—	—	8 Kb
Memory/Central Bus					
MIN/MAX Memory	256Kb / 1Mb	256Kb / 2Mb	256Kb / 2Mb	1Mb / 4Mb	2Mb / 8Mb
Effective Memory Cycle Time	640 ns	700 ns	465 ns	465 ns	134 ns
Memory Bandwidth	3 Mb/Sec	3 Mb/Sec	4.3 Mb/Sec	4.3 Mb/Sec	19 Mb/Sec
Address Length	20 Bits	20 Bits	24 Bits	24 Bits	32 Bits
Central Bus Bandwidth	3.8 Mb/Sec	2.8 Mb/Sec	3.8 Mb/Sec	3.8 Mb/Sec	56 Mb/Sec
I/O					
I/O	IMB	Sel. Chan.	IMB	IMB	1-2 IMB
Single Bus Bandwidth	3.0 Mb/Sec	2.3 Mb/Sec	3.0 Mb/Sec	3.0 Mb/Sec	3.0 Mb/Sec
Total I/O Bandwidth	3.0 Mb/Sec	2.3 Mb/Sec	3.0 Mb/Sec	3.0 Mb/Sec	2x3.0 Mb/Sec
Peripheral Protocol	HP-IB	Parallel Differential	HP-IB	HP-IB	HP-IB
Max Direct Connect Terminals	32	64	32	60	144
Total Terminals Supported (Direct connect plus multipoint)	48	96	56	96	144

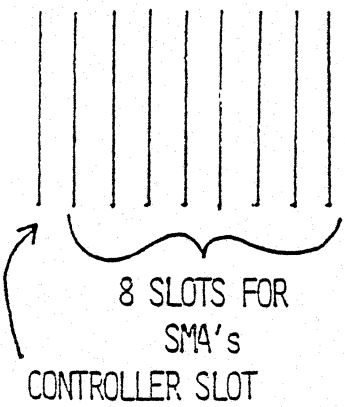
PRONTO MEMORY SUBSYSTEM FOR SERIES 40/44

PRODUCT #	DESCRIPTION	PRICE	NOTES	
30094A	MEMORY CONTROLLER	??		
30171A	256KB MEMORY MODULE	??	CONSISTS OF ONE 1/4 MB SMA	16K RAMS
30092A	512KB MEMORY MODULE	??	CONSISTS OF TWO 1/4 MB SMA'S	
30161A	1MB MEMORY MODULE		CONSISTS OF ONE 1MB SMA	64K RAMS

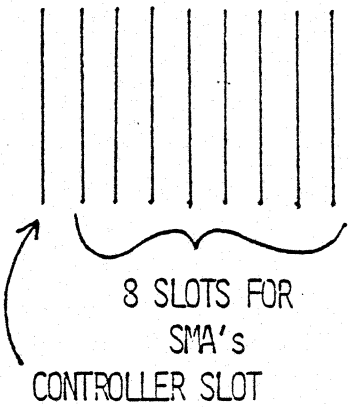
CARD CAGE SPACE AVAILABLE:

SERIES 44

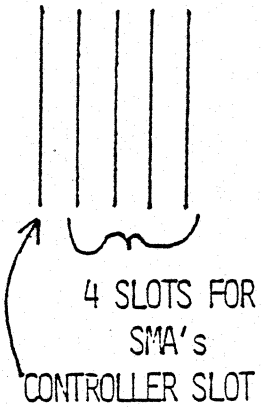
1ST CARD CAGE



2ND CARD CAGE



SERIES 40



CONFIGURATION LIMITATIONS

SERIES 44

- 4MB MAXIMUM MEMORY
- 1MB AND 1/4MB SMA'S MAY BE COMBINED IN ANY MANNER THAT RESULTS IN ONE OF THE FOLLOWING TOTAL MEMORY SIZES:
  1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0 (MB)
- IF ANY 1/4MB ARRAYS ARE USED IN THE SYSTEM, ONLY 2.0 MB'S OF MEMORY CAN BE INSTALLED PER CONTROLLER.
- IF ONLY 1MB SMA'S ARE USED, UP TO 4MB OF MEMORY CAN BE INSTALLED ON A SINGLE CONTROLLER. SECOND CONTROLLER IS NOT TO BE USED.

SERIES 40

- 2MB MAXIMUM MEMORY
- 1MB AND 1/4MB SMA'S MAY BE COMBINED TO OBTAIN SUPPORTED MEMORY SIZES OF *
  256KB, 512KB, 768KB, 1MB, 1.5MB, 2.00MB

* ONLY 4 MEMORY SLOTS ARE AVAILABLE - CAREFUL!

3000 SYSTEMS

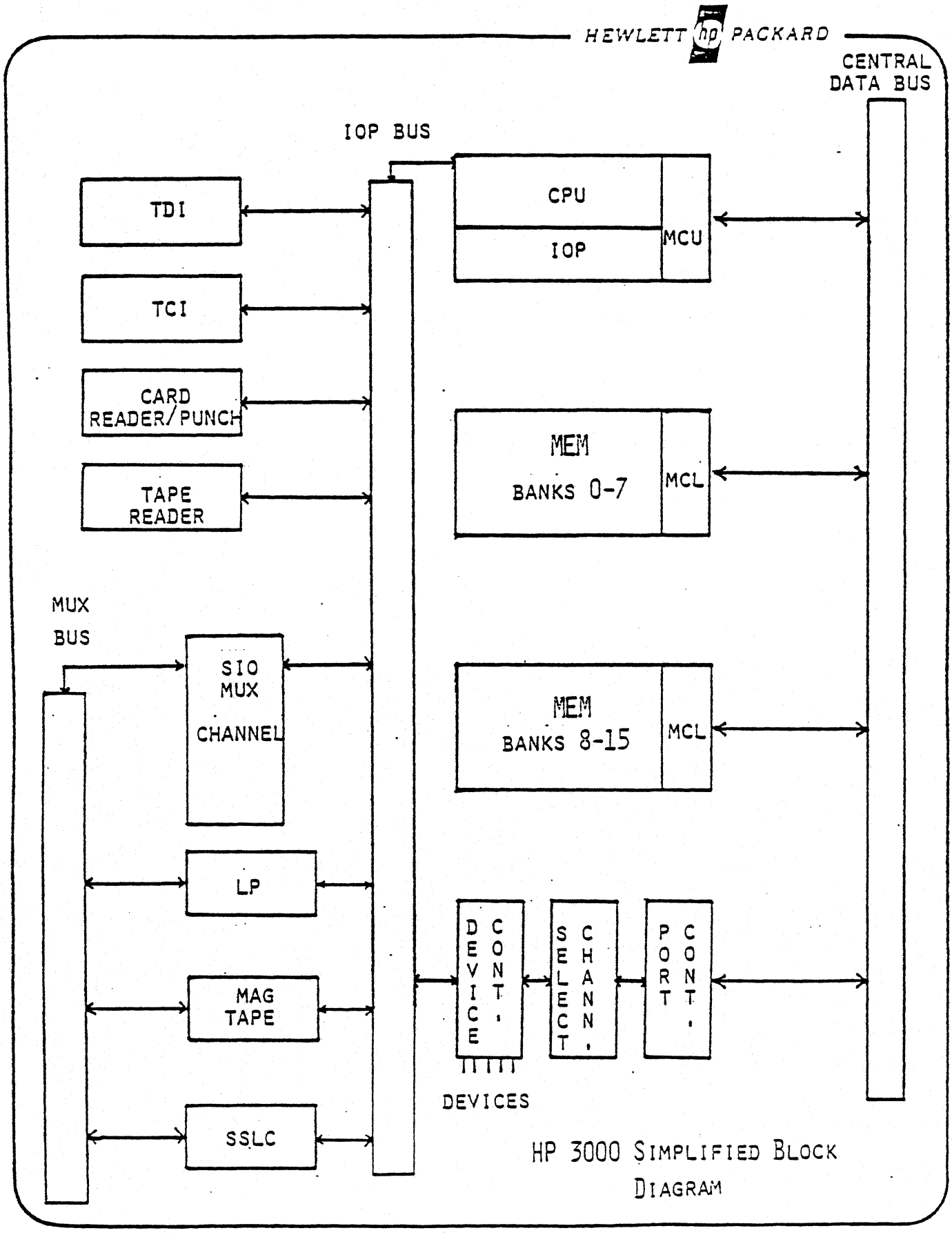
MEMORY COMPARISONS

		WORDS			SMA'S	BANKS
SYSTEM MODEL	RAM TYPE	PCA	MCL	TOTAL	MCL/ TOTAL	PCA/MCL/ TOTAL
III	16K	128K	512K	1M	4/8	2 / 8 /16
30/33	16K	64K	512K	512K	8/8	1 / 8 / 8
40	16K	128K	1M	1M	4/4	2 / 8 / 8
	64K	512K	1M	1M	2/2	4 / 8 / 8
44	16K	128K	1M	2M	8/16	2 /16 /32
	64K	512K	2M	2M	4/4	4 /32 /32

HARDWARE REFERENCE SECTION

SERIES II/III HARDWARE

- MEMORY
- CPU
- IOP
- SELECTOR CHANNEL
- MULTIPLEXER CHANNEL
- DEVICE CONTROLLER



HP 3000 SIMPLIFIED BLOCK DIAGRAM

HP 3000 SERIES III
ARCHITECTURE

SELECTOR CHANNEL

1 DISC CONTROLLER

↓

1-8 {7920 / 7925} DISC DRIVES

MULTIPLEXER CHANNEL

LINE PRINTER CONTROLLER ⟶ 1 LP

MAG TAPE CONTROLLER ⟶ 1 TO 4 TAPES

DATA COMMUNICATIONS:

- HSI (HARDWIRED SERIAL INTERFACE)
- SSLC (SYNCHRONOUS SINGLE LINE CONTROLLER)
- INP (INTELLIGENT NETWORK PROCESSOR)

CARD, PAPER TAPE DEVICES

ASYNCHRONOUS TERMINAL CONTROLLER(S)

CRT TERMINALS ⟵ PLOTTERS, PRINTERS ATTACHED

PRINTING TERMINALS

MEMORY

- MEMORY PCA'S (PRINTED CIRCUIT ASSEMBLY) CONSIST OF:
  - SEMICONDUCTOR MEMORY ARRAY (SMA)

    SERIES II: 4K RAM, 64 KB/BOARD, 256 KB-512 KB, 4 BANKS

    SERIES III: 16K RAM, 256 KB/BOARD, 256 KB-2 MB, 16 BANKS.
  - MEMORY CONTROL AND LOGGING (MCL) } → 1 EACH PER 4 SMA'S
  - FAULT CORRECTION ARRAY (FCA) }
  - FAULT LOGGING INTERFACE (FLI) } → 1 PER SYSTEM

Bank No.	System Word Capacity	PCA's Required				
		MCL	SMA	FCA	FLI	Total
0	64K	1	2	1	1	5
1	96K	1	3	1	1	6
1	128K	1	4	1	1	7
2	160K	2	5	2	1	10
2	192K	2	6	2	1	11
3	224K	2	7	2	1	12
3	256K	2	8	2	1	13

- SPECS: WRITE - 700 NSEC MINIMUM CYCLE TIME

  READ - 350 NSEC ACCESS, 700 NSEC CYCLE TIME

  READ/WRITE 1's (RW1) - 350 NSEC ACCESS, 1050 CYCLE TIME

  NO OPERATION (NOP) - 700 NSEC CYCLE TIME

MEMORY (CONT.)

- SERIES II/III MEMORY WORD LENGTH IS ACTUALLY 21/22 BITS (16 DATA + 5/6 CHECK BITS FOR FAULT DETECTION AND SINGLE-FAULT CORRECTION).
- ACCESS TO MEMORY IS VIA THE CENTRAL DATA BUS (CTL BUS) AND CONTROLLED BY MODULE CONTROL UNIT (MCU).

M C U

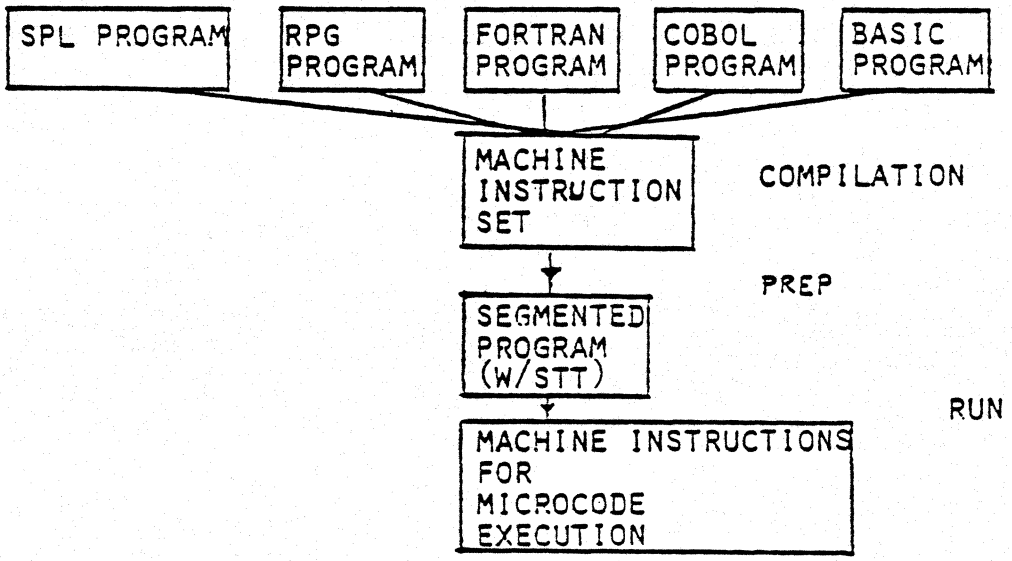
MODULE CONTROL UNIT

CONTROLS ACCESS TO CTL BUS
ONE PER MODULE

- MEMORY - MODULES 0 - 3
- SELECTOR CHANNEL - MODULE 4
- CPU - MODULE 5

CPU

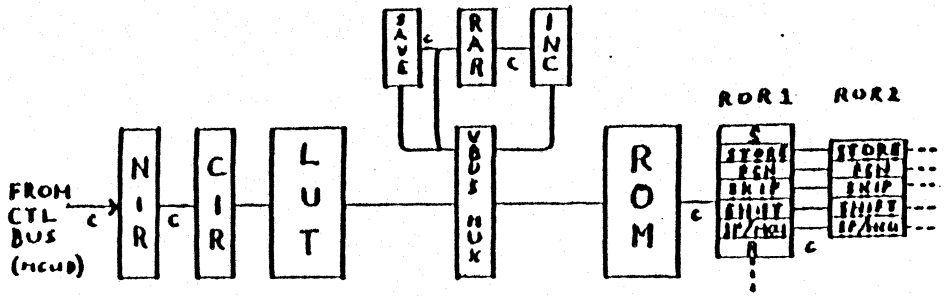
- MICROCODED CPU
- 175 NSEC CYCLE TIME
- HARDWARE - IMPLEMENTED STACK ARCHITECTURE
- DECODES MACHINE INSTRUCTION SET INTO MICROCODE INSTRUCTIONS TO EXECUTE



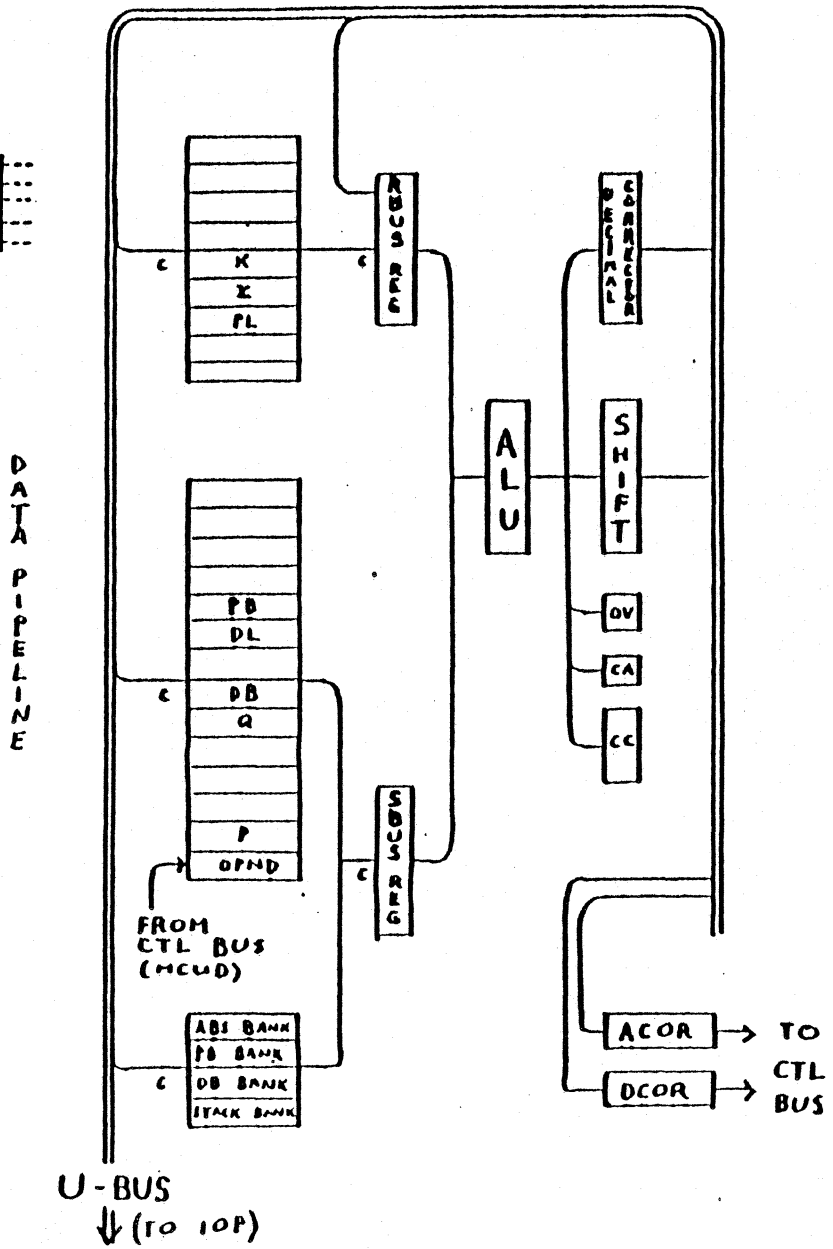
(SEE SYSTEM REF. MANUAL, PG.3-36 FOR MICROCODE INSTRUCTIONS)

-CPU LOGIC CAN BE DIVIDED INTO:

- MICROCODE PIPELINE
- DATA PIPELINE



MICROCODE PIPELINE



SIMPLIFIED CPU DIAGRAM
SERIES II/III

MICROCODE PIPELINE

1. NIR IS FETCHED FROM MEMORY CONCURRENT TO CURRENT INSTRUCTION EXECUTION.

2. CIR IS LOADED FROM NIR. (1 CLOCK CYCLE)

3. LUT RETURNS THE MICROPROGRAM STARTING ADDRESS FOR THE CIR INSTRUCTION.

4. THE V-BUS MUX OUTPUTS A 16-BIT ADDRESS TO THE ROM AND INC.

5. THE ROM OUTPUT (32-BIT INSTRUCTION) IS CLOCKED INTO ROR-1. AT THE SAME TIME, THE ADDRESS IS INCREMENTED AND SENT TO RAR. (1 CLOCK CYCLE)

6. THE OUTPUT OF THE RAR IS SENT BACK TO THE V-BUS MUX AND, ON THE NEXT CLOCK, THE INCREMENTED ADDRESS IS SENT BACK TO THE ROM.

7. IF THE MICROPROGRAM JUMPS TO A SUBROUTINE, RAR IS STORED IN SAVE.

ROR1/ROR2 FIELDS:

S - (0:4) SELECTS ONE OF 31 REGISTERS TO BE LOADED INTO THE S-BUS.

STORE - (5:9) SELECTS ONE OF 29 REGISTERS TO STORE THE U-BUS DATA.

FCN - (10:14) SPECIFIES THE FUNCTION TO BE PERFORMED BY THE ALU.

SKIP - (15:19) DETERMINES WHAT CONDITION SHALL BE TESTED FOR A POSSIBLE SKIP. ALSO CONTROLS NEXT CIR.

SHIFT - (20:22) SPECIFIES HOW THE DATA WILL BE SHIFTED.

SP/MCU - (23:27) CONTROLS SPECIAL FUNCTIONS AND MEMORY/CTL REQUESTS.

R - (28:31) SELECTS ONE OF 15 REGISTERS TO BE LOADED INTO THE R-BUS.

DATA PIPELINE

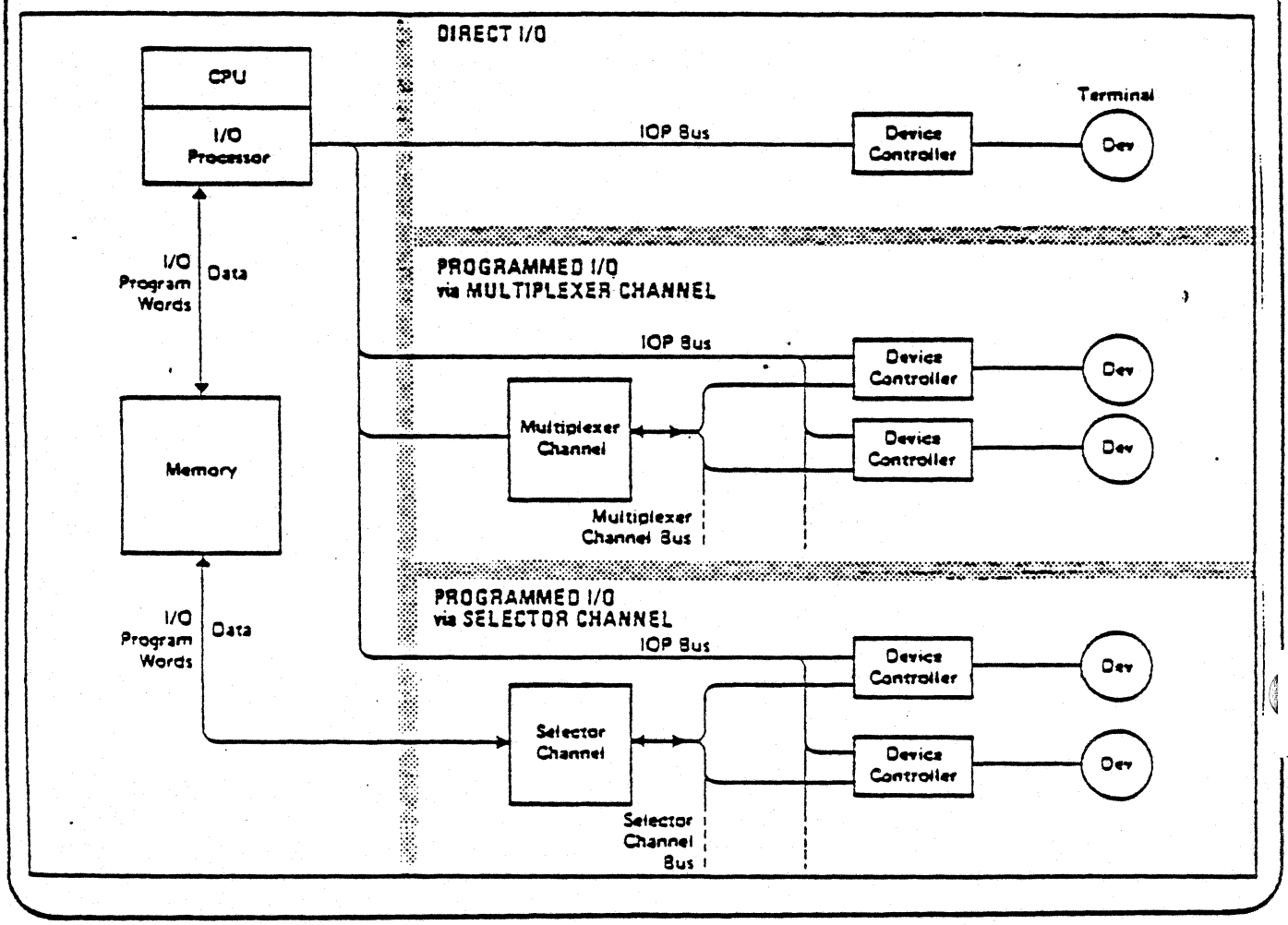
1. ON THE FIRST CLOCK CYCLE, THE ROM INSTRUCTION (32-BIT) IS LOADED INTO ROR1..
2. ROR1 DECODES THE S AND R FIELDS TO LOAD THE S-BUS REGISTERS IN ADVANCE OF THE OPERATION.
3. ON THE SECOND CLOCK CYCLE, THE REST OF ROR1 IS LOADED INTO ROR2, THE R-BUS AND S-BUS REGISTERS ARE LOADED (AS PER THE R AND S FIELDS) AND ROR1 RECEIVES THE NEXT MICRO-INSTRUCTION.
4. ROR2 DECODES THE REST OF THE FIELDS TO DETERMINE WHAT OPERATIONS ARE TO BE PERFORMED AND WHERE THE RESULT IS TO BE STORED (IF ANYWHERE).
5. ON THE THIRD CLOCK CYCLE, THE RESULT IS STORED FROM THE U-BUS, AS THE NEXT ROR2 IS LOADED.

DATA PIPELINE (CONT.)

6. IF A NEW NIR OR MEMORY OPERAND IS REQUIRED, THE MEMORY ADDRESS IS PLACED IN ACOR AND THE APPROPRIATE SP/MCU INSTRUCTION IS ISSUED. THE VALUE IS RETURNED (VIA CTL BUS) TO THE OPND REGISTER OR NIR.
7. IF A WRITE TO MEMORY IS REQUIRED, THE ADDRESS IS PLACED IN DCOR AND THE APPROPRIATE SP/MCU INSTRUCTION IS ISSUED.

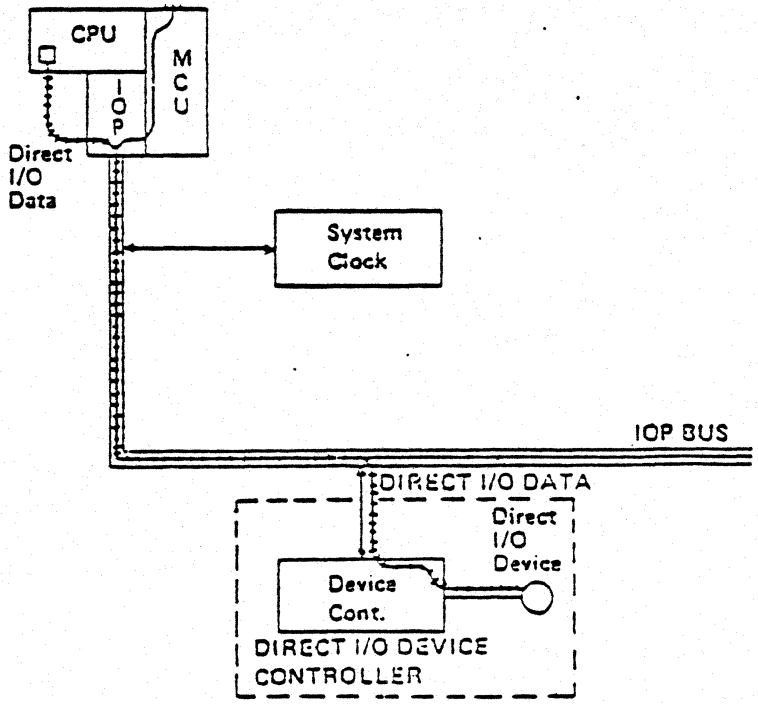
I O P
INPUT / OUTPUT PROCESSOR

- o CONTROLS ALL I/O
  - DIRECT I/O
  - PROGRAMMED I/O (SIO)
- o CONCURRENT CPU / I/O OPERATIONS
- o DEDICATED BUS


DIRECT I/O
CPU
I/O Processor
IOP Bus
Device Controller
Terminal
Dev
I/O Program Words
Data
PROGRAMMED I/O via MULTIPLEXER CHANNEL
IOP Bus
Device Controller
Dev
Multiplexer Channel
Device Controller
Dev
Memory
Multiplexer Channel Bus
PROGRAMMED I/O via SELECTOR CHANNEL
I/O Program Words
Data
IOP Bus
Device Controller
Dev
Selector Channel
Device Controller
Dev
Selector Channel Bus

DIRECT I/O (DIO)

- DIRECT I/O USED FOR:
  - TERMINALS
  - SYSTEM CLOCK
  - CARD READER / PUNCH
  - PAPER TAPE READER
- MPE USES 4 CPU INSTRUCTIONS TO TELL THE IOP TO PERFORM DIRECT I/O:
  - RIO (READ I/O)
  - WIO (WRITE I/O)
  - TIO (TEST I/O)
  - CIO (CONTROL I/O)
- TRANSFERS ONLY ONE WORD PER INSTRUCTION.



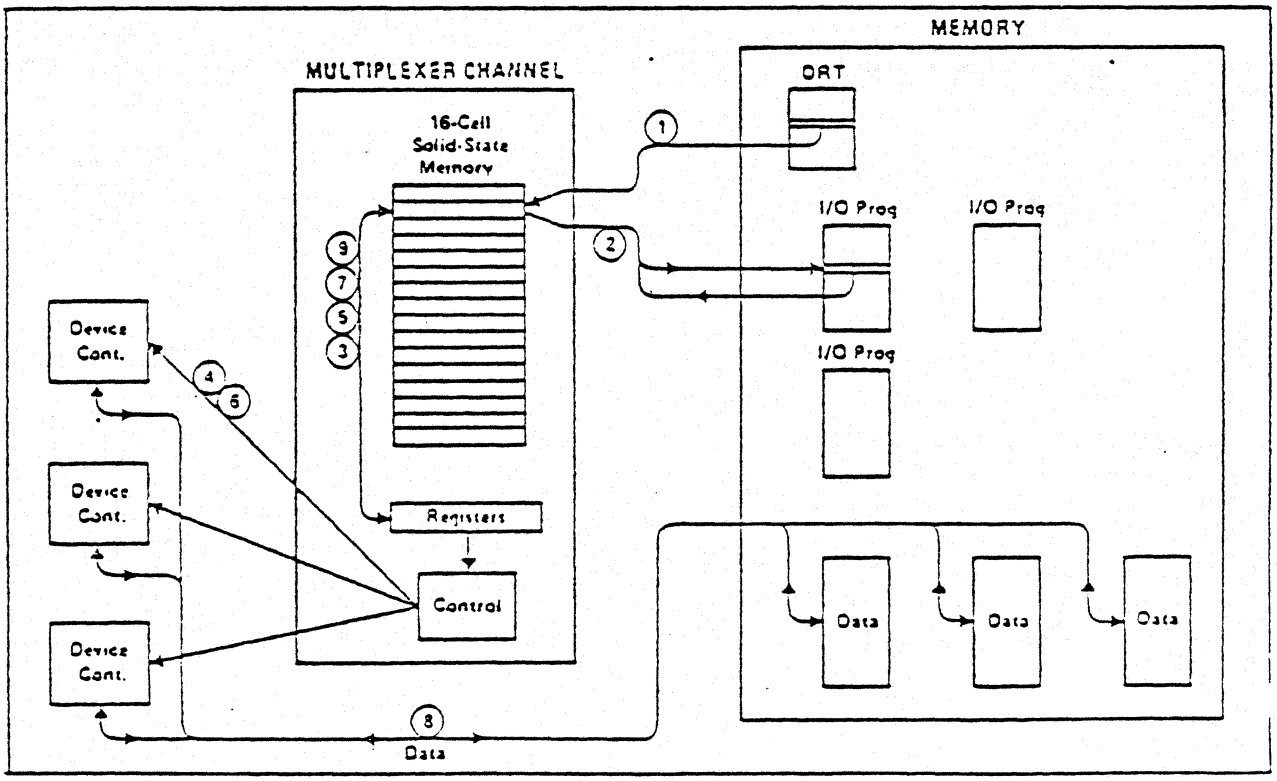
PROGRAMMED I/O

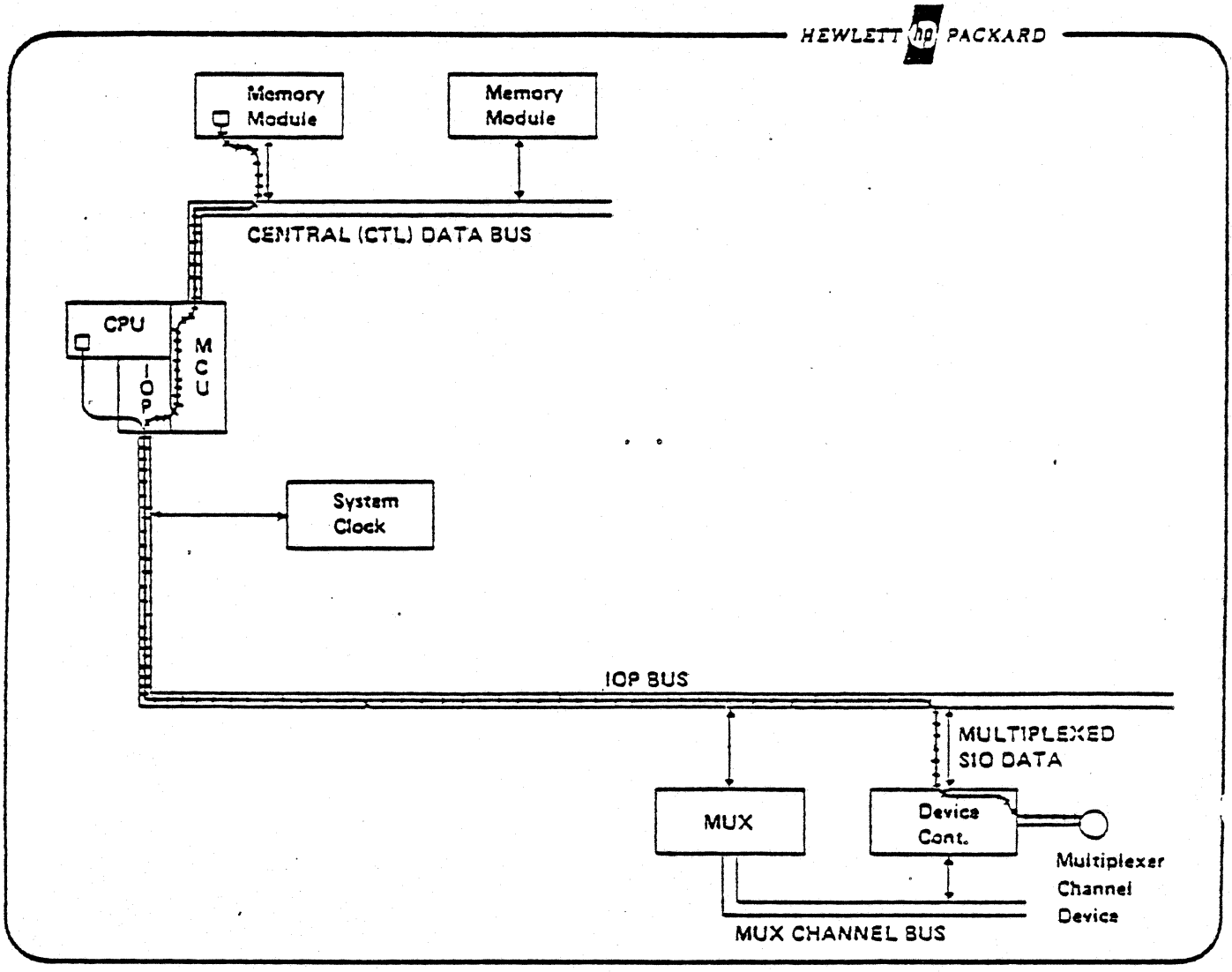
- PROGRAMMED I/O IS USED FOR ALL DEVICES HOOKED ONTO MULTIPLEXER OR SELECTOR CHANNELS. (I.E. DISC, TAPE, LP, ETC.)
- SIO (START I/O) CPU INSTRUCTION TELLS THE IOP TO INITIATE I/O FOR A PARTICULAR DEVICE.
- I/O IS ACCOMPLISHED THROUGH AN SIO PROGRAM IN MEMORY. AN SIO INSTRUCTION CONSISTS OF AN IOCW AND AN IOAW, REFERRED TO AS AN SIO ORDER PAIR.

MULTIPLEXER CHANNEL-

THE MULTIPLEXER CHANNEL ACTS AS A SWITCH TO ENABLE ONE OF ITS ASSOCIATED DEVICE CONTROLLERS TO TRANSFER ONE WORD OF DATA TO OR FROM MEMORY VIA THE TOP AND THEN ALLOW ANOTHER CONTROLLER, BASED ON PRIORITY, TO PERFORM ITS TRANSFER. AT ALL TIMES, THE MULTIPLEXER CHANNEL CONTAINS THE CURRENT I/O PROGRAM DOUBLEWORD FOR EACH OF THE POSSIBLE 16 DEVICE, CONTROLLER. TO ACCOMPLISH THIS, THE MULTIPLEXER CHANNEL HAS A 16-LOCATION, SOLID-STATE MEMORY TO CONTAIN THE 16 I/O PROGRAM WORDS, AND IS RESPONSIBLE FOR FETCHING THE NEXT I/O PROGRAM DOUBLEWORD WHEN NECESSARY.

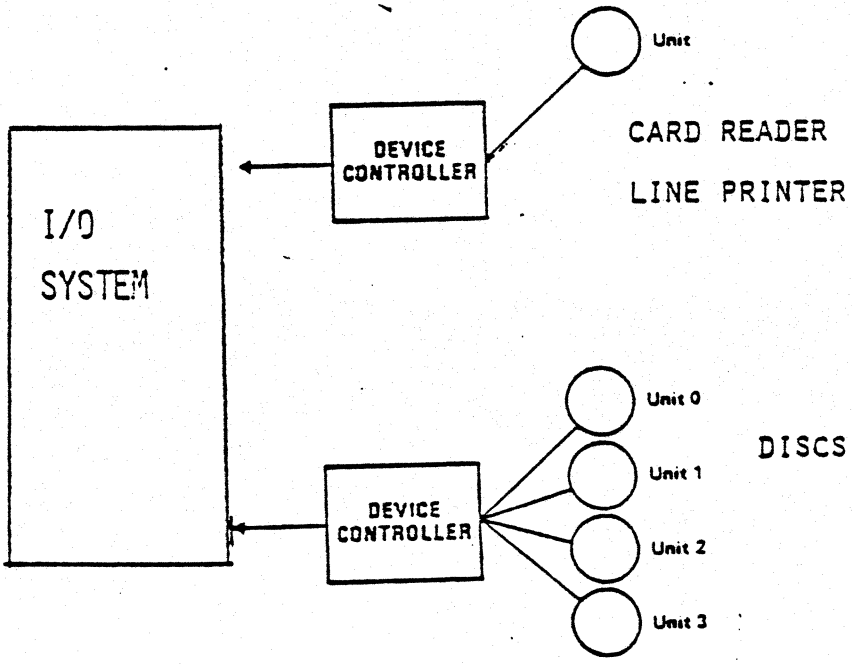
SERVICE REQUEST




HEWLETT hp PACKARD
Memory Module
Memory Module
CENTRAL (CTL) DATA BUS
CPU
IOP
MCU
System Clock
IOP BUS
MULTIPLEXED SIO DATA
MUX
Device Cont.
Multiplexer Channel Device
MUX CHANNEL BUS

DEVICE CONTROLLER

THE HARDWARE LINKAGE BETWEEN THE CPU AND THE I/O DEVICE IS CALLED THE DEVICE CONTROLLER. IT TYPICALLY CONSISTS OF ONE OR MORE LOGIC CARDS. IT MAY DRIVE ONLY ONE DEVICE SUCH AS A CARD READER OR SEVERAL DEVICES SUCH AS THE DISC DRIVE. FOR EACH CONTROLLER THERE IS ONE DRT ENTRY.



- INTERRUPT POLL
  - RESOLVES INTERRUPT PRIORITY BETWEEN CONTROLLERS.
  - CONFIGURED BY THE SEQUENCE WIRED UP ON THE BACKPLANES OF THE CARD CAGES. (BLUE AND WHITE TWISTED WIRE)
  - THE FIRST CONTROLLER THE POLL WIRES PHYSICALLY GO TO FROM THE IOP HAS THE HIGHEST PRIORITY. THE LAST CONTROLLER PHYSICALLY WIRED HAS THE LOWEST PRIORITY.

HP3000 INTERRUPT POLLING SEQUENCE

30032B	TERMINAL DATA INTERFACE
30031A	SYSTEM CLOCK
30104A	PAPER TAPE READER
3055A	SYNC. SINGLE LINE CONTROLLER (ALL USES)
30129A	7905/7920 DISC
30103A	2660 DISC
30110A	7900 DISC
30102A	2888 DISC
30032B-001,-002	TERMINAL CONTROL INTERFACE
30360A	HARDWIRED SERIAL INTERFACE
30126A	PLOTTER
30300/1A	PROGRAMABLE CONTROLLER
30115A	MAGNETIC TAPE
LINE PRINTERS	ALL
30106/7A	CARD READER
30119A	CARD READER/PUNCH
30112A	CARD PUNCH
30105A	PAPER TAPE PUNCH

HEWLETT hp PACKARD

HP 3000 SERVICE REQUEST PRIORITY

Device Name	Product Number	Transmission Mode	Transfer Rate (KW/sec)
7905/7920 Disc	30129A	S	480
2888 Disc	30102A	MS,S (1)	157
7900 Disc	30110A	MS	157
2660 Disc (2)	30103A	MS,S (1)	250
7970 Mag Tape	30115A	MS	36/18
Hardwired Serial Interface	30360	MS	125/62.5
Card Reader	30106/7A	MS	0.8/0.4
Sync. Single Line Contr.	30055A	MS	1.2
Plotter	30126A	MS	0.13
Paper Tape Punch	30105A	MA	0.04
Programmable Controller	30300/1A	MA	262
Line Printer	ALL	MA	250(3)
Card Punch	30112A	MA	0.25
Paper Tape Reader	30104A	D	0.5
Card Reader/Punch	30119A	D	
ATC	30032B	D	1.92

Transmission Mode

S - Selector Channel
MS - Multiplexor Channel (Synchronous)
MA - Multiplexor Channel (Asynchronous)
D - Used in Direct I/O mode only

MUX OUT 0.952 MBYTES
IN 1.038 MBYTES

SELECTOR CHANNEL
2.3 MBYTES

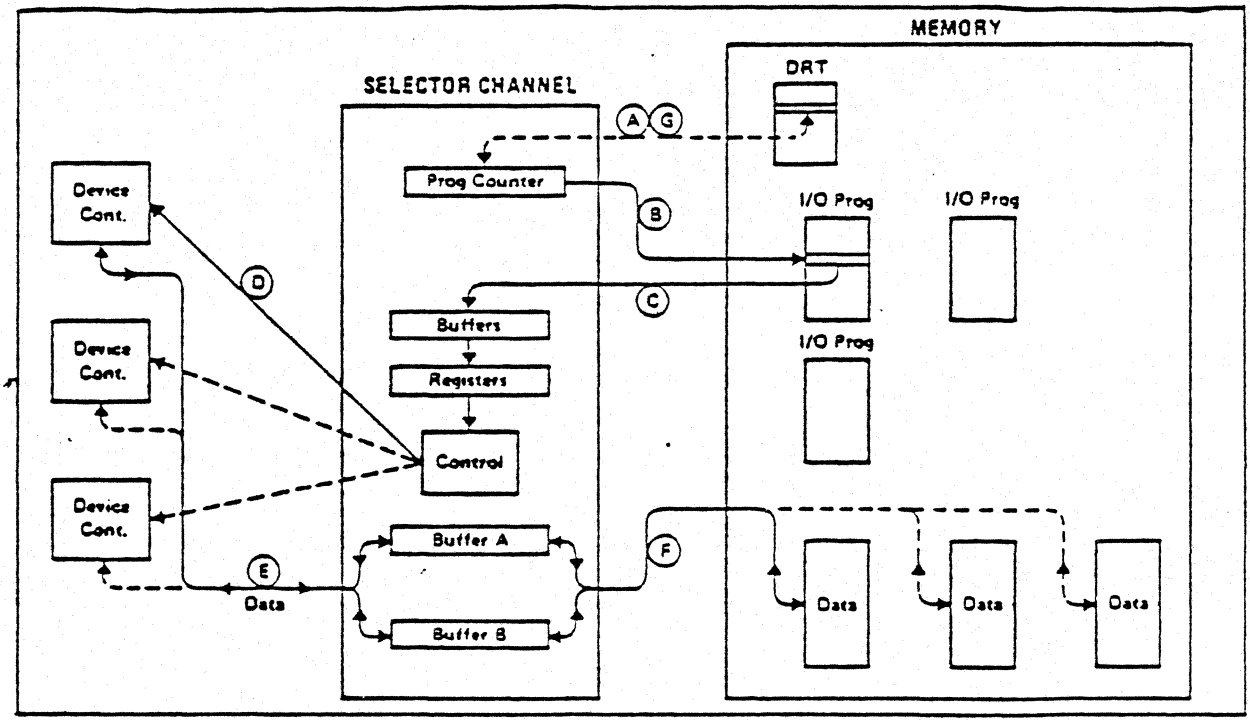
(1) These devices may be used on either the selector or multiplexor channels.

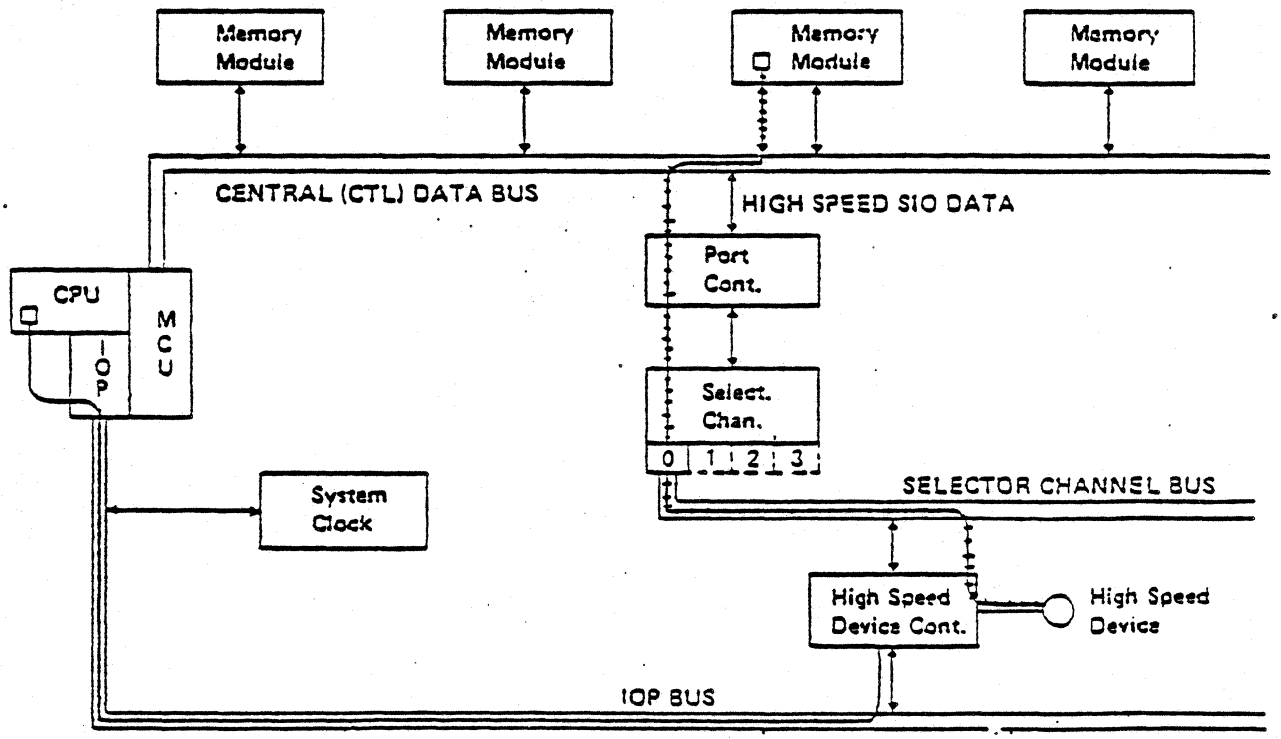
(2) This interface is double-buffered.

(3) This actual device speed is model dependent. Only the highest speed is shown.

SELECTOR CHANNEL-

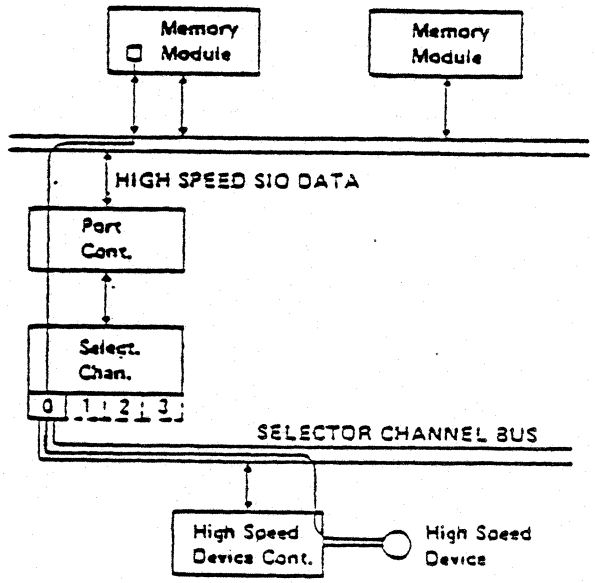
THE SELECTOR CHANNEL ACTS AS A SWITCH, BUT IN A MANNER DIFFERENT FROM A MULTIPLEXER CHANNEL. THE MULTIPLEXER CHANNEL SWITCHES BETWEEN DEVICE CONTROLLERS ON DEMAND, BASED ON HARDWARE PRIORITY, WHERAS THE SELECTOR CHANNEL MAINTAINS THE CONNECTION FOR ONE DEVICE CONTROLLER UNTIL IT HAS COMPLETED THE I/O PROGRAM. THEREFORE, ONLY ONE I/O PROGRAM IS CURRENT AT A GIVEN TIME FOR ONE CHANNEL. ANOTHER MAJOR DIFFERENCE IS THAT THE SELECTOR CHANNEL ACCESSES MEMORY DIRECTLY FOR DATA AND I/O PROGRAM WORD TRANSFERS, RATHER THAN INDIRECTLY THROUGH THE IOP. THESE FEATURES PERMIT A VERY HIGH-SPEED DATA TRANSFER RATE.



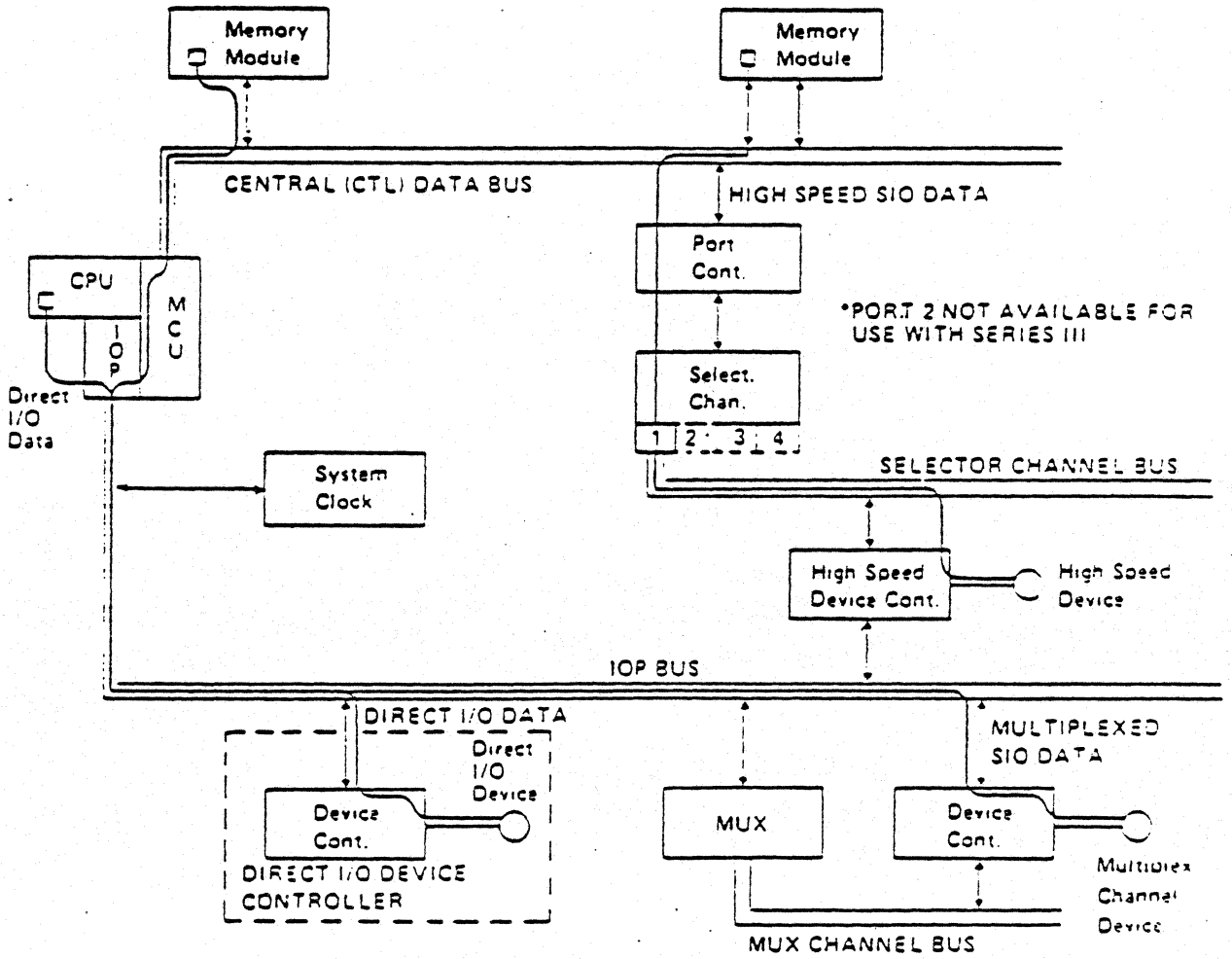

Memory Module
Memory Module
Memory Module
Memory Module
CENTRAL (CTL) DATA BUS
HIGH SPEED SIO DATA
Port Cont.
CPU
IOP
MCU
Select. Chan.
0 1 2 3
System Clock
SELECTOR CHANNEL BUS
High Speed Device Cont.
High Speed Device
IOP BUS

PORT CONTROLLER

THE PORT CONTROLLER PROVIDES FOR PORTS TO THE CENTRAL DATA BUS FOR I/O PROGRAMS AND DATA TRANSFERS BETWEEN SELECTOR CHANNELS AND MEMORY MODULES. EVEN THOUGH THERE ARE HOOKUPS FOR MORE THAN ONE SELECTOR CHANNEL ONLY TWO CAN BE INSTALLED ON A SYSTEM. ONE REASON FOR THIS IS THE BAND WIDTH OF THE CENTRAL DATA BUS.



REVIEW OF I/O DATA ROUTES

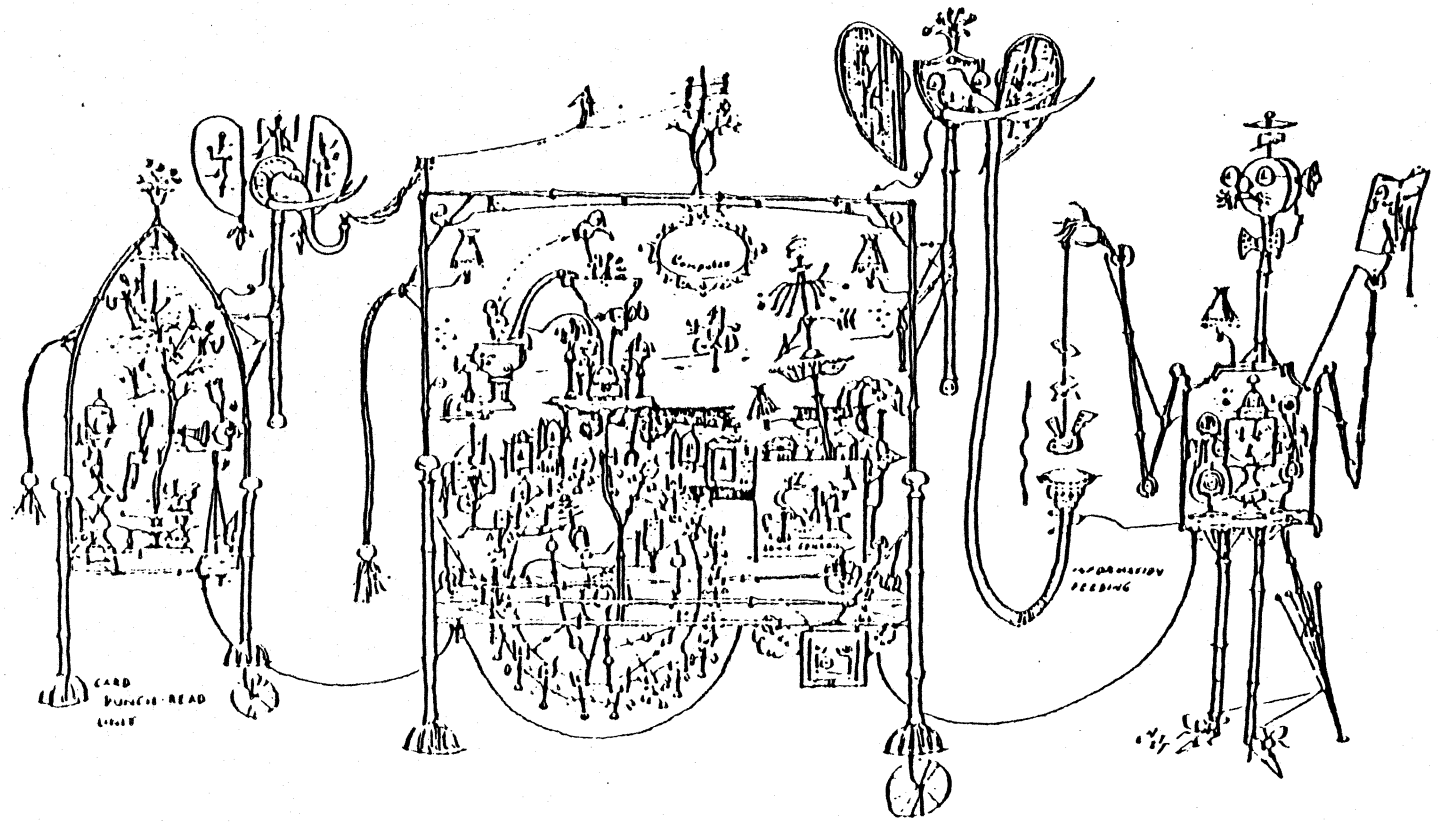


1. DIRECT I/O
2. PROGRAMMED I/O (MULTIPLEXER CHANNEL)
3. PROGRAMMED I/O (SELECTOR CHANNEL)

HEWLETT hp PACKARD

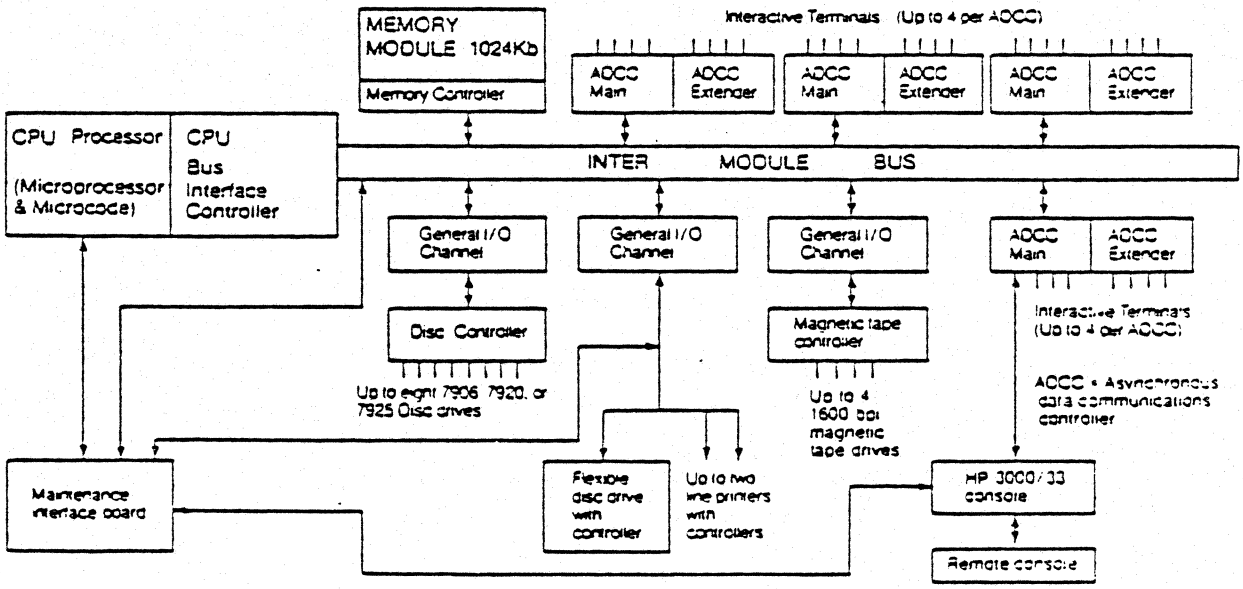
Series 33

OPERATIONAL OVERVIEW



SERIES 33 HARDWARE

- AMIGO HARDWARE (HP 300) WITH MPE.
- NO SEPARATE IOP.
- INTER-MODULE BUS (IMB) REPLACES CTL BUS/IOP BUS.
- GENERAL I/O CHANNELS REPLACE SELECTOR/MULTIPLEXER CHANNELS.
- ANYNCHRONOUS DATA COMMUNICATIONS CONTROLLER (ADCC) REPLACES ATC FOR TERMINAL I/O.



HP 3000 SERIES 33 HARDWARE ORGANIZATION (MAXIMUM CONFIGURATION)

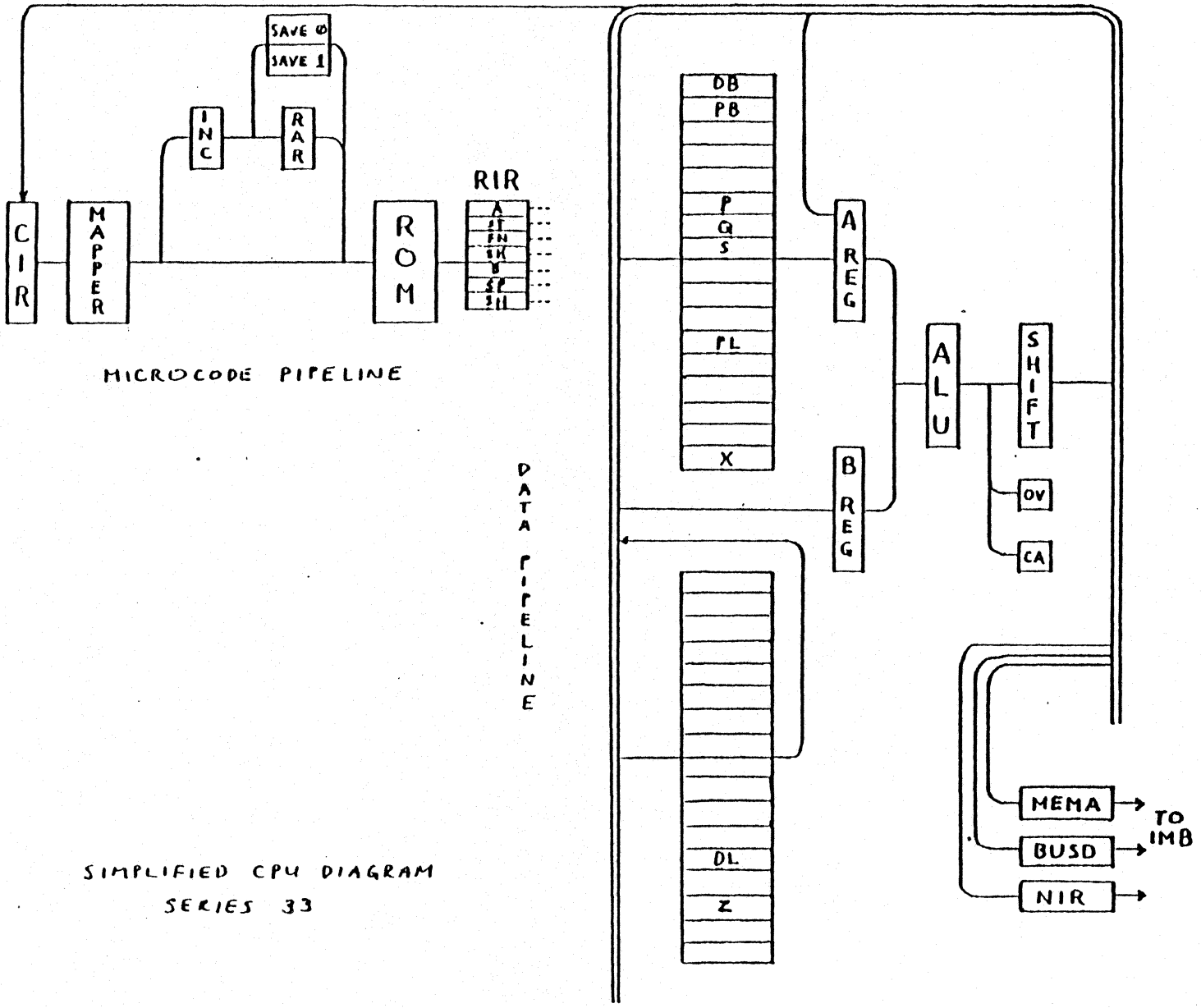
WHY IS SERIES 33 SLOWER THAN SERIES II/III?

- VERY BASIC MICROCODE PIPELINE CONCEPT ON SERIES 33. ALTHOUGH CPU CYCLE TIME IS ONLY 90 NSECS, IT CAN TAKE 3 TO 7 CLOCK CYCLES PER MICROINSTRUCTION, OR 270-630 NSECS. (COMPARE THAT TO 175 NSECS FOR SERIES II/III.)

- SERIES 33 HAS ONLY 2 HARDWARE TOS REGISTERS, COMPARED TO 4 ON SERIES II/III.

- ONLY ONE MAJOR DATA BUS, THE IMB, ON SERIES 33. SERIES II/III HAS CTL BUS AND IOP BUS.

- I/O ON SERIES 33 AFFECTS THE SYSTEM BECAUSE THERE IS NO IOP. THE CPU MUST INTERLEAVE I/O TRANSFERS BETWEEN NORMAL OPERATIONS, UPON REQUEST FROM DEVICE CONTROLLERS.



SIMPLIFIED CPU DIAGRAM
SERIES 33

GENERAL I/O CHANNEL (GIC)

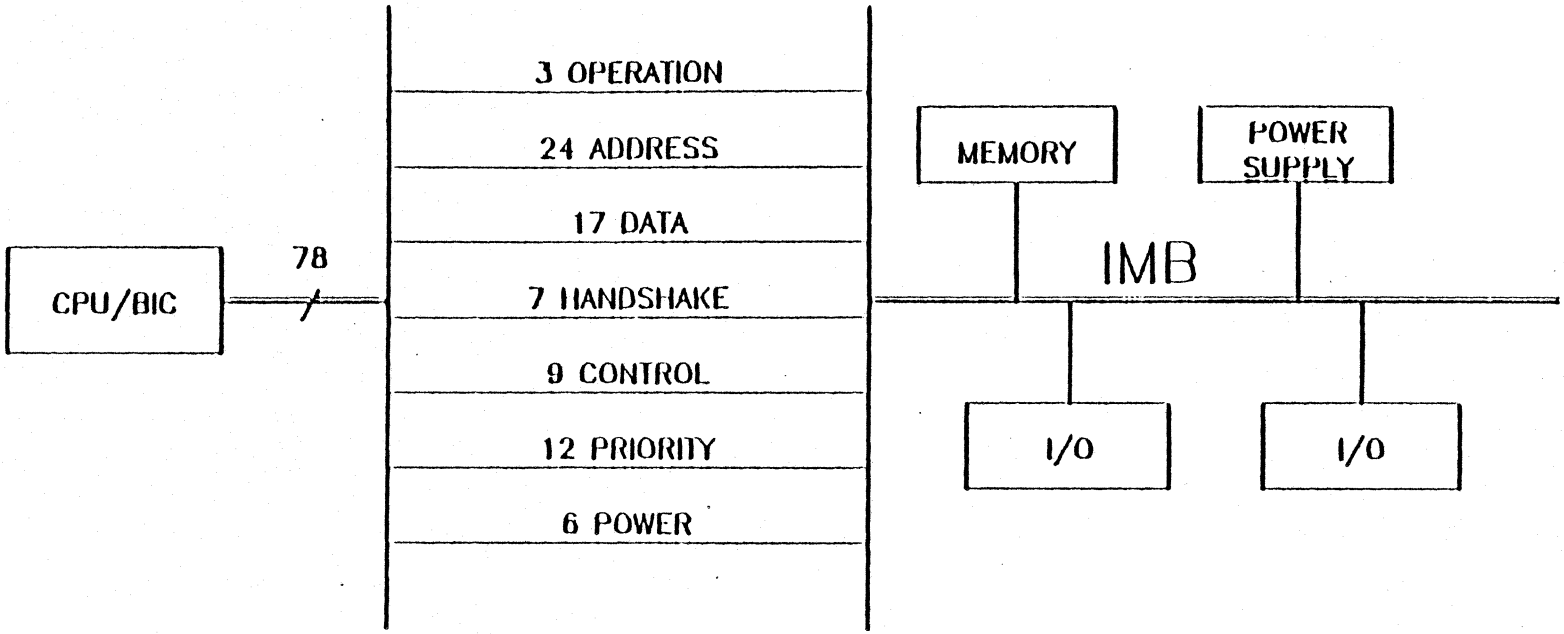
- TRANSLATES I/O COMMANDS FROM THE CPU INTO HP - IB PROTOCOL.
- HP - IB TRANSFER RATE: APPROX. 1 MB/SEC.
- GIC CONTAINS DIRECT MEMORY ACCESS HARDWARE (DMA) TO TRANSFER DATA DIRECTLY TO/FROM MEMORY VIA IMB.

DMA ALLOWS "OVERLAPPING" CPU AND I/O OPERATIONS.

- MAXIMUM LENGTH OF TRANSFER = 65 kB
- GIC TALKS TO CPU AND MEMORY IN WORDS, AND TALKS TO DEVICES IN BYTES

IMB CHARACTERISTICS
SERIES 30/33

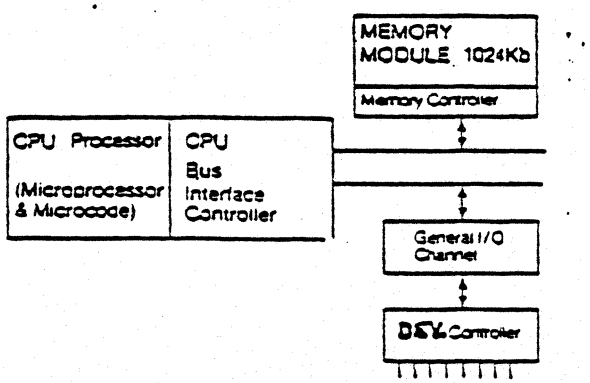
- BIDIRECTIONAL THREE-STATE TRANSCEIVERS ON OPERATION
- BUS TERMINATION BY 330 OHM NETWORK FOR 3.0 VOLT STEADY-STATE
- ONE (1) MEGABYTE/SECOND TRANSFER RATE
- 2 MEGABYTE MAIN MEMORY ADDRESSIBILITY
  (32 MEGABYTE FUTURE USE)
- MAXIMUM OF 15 I/O CHANNELS
- FULLY ASYNCHRONOUS OPERATION
- INTERLEAVED ADDRESS/DATA HANDSHAKING
- DIRECT INTER-MODULE COMMUNICATION (INHERENT DMA)



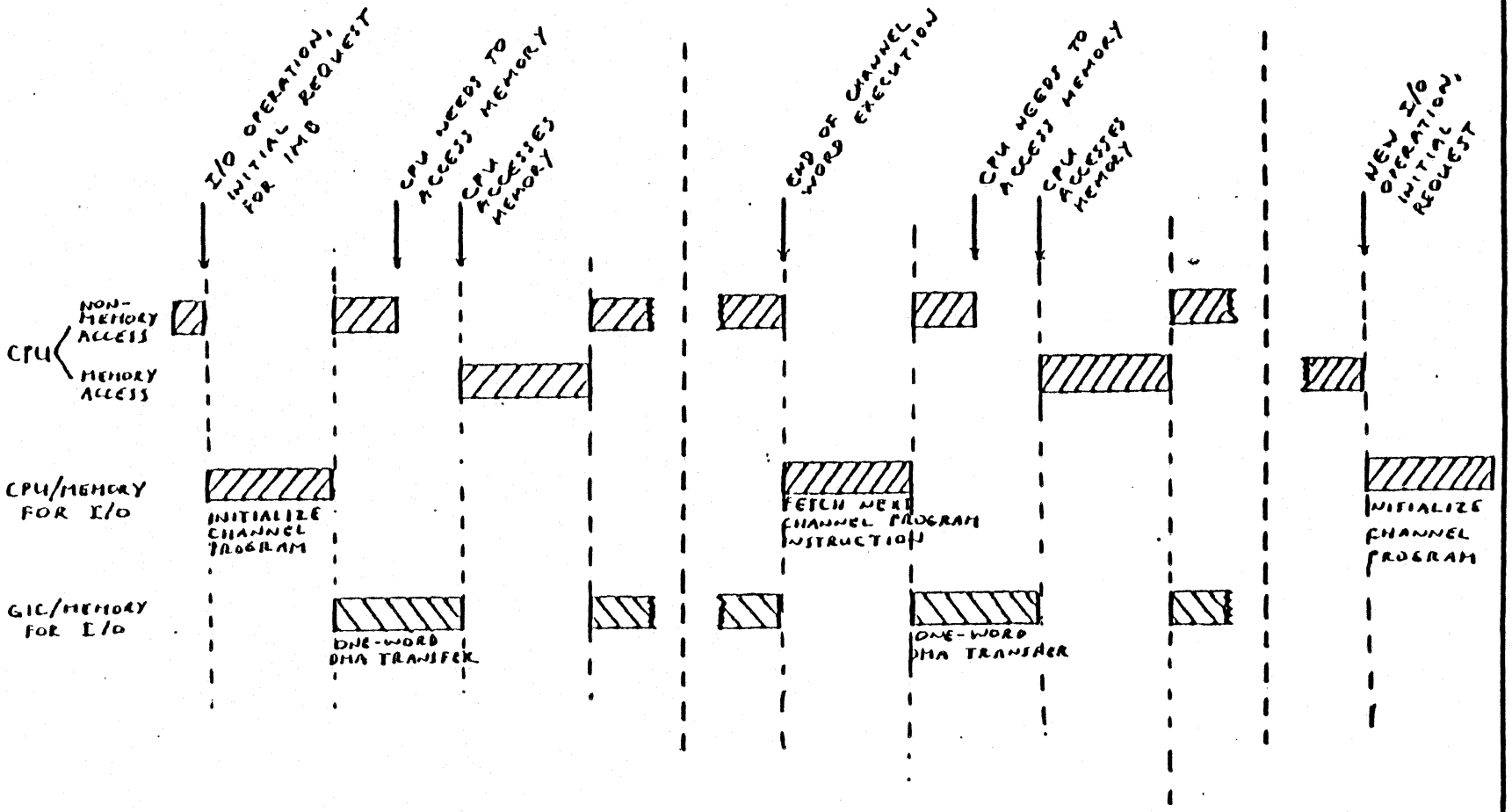
CHARACTERISTICS OF INTERMODULE BUS

SERIES 30/33

"OVERLAPPING" CPU AND I/O OPERATIONS



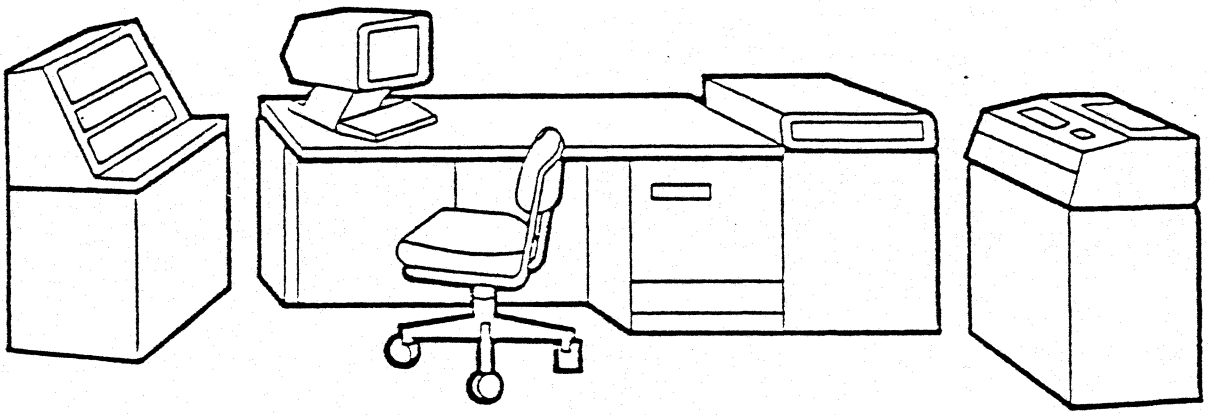
1. GIC MUST REQUEST ACCESS TO THE IMB FROM THE CPU.
2. IF THIS IS THE FIRST REQUEST FOR THE CURRENT I/O OPERATION, THE CPU INITIALIZES THE GIC'S DMA HARDWARE THROUGH A MICROCODE CHANNEL PROGRAM.
3. CPU THEN RELINQUISHES CONTROL OF THE IMB TO THE GIC.
4. GIC TRANSFERS ONE WORD OF DATA DIRECTLY TO/FROM MEMORY, INDEPENDENT OF THE CPU. CPU IS FREE FOR OTHER ACTIONS.
5. WHEN DONE, THE IMB IS RETURNED TO CPU CONTROL.
6. GIC WILL RE-REQUEST FOR THE IMB AND TRANSFER ANOTHER WORD UNTIL THE I/O OPERATION IS COMPLETE.



OVERLAPPING CPU AND I/O OPERATIONS

HEWLETT hp PACKARD

HP 3000 SERIES 44
GRIZZLY



HP 3000 SERIES 44 SYSTEM PROCESSOR UNIT

Hardware Supplied

- Central processing unit (CPU)
- 274 firmware instructions
- System clock
- Control and Maintenance Processor (CMP)
- General I/O Channel for magnetic tape drives
- General I/O Channel for line printers, disc drives, and Intelligent Network Processors (INPs)
- 1 Mb fault control memory with memory controller
- System desk mainframe, one card cage, and power supplies
- 3 spare I/O slots and space for 2 Mb memory
- Room for 32 point-to-point terminals including system console (can be expanded to provide room for 60 point-to-point terminals)
- Built-in isolation transformer

Software Supplied

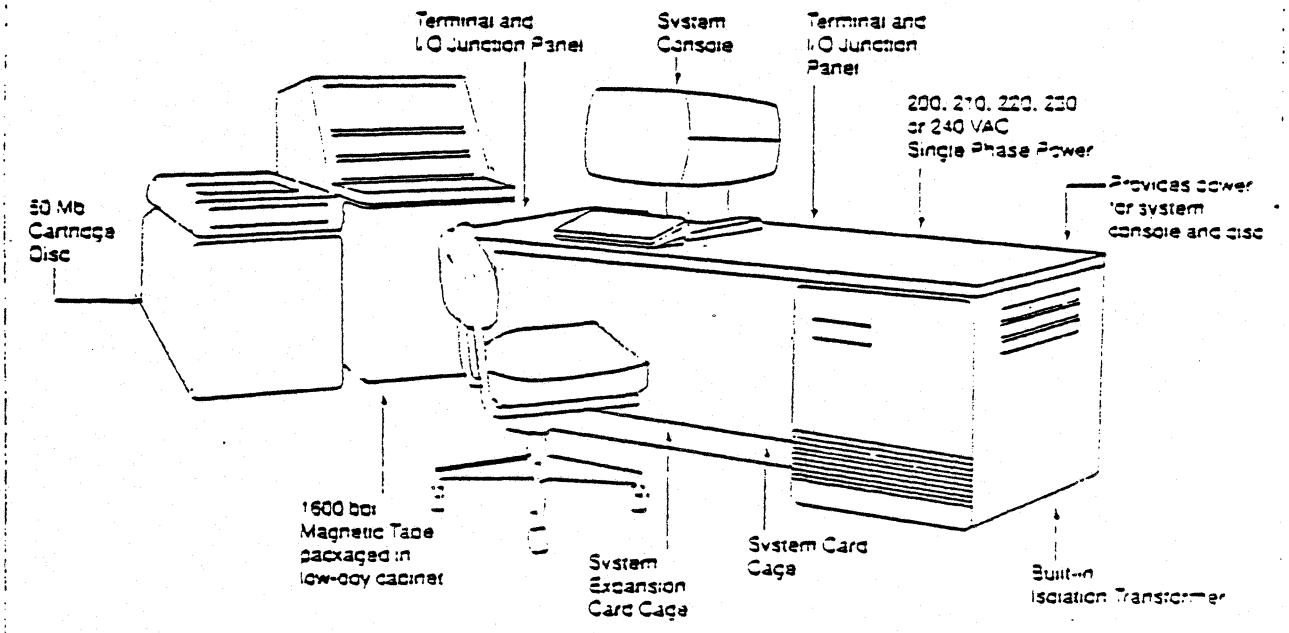
Standard on each HP 3000 system is the Fundamental Operating Software which includes:

- Multiprogramming Executive (MPE) Operating System
- Text editor (EDIT/3000)
- File copying utility (FCOPY/3000)
- Sort and merge package (SORT-MERGE 3000)
- Data base management system (IMAGE 3000)
- Data base inquiry language (QUERY/3000)
- Keyed Sequential Access Method Software (KSAM 3000)
- Forms management software (HP V/3000)
- Facility to execute compiled programs without the source language compiler on the system (except for programs written in APL 3000)
- Complete user manual set

Required Hardware Ordered Separately

- System console (Any HP 262x, 264x, or 2635B terminal)
- System console cable (order separately for 262x or 264x console)
- One Asynchronous Data Communications Controller (ACCC-Main) (provides four asynchronous terminal ports)
- System disc with controller (20 Mb, 50 Mb, or 120 Mb)
- 1600 bpi magnetic tape drive

HP 3000 Series 44 Minimum Configuration



Note: Under certain power line conditions at the system site, an isolation transformer or other line conditioning equipment may be required for add-on disc and magnetic tape drives.

Optional Hardware:

I/O EXPANSION
- 30018A-044 Asynchronous Data Communications Controller-Main
- 30019A-044 Asynchronous Data Communications Controller-Extender
- 30079A-044 General I/O Channel
- 30020A Intelligent Network Processor
- 30092A 512 Kb fault control memory module. Memory sizes supported are: 1 Mb, 1.5 Mb, 2 Mb, 2.5 Mb, 3 Mb, 3.5 Mb, and 4 Mb
- 30094A Add-on memory controller for more than 2 Mb of main memory
- 30087A Expansion kit including second card cage, cooling unit, and power supply
- 26069A-344 Line printer interface for existing 2613A/17A/19A line printers to upgrade to HP-IB. Order 261x-344 for new printers.

DISC DRIVES
- 9895A-010-333 1.2 Mb Flexible disc drive
- 7906M-102 or 7906S 20 Mb Disc drive
- 7920M-102 or 7920S 50 Mb Disc drive
- 7925M-102 or 7925S 120 Mb Disc drive
- 7925T 240 Mb Disc storage system

MAGNETIC TAPE DRIVES
- 7970E-426 or 7970E-421 Magnetic tape drive

PRINTERS
- 2601A 40 cps daisy-wheel printer
- 2608A-344 400 lpm line printer
- 2613A-344 300 lpm line printer
- 2617A-344 600 lpm line printer
- 2619A-344 1000 lpm line printer
- 2631B-331 180 cps printer
- 2680A page printer

TERMINALS
- 2621A/P, 2624A, 2626A, 2640B, 2642A, 2645A, 2647A, 2648A, 2635B, 2675A Terminals
- 3075A, 3076A, 3077A Data capture terminals

PLOTTERS
- Multi-pen plotters as terminal devices

MODEMS
- 37210T 4800 bps modem
- 37220T 9600 bps modem
- 37230T Short Haul modem
- 13265A 300 baud modem for 262x terminals

CARD READERS
- 80-column card card reader (3Q81)
- 7260A Optical mark reader as a terminal device

Optional Software:

INFORMATION SYSTEMS
- Decision Support Graphics/3000
- Text and Document Processor/3000

MANUFACTURING APPLICATIONS
- Materials Management/3000

LASER PRINTER
- Interactive Design System/3000
- Interactive Formatting System/3000

FLEXIBLE DISC
- Flexible Disccopy/3000

COMPILERS
- COBOL II/3000, COBOL/3000
- RPG/3000
- BASIC/3000
- FORTRAN/3000
- SPL/3000

DATA COMMUNICATIONS
- Distributed Systems/3000
- Remote Job Entry/3000
- Multileaving Remote Job Entry/3000
- Interactive Mainframe Link/3000
- Multipoint Terminal Software/3000

EDUCATIONAL
- Student Information System/3000
- College Information System/3000

ICF 34 'JAGUAR' PROCESSOR

JAGUAR COMMUNICATES WITH MEMORY OVER IMB.IDENTICAL TO SERIES 3x CPU

COMMUNICATES WITH I/O OVER IMB. IDENTICAL TO SERIES 3x CPU.

IMPLEMENTS 25A CHANNEL PROGRAM AND MACRO I/O INSTRUCTIONS.

DIAGNOSTICS.

REQUIRES MORE POWER.

GENERAL CPU OPERATIONS FOR S 44

1. MACRO COMES IN FROM MEMORY OVER IMB

2. LOADED INTO NIR (ALU).

3. AT EXECUTION LOADED INTO CIR (ALU).

4. LUT (CTL) TRANSLATES CIR INTO A MICROCODE STARTING ADDRESS FOR THE MICROPROGRAM TO IMPLEMENT MACRO.

5. ADDRESS IS SUPPLIED TO CTL STORE ADDRESS BUS.

6. CTL STORE PRESENTS 48 BIT MICRO INSTRUCTION TO CTL BOARD INSTRUCTION DECODING CIRCUITRY.

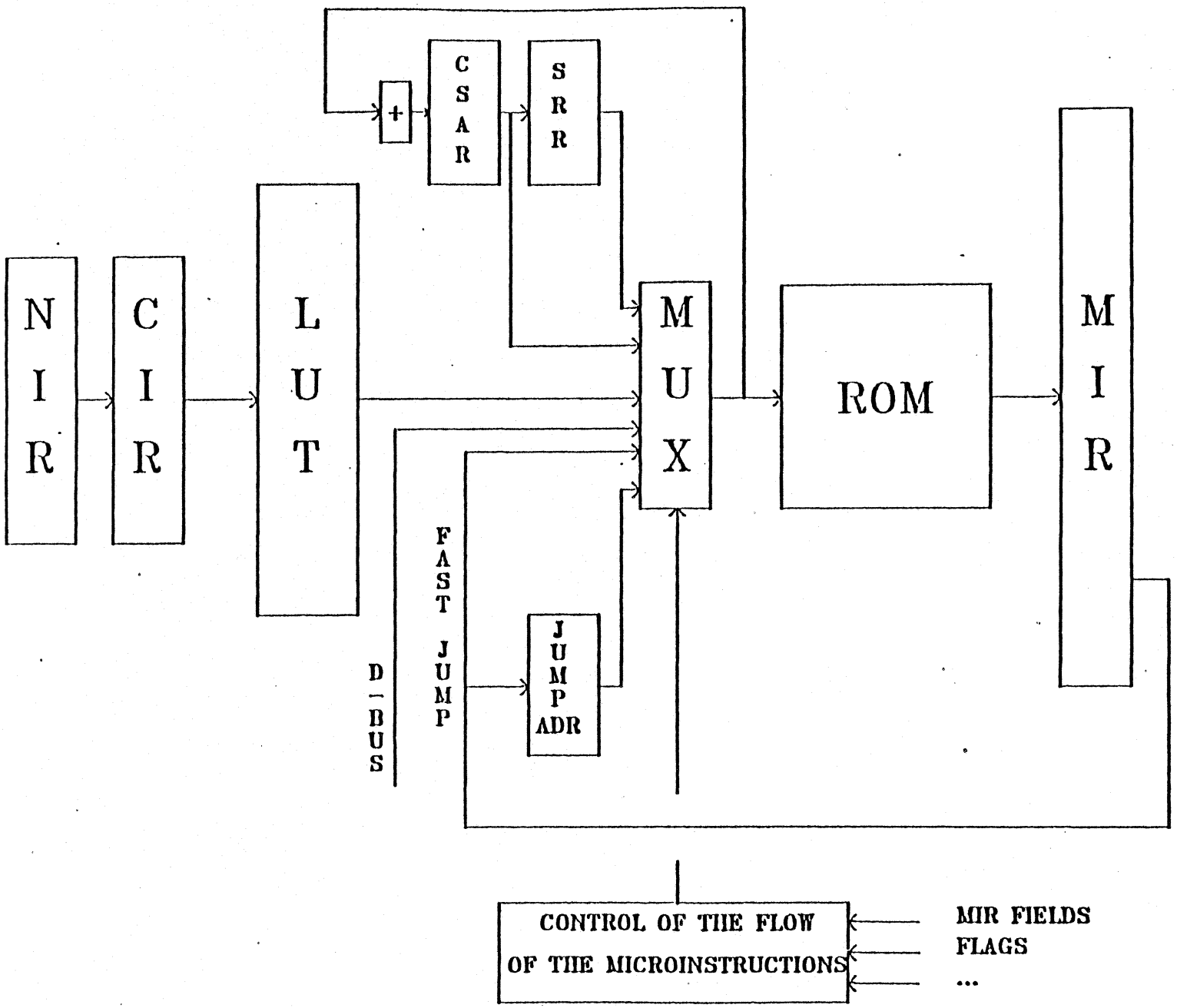
7. DECODED INFORMATION IS THEN PASSED TO ALU BOARD.

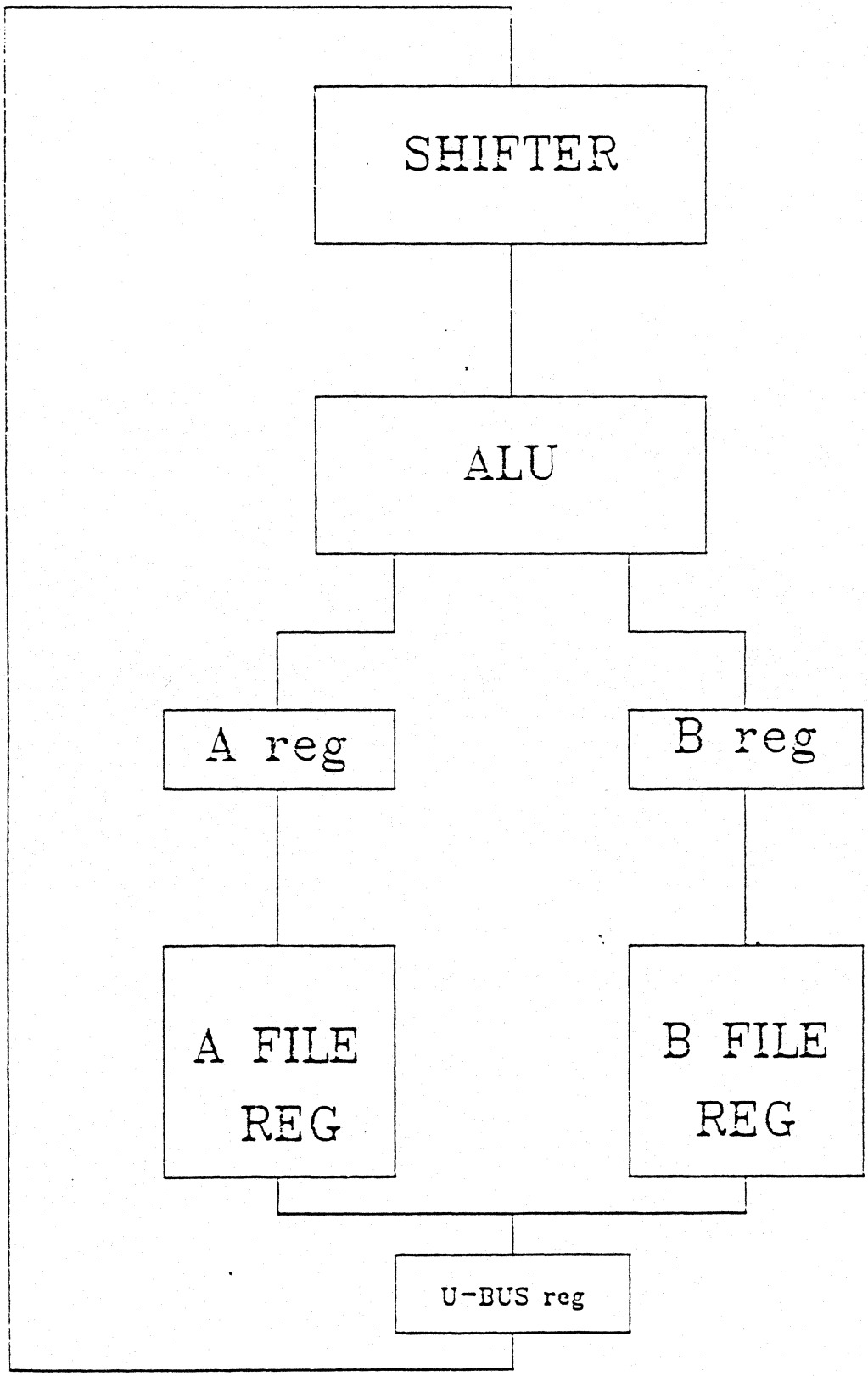
8. ALU BOARD FETCHES AND THEN EXECUTES THE GIVEN FCN .

9. RESULTS STORED INTO REGISTERS SPECIFIED BY CTL BD.

10. ONCE COMPLETED THE CTL BD THEN EXECUTES NEXT MICRO IN LINE OR PERFORMS NECESSARY PROGRAM JUMPS.

11. LAST MICRO INSTRUCTION WILL BE "NEXT" INSTR. THIS INFORMS CTL BD THAT IT IS READY TO LOAD NEXT MACRO FROM NIR TO CIR.


NIR
CIR
LUT
+
CSAR
SRR
MUX
ROM
MIR
FAST JUMP
D-BUS
JUMP ADR
CONTROL OF THE FLOW
OF THE MICROINSTRUCTIONS
MIR FIELDS
FLAGS
...


SHIFTER
ALU
A reg
B reg
A FILE REG
B FILE REG
U-BUS reg

HEWLETT PACKARD

HP 3000 Performance Guide For Installed Systems

This guide is designed to provide existing HP 3000 users with the system performance information they need to help plan for future growth through memory expansion or system upgrades. The data shown are intended to serve as guidelines of HP 3000 performance in a general-purpose EDP environment, including on-line transaction processing, batch processing, and program development.

Performance in this guide is measured in terms of transaction throughput (total number of on-line transactions per hour handled by the system), and response time (averaged for all on-line terminals) in seconds. The guide compares the performance of the most common models of the HP 3000 computer family — the Series 30, Series 33, Series II, and Series III — to that of the Series 44, which serves as an upgrade for all other HP 3000 systems. Performance of the different systems is also measured with varying amounts of main memory, and with MPE-III and MPE-IV versions of the MultiProgramming Executive (MPE) operating system. All tests use a standard application load — detailed at the end of the Performance Guide — unless it is stated that the standard load plus heavy additional on-line and batch processing has been used.

Actual HP 3000 system performance will vary, depending upon your system configuration and application load. Configuration factors affecting performance include the System Processor Unit series, the amount of main memory, the system disc(s) configuration, and terminal and data communications usage. Actual performance in any particular configuration will depend upon the application load and its mix of such factors as data entry, inquiry, program compilations, file sorts, batch processing, and data communications.

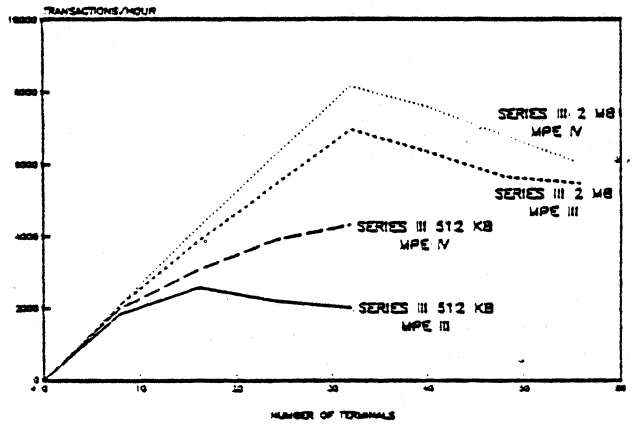
MPE-IV PROVIDES IMPROVED PERFORMANCE OVER MPE-III

Most HP 3000 Series 30, 33, II, and III systems will experience a significant improvement in transaction throughput and response time simply by converting from MPE-III to MPE-IV. The software compatibility of MPE-III and MPE-IV means that converted programs rarely, if ever, require modification or even recompilation. There is no additional charge to supported HP 3000 systems for the use of MPE-IV. Enhancements in MPE-IV that contribute to its improved transaction throughput are an improved memory manager, a more efficient file system, and greater system tuning capabilities through the dispatcher/scheduler.

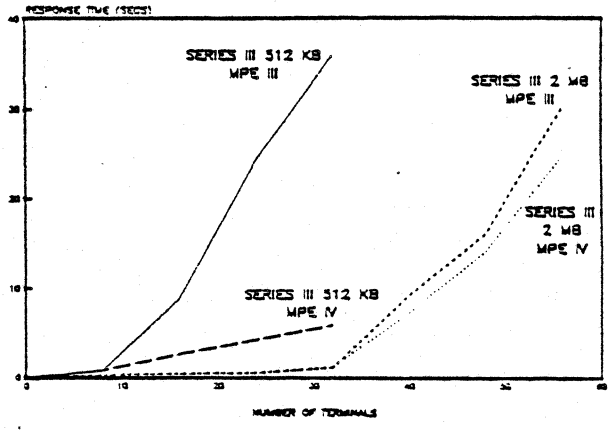
HP 3000 Series II and III

The graphs below compare transaction throughput and response times of the Series II and Series III, using MPE-III and MPE-IV. Performance of the Series II with its maximum memory of 512 Kbytes is nearly identical to that of the Series III with the same amount of memory.

SERIES III TRANSACTION RATES



SERIES III RESPONSE TIMES



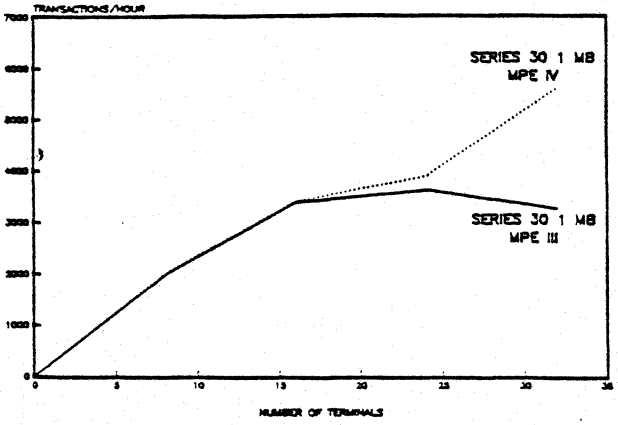
Conclusion: Using the test application, MPE-IV showed a substantial improvement in both transaction throughput and response time over MPE-IV, especially when running 32 to 56 terminals, and when using small memory sizes.

(continued)

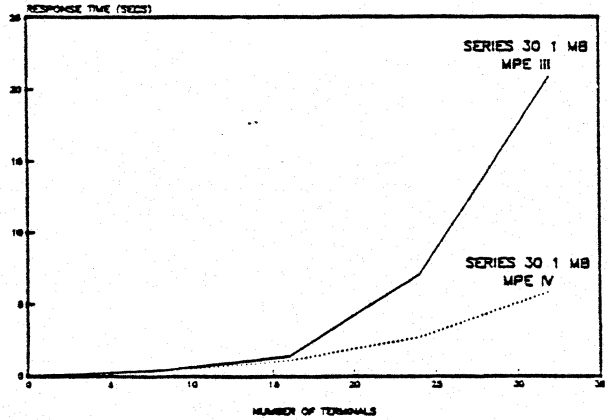
HP 3000 Series 30 and 33

The graphs below compare transaction throughput and response time of the Series 30 with one megabyte of memory using MPE-III and MPE-IV. The number of terminals was varied from eight to 32. Performance of the Series 33 is identical to that of the Series 30 with the same amount of main memory.

SERIES 30 TRANSACTION RATES



SERIES 30 RESPONSE TIMES

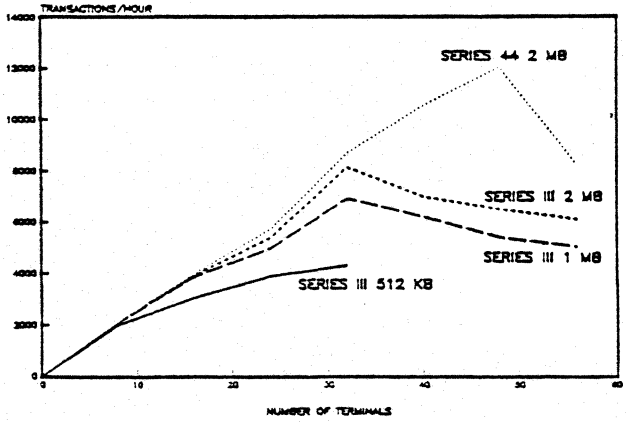


Conclusion: Series 30 and 33 systems with a large number of on-line terminals — over 24 in the test application — experience the greatest performance improvement with MPE-IV over MPE-III.

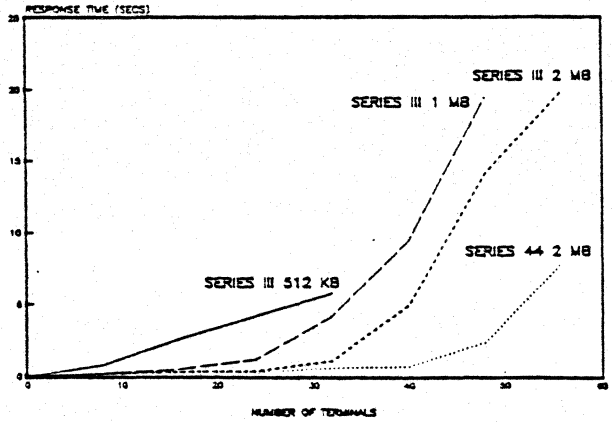
UPGRADING FROM SERIES II OR SERIES III TO SERIES 44

The HP 3000 Series 44 is the highest performance member of the HP 3000 family. The Series 44 provides a CPU with 60% more performance in raw processing power than that of the Series III, and supports up to four megabytes of main memory — twice as much as the Series III. The graphs below show the improved performance that can be achieved by upgrading from a Series II or III to a Series 44. Performance of the Series II with its maximum memory of 512 Kbytes is nearly identical to that of the Series III with 512 Kbytes. All tests were run using MPE-IV. The first two graphs (below) use the standard application load; the next two graphs (following page) use the standard application load plus heavy additional on-line and batch processsing.

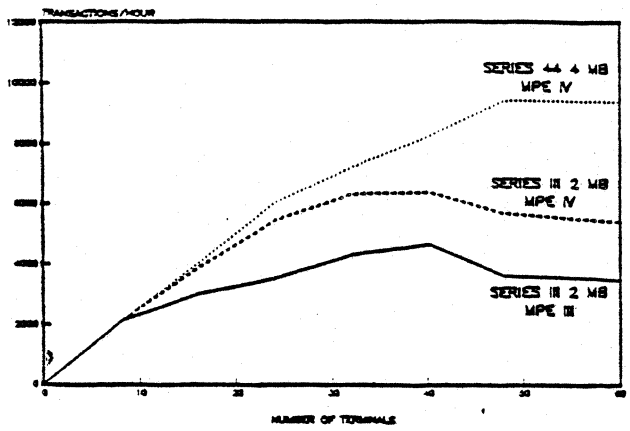
SERIES III AND SERIES 44 TRANSACTION RATES



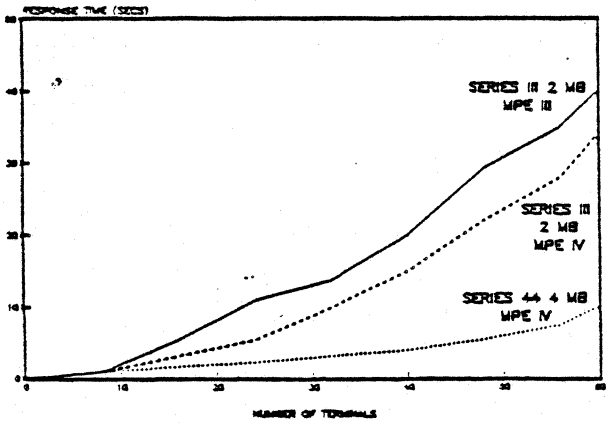
SERIES III AND SERIES 44 RESPONSE TIMES



SERIES III AND SERIES 44 TRANSACTION RATES
HEAVY LOAD



SERIES III AND SERIES 44 RESPONSE TIMES
HEAVY LOAD



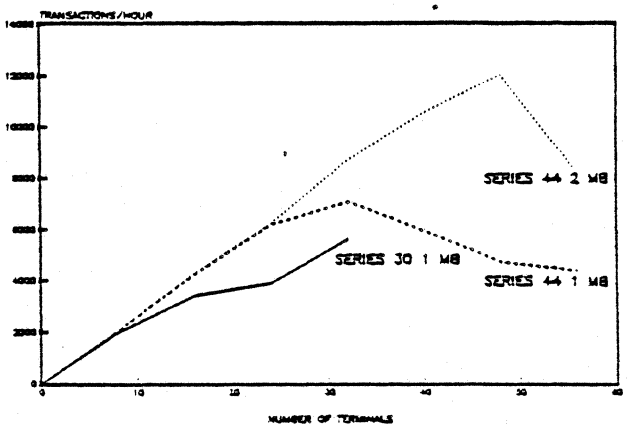
Conclusion: For the test applications, the Series 44 continued to provide transaction throughput and response time improvements at and beyond the terminal load where the Series III had achieved its maximum throughput. The Series 44 also provides extensive growth capability through the addition of main memory.

UPGRADING FROM SERIES 30 OR SERIES 33 TO SERIES 44

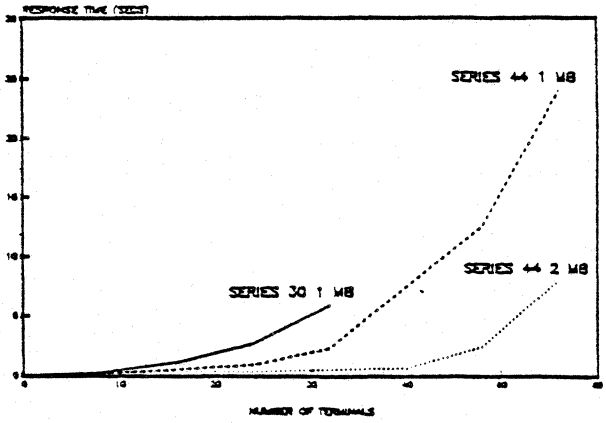
Upgrades from the Series 30 or 33 to the Series 44 are fully peripheral compatible as well as software compatible. Consequently, these upgrades offer very favorable price/performance, and can be implemented with a minimum of disruption to your operating routine. Series 44 upgrades from the Series 30 or 33 typically require only a few hours for on-site installation.

The graphs below show the improved performance resulting from upgrade of a Series 30 or 33 to a Series 44. Performance of the Series 33 is identical to that of the Series 30. All tests were run using MPE-IV.

SERIES 30 AND SERIES 44 TRANSACTION RATES



SERIES 30 AND SERIES 44 RESPONSE TIMES



Conclusion: The Series 44 provided significant transaction throughput and response time improvements over the Series 30 and 33 at essentially all terminal loads. The improvement in transaction throughput and response time provided by the Series 44 over the Series 30 or 33 increases with the load of the application.

(continued)

SERIES 44 THROUGHPUT GAINS OVER SERIES 30/33 AND SERIES III

The following charts extract data from the preceding graphs and reduce it to a set of gain factors comparing expected throughput of the Series 44 to that of the Series 30/33 and the Series III. These gain factors, based on the test applications in this guide, are intended as guidelines only for gauging approximate Series 44 throughput with different amounts of memory. Actual throughput gains achieved in practice will vary, depending upon your application.

How Gain Factors Were Derived

There are two sets of gain factors: one comparing Series 44 throughput with that of a 1 Mb Series 30/33; the other comparing Series 44 throughput with that of a 2 Mb Series III. MPE-IV was used throughout, so the gain factors show the improvement in throughput of the Series 44 with MPE-IV over and above that of the Series 30/33 or Series III with MPE-IV. The two sets of throughput factors were derived by dividing:

- the throughput (transactions per hour) of the Series 44 with the amount of memory indicated below, at the number of terminals for which maximum throughput was achieved, by
- the throughput of the Series 30/33 with 1 Mb memory (for the first set of factors), and of the Series III with 2 Mb memory (for the second set of factors), at the number of terminals for which maximum throughput was achieved on those systems.

This method was used to calculate all gain factors for 1 Mb, 2 Mb, and 4 Mb of Series 44 memory. The standard application load was used at 1 Mb and 2 Mb; the heavy workload was used at 4 Mb. Extrapolation from these data points was used to derive gain factors for the Series 44 with 1.5 Mb. These data points are the larger values in the ranges shown below.

The number of terminals in use when maximum throughput was achieved, as well as the average response time of those terminals, varied according to the system and its memory size, as can be seen on the preceding graphs in this guide.

Amount of Series 44 Memory	Expected Series 44 Throughput Gains over 1 Mb Series 30/33
1.0 Mb	1.0 – 1.3
1.5 Mb	1.5 – 1.8
2.0 Mb	2.0 – 2.3

Amount of Series 44 Memory	Expected Series 44 Throughput Gains over 2 Mb Series III
2 Mb	1.3 – 1.5
4 Mb	1.5 – 1.8

TEST ENVIRONMENTS

The standard application workload which was used for these tests was chosen to represent a general-puroose EDP environment. A mix of the following activities was used.

- **Program Development:** 1200 line COBOL programs: compilation and preparation.
- **V/3000 and IMAGE/3000:** COBOL program verified and entered data against an IMAGE data base, then updated that data base. The data base consisted of 50,000 records organized into ten detail and 18 master data sets; V/3000 data entry activity consisted of a series of three screens.
- **Data Inquiry:** QUERY/3000 was used for on-line data base inquiry and reporting; the data base used was the same as described above; the inquiries were FIND's and REPORT's.
- **MPE Commands:** Commands such as LISTF, HELP, and SHOWJOB were executed.
- **Batch:** A COBOL program read 10,000 records from a file and sorted them on two keys; a COBOL report program produced a sorted report from the data base, printing summary totals for each sort field; the file was sorted twice and two reports generated.

Some tests used the standard workload plus additional on-line and batch processing. The additional processing workload consisted of:

- Two COBOL compiles of 1200 lines each with a maximum stack size of 25,000 words
- Two batch jobs as described in the standard application workload above. The jobs ran concurrently at all times; upon a job completion, a new job was initiated.
- The V/3000 application described above was modified for larger memory utilization.

As the number of terminals varied, the mix of terminals was held fixed. The following mix of on-line terminals was used in the test environment: COBOL program development, 10% of terminals, V/3000 and IMAGE/3000 updating, 75% of terminals; QUERY/3000, 12% of terminals; MPE commands, 3% of the terminals. Series 44 configurations used dual master disc drives attached to the same General I/O Channel (GIC) at 1 Mb and 2 Mb.

Note. All tests were conducted using 2400 bps terminals. Data entry with V/3000 was done in block mode. Performance results shown here apply to the test application mix only, which is intended to be indicative of a general-purpose EDP environment. Transaction throughput and response times on HP 3000 systems will depend upon your specific configuration and application.

The indicated throughput gains were derived using the application workloads descrived in this guide only. HEWLETT–PACKARD DOES NOT WARRANT OR REPRESENT THAT THE INDICATED THROUGHPUT GAINS WILL BE ACHIEVED FOR ANY PARTICULAR CUSTOMER APPLICATION.

HEWLETT PACKARD

Printed in U.S.A. 3/81

SYSTEM SUMMARY OBJECTIVES

- SE 335 Course Overview
- Tie All Subjects Together
- Aid in System Troubleshooting

LENGTH : 1/2 Day

SUMMARY WILL ENCOMPASS THE FOLLOWING:

- System Startup
- Logon
- Dispatcher
- Memory Manager
- Logoff

SYSTEM SUMMARY SCENARIO

I. Bootstrap System

II. Logon to OPERATOR.SYS

III. enter :SHOWJOB

IV. enter :RUN MYPROG

   A. Read from buffer - record not there.

   B. Program pcals internal procedure - utilize STT.

   C. Program pcals an absent segment.

V. enter :BYE

I. Bootstrap System

<INITIAL'S STACK>

INITIAL - System Process that boots system.

-INITIAL'S environment set up by microcode.

-Speed sense console; automatic for SII/III/40/44/64
; hit carriage return for S30/33

-If RELOAD; Disc Cold Load Info Table (on ldev 1) built to point to directory, virtual memory, MPE, free space table - loads from tape.

-If not RELOAD; uses existing Disc Cold Load Info Table to find above info.

-Initializes MPE tables in memory

-Create system processes, some awakened and put into junk wait - others not awakened until PROGEN, ie., DEVREC, PFAIL, etc.

-Awakens PROGEN - puts on DISPQ

-ASMB(Disp)

PROGEN EXECUTES <PROGEN'S STACK>

-Requests date and time

-Initializes system clock

-Awakens system processes not already awake

-System now up

-PROGEN junk waited

I. Logon to OPERATOR.SYS

-walk up to unowned terminal

-Hit carriage return on terminal, terminal interrupts

-TIP executes; handles interrupt <ICS>

-Wake up SYSIOPROC

SYSIOPROC EXECUTES <SYSIOPROC'S STACK>

-Pcals terminal monitor; process interrupt

-Wake up DEVREC

DEVREC EXECUTES <DEVREC'S STACK>

-Output prompt

-Read JOB, HELLO, DATA command

-Build JMAT entry; state=introduced

-Check passwords

-Wake up UCOP

-DEVREC waits

UCOP EXECUTES <UCOP'S STACK>

-Get PCB, STACK, JIT, JDT, JPCNT

-Allocate $STDIN/LIST

-Build PCBX, MORGUE stack markers

-Wake up Command Interpreter (CI)

CI EXECUTES <CI'S STACK>

-Calls procedure INITJSMP (module NURSERY) to finish CI's initialization

-Issues prompt

-Reads command

III. Enter :SHOWJOB

As each character is read an interrupt is generated and TIP is activated. Tip places the character into a TBUF; the carriage return (c/r) causes the command string to be passed from the TBUF to CI's stack.

-Command passed to command image in CI's stack

-CI parses command and uses a routine to hash into the MPE command dictionary.

-CI finds the SHOWJOB command and the corresponding PLABEL for the command executor

-Executor code examines JMAT entry

-Outputs info to $STDLIST

-Exits back to CI code

-CI outputs prompt

-CI reads command

V. Enter :RUN MYPROG

-CI parses RUN command

-CI creates MYPROG process, loads it and awakens it

-CI junk waits

MYPROG EXECUTES <MYPROG'S STACK>

Assume MYPROG has:
- o $STDIN/$STDLIST open
- o 2 LACB's in PXFILE area
- o PACB (buffered) in XDS

-MYPROG Fopens existing disc file (old file) called MYFILE as first opener

-Creates PACB in XDS, FCB placed in shared FCB XDS

A. PROGRAM EXECUTION

FREAD one record (on first read the whole buffer is filled)

```
   ATTACHIO  gets DRQE

      AWAKEIO  chooses monitor

         SIODM  chooses driver

            DRIVER-INITIATOR  starts physical I/O

         SIODM

            DRIVER-INITIATOR  starts another physical I/O
                              if there is one on that device
                              (Ripple effect)
         SIODM


      AWAKEIO

   ATTACHIO

      WAIT  BIO bit set on in MYPROG's PCB entry.
            ASMB(Disp) is then issued.  A 6-word stack
            marker is placed on MYPROG's stack.
```

Microcode now goes to Dispatcher's stack marker on ICS and begins execution.

```
DISP EXECUTES     <ICS>

      SAVESTATE   places MYPROG's S, Z, Q register values in
                  PXFIXED

      RESCHEDULE  MYPROG's priority = 154.
                  Check CPU time since last priority changed.
                  Finds CPU time exeeded AST.
                  Add 2 to priority.
                  Places process in correct spot on DISPQ.

DISPATCHER

      -Chooses next process on DISPQ (Process B)

      -Puts its register values from PXFIXED onto ICS

      -IXIT

      -Process B is now launched

PROCESS B EXECUTES     <PROCESS B'S STACK>

      -HOWEVER, MYPROG's I/O now completes

      -External interrupt occurs

      -6-word stack marker placed on Process B's stack, save
       process' S, Z, Q on ICS

         <ICS>

      GIP
         AWAKEIO
            SIODM
               DRIVER-COMPLETOR
            SIODM
               AWAKE   checks priority of MYPROG versus Process B.
                       MYPROG more urgent so Awake issues
                       ASMB(Disp) which doesn't take effect yet
                       because currently executing on ICS.
            SIODM
         AWAKEIO
      GIP   issues IXIT - the pending DISP request will now take
            effect.
```

DISP <ICS>

SAVESTATE

RESCHEDULE

Dispatcher now looks at head of DISPQ where we find MYPROG.

-Places MYPROG's S, Q, Z registers on ICS and IXITS

-6-word stack marker is popped from stack

-Set up WAIT code environment

MYPROG EXECUTES <MYPROG'S STACK>

WAIT

ATTACHIO returns DRQE

FREAD scans buffers for desired record and transfers data to stack.

B. MYPROG makes internal Pcal, utilizing STT.

C. MYPROG makes external Pcal to segment #166

-CST 166 absent

-Internal interrupt occurs

-Microcode checks "A" bit in CST entry; it is on

-Traps to ININ to service

ININ internal interrupt handler

QUEUEONSEG

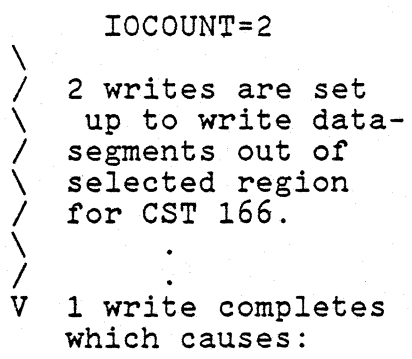
ADDTOLOCALITY add CST 166 to SLL

WAIT 6-word stack marker placed on stack. Set MEM, SAR, SW bits on in PCB entry. Issue ASMB(Disp)

```
DISP   <ICS>

   SAVESTATE

   RESCHEDULE  decrease MYPROG's priority by one, 154 to 155,
               because MEM waited.

     -Dispatcher looks at head of DISPQ, still MYPROG

     -Finds SAR, MEM bits set

     SWAPIN  chooses SLL entry for CST 166 to swapin.

        FETCHSEGMENT  checks maxavail region cell, no avail
                      region large enough

           MAKEROOM  scan pointer looks for segment not
                     referenced recently.

              MAKEOC  found a segment to make into OC -
                      background write set up (1 write).

                 RELEASEREGION  combine and fix up new area just
                                found, combining with another
                                previously made avail region (1
                                write more)
              MAKEOC

           MAKEROOM

      ***  THERE ARE NOW 2 WRITES TOTAL TO CLEAN OUT AREA  ***
      ***                IOCOUNT = 2                      ***

        FETCHSEGMENT  now an area large enough?  YES

           RESERVEREGION

              CLEANREGION  force the 2 writes

           RESERVEREGION

        FETCHSEGMENT  build segment read request - it is unqueued
                      at this time.
     SWAPIN

DISP

     -Dispatcher picks next process which is Process B

     -Puts S, Q, Z registers from PXFIXED on ICS, issues IXIT
```

PROCESS B EXECUTES <PROCESS B'S STACK>

Process B runs -- in the meantime in hardware land --



-External interrupt

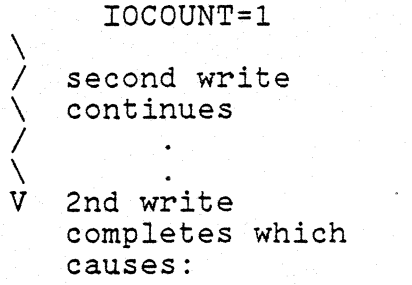
WAIT 6-word stack marker placed on Process B's stack.
Places S, Q, Z on ICS.

<ICS>

```
GIP
   AWAKEIO
      SIODM
         SEGWRITECOMPLETOR  release DRQE - decrement
                            IOCOUNT=1
      SIODM
   AWAKEIO
GIP  issues IXIT
```

PROCESS B CONTINUES EXECUTION <PROCESS B'S STACK>

-- In the meantime in hardware land --



-External interrupt

WAIT 6-word stack marker placed on Process B's stack.
S, Q, Z placed on ICS.
<ICS>

```
GIP
   AWAKEIO
      SIODM
         SEGWRITECOMPLETOR  iocount=0.
            PROCESSINITMSG  release unneeded pages.
               QUEUEDISQUEQ  queue CST 166 read now.
            PROCESSINITMSG
         SEGWRITECOMPLETOR
      SIODM
   AWAKEIO
GIP  issues IXIT

PROCESS B CONTINUES EXECUTION  <PROCESS B'S STACK>

                -- In the meantime in hardware land --

                                                \
                                                /  read of CST 166
                                                \  takes place
                                                /       .
                                                \       .
                                                V  and completes
                                                   causing :
-External interrupt

WAIT  6-word stack marker placed on Process B's stack.

                    <ICS>
GIP
   AWAKEIO
      SIODM
         SEGREADCOMPLETOR  release DRQE.
            PROCESSCOMPMSG  flag region as assigned.
               UNDEFERSEGSIMPQ  undefer any processes
                                waiting for CST 166.
                  AWAKE  want to awaken MYPROG.  Check its
                         priority.  It is greater than
                         Process B's, therefore awake issues
                         ASMB(Disp) which sets Dispatcher
                         pending flag on ICS - cannot
                         execute DISP at this time.
               UNDEFERSEGSIMPQ
            PROCESSCOMPMSG
         SEGREADCOMPLETOR
      SIODM
   AWAKEIO
GIP  issues IXIT - DISP pending bit set, DISP occurs now.
                    <ICS>
```

```
      DISP
         SAVESTATE
         RESCHEDULE

      -Dispatcher chooses from top of DISPQ

      -MYPROG is chosen

      -Sets up S, Q, Z values from PXFIXED to ICS

      -issues IXIT

MYPROG EXECUTES    <MYPROG'S STACK>

        .
        .
        .
 MYPROG ENDS



CI EXECUTES    <CI'S STACK>

      -Returns MYPROG's PCB, stack entries

      -Issue prompt

      -Read command
```

V. Enter :BYE

-Turn off CI's alive bit in PCB

-Close all files

-Must kill off all sons before CI can terminate

-Set mourning wait bit in PCB

-Set JMAT entry to terminating

-Delete all TEMP files

-Print time messages to $STDLIST

-Update directory entries

-Close $STDIN/LIST

-Release JCUT/JMAT entries

-Make a UCOP request entry

-Wake up UCOP

UCOP EXECUTES <UCOP'S STACK>

-Chooses next entry on UCOP's queue

-Detects terminating PCB

-Returns XDS, Stack entries

END OF SUMMARY

APPENDIX

HOMEWORK & reading MATERIAL

MPE MODULE

HOMEWORK

1. READ SLIDES
2. ANSWER QUESTIONS

HEWLETT hp PACKARD

INITIAL (ØØ)

Resides in PUB.SYS.
Only used at SYSTEM STARTUP.

When the tape is loaded, an executable copy is made on disc to be loaded into memory.
19500 Lines of code.
117744 CPU instructions.
15 Segments.

SYSDUMP (Ø1)

* O.P. (System Supervisor) capability required.
* Not a system process.
* Used to create various versions of the system on Mag Tape.
* :SYSDUMP $Null can be used to interrogate the system on-line.

  ** WARNING -- Answering "Y" to "System SL changes?" will lock up the system by locking up the SL. **

  ** WARNING -- System Directory lock-up will occur for short period of time giving impression system is hung. **

HEWLETT PACKARD

(32050) SEGDVR (03)

Create SEGPROC as a son process.
User interface to SEGMENTER.
Checks the command.
Sends MAIL to SEGPROC to perform
the command.
Need to allocate SEGDVR to help
allocate SEGMENTER.

HEWLETT PACKARD

(32050) SEGPROC (02)

Runs as a separate process.
Has the code to actually execute
the SEGMENTER command.
Is a son to your process via SEGDVR.
Need to allocate SEGPROC to help
allocate SEGMENTER.

HEWLETT hp PACKARD

(32050) SEGUTIL (71)

Segment lives in SYS SL.

Performs numerous utility functions for the SEGMENTER.

LOAD (Ø5)

Loads program files to get them ready to run.

Resolves all externals of the program and group libraries.

Outputs the LMAP.

Runs as a System Process.

USER CONTROLLER PROGRAM (UCOP) (Ø7)

Runs as a system process.
Information passed thru the UCOP Request Queue.
Builds the UMAIN (Command Interpreter) process data stack.
Makes the UMAIN entry in the PCB.
Allocates $STDIN & $STDLIST devices.
FILLS IN entry in Job Master Table (JMAT).
Gets Job Process Count Table (JPCNT) entry & fills it in.
Activates the CI process for the new JOB/SESSION.
Dispatcher runs C. I.

DEVICE RECOGNITION (Ø8) (DEVREC)

Handles Device recognition when an unexpected interrupt occurs.

DEVREC is then awakened by the device driver.

Reads the HELLO, JOB or DATA command into a System Buffer & checks for spelling.

UCOP then handles setting up the JOB/SESSION.

Calls Auto Volume Recognition (AVREC) upon tape ready int.

Awakes PVPROC upon PV or SDISC UPON ONLINE interrupt.

Runs as a System Process.

PROGEN (Ø9)

It is a system process.
It is called the father of MPE
It is always PCB 1.

It is created and first run by
INITIAL.
Handles all console operator
commands executable after
Control 'A'.

INTERNAL INTERRUPT HANDLER (ININ) (10)

Always Code Segment 1.

The microcode must be able to map to the different interrupt handler routines:

- An internal interrupt is an error condition detected by CPU hardware.
- A trap is an error condition detected by microcode.

The outer block becomes the STT.

If Power Fail finishes then Halt 7.

EXT. PROG. LABEL (%)	STT NO. (%)	INTERRUPT TYPE	PARAMETER*	EXECUTING STACK**
100401	1	Bounds Violation		
101001	2	Illegal Memory Address		
101401	3	Non-Responding Module		
102001	4	System Parity Error		ICS
102401	5	Address Parity Error		ICS
103001	6	Data Parity Error		ICS
103401	7	Module Interrupt	Module No.	ICS
104001	10	(Unused)		
104401	11	Power Fail		ICS
105001	12	(Unused)		
105401	13	(Unused)		
106001	14	(Unused)		
106401	15	(Unused)		
107001	16	(Unused)		
107401	17	(Unused)		
110001	20	Unimplemented Instruction		
110401	21	STT Violation		
111001	22	CST Violation		

ININ (CONT)

EXT. PROG. LABEL (%)	STT NO. (%)	INTERRUPT TYPE	PARAMETER*	EXECUTING STACK**
111401	23	DST Violation		
112001	24	Stack Underflow		
112401	25	Privileged Mode Violation		
113001	26	(Unused)		
113401	27	(Unused)		
114001	30	Stack Overflow		ICS
114401	31	User Traps		
		a. Integer Overflow	%000001	
		b. Floating-Point Over.	%000002	
		c. Floating-Point Under.	%000003	
		d. Integer Divide by 0	%000004	
		e. Floating-Point Divide by 0	%000005	
		f. Ext. Prec. Floating-Point Overflow	%000010	
		g. Ext. Prec. Floating-Point Underflow	%000011	
		h. Ext. Prec. Floating-Point Divide by 0	%000012	
		i. Decimal Overflow	%000013	
		j. Invalid ASCII digit	%000014	
		k. Invalid Dec. digit	%000015	
		l. Invalid Source Word Count	%000016	
		m. Result Word Count Overflow	%000017	
		n. Decimal Divide by 0	%000020	
115001	32	(Unused)		
115401	33	(Unused)		
116001	34	(Unused)		
116401	35	(Unused)		
117001	36	(Unused)		
117401	37	Absent Code Segment		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120001	40	Trace		
		a. On PCAL	P-Label	
		b. On EXIT	N	
		c. On IXIT	0	
120401	41	STT Entry Uncallable	P-Label	
121001	42	Absent Data Segment	DST No.	
121401	43	Power On		ICS
122001	44	Cold Load		ICS
		a. System I/O (SIO)	0	
		b. Direct I/O (DIO)	Label	

*Unless noted, the parameter is the External Program Label.
**Unless noted, interrupts are serviced on the User Stack.

All User Traps (STT No. %31) are enabled by the User Traps bit in the Status register.

MEMLOGP (11)

This is a system process.
Created by INITIAL.
Activated by PROGEN.
  *PROGEN waits for MEMLOGP to open the data file.
  *MEMLOGP then awakens PROGEN.
If =SHUTDOWN then MEMLOGP is awakened by PROGEN and closes the data file.
MEMLOGP then awakens PROGEN.
Most of the time MEMLOGP is suspended waiting for a timer request to time out.
This is the large entry usually found in the TRL of system dumps.

SYSTEM LOGGING PROCESS

(LOG) (12)

It is a system process.

It is created by INITIAL.

Initially awakened by PROGEN to open the LOG file.

Puts itself to sleep until some other system process wants to have an event logged.

C. I. executing : SWITCHLOG causes opening a new LOG file.

I/O DRIVERS (13-27)

DRIVERS DO NOT RUN AS SYSTEM PROCESSES.
THEY ARE CALLED USING THE PLABEL FROM THE DLT ENTRY FOR THAT DRIVER.

THEY GENERALLY HAVE TWO MAIN SECTIONS:

*INITIATOR - STARTS THE I/O BY CODING UP THE CHANNEL INSTRUCTIONS AND STARTING THEM TO EXECUTE.

* COMPLETOR - CALLED BY THE INTERRUPT HANDLER AFTER RECOGNIZING THE INTERRUPT CHANNEL PROGRAM.

*NOTE - ALL MODULES ARE UNIQUE TO SERIES II/III EXCEPT 26, WHICH IS UNIQUE TO SERIES 33.

PFAIL (30)

Runs as a system process.
It is not memory resident.
Created by INITIAL.
INITIAL sets JUNKWAIT in PCB.
POWERON of ININ:

*Initiates:

#The data registers of the CPU.
#The system console (send **POWERFAIL** message).
#Restart all disc I/O.
Restart the system clock.

*Awakens PFAIL process.

PFAIL:

*Call the Initialization section of the driver for each device using the Initialization PLABEL from DLT(7). This happens for any physical device controller except Mag Tape because the tape unloads on PF.
*Abort the I/O on any non-disc SIO device
*Write PF LOG record.
*Send "**POWERFAIL**" to all logged on terminals.
*Allow unstarted I/O requests to complete.
*Set SYSTEMUP.
*JUNKWAIT the PFAIL process.

PVPROC (31)

Runs as a system process.
Appears in a dump as "SYST".
Contains the disc autorecognition code.
Initially awakened by PROGEN which
  records mounted non-system volumes
  then awakes PROGEN.
Reawoken by DEVREC when a disc is
  is physically mounted or dismounted.
Calls SDISC code if a serial disc
  is mounted instead of a PV.
Modifies the Mounted Volume Set
  Table to reflect current on-line
  discs.

VINIT (32)

Contains 4 Segments:

*VINITCI - Handles command parsing for the VINIT Subsystem.
- Contains a number of general purpose procedures needed by the subsystem.
- Contains the routines for the COPY & SCRATCH commands.

*NEWPACK - Contains the code needed to implement the INIT and DTRACK commands.
- Contains the code for Defective Track Table (DTT) handling

*PVSTATUS - Handles most of the listing commands.

*CONDENSE - Handles the COND command.

The message "**OUTPUT FILE OPEN ERROR**" means FOPEN to $STDLIST failed.

Executed by typing :VINIT. but kept as PVINIT.PUB.SYS

HP IB DRIVERS (35 - 39)

* ARE ALL HP IB INTERFACES FOR SERIES II/III AND HP IB DEVICES.

MAKECAT (40)

Runs as a utility program to install the Message & Help catalogs.

Also invoked by SYSDUMP for "MESSAGE CATALOG CHANGES?".

Contains 3 segments:

- *PRIV - Has four entry points:
  - #MAIN - Specify run with no entry point.
    - Creates a user message catalog.
  - #BUILD - Creates and installs a new system message catalog.
  - #DIR - Installs an old system message catalog.
  - #HELP - Creates and installs a new HELP catalog.
- *USER - Contains general procedures for MAKECAT.
- *PRIV1 - Contains one procedure for error handling.

FILEACC (50)

CONTAINS CODE FOR THE FILE SYSTEM INTRINSICS.

19,641 LINES OF CODE

TOTAL CODE = %44620

COMMAND INTERPRETER

COMM'INIT (51)

*Contains the code needed to execute most user commands.
*Contains the code executed by all UMAIN processes.
*This module currently adds %102550 (31384 decimal) instructions.
*Adds 13 segments to the system SL:

#CIINIT - Command Interrogator. This is the block of code that executes as the UMAIN code segment.
#CIERR - Command executors for error handling and misc. CI routines such as WELCOME message handling.
#CIALTORG - Executes organizational management commands [:LIST--(except :LISTVS) :ALT--, :RESETACCT, :NEWVSET].

COMM' INIT (CONT)

#CIORGMAN - Executes organizational management commands [:NEW-- (except :NEWVSET), :PURGE--]

#CIMISC - Executors for :COMMENT, :SHOWJCW, :SETJCW, :IF, :ELSE, :ENDIF, :DSTAT, :VSUSER, :MOUNT, :DISMOUNT.

#CIUSERUTIL - Executors for :DEBUG, :HELP :TELL, :TELLOP, :WARN, :FREERIN, :GETRIN :SHOWTIME, :CONTINUE, :RESUME, :ABORT, :SPEED, :SHOWME, :PTAPE.

#CISUBS - Subsystem executors.

#CIPREPRUN - Executors for :PREP and :RUN as well as :SETDUMP, :RESETDUMP, :SETMSG, :MRJE, :APL.

#CISYSMGR - Executes system supervisor (for the most part) commands [:SHOWQ, :REPORT, :JOBPRI, :SHOWLOG, :SWITCHLOG, :RESUMELOG, :QUANTUM, :ALLOCATE, :DEALLOCATE, :SYSDUMP].

COMM'INT (CONT)

#CIFILEM - CI File Manager. Executors for :SECURE, :RELEASE, :SAVE, :PURGE :RENAME, :RESET, :CRESET.

#CICOMSYS - CI Communications Systems. Executor for :CLINE, :DSLINE, :RFA, :REMOTE commands.

STORE/RESTORE (52)

This is the system code for
:STORE and :RESTORE.
Called by CI upon recognition
of these commands.
Results in 3 segments in the SL:
#CXSTOREST - Main code for STORE.
#STORE - Procedures for store.
#RESTORE - Main code for RESTORE.
Can run the P-file which calls STORE
followed by a RESTORE.

(DIRC) DIRECTORY (53)

* Adds one segment to the SL.
* Handles Directory Management and all directory search routines.
* Contains the code to bind groups to private volumes.
* External callable procedures:

#DIRECLOGON - Called by NURSERY only.
#DIRECLOGOFF - Called by MORGUE only.
#DIRECSCAN
#DIRECADJUST
#DIRECPURGE
#DIRECPURGEFILE
#DIRECINSERTFILE
#DIRECFINDFILE
#DIRECFIND
#DIRECINSERT
#ACCCHECK
#DIRECBIND - Called by PVSYSM only.
#DIRECUNBIND - Called by PVSYSD only.

ALLOCATE (54)

*Handles device allocation/deallocation.
*Adds 2 segments to the SL.

ALLOCATE Segment:

- *ALLOCATE - Real and virtual device allocation for non-disc devices.
- *FORMSALIGN - Writes out the forms alignment header to spooled device aligned.
  and asks operator if they are
- *ASKOP - Issues I/O requests to console and checks responses.
- *PRIMEDEVICE - Does ATTACHIO setup to allocate the device (FILEOPEN), prints header and calls FORMSALIGN if forms are specified.

ALLOCATE (CONT)

ALLOCUTIL Segment:

*SROOSTER - Called by PROGEN and spooler to waken output spooling processes.

*ALLOCENTRY - Expands the Extended Device Directory (XDD).

*DEALLOCENTRY - Returns space from the XDD.

*SLINKXDD - Links XDD entries.

*DELINKENTRY - Delinks JMAT/XDD entries.

*SRELINKODD - Relinks entries in Output Device Directory (ODD).

*SPUTXDD - Puts entries in XDD.

*SREMOVEXDD - Removes entries in XDD.

*SFINDODD - Finds ODD entries by device file ID.

*SFINDIDD - Finds IDD entries by device file ID.

ALLOCATE (CONT)

ALLOCUTIL (CONT):

*XDOSPOOLINFO - Updates XDO entries for spooled devices.

*SPOOLEDDEV - Checks if a device is spooled.

*DISKDEALLOC - Deallocates disc devices and space in virtual memory.

*DSPSEUDOTERM - Determins if a given LDEV is a DS Pseudo Terminal.

*DISCALLOC - Allocates disc devices and space in virtual memory.

*FREEDEVICE - If terminal or spooled then device close the device.

*DOUBLETIME - Returns current date/time in double word format of 1/0: 7/year; 9/day; 5/hour; 6/minute; 4/quad sec.

*DEALLOCATE - Logically releases the device from the user.

ALLOCATE (CONT)

ALLOCUTIL (CONT):

*GETCLASS - Returns all of the LDEV numbers listed for a given class name.

*PUTDEV/GETDEV -. Puts/gets an LPDT/LDT entry.

*GETDEVINFO -

#If class name passed then return:

=>Index into Device Class Table (DCT).

=>Device Type.

=>LPDT entry of first device in that class.

#If LDEV passed then return:

=>LDEV number.

=>Device Type.

=>LPDT entry (2 words).

=>LDT entry (5 words).

ALLOCATE (CONT)

ALLOCUTIL (CONT):

*DISKSPACE - Performs the manipulation necessary to the Disc Free Space table to get or release disc space.

*XDISKSPACE - Gets NSECT worth of disc space at the address specified on the disc specified. This procedure calls DISKSPACE to actually get the space.

HARDRES (55)

*Adds one segment to the SL.
*It is one of the 2 memory resident SL segs.
*5970 Lines if code (1918 Version).
*%16454 (7468 Decimal) Instructions.
*Contains routines to handle System
  Clock board (DRT 3):
  =>Interrupt Handler (TICK).
  =>HELP Terminal Handler.
Async Terminal Interrupt Handlers:
  =>Terminal Interrupt Processor (TIP).
  =>Full-Duplex Modem Int. Handler (DSET1).
  =>Half-Duplex Modem Int. Handler (DSET2).
Paper Tape Reader Int. Processor (PTRIP).
*General SIO Int. Processor (GIP).
*SIO Device Monitor (SIODM).
*SYSIOPROC - Becomes the system process SYSIO.
*PROCEDURE ATTACHIO.
*PROCEDURE SUDDENDEATH.
*Procedures IOIMPEDE/IOUNIMPEDE.
*Procedures to handle 'Break' from
  terminals.
*Procedures to write priority mes-
  sages to the system console.
*Procedures to get/return TBUF's, SBUF's, IOQ's, DRQE's.
*Procedures to implement I/O
  Resource Queuing.

ABORTDUMP (58)

*Adds one segment to the SL.
*Contains code for user error handling with the procedures:
=>ABORT - Handles all program aborts and formats/prints the abort message.
=>ININRETURN - Handles return to ININ from FIRMWARESIM routines.
=>INTRINSIC QUIT
=>INTRINSIC QUITPROG
=>INTRINSIC ARITRAP
=>INTRINSIC XARITRAP
=>INTRINSIC XSYSTRAP
=>INTRINSIC XCONTRAP
=>INTRINSIC RESETCONTROL
=>INTRINSIC STACKDUMP
=>PROCEDURE DEC'SIM'TRAP - Simulates traps for decimal firmwaresim routines.
=>PROCEDURE DEC'SIM'TRAP' - Interfaces traps from DVID, MPYD & EDIT with the regular trap mechanism.

MESSAGE (59)

*Adds one segment to the SL.
*Contains Reply Information Table
(RIT) handling procedures.
*Contains the procedures:
=>INITMSG:
#Called at system startup by PROGEN.
#Copies message directory from user
labels of CATALOG.PUB.SYS into
a data segment.
#Puts disc address of CATALOG.PUB.SYS
into SYSGLOB %371/372.
#Puts message directory DST#
into SYSGLOB%373.
=>GENMSG:
#Called by all of MPE to
generate messages.
#Can direct the message to $STDLIST,
console (to PROGEN via RIT), or file.
#Can insert up to 5 parms/msg.
=>FINDMSG:
#Fetches one line of message from
the file CATALOG.PUB.SYS.
#If message is not found then does
a binary search.
=>FORMSG:
*Prep message for routing.

PROCSEG (60)

*Adds one segment to the SL.
*Contains routines that control and get information about processes.
*Contains rountines to implement the intrinsics:

=>GETORIGIN
=>GETPROCINFO
=>SUSPEND
=>FATHER
=>CALENDAR
=>CAUSEBREAK
=>ACTIVATE
=>CLOCK
=>PAUSE

*Contains the procedures:

=>ABORTPROG - Used for :ABORT or break followed by :RUN or sub-system call or :BYE.
=>BREAKSS - Entry point to BREAKJOB used for handling Control "Y".
=>SHOWPROC - Gets process information for :SHOWQ.

PROCSEG (60) (CONT)

*Contains all of the routines needed
to handle process-to-process
communications within the
same CPU.

*This is called the MAIL system and
includes the intrinsics:
=>MAIL - Return status of mail.
=>SENDMAIL
=>RECEIVEMAIL

*Contains the MAil utility procedures:
=>ABORTMAIL - Called from EXPIRE.
=>ABORTMAILINFO - Clear mail & return
mail box XDS.
=>SENDMAILINFO - Set mail info in
P-P Communication table.
=>GETMAILINFO - Get mail info from
P-P Communications Table.
=>SETMAILSTATUS - Set mail status
in PCBN(9).(3:2).
=>GETMAILSTATUS - Gets mail status
in PCBN(9).(3:2).
=>CHECKMAILPCB - Checks validity
of PIN passed to mail intrinsic.

NON-RESIDENT I/O (62) (NRIO)

*Adds one segment to the SL.
*Contains miscellaneous swapable I/O routines:
  =>IOCONTROL - Calls ATTACHIO to generate a blocked control request.
  =>DEVICESTATUS - Returns the device hardware interrupt status from the DIT for devices other than terminals. If Programable Controller then also put the status in the DIT.
  =>RESETBREAKBITS - Resets break and Control "Y" bits in the LPDT.
  =>SETTERMTYPE - Checks for valid term type and checks that the speeds are valid for the type specified (FCONTROL (38) & HELLO--;TERM=).
  =>INITCHANNEL - Initializes the TDI/TCI boards (except Diag. channel).
  =>TERMINIT - Called by PROGEN to initialize Diag channels on TDI then calls INITCHANNEL.
  =>GETSYSBUF - gets a System Buffer.
  =>ABORTPROCIO - Aborts all non-disc I/O requests related to a given process.

NRIO (CONT)

=>ABORTIO - Aborts I/O requests on all devices except Disc, ATC, SSLC OR HSI. (NOTE: Disc requests are never aborted. HSI & SSLC are aborted by CS.)

=>ABORTIOX - Aborts I/O requests on all devices except Disc & CS. Called by FS upon certain errors.

=>IOMESSPROC - IOMS system process code-Interfaces between LOGERROR and LOG process.

=>INITIO - Called by PROGEN to initiate all I/O devices. Initialization is accomplished from the highest LDEVN to lowest.

=>VALIDDEVTYPE - Checks for LDEV & function code validity for ATTACHIO calls.

=>IOTABLEINFO - Interfaces "MEASIO" to Series III & Series 33 I/O tables.

PROCESS CREATION (63)

PCREATE

*Adds one segment to the SL.
*Contains routines to create processes:
  =>SUBQUEUE - Returns characters of the specified subqueue, i.e., AS, BS, CS, DS, ES.
  =>GETPRIORITY - Gets a new priority for a specified process which is the caller or the caller's son.
  =>PROCREATE - Sets up a process given one instruction and a data segment.
  =>CHECKPRIORITY - Checks the validity of the priority class specified for the specified process. Classes are:
    A - Absolute priority number.
    M - Main queue (AS, BS).
    S - Subqueue (CS, DS, ES).
  =>CREATE - Called to set up son processes. Handles the :RUN command parameters then calls PROCREATE. (INTRINSIC).

PCREATE (CONT)

=>LOG - Checks if logging is required for the specified type of record, formats the LOG record, outputs the record to the log buffer and awakens the LOG process if the buffer is full.

MORGUE (64)

*Adds one segment to the SL.
*Contains routines to clean up
and get rid of processes.
=>CLEANUPFILES - Cleans up all files.
=>CLEANUPVOLUMES - Disassociates the process from any mounted volume sets.
=>ABORTRIN - Unlocks all RIN's that a process may have left locked when terminating.
=>GETPROCID(Y) - Returns the PIN of the Yth son of the caller (INTRINSIC).
=>BURRYPROC - Deletes a son process from the system when the father is being terminated.
&Remove it from the scheduling queue.
&Remove it from process tables (PCB, DST, XCST, etc.).
&Activates its father if required.
=>KILL - Sets KILL bit in PCB and calls BURRYPROC. Aborts any active I/O related to the process. (INTRINSIC).
=>CLEANUPJOB - Called by EXPIRE when the UMAIN process is terminating. Writes logoff message, gets rid of all job table entries and logs the event.

MORGUE (CONT)

=>EXPIRE - Called from TERMINATE for any process terminaton. If UMAIN then call CLEANUPJOB else awake father. Calls BURRYPROC to kill active sons if UMAIN.

=>TERMINATE/TERMINATE' - Does some cleanup then calls EXPIRE.
&TERMINATE is an INTRINSIC.
&TERMINATE' is an entry point used by SPL. The SPL construct "END." generates a PCAL to TERMINATE'.

BIPCS 65

* PROCEDURES THAT PROVIDE WAIT/WAKEUP MECHANISM FOR INTERPROCESS COMMUNICATION

* CREATION AND DELETION OF PORT DATA SEGMENTS AND MESSAGE QUEUE ENTRIES

* TIMEOUT PROCEDURES

* MAINTAINS BUFFERING

CACS 66

* HEART OF MESSAGE FILE IMPLEMENTATION

* PROCEDURES TO IMPLEMENT MESSAGE FILE I/O

CHECKER (69)

*Adds one segment to the SL.
*Contains routines to handle error enabling and checking on what the user is doing including calls to intrinsics by the user.
*Contains code for the intrinsics:
  =>PROCTIME
  =>GETPRIVMODE
  =>GETUSERMODE
*Contains some IMAGE internal queue procedures.

UTILITY(70)

*Adds two segments to the SL.
*Contains code for miscellaneous utility intrinsics:

@ UTILITY1

=>BINARY	=>SEARCH
=>ASCII	=>WHO
=>DASCII	=>MYCOMMAND
=>PRINT	=>READX
=>PRINTOP	=>READ
=>DBINARY	=>PRINTOPREPLY

@ UTILITY2

=>FCARD
=>PTAPE
=>CTRANSLATE (ASCII/EBCDIC conversion)

SEGUTIL (71)
SEGMENTER UTILITY

*Adds one segment to the SL.
*Contains the code initially called by C.I. when :SEGMENTER or :PREP commands are issued.
*Contains the code for the intrinsic CLEANUSL.

LOADER1 (72)

*Adds one segment to the SL.

Contains routines needed to get a program loaded and ready to execute.

*Includes the code for the intrinsics:

=>LOADPROC

=>UNLOADPROC

RINS (73)

*Adds one segment to the SL.
*Contains routines needed to
  manage local and global RINS.
*Contains the code for the intrinsics:
  =>LOCKGLORIN
  =>UNLOCKGLORIN
  =>FREELOCRIN
  =>LOCKLOCRIN
  =>GETLOCRIN
  =>UNLOCKLOCRIN

JOBTABLE (74)

*Adds one segment to the SL.
*Contains routines to handle the Job Master Table (JMAT), Job Information Table (JIT) and Job Directory Table (JDT).
*Contains the routines to link #S & #J information to the correct devices for all user and console operator commands using this format.
*Handles the intrinsics:
  =>GETJCW
  =>PUTJCW
  =>FINDJCW
  =>SETJCW

DEBUG 75

* CREATION OF BREAKPOINT TABLE

* ALL PROCEDURES NEEDED TO IMPLEMENT DEBUG EXCLUDING DECOMP FACILITY

NURSERY (76)

*Adds one segment to the SL.
*Contains the routines needed to give birth to processes (INITJSMP, etc.).
=>Completes JMAT entry.
=>Initializes the JCUT entry.
=>Initializes the JIT entry.
=>Checks the logon message against the system directory including passwords if furnished in the logon.
=>Completes IDD & ODD entries.
=>Schedules the job for DISPATCHER.
=>Open the $STDIN & $STDLIST devices.
=>Prints the logon message to $STDLIST and console.
=>Checks the :HELLO command keywords (IN PRI; HIPRI; TERM=; etc.).

FIRMWARESIM (78)

*Adds 3 segments to the SL:
=>FIRMWARESIM1 thru FIRMWARESIM3.

*Contains the routines needed to simulate all of the decimal and double precision arithmetic CPU instructions normally executed on the upper ROM board.

*The procedures are called by ININ after the microcode detects an unimplemented instruction internal interrupt.

*The segments *must* be in the SL or INITIAL will not be able to handle ININ correctly to bring up the system.

SPOOLING (79)

*Adds one segment to the SL.

*Contains miscellaneous routines
for SPOOLING:

=>Handles forms messages for spooling.

=>Handles headers and trailers.

=>Generates spooler LOG entries.

=>Allocates and deallocates devices
for the spooler.

=>Contains the code to read and
write the spooler files.

=>Contains the code that becomes the
spooler system processes (marked
"SPOOL" in the PCB table of dumps).

=>PROCEDURE SPOOLOUT

=>PROCEDURE SPOOLIN

SPOOLCOMS (80)

```
*Adds two segments to the SL.
*Contains routines to handle console
  operator/user commands:
  @ SPOOLCOMS1
     =>:STARTSPOOL
     =>ALTSPOOLFILE
     =>ALTJOB
     =>DELETESPOOLFILE
     =>ABORTJOB ("=" & ":" FORM)
     =>STREAM
  @ SPOOLCOMS2
     =>SHOWIN
     =>SHOWOUT
     =>SHOWDEV
     =>SHOWJOB
```

PVSYS (81)

*Adds 3 segments to the SL.

=>PVCOMSEG

& Contains code for :DSTAT :VSUSER commands.

=>PVSYSM

& Contains code to :MOUNT volume sets.

=>PVSYSD

& Contains code to *DISMOUNT volume sets.

& Contains MVTAB & PVUSER table handlers.

& Contains code to bind user's JIT for logon PV groups.

UDC (82)

* ADDS 1 SEGMENT TO SL.
* CONTAINS ALL CODE FOR USER DEFINED COMMANDS INCLUDING PROCEDURES
  - UDC
  - INITUDC
  - SEARCHUDC
  - FEEDCI
  - ETC.
* SETS UDC DST# IN DB+250 IN UMAIN STACK.
* NO SPECIAL CAPABILITY REQUIRED FOR USE.
* FILE COMMAND. PUB. SYS CONTAINS ENTRY FOR EVERY USER ON SYSTEM WITH ACTIVE UDC'S

USER (83)

*Adds one segment to the SL.
*Handles GENMESSAGE intrinsic for
  user message catalogs.
*To use GENMESSAGE, user must:
  =>Create a user catalog by
    running MAKECAT.PUB.SYS.
  =>FOPEN the catalog with
    FOPTIONS=5, AOPTIONS=%420.
*Also contains code for intrinsics:
  =>FMTDATE
  => FMTCALENDAR
  =>FMTCLOCK
  =>DATELINE

HELPUSER (84)

*Adds one segment to the SL.

*For SYSDUMP "SL CHANGES?", HELPUSER must be Permanent segment, i.e. HELPUSER,U84U00B,P

*Contains the code to execute the HELP Subsystem.

*searches the file CICAT.PUB.SYS to find the HELP text to print.

*Contains the code to print the caret (^) for error messages.

OPCOMMAND (85)

HANDLES ALL THE OPERATOR COMMANDS AND THE DISTRIBUTED CONSOLE.

IT IS ARRANGED IN THREE SEGMENTS BASED ON FREQUENCY OF USE.

LABSEG (86)

* ADDS 2 SEGMENTS TO SL.
* TOTAL CODE ABOUT 5260 WORDS.
* ENTERED VIA PROCEDURE AVREC FROM DEVREC
  - DEVREC CALLS AVREC WHEN INTERRUPT OCCURS FROM AN UNOWNED, NON-JOB/DATA ACCEPTING DEVICE.
* CONTAINS CODE FOR FOLLOWING:
  - LABEL OR NOLABEL
  - ANSI OR IBM
  - VOLUME ID VERIFICATION
  - EXPIRATION CHECK ON WRITES
  - REEL SWITCHING
  - WRITES TAPE LABELS NOT USER LABELS

SDISC (87)

* ADDS 2 SEGMENTS TO SL.
* GENERALLY SELF-CONTAINED
  - ONLY PROCEDURES CALLED FROM OUTSIDE SDISC ARE:
    * SDISCIO - CALLED BY ATTACHIO
    * FINDSDISCGAP - CALLED BY SYSDUMP FOR CONTIGUOUS BLOCKS.
* SDISCIO IS ATTACHIO COUNTERPART
* PERMITS 2 TYPES OF RECORD FORMAT:
  - DATA RECORD (USER MODE)
  - CONTIGUOUS BLOCK (PRIVILEGE MODE)

MEASIO (88)

CONTAINS ALL THE PROCEDURES NEEDED FOR MONITORING A SYSTEM.

USED MOSTLY FOR PERFORMANCE ANALYSIS

HEWLETT hp PACKARD

LOGSEG1 (90) LOGSEG2 (91)

CONTAINS ALL THE PROCEDURES RELEVANT TO USER LOGGING.

KERNELC 92

*DISPATCHER AND MAM PROCEDURES

*UTILITY PROCEDURES FOR OBTAINING AND RELEASING SYSTEM TABLES

*PROCEDURES FOR MEMORY RESIDENT MESSAGE FACILITY

HEWLETT PACKARD

KERNELD 93

*ALLOCATION/DEALLOCATION OF CST/CSTX ENTRIES

*CREATION/DELETION OF XDS's

*XDS MANAGEMENT

*DATA SEGMENT CONTRACTION/EXPANSION

MISCSEGC 95

*PSEUDOINT- SWITCHES PROCESS TO PROPER CODE

*THISCPU- DETERMINES TYPE OF CPU

*GETDSDEVICE- TELLS IF DEVICE IS A DS DEVICE

*MMSTAT- MONITOR AND COLLECT PERFORMANCE MEASUREMENT INFO

MEASSEG (96)

CONTAINS MOST OF THE CODE FOR THE INTERNAL MEASUREMENT FACILITY:

- STARTSTATISTICS - FOR ENABLING STATISTICS GATHERING, INCLUDING SETTING UP THE MEASUREMENT XDS FOR CLASSES Ø AND I5.
- FINDDEVICES - USES THE LPDT AND LDT TO FIND ALL LDEV #'s (AND THE TOTAL # OF DEVICES) OF A GIVEN DEVICE CLASS. THE CLASS CAN BE DISCS, TERMINALS, LINE PRINTER, OR MAG TAPE.

FINDDEVICES IS USED ONLY BY STARTSTATISTICS

MEASSEG (96)

(CONT'D)

- FORMATSEG - CALLED BY STARTSTATISTICS. USED TO OBTAIN AND FORMAT THE XDS FOR CLASSØ STATISTICS.

- FORMATPROCSEG - CALLED BY STARTSTATISTICS. USED TO OBTAIN AND FORMAT THE XDS FOR CLASS 15 STATISTICS.

- ENABLECLASS - CALLED BY STARTSTATISTICS TO ENABLE STATISTICS GATHERING OF ONE OF FIFTEEN CLASSES. SETS THE APPROPRIATE BITS IN SYSGLOB CELL %262 AND IN PXFIXED (%1ØØ) OF THE CALLERS STACK (CALLED PCLASENABLEMAST).

- GETSTATISTICS - USED TO ACCESS THE MEASUREMENT XDS FOR CLASSES Ø AND 15.

FILEIO 97

CONTAINS ALL FILE SYSTEM CODE NOT IN FILEACC.

%34540 MACHINE INSTRUCTIONS.

19,714 LINES OF CODE.

DEBUG UTIL 98

* DECOMP FACILITY ENABLES THE TRANSALTION OF OCTAL TO MNEMONIC CODE

CHECK FOR UNDERSTANDING

1. WHAT IS AN MPE MODULE?

2. HOW ARE MPE DRIVERS INVOKED?

3. WHICH SYSTEM PROCESS HANDLES ALL UNANTICIPATED SYSTEM INTERRUPTS; ie, WHEN A MAG TAPE DRIVE IS SWITCHED ONLINE.

4. HOW OFTEN DOES THE SYSTEM CLOCK INTERRUPT THE CPU?

5. WHAT MPE CODE GIVES A PROCESS CONTROL OF THE CPU?

6. WHICH MODULE(S) CONTAINS THE CODE THAT HANDLES THE PREPing OF A USL FILE?

7. WHEN A USER GETS A STACK OVERFLOW, WHICH MPE MODULE WILL SERVICE IT?

8. WHICH SYSTEM PROCESS SETS UP THE SESSION ENVIRONMENT WHEN :HELLO IS ENTERED ON THE TERMINAL?

9. WHAT ARE THE TWO MAIN FUNCTIONS OF A PERIPHERAL DRIVER?

10. WHO HANDLES THE LOGICAL ALLOCATION OF A MAG TAPE DEVICE TO A PROCESS REQUESTING I/O TO IT?

11. WHAT ARE INTRINSICS?

12. WHO HANDLES THE :PREP COMMAND?

13. WHAT MPE CODE EXECUTES THE HELP SUBSYSTEM?

14. WHO HANDLES THE MANIPULATION OF THE BREAKPOINT TABLE?

15. WHAT MPE CODE HANDLES BOOTSTRAPPING?

16. NAME THE TWO TYPES OF INTERRUPTS.

17. WHO IS THE FATHER OF MPE AND WHY?

18. WHO GETS RID OF USER PROCESSES?

19. WHO SETS UP USER PROCESSES?

20. WHAT MPE CODE RESOLVES PROGRAM FILE EXTERNAL REFERENCES?

22. WHICH MPE MODULE CONTAINS CODE TO HANDLE THE MANAGEMENT OF MEMORY?

23. WHAT MPE CODE IS INVOKED WHEN THE "BREAK" KEY ON A TERMINAL IS HIT?

24. NAME THE MPE CODE THAT MUST GET "INVOLVED" WHEN A SERIAL DISC IS MOUNTED.

24.1 HOW OFTEN IS THE SYSTEM CHECKED FOR AN ACTIVE PROCESS IF THE SYSTEM IS IDLE?

25. NAME THE TWO TYPES OF I/O.

26. WHAT TYPE IS USED FOR TERMINALS AND WHAT TYPE IS USED FOR DISC ?

27. WHO EXECUTES THE PAUSE INSTRUCTION?

28. WHO INVOKES PROLEN?

29. WHO CHECKS THE HELLO COMMAND KEYWORDS (INPRI, HIPRI, TERM=, etc.)?

30. WHAT MPE CODE DETERMINES WHAT TYPE OF CPU YOU ARE EXECUTING ON?

SERIES 30/33/44/II/III

(Subsystems that are included in FOS)

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
32201A	EDIT /3000	EDITOR	
32212A	FCOPY/3000	FCOPY	
32214B	SORT-MERGE/3000	SORT, MERGE	SORTLIB,U00U214A.HP32214.SUPPORT MERGELIB, " " SMUTILS, " "
32209A	V/3000	FORMSPEC, REFSPEC, REFORMAT, ENTRY, FRMSPCFF, RFMSPCFF, CONVERT, VERRMSGS	V3000'1,U15U209A.HP32209.SUPPORT V3000'2, " " " V3000'3, " " " V3000'4, " " " V3000'5, " " " V3000'6, " " " V3000'7, " " " V3000'8, " " "
	IMAGE/3000	DBUTIL,DBLOAD, DBUNLOAD, DBSTORE, DBRESTOR, DBRECOV, DBSCHEMA, DBDRIVER, DBDUMP	IMAGE01, ⋮ IMAGE11,
	KSAM/3000	DSAMUTIL	KSAM01, ⋮ KSAM07,
	QUERY/3000	QUERY	

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
			RPG'CALC2,U01U104A.HP32104.SUPPORT RPG'IMAGE'IO,U01U104A.HP32104.SUPPORT RPG'INIT2,U01U104A.HP32104.SUPPORT
32105A	APL/3000	APL	APLSEG01,U01U105A.HP32105.SUPPORT
32190A	DS/3000	DSMON, DSTEST, IODSO, IODSTRMO, DS2026, DS2026CN, DSCOPY, NETCAT	DSSEG1,U00U190A.HP32190.SUPPORT,S DSSEG2,U00U190A.HP32190.SUPPORT,S DSSEG3,U00U190A.HP32190.SUPPORT,S DSSEG4,U00U190A.HP32190.SUPPORT,S DSSEG5,U00U190A.HP32190.SUPPORT,S DSMISC,U01U190A.HP32190.SUPPORT,S DSIOM ,U01U190A.HP32190.SUPPORT,S DSRTECALLS,U02U190A.HP32190.SUPPORT,S
32192A	MRJE/3000	MRJEMON, MRJELOGR, MRJEOUT, MRJE, CSDMRJEO, * DRIVERS=CSSMRJEO	MRJEMISC1 ,U05U192A.HP32192.SUPPORT ,S MRJEMISC2 ,U05U192A.HP32192.SUPPORT ,S MRJESLCP ,U13U131A.HP30131.SUPPORT ,S
32193A	MTS/3000	MPMON, MPTEST, MPCONFIG, CSDMTSO, * DRIVERS=CSSBSC1	BSCSLCP1,U10U131A.HP30131.SUPPORT,S MPMONCMD,U03U193A.HP32193.SUPPORT,S
32205B	SCIENTIFIC LIBRARY/3000		S'LIB'01,U00U205B.HP32205.SUPPORT S'LIB'02,U00U205B.HP32205.SUPPORT
32206A	DEL/3000	FORMAINT	DEL'TERM'I'O,U01U206A.HP32206.SUPPORT DEL'FORM'I'O,U01U206A.HP32206.SUPPORT DEL'OPEN'CLOSE,U01U206A.HP32206.SUPPORT DEL'EDITS,U01U206A.HP32206.SUPPORT
32213C	COBOL/3000	COBOL	
32222A	TRACE/3000 (N/A TO SERIES II)		TRACE0',U00U222A.HP32222.SUPPORT,P TRACE1',U00U222A.HP32222.SUPPORT,P

(cont.)

SERIES

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
32223A	CROSS ASSEMBLER-2100	XA2100	
32226A	CROSS LOADER-2100	XL2100	
32229A	IML/3000	IOM3270, TTSSON, TTSUSER, TTSMON, CATIML, CSDIMLO	TTSINTR ,U04U229A.HP32229.SUPPORT TTSCONFG ,U05U229A.HP32229.SUPPORT IOMONITOR3270 ,U07U229A.HP32229.SUPPORT,S
32233A	COBOLII/3000	COBOLII, COBCAT, COBEDIT, COBCNY, P03P233A, T02T233A	
32238A	OPT/3000	OPT	
36578A	TDP/3000	TDP, SCRIBE, DHANDLE, EXCEPT, HELPFILE, FILESYS	

SERIES II/SERIES III

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
30126A	CALCOMP/3000	IOPLOTO	CALCOMPSEG,U00U126A.HP30126.SUPPORT
30130E	RJE/3000	RJE	
30300B	30361B PROGRAMMABLE CONTROLLER/BCS		
30301B	30361B-1 PROG. CTRLR/RTE-C	XRTCGEN	DNLDSYS,U01U301B.HP30301.SUPPORT DNLDUSER,U03U301B.HP30301.SUPPORT
32100A	SPL/3000	SPL	
32101B	BASIC/3000	BASIC	
32102B	FORTRAN/3000	FORTRAN	
32103B	BASICOMP/3000	BASICOMP	B'BASICOUT,U01U103B.HP32103.SUPPORT B'BASICIN ,U01U103B.HP32103.SUPPORT B'FILEIO ,U01U103B.HP32103.SUPPORT B'FUNC ,U01U103B.HP32103.SUPPORT B'ERROR ,U01U103B.HP32103.SUPPORT
32104A	RPG/3000	RPG U01U104A.CREATOR.SYS	RPG'CALC,U01U104A.HP32104.SUPPORT RPG'IMAGE,U01U104A.HP32104.SUPPORT RPG'INIT,U01U104A.HP32104.SUPPORT RPG'ERROR,U01U104A.HP32104.SUPPORT RPG'DUMP,U01U104A.HP32104.SUPPORT RPG'IO,U01U104A.HP32104.SUPPORT RPG'ISAM,U01U104A.HP32104.SUPPORT RPG'WORKSTN,U01U104A.HP32104.SUPPORT

(cont.)

SERIES 33/30/44

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
30130E	RJE/3000	RJE	
32100A	SPL/3000	SPL	
32101B	BASIC/3000	BASIC	
32102B	FORTRAN/3000	FORTRAN	
32103B	BASICOMP/3000	BASICOMP	B'BASICOUT,U01U103B.HP32103.SUPPORT B'BASICIN ,U01U103B.HP32103.SUPPORT B'FILEIO ,U01U103B.HP32103.SUPPORT B'FUNC ,U01U103B.HP32103.SUPPORT B'ERROR ,U01U103B.HP32103.SUPPORT
32104A	RPG/3000	RPG, U01U104A.CREATOR.SYS	RPG'CALC,U01U104A.HP32104.SUPPORT RPG'IMAGE,U01U104A.HP32104.SUPPORT RPG'INIT,U01U104A.HP32104.SUPPORT RPG'ERROR,U01U104A.HP32104.SUPPORT RPG'DUMP,U01U104A.HP32104.SUPPORT RPG'IO,U01U104A.HP32104.SUPPORT RPG'ISAM,U01U104A.HP32104.SUPPORT RPG'WORKSTN,U01U104A.HP32104.SUPPORT RPG'CALC2,U01U104A.HP32104.SUPPORT RPG'IMAGE'IO,U01U104A.HP32104.SUPPORT RPG'INIT2,U01U104A.HP32104.SUPPORT
32190A	DS/3000	DSMON, DSTEST, IODSO, IODSTRMO	DSSEG1,U00U190A.HP32190.SUPPORT,S DSSEG2,U00U190A.HP32190.SUPPORT,S

(cont.)

SERIES 30/33/

PRODUCT NO. & VERSION	NAME	FILES in PUB.SYS unless otherwise noted.	SEGMENTS in SL.PUB.SYS to be deleted when this product is deleted and file where segments are located.
		DS2026, DS2026CN, DSCOPY, NFTCAT	DSSEG3,U00U190A.HP32190.SUPPORT,S DSSEG4,U00U190A.HP32190.SUPPORT,S DSSEG5,U00U190A.HP32190.SUPPORT,S DSMISC,U01U190A.HP32190.SUPPORT,S DSIOM ,U01U190A.HP32190.SUPPORT,S DSRTECALLS,U02U190A.HP32190.SUPPORT,S
32192A	MRJE/3000	MRJEMON, MRJELOGR, MRJEOUT, MRJE, CSDMRJEO	MRJEMISC1 ,U05U192A.HP32192.SUPPORT ,S MRJEMISC2 ,U05U192A.HP32192.SUPPORT ,S
32193A	MTS/3000	MPMON, MPTEST,	MPMONCMD,U03U193A.HP32193.SUPPORT,S
32199A	DISCCOPY/3000	DISCCOPY, DCOPYMSG	
32205B	SCIENTIFIC LIBRARY/3000		S'LIB'01,U00U205B.HP32205.SUPPORT S'LIB'02,U00U205B.HP32205.SUPPORT
32206A	DEL/3000	FORMAINT	DEL'TERM'I'O,U01U206A.HP32206.SUPPORT DEL'FORM'I'O,U01U206A.HP32206.SUPPORT DEL'OPEN'CLOSE,U01U206A.HP32206.SUPPORT DEL'EDITS,U01U206A.HP32206.SUPPORT
32213C	COBOL/3000	COBOL	
32229A	IML/3000	IOM3270, TTSSON, TTSUSER, TTSMON, CATIML, CSDIMLO	TTSINTR ,U04U229A.HP32229.SUPPORT TTSCONFG ,U05U229A.HP32229.SUPPORT IOMONITOR3270 ,U07U229A.HP32229.SUPPORT,S
32233A	COBOLII/3000	COBOLII, COBCAT, COBEDIT, COBCNV, P03P233A, T02T233A	
32230A	OPT/3000	OPT	
36578A	TDP/3000	TDP, SCRIBE, HANDLE, EXCEPT, HELP, FILE, FILESYS	

HEWLETT hp PACKARD

PROCESS LEVEL REDIRECTION

OF

$STDIN and $STDLIST

and

PASSING STRINGS

TO

PROCESSES

I. PRODUCT IDENTIFICATION

A. Name and Project Number

Redirection of $STDIN and $STDLIST - TRC #48

B. Product Abstract

This project addresses the issues involved in extending the current assignments of the system defined files $STDIN and $STDLIST to user specified files at user process creation time. In particular, it describes a new intrinsic for creating user processes with $STDIN and $STDLIST set to files other than the system defaults and a modification to the MPE :RUN command to allow specification of $STDIN and $STDLIST for the program being executed.

The other portion of the project, dealing with $STDIN and $STDLIST redirection at the job/session level is still under investigation.

C. Project Personnel

Chris Moeller

II. PRODUCT SPECIFICATIONS

A. User Definition

Every process created by a user must have available to it a standard input stream and a standard output stream for the duration of its existence. In general, a user process will need to read data to perform its computations and this function is fulfilled by the standard input stream. Of course, the user can request input from other files that he specifies. A user process will also need to send results to the outside world and this is 1 of the 2 functions fulfilled by the standard output stream. The other (and more important) function of the standard output stream is to provide a place to send messages concerning errors and aborts. If the user process should terminate in an unexpected manner, the user must be informed.

These standard input and output streams are currently provided to the user in the form of 2 system defined files called $STDIN and $STDLIST. These files are automatically assigned to every user process at creation time and they remain in effect until the process is terminated. The settings of $STDIN and $STDLIST for user processes are always defined to be the $STDIN and $STDLIST of the job/session user-main process (CI). For sessions, this is the interactive terminal; for jobs, it is the spooled input and spooled output devices. The settings of $STDIN and $STDLIST for user processes may not be redefined.

This project will provide users with the capability to define $STDIN and $STDLIST for their processes to be any file they desire. This will be done when specifying a program to be executed (which causes a process to be created) and when the user calls an intrinsic to explicitly create a new process on his behalf.

B. Detailed Functional Specifications

On the following pages are the descriptions for the new process creation intrinsic and the modification to the :RUN command. Included after each description are examples showing the use of the new facilities.

1.1 DEFINITION OF NEW PROCESS CREATION INTRINSIC

A new, extensible intrinsic to create a process with $STDIN and $STDLIST set to files other than the calling process'S $STDIN and $STDLIST is defined as follows:

```
                        I     I       BA          IA       LA        OV
CREATEPROCESS (error, pin, progname, itemnums, items);
```

pin — An integer in which the PIN of the newly created process is returned to the calling process. If there was an error in creating the new process (i.e. parameter error > 0) , a 0 is returned.

progname — A byte array containing a string, terminated by a non-alphanumeric character other than period or slash, specifying the name of the program file to be run by the new process. Note that this may be a fully qualified file name (with group and account).

itemnums — An array containing the item numbers (in any order) of the options a user wishes to be used in creating the new process. This array must contain a 0 as its last element to indicate the end of the option list.

items — An array containing the items (in the same order as the item numbers in array itemnums) to be used in creating the new process.

error — An integer indicating success or failure type:

0 process created as requested

1 caller lacks PH capability

2 required parameter (other than "error") omitted

3 parameter address (other than "error") out of bounds

4 out of system resources (PCBs, DSTs, etc.)

5 process not created because an invalid itemnumber was specified

6 process not created because progname does not exist

7	process not created because progname specifies an invalid program file
8	process not created because entryname does not exist or is invalid
-9	process created with default stacksize from progfile (specified stacksize was < 512)
-10	process created with default dlsize from progfile (specified dlsize was < 0)
-11	process created with default maxdata from progfile (specified maxdata was <= 0)
-12	process created with dlsize rounded up to next 128 word multiple
-13	process created with maxdata decreased to configuration maximum
-14	process created with maxdata increased to dlsize + globsize + stacksize (globsize is defined to be primary DB space + secondary DB space)
15	process not created because dlsize + globsize + stacksize was > configuration maximum stack size
16	process not created because a 'hard' load error occured (e.g. i/o error reading prog file, etc.)
17	process not created because an illegal value was specified for priorityclass
18	process not created because specified $STDIN could not be opened
19	process not created because specified $STDLIST could not be opened

Condition codes returned:

CC= Successful - error = 0
CC> Successful - error (<0) is a warning number
CCL Unsuccessful - error (>0) is an error number

Parameters error, pin, and progname are required; itemnums and items are optional.

The item numbers in the array itemnums indicate the options to be applied in creating the new process. The corresponding items in the array items give the information necessary for each particular option to be used.

The following is a list of the currently defined item numbers and the corresponding items:

itemnumber	item
1	A pointer to a byte array containing the name of the entry point in the program where the new process is to begin execution. The name is specified as a string of characters terminated by a blank.
2	An integer containing a parameter to be passed to the new process (accessed through Q-4 of the outer block).
3	A logical containing the load option flags to be used in loading the program file for the new process. This parameter has the same definition as the "flags" parameter of the CREATE intrinsic.
4	An integer specifying the initial stack size (Q - Z)
5	An integer specifying the initial dlsize (DL - DB) for the new process.
6	An integer specifying the maximum stack size (DL - Z) for the new process (i.e. maxdata).
7	A string of 2 ASCII characters specifying the priority class in which the new process is to be scheduled.
8	A pointer to a byte array containing the definition of a file to be used as $STDIN for the new process (see description below).
9	A pointer to a byte array containing the definition of a file to be used as $STDLIST for the new process. (see description below). for the new process.
10	A logical value indicating suspension and anticipated source of re-activation. Specification of this parameter causes the newly created process to be ACTIVATEd automatically upon creation completion. The meanings of the individual bit fields of this parameter are the same as those of the "susp" parameter of the ACTIVATE intrinsic.

Item number 8 indicates that the corresponding item in the item array is the address of a byte array which contains the definition of the file to be used as $STDIN for the new process. This byte array must contain an ASCII string (terminated by carriage return) which is the right hand side of a file equation specifying the file to be used as $STDIN (i.e. everything after the ":FILE formaldesignator = " portion of the file equation).

Item number 9 indicates that the corresponding item in the item array is the address of a byte array which contains the definition of the file to be used as $STDLIST for the new process. This array is defined as above for $STDIN.

If item numbers 8 or 9 are not specified, the default $STDIN and $STDLIST will be used in creating the new process. These defaults are the current $STDIN and $STDLIST files for the creating (father) process.

Note that Process-Handling capability is required to call this intrinsic and that it may not be called in split stack mode. If it is called in split stack mode, the calling process will be aborted. The calling process will also be aborted if the intrinsic is called with an invalid address for parameter "error" or if it is omitted.

1.. EXAMPLE OF CREATEPROCESS INTRINSIC CALL

The following example illustrates the use of the new CREATEPROCESS intrinsic to create a process with $STDIN and $STDLIST set to files other than the default $STDIN/$STDLIST of the user-main process.

Assume it is desired to create a process with an initial stack size of 1K words and a DL area of 256 words. The process will be passed a parameter value of 1 (referenced through Q-4 of the outer block). $STDIN for the process will be an existing disk file named "IN". $STDLIST for the process will be a tape file named "LIST" with 512 byte fixed length records.

```
begin
  << program to create a special process >>

  integer error, pin, param := 1, stacksize := 10000;
  byte array testprog(0:8) := "TESTPROG ";

  integer array options(0:10);
  logical array items(0:10);

  equate cr = %15;      << carriage return character >>
  << *** definition of new process's $STDIN *** >>
  byte array infile(0:50) := "IN,OLD; MULTI",cr;

  << *** definition of new process's $STDLIST *** >>
  byte array outfile(0:50) := "LIST; REC=-512; DEV=TAPE;",
                              " ACC=OUT; MULTI",cr;

  intrinsic CREATEPROCESS, ACTIVATE, SUSPEND;


  << set up arrays to specify special options to be used >>
  options(0) := 8;    items(0) := @infile;
  options(1) := 9;    items(1) := @outfile;
  options(2) := 4;    items(2) := stacksize;
  options(3) := 5;    items(3) := 256;    << dlsize >>
  options(4) := 2;    items(4) := param;
  options(5) := 0;    << terminates option list >>

  CREATEPROCESS (error, pin, testprog, options, items);

  if error = 0 then
    begin   << successful creation >>
      ACTIVATE (pin);
        ...
    end
  else if error < 0 then
    begin   << process created, but with unexpected result >>
      print (error and some message);
      ACTIVATE (pin);
        ...
    end
  else
    begin   << process NOT created >>
      print (error and some message);
        ...
    end;

  if pin <> 0 then SUSPEND (2);   << wait for son to activate >>
end.
```

3.1 DEFINITION OF NEW :RUN COMMAND

An extension will be made to the existing MPE :RUN command to allow the user to specify files to be used as $STDIN and $STDLIST by the program about to be executed. This extension will be in the form of 2 additional parameters to the :RUN command, as described below:

```
:RUN progfile ... [;STDIN=[*formaldesig] ][;STDLIST=[*formaldesig] ]
                          fileref                  fileref[,NEW]
                          $NULL                    $NULL
```

formaldesig	The formal file designator for a file previously specified in a file equation.
fileref	The name of an old permanent disc file or, if the NEW option is specified, the name to be assigned to a job/session temporary disc file created with system defaults.

The system defaults for the NEW file would be fixed length ASCII 132 byte records with a maximum file size of 1023 records.

If nothing is specified on the right-hand side of the "=" sign or if "STDIN=" and "STDLIST=" are not specified on the :RUN command, the job/session main $STDIN and $STDLIST files are assumed.

3.2 EXAMPLES OF NEW :RUN COMMAND USE

The following examples illustrate several possible combinations of $STDIN and $STDLIST files used in running a program.

The first example runs a program with $STDIN set to an old disc file named INPT and $STDLIST set to the line printer.

```
:FILE LPFILE; DEV=LP
:RUN TESTPROG; MAXDATA=100000; STDIN=INPT; STDLIST=*LPFILE
```

The next example runs a program with $STDIN set to a tape file with fixed length 128 byte records. $STDLIST is set to the default value (terminal for sessions; line printer for jobs).

```
:FILE INTAPE; DEV=TAPE; REC=-128
:RUN TESTPROG; STACKSIZE=5000; STDIN=*INTAPE
```

The next example runs a program with $STDIN set to an old disc file named INPT (but referenced through a file equation) and $STDLIST set to a temporary disc file named RESULTS which will be created automatically by the :RUN command.

```
:FILE INFILE=INPT,OLD; DISC
:RUN TESTPROG; DEBUG; STDIN=*INFILE; STDLIST=RESULTS,NEW
```

The next example runs a program with $STDIN set to an old disc file named A and $STDLIST set to an old disc file named B.

```
:RUN TESTPROG; STDIN=A; STDLIST=B
```

The next example runs a program with $STDLIST set to $NULL and $STDIN set to the default value (terminal for sessions; spoolfile for jobs).

```
:RUN TESTPROG,ENTRY; LIB=G; STDLIST=$NULL
```

The last example gives 2 :RUN commands which both run a program with $STDIN and $STDLIST set to the default values.

```
:RUN TESTPROG

:RUN TESTPROG; STDIN= ; STDLIST=
```

III. PROJECT OBJECTIVES (Revised)

A. Hardware Requirements and Dependencies

1 Series III or Series 33

B. Software Requirements and Dependencies

None

C. Testing Objectives

The CREATEPROCESS intrinsic will 1st be tested for correct operation without specifying files for $STDIN and $STDLIST. This includes testing to insure that all of the error conditions are reported correctly to the user through parameter "error". Thereafter, the intrinsic will be tested specifying a large variety of $STDIN and $STDLIST files. The :RUN command will also be tested in this fashion.

The AM/3000 project of the BSP Lab will be using both the intrinsic and the :RUN command in their beta test site release, which will provide an additional set of tests.

D. Project Personnel

1 man/month

E. Computer Equipment

2 hours/day for the duration of the project

F. Schedule Estimates and Milestones

Sep. 25 - ERS approved
Oct. 15 - Begin design & coding
Nov. 12 - CREATEPROCESS intrinsic coded & tested
Nov. 19 - Extensions to :RUN command coded & tested

1.1 BACKGROUND

Currently in MPE, a process may pass information to another process at process creation time. This information is in the rather limited form of a single word integer which can be accessed through Q-4 of the outer block of code. A much more general approach would be to allow a process to pass an arbitrary amount of information to the created process in the form of a string of ASCII characters. This would provide a very convenient mechanism for sending information to a process on a 1-time basis at creation time. While this facility could be used at all levels of processes (system and user) within MPE, it would be most useful at the Command Interpreter level. There it would provide a simple link from the process running a program back to the Command Interpreter level where additional information could be obtained (from the :RUN command). This would be useful not only to users, but also to our subsystems. Subsystem programs could be altered to make use of a passed string to allow single-line invocation for simple operations.

1.2 APPROACH

After considering several options, the best implementation for this capability appears to be to place the string in the created process's stack immediately after the DB global data area and before the initial stack marker to TERMINATE. A DB relative byte pointer to the string will be provided at Q-5 and a byte count of the length of the string will be provided at Q-6 (see diagram on last page).

In order to make use of the passed strings, MPE must be changed in 2 places. The 1st is in the new intrinsic CREATEPROCESS. Additional item numbers and error returns must be added to implement the ability to pass a string to a newly created process. The 2nd is on the :RUN command itself. A new keyword parmeter must be added which will allow specification of a string to be passed to the process created to run the user's program. The external specifications of these changes are described below.

1.3 SPECIFICATIONS

The definition of intrinsic CREATEPROCESS will be changed to include 2 more item numbers, 1 new error return, and modifications to 2 of the existing error returns (see description of CREATEPROCESS).

The following item numbers and items will be added:

itemnumber	item
11	A pointer to a byte array containing a string of information to be passed to the new process. The length of the string is specified with itemnumber 12.

12 An integer specifying the length in bytes of the string specified with item number 11.

Item numbers 11 and 12 indicate that a string is to be passed to the new process. The string will be placed just after the global area of new process's stack. A DB relative byte pointer to the string in the new process's stack will be placed at Q-5 of the stack (where Q is the initial value of the Q-register at activation time) and the length of the string in bytes will be placed at Q-6. If no string is specified to be passed to the new process, Q-5 and Q-6 will both contain 0.

The following error returns will be modified or added to the set of error returns from CREATEPROCESS:

-14 process created with maxdata increased to dlsize + globsize + stringsize + stacksize (globsize is defined to be primary DB space + secondary DB space; stringsize is defined to be the space, if any, occupied by a passed string)

-15 process not created because dlsize + globsize + stringsize + stacksize was > configuration maximum stack size

-16 process not created because string to be passed to new process was invalid (pointer without length, length without pointer, or length exceeds stack size of calling process)

The definition of the MPE :RUN command will be changed to include a new keyword parameter.

:RUN progfile ... [;INFO=string]

string A string of ASCII characters delimited by a pair of 's or "s.

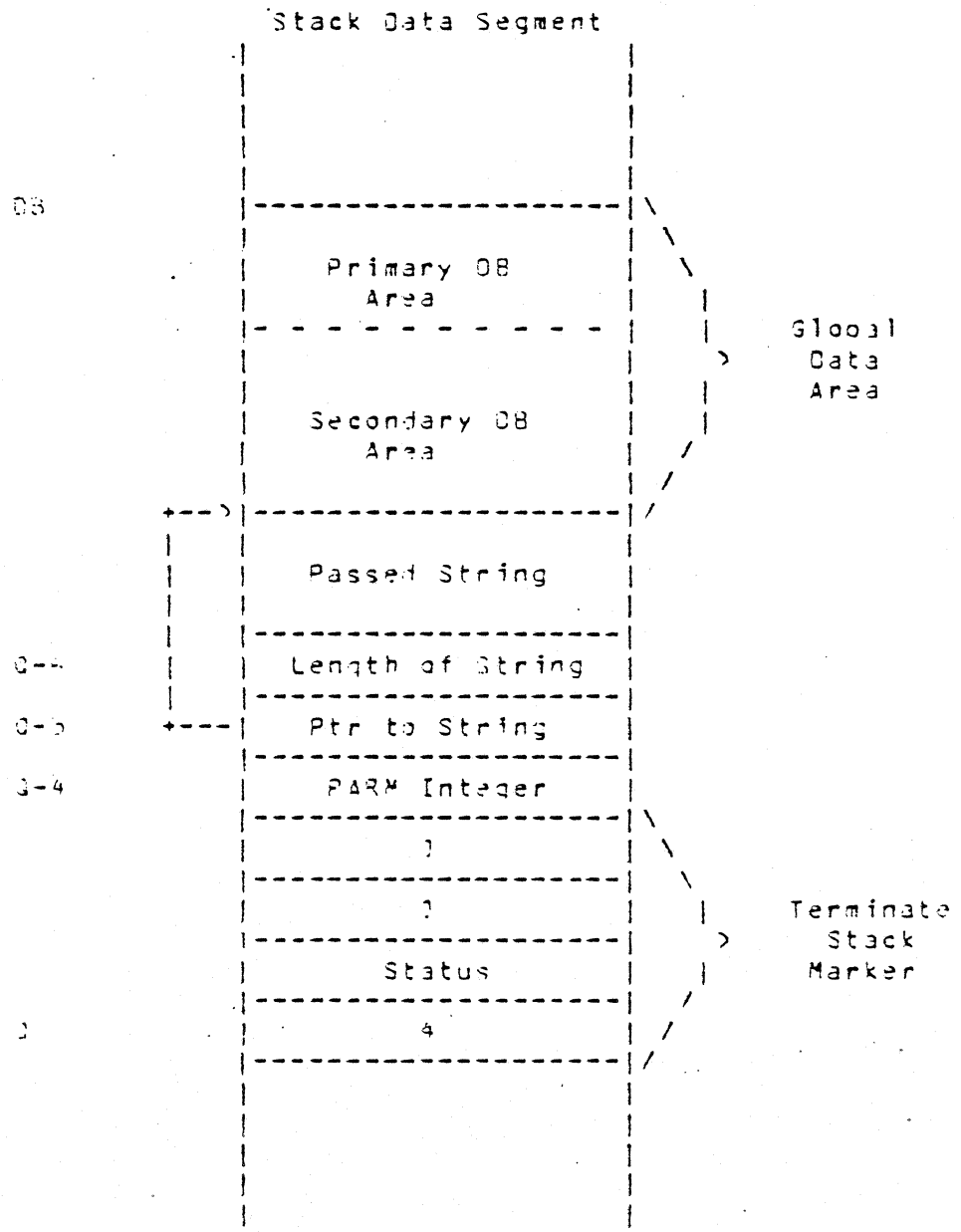
If the delimiting character is desired within the string, it can be doubled as in most programming languages. An example appears below.

:RUN MYPROG; MAXDATA=20000; INFO='A test with '' and " characters'

The string passed to the created process would be

A test with ' and " characters

FORMAT OF STACK DATA SEGMENT AT PROCESS CREATION TIME



INTERPROCESS COMMUNICATION AND CIRCULAR FILES

INTRODUCTION

Interprocess communication (IPC) is a facility of the file system which permits multiple user processes to communicate with each other in an easy and efficient manner. To accomplish this, IPC uses message files as the interface between user processes. These message files act as first-in-first-out queues of records, with entries made by FWRITEs and deletions made by FREADs: one process may submit records to the file with FWRITE instructions while another process takes records from the file using FREADs.

Occasionally a process may attempt to read a record from an empty message file, or write a record to a message file that is full. In such situations, the file system will usually cause the process to wait until its request can be serviced; that is, until another process either writes a record to the empty file or reads a block of records from the full file.

There is a unidirectional flow of information between a given process and a message file: a process identified as a "reader" may only read from the file, and not write; a "writer" process may only write to the file, and not read. (If it is necessary for the same process to read and write, it may open the message file twice, once as a reader and once as a writer.) More than one message file may be associated with a process, and the process may be configured as a reader to some of the files and as a writer to others. A given message file typically has one reader, though more are allowed, and one or more writers.

Applications for IPC exist wherever it is necessary for processes to communicate with one another. In the case of a father process with several sons, message files may serve as interfaces between the processes: through one file, the father may direct the activities of the sons; through another, the sons may inform the father of their progress. Message files may also serve as object managers during data base operations: several writers may send information to a file which serves as the single source from which the data base actually receives the information.

OPERATION

Message files are maintained and manipulated by several intrinsics. The FOPEN, FREAD, FWRITE, FCONTROL, and FCLOSE intrinsics operate upon the files to yield a unidirectional, first-in-first-out message queue.

FOPEN — Establishes a connection to a message file. With FOPEN, a user process identifies itself as either a reader or writer; readers access the head of the message file and writers access the tail. Incompatible parameters that are specified with FOPEN are changed. For example, since messages are read or written to the file one record at a time, a multirecord parameter is changed. If FOPEN is used to access a new file, a new message file is created.

> Note: There are different AOPTION specifications for slightly different types of writer processes. In one case, if a writer is the first accessor to a message file, the file's contents are purged; in another case, the writer simply appends records to the tail of the file. These AOPTIONs are discussed in a later section.

FREAD — Reads one record from the head of a message file. The record is copied to the reader's TARGET area and is logically deleted from the message queue; the next record is now at the head of the file. If a process tries to read from an empty message file, the file system causes it to wait until a writer process enters a record to the file; if there are no writers associated with the message file, an end-of-file indication, CCG, is returned.

> Note: If the message file is empty and there are no writers, the process will wait if there is an FCONTROL 45 in effect, or if this is the first FREAD after the reader's FOPEN.

FWRITE — Appends one record to the tail of a message file. If a process tries to write to a file that is full, the file system causes it to wait until a reader process takes a block of records from the file; if there are no readers associated with the record file, an end-of-file indication, CCG, is returned.

> Note: If the message file is full and there are no readers, the process will wait if there is an FCONTROL 45 in effect, or if this is the first FWRITE after the writer's FOPEN.

FCONTROL	Supplies various control functions during a process that is using a message file. These control functions permit a process to take advantage of the additional features of IPC, which are discussed in detail in the next section.
FCLOSE	Breaks a process' connection with a message file. If the process reopens the same file later, it may do so as either a reader or a writer; regardless of what it was previously.

ADDITIONAL FEATURES

Besides the regular attributes of IPC and message files, several features are available for use with these facilities. Writer ID's and nondestructive reads are specifically intended for use with IPC; time-outs, copy access, the global multiaccess option and the ability to append to variable-length files are general enhancements to the file system.

Writer ID's. When a writer process opens a message file, the file system assigns a unique 16-bit ID number to the writer. Each record the process writes to the message file is prefixed with this number by FWRITE. When the writer closes the file, the ID number is no longer associated with the process and may be reused. Whenever a writer opens or closes a message file, records are written to the file indicating these actions. Record prefixes and open/close records are usually transparent to the readers of the message file, but by issuing an FCONTROL 46, the reader process may see them. The interested reader may use the writer ID's to determine the source of the records it is receiving.

Time-outs. A reader or a writer process may limit the length of time it will wait to be serviced. By issuing an FCONTROL 4, a reader may specify the maximum number of seconds it will permit the file system to keep it waiting for a record to be written to an empty message file; a writer may also use FCONTROL 4 to specify the maximum number of seconds it will wait for a record to be read from a full file.

Copy access. When records are read from a message file, FREAD logically deletes them as it reads. In order to copy a file without destroying it, the message file must be opened with the file copy option specified in the AOPTIONs of the FOPEN, or the COPY keyword may be specified in a FILE equation. When this option is selected, the message file is treated as a standard sequential file rather than as a message queue, and may be copied safely. The file may then be read by logical record or by block, and information may be written to it by block.

Nondestructive read. By issuing an FCONTROL 47, a reader may avoid deleting the next record it reads; the record will remain at the head of the message queue. This feature differs from the copy access feature in that it is a temporary condition: the second FREAD following the FCONTROL 47 will reread the record and delete it in the usual manner.

Global multiaccess. When the global multiaccess option is requested, processes located in different jobs or sessions may open the same file. The file must be accessible; global multiaccess is only unavailable to message files when they have been opened in exclusive copy mode. The global multiaccess option may be requested in the AOPTIONs of the FOPEN to the file, or by using the GMULTI keyword in a FILE command to create the file.

Appending to variable-length files. Variable-length files may now be opened with append access. It is not necessary to have records of the maximum possible size, so space is conserved.

USING IPC

Message files can be created in several ways. When a user process opens a new file and indicates in the AOPTIONs that it will be a message file, the FOPEN intrinsic creates the new message file. In order to create a message file with the BUILD command, use the MSG keyword; for example, to build a message file named SARA, enter:

```
:BUILD SARA; MSG
```

A new message file may also be created with a FILE command. Use the MSG keyword for a new file:

```
:FILE LISBETH, NEW; MSG
```

A message file named LISBETH is created.

When you perform a LISTF,2 command, message files will be identified by an "M" in the TYP field; SARA is identified here:

FILENAME	CODE	LOGICAL RECORD					SPACE		
		SIZE	TYP	EOF	LIMIT	R/B	SECTORS	#X	MX
SARA		128W	VBM	0	1031	1	258	1	8

Other types of files are similarly indicated by a token in the TYP field:

K -- identifies a KSAM file
R -- identifies a Relative I/O file
O -- identifies a Circular file

A blank in this column indicates a standard sequential file. Circular files are discussed in a later section.

Occasionally, you might create a message file and specify a certain number of records for the file to contain, only to discover that the file system has allocated more records for the file than you requested. The reason for this is that the file system is maintaining the necessary internal structure for the message file. The file system has four basic rules for establishing this structure when the message file is created:

1 Since records are written to the message file every time a writer process opens or closes the file, the file system adds two records to the requested number to allow for a minimum of one open and one close operation.

2 The requested number of records is rounded up to fill an even number of blocks.

3 The file system adds an extra block to the message file for the file label to occupy. (This block is transparent to you.)

4 Each extent is the same size; that is, the file system assigns the same number of blocks to each extent.

For example, suppose you want to create a message file named ODDSIZE:

```
:BUILD ODDSIZE; MSG; REC=,3; DISC=51,8
```

You have specified a message file with fifty-one records, three records per block, that occupies eight extents. The file system will adjust the number of records to conform to the rules for message file structure:

The file system adds two records to allow for one open and one close indication; the number of records goes from 51 to 53.

The number of records is rounded up to 54 to provide an even number of blocks. With three records per block, 54 records will fill 18 blocks.

An additional block is added to the file to accommodate the file label. Now the file contains 19 blocks.

The eight extents must all be the same size, so the number of blocks is increased from 19 to 24. Each extent now contains three blocks.

Of the 24 blocks in ODDSIZE, 23 are data blocks and one contains the file label, which is invisible to you. With three records per block, 23 blocks contain a total of 69 data records!

FEATURES OF INTRINSICS FOR MESSAGE FILES

There are a few features of several intrinsics which apply specifically to message files. Most of these features are found in FOPEN and FCONTROL, but several other intrinsics are also affected.

Parameters that are omitted in the following descriptions retain their normal range of values and their normal default values.

FOPEN

FOPTIONS: (2:3) - File type. Determines the type of file to create for a new file. If the file is old, this field is ignored.

0 - Ordinary file

1 - KSAM file

2 - Relative I/O file

4 - Circular file; discussed in the next chapter

6 - Message file

Note: A Default Designator, FOPTION (10:3), of any value other than "filename" will override this field. Exception: KSAM will refuse to open the file.

(8:2) - Record format. Message files are always internally formatted as variable-length records. However, a message file can appear as a fixed file to an opener. There is no difference for a writer, but a reader will have the portion of his target area which exceeds the record filled with blanks or zeroes.

0 - Fixed

1 - Variable

2 - Undefined; changed to variable

AOPTIONS: (3:1) - File copy. This feature permits a message file to be treated as a standard sequential file, so it can be either copied (logical record read) or replicated (block read and write) to another file.

0 - The file will be accessed in its native mode; that is, a message file will be treated as a message file.

1 - The file is to be treated as a standard, sequential file with variable-length records. This allows nondestructive reading of an old message file at either the logical record or physical block level. Only block level access is permitted if the file is opened with write access. These blocks are checked for proper message file format to prevent incorrectly formatted data from being written to the message file while it is vulnerable.

Setting this bit on causes all the remaining file parameters to have their normal default: Exception: Exclusive field -- read = semi, write = exclusive.

(5:2) - Multiaccess mode. This feature permits processes located in different jobs or sessions to open the same file.

0 - No multiaccess. This value is changed to 2 to allow global multiaccess.

1 - Only intra-job multi-access allowed.

2 - Inter-job multi-access allowed; this is the same as specifying the GMULTI option in a FILE command.

3 - Undefined. If this is specified, the FOPEN will be rejected with an error code of 40: ACCESS VIOLATION.

(7:1) - Inhibit buffering. For message files, this bit is set off.

Note: Readers may open a message file with NOBUF if they are in copy mode; this determines whether they will be reading the file record by record or block by block.

(8:2) - Exclusive.

0 - Default; set to 1 to allow one reader and one writer.

1 - Exclusive; allow one reader and one writer.

2 - Semi; allow one reader and multiple writers.

3 - Share; allow multiple readers and writers.

(11:1) - Multirecord. For message files, this bit is set to 0.

(12:4) - Access type. These bits specify whether the user will be a reader or a writer process.

0000 - READ access only. The FWRITE, FWRITEDIR, FUPDATE, FPOINT, FSPACE, FREADDIR, and FREADSEEK intrinsic calls cannot reference this file. This access type requires both read and write access capability to the file. A process that has opened a file with this access type is a "reader."

0001 - WRITE access only. If this is the first accessor to the file and the process has write access capability, then the file's contents are purged. If this is not the first accessor to the file, the access type is set to APPEND. The FWRITEDIR, FUPDATE, FPOINT, FSPACE, FREADDIR, and FREADSEEK intrinsics cannot reference this file. A process that has opened a file with this access type is a "writer."

0010 - WRITE SAVE access. Set to APPEND access.

0011 - APPEND access only. The FREAD, FREADDIR, FUPDATE, FPOINT, FSPACE, FWRITEDIR, and FREADSEEK intrinsics cannot reference this file. This access type requires append capability to the file. A process that has opened a file with this access type is a "writer."

DEVICE: This field is relevant only if this is a new file. The DEVICE field must either be omitted or specify a disc; specification of any device other than a disc causes the device to be opened.

NUMBUFFERS: (0:11) - Ignored.

(11:5) - Value between 2 and 31; default is 2. This parameter must not exceed the physical record capacity of the file.

FILESIZE: Must equal or exceed two physical records and/or three logical records; the default is 64 logical records. The number of records is rounded up to completely fill the last block and to make the last extent the same size as the other extents.

FCONTROL

A few controlcodes deal specifically with IPC. Those not mentioned here are invalid when IPC is being used.

CONTROLCODE	PARAM	DESCRIPTION
2	-	Complete all I/O; ignored in the case of message files.
4	integer	Set time-out interval. This applies to both FREADs and FWRITEs. The time-out will be armed at the beginning of the I/O request and cleared when the I/O completes. PARAM specifies the length of the time-out in seconds. A value of zero disables time-outs on the file.
6	-	Write end-of-file. Used only to checkpoint the state of the file by writing out the file label and buffer area to disc; this ensures that the message file can span system crashes. No EOF is written.

45	TRUE	Enable extended wait. Permits a reader to wait on an empty file that is not associated with any writer, or a writer to wait on a full file that has no reader. This FCONTROL will remain in effect until another FCONTROL 45 is issued with a PARAM value of FALSE.
	FALSE	Disable extended wait. Specifies that when an FREAD encounters an empty file that has no writer, or an FWRITE encounters a full file that has no reader, it will return an end-of-file condition. (Default.)
46	TRUE	Enable reading the writer's ID. Each record read will have a two-word header. The first word will indicate the type of record: 0 - data 1 - open 2 - close The second word will contain the writer's ID number. If the record is a data record, one or more words of data will follow the header; open and close records contain no more information.
	FALSE	Disable reading the writer's ID. Only data is read to the reader's TARGET area. The open and close records are skipped and deleted by the file system when they come to the head of the message queue, and the two-word header is transparent to the reader. (Default.)
47	TRUE	Nondestructive read. The next FREAD by this reader will not delete the record. Subsequent FREADs will be unaffected.
	FALSE	The next FREAD by this reader will delete the record. (Default.)

FCHECK

There is one error message that is returned only when using IPC:

151 CURRENT RECORD WAS LAST RECORD WRITTEN BEFORE SYSTEM CRASHED.

This message is returned when this record is read following system startup.

FGETINFO

The value returned in RECSIZE will indicate the user's data record size, and the value returned in EOF will indicate the number of data records, unless an FCONTROL 46 is in effect. When an FCONTROL 46 is in effect, the value returned in RECSIZE will be the size of the user's data records, including the two-word header; the number of records returned in EOF will include open, close and data records.

The value returned in BLKSIZE reflects the actual blocksize of the file. When the file is created, the blocksize is computed by the following algorithm:

BLOCKSIZE := ((RECORDSIZE + 2) * BLOCKING FACTOR) + 2

where RECORDSIZE and BLOCKSIZE are in words. For example, with a recordsize of 100 words and a blocking factor of 10, the blocksize would be 1022 words.

FFILEINFO

Two values for ITEMVALUE are specifically for use with IPC:

Item #	Type	Description
34	integer	The current number of writers.
35	integer	The current number of readers.

When you are using IPC, certain intrinsics are not allowed. These intrinsics are FPOINT, FREADDIR, FREADSEEK, FSPACE, FUPDATE, and FWRITEDIR. The FSETMODE intrinsic is ignored.

CIRCULAR FILES

Circular files are wrap-around structures which behave as standard sequential files until they are full. As records are written to a circular file, they are appended to the tail of the file; when the file is filled, the next record added causes the block at the head of the file to be deleted and all other blocks to be logically shifted toward the head of the file. Circular files may not be simultaneously accessed by both readers and writers. When the file has been closed by all writers, it may be read; a reader takes records from the circular file one at a time, starting at the head of the file.

Circular files are particularly useful as history files, when a user is interested in the information recently written to the file and is less concerned about earlier material that has been deleted. These history files are frequently used as debugging tools: diagnostic information may be written to the file, and the most recent and relevant material can be saved and studied.

Creating a circular file is similar to creating a message file. When a user process opens a new file and indicates in the AOPTIONs that it will be a circular file, the FOPEN intrinsic creates the new circular file. In order to create a circular file with the BUILD command, use the CIR keyword; for example, to build a circular file named CIRCLE, enter:

```
:BUILD CIRCLE; CIR
```

A new circular file may also be created with a FILE command. Use the CIR keyword for a new file:

```
:FILE ROUND, NEW; CIR
```

A circular file named ROUND is created.

When you perform a LISTF,2 command, circular files will be identified by an "O" in the TYP field; CIRCLE is identified here:

FILENAME	CODE	LOGICAL RECORD					SPACE		
		SIZE	TYP	EOF	LIMIT	R/B	SECTORS	#X	MX
CIRCLE		128W	FBO	0	1023	1	128	1	8

FEATURES OF INTRINSICS FOR CIRCULAR FILES

Most intrinsics treat circular files the same way they treat regular disc files, but some have special features which apply specifically to circular files; most of these features are found in FOPEN, but a few other intrinsics are also affected.

Parameters that are omitted in the following descriptions retain their normal range of values and their normal default values.

FOPEN

FOPTIONS: (2:3) - File type. Determines the type of file to create for a new file. If the file is old, this field is ignored.

0 - Ordinary file

1 - KSAM file

2 - Relative I/O file

4 - Circular file

6 - Message file

AOPTIONS: (5:2) - Multiaccess mode. This feature permits processes located in different jobs or sessions to open the same file.

0 - No multiaccess. For a writer, this value is changed to a 2 for global multiaccess.

1 - Only intra-job multiaccess allowed.

2 - Inter-job multiaccess allowed; this is the same as specifying the GMULTI option in a FILE command.

3 - Undefined. If this is specified, the FOPEN will be rejected with an error code of 40: ACCESS VIOLATION.

(7:1) - Inhibit buffering. For write access to circular files, this bit is always set off. Readers may open a circular file with NOBUF if they are in copy mode; this determines whether they will be reading the file record by record or block by block.

(8:2) - Exclusive. The values for this field are the same as for any disc file, but they have different meanings for the readers and writers of a circular file:

USER VALUE	Changed to: READER	WRITER
EXCLUSIVE	EXCLUSIVE	EXCLUSIVE
SEMI	SHARE	EXCLUSIVE
SHARE	SHARE	SHARE
Default	SHARE	EXCLUSIVE

For readers, SHARE means "allow other readers;" for writers, SHARE means "allow other writers."

(11:1) - Multirecord. When a reader specifies this option, the file will be accessed NOBUF; for writers, this bit is set to zero.

(12:4) - Access type. These bits specify whether the user will be a reader or a writer process.

0000 - READ access only. The FWRITE, FWRITEDIR, and FUPDATE intrinsic calls cannot reference this file.

0001 - WRITE access only. If this is the first accessor to the file, then the file's contents are purged. If this is not the first accessor to the file, the access type is set to APPEND.

0010 - WRITE SAVE access. Set to APPEND access.

0011 - APPEND access only. The FREAD, FREADDIR, FUPDATE, FPOINT, FSPACE, FWRITEDIR, and FREADSEEK intrinsics cannot reference this file.

Note: Circular files allow variable-length records with append access.

Any other access types are invalid.

FILESIZE: The number of records is rounded up to completely fill the last block.

FWRITE

This intrinsic logically appends the user's record to the end of the file. If the file is full, the first block is deleted, the remaining blocks are logically shifted to the file's head, and the new record is appended to the end of the file.

FCLOSE

For circular files, deletion of disc space beyond the end-of-file is not allowed.

Several intrinsics are not allowed when circular files are used. The FUPDATE, FWRITEDIR, and FDELETE intrinsics are not permitted, and will return a CCL condition if they are used. FPOINT and FSPACE are allowed only for read access.

EXAMPLES

The following programs illustrate the use of IPC via message files. Intrinsics called within the programs manipulate the message files to produce a unidirectional flow of information.

In these two programs, the first is sending information to the second through a message file: it reads data from a data file and writes it to MSGFILE2; the second program can then read this data from MSGFILE2.

```
$CONTROL USLINIT

<< Purpose: >>
<< Read data from a data file and send to another process. >>

BEGIN
  LOGICAL EOF := FALSE;
  INTEGER DATA'FILE, LEN, PIN, IN'FILE, OUT'FILE;
  BYTE ARRAY IN'FILE'NAME (0:8) := "MSGFILE1 ";
  BYTE ARRAY OUT'FILE'NAME (0:8) := "MSGFILE2 ";
  BYTE ARRAY DATA'FILE'NAME (0:8) := "DATA ";
  BYTE ARRAY PRINTPROC (0:8) := "PRNTPROC ";
  ARRAY MESSAGE (0:39);

  INTRINSIC ACTIVATE, CREATE, FCLOSE, FOPEN, FREAD, FWRITE,
            QUITPROG, PRINT, READ;

  << Create entries for the message files in the directory: >>

  IN'FILE := FOPEN (IN'FILE'NAME, %30104, %1300);
  IF < THEN QUITPROG (1);
  FCLOSE (IN'FILE, 2, 0);        << Save file as session temporary. >>
  IF < THEN QUITPROG (2);
  OUT'FILE := FOPEN (OUT'FILE'NAME, %30104, %1301);
  IF < THEN QUITPROG (3);
  FCLOSE (OUT'FILE, 2, 0);       << Save file as session temporary. >>
  IF < THEN QUITPROG (4);

  << Create and activate the print process: >>

  CREATE (PRINT'PROC,,PIN);
  IF < THEN QUITPROG (5);
  ACTIVATE (PIN);
  IF <> THEN QUITPROG (6);
```

```
    << Open message file for traffic from print process: >>

    IN'FILE := FOPEN (IN'FILE'NAME, %30106, %1300);
    IF < THEN QUITPROG (7);

    << Open message file for traffic to print process: >>

    OUT'FILE := FOPEN (OUT'FILE'NAME, %30106, %1301);
    IF < THEN QUITPROG (8);

    << Open data input file: >>

    DATA'FILE := FOPEN (DATA'FILE'NAME, %7, 0);
    IF <> THEN QUITPROG (9);

    WHILE NOT EOF DO BEGIN
      LEN := FREAD (DATA'FILE, MESSAGE, -80);
      IF < THEN QUITPROG (10);
      IF > THEN EOF := TRUE
      ELSE BEGIN
        FWRITE (OUT'FILE, MESSAGE, -LEN, 0);
        IF <> THEN QUITPROG (11);
      END;
    END  << WHILE >>;

    FCLOSE (OUT'FILE, 4, 0);          << No more data to send:  EOF >>
    IF < THEN QUITPROG (12);

    FREAD (IN'FILE, MESSAGE, 1);      << Wait for print to finish. >>
    IF < THEN QUITPROG (13);

    FCLOSE (IN'FILE, 4, 0);
    IF < THEN QUITPROG (14);
  END.
```

```
$CONTROL USLINIT

<< Purpose: >>
<< Receive data from other process and print it. >>

BEGIN
  LOGICAL EOF := FALSE;
  INTEGER LEN, IN'FILE, OUT'FILE;

  BYTE ARRAY IN'FILE'NAME (0:8) := "MSGFILE2 ";
  BYTE ARRAY OUT'FILE'NAME (0:8) := "MSGFILE1 ";
  ARRAY MESSAGE (0:39);

  INTRINSIC FCLOSE, FOPEN, FREAD, FWRITE, QUITPROG, PRINT;

  << Open message file for traffic from other process: >>

  IN'FILE := FOPEN (IN'FILE'NAME, %30106, %1300);
  IF < THEN QUITPROG (13);

  << Open message file for traffic to other process: >>

  OUT'FILE := FOPEN (OUT'FILE'NAME, %30106, %1301);
  IF < THEN QUITPROG (14);

  WHILE NOT EOF DO BEGIN
    LEN := FREAD (IN'FILE, MESSAGE, -80);
    IF < THEN QUITPROG (15);
    IF > THEN EOF := TRUE
    ELSE PRINT (MESSAGE, -LEN, 0);
  END  << WHILE >>;

  << Now signal other process; we are done. >>

  FCLOSE (OUT'FILE, 4, 0);
  IF < THEN QUITPROG (16);

  FCLOSE (IN'FILE, 4, 0);
  IF < THEN QUITPROG (17);

END.
```

MORE IPC.

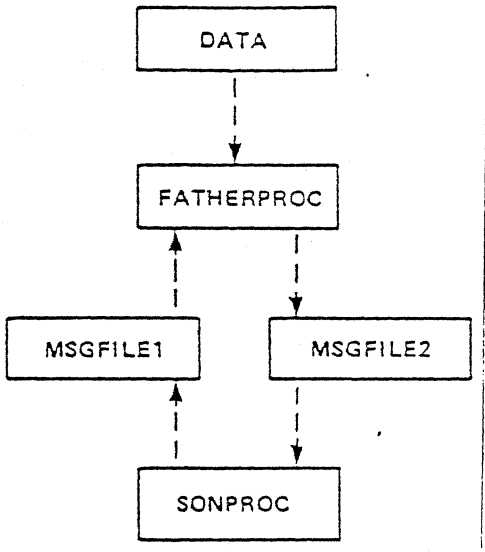


Figure 3-2. Data paths among processes and message files.

```
$CONTROL USLINIT
 IDENTIFICATION DIVISION.
 PROGRAM-ID. FATHERPROC.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. HP3000.
 OBJECT-COMPUTER. HP3000.
 SPECIAL-NAMES.
 CONDITION-CODE IS CC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 DATA-FILE        PIC S9(4) COMP.
 01 LEN              PIC S9(4) COMP.
 01 PIN              PIC S9(4) COMP.
 01 IN-FILE          PIC S9(4) COMP.
 01 OUT-FILE         PIC S9(4) COMP.
 01 IN-FILE-NAME     PIC X(9) VALUE "MSGFILE1 ".
 01 OUT-FILE-NAME    PIC X(9) VALUE "MSGFILE2 ".
 01 DATA-FILE-NAME   PIC X(5) VALUE "DATA ".
 01 PRINTPROC        PIC X(9) VALUE "PRNTPROC ".
 01 MESSAGE-BUF      PIC X(80).
 01 EOF-VAR          PIC X.
    88 EOF           VALUE "E".
* ERROR VARIABLES
 01 ERROR-BUFFER.
    05 FILLER        PIC X OCCURS 1 TO 80 TIMES
                     DEPENDING ON LEN.
 01 ERR-NUM          PIC S9(4) COMP.
 01 FILE-NUM         PIC S9(4) COMP.
 01 QUIT-PARM        PIC S9(4) COMP.
 PROCEDURE DIVISION.
 MAIN PROCESSING SECTION.
```

```
$DEFINE %QUITPROG=                                              QUITPROG
          MOVE !1 TO QUIT-PARM                                  QUITPROG
          MOVE !2 TO FILE-NUM                                   QUITPROG
          PERFORM PRINT-ERROR#                                  QUITPROG
 DRIVER-PARA.
      PERFORM INIT-PARA.
      MOVE "F" TO EOF-VAR.
      PERFORM LOAD-PARA UNTIL EOF.
      PERFORM CLOSE-PARA.
      STOP RUN.
*
* Create entries for the message files in the directory.
*
* Note that IN-FILE-NAME ("MSGFILE1") is opened with FOPTIONs
* %30004:  this indicates a new ASCII message file.
*
 INIT-PARA.
      CALL INTRINSIC "FOPEN"
            USING IN-FILE-NAME %30004
            GIVING IN-FILE.
      IF CC NOT = 0
         %QUITPROG(1#,IN-FILE#).
      CALL INTRINSIC "FCLOSE" USING IN-FILE %2 %0.
      IF CC NOT = 0
         %QUITPROG(2#,IN-FILE#).
*
* Note that OUT-FILE-NAME ("MSGFILE2") is opened with FOPTIONs
* %30004:  this indicates a new ASCII message file.
*
      CALL INTRINSIC "FOPEN"
            USING OUT-FILE-NAME %30004
            GIVING OUT-FILE.
      IF CC NOT = 0
         %QUITPROG(3#,OUT-FILE#).
      CALL INTRINSIC "FCLOSE" USING OUT-FILE %2 %0.
      IF CC NOT = 0
         %QUITPROG(4#,OUT-FILE#).
*
* Create and activate the print process.
*
      CALL INTRINSIC "CREATEPROCESS" USING \ PIN PRINTPROC.
      IF CC NOT = 0
         %QUITPROG(5#,-1#).
*
* Open message file for traffic from print process.
*
* Note that IN-FILE-NAME ("MSGFILE1") is opened with FOPTIONs
* %106 and AOPTIONs %1100:  %106 indicates an old temporary
* ASCII file and %1100 indicates a reader process with exclu-
* sive access and multiaccess capability.  MSGFILE1 has already
* been designated as a message file.  Since only one reader and
* one writer process will be accessing the message file,
* exclusive access mode is specified.
*
```

```
        CALL INTRINSIC "FOPEN"
              USING IN-FILE-NAME %106 %1100
              GIVING IN-FILE.
        IF CC NOT = 0
           %QUITPROG (7#,IN-FILE#).
    *
    * Open message file for traffic to print process.
    *
    * Note that OUT-FILE-NAME ("MSGFILE2") is opened with FOPTIONs
    * %106 and AOPTIONs %1101:  %106 indicates an old temporary
    * ASCII file and %1101 indicates a writer process with exclu-
    * sive access and multiaccess capability.  MSGFILE2 has already
    * been designated as a message file.  Since only one reader and
    * one writer process will be accessing the message file,
    * exclusive access mode is specified.
    *
        CALL INTRINSIC "FOPEN"
              USING OUT-FILE-NAME %106 %1101
              GIVING OUT-FILE.
        IF CC NOT = 0
           %QUITPROG(8# OUT-FILE#).
    *
    * Open data input file.
    *
    * Note that DATA-FILE-NAME ("DATA") is opened with FOPTIONs %3
    * and AOPTIONs 0:  %3 indicates an old permanent or temporary
    * file and 0 indicates read only access.  The file system will
    * change the FOPTIONs to specify an ASCII file.
    *
        CALL INTRINSIC "FOPEN"
              USING DATA-FILE-NAME %3 %0
              GIVING DATA-FILE.
        IF CC NOT = 0
           %QUITPROG(9#,DATA-FILE#).
    *
    * Load input to message file.
    *
     LOAD-PARA.
        CALL INTRINSIC "FREAD"
             USING DATA-FILE MESSAGE-BUF -80
             GIVING LEN.
        IF CC NOT = 0
           IF CC LESS THAN 0 THEN
              %QUITPROG(10#,DATA-FILE#)
           ELSE
              MOVE "E" TO EOF-VAR
        ELSE
           COMPUTE LEN = - LEN
           CALL INTRINSIC "FWRITE"
                 USING OUT-FILE MESSAGE-BUF LEN %0
           IF CC NOT = 0
              %QUITPROG(11#,OUT-FILE#).
     CLOSE-PARA.
        CALL INTRINSIC "FCLOSE" USING OUT-FILE %4 %0.
        IF CC NOT = 0
           %QUITPROG(12#,OUT-FILE#).
```

```
*
* Wait for print to finish.
*
       CALL INTRINSIC "FREAD" USING IN-FILE MESSAGE-BUF %1.
       IF CC < 0
          %QUITPROG(13#,IN-FILE#).
       CALL INTRINSIC "FCLOSE" USING IN-FILE %4 %0.
       IF CC NOT = 0
          %QUITPROG(14#,IN-FILE#).
*
* General error routine.
*
 PRINT-ERROR SECTION.
 WHAT-TYPE.

       IF FILE-NUM IS NOT NEGATIVE THEN
          CALL INTRINSIC "FCHECK" USING FILE-NUM ERR-NUM
          MOVE 80 TO LEN
          CALL INTRINSIC "FERRMSG" USING ERR-NUM ERROR-BUFFER LEN
          DISPLAY ERROR-BUFFER.
       IF QUIT-PARM IS NOT NEGATIVE THEN
          CALL INTRINSIC "QUITPROG" USING QUIT-PARM.

$CONTROL USLINIT
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SONPROC.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. HP3000.
 OBJECT-COMPUTER. HP3000.
 SPECIAL-NAMES.
 CONDITION-CODE IS CC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 LEN              PIC S9(4) COMP.
 01 IN-FILE          PIC S9(4) COMP.
 01 OUT-FILE         PIC S9(4) COMP.
 01 IN-FILE-NAME     PIC X(9) VALUE "MSGFILE2 ".
 01 OUT-FILE-NAME    PIC X(9) VALUE "MSGFILE1 ".
 01 MESSAGE-BUF      PIC X(80).
 01 EOF-VAR          PIC X.
    88 EOF           VALUE "E".
* Error variables.
 01 ERROR-BUFFER.
    05 FILLER        PIC X OCCURS 1 TO 80 TIMES
                     DEPENDING ON LEN.
 01 ERR-NUM          PIC S9(4) COMP.
 01 FILE-NUM         PIC S9(4) COMP.
 01 QUIT-PARM        PIC S9(4) COMP.
 PROCEDURE DIVISION.
 MAIN-PROCESSING SECTION.
$DEFINE %QUITPROG=                                          QUITPROG
       MOVE !1 TO QUIT-PARM                                 QUITPROG
       MOVE !2 TO FILE-NUM                                  QUITPROG
       PERFORM PRINT-ERROR#                                 QUITPROG
```

```
 DRIVER-PARA.
      PERFORM OPEN-PARA.
      MOVE "F" TO EOF-VAR.
      PERFORM READ-PARA UNTIL EOF.
      PERFORM CLOSE-PARA.
      STOP RUN.
*
* Open message file for traffic from other process.
*
* Note that IN-FILE-NAME ("MSGFILE2") is opened with FOPTIONs
* %106 and AOPTIONs %1100:  %106 indicates an old temporary
* ASCII file and %1100 indicates a reader process with
* exclusive access and multiaccess capability.  MSGFILE2 has
* already been designated as a message file.  Since only one
* reader and one writer process will be accessing the message
* file, exclusive access mode is specified.
*
 OPEN-PARA.
      CALL INTRINSIC "FOPEN"
           USING IN-FILE-NAME %106 %1100
           GIVING IN-FILE.
      IF CC NOT = 0
         %QUITPROG(15#,IN-FILE#).
*
* Open message file for traffic to other process.
*
* Note that OUT-FILE-NAME ("MSGFILE1") is opened with FOPTIONs
* %106 and AOPTIONs %1101:  %106 indicates an old temporary
* ASCII file and %1101 indicates a writer process with exclu-
* sive access and multiaccess capability.  MSGFILE1 has already
* been designated as a message file.  Since only one reader and
* one writer process will be accessing the message file,
* exclusive access mode is specified.
*
      CALL INTRINSIC "FOPEN"
           USING OUT-FILE-NAME %106 %1101
           GIVING OUT-FILE.
      IF CC NOT = 0
         %QUITPROG(16#,OUT-FILE#).
*
* Read messages from message file.
*
 READ-PARA.
      CALL INTRINSIC "FREAD"
           USING IN-FILE MESSAGE-BUF -80
           GIVING LEN.
      IF CC NOT = 0
         IF CC LESS THAN 0 THEN
            %QUITPROG(17#,IN-FILE#)
         ELSE
            MOVE "E" TO EOF-VAR
*
* Print message out.
```

```
*
      ELSE
         COMPUTE LEN = - LEN
         CALL INTRINSIC "PRINT"
               USING MESSAGE-BUF LEN %0
         IF CC NOT = 0
            %QUITPROG(18#,2#).
*
* Now signal the other process; we are done.
*
 CLOSE-PARA.
      CALL INTRINSIC "FCLOSE" USING OUT-FILE %4 %0.
      IF CC NOT = 0
         %QUITPROG(19#,OUT-FILE#).
      CALL INTRINSIC "FCLOSE" USING IN-FILE %4 %0.
      IF CC NOT = 0
         %QUITPROG(20#,IN-FILE#).
*
* General error routine.
*
 PRINT-ERROR SECTION.
 WHAT-TYPE.

      IF FILE-NUM IS NOT NEGATIVE THEN
         CALL INTRINSIC "FCHECK" USING FILE-NUM ERR-NUM
         MOVE 80 TO LEN
         CALL INTRINSIC "FERRMSG" USING ERR-NUM ERROR-BUFFER LEN
         DISPLAY ERROR-BUFFER.
      IF QUIT-PARM IS NOT NEGATIVE THEN
         CALL INTRINSIC "QUITPROG" USING QUIT-PARM.
```

WHY IPC?

** A COMPARISON OF MESSAGE FILES .VS. CURRENT MPE COMMUNICATION TOOLS

* SINCE THE BEGINNING, MPE HAS OFFERED A VARIETY OF TOOLS TO ALLOW CUSTOMERS TO PERFORM INTERPROCESS COMMUNICATION AND CONTROL. HOWEVER, THESE MECHANISMS ARE TOO RESTRICTIVE TO ALLOW APPLICATIONS TO FULLY TAKE ADVANTAGE OF MPE'S MULTIPROCESS CAPABILITY.

** GENERAL ADVANTAGES OF IPC

* SIMPLICITY:

MSG FILES ARE ACCESSED WITH THE SAME STANDARD FILE SYSTEM INTRINSICS THAT THE CUSTOMER USES TO ACCESS HIS SEQUENTIAL FILES. THE OTHER MECHANISMS EACH HAVE THEIR OWN SET OF INTRINSICS.

A BONUS OF USING THE FILE SYSTEM IS THAT PROGRAMS WRITTEN TO USE STANDARD, SEQUENTIAL FILES MAY HAVE THEIR OPENS REDIRECTED TO MESSAGE FILES. THIS PERMITS CHAINING TOGETHER OF PROGRAMS IN A MANNER SIMILAR TO UNIX. EX: CONNECT THE SPL COMPILER'S NEW FILE WITH THE INPUT FILE TO THE CROSS REFERENCE PROGRAM.

* TIMEOUTS:

THIS OPTIONAL FEATURE PERMITS DETECTION OF DEADLOCKS.

WHY IPC?

* OBITUARY NOTICES:

  WHEN A WRITER TO A MESSAGE FILE EXPIRES, A CLOSE RECORD IS AUTOMATICALLY WRITTEN TO THE FILE. THIS CAN OPTIONALLY BE GIVEN TO THE READER, INFORMING HIM OF HIS PEER'S DEMISE.

* CHOICE OF NOTIFICATION:

  MESSAGE FILE COMPLETIONS CAN BE HANDLED WITH WAIT OR NOWAIT (IOWAIT)I , NEITHER OF WHICH REQUIRE PM.

* APPLICATIONS MAY SPAN COMPUTERS:

  COMBINING MESSAGE FILES WITH DS ALLOWS PROCESSES TO BE REMOTE FROM THEIR FILES AS WELL AS FROM EACH OTHER.

* DEBUGGING:

  PROCESSES CAN BE CERTIFIED BY SUBSTITUTING SCRIPT FILES FOR THEIR MESSAGE FILES.

* FLEXIBILITY:

  MESSAGE FILES' ATTRIBUTES CAN BE CHANGED WITH A SIMPLE FILE COMMAND.

* BROADER SCOPE OF COMMUNICATION:

  ANY PROCESS MAY COMMUNICATE WITH ANY OTHER, SUBJECT TO THE NORMAL FILE SECURITY PROVISIONS.

WHY IPC?

** IPC .VS. MAIL

ATTRIBUTE	MAIL	MSG FILES
SCOPE	FATHER/SON	ANY PROCESS
CAPACITY	ONE MSG	FILE SIZE (POTENTIALLY THOUSANDS OF MSGS)
BIDIRECTIONAL	YES, HOWEVER, ONE MSG CAPACITY FOR BOTH DIRECTIONS	NO, REQUIRES 2 FILES
WAIT	ONLY ON ONE MAIL-BOX	WAIT ON MSG FILES PLUS OTHER FILES (MUST HAVE PM)
AUTO OBITUARY NOTICES	YES	YES, IF READER IS USING EXTENDED READ

WHY IPC?

** PROCESS SIGNALING (ACTIVATE/SUSPEND INTRINSICS) .VS. IPC

ATTRIBUTE	ACTIVATE/SUSPEND	MSG FILES
SCOPE	FATHER/SON	ANY PROCESS
FIND SOURCE OF ACTIVATION?	YES	YES
MULTIPLE ACTIVATIONS SAVED?	NO	YES
INFORMATION PASSED?	NO	YES
WAIT?	ONLY ON ACTIVATE INTRINSIC	WAIT ON MSG FILE PLUS OTHER FILES

WHY IPC?

** RESOURCE ALLOCATION: IPC .VS. RINS

MSG FILES CAN BE USED AS A RESOURCE ALLOCATION MECHANISM BY BUILDING A MSG FILE AND THEN PUTTING ONE RECORD INTO IT (NOTE THAT N RESOURCES CAN BE MANAGED BY PUTTING N RECORDS INTO THE FILE). ALL RELEVANT PROCESSES WOULD THEN OPEN THE FILE WITH READ AND WRITE ACCESS. GETTING A RESOURCE WOULD THEN CONSIST OF READING FROM THE FILE. WRITING THE RECORD BACK TO THE FILE RELEASES THE RESOURCE.

ATTRIBUTE	RINS	MSG FILE
SCOPE	GLOBAL TO SYSTEM	ANY PROCESS
INFO PASSED?	NO	YES
GET "IN LINE" BEFORE SUSPEND?	NO	YES (NOWAIT I/O)
# UNITS MANAGED	ONE	ANY NUMBER
WAIT	ONLY ON RINS	WAIT ON MSG FILE PLUS OTHER FILES
SYSTEM AUTO RELEASE OF RINS ON TERMINATION?	YES	NO

WHY IPC?

THE PRECEDING METHOD MIMICKS THE MPE RIN CONCEPT. MSG FILES ALSO SUPPORT A TOTALLY DIFFERENT APPROACH TO RESOURCE-MANAGEMENT. THIS CONSISTS OF ALLOWING ONLY ONE "EXPERT PROCESS" (CALLED AN OBJECT MANAGER) TO ACTUALLY ACCESS THE RESOURCE. PROCESSES DESIRING TO USE THE OBJECT DO SO BY SENDING MESSAGES TO THE OBJECT MANAGER. AT THE COMPLETION OF THE COMMAND, THE MANAGER MAY ELECT TO SEND A STATUS MESSAGE BACK TO THE REQUESTOR.

Message Files

Uses the file system to allow interprocess communications between any set of processes.

Basics

*Msg files function as a FIFO queue.
 .msgs stored in file until read
 .msgs deleted as they are read
 .unidirectional msg flow through the file

*Boundary conditions.
 .Empty file - has writer, reader is waited; no writer, CCG
 .Full file - has reader, writer is waited; no writer, CCG

File characteristics

*Record format
 var, fixed ascii, binary cctl blocking factor

*1-32 extents user labels file code

*New job temporary permanent

*Access options (default shown first)
 .gmulti/multi wait/nowait in/out/outkeep/append
  nocopy/copy
 .Exclusive field (R=allow reader, W=allow writer)
  exc W/R semi (ear) Ws/R shr Ws/Rs

*Security
 .input requires read/write access
 .output requires append access

Commands

Msg files must be created as msg files with either BUILD or FILE commands.

BUILD JOE;MSG FILE JOE,NEW;MSG

FILE COMMAND - STD/MSG COPY/NOCOPY MULTI/GMULTI EAR/SEMI

LISTF command - Msg file has an "M" suffix in the type field.

Intrinsics

*Intrinsics allowed
FOPEN, FCLOSE, FREAD, FWRITE, FCONTROL, FCHECK, FGETINFO, FFILEINFO, FLOCK, FUNLOCK, FREADLABEL, FWRITELABEL, FRELATE, FSETMODE.

*Foptions
(2:3) - file type. 0-STD, 1-KSAM, 2-RIO, 3-MSG, 4-CIR

*Aoptions
Multirecord, inhibit buf set off
(5:2) - multiaccess.
0-set to 2; 1-job local; 2-any process; 3-undefined
(4:1) - 0-wait I/O; 1-nowait I/O
(3:1) - 0-normal access; 1-copy mode

*Control codes

code	parm	description
2	-	Complete I/O, ignored.
3	-	Read hardware status word.
4	0	Disable timeout.
	<> 0	Set timeout interval in seconds.
6	-	Write EOF, writes buffers and label to disc.
43	-	Abort nowait I/O.
45	false	Read against an empty, writerless file returns CCG.
	true	Read against an empty, writerless file is waited. This control also applies to the writer case.
46	false	Read only data portion of writers' data records.
	true	Read writer open, close, and data records. Prefix: word 0 - rec type (0-data, 1-open, 2-close) word 1 - writer ID word 2-n - data (only for data records)
47	false	Next read will delete the first record.
	true	Next read will not delete the record.

*FFILEINFO returns

item	type	description
34	int	Current number of writers
35	int	Current number of readers

```
0(   ) 1    <<
00000 1
00000 1    Message File Access Procedures
00000 1    ------------------------------------
00000 1
00000 1    The IPC code spans three areas.
00000 1
00000 1    1. File System Code.
00000 1       Since msg files have their own file type (and, hence, their
00000 1       own AFT type), it is necessary to merely add a new statement
00000 1       to each of the intrinsics' file-type case statement.  For most of
00000 1       the intrinsics this statement is very simple, consisting of either a
00000 1       call to a msg access procedure (read, write, and control) or
00000 1       an error (point, readseek, etc).  Only the FOPEN intrinsic
00000 1       requires substantial modification to support msg files.
00000 1
00000 1    2. Access Procedures.
00000 1       This is implemented by the code of this listing.
00000 1
00000 1    3. Basic IPC.
00000 1       This code provides the wait/wakeup mechanism.
00000 1
00000 1
00000 1    .............       .............       .............
00000 1    :           :       :           :       :           :
00000 1    : File Syst :       : Access    :       : Basic IPC :
0(   0 1   : Callable  :------>: Procedures :------->:           :
00000 1    : Intrinsics:       :           :       .............
00000 1    :           :       :           :
00000 1    .............       :           :--     .............
00000 1                        :           :  --   :           :
00000 1                        .............    -- : File System :
00000 1                                       ->: Uncallable  :
00000 1                                         : Procedures  :
00000 1                                         :           :
00000 1                                         .............
00000 1
00000 1    The remainder of this section discusses the access procedures.
00000 1
00000 1
00000 1    This set of procedures is the equivalent of FREAD's/FWRITE's IOMOVE
00000 1    for message files.  It is the heart of the Msg file implementation.
00000 1    The Msg file access method is more complex than that of regular
00000 1    sequential files for the following reasons.
00000 1
00000 1      1. A wait/wakeup mechanism is required for the "read from an
00000 1         empty file" and "write to a full file" cases.  This
00000 1         mechanism must support timeouts.
00000 1
00000 1      2. Like circular files, the disc file eof can wrap around the
00000 1         extent map.
00000 1
0    0 1     3. Both the reader and the writer(s) must use buffers in a
00000 1         coordinated and efficient manner.
00000 1
0000         4. Disc space extents may be thrown away at close time after
0000 IPC.129    they have been read providing there are no current writers.
```

00000 1 Design highlights.
00000 1 ----------
00000 1
00000 1 1. File Boundary Conditions
00000 1
00000 1 There are two cases when a process is forced to wait on an
00000 1 action from one or more other processes.
00000 1
00000 1 a) A reader accessing an empty file, the process must
00000 1 wait until a writer writes a record.
00000 1
00000 1 b) A writer encountering a full file, the process must wait
00000 1 until a reader reads enough records on disc to free a
00000 1 physical block.
00000 1
00000 1 a) Impeded reader
00000 1
00000 1 This has a classic, straightforward solution. The reader
00000 1 decrements a counting semaphore (number of records in the
00000 1 file). When the semaphore goes to zero there are no more
00000 1 records left and the reader is placed on the wait queue
00000 1 (which is a disabled port--see basic IPC description).
00000 1 Each time a writer writes a record, the writer checks the
00000 1 wait queue. If it is nonempty, the head entry is taken
00000 1 from the queue and sent to the reader's reply port.
00000 1 Otherwise the counting semaphore is incremented.
00000 1
00000 1 This mechanism is gummed up somewhat by extended read mode.
00000 1 If not in extended read then only data records count in the
00000 1 counting semaphore. Extended read causes all records (data,
00000 1 open, and close) to be counted.
00000 1
00000 1 b) Impeded writer
00000 1
00000 1 When the file is opened the amount of free file space (in
00000 1 max-sized records) is calculated. Each write first subtracts
00000 1 its record size and record overhead from the free space. If
00000 1 there is not enough space the writer is placed on the write
00000 1 wait queue.
00000 1
00000 1 The reader returns free space a block's worth at a time.
00000 1 The wait queue is then reduced until the freed bytes
00000 1 have been filled. The actual write is performed by the
00000 1 reader at this time. This is to 1) expedite the no wait
00000 1 writers' data and 2) to insure that the released writers
00000 1 will fill the file in the same order as they were freed
00000 1
00000 1 Both readers and writers have "reply ports" which receive
00000 1 the completed read/write requests (refer to the basic IPC
00000 1 documentation for a description of ports). The current
00000 1 limitation of IOWAIT is one request outstanding, so the
00000 1 reply ports will only have a maximum queue depth of one
00000 1 entry.
00000 1
00000 1
00000 1 2. Disc Space Management
00000 1

Since the reader logically destroys records as they are read, it it desireable to also physically delete them as well. The unit of allocation on the disc is the extent. The spent extents are only deleted at close time when there are no current writers. To delete extents while there are writers can lead to needless work where the reader is deleting old extents and the writer is obtaining new ones.

3. Buffering

Both reader and writer buffers are contained in the same buffer pool. Reader buffers are the "head" of the queue whereas writer buffers comprise the "tail" of the queue. The buffer management runs in one of two modes.

a) Coupled Mode.

All records are contained in the buffers. A spent read buffer is added to the tail of the write buffer list. Writers add to tail buffer in the list. If this happens to be a read buffer, then no disc I/O is initiated. Note that if all the accessors were to close then all data would be written to the disc. When the number of records can no longer be held in the buffers, uncoupled mode is entered.

b) Uncoupled Mode.

The records cannot be contained in the buffer area. In this case the reader and writers use their own buffers independently. Note that if the reader reads sufficient records such that the records do fit in the buffers, the buffering mode will automatically revert to coupled mode.

4. Record formatting.

Data records are written in standard variable length format. Records do not span blocks. Data record headers and open and close records are written starting from the bottom of the buffer and working up (whereas data records' data are written starting from the beginning of the buffer).

This method makes it possible to use the normal file system access method when accessing the file in "copy" mode. When a new record and its header can no longer fit into the buffer, the buffer's block is written to disc and the record and its header are placed into the next block.

5. Miscellaneous

a) No wait FREADs and FWRITEs are waited for disc I/O. To do disc I/O without wait brings in additional complexity for very little gain to the user. Implicit disc I/Os

with wait have probably already occurred to make present the ACB and file system code.

b) Anticipatory reading is done. However a write buffer is not written until the current record will not fit into it. Having two or more write buffers should keep the writers waiting on disc I/O to a minimum.

c) A bit map is kept for all currently allocated writer IDs. It is managed by the GETID and RELID procedures. Currently a maximum of 256 concurrent writers is allowed.

d) A disc error encountered by a reader prevents any more reads.

e) A disc error encountered by a writer prevents any more writes.

f) Unless otherwise noted, DB is always set to the stack upon entry/exit to these procedures.

g) There are two wait queues, one for readers and one for writers. It is possible that they could both be nonempty. Say the file has a capacity of n records. N records are written, the n+1 write is queued up. Then n nowait reads are done with no complementary IOWAITs or IODONTWAITs being issued. The next read will be queued.

6. Optimization.

a. "Immediate" nowait I/O completion.

When it has been determined at FREAD/FWRITE time that the request will not have to wait on a boundary condition, reply ports are not used. Instead a two word "message area" in the LACB is used to house the completion message. A special pattern is placed into the AFT reply port word. When IOWAIT (or FINDWAITINGIO) sees this pattern it knows that the request has completed.

b. Updating the FCB.

Everytime that a reader deletes a block it updates the start-of-file field in the FCB. This operation has been optimized such that when the reader knows that the SOF is already at zero, the FCB update is skipped.

How does the reader know? The writer sets a flag when it enters "uncoupled" buffer mode. This indicates that the writer has written to the disc which will ultimately cause the SOF to be incremented (as the blocks are read). Thus when this flag is set, the reader has to update the SOF in the FCB.

c. Semi-Exclusive Mode.

In semi-exclusive mode there may be at most one reader. The generality of a read wait queue is not needed. Instead when the reader attempts to read from the empty file, the

00000 1 reader's reply port number is placed directly into the ACB.
00000 1 If the reader has opened the file with wait, then no reply
00000 1 port is used. In this case the reader's PIN is saved in
00000 1 the ACB.
00000 1
00000 1 The above short cuts are also applied to writers when the
00000 1 file is opened exclusively (ie, there may only be one
00000 1 writer).
00000 1
00000 1 d. Delayed Awake.
00000 1
00000 1 When a reader or writer liberates a symbiotic process
00000 1 which was waiting on a boundary condition, the actual call
00000 1 to AWAKE is delayed until after the file's ACB has been
00000 1 released.

g. An end-of-file is written to checkpoint the file.

3. Writing.

The usual course of a write is to 1) insure that there is room for the record, 2) write the record (usually just a data move to an ACB buffer), and 3) if a reader is waiting, to send a successful message to the reader's reply port.

The following exceptions may occur:

a. First write.

The writer's first write after an open consists of writing the open record, the data record, and allocating space for the close record.

b. Stalled writer.

If the record will not fit into the file then the writer is waited if:

1) a reader has opened the file,

2) or this is the writer's first write after the open,

3) or the writer has specified extended wait.

Waiting consists of placing a message on the wait queue (a basic IPC port). Eventually a reader will perform the transfer and send a successful message to the writer.

c. Block indexing.

If the record will not fit into the current block then

1) if the current buffer is a write buffer, a disc write is initiated on behalf of the buffer,

2) if this current buffer is a read buffer (coupled mode) then the next buffer is chosen. Otherwise the first write buffer becomes the last (current) write buffer. Thus write buffers contain the last n blocks of the file. This is useful information to have when interpreting dumps.

The write facility has been divided into two procedures.

FCWRITE - Decides if there is room to write the record, waits the writer if not, performs user checks.

PUTRECORD - Actually performs the write, including indexing to the next block, if necessary.

PUTRECORD is also called by the liberating reader on behalf of a waiting writer.

```
0000 1
0000 1       Note - If the writer's target area is in his stack, then the
0000 1              writer's stack, the reader's stack, and the ACB must all
0000 1              be present for the move.  If the code is changed to have
0000 1              wait writers perform their own move, they must first check
0000 1              to see if there is room (after they wake up).  A speedy
0000 1              second writer could have taken the space away.
0000 1
0000 1
0000 1    4. Reading.
0000 1
0000 1       Most read requests read a record and, if the next record is in
0000 1       another block, issue an anticipatory read of the block from disc.
0000 1
0000 1       The following exceptions may occur:
0000 1
0000 1       a. Extended read mode.
0000 1
0000 1          If the file is not in extended read mode, then only data
0000 1          records are desired.  Thus possible open/close records must
0000 1          be flushed from the front of the file before the actual read
0000 1          to the user's target area.  If the reader must wait (due to
0000 1          an empty file) then when the reader is awakened by the
0000 1          writer, the flushing process must be repeated before the
0000 1          actual read can commence.
0000 1
0000 1       b. Empty file.
0000 1
0000 1          The reader is waited if:
0000 1
0000 1          1. one or more writers has opened the file,
0000 1
0000 1          2. or this is the reader's first read to the file after
0000 1             the open,
0000 1
0000 1          3. or the reader has specified extended wait.
0000 1
0000 1          Waiting consists of placing a mesage on the wait queue
0000 1          (a basic IPC port).  Eventually a writer will deposit a record
0000 1          and send a successful message to the reader's reply port.  The
0000 1          reader then (or at IOWAIT time) performs the data transfer.
0000 1
0000 1          Waited readers, unlike waited writers, perform their own data
0000 1          movement upon being liberated.  They can do this because they
0000 1          are freed with a claim on any one record in the file - not a
0000 1          particular one.  Thus it is permissable for other readers to
0000 1          issue FREAD/IOWAITs between another reader's FREAD and IOWAIT.
0000 1          The only constraint is that a one record (claim) be set aside
0000 1          for the first reader when he is liberated from the wait queue.
0000 1
0000 1       c. Block indexing.
0000 1
0000 1          If the record read is the last record in the block then:
0000 1
0000 1          1. A block's worth of max-sized records is added to the file's
0000 1             free space.  It is immediately applied to any waiting writers
0000 1             on the wait queue.
```

2. The file's start-of-file is advanced by one and all block numbers are decremented. Exception: if there are no more records in the file then the start-of-file and end-of-file are both reset to zero.

3. If the file is not empty then the next block is obtained as per the buffering description in "Design Highlights."

```
000 0   Begin
000 1   equate
000 1     version                              = 00,  <<05/22/80>>
000 1     update                               = 00;  <<05/22/80>>
000 1
000 1
000 1   <<Compile option
000 1
000 1     X0  - on,  Call DEBUG before calling SUDDENDEATH
000 1         - off, Call SUDDENDEATH directly>>
000 1
000 1
000 1   <<
000 1   Communications-File Basic IPC Mechanism
000 1   ------------------------------------
000 1
000 1
000 1   The objective of this set of uncallable intrinsics is to provide
000 1   a simple mechanism that will enable one process to send short
000 1   messages to another process.
000 1
000 1   The heart of this mechanism is the port.  A process desiring to
000 1   receive messages would first open (create) a port.  This process
000 1   is termed the "port manager."  When the port is created, a port
000 1   number is returned to the opener.  There is no provision to
000 1   rendezvous with known "logical" port names so potential senders
000 1   need some method of obtaining the port number from the port
000 1   manager.
000 1
000 1   Both the ports and the messages are contained in a disc resident
000 1   data segment.  There can be a total of over thiry-five hundred
000 1   open ports and outstanding messages  Thus neither ports nor
000 1   message blocks are scarce resources.
000 1
000 1
000 1   A return port is explicitly passed in FCPORTSEND so that messages
000 1   that timeout can be given back to a port belonging to the
000 1   originator.
000 1
000 1   A process desiring some service from another process would use
000 1   this mechanism in the following manner.
000 1
000 1     1. An FCPORTSEND, specifying a return port, is issued to the
000 1        proper port.  The message's parameters specify the service
000 1        to be performed.  This results in an MQE's (message queue
000 1        entry) being obtained, initialized, and queued to the tail
000 1        of the port.
000 1
000 1     2. At some later point in time the process decides to
000 1        synchronize with the request by issuing an FCPORTRECEIVE
000 1        with wait against its own return port.
000 1
000 1     3. The port's manager process steps through the messages on
000 1        the port, eventually processing the MQE used in steps 1 and
000 1        2.  This process awakens the first process by sending a
000 1        return message to the original process's return port.
000 1
```

00000 1 Note that steps 2 and 3 could have occurred in reverse
00000 1 order.
00000 1
00000 1 4. When the first process runs, it is still in FCPORTRECEIVE
00000 1 which places the second MQE on the free list and returns to
00000 1 the process's calling procedure.
00000 1
00000 1 Notes:
00000 1
00000 1 1) This set of procedures is very closely keyed to the needs
00000 1 of the MSG file access method. Any other use of this code
00000 1 is prohibited since these procedures may be obsoleted by
00000 1 future MPE message facilities.
00000 1
00000 1 2) Unless otherwise noted, these procedures require DB to be at
00000 1 the stack.
00000 1
00000 1 3) Software interrupts are not yet implemented.
00000 1 >>

Draft for 3000 User's Group : March 11, 1981

The MPE IV Kernel : History, Structure and Strategies

John R. Busch
Member of the Technical Staff
Hewlett Packard Corporation
Computer Systems Division
19447 Pruneridge Avenue
Cupertino, California
95014

Abstract

The MPE IV kernel is the result of over three years of research and development undertaken at Hewlett Packard's HP 3000 R&D lab in Cupertino. It provides a new high performance, integrated, extensible foundation for the 3000 operating system, MPE. The project's history and the kernel's characteristics are described. Project objectives, investigation approach, implementation methodology, functional characteristics, resource management objectives and strategies, and performance results are presented.

1. Introduction

The evolution of the HP 3000 family towards large main memories, fast processors, and large and distributed configurations stressed the original MPE kernel design and implementation. It became clear that just supporting the evolution in terms of kernel data structure extensibility would become a problem. Moreover, the original algorithms could not be relied upon to exploit the performance potential offered by the larger configurations.

Project objectives for a new kernel were established. Research into the growth and performance limitations of the old kernel and into state of the art approaches to resource management policies was undertaken. Alternative designs were established, implemented and evaluated.

This process, culminating in the MPE IV C-Mit, is presented in the following sections.

2. Kernel Project Objectives

The primary project objectives for the MPE IV kernel were to provide :

(1). support for the evolution of the HP 3000 family;

(2). maximum performance across the family members;

(3). high reliability and improved fault detection and recovery;

(4). increased functionality as required by the other system components of MPE IV; and

(5). simple extensibility when unforeseen system requirements surface.

3. The Investigation

The 3000 architecture, workload, and evolution were to be matched by the new kernel.

The investigation would procede by : identifying the characteristics of the workload; determining the growth limitations and performance problems of the existing kernel; researching and consulting to determine promising approaches to kernel design; and formulating alternative designs.

Instrumentation was placed into the old kernel, and the system was measured under reproducible, representative environments.

Service requirements induced by the workload on the various system servers were determined. Distributions of segment sizes, processing requirements, and access requirements to secondary store were observed.

Resource utilization was measured. Disc, main memory and processor queue lengths, request type distributions, and overall utilization were determined.

Migration among the servers and service delays were characterized. Process stop type (eg segment fault, disc I/O, terminal I/O, time sliced, etc.) distributions and service delays for each stop type were measured.

These measurements were to be used not only in isolating invariant workload characteristics and performance problems but would also be used as the base for later comparisons. (The specific growth and performance limitations of the old kernel are addressed in later sections).

The literature was researched and academicians consulted to ensure that the lessons of the past and the academic investigations whose results offered potential were taken into account. Although extensive literature was available on resource management policies, very limited literature was to be found which directly related to the segmented architecture of the 3000 family. What was acquired from the research and consulting was a set of principles, measures and ideas which could be incorporated into the design and implementation.

The investigation phase resulted in an understanding of the problems and limitations of the old kernel, and a set of alternative strategies which eliminated the limitations and problems and offered potential in satisfying the overall objectives. Rather than coming up with the

eventual design, the investigation came up with a commitment to try out the alternatives and select the best strategies for the architecture and environment based on measurement rather than intuition.

4. Performance Goals and Strategy

In this paper, a transaction is considered to be a step in an interactive session which begins when carriage return or enter is hit and terminates when the system is ready to accept further input form the session.

The global performance indices for the intended application environment are :

(1). transaction response time;

(2). transaction throughput;

(3). fairness;

(4). batch throughput.

The desired system behavior is as follows :

- For a given workload and configuration, the system should provide minimum transaction response time with maximum transaction throughput. Batch performance should be "acceptable."

- Under increasing load, the system should be stable. As the load increases, transaction response time should degrade gradually and fairly. System throughput should stay high even under very heavy loads.

- The system should dynamically tune itself to optimize performance for the current workload with the given configuration. However, explicit control over the relative service between transactions and between interactive and batch should be available to the operator and system manager.

It must be kept in mind that the bottom line performance of the system is measured by the global performance indices and not by the factors which may influence them. This suggests that the performance strategy should be directed towards optimizing the global indices and not towards optimizing indices local to each system component.

The selected overall performance strategy was to achieve maximum system performance by having the system components cooperate to optimize the global performance indices. This approach is fundamentally different from having each system component attempt local optimization and hoping the result will be good overall performance.

Measures and associated instrumentation were defined for the global and local performance indices and supported by the measurement interface. With these measures, the effects of alternative strategies could be understood and evaluated. The measurement interface through performance tools would be made available in the field so that on-site trouble shooting and tuning could be performed.

5. Implementation Methodology

In order to achieve the desired high reliability and natural extensibility, the implementation would have to be highly structured.

Interfaces between system components would be explicit, general, and adhered to. Access to kernel services and internal information would be available only through the use of explicit messages or the invocation of kernel interface intrinsics. An adequate set of interface intrinsics and a general, efficient internal message system would be required to support this structured interfacing.

Within the kernel, a structured implementation was absolutely necessary so that alternative resource management policies could easily be incorporated, coexist, and eventually be deleted.

Performance considerations at the instruction level would be of secondary concern in favor of a structured implementation. The sought after high level of system performance would be achieved through integrated, parallel policies rather than by relying on highly optimized code sequences.

The algorithm selection process and the support of MPE IV performance tools would require that complete instrumentation and an instrumentation interface be carefully designed into the new kernel.

6. The Internal Message Facility

A high speed memory resident message facility was defined and implemented. The facility is intended for the transmission of operating system status and control messages. This facility eliminates the need for supporting multiple ad hoc communication mechanisms.

The message facility associates a message harbor with each process. Each message harbor contains 32 message ports. Each message port contains a FIFO queue of messages, where a message is up to 5 words in length (maximum length is configurable).

Message intrinsics are provided to send a message to any port of any process, to determine the status of any or all message ports of a process, and to receive in a destructive or non-destructive manner the message at the head of a specified message port.

Use of the internal message facility is limited to operating system code. User level inter-process communication is available through MPE IV message files.

7. The Measurement Interface

In order to evaluate alternative strategies and to support the envisioned and yet to be envisioned MPE performance tools, an extensible measurement interface was designed and implemented.

The existing MPE measurement tools were highly dependent on the kernel implementation. They were knowledgeable of internal data structures and called very low level kernel routines or exerted direct control over resource management. Modifying the tools to support the new kernel would be an inadequate solution since the tools were inadequate for the evaluation task at hand, and future changes would create the same problem over again. The decision was made to attempt to centralize support of measurement requirements within the kernel itself, and to make the tools independent of the kernel's implementation.

The basic requirements of the existing and envisioned tools were investigated. An interface was defined which would provide the mechanisms, support structures, and the access and control intrinsics so that the information needed would be obtainable through intrinsic calls without knowledge of internal structures or policies.

The key objectives of the interface were : service (provide what's needed by the current tools); transparency (eliminate dependencies of performance tools on system internals); extensibility (meet future requirements by natural extension of the initial specification); and low overhead (so that use of the interface would minimally effect the performance of the system under test).

The resulting measurement interface supports complete system global and process local statistics gathering, selective measurement class enabling/disabling intrinsics, statistics class delivery intrinsics, and complete cleanup upon abnormal termination of a process which had enabled statistics gathering. Tools using the measurement interface require no privileged code so that system reliability is improved and the sought after independency from the kernel implementation is achieved.

8. MPE IV Kernel Resource Managers

Each resource manager operates independently through clean interfaces so that strategy or data structure changes of another resource manager will not effect him. Each resource manager is built from structured, general pieces so that alternative strategies can be easily implemented.

The management of the disc, main memory, and processor resources has the primary impact on system performance. The approaches taken towards resource management for these key resources is sketched in the remainder of this section.

8.1 Disc Management

Disc management policies have an extremely significant effect on system performance due to the workload characteristics. Transaction processing applications on the 3000 are characterized by several disc references per transaction with short processing requirements between references. In such an environment, good disc management is essential in achieving good system performance.

The goal of disc management should be to provide maximum disc subsystem throughput while minimizing the service time for the most important requests. The selection of policies for the management of disc space and the scheduling of accesses to secondary store should be based on achieving this goal.

The disc management systemi nterfaces with the file system and the memory management system when allocating disc space and servicing requests to access secondary store.

The major problems identified with the old disc management policies revolved around virtual memory management, disc access scheduling, and serial seeking.

Disc space for data segments was restricted to a single volume (i.e. virtual memory limited to the system disc). This restriction has serious detrimental effects on system growth and performance. The effect on system growth is the obvious limitation on the amount of disc space available for data segments by the size of the system disc. The performance impact of this restriction is due to the long queue length created at the system disc when the system is under memory pressure, and the resultant service delays for access requests to that volume.

Disc access scheduling did not perform requests in the order of their urgency. The scheduling policy for disc requests directed at a device was preemptive for all memory management requests and FIFO for all other requests.

The memory management replacement policy selected segments deemed not likely to be needed in the near future. In the case of data segments, a write to disc of the segment was requested, the motivation being that the write may complete before the region occupied by the segment was required so that the delay in fetching the new segment would be reduced. These "anticipatory writes" were not urgent and often unnecessary, yet the scheduling policy selected them for service before disc access requests required for the completion of important transactions. Segment fetches on behalf of batch jobs were also serviced before transaction

related service requests under the old scheduling policy.

The FIFO policy within process initiated disc access requests resulted in the accesses of less urgent processes being performed before those of more urgent processes. This increased the service time for the more urgent processes requests and thereby increased the response time for the related transactions.

Disc service was entirely serial for each disc sharing a common controller (i.e. no ovelapping.) Although the controller supports overlapped seeks, this feature was not exploited. This resulted in a disc throughput limitation per controller to one access per (avg cylinder positioning delay + avg rotational latency + avg transfer time).

MPE IV disc management solves these problems. Additionally, the general approach to disc management gives broad flexibility in scheduling policies.

In MPE IV, disc space for data segments can reside on each system volume. This multi-spindle virtual memory eliminates the limitation on total virtual disc space, and helps to balance the disc queue lengths. (Balanced disc queues are required to take advantage of parallelism in I/O offered by overlapped seeks or multiple controllers).

The MPE IV disc queues are priority ordered. The priority of a disc request is determined by the priority of the process that requires the transfer. This holds for segment transfer requests issued by the memory management system on behalf of a process as well as for file system initiated transfer requests. Anticipatory writes are given the worst priority and sit at the back of the queue so that they are performed as background activity when the device would be otherwise idle. This priority queue management integrates the disc management policies with the goals of the rest of the kernel, since priority assignments reflect the global performance goal of the system.

The feature of the disc controller which allows a seek command to be sent to a unit other than the unit owning the controller is exploited in MPE IV. The seeks for units waiting for the controller are issued during the execution of the channel program for the unit currently owning the controller. This results in the heads being in position over the proper cylinder when the next unit gets the controller. The net result is a potential maximum disc throughput per controller of 1 access per (avg rotational latency + avg transfer time). Since the disc 1 access time is dominated by the head positioning delay, this overlapping approximately doubles the maximum throughput per controller. (The overlapping seek software will only be available for the Series II and III on the C-MIT).

To achieve this maximum throughput, the disc queues for the units on the controller must be kept non-empty and balanced. This requires a sustained high level of multi-programming and a proper spreading of data across the volumes. To make response times short, the more urgent disc

requests have to be performed first. It can be seen how the interrelations between memory management, processor management and disc management impact system performance.

8.2 Memory Management

Memory management requirements for the 3000 architecture consist of free space allocation, segment replacement, and garbage collection.

Free space allocation is required when a segment fetch is to be performed. The free space allocation algorithm selects the hole into which the segment should be read. Alternative strategies include first fit, best fit, and buddy schemes.

Segment replacement must be performed when a segment fetch is required but a hole of adequate size is not available. Alternative strategies include working set type policies and least recently used type policies.

Garbage collection is required in a segmented system to combine holes into larger holes. A variable sized allocation policy tends to produce small, unusable holes scattered throughout memory. This is known as external fragmentation. Garbage collection attempts to minimize the external fragmentation by combining the small holes into larger usable holes.

The major problems identified with the old memory manager were its serial nature, high fault rate caused by the per program working set replacement policy, restricted garbage collection performed during critical periods, and an inefficient free space allocation policy.

The old memory manager was entirely serial. Once the memory manager was started on a process swap-in, he couldn't begin on a second (or more important) swap-in until all the disc transfers required to finish the first swap-in completed. This serial memory management service forced artificial limits on the multiprogramming level. For large main memories this limitation restricts the system from achieving its potential performance.

The working set per program policy caused processes to release each others localities resulting in a high fault and recovery rate.

Garbage collection could only be performed locally within a bank, and performed during allocation time, so that memory management service time was further increased.

MPE III free space allocation selected the first fit hole causing large holes to be used up before they were needed. This resulted in excess invocation of the replacement policy.

The memory management policies were entwined with the rest of the system so that minor strategy changes would require extensive development.

MPE IV memory management solves each of these problems and in addition presents a general, structured implementation which allows major strategy changes with minor development effort.

Free space allocation is implemented by a best fit policy using size ordered free lists. This scheme is very fast, and saves the big holes until they're needed.

The resulting external fragmentation is eliminated through background garbage collection. Main memory garbage collection is performed as a background activity using cpu cycles during which the processor would have otherwise been idle. Garbage collection attempts to move small assigned regions located between large holes into small fragmented holes. The large holes are combined into even larger holes, and the small holes are eliminated. This skews the distribution of hole sizes towards the large holes and eliminates the external fragmentation thereby reducing the frequency of application of the replacement policy. The garbage collection code is responsive to the system state, and returns to the dispatcher when more urgent activity becomes pending.

The memory replacement policy is a very low overhead implementation of a global least recently used (LRU) policy. When a hole of the required size is not available, segments not needed by the current multiprogramming set (as determined by a global LRU algorithm) are selected for replacement on a memory ordered basis. In the segmented architecture of the 3000 family, this replacement policy proved to be superior to the working set policies. The memory scanned LRU approach tends to release unneeded segments in adjacent regions of memory, thereby creating large holes with few replacements. In contrast, the working set policies were found to require many more segment replacements to satisfy placement requests since they freed up space randomly through memory when releasing a working set of segments.

Segment fetching is an unblocked parallel operation in which memory management code invoked directly by the dispatcher sets up the operation and
the disc management code finishes it off as the required transfers complete.

8.3 Processor Management

Processor management consists primarily of selecting the activity to which the cpu should be devoted. The major cpu activities include running system and user processes, swapping in processes, and garbage collection.

Processor management is implemented by assigning priorities to the pending activities and giving the cpu over to the activity with the most urgent priority. This function is performed by the dispatcher.

Priority assignment in the old kernel had a problem with batch jobs.

Batch jobs would migrate up in priority to compete equally with interactive processes during busy periods.

Activity selection was restricted in the old kernel due to the memory manager being serial. The consequence of this limitation was that even if plenty of free space was available, memory management could not be performed when needed for a process if a more urgent process was waiting for disc I/O to complete. It couldn't be risked to swap-in a less urgent process since the more urgent process might need memory management service soon and the memory manager would be busy. The dispatcher was forced to pause the cpu rather than to work on increasing the multiprogramming level.

The MPE IV processor management scheme is very flexible. Priority assignments and activity selection are directed towards optimizing the system performance and can be tuned by the operator.

Priority assignments are made to reflect the performance goals of the system. Each scheduling class (C,D,E) has a base priority and a limit priority. When a transaction begins or a job is introduced into the system, the related process gets its class' best priority, the class base priority. As the process uses more cpu time than that required for an average member of the class, the process is considered to be less urgent and its priority drifts towards the class's limit priority. The limit priority is the worst priority that a process in the class can get assigned to it.

The priorities of processes placed in the A or B scheduling classes are kept static over time. The filtering parameter which determines the migration rate for a C, D or E scheduled process from its class base to class limit is dynamically tuned for C scheduled processes only. Bounds on the filtering parameters for C, D and E classes are set in the :TUNE command.

This priority assignment scheme enables the dispatcher to apply a scheduling policy which approximates a "shortest processing time first" algorithm. This gives maximum system throughput and best response time for short transactions while slightly delaying the longer transactions.

The C, D, and E classes can be made to overlap so that the processes in the various classes compete with each other, or they can be made disjoint. By making them disjoint, D and E processes will always be preempted for C processes. This tuning causes batch work to be performed as background activity between bursts of interactive transactions. This is the default tuning setting.

The operator or system manager can control the base and limit priorities of each class, and the rate at which a process' priority moves from the base to the limit through the :TUNE command.

CPU activity selection procedes by inspecting the priority ordered queue of processes requiring cpu service. When a process is encountered which

is ready to run, the process is launched. If the process requires some memory scheduling, the swap-in procedure is invoked directly by dispatcher. If there's nothing better to do, main memory garbage collection takes place.

In MPE IV, swapping-in of a process is performed by nested procedures on the dispatcher's stack. The fetching of a segment on behalf of a process is a low overhead, unblocked operation which allows an unlimited degree of parallelism in memory management. The swap-in code is responsive to the system state, and returns to the dispatcher when a process more urgent than the one that's being worked-on becomes ready or requires scheduling attention.

If the queue of ready processes is empty and there are no processes requiring memory scheduling , or increasing the multi-programming level has been determined to be dangerous at this time, the dispatcher invokes the background garbage collection code which returns when more urgent activity becomes pending.

9. Performance

The performance of MPE IV as measured by system transaction throughput and mean transaction response time is published in the "HP 3000 Performance Guide for Installed Systems." Performance tests were conducted using a standard application workload which represented a general-purpose EDP environment with a mix of online data base and program development sessions and background batch jobs.

The measurement results from these tests indicated that under light loads relative to system configuration, MPE IV showed slight performance improvement over MPE III. This was anticipated, since the MPE IV kernel seeks its performance improvements through the exploitation of parallelism, and the potential for parallelism is small under light loads. As the workloads were increased, MPE IV showed substantial improvement in both transaction response time and transaction throughput over MPE III. The performance improvements were realized across the family under the configurations and workloads measured.

The behavior of the system was exactly that which was sought. The system exhibited stability and good performance across the range of workloads, processor speeds, memory sizes and system configurations examined.

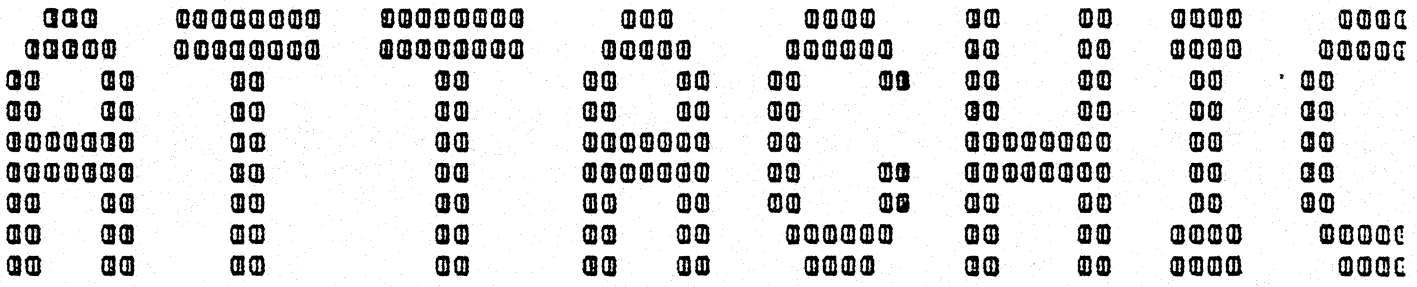
10. Conclusions

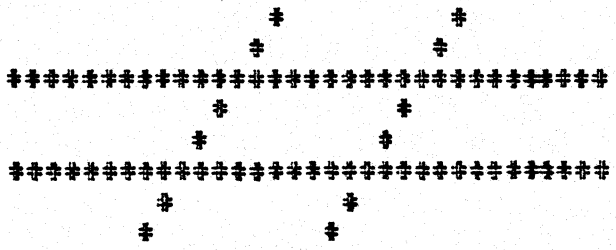
The approach to kernel design and implementation undertaken by the MPE IV kernel project resulted in an operating system foundation which naturally fits the evolving 3000 computer family to the environments it supports. The structured approach permitted alternatives to be easily implemented, and the measurement interface permitted them to be

thoroughly evaluated. The final implementation consists of integrated resource managers who cooperate to provide the best performance with the given system configuration under the current workload. The validity of the approach taken to kernel design is demonstrated by the resulting kernel's reliability and performance.

Acknowledgements

Special recognition is due to those who significantly contributed to the project's success. Professor Wesley Chu of UCLA and Professor Forrest Baskett of Stanford impacted the process through their valuable consultations. Alan Hewer, Howard Morris, and Neil Wilhelm kept things on course through their careful reviews. Ron Kolb, Ray Ventura and Bruce Blinn, through their design and development help on seek-ahead, the measurement interface, and multi-spindle virtual memory respectively, helped to speed the kernel to completion. Chris Moeller with his tuning help and Marcia McConnel and Carl Sassenrath with their debugging assistence contributed to the system's performance and reliability. Ken Spalding, through his coordinating function in the late stages, helped to get the system out the door.





DISTILLED DOWN FROM VARIOUS SOURCE MATERIALS

BY JOE ZIEGLER GSD/TSE HP-ST.PAUL

MAY 1980 AS OF MPE 1918-DC

HP32002B.01.DC

*** ATTACHIO ***

SYSTEM INTERNALS PROCEDURE

DECLARATION:

```
DOUBLE PROCEDURE ATTACHIO(LDEV,QMISC,DSTX,ADDR,FUNC,CNT,P1,P2,FLAGS)
  VALUE   LDEV,QMISC,DSTX,ADDR,FUNC,CNT,P1,P2,FLAGS;
  INTEGER LDEV,QMISC,DSTX,ADDR,FUNC,CNT,P1,P2,FLAGS;
  OPTION  UNCALLABLE,PRIVILEGED;
```

FUNCTION:

THIS PROCEDURE CONSTRUCTS AN IOQ ELEMENT AND LINKS IT TO THE APPROPRIATE DEVICE QUEUE. IF THIS IS THE FIRST ELEMENT IN THE QUEUE OR THE REQUEST SPECIFIES PREEMPTION, THE MONITOR IS CALLED TO INITIATE THE OPERATION. FOR BLOCKED REQUESTS, THE MONITOR MAY BE RECALLED BY ATTACHIO AFTER A "WAIT" IF THE REQUEST IS NOT COMPLETED WHEN THE CALLER IS AWOKEN.

IF NO IOQ ELEMENTS ARE AVAILABLE, IMPEDABLE REQUESTS ARE SUSPENDED UNTIL AN IOQ ELEMENT BECOMES AVAILABLE.

REQUESTS WHICH SPECIFY NOT IMPEDABLE ARE NOT "WAITED" FOR ANY REASON.

CALLING PARAMETERS:

LDEV - LOGICAL DEVICE NUMBER

QMISC - MISCELLANEOUS PARAMETER SPECIFIED FOR THE DEVICE. IF NOT SPECIFIED MUST BE ZERO.

DSTX - DST NUMBER OF DATA SEGMENT. IF ZERO THEN SPECIFIES THAT ADDR IS DB RELATIVE TO THE CALLERS STACK. MUST BE ZERO IF SYSTEM BUFFER IS SPECIFIED.

ADDR - DEPENDING ON FLAGS.(14:1) AND DSTX THIS MAY BE:

1.) OFFSET TO DATA IN DATA SEGMENT.
2.) OFFSET TO DATA IN CALLERS STACK.
3.) INDEX TO A SYSTEM BUFFER.

FUNC - FUNCTION CODE. DEVICE DEFINED BUT USUALLY:

0 - READ
1 - WRITE
2 - OPEN FILE
3 - CLOSE FILE
4 - DEVICE CLOSE

CNT - DATA TRANSFER COUNT:
POSITIVE FOR WORDS,
NEGATIVE FOR BYTES.

CALLING PARAMETERS: (CONT.)

P1 - PARAMETER 1. DEVICE DEPENDANT.

P2 - PARAMETER 2. DEVICE DEPENDANT.

FLAGS - CONTROL AND SPECIFICATION FLAGS.

.(0:7) - 0

.(7:2) - PREMPTION FLAGS:

1 - SOFT PREMPTION
2 - HARD PREMPTION

.(9:1) - 0

.(10:1) - SPECIAL REQUEST. DEVICE DEFINED. IF SET THEN SPECIAL HANDLING IS TO BE APPLIED TO THIS REQUES

.(11:1) - IF SET THEN THIS IS A DIAGNOSTIC REQUEST.

.(12:1) - SYSTEM BUFFER FLAG. IF SET THE ADDR IS AN INDEX RELATIVE TO THE SBUF TABLE. FOR DEVICES WHICH SUPPORT CHAINING, THE DATA IS TRANSFERED TO AND A SET OF CHAINED BEFFERS, UP TO A MAXIMUM OF 102 WORDS.

.(13:3) - REQUEST TYPE:

0 - UNBLOCKED	NO WAKE ON COMPLETION IMPEDE IF NO IOQ ELEMENT AVAILABL
1 - BLOCKED	CALLER IS TO BE WAITED UNTIL REQUEST IS COMPLETED
2 - UNBLOCKED	WAKE CALLER ON REQUEST COMPLETION IMPEDE IF NO IOQ ELEMENT AVAILABL
3 - UNBLOCKED	NO PROCESS TO BE ASSOCIATED IMPEDE IF NO IOQ ELEMENT AVAILABL
4 - UNBLOCKED	NO WAKE ON COMPLETION DO NOT IMPEDE FOR NO IOQ AVAILABL
5 - RESERVED	
6 - UNBLOCKED	WAKE ON COMPLETION BUT DO NOT IMPEDE FOR NO IOQ AVAILABL
7 - SAME AS 3	BUT DO NOT IMPEDE FOR NO IOQ ELEM

WORD OR BYTE COUNT PARAMETER:

ALL DRIVERS FOLLOW THE CONVENTION -

IF COUNT IS NEGATIVE THEN IT IS A BYTE COUNT
IF COUNT IS POSITIVE THEN IT IS A WORD COUNT

DRIVER I/O FUNCTION SPECIFICATIONS:

THE INPUT AND OUTPUT SPECIFICATIONS OF ALL THE DRIVERS HAS BEEN KEPT AS CONSISTENT AS POSSIBLE IN ORDER TO MAKE THE MOST COMMON CALLS TO "ATTACHIO" REASONABLY DEVICE INDEPENDENT.

THESE SIX FUNCTION CODES HAVE BEEN RESERVED FOR THE FOLLOWING PURPOSE FOR ALL DEVICES.

FUNCTION CODE	MEANING
0	READ
1	WRITE
2	FILE OPEN
3	FILE CLOSE
4	DEVICE CLOSE
28	GENERAL DEVICE FUNCTION

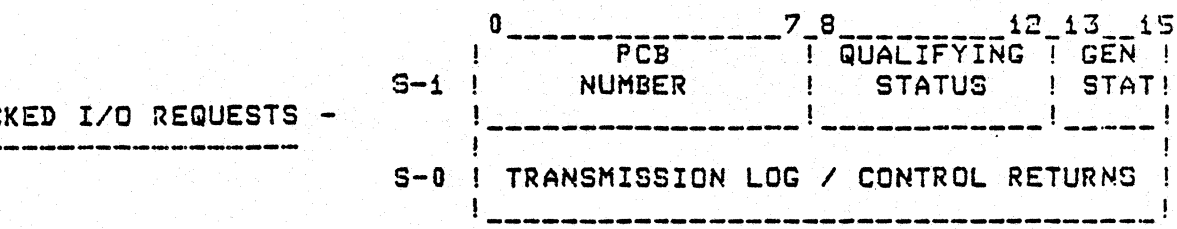
ALL OTHER FUNCTION CODES ARE SPECIFIC TO THE TYPE OF DEVICE, AND THE EXACT DEVICE ACTIVITY CAUSED BY THESE CODES ALSO VARIES WITH THE DEVICE.

FOPEN, FCLOSE, AND DEVICE CLOSE:

DEVICE	FILE OPEN	FILE CLOSE	DEVICE CLOSE
DISC	NOP	NOP	NOP
MAGNETIC TAPE	NOP	NOP	REWIND & UNLOAD RESET EOF FLAGS
LINE PRINTER	ISSUE TOP OF FORM IF NOT THE LAST THING DONE		
CARD READER/PUNCH	IF OUT, PICK	IF IN, STACK	RESET EOF FLAGS
PAPER TAPE PUNCH	PUNCH LEADER	PUNCH TRAILER	NOP
PAPER TAPE READER	NOP	NOP	RESET EOF FLAGS
PLOTTER	NOP	NOP	NOP
TERMINAL	CR/LF	CR/LF	RESET EOF FLAGS SET UP TO SPEED SENSE ON NEXT

DOUBLE WORD RETURN:

BLOCKED I/O REQUESTS -



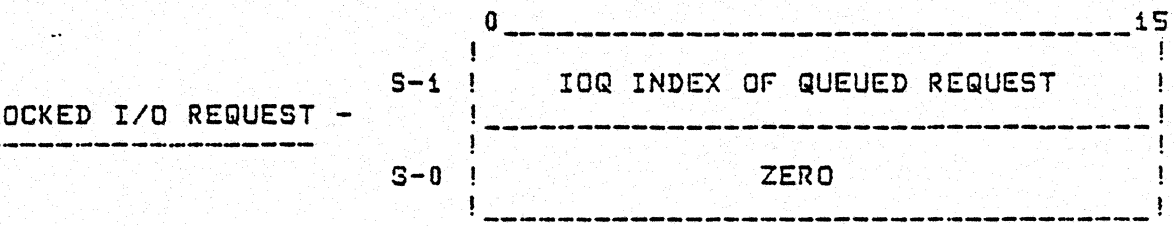
LOW ORDER WORD - TRANSMISSION LOG / CONTROL RETURNS

XLOG RETURNED WITH SAME SENSE AS CNT IN THE CALLING PARAMETERS TO "ATTACHIO".
POSITIVE - WORDS
NEGATIVE - BYTES

HIGH ORDER WORD - REQUEST COMPLETION STATUS

.(0: 7) - PCB NUMBER
.(8: 5) - QUALIFYING STATUS
.(13: 3) - GENERAL STATUS

UNBLOCKED I/O REQUEST -



LOW ORDER WORD - IOQ INDEX OF QUEUED UP REQUEST

IOQX RETURN IS USED AS THE CALLING PARAMETER TO PROCEDURE "IOSTATUS", TO DETERMINE THE COMPLETION STATUS OF THE REQUEST.

IF THE REQUEST TYPE IN FLAGS SPECIFIED NON-IMPEDABLE REQUEST, THEN THE IOQX WILL BE 0 IF NO IOQ ELEMENT WAS AVAILABLE.

HIGH ORDER WORD - 0

FOR TYPE 3 REQUESTS, IF ADDR IS NOT ZERO THEN IT IS ASSUMED TO BE A SYSTEM BUFFER INDEX. AT THE COMPLETION OF THE REQUEST, THE SBUF(S) POINTED TO BY ADDR ARE RETURNED TO THE FREELIST BY THE I/O SYSTEM.

*** I/O SYSTEM ***

I/O SYSTEM STATUS RETURNS:

QUALIFYING (8:5)	GENERAL (13:3)	OVERALL (8:8
1 - WAITING FOR COMPLETION	0 - PENDING	1
2 - DOING ERROR RECOVERY		2
3 - NOT READY WAIT		3
4 - NO WRITE WRING WAIT		4
5 - NEW PAPER TAPE WAIT		5
0 - NORMAL	1 - SUCESSFUL	1
1 - READ TERMINATED WITH SPEC CHAR		%11
2 - TAPE RETRY FOR SUCCESS REQUIRED		%21
3 - LOW TAPE OR END OF TAPE AFTER WRITE		%31
1 - PHYSICAL END OF FILE	2 - END OF FILE	%12
2 - :DATA		%22
3 - :EOD		%32
4 - :HELLO		%42
5 - :BYE		%52
6 - :JOB		%62
7 - :EOJ		%72
1 - TERMINAL PARITY ERROR	3 - UNUSUAL CONDITION	%13
2 - TERMINAL READ TIMED OUT		%23
3 - I/O ABORTED EXTERNALLY		%33
4 - DATA LOST		%43
5 - DATA SET READY DROPPED OR UNIT NOT ON LINE		%53
6 - ABORTED BECAUSE OF POWER FAIL		%63
7 - BOT AND BSR, BSF REQUEST		%73
%10 - TAPE RUNAWAY		%103
%11 - EOT AND WRITE REQUEST		%113
%12 - NO WRITE RING AFTER CONS/OP/REPLY		%123
%13 - END OF TAPE (PAPER TAPE LOW)		%133
%14 - PLOTTER LIMIT SWITCH REACHED		%143
%15 - ENABLE SUBSYS BREAK AND NO CONTROL-Y PIN.		%153
%16 - READ TIME RETURNED OVERFLOW		%163
%20 - WRITE AND NO CARD IN WAIT STATION		%203
0 - INVALID REQUEST	4 - IRRECOVERABLE ERROR	4
1 - TRANSMISSION ERROR		%14
2 - I/O TIME OUT		%24
3 - TIMING ERROR		%34
4 - SIO FAILURE		%44
5 - UNIT FAILURE		%54
6 - INVALID DISC ADDRESS		%64
7 - TAPE PARITY ERROR		%74
%11 - PAPER TAPE TAPE ERROR		%114
%12 - SYSTEM ERROR		%124
%13 - INVALID SBUF INDEX		%134

*** DEVICE & FUNCTION SPECIFIC PARAMETERS ***

DEV / OPERATION

DISC

====	FUNCT	P1	P2	XLOG	
READ	0	DBL DISCADDR		----	ODD BYTE CNT ALWAYS RNDED
WRITE	1	DBL DISCADDR		----	CNT RNDED-UP TO NEXT MULT OF 128. RMNDR OF SECT FI WITH LAST DATA WORD SENT.
FILE CLOSE	3	----	----	----	
DEVICE CLOSE	4	----	----	----	
ZERO FILL	5	DBL DISCADDR		----	(%000000) 1 SECT/CALL
SPACES FILL	6	DBL DISCADDR		----	(%020040) 1 SECT/CALL
REQST STATUS	7				
FORMAT TRACK	8	DBL DISCADDR		----	INIT C/H/S = SEEK C/H/S
INITLZE TRACK	9	DBL DISCADDR		----	INIT C/H/S & FLG @ ADDR-2 ADDR-1. (NOT DONE BY IOCD
RD FULL SECTOR	10	DBL DISCADDR		----	138 WORDS / SECT
WRITE LABEL	11	DBL DISCADDR		----	0/0 (REL SECTOR ZERO)
RD W/O SPARING	12	DBL DISCADDR		----	

MAG-TAPE

========	FUNCT	P1	P2	XLOG	
READ	0	EOF	----	XCNT	
WRITE	1	----	EOT FLG	----	
FILE OPEN	2	----	----	----	NOP
FILE CLOSE	3	----	----	----	NOP
DEVICE CLOSE	4	----	----	----	REWIND & UNLOAD
REWIND	5	----	----	----	
WRITE EOF	6	----	----	----	
FWRD SPC FILE	7	----	----	----	
BKWD SPC FILE	8	----	----	----	
REWIND & UNLOAD	9	----	----	----	
GAP FWRD	10	----	----	----	
FWRD SPC RECD	11	----	----	----	(READ 0 WDS WILL ALSO FSR
BKWD SPC RECD	12	----	----	----	

LINE-PRINTER

==========	FUNCT	P1	P2	XLOG	
WRITE	1	VFC	MODE	----	
FILE OPEN	2	----	----	----	TOF IF NOT THERE
FILE CLOSE	3	----	----	----	TOF IF NOT THERE
DEVICE CLOSE	4	----	----	----	TOF IF NOT THERE
DOWNLOAD VFC	64	6/8-LPI	----	----	CNT,ADDR&DSTX=VFC BUFFER
SET LEFT MRGIN	65	MARGIN	FLAG	----	IF P2.(15) SET, STORE P1 NEW DEFAULT LEFT MARGIN.

*** DEVICE & FUNCTION SPECIFIC PARAMETERS ***

DEV / OPERATION

CARD-READER

	FUNCT	P1	P2	XLOG	
READ	0	EOF SPEC	MODE	----	MODE=ASCII/BINARY
FILE OPEN	2	----	----	----	
FILE CLOSE	3	----	----	----	
DEVICE CLOSE	4	----	----	----	CLR EOF STATUS IN LPDT

CARD-RDR/PUNCH

	FUNCT	P1	P2	XLOG
READ	0	EOF SPEC	MODE	----
WRITE	1	----	MODE	----
FILE OPEN	2	----	----	----
FILE CLOSE	3	----	----	----
DEVICE CLOSE	4	----	----	----
CONTROL	5	----	FUNCTION	----

PAPER-TAPE-RDR

	FUNCT	P1	P2	XLOG
			STOP CHR	
READ	0	EOF	MODE	----
FILE OPEN	2	----	----	----
FILE CLOSE	3	----	----	----
DEVICE CLOSE	4	----	----	----

PAPER-TAPE-PNCH

	FUNCT	P1	P2	XLOG
WRITE	1	SPC-CTL	MODE	----
FILE OPEN	2	----	----	----
FILE CLOSE	3	----	----	----
DEVICE CLOSE	4	----	----	----
SWITCH TAPE	5	----	----	----

PLOTTER

	FUNCT	P1	P2	XLOG	
WRITE	1	----	----	----	
FILE OPEN	2	----	----	----	MASTER CLEAR
FILE CLOSE	3	----	----	----	NOP
DEVICE CLOSE	4	----	----	----	NOP
RTRN DEV SUBTYP	5	----	----	----	

*** DEVICE & FUNCTION SPECIFIC PARAMETERS ***

DEV / OPERATION

TERMINALS

	FUNCT	P1	P2	XLOG	
READ	0	EOF SPEC	STP CHR	----	P1.(0:1) = 1 = NO LF ON .(13:3) = EOF SPEC P2.(0:8) = IF<>0 SPEC E .(10:1) = OWN DC1/DC2 .(11:2) = 0/<>0 ASCII/
WRITE	1	LINECTRL	FLAG	----	P1 = 0 INSERT NORMAL CR/ = 1 USE FIRST BYTE OF > 1 USE P1 FOR VFC P2.(11:2) = IF<>0 BINARY .(15:1) = PRESPACE FLA
FILE OPEN	2	----	----	----	NOP / ALLOCATE WITH DEFA
FILE CLOSE	3	----	----	----	DUMPS PARTIAL REC & CR/L
DEVICE CLOSE	4	----	----	----	DISCONNECT IF MODEM HANG
SET TIMEOUT	5	#SECONDS	----	----	P1 = 0 TO CLEAR TIMEOUT
SET IN SPEED	6	CHRS/SEC	----	OLDSPEED	(RET XLOG OLD CHARS/SEC
SET OUT SPEED	7	CHRS/SEC	----	OLDSPEED	(RET XLOG OLD CHARS/SEC
SET ECHO ON	8	----	----	OLDSTATE	(RET XLOG 1/0 = ON/OFF
SET ECHO OFF	9	----	----	OLDSTATE	(RET XLOG 1/0 = ON/OFF
DISABL BREAK	10	----	----	----	
ENABLE BREAK	11	----	----	----	
DISABL CTL-Y	12	----	----	----	
ENABLE CTL-Y	13	----	----	----	
DISABL PTAPE	14	----	----	----	
ENABLE PTAPE	15	----	----	----	
DISABL TIMER	16	----	----	----	
ENABLE TIMER	17	----	----	----	
READ TIMER	18	----	----	SEC/100	(RET XLOG IN 1/100 SECS
* DISABL PARITY	19	----	----	----	
* ENABLE PARITY	20	----	----	----	
LOGGED ON	21	TYPE	----	----	P1=1 FOR SESSION ENABL B
NOT USED	22				
SET TERMTYPE	23	TERMTYPE	----	----	P1= MPE TERM TYPE
ALLOCATE TERM	24	TERMTYPE	SPEED	----	P2= SPEED IN CHARS/SECON
CL FLUSH & WRT	25	LINECTRL	----	----	CLR PARTIAL RECRD & WRIT
CTR-X ECHO ON	26	----	----	----	
CTR-X ECHO OFF	27	----	----	----	
INVALID REQST	28				NOP
PTAPE READ	29	DISCADD1	DISCADD2	----	P1&2=SECT ADD OF SPOOL B
SET BREAK MODE	30	ON/OFF	----	----	P1 ODD/EVEN = SET/CLR BR
SET CONSOL MODE	31	ON/OFF	----	----	P1 ODD/EVEN = SET/CLR CN
* SET PARITY	32	PARITY	----	----	P1 = 0 - NONE; BIT-8 ALW 1 - NONE; BIT-8 ALW 3 - ODD ; PRTY GEN/ 4 - EVEN; PRTY GEN/
ALLOCATE TERM	33	TERMTYPE	SPEED	----	
SET TERM TYPE	34	TERMTYPE	----	----	
GET TERM TYPE	35	----	----	TERMTYPE	
GET OUT SPEED	36	----	----	OUTSPEED	
SET STOP CHRS	37	STOPCHRS	----	----	P1.(0:8) = SUB SYS BREAK P1.(8:8) = END OF RECORD P1 = 0 DISABLE SPECIAL C
SET CONSL INT	38	ON/OFF	----	----	P1 ODD = ENABLE CTL-A R EVEN = DISABL CTL-A R

* INVALID FUNCTION FOR 2635-K OR 2645-K TERMS

END OF FILE SPECIFICATIONS

EOF FLAGS IN LPDT

The Logical to Physical Device Table (LPDT) contains a 3-BIT EOF FIELD. The I/O-SYSTEM twiddles these bits to keep track of possible EOF conditions caused by the last READ request.

EOF CODE	CONDITION DETECTED
0	NO EOF
1	HARDWARE EOF or :EOF:
2	:DATA or DATA
3	:EOD
4	HELLO
5	BYE
6	:JOB
7	:EOJ

EOF DETECTION

Some DEVICES can generate a HARDWARE signal to indicate an EOF condition while others must use a specific DATA PATTERN in order to signal that an END OF FILE has been reached.

for DISCS — the File System keeps track of the LOGICAL EOF and no other indication is recognized.

for MAG-TAPE — an actual TAPE-MARK is WRITTEN and SENSED by the TAPE DRIVE and is indicated in the STATUS return

on PAPER TAPE — a data pattern must be used. :EOF: will be recog nized by the DRIVER whether in ASCII or BINARY MO 30 FEED-FRAMES at the start of a RECORD will caus the DRIVER to issue an OUT OF TAPE message, and t wait for another tape.

the TERMINAL — also uses the :EOF:, but it is ignored when in th BINARY MODE.

and CARD READERS — can go either way with the HARDWARE EOF BUTTON to generate a EOF when the STACKER goes EMPTY, or a :EOF: CARD encountered.

EOF SPECIFICATION

Except for the TAPE-MARK, which always results in an EOF return, the EOF condition, (GENERAL STATUS.(13:3)), is only returned if the READ REQUEST enabled the CLASS of EOF detected. This is done passing an EOF SPEC in P1.(13:3) of the request.

Reads with an EOF SPEC of ZERO are always initiated. Before any PHYSICAL READ is done, the driver passes back any residue from a previous READ where the CNT was satisfied or the EOR was encounter before the, device dependent, MINIMUM PHYSICAL READ was used up. This is done just as if the PHYSICAL READ had really taken place.

EOF SPECS	INTERPRETATION
0	Reset EOF condition and READ
1	Check for HARDWARE EOF or :EOF:
2	Check for :DATA, :EOD, or :EOJ
3	Check for DATA, HELLO, BYE, and JOB
4	Check for :DATA, :JOB, and :EOJ
5	READ but do not reset the EOF, pass back saved data or check read data for an EOF

Before READS with an EOF SPEC other than 0 or 5 are initiated, the previous EOF condition is checked against the EOF SPEC. If the previous EOF condition was 1 (HARDWARE) or the EOF SPEC > 1 AND previous EOF > 1, then the previous EOF condition is passed back and no READ takes place.

When a READ is initiated, the previous EOF condition is cleared. The data is checked on completion of the READ to see if an EOF of the type specified has occurred. If an EOF of the type SPECIFIED has occurred then an EOF indication is returned to the user and th type of EOF is saved in the LPDT.

```
<<IOPTRD0 - MODULE 13>>
<< HP32002B MPE SOURCE B.01.00 >>

$TITLE "IOPTRDR0 - DRIVER FOR  PAPER TAPE READER"
$THIRTY
$CONTROL PRIVILEGED,UNCALLABLE
```

DRIVER REQUEST CODES:

0 - READ
P1.(13:3) EOF SPECIFICATION

P2 (0:8) SPECIAL READ TERMINATION CHARACTER
(10:3) - 0 ASCII
- 1 BINARY

2 - OPEN FILE - NOP

3 - CLOSE FILE - NOP

4 - CLOSE DEVICE - RESET EOF FLAGS

STATUS RETURNS:

QUALIFYING (8:5)	GENERAL (13:3)	OVERALL (8:8)
	0 - PENDING	0
	1 - SUCCESSFUL	1
X - SEE EOF SPECIFICATIONS	2 - EOF	%X2
3 - ABORTED EXTERNALLY	3 - UNUSUAL CONDITION	%33
6 - POWER FAIL ABORT		%63
0 - INVALID FUNCTION	4 - IRRECOVERABLE	4
1 - TRANSMISSION		%14
2 - I/O TIMEOUT		%24
4 - SIO FAILURE		%44
5 - UNIT FAILURE		%54

WHENEVER THE OUT-OF-TAPE CONDITION IS DETECTED OR IN ASCII MODE, IF 30 NULL FRAMES ARE DETECTED AN OUT-OF-TAPE MESSAGE IS ISSUED AND THE DRIVER WAITS FOR THE DEVICE TO BE MADE READY AGAIN. THIS DEVICE SUPPORTS CHAINED SYSTEM BUFFERS.

```
<<IOPTPN0 - MODULE 14>>
<< HP32002B MPE SOURCE B.01.00 >>

$CONTROL PRIVILEGED,UNCALLABLE
$THIRTY
$TITLE "2895A PAPER TAPE PUNCH DRIVER"

DRIVER REQUEST CODES:

  1 - WRITE        - P1              - VERTICAL FORMAT IF ASCII
                                     0 - PRINT AND SINGLE SPACE(XOFF/CR/LF)
                                     1 - USE FIRST DATA BYTE AS VERTICAL
                                         FORMAT SPECIFICATION.
                                         %53 "+" PUNCH AND XOFF/CR (NO LF)
                                         %60 "0" PUNCH AND DOUBLE SPACE
                                                 (XOFF/CR/LF/LF).
                                         %200-%277 PUNCH AND N-%200 SPACES
                                                 (XOFF/CR/N-%200 LFS).
                                         %320 PUNCH (NO CR/LF).

                   - P2.(11:2) - ASCII/BINARY MODE
                                 0 - ASCII
                                 1 - COLUMN BINARY
                   - P2.(15:1) - 0 - POSTSPACE.PUNCH DATA THEN PUNCH
                                     SPACING CHARACTERS.
                                 1 - PRESPACE.PUNCH SPACING CHARACTERS
                                     THEN PUNCH DATA.
  2 - OPEN FILE    - P2.(13:3) - TYPE OF ACCESS
                                 0 - OPEN FOR READ ONLY
                                 1 - OPEN FOR WRITE ONLY
                                 2 - OPEN FOR READ AND WRITE
  3 - CLOSE FILE
  4 - CLOSE DEVICE
  5 - REQUEST NEW TAPE           PUNCH TRAILER
                                 ISSUE NEW TAPE REQUEST TO OPERATOR
                                 PUNCH LEADER

STATUS RETURNS:

  QUALIFYING (8:5)              GENERAL (13:3)      OVERALL (8:8)
 ------------------------      ------------------  ---------------
                                 0 - PENDING                 0

    0 - NO ERRORS                1 - SUCCESSFUL              1
    3 - TAPE LOW AFTER PUNCH                               %31

    X - SEE EOF SPECIFICATIONS   2 - EOF                    %X2

    3 - ABORTED EXTERNALLY       3 - UNUSUAL CONDITION      %33
    6 - POWER FAIL ABORT                                    %63

    0 - INVALID FUNCTION         4 - IRRECOVERABLE           4
    1 - TRANSMISSION                                        %14
    2 - I/O TIMEOUT                                         %24
    4 - SIO FAILURE                                         %44
    5 - UNIT FAILURE                                        %54

IN BINARY MODE, THE DATA IS PUNCHED AS PRESENTED, NO SPACING CHARS
OR RECORD DELIMITERS ARE ADDED.  THIS DEVICE SUPPORTS CHAINED SBUFS.
```

```
<<IOPLOT0 - MODULE 15>>
<< HP32002B MPE SOURCE B.01.00 >>


$CONTROL PRIVILEGED,UNCALLABLE
$THIRTY
$TITLE "30226A PLOTTER INTERFACE DRIVER"


DRIVER REQUEST CODES:

    1 - WRITE
    2 - OPEN FILE               - MASTER CLEAR
    3 - CLOSE FILE              -     NOP
    4 - CLOSE DEVICE            -     NOP
    5 - RETURN DEVICE SUBTYPE


STATUS RETURNS:

  QUALIFYING (8:5)              GENERAL (13:3)      OVERALL (8:8)
 -------------------------     ----------------    ---------------
    1 - WAITING FOR COMPLETION    0 - PENDING             %10
    3 - NOT READY WAIT                                    %30

    0 - NO ERRORS                 1 - SUCCESSFUL            1

    3 - ABORTED EXTERNALLY        3 - UNUSUAL CONDITION   %33
    6 - POWER FAIL ABORT                                  %63
  %14 - LIMIT SWITCH REACHED                             %143

    0 - INVALID FUNCTION          4 - IRRECOVERABLE         4
    1 - TRANSMISSION                                      %14
    3 - TIMING ERROR                                      %34
    4 - SIO FAILURE                                       %44
    5 - UNIT FAILURE                                      %54
```

```
<<IOMDISC0 - MODULE 16>>
<< HP32002B MPE SOURCE B.01.00 >>

$CONTROL PRIVILEGED,UNCALLABLE,MAIN=IOMDISC0
$THIRTY
$TITLE "7900A AND 2888A MOVING HEAD DISC DRIVER"

DRIVER CALLING SEQUENCE

   DRIVER REQUEST CODES

      0 - READ              - EVEN NUMBER OF BYTES TRANSFERED ON
                              ALL READS.  ODD BYTE COUNTS ARE
                              ROUNDED UP.

      1 - WRITE             - MULTIPLES OF 128 WORDS, ( 1 SECTOR ),
                              ALWAYS WRITTEN.  THE LAST WORD IN
                              BUFFER WILL FILL REMAINDER OF SECTOR
                              WHERE CNT IS NOT AN EVEN MULTIPLE.

      2 - FILE OPEN

      3 - FILE CLOSE

      4 - DEVICE CLOSE

      5 - FILL WITH ZEROS   - WRITES ONE WORD ( ZERO ). CONTROLLER
                              WILL FILL REMAINDER OF SECTOR WITH SAME.

      6 - FILL WITH BLANKS  - WRITES ONE WORD ( %020040 ). CONTROLLER
                              WILL FILL THE REMAINDER OF SECTOR.

   NOTE THAT THE BUFFER ADRESS WORD OF THE IOQ IS USED AS THE BUFFER
   IN ALL FILL OPERATIONS.

   PARAMETERS P1 AND P2 FORM A DOUBLE-WORD DISC ADDRESS.  THIS IS A
   VOLUME RELATIVE SECTOR ADDRESS,  STARTING FROM SECTOR 0 .

   CNT PARAMETER IS WORD/BYTE COUNT.  ODD BYTE COUNTS WILL BE ROUNDED
   UP TO AN EVEN NUMBER.

DRIVER RETURN CODES:

   GENERAL STATUS              QUALIFYING STATUS                OVERALL
   0 - PENDING                 1 - WAIT FOR COMPLETION           %10
   1 - SUCCESSFUL              0 - NORNAL COMPLETION             %01
   4 - IRRECOVERABLE ERROR     0 - INVALID FUNCTION              %04
                               1 - DISC XFER ERROR               %14
                               3 - I/O XFER ERROR                %34
                               4 - SIO FAILURE                   %44
                               5 - DISC UNIT FAILURE             %54
                               6 - INVALID DISC ADRESS           %64


THE CALLING PARAMETERS FOR BOTH FIXED AND MOVING HEAD DISC ARE
PRETTY MUCH THE SAME,  EXCEPT FOR THOSE DISC TYPES WHICH CAN
SUPPORT PRIVATE VOLUMES AND SERIAL DISC. ( SEE IOMDISC1 ).

ALL DISCS SUPPORT CHAINED I/O OPERATIONS
```

$TITLE "7900A AND 2888A MOVING HEAD DISC DRIVER"

DISC CONTROLLER STATUS:

BIT	7900A	2888A
0	SIO OK	SAME
1	SEEK CHECK	200 TPI
2	INTERRUPT REQUEST	SAME
3	ON LINE	SAME
4	DRIVE UNSAFE	SAME
5	DRIVE NOT READY	SEEK INCOMPLETE
6	ACCESS NOT READY	UNIT BUSY
7	UNUSED	PACK CHANGE
8-12	CONTROLLER STATUS	SAME
13-15	DISC UNIT NUMBER	SAME

ENCODED CONTROLLER STATUS:

CODE	7900A	2888A
0	NO ERROR	SAME
1	ILLEGAL OPCODE	SAME
3	UNUSED	CYL # TOO BIG
4	SECTOR # TOO BIG	HEAD # TOO BIG
5	UNUSED	TIME OUT
6	DEFECTIVE TRACK	SAME
7	HEADS MISPOSITIONED	SAME
%10	UNUSED	CYCLIC ADRESS ERROR
%11	CYCLIC ERROR	CYCLIC DATA ERROR
%12	I/O PROG ERROR	SAME
%13	UNUSED	SEQUENCE ERROR
%14	CYL OVERRUN	SAME
%15	ZERO SECTOR COUNT	SAME
%16	DATA OVERRUN	SAME
%20	UNUSED	ILLEGAL TERMINATION
%22	HD/SECT COMPARE ERROR	UNUSED
%23	ACCESS NOT READY	DRIVE ERROR
%24	XFER ERROR	SAME
%26	DATA PROTECTED	UNUSED
%37	DRIVE ATTENTION	SAME

IRRECOVERABLE ERRORS ARE RETURNED ONLY AFTER THE OPERATION HAS BEEN RETRIED 10 TIMES. IN ADDITION, A RECALIBRATE AND 10 MORE RETRIES ARE AUTOMATIC FOR MOVING HEAD DISC.

WHENEVER AN IRRECOVERABLE TRACK SPECIFIC ERROR IS DETECTED, A TRACK BY TRACK TRANSFER IS INITIATED. DURING THIS PROCESS, IF AN ERROR IS DETECTED, THE TRACK NUMBER IS ENTERED INTO THE DEFECTIVE TRACK TABLE, [SECTOR 1 OF EACH DISC], AS A SUSPECT BAD TRACK.

```
<<IOFDISC0 - MODULE 17>>
<< HP32002B MPE SOURCE B.01.00 >>
$CONTROL MAIN=IOFDISC0
$CONTROL PRIVILEGED,UNCALLABLE
$TITLE "2660A FIXED HEAD DISC DRIVER"
$THIRTY
```

DRIVER CALLING SEQUENCE

DRIVER REQUEST CODES:

0 - READ
1 - WRITE
2 - FILE OPEN
3 - FILE CLOSE
4 - DEVICE CLOSE
5 - FILL WITH ZEROS
6 - FILL WITH BLANKS

COUNT - WORD/BYTE COUNT. BYTE COUNT WILL BE ROUNDED UP TO AN EVEN #

PARAMETERS P1 AND P2 FORM A DOUBLE WORD DISC ADRESS. P1 MUST ALWAYS BE ZERO.

NOTE THAT THE BUFFER ADRESS WORD IN THE IOQ IS USED AS THE BUFFER FOR ALL FILL OPERATIONS.

DRIVER RETURN CODES:

GENERAL STATUS	QUALIFYING STATUS	OVERALL
0 - PENDING	1 - WAIT FOR COMPLETION	%10
1 - SUCCESSFUL	0 - NORMAL COMPLETION	%01
4 - IRRECOVERABLE ERROR	0 - INVALID FUNCTION	%04
	1 - DISC TRANSMISSION ERR	%14
	3 - I/O TRANSMISSION ERROR	%34
	5 - DISC UNIT FAILURE	%54
	6 - INVALID DISC ADRESS	%64

```
<<IOTAPE0 - MODULE 18>>
<< HP32002B MPE SOURCE B.01.00 >>

$CONTROL MAIN=IOTAPE0
$CONTROL PRIVILEGED,UNCALLABLE
$THIRTY
$TITLE "7970A/B/E MAGNETIC TAPE DRIVER"

DRIVER REQUEST CODES:

    0 - READ
        P1(13:3) - END OF FILE SPECIFICATION
    1 - WRITE
        P2(15:1) - IF SET THEN WRITE PAST END OF TAPE MARK
                   IF CLEAR THEN RETURN ERROR IF EOT HAS BEEN DETECTED
    2 - OPEN FILE (NO OPERATION)
    3 - CLOSE FILE (RESET EOF FLAGS IN LPDT)
    4 - CLOSE DEVICE (RESET EOF FLAGS AND REWIND TAPE)
    5 - REWIND
    6 - WRITE TAPE MARK
    7 - FORWARD SPACE FILE
    8 - BACKSPACE FILE
    9 - REWIND AND UNLOAD
   10 - GAP
   11 - FORWARD SPACE RECORD
   12 - BACKSPACE RECORD

COUNT - WORD/BYTE COUNT. NOTE THAT ALTHOUGH THE BYTE COUNT RETURNED
        WILL REFLECT THE TRUE INFORMATION TRANSFER, AN EVEN NUMBER OF
        BYTES WILL ALWAYS BE PHYSICALLY TRANSFERRED.


DRIVER RETURN CODES:

  GENERAL STATUS (13:3)        QUALIFING STATUS (8:5)     OVERALL (8:8)

    0 - PENDING                  1 - WAITING FOR COMPLETION   %10
                                 2 - DOING ERROR RECOVERY     %20
                                 3 - NOT READY WAIT           %30
                                 4 - NO WRITE RING WAIT       %40

    1 - SUCCESSFUL               0 - NO ERRORS                 1
                                 2 - RETRY WAS NECESSARY      %21
                                 3 - EOT AFTER WRITE          %31

    2 - END OF FILE              1 - A TAPE MARK WAS READ OR  %12
                                     P1 WAS NONZERO AND THE
                                     LAST RECORD READ WAS A
                                     TAPE MARK
                               2 - 7 THE DATA DEPENDENT EOF
                                     CONDITION AS SPECIFIED
                                     BY EOFCHECK

    3 - UNUSUAL CONDITION        3 - REQUEST ABORTED          %33
                                 6 - POWERFAIL ABORT          %63
                                 7 - BOT AND BACKSPACE REQ.   %73
                               %10 - TAPE RUNAWAY            %103
                               %11 - EOT AND WRITE REQ.      %113
```

<<IOTAPE0 - MODULE 18>> (CONT.)

$TITLE "7970A/B/E MAGNETIC TAPE DRIVER"

DRIVER RETURN CODES: (CONT.)

4 - IRRECOVERABLE ERROR	0 - INVALID REQUEST	4
	1 - TRANSMISSION ERROR	%14
	3 - TIMING ERROR	%34
	4 - SIO FAILURE	%44
	5 - UNIT FAILURE	%54
	7 - TAPE PARITY ERROR	%74
	%12 - SYSTEM ERROR	%124

TAPE CONTROLLER STATUS

BITS	USE
0	SIO OK
1	ODD BYTE READ
2	INTERRUPT REQUEST
3- 4	UNIT NUMBER
5	END OF TAPE
6	WRITE PROTECTED
7	UNIT READY
8	BEGINNING OF TAPE
9	800/1600 CPI DRIVE
10	LAST OPERATION WAS A WRITE
11	END OF FILE MARK DETECTED
12-14	ENCODED ERROR STATUS
15	9/7 TRACK DRIVE (ALWAYS 0)

ENCODED ERROR STATUS

000	UNIT READY
001	TRANSFER ERROR
010	COMMAND REJECTED
011	TAPE RUNAWAY
100	TIMING ERROR
101	TAPE ERROR
110	UNUSED
111	NO ERROR ENCOUNTERED

```
<<  MPE MODULE 19, IOLPRT0  >>
<< HP32002B MPE SOURCE B.01.00 >>

$TITLE "CDC, TALLEY, DATAPRODUCTS, DATAPRINTER AND HP2608 DRIVER"
$THIRTY
$CONTROL PRIVILEGED, UNCALLABLE, MAIN=IOLPRT0
```

DRIVER REQUEST CODES:

OPERATION	QFUNC	PARAMETERS (QPAR1, QPAR2, QWBCT)
WRITE	1	QPAR1 -- Vertical Format Specification. QPAR2 -- Mode control flags. QWBCT -- <0 ==> bytes, >0 ==> words, =0 ==> advance paper only.
FILE OPEN	2	No params. Page eject if not already at top of form.
FILE CLOSE	3	Same as FILE OPEN.
DEVICE CLOSE	4	No params. Page eject if not already at top of form. Then, for 2608 only, send Master Clear to printer and download system default left margin.
DOWNLOAD VFC (2608 only)	%100	QPAR1 -- 6 or 8 (lines per inch). Any other value ==> 6. QPAR2 -- Not used. QWBCT -- Form (VFC) length. <0 ==> bytes, >0 ==> words, =0 ==> reload printer's internal VFC. BANK/BUFADDR -- Address of VFC buffer to be sent to the printer.
SET LEFT MARGIN (2608 only)	%101	QPAR1 -- Logical to physical column offset, 0 to 15. Any other value ==> 15. QPAR2 -- If bit 15 is set, store QPAR1 as system default left margin offset.

If function code %100 or %101 is sent to a printer which is not a 2608, or if any function code not listed above is sent to any printer, it will be rejected as an invalid request (see completion statuses below).

$TITLE "CDC, TALLEY, DATAPRODUCTS, DATAPRINTER AND HP2608 DRIVER"

For QFUNC = 1 (write), QPAR1 and QPAR2 are interpreted as follows:

QPAR1	MEANING
%53,	ASCII "+", suppress space
%55,	ASCII "-", triple space, no automatic perforation skip
%60,	ASCII "0", double space, no automatic perforation skip
%61,	ASCII "1", top of form*
%200-%277,	slew N-%200 lines (0-63)
%300-%317,	skip to VFC channel N-%277 (within limits of printer)
%320,	append to current buffer without printing
1,	the Vertical Format character is the first byte in the data buffer. It is not sent to the printer as data. If the first byte is not one of the above, a single space is assumed. Use QPAR2.(14:1) to determine if the space is with or without automatic page eject. The format character is included in the word/byte count in QWBCT, therefore QWBCT may not be 0 if QPAR1 = 1. Such a condition will cause an immediate completion with Invalid Request status (QSTAT.(8:8) = 4).
ANY OTHER,	assume single space. Check QPAR2.(14:1), as above.

* Page ejects (skips to VFC channel 1) are suppressed whenever the previous operation ended with a page eject.

QPAR2 -- Mode control flags. These are two 1-bit flags.

QPAR2.(15:1) -- Prespace flag. If set, the request is for a space-then-print, or prespace, operation. Since all HP printers are mechanically postspace printers, prespace mode must be simulated. This is done by printing any previously transmitted data using the Vertical Format specification of the current QPAR1, then filling the printer buffer (without printing) with the current contents of BANK/BUFADDR.

If the Prespace flag is clear, IOLPRT0 performs a postspace operation, filling the printer buffer from BANK/BUFADDR and printing using the Vertical Format specification of QPAR1.

NOTE: If a postspace operation follows a prespace operation, IOLPRT0 first prints any previously existing data with a skip to VFC channel 3 (single space with automatic page eject -- QPAR2.(14:1), the no-auto-page-eject flag, is not examined). It then executes the postspace request.

QPAR2.(14:1) -- No automatic page eject flag. If set, the single space called for by the default carriage control will be executed as a slew, that is, without regard to the logical bottom of form or the paper perforation. If the flag is clear, the default carriage control is executed as a channel skip to VFC channel 3. When using the 6 or 8 lines per inch standard VFC pattern, this causes a skip to the top of the next form if the spacing would go beyond the logical bottom of form.

$TITLE "CDC, TALLEY, DATAPRODUCTS, DATAPRINTER AND HP2608 DRIVER"

STATUS RETURNS:

GENERAL (13:3)	QUALIFYING (8:5)	OVERALL (8:8)	
1 - Successful	0	1	Operation complete, no errors.
3 - Unusual	3	%33	Request aborted externally.
	6	%63	Power fail abort.
	%21	%213	Printer powered on or master cleared. VFC/left margin/top of form environment must be restored.
	%27	%273	VFC has been reset to default at the printer. Custom VFC environment must be restored.
4 - Irrecoverable	0	4	Invalid request (function or parameter.
	1	%14	Transfer error: a) Illegal memory address b) Memory parity error c) Parity error in transfer
	2	%24	Timeout error, printer took too long.
	4	%44	SIO failure, couldn't start SIO program.
	%12	%124	No message link buffers, couldn't write message to operator console.

```
<<IOCDRD0 - MODULE 20>>
<< HP32002B MPE SOURCE B.01.00 >>

$TITLE "DEC 26,1975   CARD READER DRIVER  -  IOCDRD0(20)"
$CONTROL PRIVILEGED,UNCALLABLE,MAIN=IOCDRD0
$THIRTY

CARD READER  REQUEST CODES:

  OPERATION          FUNC          PARAMETERS

    READ              0      P1 - END OF FILE SPECIFICATION

                             P2.(11:2) - DATA MODE

                                   0 - PACKED ASCII
                                       COUNT TRUNCATED TO 80 BYTES MAX
                                   1 - COLUMN BINARY
                                       COUNT TRUNCATED TO 80 WORDS MAX

  ALL ODD BYTE COUNTS ARE ROUNDED UP TO AN EVEN NUMBER


  FILE  OPEN          2      NO OPERATION
  FILE  CLOSE         3      NO OPERATION
  DEVICE CLOSE        4      CLEAR EOF STATUS IN LPDT

STATUS RETURNS:

  QUALIFYING (8:5)              GENERAL (13:3)        OVERALL (8:8)
 ------------------------      ------------------    ---------------

                                 0 - PENDING                  0

                                 1 - SUCCESSFUL               1

    X - SEE EOF SPECIFICATIONS   2 - END OF FILE            %X2

    3 - ABORTED EXTERNALLY       3 - UNUSUAL CONDITION      %33
    6 - POWER FAIL ABORT                                    %63

    0 - INVALID FUNCTION         4 - IRRECOVERABLE            4
    4 - START SIO FAILURE                                   %44
  %12 - NO MESSAGE LINK BUFFERS AVAIL.                      %124
```

```
<<IOTERM0 - MODULE 22>>
<< HP32002B MPE SOURCE B.01.00 >>

$TITLE " MAY 27,1976   ASYNCHRONOUS TERMINAL MONITOR"
$THIRTY
$CONTROL PRIVILEGED, UNCALLABLE, MAIN = IOTERM0

TERMINAL REQUESTS:

  OPERATION          FUNC          PARAMETERS

    READ              0      P1.( 0:1)  = NOCRLF IF SET
                               .(13:3)  = EOF  SPECIFICATION

                             P2.(0:8 )  - IF <> 0 THEN SPECIAL RD STOP CH
                               .(11:2)  - IF 0 THEN ASCII ELSE BINARY
                               .(10:1)  - OWN DC1/DC2 HANDSHAKE

    WRITE             1      P1.        - VERTICAL FORMAT SPECIFICATION

                                 1      - USE 1ST DATA CHAR AS FORMAT SPE

                               %53      - "+", CR ONLY NO LF
                               %60      - "0", DOUBLE SPACE
                               %61      - "1", TOP OF FORM

                               %200-%277,  N-%200 LF'S AND A CR

                               %320- NO CR OR LF

                               ALL OTHERS, DO A CR/LF

                             P2.(15:1)  - PRESPACE FLAG
                               .(11:2)  - IF NOT 0 THEN BINARY


  FILE OPEN           2      IF ONLINE NOP ELSE ALLOCATE WITH DEFAULT
  FILE CLOSE          3      CLEAN UP AND CRLF IF NOT NEW LINE
  DEVICE CLOSE        4      DISCONNECT

  SET READ TIMEOUT    5      P1 - IF 0 CLEAR TIMEOUT ELSE TIME IN SECONDS

  SET INPUT SPEED     6      P1 - SPEED IN CHARACTERS/SECOND
                                   OLD SPEED RETURNED IN COUNT
  SET OUTPUT SPEED    7      P1 - SPEED IN CHARACTERS/SECOND
                                   OLD SPEED RETURNED IN COUNT

  ENABLE ECHO         8      RETURN OLD ECHO SETTING IN COUNT
  DISABLE ECHO        9      RETURN OLD ECHO SETTING IN COUNT

  DISABLE BREAK       10
  ENABLE  BREAK       11

  DISABLE CONTROL Y   12
  ENABLE CONTROL Y    13

  DISABLE TAPEMODE    14
  ENABLE  TAPEMODE    15
```

<<IOTERM0 - MODULE 22>> (CONT.)

$TITLE " MAY 27,1976 ASYNCHRONOUS TERMINAL MONITOR"

TERMINAL REQUESTS: (CONT.)

DISABLE READ TIMER	16	
ENABLE READ TIMER	17	
RETURN READ TIME	18	COUNT CONTAINS READ TIME IN 1/100 SECONDS
DISABLE PARITY CHK	19	INVALID IF TTYPE OF 8-BIT DATA SEND/RECEIVE DEVICE (HP2635/45K)
ENABLE PARITY CHK	20	INVALID IF TTYPE OF 8-BIT DATA SEND/RECEIVE DEVICE(HP2635/45K)
LOGGED ON	21	P1 - IF 1 THEN SESSION, SO ENABLE BREAK IF 2 THEN JOB, NO BREAK IF 0 THEN DATA, NO BREAK
NOT USED	22	
SET TERM TYPE	23	P1 - TERM TYPE AS SPECIFIED IN THE MPE ERS
ALLOCATE TERMINAL	24	P1 - TERM TYPE AS SPECIFIED IN THE MPE ERS P2 - SPEED IN CHARACTERS/SECOND
CLR FLUSH & WRITE	25	PARAMETERS SAME AS WRITE
CNTRL X ECHO ON	26	
CNTRL X ECHO OFF	27	
NO OPERATION	28	
PTAPE READ	29	P1, P2 - SPOOL BUFFER BEGINNING DISC ADDRESS
SET BREAK MODE	30	P1 - ODD = SET, EVEN = CLEAR BREAK MODE
SET CONSOLE MODE	31	P1 - ODD = SET, EVEN = CLEAR CONSOLE MODE
SET PARITY	32	P1 - 0 - NO PARITY GENERATE, BIT 8 ALWAYS 0 1 - NO PARITY GENERATE, BIT 8 ALWAYS 1 2 - EVEN PARITY ON WRITE, EXPECTED ON 3 - ODD PARITY ON WRITE,EXPECTED ON RE
*****NOTE: FUNC. INVALID IF TTYPE=HP2635/45K****************		
ALLOCATE TERMINAL	33	P1 - TERM TYPE AS SPECIFIED IN THE MPE ERS P2 - SPEED IN CHARACTERS/SECOND
SET TERM TYPE	34	P1 - TERM TYPE AS SPECIFIED IN THE MPE ERS
RETURN TERM TYPE	35	TERMINAL TYPE RETURNED IN COUNT WORD
RETURN OUT SPEED	36	OUTPUT SPEED RETURNED IN COUNT WORD
SET STOP CHARS	37	P1 - IF 0 THEN DISABLE SPECIAL EOR AND SUBSYSTEM BREAK CHARACTERS ELSE .(0:8) = SUB SYSTEM BREAK CHARACTER .(8:8) = END OF RECORD CHARACER
SET CONS INTERRUPT	38	P1 - ODD,ENABLE CNTRL A; EVEN DISABLE CNTRL

```
$TITLE " MAY 27,1976   ASYNCHRONOUS TERMINAL MONITOR"
```

TERMINAL STATUS RETURNS:

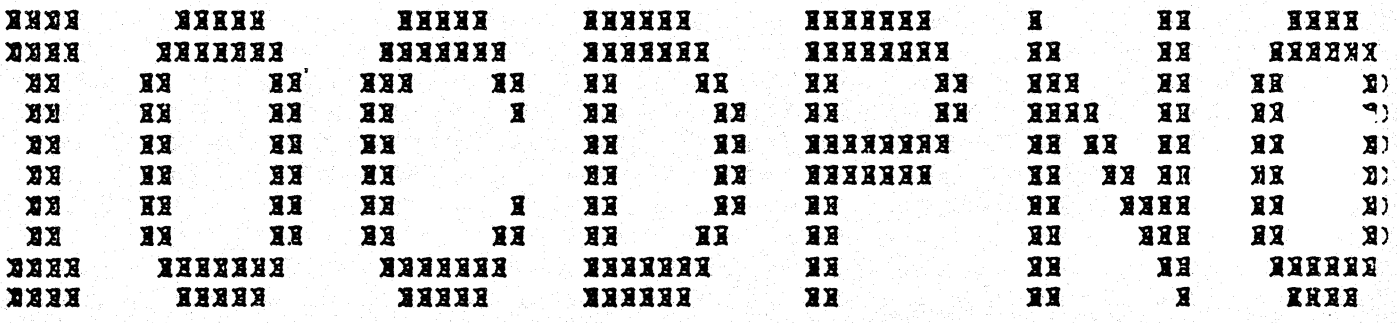
```
      %00  - NOT COMPLETED OR NOT STARTED

      %01  - NORMAL COMPLETION
      %11  - COMPLETED ON SPECIAL READ STOP CHARACTER

      %X2  - END OF FILE

      %13  - PARITY ERROR ON READ
      %23  - READ TIMED OUT
      %33  - I/O ABORTED EXTERNALLY
      %43  - DATA LOST, NO BUFFER AVAILABLE OR TOO LONG TAPEMODE RECOR
      %53  - NOT ON LINE, DATA SET NOT READY OR DISCONNECT
     %153  - ENABLE SUB SYS BRK REQUESTED & NO CY PIN
     %163  - READ TIME RETURNED ) (2**16-1) HUNDREDTHS
     %173  - READ STOPPED WHEN A BREAK DETECTED

      %04  - INVALID REQUEST, FUNCTION OR PARAMETER
      %34  - TIMING ERROR, UNIT WAS NOT SERVICED IN TIME
```

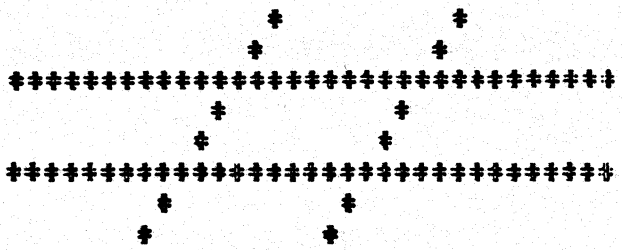
```
<<IOPRPNO - MODULE 24>>
<< HP32002B MPE SOURCE B.01.00 >>

$CONTROL PRIVILEGED,UNCALLABLE
$THIRTY
$TITLE "2894A PRINTING READER PUNCH DRIVER"

DRIVER REQUEST CODES:

    0 - READ
        P1.(13:3) - END OF FILE SPECIFICATION
                    0 - RESET EOF AND READ
                    1 - CHECK FOR HARDWARE EOF
                    2 - CHECK FOR SUPERCOLON EOF
                    3 - CHECK FOR COLON EOF
        P2.(11:2) - ASCII/BINARY MODE
                    0 - ASCII
                    1 - COLUMN BINARY
    1 - WRITE
        P2.(10:1) - SYSTEM WRITE
                    0 - USER INITIATED WRITE
                    1 - SYSTEM INITIATED WRITE  (HEADER CARD)
           (11:2) - ASCII/BINARY MODE
                    0 - ASCII
                    1 - COLUMN BINARY
    2 - OPEN FILE
        P2.(13:3) - TYPE OF ACCESS
                    0 - OPEN FOR READ ONLY
                    1 - OPEN FOR WRITE ONLY
                    2 - OPEN FOR READ AND WRITE
    3 - CLOSE FILE
    4 - CLOSE DEVICE
    5 - CONTROL
        P1.(6:1)    0 - SELECT NO INHIBIT FEED ON WRITE
                    1 - SELECT INHIBIT FEED ON WRITE
           (7:1)    0 - SELECT PUNCH ON WRITE
                    1 - SELECT NO PUNCH ON WRITE
           (8:1)    0 - SELECT PRINT ON WRITE
                    1 - SELECT NO PRINT ON WRITE
           (9:1)    0 - SELECT PRINT AND PUNCH SAME DATA
                    1 - SELECT PRINT AND PUNCH SEPARATE DATA
           (10:1)   0 - SELECT PRIMARY STACKER
                    1 - SELECT SECONDARY STACKER
           (11:1)   0 - SELECT PRIMARY HOPPER
                    1 - SELECT SECONDARY HOPPER

STATUS RETURNS:

  QUALIFYING (8:5)                GENERAL (13:3)      OVERALL (8:8)
    1 - WAITING FOR COMPLETION      0 - PENDING              %10
    3 - NOT READY WAIT                                       %30
                                    1 - SUCCESSFUL             1
    X - SEE EOF SPECIFICATIONS      2 - EOF                  %X2

    3 - ABORTED EXTERNALLY          3 - UNUSUAL CONDITION    %33
    6 - POWER FAIL ABORT                                     %63

    0 - INVALID FUNCTION            4 - IRRECOVERABLE          4
    1 - TRANSMISSION                                         %14
    2 - I/O TIMEOUT                                          %24
    5 - UNIT FAILURE                                         %54
```

```
$TITLE "2894A PRINTING READER PUNCH DRIVER"

DEFINITION OF DIT WORD 9(DACESS AND DACCP)
      BITS (0:5) ARE HOPPER/STACKER RELATED. THEIR MEANING FOLLOWS
           (0:1) =0 80 COLS OF DATA FROM NEXT CARD LEAVING THE
                    HOPPER WILL BE STORED IN THE MEMORY OF DEVICE
                 =1 DATA IN MEMORY OF DEVICE IS CLEARED WHEN THE
                    NEXT CARD PASSES THE READ STATION.
           (1:1) =0 STACKER CONTROL. CARDS ARE DISCHARGED TO THE
                    STACKER #1(RIGHT HAND-STACKER).
                 =1 STACKER CONTROL. CARDS ARE DISCHARGED TO THE
                    STACKER AS SPECIFIED BY BIT (2:1) BELOW.
           (2:1)    SELECT STACKER BIT.  THIS BIT IS EFFECTIVE FOR
                    STACKING WHEN BIT (1:1) IS A 1.  THE STACKER
                    SELECTED IS AS FOLLOWS:
                 =0 SELECT STACKER #1(RIGHT HAND-STACKER).
                 =1 SELECT STACKER #2(LEFT HAND-STACKER).
           (3:1) =0 HOPPER SELECT.  WHEN 0, SELECT FROM PRIMARY HOPPER
                    (HOPPER#1, ALSO KNOWN AS THE REAR HOPPER).
                 =1 HOPPER SELECT. WHEN 1, SELECT FROM SECONDARY
                    HOPPER.(HOPPER#2, OR FRONT HOPPER).
           (4:1) =0 INHIBIT INPUT FEED. WHEN 0, ALLOWS A CARD TO BE
                    FED FROM THE HOPPER TO THE WAIT STATION WHEN THE
                    CURRENT CARD IN WAIT STATION IS SENT TO STACKER
                 =1 INHIBIT INPUT FEED. WHEN 1, NO CARD CAN BE FED
                    FROM EITHER HOPPER TO THE WAIT STATION. HOWEVER,
                    A CARD CURRENTLY IN THE WAIT STATION MAY BE FED
                    TO A STACKER.
           (5:1) =0 DRIVER INTERNAL FLAG. WHEN 0,NO EOF HAS BEEN
                    DETECTED.
                 =1 AN EOF HAS BEEN DETECTED ON A READ OPERATION.
           (6:1) =0 INTERNAL BUFFER FLAG.  WHEN 0, NO INTERNAL
                    BUFFER IS BEING USED.
                 =1 AN INTERNAL AUXILLARY BUFFER(AT END OF SIO AREA
                    IS BEING USED.
           (7:3)    SEE PRINT/PUNCH OPTIONS IN DRIVER REQUEST CODES
                    UNDER (5-CONTROL, P1.(7:3)).
           (10:4) - UNUSED.
           (14:1)=0 NO FCONTROL HAS OCCURED FOR THIS FILE. USE
                    DEFAULT SETTINGS.
                 =1 FCONTROL HAS OCCURRED. USE SETTINGS IN THIS
                    DIT WORD TO INDICATE HOW TO OPERATE DEVICE.
           (15:1)=0 NO TIMER REQUEST IS PENDING(SEE DIT WORD %12).
                 =1 A TIMER REQUEST INDEX IS IN WORD %12 OF DIT AND
                    IS CURRENT(AN ABORTTIMEREQ MUST BE DONE ON INTRPT)

IOQ(QMISC) WORD DEFINITION FOLLOWS NEXT:

           (0:1) =0 NULL. NO FUNCTION
                 =1 I/O INITIATED,WAITING FOR COMPLETION INTERRUPT
           (1:1) =0 NULL. NO FUNCTION
                 =1 WAITING ON A "NOT READY" INTERRUPT TO BRING
                    DEVICE BACK ONLINE.
           (2:1) =0 CURRENT OPERATION IS NOT A WRITE OPERATION.
                 =1 CURRENT OPERATION IS A WRITE OPERATION.
           (3:1) =0 NORMAL.
                 =1 WAITING FOR THE OPERATOR TO CLEAR THE
                    CARD PATH AFTER A READ CHECK OR INVALID
                    CODE.
           (4:12)   NOT USED AT PRESENT.
```

```
<<IOMDISC1 - MODULE 27>>
<< HP32002B MPE SOURCE B.01.00 >>

$TITLE "7905/7920/7925 MOVING HEAD DISC DRIVER"
$THIRTY
$CONTROL PRIVILEGED,UNCALLABLE,MAIN=IOMDISC1

DRIVER CALLING SEQUENCE:

   DRIVER REQUEST CODES
     0 - READ
     1 - WRITE
     2 - FILE OPEN
     3 - FILE CLOSE
     4 - DEVICE CLOSE
     5 - FILL WITH ZEROS
     6 - FILL WITH BLANKS
     7 - REQUEST STATUS
     8 - FORMAT TRACK
     9 - INITIALIZE TRACK
    10 - READ FULL SECTOR
    11 - WRITE LABEL (SECTOR 0)
    12 - READ (WITH SPARING DISABLED)

   COUNT - WORD/BYTE COUNT. ODD BYTE COUNT WILL BE ROUNDED UP.

   PARAMETERS P1 AND P2 FORM A DOUBLE WORD DISC ADRESS FOR ALL XFERS.

DRIVER RETURN CODES:

   GENERAL(13:3)          QUALIFYING(8:5)              OVERALL(8:8)
   0 - PENDING            1 - WAIT FOR COMPLETION      %10
   1 - SUCCESSFUL                                      %01
   4 - IRRECOVERABLE      0 - INVALID FUNCTION         %04
       ERROR              1 - TRACK ERROR              %14
                          3 - XMISSION ERROR           %34
                          4 - SIO FAILURE              %44
                          5 - UNIT FAILURE             %54
                          6 - INVALID DISC ADRESS      %64
                          %12 - SYSTEM ERROR           %124
```

```
$TITLE "7905/7920/7925 MOVING HEAD DISC DRIVER"


DISC CONTROLLER HARDWARE STATUS RETURNS:

   STATUS 1 (RETURNED BY TIO,SENSE,ETC.)
   BITS     USE
     0      SIO OK
     1      RIO/WIO OK
     2      INTERRUPT REQUEST
   3-7      TERMINATION STATUS
               0 - NORMAL COMPLETION
               1 - ILLEGAL COMMAND
               7 - CYL COMPARE ERROR
             %10 - UNCORRECTABLE DATA ERROR
             %11 - HEAD-SECTOR COMPARE ERROR
             %12 - SIO PROGRAM ERROR
             %14 - END OF CYLINDER
             %16 - OVERRUN
             %17 - POSSIBLY CORRECTABLE DATA ERROR
             %20 - ILLEGAL ACCESS TO SPARE TRACK
             %21 - ATTEMPT TO ACCESS DEFECTIVE TRACK
             %22 - HEAD MOVEMENT DURING DATA OPERATION
             %23 - DISC DRIVE (STATUS-2) ERROR
             %26 - ATTEMPT TO WRITE ON PROTECTED TRACK
             %27 - DRIVE UNAVAILABLE
             %37 - DRIVE ATTENTION
  8-12      UNUSED
 13-15      UNIT NUMBER

NOTE THE FOLLOWING DIFFERENCE IN THE RETURN STATUS COMMAND

      0     SPARE TRACK
      1     PROTECTED TRACK
      2     DEFECTIVE TRACK


   STATUS 2 WORD
      0     DRIVE ERROR
    3-6     DRIVE TYPE NYBBLE
      8     ATTENTION
      9     PLATTER (7905) OR PACK (7920/7925) READ ONLY
     10     FORMAT
     11     DRIVE FAULT
     12     FIRST STATUS
     13     SEEK CHECK
     14     DRIVE NOT READY
     15     DRIVE BUSY
```





DISTILLED DOWN FROM VARIOUS SOURCE MATERIALS

BY JOE ZIEGLER GSD/TSE HP-ST.PAUL

JUN 1980 AS OF ATHENA 2011

HP32002B.02.00

*** IOCDPN0 ***

PROGRAM DESCRIPTION
==========

THIS PROGRAM PROVIDES THE MEANS TO GENERATE ATTACHIO CALLS.
COMMANDS ARE ALSO PROVIDED TO DISPLAY AND MODIFY SEVERAL OF THE
I/O SYSTEM TABLES. SEVERAL UTILITY FUNCTIONS ARE ALSO PROVIDED.

C? - INPUT A COMMAND

FORMAT OF COMMANDS:

OP [[=] PARAMETER [, PARAMETER]]

COMMANDS MAY BE ONE OR MORE LETTERS. THE OP'S ARE LISTED ABOVE WITH
THE NUMBER OF CHARACTERS REQUIRED TO UNIQUELY SPECIFY THE COMMAND.
COMMAND COMMENTS MAY BE ENCLOSED IN BRACKETS AS IN SPL. MORE THAN ON
COMMAND MAY BE INPUT AT A TIME BY SEPERATING THEM WITH SEMICOLONS.

PARAMETER MAY BE NULL, A NUMBER(% FOR OCTAL) OR 1 TO 2 NON QUOTE
CHARACTERS WITHIN QUOTES. THE ASCII AND ANOTE PARAMETER MAY BE NULL
OR A STRING WITHIN QUOTES. IF A 2ND PARAMETER IS SPECIFIED, POSITIVE
INDICATES WORDS AND NEGATIVE INDICATES BYTES. NO 2ND PARAMETER DEFAU
IS 1. PARAMETERS ARE DELIMITED BY A COMMA OR CARRAIGE RETURN.
NOTE, THERE IS NO DEFAULT USE OF OCTAL NUMBERS!

SYNTAX OF PARAMETERS -

```
EXPRESSION ::= TERM ! TERM + TERM ! TERM - TERM ! - TERM
TERM       ::= FACTOR ! FACTOR * FACTOR ! FACTOR / FACTOR
FACTOR     ::= CONSTANT ! ( EXPRESSION )
CONSTANT   ::= NUMBER ! "CHAR" ! "CHARCHAR"
```

COMMANDS OR A PROGRAM MAY BE GOTTEN FROM AN EDITOR FILE USING THE
DO COMMAND. WHEN THE DO COMMAND IS EXECUTED, COMMANDS AND PROGRAM
STATEMENTS IN THE EDITOR FILE SPECIFIED ARE EXECUTED AS IF THE HAD
BEEN ENTERED ON THE INPUT DEVICE.

PROGRAMS MAY BE WRITTEN TO EXECUTE A SEQUENCE OF COMMANDS, MUCH AS
IF THEY WERE ENTERED BY THE OPERATOR. PROGRAM STATEMENTS ARE
DISTINGUISHED BY BEGINING WITH A STATEMENT NUMBER. THE STATEMENTS AR
EXECUTED IN ORDER OF STATEMENT NUMBER WHEN THE PROGRAM IS RUN.
A STATEMENT MAY BE REPLACE BY ENTERING THE REPLACEMENT STATEMENT
WITH THE STATEMENT TO BE REPLACED NUMBER. A STATEMENT MAY BE DELETED
BY ENTERING THE STATEMENT NUMBER WITH NO STATEMENT. THE STORED PROGR
IS RUN FROM THE BEGINNING N TIMES USING THE "R" COMMAND. THE
PROGRAM MAY BE CLEARED AND LISTED WITH THE "CL" AND "LI" COMMANDS.

4100 WORD READ/WRITE BUFFER.
UP TO 50 SAVED IOQ INDEXES FOR UNBLOCKED I/O

IF THE SBUF FLAG IS SET THEN SBUFS ARE GOTTEN AND THE I/O IS DONE TO
OR FROM THEM. SBUF I/O CAN ONLY BE DONE WITH WAIT CODES 1 AND 3. NO

THE DRT NUMBER IS SET FROM SYSTEM TABLES USING THE LDEV SPECIFICATIO

CONTROL Y CAUSES THIS PROGRAM TO ABORT REPEATED OPERATIONS SUCH AS
PRINTING TABLES, EXECUTING THE ATTACHIO CALLS AND PROGRAM EXECUTION.

*** IOCDPN0 ***

COMMAND AND PARMS TABLE

OP	FUNCTION	PARAMETER	DEFAULT
A	FILL BUFFER WITH ASCII STRING	ASCII STRING	")" TO "Z"
ANOTE	PRINT STRING ON LIST DEVICE	ASCII STRING	NULL STRIN
B	FILL BUFFER WITH PARAMETER	PARAM, COUNT	0, ALL
C	SET COUNT PARAMETER TO ATTACHIO	COUNT	0
CH	CHECK RESULTS OF UNBLOCKED I/O		
CIO	DO CIO INSTRUCTION	CONTROL WORD	ERROR
CL	CLEAR PROGRAM AREA	FIRST, LAST	ALL
CM	CLEAR MONITORING TABLE		
D	DISPLAY ATTACHIO PARAMETERS		
DA	DISPLAY BUFFER IN ASCII	OFFSET,COUNT	0, XLOG
DB	DISPLAY BUFFER IN OCTAL	OFFSET,COUNT	0, XLOG
DD	DISPLAY DIT	OFFSET,COUNT	ALL
DE	CALL DEBUG		
DEC	CONVERT PARAM TO DECIMAL	PARAM	ERROR
DO	GET CMNDS AND/OR PROG FROM A FILE	FILE NAME	
DQ	DISPLAY IOQS	NUMBER OF IOQS	ALL
DRT	SET DRT NUMBER	DRT#	ERROR
DS	DISPLAY LAST STATUS AND XLOG		
DSIO	DISPLAY SIO AREA	OFFSET,COUNT	ALL
E	EXECUTE ATTACHIO CALL N TIMES	EXECUTION COUNT	1
EX	TERMINATE PROGRAM		
F	SET FUNCTION PARAMETER TO ATTACHIO	FUNCTION CODE	0
FL	SET FLAGS PARAMETER TO ATTACHIO	FLAGS	1
GO	GOTO STATEMENT NUMBER	STATEMENT #	1ST STATEMEN
H	CALL HELP		
I	INCREMENTAL FILL OF BUFFER	INCREMENT	1
L	SET LOGICAL DEVICE NUMBER	LDEV	ERROR
LI	LIST PROGRAM	FIRST, LAST	ALL
LD	SET OPTION LIST DEVICE	SEE BELOW	$STDLIST
	0=$STDLIST, 1=FORMAL DESIGNATOR	"LIST", 2+=ATTACHIO	LDEV
M	MONITOR TERMINAL ACTIVITY	MONITORING CODE	0
MB	MODIFY BUFFER	OFFSET,COUNT	ERROR
MD	MODIFY DIT	OFFSET,COUNT	ERROR
MQ	MODIFY FIRST IOQ	OFFSET,COUNT	ERROR
N	NULL INPUT EXECUTES 1 ATTACHIO	0/1 (OFF/ON)	1
O	CONVERT PARAM TO OCTAL	PARAM	ERROR
P	PRINT TERMINAL ACTIVITY TABLE	NUMBER OF ENTRIES	ALL
PA	SET P1 PARAMETER TO ATTACHIO	P1	0
PB	SET P2 PARAMETER TO ATTACHIO	P2	0
Q	SET QMISC PARAMETER TO ATTACHIO	QMISC	0
R	RUN PROGRAM N TIMES	RUN COUNT	1
RIO	DO RIO INSTRUCTION		
RUN	CREATE AND RUN A PROGRAM	PROGRAM FILE NAME	
SH	PRINT DO AND PROGRAM STATEMENTS	0/1 (OFF/ON)	1
T	PRINT AVERAGE TIME FOR LAST ATTACHIO CALLS		
TIO	DO TIO INSTRUCTION		
V	PUT RECORD NUMBER IN BUFFER	0/1 (OFF/ON)	1
WIO	DO WIO INSTRUCTION	WRITE DATA	ERROR
X	EXPLAIN COMMANDS		

*** IOCDPN0 ***

CONTROL I/O DATA BUFFER:

A	ASCII BUF FILL
B	OCTAL BUF FILL
DA	SHOW BUF ASCII
DB	SHOW BUF OCTAL
I	INCREMENT FILL
MB	MODIFY BUFFER

These allow user to create and fill data buffer with desired patterns of binary or ascii data to be written on output devices and to display all or any part of the data read from input devices in either ascii or binary. The max. size of buffer is 4100 words. the default on display commands is to use the XLOG if data was just read in, or the COUNT if buffer was just filled.

SET PARMS FOR ATTACHIO:

L	SET LDEV
F	SET FUNCTION
C	SET COUNT PARAM
FL	SET FLAGS
PA	SET P1
PB	SET P2
Q	SET QMISC

These functions dont cause any immediate activity, they merely establish the values of the various parameters to be used in the next call to ATTACHIO.

Once they are set, they will remain the same until changed. That is except for QMISC which can be altered by some of th dirvers to return misc info. See TABLES MANUAL after DIT info for device depende use of QMISC

D	DISPLAY PARAMS

Lists the current setting of all parms.

MAKE CALL TO ATTACHIO:

E	CALL ATTACHIO
DS	DISPLAY STATUS
CH	CHECK RESULTS
T	PRINT TIME
N	NULL INPT DO E
V	ADD REC# IN BUF

Allows user to execute calls directly to I/O SYSTEM without going thru the FILESY ATTACHIO builds an I/O QUEUE entry and links it into the device QUEUE as specif ied by the FLAGS parm. Status is return when I/O is completed for WAIT I/O calls It is displayed automatically, only if other than normal completion. For NOWAI calls, user must check the progress of the request before he can presume it has completed and the status will be valid.

WRITING DO PROGRAMS:

DO	READ CMND FILE
SH	PRINT DO & PROG
R	RUN STORED PROG
LI	LIST PROGRAM
CL	CLR PROG AREA
GO	GOTO STATEMENT#

Allows the user to enter sequences of commands with LINE#s and run them as a program in interpretive mode. Being as the "R"un command does not clear any of the parm values of buffers, the actual operation may be a mix-match of commands and program steps. For instance the sam program may be executed on any LDEV as established by the "L" command, as long as the program doesn't change the "L" pa

CONTROL & DOCUMENT ASST:

X	EXPLAIN CMDS
AN	DO ANOTE
LD	SET LIST DEV
DEC	DECIMAL CONVERT
O	OCTAL CONVERT
EX	TERMINATE

User may list programs or data to any output device on the system or send it via a file equate to DISC.

Using "LD i" sends output to formal file desig. "LIST". This allows spooling of printout. When the LDEV is supplied, th output is sent directly to the device ev if it happens to be spooled or owned.

FALLTHROUGH OPERATIONS:

		Note: In DEBUG an accidental "E" will ge
DE	CALL DEBUG	you out, but in IOCDP#0 it may get you i
H	CALL HELP	Bring up "HELP" term in debug. SER II & pre LC SER III only. Must have a termin on the old-style SYSTEM CLOCK CONSOLE PC
(C)	MPE COMMAND	Any "PROGRAM EXECUTABLE MPE COMMAND"

MONITOR TERM ACTIVITY:

M	MONITOR TERM
P	PRT MNTR TABLE
CM	CLR MNTR TABLE

I/O TABLE FUNCTIONS: Allows user to display and modify the most prominent of the device related I/O tables.

DD	DISPLAY DIT
MD	MODIFY DIT
DQ	DISPLAY IOQS
MQ	MODIFY IOQ

DSI	DISPLAY SIO PROG	Octal display of SIO ORDER PROG. for DEV

DO DIRECT I/O COMMANDS:

DRT	SET DRT#	Allows DRT#s not associated with an LDEV such as SYS/CLK, MEMLOG/INTRFC, & TCI. Primarily for use with DIO commands.
RIO	DO RIO INSTR	These execute DIRECT I/O commands for
WIO	DO WIO INSTR	SER II/III. They must not be attempted
CIO	DO CIO INSTR	on a 30 or 33. They use the DRT# of
TIO	DO TIO INSTR	the current "L" ldev if a "DRT" command has not been encountered

The following is a list of the commands recognized by IOCDPN0, grouped according to function with detailed description of the parameters and operation.

USER'S I/O BUFFER FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
A	ASCII STRING	")" TO "Z"	A"SOME TEXT"

FILL THE BUFFER WITH AN ASCII STRING. BUFFER REFERS TO A USER'S I/O BUFFER DECLARED WITHIN THIS PROGRAM. MAXIMUM SIZE IS 4100 WORDS. IF I/O IS PENDING, A PENDING MESSAGE WILL BE PRINTED. NO PARAMETER INPUT DEFAULT TO THE ENTIRE BUFFER FILLED WITH)*+,-./0123456789 :;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ REPETITIVELY.

B	PARAM,COUNT	0,ALL	B0,128

FILL THE BUFFER WITH THE PARAMETER INPUT FOR THE SPECIFIED COUNT. IF NO COUNT SPECIFIED, ALL 4100 WORDS OF THE BUFFER WILL BE FILLED. NO PARAMETER RESULT IN ENTIRE BUFFER FILLED WITH ZEROS. IF I/O IS PENDING, A PENDING MESSAGE WILL BE PRINTED.

DA	OFFSET,COUNT	0,XLOG	DA0,20

DISPLAY THE BUFFER IN ASCII FROM A GIVEN OFFSET FOR A LENGTH OF COUNT. NO LENGTH SPECIFIED RESULT IN THE LENGTH OF THE LAST TRANSMISSION (XLOG) COUNT DISPLAYED.

DB	OFFSET,COUNT	0,XLOG	DB8,3

DISPLAY THE BUFFER IN OCTAL WORDS FROM A GIVEN OFFSET FOR A LENGTH OF COUNT. NO LENGTH SPECIFIED RESULT IN THE LENGTH OF THE LAST TRANSMISSION (XLOG) COUNT DISPLAYED.

I	INCREMENT	1	I2

THE USER'S I/O BUFFER WILL BE FILLED RECURSIVELY AS FOLLOWS: B(N)=B(N-1)+INCREMENT WHERE N GOES FROM (1-4100) TO ACCOMMODATE THE ENTIRE BUFFER.

MB	OFFSET,COUNT	ERROR	MB7,1

MODIFY THE USER'S I/O BUFFER FROM A GIVEN OFFSET FOR A LENGTH OF COUNT. THE LOCATION OF EACH CONSECUTIVE WORDS FOR MODIFICATION WILL BE PROMPTED UNTIL SPECIFIED COUNT EXPIRES. ON PROMPTS FOR MODIFICATION, A (CR) DENOTES NO CHANGES MADE.

ATTACHIO PARAMETERS FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
D			D

DISPLAY THE PARAMETERS SET UP FOR THE ATTACHIO CALL.

```
DSTAT:=ATTACHIO (LDEV,QMISC,DSTX,FUNC,CNT,P1,P2,FLAGS);

IOCDPN0 EQUIVALENCES:          L  --> LDEV
                               Q  --> QMISC
                               F  --> FUNC
                               C  --> CNT
                               PA --> P1
                               PB --> P2
                               FL --> FLAGS
```

C	COUNT	0	C128

SET THE COUNT IN THE ATTACHIO PARAMETER. NOTE THAT USER'S I/O BUFFER IS 4100 WORDS MAXIMUM.

F	FUNCTION CODE	0	F2

SET THE FUNCTION CODE ATTACHIO PARAMETER.

```
GENERAL FUNCTION CODES ARE:    0 -- READ
                               1 -- WRITE
                               2 -- FILE OPEN
                               3 -- FILE CLOSE
                               4 -- DEVICE CLOSE
                              >5 -- DEVICE DEPENDENT
```

FL	FLAGS	1	FL%11

SET THE FLAG WORD IN THE ATTACHIO PARAMETER.

```
FLAGWORD (12:1) -- SYSTEM BUFFER FLAG
         (13:3) -- REQUEST TYPE (0-7)
                   [1=BLOCKED,REST=UNBLOCKED]
```

L	LDEV	ERROR	L7

SET LOGICAL DEV# IN THE ATTACHIO PARAMETER.

PA	P1	0	PA1

SET P1 IN THE ATTACHIO PARAMETER, WHERE P1 IS DEVICE DEPENDENT.

PB	P2	0	PB%22

SET P2 IN THE ATTACHIO PARAMETER, WHERE P2 IS DEVICE DEPENDENT.

Q	QMISC	0	Q1

SET QMISC IN THE ATTACHIO PARAMETER, WHERE QMISC IS DEVICE DEPENDENT.

ATTACHIO EXECUTION RELATED FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
E	EXECUTION COUNT	1	E5

EXECUTES THE ATTACHIO CALL FOR THE COUNT SPECIFIED. NO COUNT SPECIFED, EXECUTION DONE ONCE ONLY.

CH			CH

CHECK RESULTS OF UNBLOCKED I/O. CHECK IF ANY I/O PENDING. IF ANY I/O PENDING, A PENDING MESSAGE WILL BE PRINTED. IF ANY ERRORS, STATUS WILL BE PRINTED.

DS			DS

DISPLAY LAST STATUS AND LOG RESULTING FROM THE ATTACHIO CALL. IF ANY I/O PENDING, A PENDING MESSAGE WILL BE PRINTED.

N	0/1 (OFF/ON)	0	N1

NULL INPUT FLAG SET EXECUTES ONE ATTACHIO CALL IF A NULL INPUT IS ENCOUNTERED.

T			T

DISPLAY THE AVERAGE EXECUTION TIME IN THE LAST ATTACHIO CALL. BY AVERAGE, IT IS MEANT AN AVERAGE IS TAKEN IF EXECUTION DONE MORE THAN ONCE. (IE. E20)

V	0/1 (OFF/ON)	0	V1

PRIOR TO EXECUTING ATTACHIO CALL, VFLAG SET PLACES THE EXECUTION COUNT INTO THE FIRST WORD OF THE USER'S I/O BUFFER. ON AN ATTACHIO WRITE OF 5 RECORDS, ONE WOULD SEE THE NUMBER 5 ON THE FIRST WORD OF THE FIFTH RECORD WHEN READ BACK.

STORED PROGRAM, DO FILE RELATED FUNCTIONS:

A MAXIMUM OF 500 WORDS ARE ALLOCATED WITHIN THIS PROGRAM FOR STORING COMMANDS WHICH THE USER WANTS TO SAVE FOR LATER EXECUTION. THIS IS CALLED THE STORED PROGRAM. WHEN A STATEMENT# IS ENTERED PRECEDING A COMMAND, THE LINE IS STORED IN THIS AREA.

EXAMPLE STATEMENT LINES:

```
10 L7
20 F0
30 C128
40 E20
```

THE STATEMENT NUMBER WITH STATEMENT ARE SEQUENTIALLY INSERTED INTO THE PROGRAM AREA. STATEMENT ENTERED WITH STATEMENT# AS ALREADY IN PROGRAM AREA WILL REPLACE THE PREVIOUS STATEMENT. A STATEMENT# ENTERED WITH NO STATEMENT WILL DELETE THE EXISTING STATEMENT IN THE PROGRAM AREA.

OP	PARAMETER	DEFAULT	EXAMPLE
CL	FIRST,LAST	ALL	CL20,40

CLEAR THE STORED PROGRAM STARTING FROM THE FIRST STATEMENT# TO THE LAST STATEMENT# ENTERED. IF NO PARAMETER, CLEAR THE ENTIRE PROGRAM AREA.

GO	STATEMENT#	1ST STATEMENT#	GO30

BEGIN EXECUTING STORED PROGRAM STARTING AT THE STATEMENT# ENTERED.

LI	FIRST,LAST	ALL	LI10,50

LIST THE STORED PROGRAM STARTING FROM THE FIRST STATEMENT# TO THE LAST STATEMENT# ENTERED. IF NO PARAMETER, THE ENTIRE STORED PROGRAM IS LISTED.

R	RUN COUNT	1	R5

EXECUTE THE STORED PROGRAM FOR THE NUMBER OF TIMES SPECIFIED IN THE COUNT.

DO	FILENAME		

GET COMMANDS AND/OR PROGRAM FROM A DO FILE. COMMANDS GOTTEN ARE EXECUTED IMMEDIATELY WHEREAS PROGRAM STATEMENTS ENCOUNTERED ARE STORED INTO THE PROGRAM AREA. DO FILES CAN BE SEEN AS A PSEUDO-BATCH TYPE OPERATION.

NOTE: COMMANDS AND/OR STORED PROGRAMS (STATEMENT# PRECEDING COMMANDS) CAN BE ENTERED INTO A FILE UNDER THE 3000 EDITOR. SUCH A FILE IS REFERRED TO BY BY IOCDPN0 AS A DO FILE.

SH	0/1 (OFF/ON)	1	SH1

WITH SHOW FLAG SET, EACH COMMAND AND/OR (PROGRAM STATEMENT) GOTTEN FROM A STORED PROGRAM OR A DO FILE ARE PRINTED ON THE $STDLIST.

I/O INSTRUCTION FUNCTIONS:

[IMPLEMENTED ON THE SERIES 2 & 3 ONLY]

OP	PARAMETER	DEFAULT	EXAMPLE
CIO	CONTROL WORD	ERROR	CIO%140000
	PERFORM CONTROL I/O INSTRUCTION TO DEVICE WITH THE INPUT CONTROL WORD.		
RIO			RIO
	PERFORM READ I/O INSTRUCTION TO DEVICE. IF DEVICE READY, DATA WORD PRINTED. IF DEVICE NOT READY, DEVICE STATUS IS PRINTED.		
TIO			TIO
	PERFORM TEST I/O INSTRUCTION TO OBTAIN THE STATUS FROM THE DEVICE FOR PRINTING.		
WIO	WRITE DATA	ERROR	WIO1
	PERFORM WRITE I/O INSTRUCTION WITH DATA WORD TO THE DEVICE. IF DEVICE READY, DATA WORD IS TRANSMITTED. IF DEVICE NOT READY, DEVICE STATUS IS PRINTED.		

TERMINAL MONITORING FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
CM			CM
	CLEAR MONITORING AREA. USED IN CONJUNCTION WITH COMMANDS "M" AND "P". SINCE MONITORING ENTRIES ARE KEPT IN A SYSTEM BUFFER, THIS FUNCTION ZERO THIS SBUF AREA.		
M	MCODE	0	M%22
	SET TERMINAL MONITORING CODE. IN A TERMINAL DIT IN WORD %41, THE LAST 6 BITS APPLIES TO MONITOR FUNCTION AND CONTROL CODE. THESE ARE SET ACCORDINGLY BY TH MCODE INPUT FOR MONITORING TERMINAL ACTIVITIES. USED IN CONJUNCTION WITH COMMANDS "CM" AND "P".		
P	#ENTRIES	ALL	P
	PRINT TERMINAL MONITOR TABLE FOR NUMBER OF ENTRIES SPECIFIED (0-32). NO PARAMETER RESULT IN ALL ENTRIES PRINTED. ENTRIES WILL NOT EXIST OR BE PRINTED UNLESS THE MONITORING CODE WAS PREVIOUSLY SET. USED IN CONJUNCTION WITH COMMANDS "CM" AND "M".		

I/O TABLE RELATED FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
DD	OFFSET,COUNT	ALL	DD

DISPLAY THE DEVICE INFORMATION TABLE (DIT) FOR THE LDEV. CAN DISPLAY DIT FROM A GIVEN OFFSET FOR A LENGTH OF COUNT OR DISPLAY ENTIRE DIT WITH NO PARAMETER SPECIFIED. LDEV IS SET BY COMMAND "L LDEV#".

DQ	NO.OF IOQS	ALL	DQ

DISPLAY THE I/O QUEUE ELEMENT (IOQ) UP TO THE NUMBER OF IOQ ENTRIES ENTERED. NO PARAMETER DISPLAY ALL IOQ ENTRIES CURRENTLY PENDING.

DSIO	OFFSET,COUNT	ALL	DSIO

DISPLAY THE SIO OR CHANNEL PROGRAM FOR THE LDEV STARTING AT A GIVEN OFFSET FOR A LENGTH OF COUNT. NO PARAMETERS DISPLAY THE ENTIRE SIO OR CHANNEL PROGRAM.

MD	OFFSET,COUNT	ERROR	MD3,1

MODIFY THE DIT TABLE STARTING AT A GIVEN OFFSET FOR A LENGTH OF COUNT. THE DIT IS THE ONE ASSOCIATED WITH LDEV, SET BY THE COMMAND "L LDEV#".

MQ	OFFSET,ERROR	ERROR	MQ0,3

MODIFY THE FIRST IOQ ENTRY STARTING AT A GIVEN OFFSET FOR A LENGTH OF COUNT. THE IOQ ENTRY IS THE ONE ASSOCIATED WITH LDEV, SET BY THE COMMAND "L LDEV#".

UTILITY FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
DE			DE

ENTER INTO THE MPE FACILITY DEBUG.

H			H

ENTER INTO MPE LOW LEVEL SYSTEM DEBUGGER CALLED HELP.

MISCELLANEOUS FUNCTIONS:

OP	PARAMETER	DEFAULT	EXAMPLE
AN			AN"HELLO"
	DISPLAY THE ASCII STRING INPUT ON THE LIST DEVICE.		
LD	0,1,2+	$STDLIST	LD6
	REDIRECT THE LIST DEVICE TO THE $STDLIST, A FORMAL DESIGNATOR LIST, OR ANY LIST DEVICE LDEV#.		
DRT	DRT#	ERROR	DRT16
	ENTRY OF DRT# TO OVERRIDE DRT# PICKED UP BY COMMAND "L LDEV#" WHICH GOT ITS DRT# THRU THE ILT TABLE. USED IN CONJUNCTION WITH THE I/O INSTRUCTION FUNCTIONS, ALLOW ACCESS TO DEVICE WITH INACCESSIBLE DRT# (IE. CAN DO A CIO TO A TERMINAL'S TCI BOARD).		
DEC	PARAM	ERROR	DEC%100
	CONVERT THE INPUT PARAMETER TO BE REPRESENTED AS A DECIMAL VALUE.		
OCT	PARAM	ERROR	OCT200
	CONVERT THE INPUT PARAMETER TO BE REPRESENTED AS AN OCTAL VALUE.		
X			
	LIST EXPLANATIONS OF ALL THE COMMANDS.		
EX			EX
	TERMINATE PROGRAM AND EXIT.		
<C>	<MPE COMMANDS>		
	MPE COMMANDS ENTERED ARE PASSED TO THE COMMAND INTRINSIC FOR EXECUTION.		

*** IOCDPN0 ***
PROGRAM INTERNAL INFO

SYNTAX OF PARAMETERS -

```
EXPRESSION ::= TERM ! TERM + TERM ! TERM - TERM ! - TERM
TERM       ::= FACTOR ! FACTOR * FACTOR ! FACTOR / FACTOR
FACTOR     ::= CONSTANT ! ( EXPRESSION )
CONSTANT   ::= DEC NUMB ! "%" OCT NUMB ! "CHAR" ! "CHARCHAR"
```

PARAMETERS ARE DELIMITED BY A COMMA OR CARRIAGE RETURN
SIGN IS ALLOWED BEFORE DECIMAL VALUES, I.E. [-89],
BUT NOT BEFORE OCTAL VALUES, I.E. [%177657].

SPACES ARE OPTIONAL AND IGNORED ((COMMENTS)) ARE ALLOWED.

STORED PROGRAM FORMAT -

```
WORD# 0  STATEMENT NUMBER IN BINARY
WORD# 1  STATEMENT LENGTH IN BYTES
WORDS 2  THRU N STATEMENT IN ASCII
```

NEXT STATEMENT IS FOUND BY ADDING STATEMENT (LENGTH+5)/2
TO CURRENT STATEMENT POINTER.
STATEMENTS ARE STORED IN ORDER OF STATEMENT NUMBER. EACH
STATEMENT OCCUPING A BLOCK AS DESCRIBED ABOVE.
LAST STATEMENT IS DENOTED BY HAVING A STATEMENT NUMBER OF
10000 AND A STATEMENT LENGTH OF ZERO.

COMMAND RECOGNITION LIST -

COMMANDS ARE 1 OR MORE LETTERS WITH OPTIONAL SPACE,
PARAMETERS ARE NUMBERS OR STRINGS IN QUOTES,
PROGRAM STATEMENTS ARE PRECEDED BY A STATEMENT NUMBER.

A	ASCII BUF FILL	AN	DO ANOTE
B	OCTAL BUF FILL	C	SET COUNT PARAM
CH	CHECK RESULTS	CIO	DO CIO INSTR
CL	CLR PROG AREA	CM	CLR MNTR TABLE
D	DISPLAY PARAMS	DA	SHOW BUF ASCII
DB	SHOW BUF OCTAL	DE	CALL DEBUG
DEC	DECIMAL CONVERT	DD	DISPLAY DIT
DRT	SET DRT#	DO	READ CMND FILE
DQ	DISPLAY IOQS	DS	DISPLAY STATUS
DSI	DISPLAY SIO PROG	E	CALL ATTACHIO
EX	TERMINATE	F	SET FUNCTION
FL	SET FLAGS	GO	GOTO STATEMENT#
H	CALL HELP	I	INCREMENT FILL
L	SET LDEV	LD	SET LIST DEV
LI	LIST PROGRAM	M	MONITOR TERM
MB	MODIFY BUFFER	MD	MODIFY DIT
MQ	MODIFY IOQ	N	NULL INPT DO E
O	OCTAL CONVERT	P	PRT MNTR TABLE
PA	SET P1	PB	SET P2
Q	SET QMISC	R	RUN STORED PROG
RIO	DO RIO INSTR	SH	PRINT DO & PROG
T	PRINT TIME	TIO	DO TIO INSTR
V	ADD REC# IN BUF	WIO	DO WIO INSTR
X	EXPLAIN CMDS		

*** IOCDPN0 ***

HOW TO USE IT

V * E * R * Y * * * C * A * R * E * F * U * L * L * Y

Some understanding of "ATTACHIO" is required, in order to use this UTILITY inteligently. Basically all it does for you, is allow you an interactive means of entering parameters and calling this SYSTEM INTERNAL PROCEDURE. In so doing, you are bypassing the MPE FILESYS and all the normal checks for advisability and appropriateness of your requests. It does give the knowlegable user a very powerfull tool in diagnosing I/O related problems. With very little effort you can access information which is otherwise apparent only to the operating system. You can pick a single card, look at hardware device status, read and check Labels on LABELED TAPES, check the formatting in SECTOR PREAMBLE areas on disk, check STORE TAPES for proper formatting, etc. In short, you can get at anyhing that the system can get at.

EXAMPLE

READING A TAPE LABEL

```
11:12/#S15/20/LOGON FOR: MANAGER.SYS,PUB ON LDEV #20
:RUN IOCDPN0
11:13/14/VOL MYTAPE(ANSI) MOUNTED ON LDEV# 7

TEST IO   (B.01.01)   TYPE X FOR COMMAND LIST
C?L 7                                  << SELECT LDEV 7  >>
C?F 0                                  << FUNCTION = READ >>
C?C 100                                << REQUEST 100 WDS >>
C?FL 1                              << SET WAIT - MODE FLAG >>
C?E                                   << POST REQ AND WAIT >>
C?DA                               << DUMP DATA READ IN ASCII >>
VOL1MYTAPE
              1
C?E                                    << GET NEXT RECORD >>
C?DA
HDR1EDT      .        MYTAPE00010001       80010 80321     0HP M
PE 3000
C?E                                      << AND NEXT >>
C?DA
HDR2F   80   80

C?E
LDEV = 7, FUNC = 0, XLOG = 0, STATUS = 12      ( EOF )
C?
                                      ( this is the I/O REQ. STATUS )
                                      ( not to be confused with any )
                                      ( hardware device status word )
C?
C?F 5    << REWIND & DONT UNLOAD >>
C?E
C?EX            << EXIT >>
```

*** IOCDPN0 ***

EXAMPLE OF USING PREPARED PROGRAM

```
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)   TYPE X FOR COMMAND LIST
C?DO LPRIPPLE.IOCDPN0.ZIEGLER
C?LI
10   ANOTE "     ** LPRIPPLE **"
20   ANOTE                                        ( canned routine )
30   ANOTE "TYPE 'R NN' TO LOOP NN TIMES"         ( prints 2 pages )
40   L  6                                         ( of 132 byte/ln )
50   FL 1                                         ( ripple-pattern )
60   PA 0                                         ( with no auto-  )
70   PB 0                                         ( skip over perf )
80   ANOTE "MOVE PRINTER TO TOP OF FORM"
90   F  2       <<  FILE OPEN  >>
100  E
110  ANOTE "  SET NO SPACE OVER PAGE"
120  PB  2
130  ANOTE "  RIPPLE PRINT -2 PAGES-"
140  C  66
150  A       <<  DEFAULT = RIPPLE  >>
160  F  1
170  E  132
180  PB 0
190  F  3       <<  FILE CLOSE >>
200  E
C?R
     ** LPRIPPLE **                              ( run prog as is )

TYPE 'R NN' TO LOOP NN TIMES
MOVE PRINTER TO TOP OF FORM
  SET NO SPACE OVER PAGE
  RIPPLE PRINT -2 PAGES-
C?170 E 20                                       ( change program )
C?110                                            ( to suit needs  )
C?120
C?130 ANOTE "  RIPPLE PRINT 20 LINES"
C?R
     ** LPRIPPLE **                              ( run as modified)

TYPE 'R NN' TO LOOP NN TIMES
MOVE PRINTER TO TOP OF FORM
  RIPPLE PRINT 20 LINES
C?EX

END OF PROGRAM
```

*** IOCDPN0 ***

USING EDITOR TO PREPARE A PROGRAM

It is a fairly simple thing to prepare a set of your own "DO" files from which you can quickly design your own specific device tests or excercisers. Use lots of "<< COMENTS >>" to jog your memory, when you need to use the same routine at a much later date.

```
:EDITOR
HP32201A.7.07 EDIT/3000  THU, JUN 12, 1980,  8:39 PM
(C) HEWLETT-PACKARD CO. 1979
/A
    1      10   ANOTE  "OUTHARD"
    2      20   L  6      <<  USES A HARD PRE-EMPTIVE  >>
    3      30   PA        <<  REQUEST TO TRY TO BREAK  >>
    4      40   PB        <<  LOOSE A HUNG-UP PRINTER  >>
    5      50   FL %402
    6      60   F  1      <<  IF LP TAKES OFF RUNNING  >>
    7      70   C  1      <<  AFTER THIS, THE CAUSE OF >>
    8      80   A  "  "   <<  THE HANG WAS A MISSING   >>
    9      90   E         <<         INTERUPT          >>
   10      //
...
/
/K OUTHARD,UNN
/E

END OF SUBSYSTEM
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)   TYPE X FOR COMMAND LIST
C?DO OUTHARD
C?LI
10   ANOTE  "OUTHARD"
20   L  6      <<  USES A HARD PRE-EMPTIVE  >>
30   PA        <<  REQUEST TO TRY TO BREAK  >>
40   PB        <<  LOOSE A HUNG-UP PRINTER  >>
50   FL %402
60   F  1      <<  IF LP TAKES OFF RUNNING  >>
70   C  1      <<  AFTER THIS, THE CAUSE OF >>
80   A  "  "   <<  THE HANG WAS A MISSING   >>
90   E         <<         INTERUPT          >>
C?R
OUTHARD
C?EX

END OF PROGRAM
```

*** IOCDPN0 ***

DEVELOPING, KEEPING AND DOCUMENTING "DO" PROGRAMS

Program files do not have to be self contained, whatever parameters you have set from keyboard remain unchanged unless the program resets them to new values. You can develop "DO" routines interactively to do repetive tasks, and retain specific control on keyboard. You can use a :FILE equate for "LIST" along with the "LD 1" command to save programs developed and checked ON-LINE.

```
:FILE LIST=DOFILE,NEW;DEV=DISC;REC=-72,16,F,ASCII;SAVE
:
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)    TYPE X FOR COMMAND LIST
C?L 3
C?F 0
C?PB (2*5+2)*48     << CYL-2 * 5 HEADS/CYL + HEADS 0 & 1 >>
C?PA                << * 48 SECS/HED = CYL-2,HED-2,SEC-0 >>
C?
C?10 ANOTE "UNIT# & LDEV# 2 BYTES"
C?20 DD 3
C?30 E
C?40 ANOTE "LOGICAL DISK ADDRESS"
C?50 DD 10,2  << VOLUME RELATIVE SECTOR ADD  %PA %PB >>
C?60 ANOTE "PHYSICAL DISK ADDRESS"
C?70 DD 12,2  << XXXXXXXXCCCCCCCC  HHHHHHHHSSSSSSSS  >>
C?80 ANOTE "LAST DISK STATUS RETN"
C?90 DD 18,2  << DEVICE STATUS WORD # 1 AND WORD # 2 >>
C?100 ANOTE "DISK INTERFACE STATUS"
C?110 DD 6,2  << TIO OR SIO CTRLR STAT & ERROR STAT  >>
C?LI
10    ANOTE "UNIT# & LDEV# 2 BYTES"
20    DD 3
30    E
40    ANOTE "LOGICAL DISK ADDRESS"
50    DD 10,2  << VOLUME RELATIVE SECTOR ADD  %PA %PB >>
60    ANOTE "PHYSICAL DISK ADDRESS"
70    DD 12,2  << XXXXXXXXCCCCCCCC  HHHHHHHHSSSSSSSS  >>
80    ANOTE "LAST DISK STATUS RETN"
90    DD 18,2  << DEVICE STATUS WORD # 1 AND WORD # 2 >>
100   ANOTE "DISK INTERFACE STATUS"
110   DD 6,2   << TIO OR SIO CTRLR STAT & ERROR STAT  >>
C?R
UNIT# & LDEV# 2 BYTES
041421: 001003
LOGICAL DISK ADDRESS
041430: 000000  001100
PHYSICAL DISK ADDRESS
041432: 000002  001000
LAST DISK STATUS RETN
041440: 000001  001040
DISK INTERFACE STATUS
041424: 100002  000000
C?
C?LD 1       << LIST TO "LIST" DESIGNATOR >>
C?LI
C?LD      << CLOSE FILE & RETURN TO $STDLIST >>
C?
```

DEVELOPING, KEEPING AND DOCUMENTING "DO" PROGRAMS

Now we can check that the file was created, :RENAME it, and use it on a different LDEV. This is a 7925 DISC so I enter the LDEV, PA, PB, & FUNCT accordingly.

```
C?
C?                                                    (break)
:LISTF DOFILE,2
ACCOUNT=  ZIEGLER      GROUP=  IOCDPN0

FILENAME  CODE  ------------LOGICAL RECORD------------  ----SPACE---
                  SIZE  TYP         EOF       LIMIT R/B  SECTORS #X M

DOFILE             72B  FA           11        1023  16       45  1


:RENAME DOFILE,DDISKST                              ( only needed if  )
:RESUME                                             ( file equate is  )
READ PEND'NG                                        ( to be re-used.  )

C?CL
C?LI
NO PROGRAM
C?L 2                                               ( set LDEV, FUNCT )
C?F 0                                               ( COUNT & ADDRESS )
C?C 4096                                            ( relative sector )
C?PB (4*9+3)*64+7     << CYL-4, HD-3, SEC-7 >>
C?PA
C?DO DDISKST                                        ( use "DO" program)
C?R                                                 ( to get pertinant)
UNIT# & LDEV# 2 BYTES                               ( info from DIT   )
041360: 000402
LOGICAL DISK ADDRESS
041367: 000000  004707
PHYSICAL DISK ADDRESS
041371: 000004  001407
LAST DISK STATUS RETN
041377: 000000  003040
DISK INTERFACE STATUS
041363: 100001  000000
C?
C?LD 6   << LIST TO LINE PRINTER TO DOCUMENT >>
C?LI
C?EX

END OF PROGRAM
:
```

*** IOCDPN0 ***

OTHER TRICKS AND TWIDDLES

You can usually clear up a HUNG-PORT condition, even if there is no SESSION on the PORT and an :ABORTIO won't clear it. Say for example that LDEV 24 is reported to be HUNG and I would like to avoid a COOLSTART right now.

```
:SHOWJOB

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#S186   EXEC        20  20       FRI 11:10A  OPERATOR.SYS
#S200   EXEC        24  24       THU  4:56P  FIELD.SUPPORT
#S174   EXEC        23  23       FRI  9:21A  JOSEPH.ZIEGLER

3 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    3 EXEC; INCL 3 SESSIONS
    0 SUSP
JOBFENCE= 0; JLIMIT= 4; SLIMIT= 16

:SHOWDEV 24
LDEV  AVAIL     OWNERSHIP       VOLID            ASSOCIATION

 24 A UNAVAIL   #S200: 2 FILES
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)   TYPE X FOR COMMAND LIST
C?L 25
C?DD 0,16   << LOOK AT FIRST PART OF DIT FOR FREE PORT >>
042170: 100400  000000  000000  002431  177144  040654  000000  00122
042200: 000000  014000  005602  000000  000000  000000  000000  00000
C?L 24
C?DD 0,16   << COMPARE WITH DIT FOR HUNG PORT >>
042124: 140402  000000  034443  002030  177144  040654  000000  00522
042134: 002400  010222  004662  000415  000000  001000  000000  00000
C?
C?FL %420   << POST Q AS HARD PRE-EMPT, SPECIAL HANDLING, NO-WAIT >>
C?F  4                      << DO A DEVICE CLOSE >>
C?E
C?DD 0,16   << NOW CHECK FIRST PART OF DIT AGAIN >>
042124: 100600  000000  000000  002030  177144  040654  000000  00522
042134: 002000  014000  004702  000000  000000  000000  000000  00000
C?
:SHOWDEV 24
LDEV  AVAIL     OWNERSHIP       VOLID            ASSOCIATION

 24 A AVAIL
:RESUME
READ PENDING

C?DD 0,16   << CHECK DIT AGAIN AFTER SPEED-SENSE >>
042124: 140402  000000  033273  002030  177144  040654  000000  00522
042134: 002400  010202  004702  000364  000000  002000  000000  00000
C?EX

END OF PROGRAM
```

OTHER TRICKS AND TWIDDLES

Using the DIRECT I/O commands in IOCDPN0, i.e. TIO, CIO, WIO, & RIO, the user can circumvent not only the FILESYS but also the I/O MONITOR and DRIVERS and go directly to the DEVICE CONTROLLER. This, of cours greatly increases the danger of disrupting normal system activity, bu for the knoledgable user, it increases the power of this programmatic trouble-shooting tool proportionally.

Note: Curent version only includes DIRECT I/O INSTRUCTIONS for the SER II/III & SER III-LC. They are invalid on MOD 30/33.

Note: All DIO activity takes place in this CODE and DRT #s and DATA/ COMMANDS are transfered to/from this PROGRAMS STACK. Unless otherwise specified the DRT used is that associated with LDEV#

Ref: Machine Instruction Set Manual for details of DIO
Ref: 3000 SER II/III CE HANDBOOK, PERIPHERALS SECT. for format of COMMAND & STATUS WORDS.

OTHER TRICKS AND TWIDDLES

Using the "DRT" command, you can establish a DRT NUMBER to be used in the execution of the DIRECT I/O COMMANDS independent of any LDEV# set up by the "L" command. **

This makes it possible to do DIRECT I/O to device controllers which do not show up in the I/O CONFIGURATION and which have no LOGICAL DEV ICE NUMBER associated with them.

Typical of these "UNLISTED" device controllers are:

	SER II & III	SER III-LC	
FAULT LOGGING INTERFACE	FLI	CLK/FLI	DRT#2
SYSTEM CLOCK	SYS CLK/CONS *	CLK/FLI	DRT#3
TERMINAL CONTROL INTERFACE	TCI	TCI	DRT#8 **
	TCI	TCI	DRT#9 **

Notes: * On the 2s & early 3s there was a controller for a hardwired terminal (i.e. CONSOLE) along with CLOCK function on the SYS CLK / CONSOLE PCA. This was unused by the OP/SYS on MPE-B but could be accessed as the "HELP-TERMINAL" by means of a special section of code left in CLOCKIO software for LOW LEVEL SYSTEM DEBUGGING.

** Possibly 1 or 2 TCI PCAs associated with any TDI CARD for use of MODEM SUBTYPES 1,2 or 3. These must be assigned to the next DRT #s immediately following the associated TDI PC

EXAMPLE:

Line printer "HUNG" in the middle of output. STOPSPOOL has no effect. ABORTIO X, no effect. No "LDEV #X NOT READY" message. Printer toggled OFF & ON-LINE, still no message.

Happened before and user had to do a WARMSTART to clear it. Settin TAPE DRIVE to ON-LINE no attempt to read LABEL or issue "VOLUME XXX MOUNTED" message.

Disc access is ok, so the INT POLL is good down to DISC INTRFC. RJELINE is working, so the MUX CHAN is at least alive.

To find out if one of the TCI cards is intermittantly failing to re spond to the CLEAR INTERUPT signal generated by IXIT instruction an as a result, has blocked the INTERUPT POLL for all controllers belo it, we want to issue a MASTER-CLEAR to each TCI in turn.

```
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)    TYPE X FOR COMMAND LIST
C?DRT 8              << SET DRT# OF FIRST TERM.CONTRL.INTFC "TCI" >>
WARNING:  DRT# CHANGED ONLY--OVERRIDES LDEV
C?CIO %100000        << ISSUE DIRECT COMMAND I/O - MASTER CLEAR >>
C?DRT 9              << SET DRT# OF SECOND TERM.CONTRL.INTFC "TCI" >>
WARNING:  DRT# CHANGED ONLY--OVERRIDES LDEV
C?CIO %100000
C?L 24
WARNING:  LDEV OVERRIDES DRT#
C?EX
```

OTHER TRICKS AND TWIDDLES

Here are some examples of canned "DO" programs which I use to look for problems on SYSDUMP or STORE tapes. I know that there are several tape analyzer programs around, but they operate under the optomistic presumption that the FORMAT of the tape is correct.

To really get a handle on what is wrong, and to be able to offer an informed guess as to how it got that way, I prefer the more direct form of analysis and the added flexibility offered by this approach.

TPDIRC - Skips over 2 FILEMARKS which should be the OP-SYS on a SYSDUMP or NULL-FILES on a STORE TAPE; then reads and dumps the next 2 FILES which should be the STORE/RESTORE HEADER and the TAPE DIRECTORY.

```
10   A   << PRINTS DIRECTORY FOR SYSDUMP/STORE >>
20   A            << TAPE REEL # 1 >>
30   ANOTE  "TPDIRC   ( HIT CONT-Y TO STOP )"
40   FL 1
50   PA
60   PB
70   L  6
80   F  2    << FOPEN  TOP OF FORM >>
90   E
100  LD 6
110  L  7
120  C  1024
130  F  5      << REWIND  ON-LINE >>
140  E
150  A  << ***** DELETE 160 & 170 FOR REEL#2 AND UP ***** >>
160  F  7
170  E  2      << SKIP TWO FILES  >>
180  B          <<  CLEAR BUFFER  >>
190  F  0  << GET STORE/RESTORE LABEL >>
200  E
210  DA 0,17   << PRINT ASCII  >>
220  DB         << PRINT BINARY >>
230  LD
240  DA 0,17   << DISP HEADER  >>
250  B
260  ANOTE  " "
270  ANOTE  "  HIT CONTROL-Y AFTER NEXT 'STATUS = 012'"
280  LD 6
290  F  7
300  E
310  F  0
320  C  4096
330  B
340  E
350  DA
360  GO 320
370  F  0
380  LD 6
390  C  1024
400  B
410  F  0
420  E
430  DA
440  GO 400
```

*** IOCDPNO ***

TPSCAN

TPSCAN - Skips over 4 FILEMARKS and prints the FIRST 32 BYTES of the FIRST RECORD in each file, this should be the FILE NAME for each file or the STORE/RESTORE TRAILER.

```
10   A  <<  SCAN FOR FILE LABELS & END OF REEL  >>
20   A  <<  MARK ON STORE OR SYSDUMP REEL # 1   >>
30   B
40   ANOTE  "TPSCAN   ( HIT CONTROL-Y TO STOP )"
60   PA
70   PB
80   FL 1
90   L  6          << LDEV=6 LP >>
100  F  2  << FORCE TOP OF FORM ON PRINTER >>
110  E
120  LD 6     << SET OUTPUT TO PRINTER >>
130  L  7        << LDEV=7 TAPE >>
140  F  5      << REWIND  ON-LINE >>
150  E
160  F  7      << SKIP 4 FILE MARKS >>
170  E  4  << ***** DELETE STEP 170 FOR REEL # 2 AND UP ***** >>
180  LD
190  ANOTE  " "
200  ANOTE  "  HIT CONTROL-Y AFTER NEXT STORE/RESTORE"
210  C  20 << FIRST PART OF RECORD ONLY >>
220  B
230  F  0      << READ 20 WORDS >>
240  E
250  LD 6
260  DA 0,16       << PRINT >>
270  LD
280  DA 0,16       << DISPL >>
290  F  7   << SPACE TO NEXT FILE >>
300  E
310  GO 220        << LOOP >>
```

TPSCANBK

TPSCANBK - Scans backward printing the FIRST 32 BYTES of the FIRST RECORD of each FILE.

```
10    A  <<  SCANS BACKWARD LOOKING FOR FIRST   >>
20    A  <<  VALID STORE/RESTORE LABEL ON TAPE  >>
30    B
40    ANOTE  "TPSCANBK ( HIT CONTROL-Y TO STOP )"
50    PA
60    PB
70    FL 1
80    L  6
90    F  2   << FOPEN TO LP DO TOP OF FORM >>
100   E
110   LD 6   << SET OUTPUT TO LINE PRINTER >>
120   L  7         << SELECT TAPE >>
130   C  100
140   F  8       << BACKSPACE FILEMARK >>
150   E
160   F  7       << SPACE FWD OVER MARK >>
170   E
180   B           << ZERO OUT BUFFER >>
190   C  100
200   F  0       << READ FIRST RECORD >>
210   E
220   DA 0,16
230   DB 0,40    << PRINT ON PRINTER >>
240   LD
250   DA 0,16      << DISP ON TERM >>
260   DB 0,40
270   F  8       << BACKSPACE ANOTHER >>
280   E  2
290   GO 160
```

*** IOCDPN0 ***

MONITORING TERMINAL AND MODEM ACTIVITY

Using the three commands "CM","M", & "P" it is possible to monitor the activity on any TERMINAL on the system. These are handy to check for activity on the modem control and status lines to TCI.

MONITOR CODE is the LOW ORDER 6 BITS of WORD #33 or %41 of TERM DIT
Ref: MPE TABLES MAN. pp-13-33 for DMONTR meanings.

IOCDPN0 FORM is (M % x y)

x = 1 - Monitor DSET1,DSET2, & DSETCONTROL
2 - Monitor TIP
4 - Monitor TERM

y = 0 - Null or none
1 - Call "HELP"
2 - Monitor Activity
3 - Form DELTA-TIME HISTOGRAM
7 - Monitor CALLS/COUNTS/INITS

EXAMPLE:

```
:RUN IOCDPN0.PUB.SYS

TEST IO  (B.01.01)   TYPE X FOR COMMAND LIST
C?L24
C?DD
044124: 140402  000000  041564  002030  177144  042654  000000  005224
044134: 002400  010121  004662  000415  000000  001000  000000  000000
044144: 000000  000000  177777  000000  032250  177777  000000  012000
044154: 000000  000000  000000  000000  000000  000047  000000  000000
044164: 000000  040023  000000  000000
C?CM
C?M %22
C?P
                                          MODEM T L R B P D Q
TIME  ID  DS UNIT  P1     P2     MNTR ABSFC M P P R R F F FNC P1 STAT RQ

99999 TPR  RD  4 140402 021523 U      1
8024  TPR  RD  4 140002 023415 U      1
1     TPW  RD  4 140002 000012 U      1
305   TPW WRT  4 141401 000015 U      1
5     TPW WRT  4 141401 000012 U      1
126   TPW WRT  4 141401 000015 U      1
4     TPW WRT  4 141401 000012 U      1
14    TPW WRT  4 141401 000005 U      1
11    TPR WRT  4 140421 021406 U      1          B
152   TPW WRT  4 141401 000015 U      1
5     TPW WRT  4 141401 000012 U      1
166   TPW WRT  4 141401 000015 U      1
4     TPW WRT  4 141401 000012 U      1
27    TPW WRT  4 141401 000005 U      1
11    TPR WRT  4 140421 021406 U      1          B
179   TPW WRT  4 141401 000005 U      1
10    TPR WRT  4 140421 021406 U      1          B
19    TPS WRT  4 002400        U      1                                 00
C?
```

*** IOCDPN0 ***

MONITORING TERMINAL AND MODEM ACTIVITY
==========
(CONT.)

```
C?L24
C?CM
C?M %12
C?P
                                                    MODEM T L R B P D Q
TIME  ID  DS UNIT  P1      P2      MNTR ABSFC M P P R R F F FNC P1 STAT RQ

99999 TMI NUL  4                  UA     1                        0 003      04
1     TMI NUL  4                  UA     1          H             31 001      00
0     TMI NUL  4                  UA     1        H H             31 001      00
1     TMO NUL  4                  UA     1      C H H             31      001 00
8     TMI NUL  4                  UA     1      C H H              1 320      00
1     TMI NUL  4                  UA     1      C H H              1 320      00
1     TMO WRT  4                  UA     1      C H H              1      001 00
2     TMI WRT  4                  UA     1      C H H              0 005      00
0     TMI WRT  4                  UA     1      C H H              0 005      00
2520  TMI NUL  4                  UA     1      C H H              0 005      04
1     TMO NUL  4                  UA     1      C H H              0      001 04
1     TMI NUL  4                  UA     1      C H                0 003      00
11    TMI NUL  4                  UA     1      C H S              1 000      00
1     TMI NUL  4                  UA     1      C S S              1 000      00
2     TMO WRT  4                  UA     1      C S S              1      001 00
1     TMI WRT  4                  UA     1      C S                0 003      00
2     TMI WRT  4                  UA     1      C S C             31 000      00
56    TMI NUL  4                  UA     1      C S C             31 000      00
1     TMI NUL  4                  UA     1      C C C             31 000      00
0     TMO NUL  4                  UA     1        C C             31      001 00
1     TMI NUL  4                  UA     1        C                0 003      00
0     TMI NUL  4                  UA     1                         0 003      00
C?M %23
C?CM
C?P
                                                    MODEM T L R B P D Q
TIME  ID  DS UNIT  P1      P2      MNTR ABSFC M P P R R F F FNC P1 STAT RQ

0     TPR WRT  0 000000 000000
5     TPR ELF  0 000000 000000
0     TPR NUL  0 000003 000000
0     TPR NUL  0 000000 000000
0     TPR NUL  0 000000 000001
0     TPR NUL  0 000000 000000
C?EX

END OF PROGRAM
```

TERMINAL TYPE INFORMATION

TERM TYPE	SUPPORTED TERMINAL	CHARACTER DELAYS 10 CR LF	15 CR LF	30 CR LF	60 CR LF	120 CR LF	240 CR LF	ECHOED BS CHAR	FORM FEED	COMMENTS
0	ASR-33	1 0						\	NO	STANDARD TELETYPE SUPPORT
1	ASR-37		2 0					LF	YES	
2	ASR-35	1 0						\	YES	
3	EXECUPORT	1 0	3 0	5 0				LF	YES	GOOD ALSO FOR TI-700 SERIES, DECWRITER I, II
4	DATAPOINT	0 0	0 4	0 4	0 4	0 4	0 4	EM	NO	HP2600 TERMINAL
5	MEMOREX	10 1	10 1	25 1	45 1			LF	YES	
6	TERMINET	0 3	0 5	0 10		0 40		LF	YES	USE FOR DECWRITER III AT 1200 BAUD.
9	MINI BEE	0 0	0 4	0 4	0 4	0 4	0 4	NONE	YES	MOST "DUMB" CRTS
10	HP264x	0 0	0 0	0 0	0 0	0 0	0 0	NONE	YES	USES ENQ/ACK INSTEAD OF DELAYS
11	HP264x	0 0	0 0	0 0	0 0	0 0	0 0	NONE	YES	FULL "ENTER" SERVICE
12	HP2545K	0 0	0 0	0 0	0 0	0 0	0 0	NONE	YES	8-BIT KATAKANA TERMINAL PARITY NOT ALLOWED
13	TELENET	0 0	0 0	0 0	0 0	0 0	0 0	NONE	YES	ECHO TURNED OFF
14	MULTIPOINT	- - - - - NOT APPLICABLE - - - - - -							YES	
15	HP2635	0 0	0 0	0 0	0 0	0 0	0 0	LF	YES	USES 8TH BIT TO SELECT 2ND CHARACTER SET
16	HP2635A/B	0 0	0 0	0 0	0 0	0 0	0 0	LF	YES	STANDARD 2635 TERMTYPE
18	(ANY)	0 0	0 0	0 0	0 0	0 0	0 0	NONE	YES	"PROTOCOL-LESS" TERMTYPE
19	HP2631B	0 0	0 0	0 0	0 0	0 0	0 0	LF	YES	FOR REMOTE SPOOLED PRINTER
77	UNDEFINED	1 0	0 4	0 4	0 4	0 4	0 4	\	YES	USED WHEN NO TYPE SPECIFIED (SYSDUMP OR ;TERM=)

NOTES:

A. ALL TERMINAL TYPES (EXCEPT 18) TRANSMIT A DC1 (X-ON) TO THE TERMINAL TO START A READ.

B. TERMINAL TYPES 0 THROUGH 6 TRANSMIT A DC3 (X-OFF) TO THE TERMINAL BEFORE EACH CR TRANSMITTED AS PART OF SINGLE-SPACE CARRIAGE CONTROL. THIS STOPS A PAPER TAPE READER AT THE END OF EACH RECORD WHEN THE TAPE IS SUBSEQUENTLY READ BACK IN. (THE READER IS RESTARTED BY THE DC1 TRANSMITTED AT THE START OF EACH READ.)

C. TERMINAL TYPES 10, 11, 12, 15, AND 16 TRANSMIT ENQ AFTER EACH 80 CHARACTERS OF OUTPUT DATA. THE DRIVER (INTERRUPT HANDLER) THEN SUSPENDS UNTIL THE DEVICE RESPONDS WITH AN ACK, OR UNTIL 10 SECONDS HAVE ELAPSED. FOR TERMINAL TYPES 10, 11, AND 12, OUTPUT RESUMES AT THAT POINT; FOR TERMINAL TYPES 15 AND 16, ANOTHER ENQ IS SENT AND A NEW TIMEOUT IS STARTED.

D. TERMINAL TYPE 11 ALLOWS THE USE OF BLOCK MODE "ENTER" IN THE MPE COMMAND INTERPRETER AND SUBSYSTEMS. HOWEVER, THE USER MUST MANUALLY POSITION THE CURSOR IN THE FIRST COLUMN TO

BE TRANSMITTED, AND YOU MAY NOT LOG ON WITH "ENTER". PROGRAMS THAT MUST CONTROL BLOCK MODE "ENTER" OPERATIONS SHOULD USE TERMINAL TYPE 10.

E. TERMINAL TYPES 12 AND 15 DO NOT ALLOW THE USE OF THE PARITY CHECK FEATURE. THIS IS BECAUSE THESE TERMINAL TYPES TRANSMIT AND RECEIVE 8-BIT DATA.

F. THE REASON THAT CONTROL-Y IS ECHOED FOR BACK SPACE ON TERMINAL TYPE 4 IS THAT THIS TERMINAL INTERPRETS ONLY CONTROL-Y AS A "CURSOR LEFT" OPERATION.

G. IF A FORM FEED CHARACTER IS SENT TO A TERMINAL THAT DOES NOT RESPOND TO FORM FEED ("NO" IN THE TABLE) THE DRIVER WILL SEND A LINE FEED INSTEAD.

H. FOR TERMINALS THAT DO RESPOND TO FORM FEED, THE DRIVER INSERTS CHARACTER DELAYS AFTER THE FORM FEED TO ALLOW THE PAPER TO BE POSITIONED. THERE IS A DIFFERENT DELAY FOR EACH SPEED:

SPEED IN CPS	10	15	30	60	120	240
CHARACTER DELAYS	20	30	60	120	240	255

I. ALL CHARACTER DELAYS ARE ACCOMPLISHED BY TRANSMITTING AN ALL "ONES" CHARACTER (INCLUDING THE START BIT). THIS HAS THE EFFECT OF DELAYING ONE CHARACTER TIME, BUT THE TERMINAL DOES NOT "SEE" A CHARACTER BECAUSE THE STATE OF THE LINE NEVER GOES TO 0 (SPACE).

J. ONLY TERMINAL TYPES 4, 6, 9, 10, 12, 13 (1918 RPE), 15, AND 16 ARE SUPPORTED ON SERIES 30/33 SYSTEMS. BECAUSE THE ADCC CAN NOT TRANSMIT AN ALL "ONES" CHARACTER, NULLS ARE USED FOR DELAY. ALSO, YOU MAY ONLY LOG ON TO THE 30/33 AT NONE-0 OR EVEN PARITY.

K. YOU MAY LOG ON TO THE SERIES II/III WITH ANY PARITY. NONE-0 (BIT 8 FORCED TO 0) WILL ASSUME ODD PARITY, WHILE NONE-1 WILL ASSUME EVEN. YOU CAN PROGRAMMATICALLY CHANGE THE PARITY AT ANY TIME. RPE NORMALLY IGNORES PARITY ERRORS FROM TERMINALS, BUT YOU MAY PROGRAMMATICALLY ENABLE THIS CHECKING; THE FILE SYSTEM WILL RETURN A PARITY ERROR CODE TO YOUR PROGRAM IF SUCH OCCURS.

L. THE DRIVER SENDS NO PROTOCOL CHARACTERS WHATSOEVER FOR TERMINAL TYPE 10 (INCLUDING X-ON).

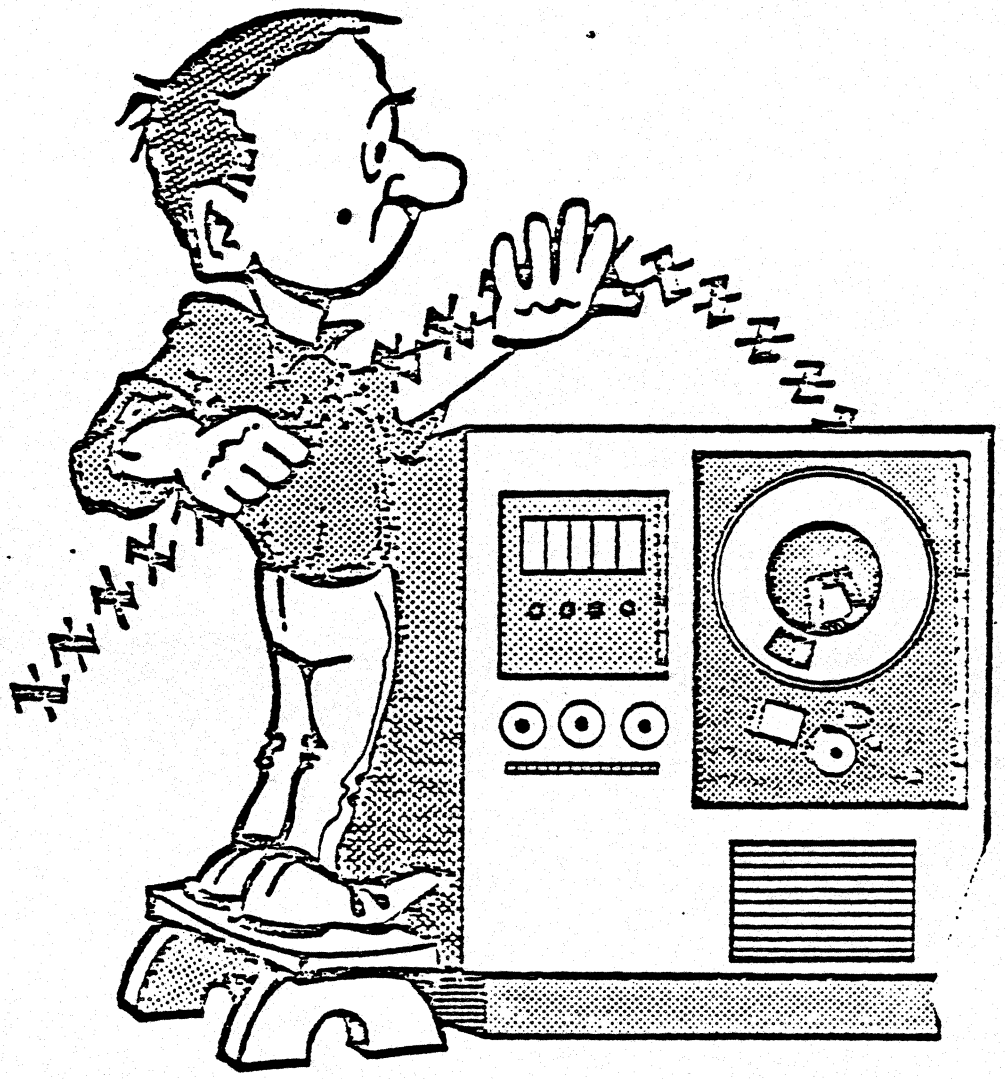
M. X-ON/X-OFF (DC1/DC3) FLOW CONTROL IS FULLY SUPPORTED ON REVISIONS OF RPE DATE-CODED 2020 OR LATER. TERMINALS MAY USE THIS TO STOP THE SYSTEM FROM TRANSMITTING CHARACTERS BY SENDING A DC3; ANY WRITE IN PROGRESS WILL BE STOPPED AND NO NEW WRITE WILL BE STARTED UNTIL THE TERMINAL SENDS A DC1.

N. TERMINAL TYPE 19 IS FOR THE 2631B REMOTE PRINTER ONLY! IT USES A SPECIAL PROTOCOL AT THE END OF EACH LINE; IF YOU LOG ON WITH THIS TERMINAL TYPE, YOUR TERMINAL WILL APPEAR "HUNG." MANY ;ABORTIO5 ON THE CONSOLE WILL TYPE YOUR DEVICE. THE SPECIAL PROTOCOL USED IS AS FOLLOWS: AT THE END OF EACH LINE, THE DRIVER SENDS <ESC> ? TO THE 2631. THIS IS A REQUEST FOR INTERFACE STATUS; THE 2631B RESPONDS WITH A ONE-CHARACTER STATUS STRING THAT TELLS THE DRIVER IF THERE IS PAPER IN THE PRINTER, WHETHER IT IS ON-LINE, AND WHETHER THE LAST LINE SENT HAD PARITY OR FRAMING ERRORS.

O. THE 2615B APPEARS TO WORK JUST FINE IN PLACE OF THE 2631B. HOWEVER, DO NOT LOG-ON WITH THE 2615 IF IT IS CONFIGURED AS A REMOTE SPOOLED PRINTER! THE SYSTEM WILL LOOP AND WILL BE HUNG.

P. YOU CAN CONFIGURE OTHER DEVICES FOR TERMINAL SPOOLING; HOWEVER, FORM FEED CAPABILITY SEEMS TO BE LOST. THE ONLY THING THAT WORKS FOR SURE IS A 2631B CONFIGURED AT TERMINAL TYPE 19, DEVICE TYPE 32 (PRINTER), DEVICE SUBTYPE 14 (LOCAL/HARDWIRE) OR 15 (MODEM), AND SPEED CONSISTENT WITH COMMUNICATIONS LINE REQUIREMENTS.

INITIAL AND SYSDUMP



INITIAL OUTER BLOCK

```
50050     $CONTROL SEGMENT=BOOTSTRAP
50052       DISCBOOT:
50054               LASTLOADMODE:=X;  ((FROM INFOTABLE--SEE BOOTSTRAP))
50056               LOADFROMTAPE  := FALSE;
50058       TAPELOAD:
50060               PUSH(DB,Z,Q,S);
50062               DBVALUE := TOS;
50064               DEL;
50066               ZVALUE := TOS;
50068               QVALUE := TOS;
50070               SVALUE := TOS;
50072               MAINSEG1;
50074               MAINSEG1B;
50076               MAINSEG2;
50078               MAINSEG3;
50080               MAINSEG4;
50082     END ((PROGRAM "INITIAL")) ;
```

HOW

INITIAL

BRINGS UP

THE

SYSTEM

MAINSEG 1

*LOADS BY MICROCODE

*SETS UP TABLES, MEMORY, AND DISC FOR INITIAL TO RUN
SPEED SENSE - AUTOMATIC ON SERIES III
HIT CARRIAGE RETURN ON SERIES 33

*IF DISC:
- ASKS WHICH OPTION (WARM/COOL)

*IF TAPE:
- ASKS WHICH OPTION (REL/COLD/UPD)
- IF UPDATE, READS DISC COLD LOAD INFO TABLE FOR ADDRESS OF CTABØ, APPROPRIATE RECORD OF OLD CONFDATA, USES THIS FOR I/O CONFIGURATIONS,
- IF COLD/RELOAD, USES CONFIGURATIONS FROM TAPE.
- IF SERIAL DISC, ASKS OPERATOR TO NON-UNIT ZERO AND PUT SYSDISC BACK ON UNIT Ø.

*IF NOT WARMSTART, DOES I/O CONFIGURATION CHANGES

*IF NOT RELOAD, USES DISC COLD LOAD INFO TABLE TO FIND MPE TABLES (DIRECTORY, VIRTUAL MEMORY, ETC.)
- CHECK COLD LOAD ID FROM COLD LOAD INFO TABLE AGAINST THE VOLUME TABLE. IF BAD, "VOLUME TABLE DESTROYED, MUST RELOAD."
- CHECK RELOAD FLAG FROM COLD LOAD INFO TABLE, IF ON, "PREVIOUS RELOAD ABORTED MUST RELOAD."

*IF RELOAD:
- BUILDS NEW DISC COLD LOAD INFO TABLE AND FREE SPACE TABLE.
- LOADS FROM TAPE: VOLUME TABLE, DIRECTORY, VIRTUAL MEMORY, RIN TABLE.

MAINSEG 1 (CONT'D.)

- IF OPTION NULL, ONLY BUILDS MANAGER.SYS, PUB
- IF OPTION ACCOUNTS, DELETES FILE ENTRIES.

*IF TAPE
- SETS TAPE LOAD FLAG

*UPDATES COLD LOAD INFO TABLE AND WRITES TO DISC
*CHECKS VOLUME TABLE AGAINST ALL MOUNTED VOLUMES
- IF VOLUME MISSING, "MOUNT CORRECT VOLUME OR RELOAD"

*HANDLES DISC VOLUME CHANGES
*CHECKS ALL SYSTEM DOMAIN VOLUMES FOR VALID LABELS.
*SCANS DEFECTIVE TRACKS TABLE(S) FOR SUSPECT TRACKS
*AFTER DISC CHANGES, NON-SYSTEM DOMAIN DISCS ARE ADDED TO THE VOLUME TABLE.
VM CHANGES - BUT LDEV 1 CAN ONLY BE CHANGED ON RELOAD.
*IF RELOAD, CHANGES, DIRECTORY, RIN TABLE SIZES
*BUILD DISC BOOTSTRAP PROGRAM AND PUT ON DISC
*MOVE INITIAL AND ITS STACK TO HIGH CORE.

MAINSEG1B *INITIALIZES SYSGLOB, DRT, DST, CST, CSTX, DIRECTORY DSEG, DIRECTORY SPACE DSEG, RIN TABLE.

*PURGES FILES ON NEWLY REASSIGNED TRACKS.
*IF RECOVER LOST DISC SPACE, RECREATES DISC FREE SPACE TABLES BY READING EACH FILE LABEL AND REMOVING FREE SPACE FOR EACH EXTENT
*CLOSES FILES/MOUNTS THAT WERE LEFT OPEN.
*IF RELOAD, COPIES DIRECTORY DIRECTLY FROM TAPE (INCLUDING FILE ENTRIES) ZEROES OUT UNUSED PORTIONS OF DIRECTORY.
*IF TAPE, READS SYSTEM FILES

MAINSEG2

* UPDATE COLD LOAD ID IN SYSTEM PROGRAM FILES IF NOT LOAD FROM TAPE -ELSE-
* CREATE LOADMAP FILE.
* LOAD ALL SYSTEM DOMAIN USER FILES. - IF RELOAD
* INITIALIZE
  - TBUFS
  - PCB
  - ICS
  - CS DRIVER TABLE
  - IOQ
  - DISC REQ TABLE
  - ILT
  - DIT
  - DLT
  - SYSTEM BUFFERS
  - SWAP TABLE
  - CSTBLK.
  - SPECIAL REQ TAB.
  - MSG HARBOR TAB

    PRI MSG TABLE

    MEASINFO TABLE
  - VDSMTAB

    ARSBM TABLE

    ARLD TABLE
  - LPDT
  - TRL
  - JOB TABLES
  - SIR TABLE

    MONITOR BUFFER

    END OF BANK Ø

    DEPENDENT MPE

MAINSEG 3

*If load from tape then write loadmap to Disc.

*Load SL Segments.

*Load Segment 1 (ININ)

- Requires special attention by sysdump.
- Outer block modified since microcode must be able to map to appropriate procedure

Build ICS.

*Creates all System I/O Processes.

*Load Drivers and Interrup Handlers.

Global Note

**If switch Register.(8:8)< > Restart Device DRT then enter HELP at beginning of MAINSEG 1B, 2, 3, 4 and assemble (HALT) at end of MAINSEG 4.

MAINSEG4

* Table Setup Segment
  - CS Data Segment
  - Disc Free Space Table
  - Reply Information Table
  - UCOP Request Queue
  - P-P Communications Table
  - Job Process Cross Reference Table
  - System JIT
  - System JDT
* If Warmstart Recover
  - JMAT
    * Look for Jobs that specified restart
  - Input Device Directory (IDD)
    * Info on Input Spool Files
  - Output Device Directory (ODD)
    * Info on Output Spool Files
  - Else
    * Initialize JMAT, IDD, ODD
* Initialize FMAVTABLE
* Recover/Initialize welcome MSG
* Initialize CI logon DST.
  Tape label table
* Write LDT, to Disc

MAINSEG4 (CONT'D.)

* INITIALIZE ASSOCIATION TABLE
  MVTAB, PVTAB, BREAKPOINT TAB
* IF LOGGING INITIALIZE LOG BUFFERS AND PROCESS
* CREATE MEMLGSTOP
  - PV REC
  - UCOP
  - PFAIL
  - DEVREC
  - LOADER
  - PROGEN
* WRITE LOADMAP TO DISC
  - DIRSPACE
  - SEGMENT TABLE
* SET UP AVAILABLE REGION LISTS
* ASSEMBLE (DISP);

PROGENITOR

* After ~~HESM(,2)~~ from INITIAL then
  - Initiate Console & Sysdisc
  - Request Date & Time
  - Initialize System Clock
* Awake Following Processes
  - Logging
  - MemlogP
  - PVPROC
  - UCOP
  - Spooling
* Send Welcom Message to Console
* Wait
  - Set Junkwait in PCB #1 (PROGEN)
  - Get TPRI (150)
  - Wait for A^C
* Output to console queued if read pending,
  i.e., an "=" prompt unanswered

*** MPE IS NOW UP ***

WHEN YOU SAY :SYSDUMP...

* OP CAPABILITY REQUIRED OF THE USER.
* COMMAND INTERPRETER ISSUES :FILE EQUATIONS FOR DUMP AND LIST DEVICES.
* COMMAND INTERPRETER RUNS SYSDUMP AS A PROGRAM
* SYSDUMP CONTAINS 6 SEGMENTS:
    0 MPECHECK
    1 INIALIZE
    2 SYSTEMCH
    3 IOCHANG
    4 DUMPTAPE
    5 SYSDUMP.
* SYSDUMP WAS RESEGMENTED FOR THE A - MIT.
* ENTRY POINT DEFAULTS IS USED BY THE LAB FOR THE MIT BUILD.

SYSDUMP OUTER BLOCK

```
18022     $CONTROL SEGMENT=SYSDUMP
18024              DEFAULT := FALSE;  (( SYSDUMP ENTRY POINT ))
18026     DEFAULTS: PUSH(STATUS);  (( PDEFAULT ENTRY POINT ))
18028              TOS.(2:1) := 0;  ((DISABLE TRAPS))
18030              SET(STATUS);
18032              INITIALIZATION;
18034              IF YESANSWER(2) THEN
18036               BEGIN      (( TRUE IF CHANGES REQUESTED ))
18038                INITIALIZE'CH;
18040                DO
18042                 BEGIN
18044                  TCLASS := 0;   (( NO ENTRIES IN TEMPCLASS ))
18046                  TCLASS(1) := 4;((TEMPCLASS LENGTH IN BYTES))
18048                 WHILE YESANSWER(3) DO IO'CONFIG'CH;
18050                 END
18052                UNTIL CHECKDEV;
18054                IF YESANSWER(24) THEN SYSTEM'TABLE'CH;
18056                IF YESANSWER(61) THEN MISC'CONFIG'CH;
18058                IF YESANSWER(75) THEN LOGGING'CH;
18060                IF YESANSWER(35) THEN DISK'ALLOC'CH;
18062                IF YESANSWER(38) THEN SCHEDULING'CH;
18064                IF YESANSWER(39) THEN SEG'LIMIT'CH;
18066                IF YESANSWER(57) THEN SYSTEM'PROG'CH;
18068                IF YESANSWER(48) THEN SYSTEM'SL'CH;
18070                IF DEFAULT THEN BUILD'MPECHECK;
18072               END;
18074              IF GETDUMPDATE THEN
18076               BEGIN      (( TRUE IF DUMP DATE SUPPLIED ))
18078                GET'FILE'SUBSET;
18080                IF YESANSWER(82) THEN ((LIST FILES DUMPED?))
18082                 LISTFILES := TRUE;
18084               END;
18086              DUMPTAPE(LISTFILES);
18088              LIST'SYS'FILES;
18090             END.
```

SEGMENT Ø - MPECHECK

IF DEFAULTS BUILDS THE MPE CHECK FILE

SEGMENT 1 - INIALIZE

CHECK FOR OP CAPABILITY
INITIALIZE DL FOR EXPANDABLE TABLES

CS TABLE	NEW VTAB
DVR TABLE	OLD VTAB
LPDT	RIN
LDT	CTAB
DCT	CTABØ
LDTX	

SEGMENT 2 - SYSTEMCH

SYSTEM TABLE CHANGES
MISC CONFIGURATION CHANGES
DISC ALLOCATION CHANGES
SCHEDULING CHANGES
(NOTHING UNDER THIS SECTION IN C-MIT BUT STILL
EXISTS FOR JOB STREAM COMPATABILITY)
SEGMENT LIMIT CHANGES
SYSTEM PROGRAM CHANGES
SL CHANGES

SEGMENT 3 - IOCHANG

LIST I/O CONFIGURATION
LIST CS DEVICES
LIST CLASSES
I/O CONFIGURATION CHANGES
CLASS CHANGES
CHECK FOR VALID I/O CONFIGURATION

SEGMENT 4 - DUMPTAPE

SET UP INITIAL PROGRAM
GENERATE I/O PROGRAM FOR LOAD/ENABLE MICRO-CODE
DUMP ICS & LOW CORE FOR INITIAL
DUMP TABLES
DUMP ININ & CST
DUMP INITIAL'S DB AREA
RIN TABLE
DIRECTORY
SYSTEM SL
SYSTEM PROGRAMS
PASS DUMP FILE SUBSETS TO RESTORE TO DUMP USER FILES

SEGMENT 5 - SYSDUMP

UTILITY PROCEDURES
MESSAGE
DUMPDATE
DUMP FILE SUBSETS
LIST FILES DUMPED/NOT DUMPED