OPERATING SYSTEM INTERNALS

VAX/VMS OPERATING SYSTEM INTERNALS COURSE DESCRIPTION

This intensive lecture course provides an understanding of the various components, data structures, and operations of the VAX/VMS operating system. It also facilitates the subsequent examination and analysis of system source listings.

VAX/VMS OPERATING SYSTEM INTERNALS COURSE OBJECTIVES

Students will learn to:

* Analyze the contents, use, and relationship of selected components and data structures of the VAX/VMS operating system.
* Write Kernal mode code.
* Read Operating system code.
* Analyze crash dumps, using the System Dump Analyzer.
* Analyze the operations including:
  - Scheduling
  - Paging
  - Swapping
  - Process creation and deletion
  - System initialization
  - Image activation
  - Synchrinization

VAX/VMS OPERATING SYSTEM INTERNALS COURSE PREREQUISITES

Knowledge of the material contained in the following courses:

* **VAX/VMS UTILITIES AND COMMANDS**

* **ASSEMBLY LANGUAGE LANGUAGE PROGRAMMING IN VAX-11 MACRO**

* **PROGRAMMING VMS IN VAX-11 language**
  (Now called "Utilizing VMS features from VAX-11 language")

* (Optional) VAX/VMS System Management

1) Read 1ST 12 chapters SYS. Services REF. Manual
2) OSI Handout / IDSM references
3) Do LAB PROBS
4) Read ENTIRE IDSM
5) Do it AGAIN FOR V4.X

VAX/VMS OPERATING SYSTEM INTERNALS COURSE OUTLINE

MONDAY:

THE PROCESS

SYSTEM COMPONENTS

TUESDAY:

SYSTEM MECHANISMS

DEBUGGING

WEDNESDAY:

SCHEDULING

PAGING

IMAGE ACTIVATION

THURSDAY:

SWAPPING

PROCESS CREATION/DELETION

FRIDAY:

SYSTEM INITIALIZATION

TABLE OF CONTENTS

THE PROCESS

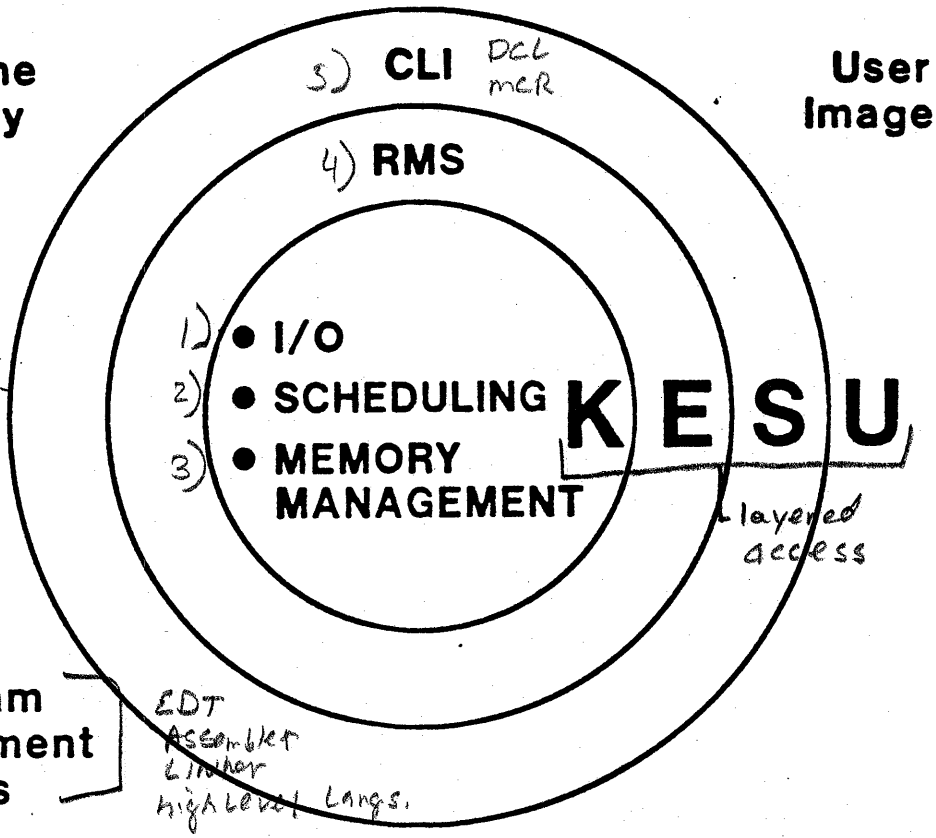
OBJECTIVES

* learn how various data structures are used by/for the process
* techniques used in modifying certain areas of process context (programming, commands, etc.)

READINGS

* IDSM Chapter 1 1-4 to 1-18
  IDSM Appendices A - E

ACCESS MODES AND COMPONENTS



Runtime Library

sys. services
high overhead

User Images

Call DCL
RMS
Sys. Serv.
(kernal)

Program Development Tools

EDT
Assembler
Linker
high level Langs.

1) $QIO
$assign
$cancel
I/O drivers (routines)
I/O data base

4) $open
$close
$GET
$PUT
→ $QIO

2) $CREPRC
$ SETIMR
$ Wake
Swapper (process)
Scheduler (routine)

3) $AQJWSL
$CRETVA
$CREPSC
Swapper
Pager (routine)

5) Commands
$show system
↓
$GETJPI

LOCATION OF CODE AND DATA

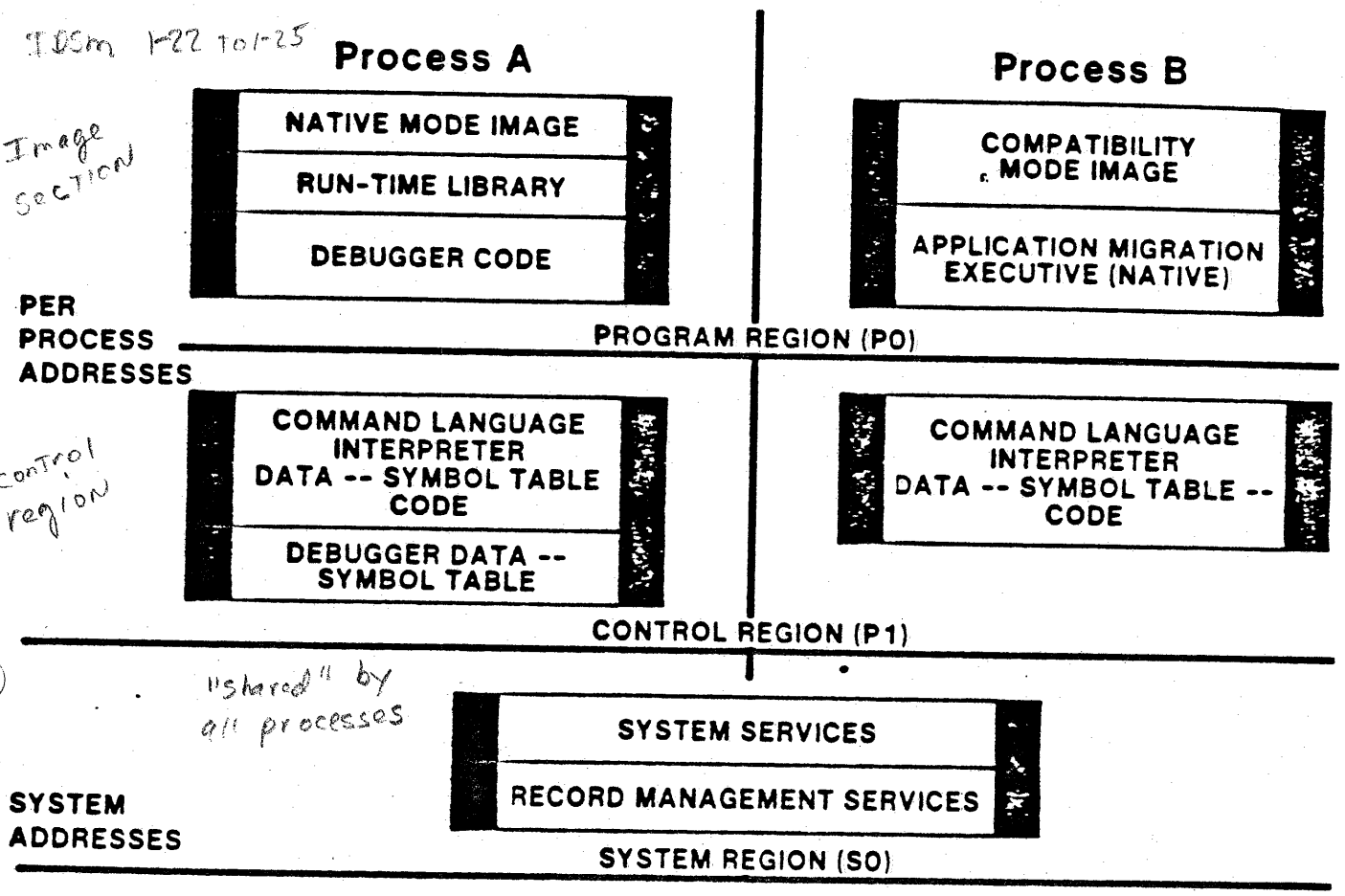
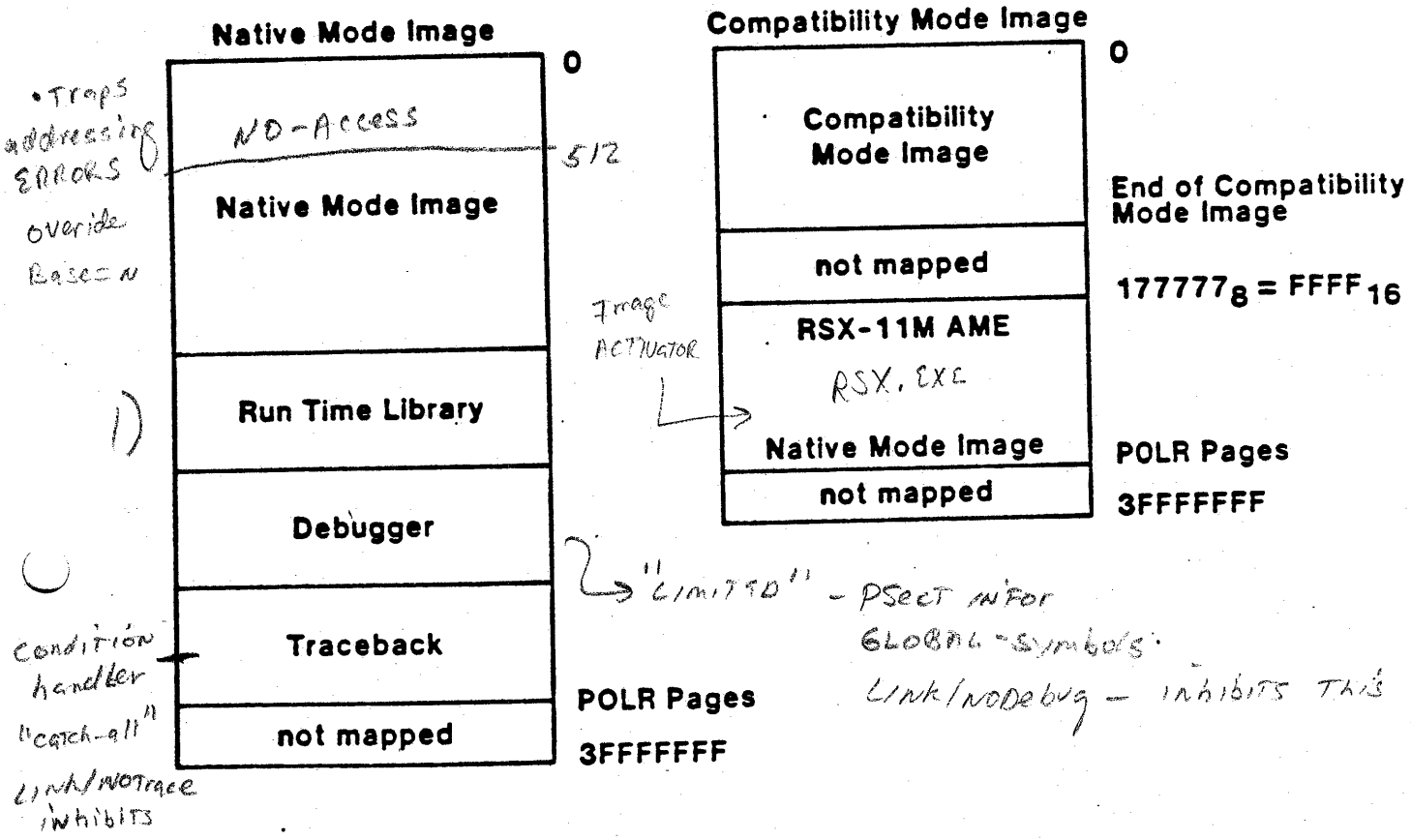


Image Header → native or compatibility

Linker Ref. manual chapter 5

PØ VIRTUAL ADDRESS SPACE

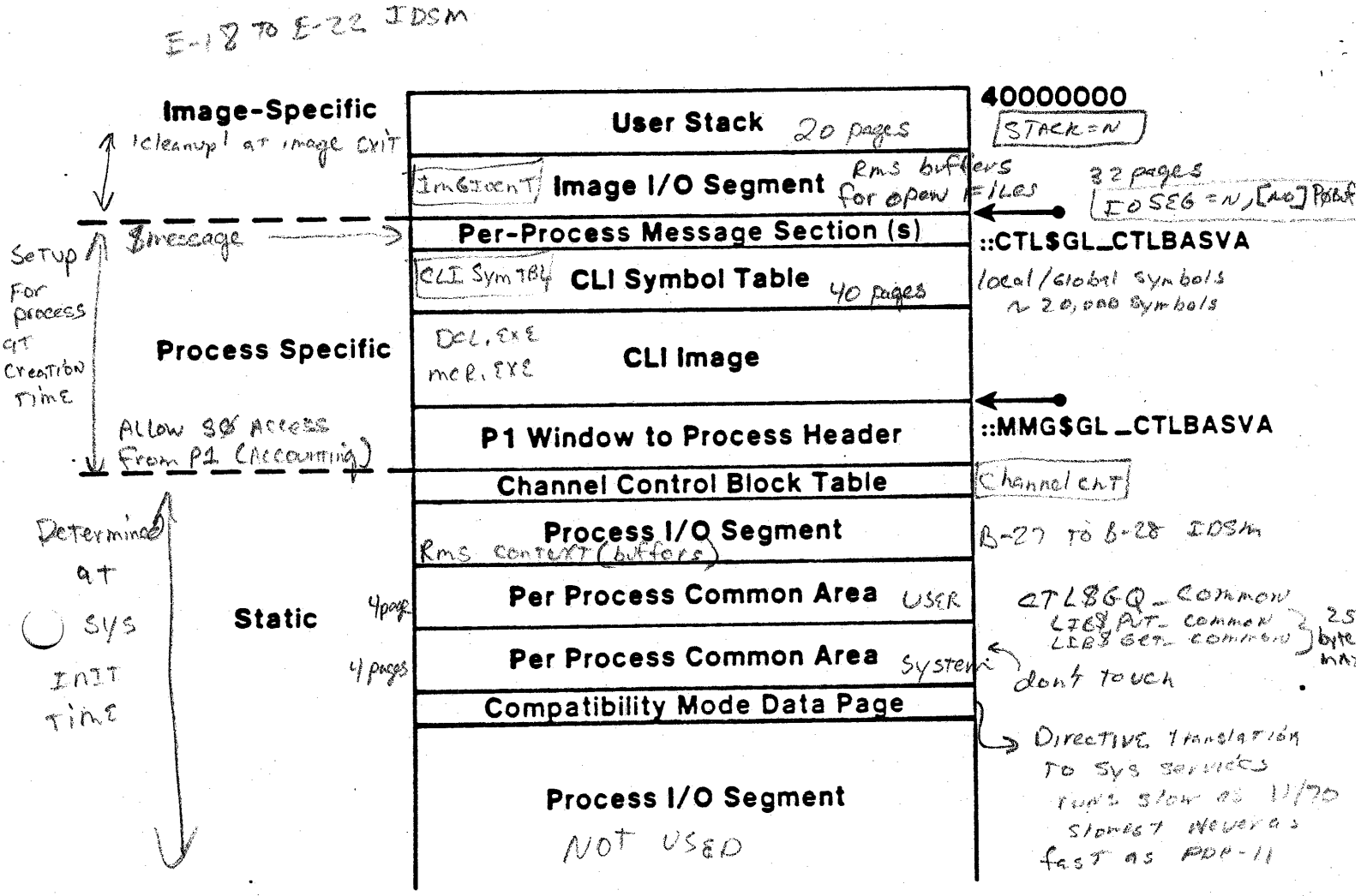


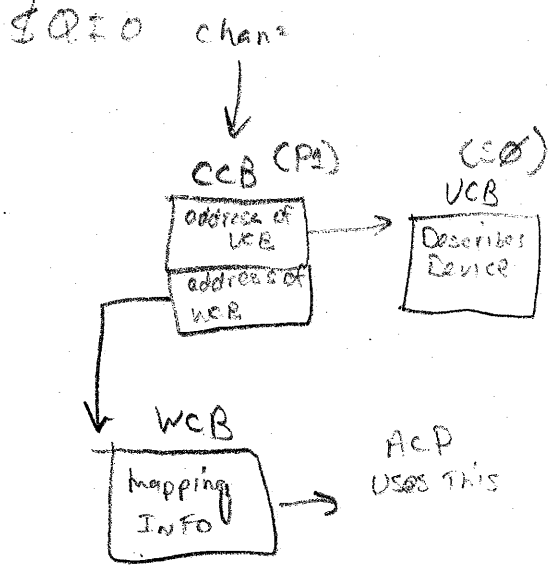
→ "LIMITED" - PSECT NAMES FOR GLOBAL SYMBOLS
LINK/NODEBUG — INHIBITS THIS

1) VMS RTL
LIBRSHR
any other shr. images

LINK/NOSYSSHR — if run out of virtual pages
↓
STARLET.OLB
/NOSYSLIB — if no VMS routines but your own

P1 VIRTUAL ADDRESS SPACE



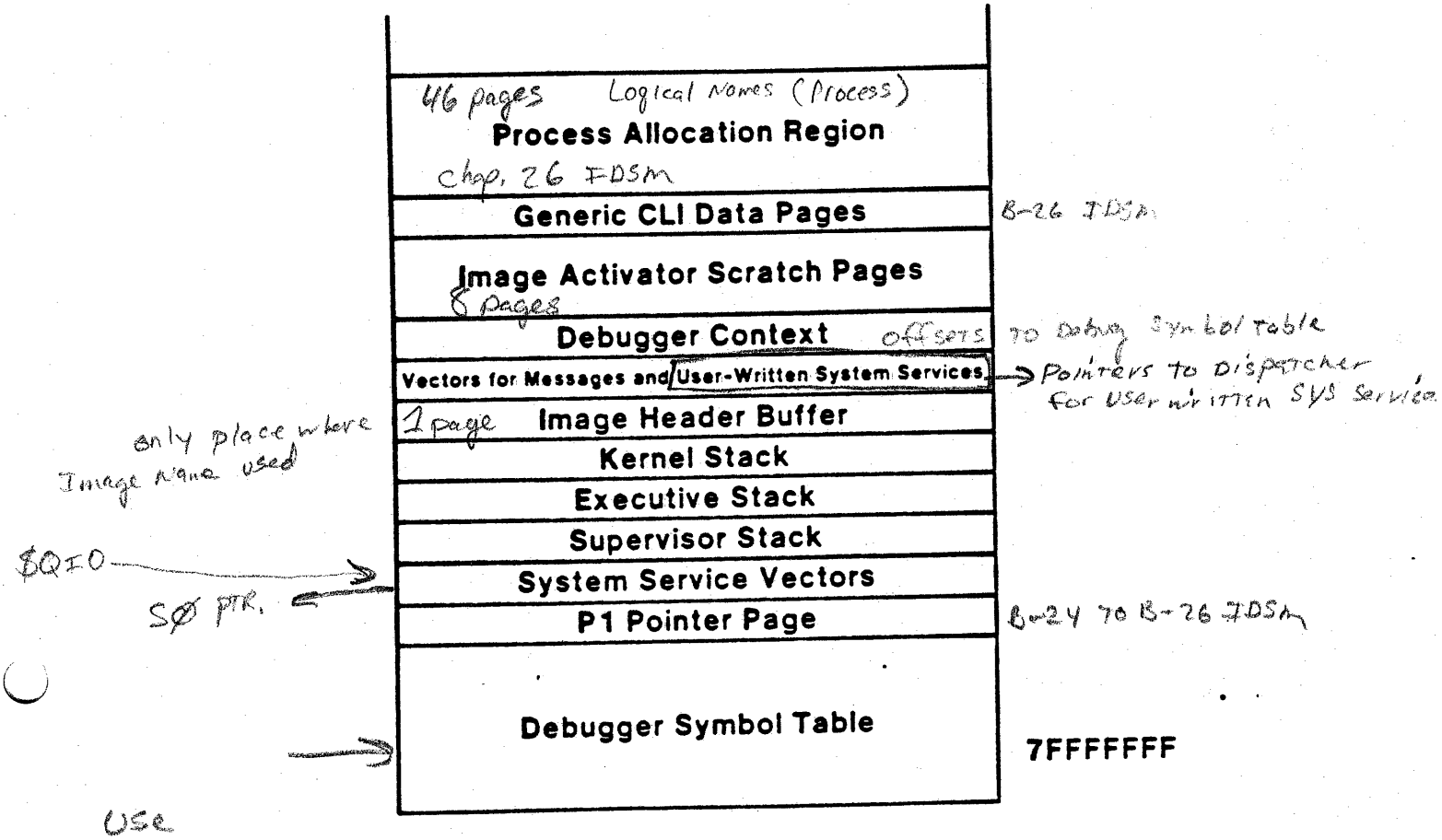


```
        .TITLE  COMMON1         Demonstrate P1 common area use

        .PSECT  DATA,NOEXE

MESSAGE:
        .ASCID  /P1 common strings/     ;Common area string
CODE:   .LONG   0                       ;Storage for errors

        .PSECT  CODE,NOWRT,EXE,SHR

        .ENTRY  COMMON1,        ^M<>    ;Null entry mask

        PUSHAL  MESSAGE                 ;Push argument on stack
        CALLS   #1,G^LIB$PUT_COMMON     ;Write string to P1 common
        BLBC    R0,10$                  ;Check status
        RET

10$:    MOVL    R0,CODE                 ;Copy error code
        PUSHL   CODE                    ;Put error code on stack
        CALLS   #1,G^LIB$STOP           ;Bomb with messages

        .END    COMMON1
```

```
        .TITLE  COMMON2         Program to read P1 common area

        .PSECT  DATA,NOEXE
MESSAGE_DESC:
        .BLKW   1                       ;Length fills at run time
        .BYTE   14,2                    ;Type=text  Class=dynamic
        .ADDRESS        MESSAGE_BUF
MESSAGE_BUF:
        .BLKB   252                     ;Buffer for P1 string
CODE:   .LONG   0                       ;Storage for error messages

        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  COMMON2,        ^M<>    ;Null entry mask

        PUSHAQ  MESSAGE_DESC            ;Push descriptor for string
        CALLS   #1,G^LIB$GET_COMMON     ;Get P1 string and length
        BLBC    R0,10$                  ;Check status

        PUSHAQ  MESSAGE_DESC            ;Push descriptor for return
        CALLS   #1,G^LIB$PUT_OUTPUT     ;Display P1 common string
        BLBC    R0,10$                  ;Check status
        RET                             ;Finished

10$:    MOVL    R0,CODE                 ;Storage for error message
        PUSHL   CODE                    ;Push error value
        CALLS   #1,G^LIB$STOP           ;Bomb with error messages

        .END    COMMON2
```

```
$RUN COMMON1
$RUN COMMON2

P1 common string
$
```

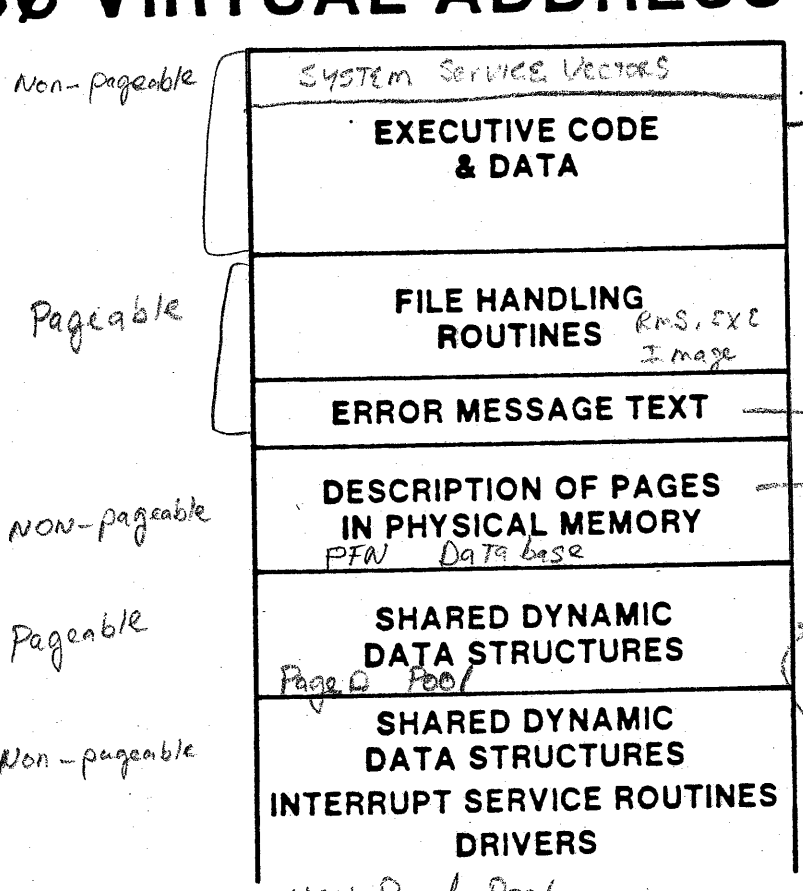
P1 VIRTUAL ADDRESS SPACE



Use
compile /Debug
Link / Debug
for "Full"
Capability

1-23 TO 1-24 IOSm

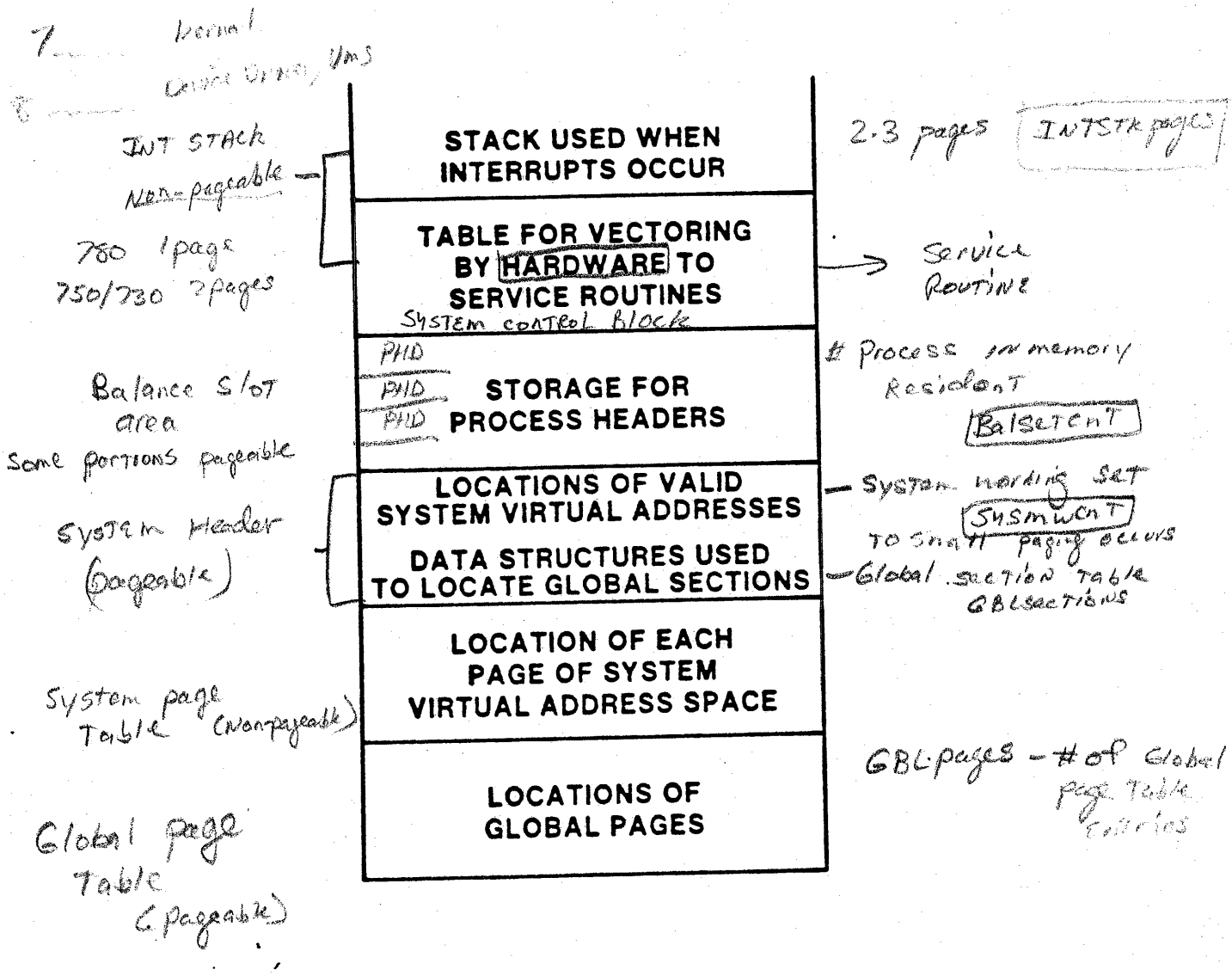
SØ VIRTUAL ADDRESS SPACE



SYSGEN Parms

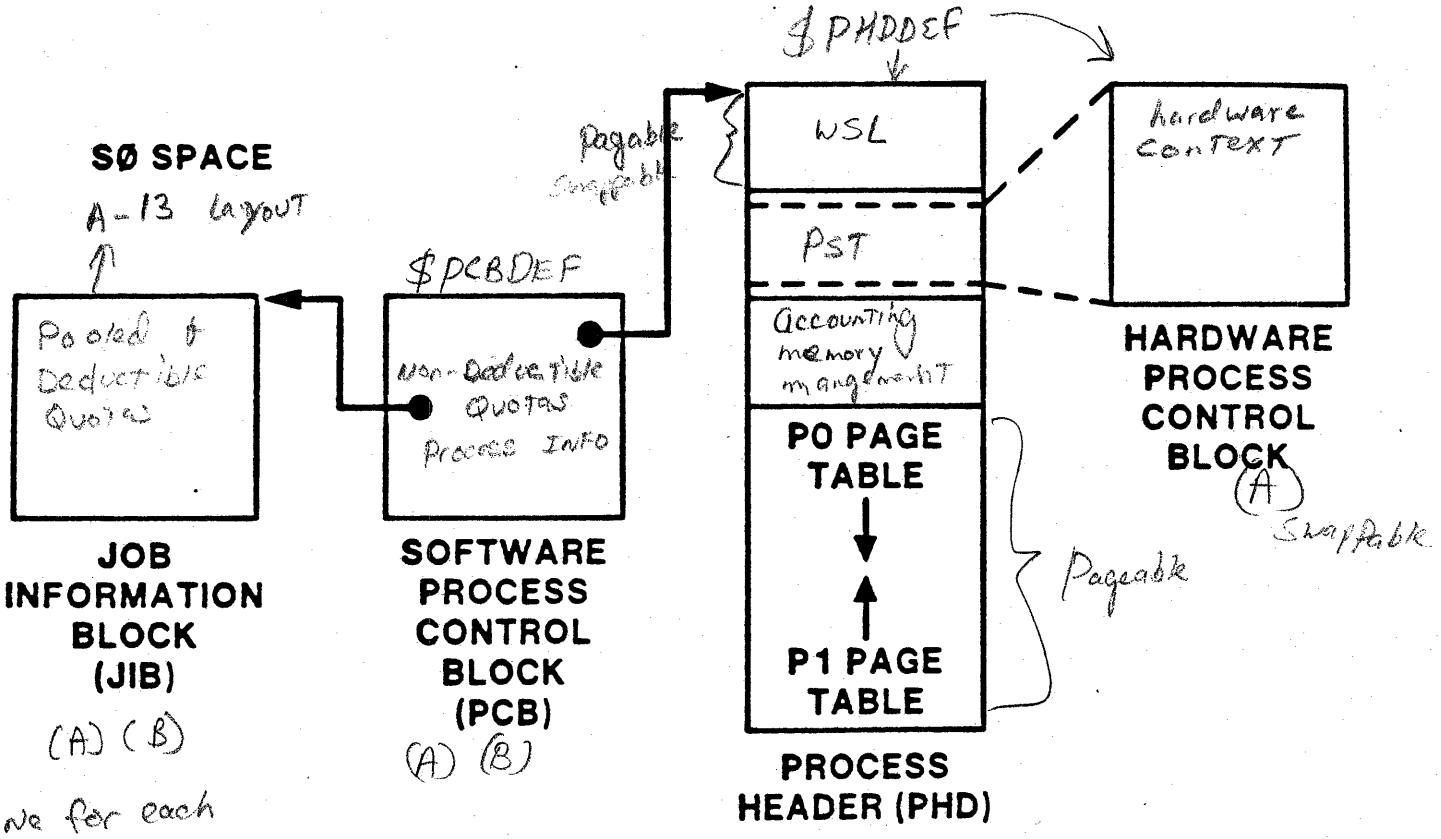
Paged Pool — PagEDYN

NONPaged Pool — NPagEDYN


STACK USED WHEN INTERRUPTS OCCUR
TABLE FOR VECTORING BY HARDWARE TO SERVICE ROUTINES
STORAGE FOR PROCESS HEADERS
LOCATIONS OF VALID SYSTEM VIRTUAL ADDRESSES
DATA STRUCTURES USED TO LOCATE GLOBAL SECTIONS
LOCATION OF EACH PAGE OF SYSTEM VIRTUAL ADDRESS SPACE
LOCATIONS OF GLOBAL PAGES
INT STACK
Non-pageable
780 1 page
750/730 2 pages
System control block
PHD
PHD
PHD
Balance Slot area
Some portions pageable
System Header
(pageable)
System page Table (nonpageable)
Global page Table
(pageable)
2-3 pages
INTSTK pages
Service Routine
Process in memory Resident
BalSetCnt
System working set
SysmwCnt
To small paging occurs
Global section Table
GBLsections
GBL pages - # of Global page Table Entries

DATA STRUCTURES SUMMARY OVERVIEW

1-3 IDSM

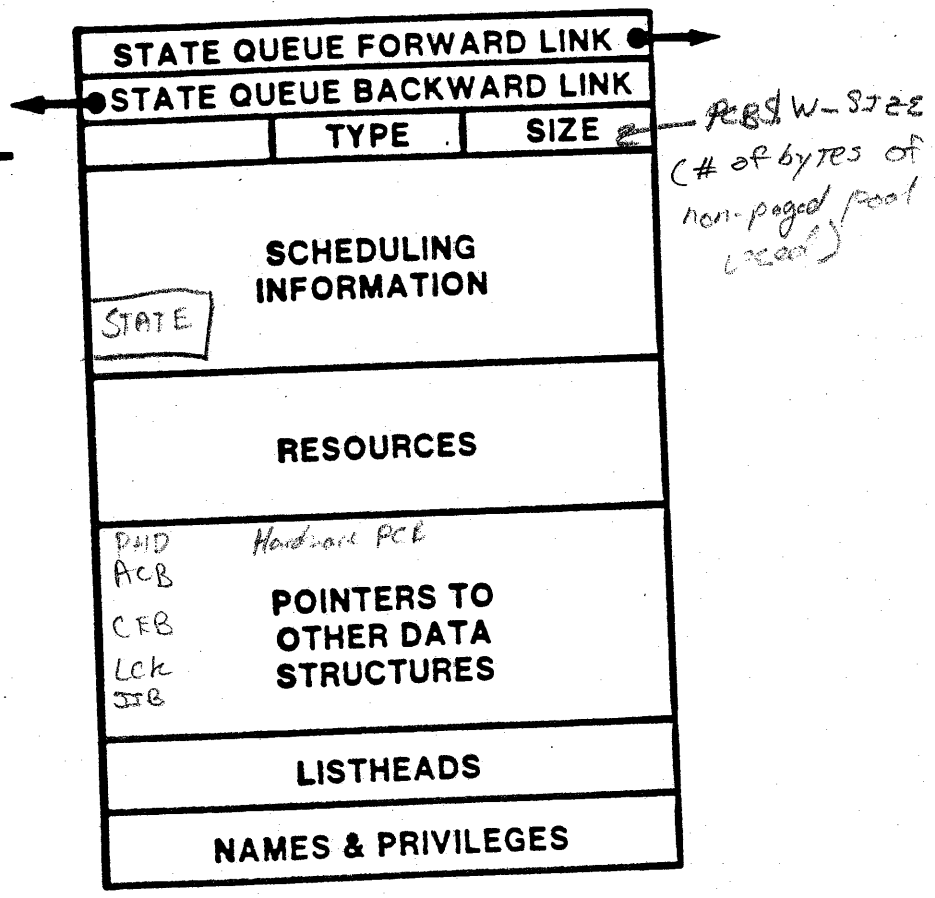


(A) Non-Paged

(B) Non-Swapped

D-11 TO D-12 IOSM

SOFTWARE PROCESS CONTROL BLOCK (PCB)



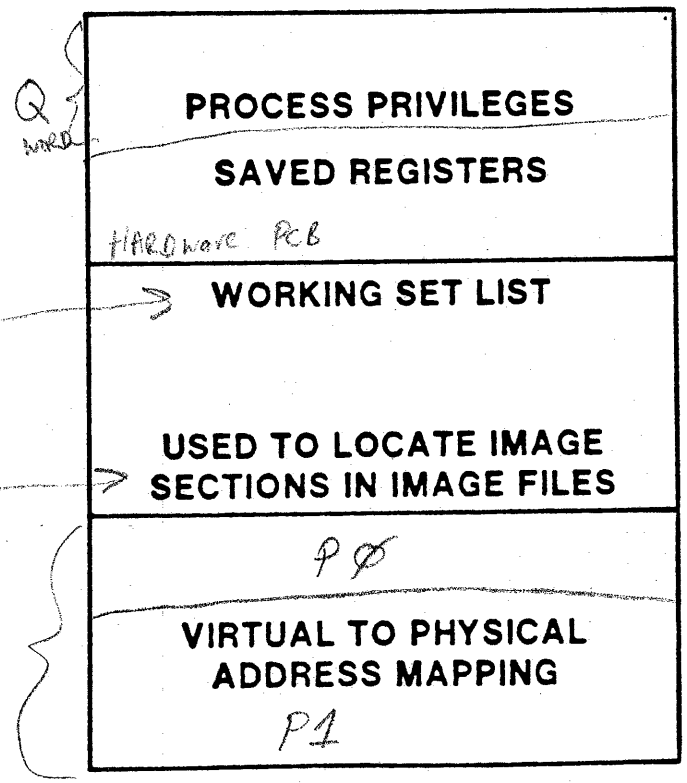
PCB$ A-18

PCB$K_LENGTH — actual size in bytes of data structure
JIB$

D-13 TO D-15 ZOSM

PROCESS HEADER (PHD)

A-23 Ref



process Section table

Process page Tables

Virtual Page CNT

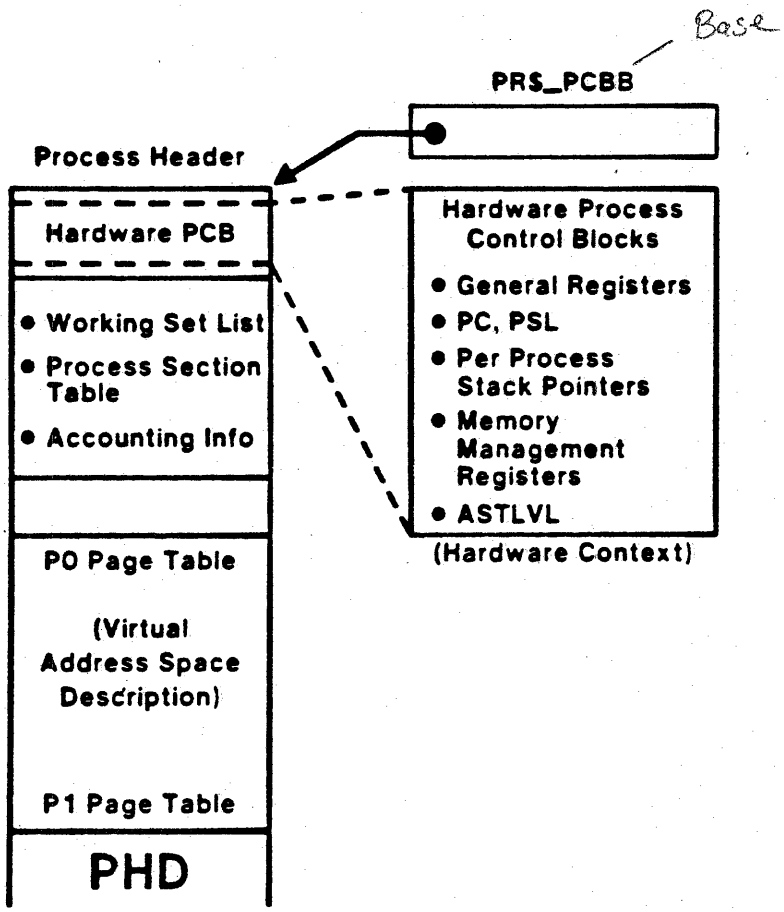
128 pages mapped per PT page

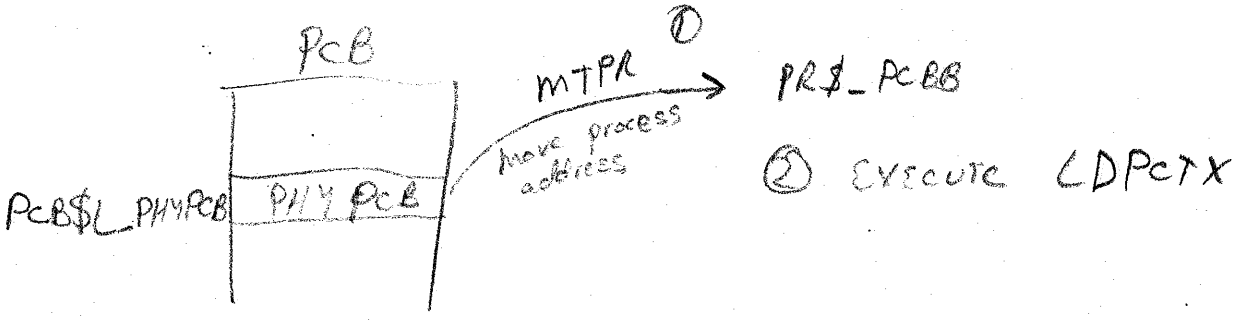
SPTES = $\frac{\text{VirtualPageCnt}}{128}$ * 4 bytes

1/2 page to map 8192 virt pages

Total COST = BalSetCnt * SPTES

RELATIONSHIP OF HARDWARE PCB AND PHD



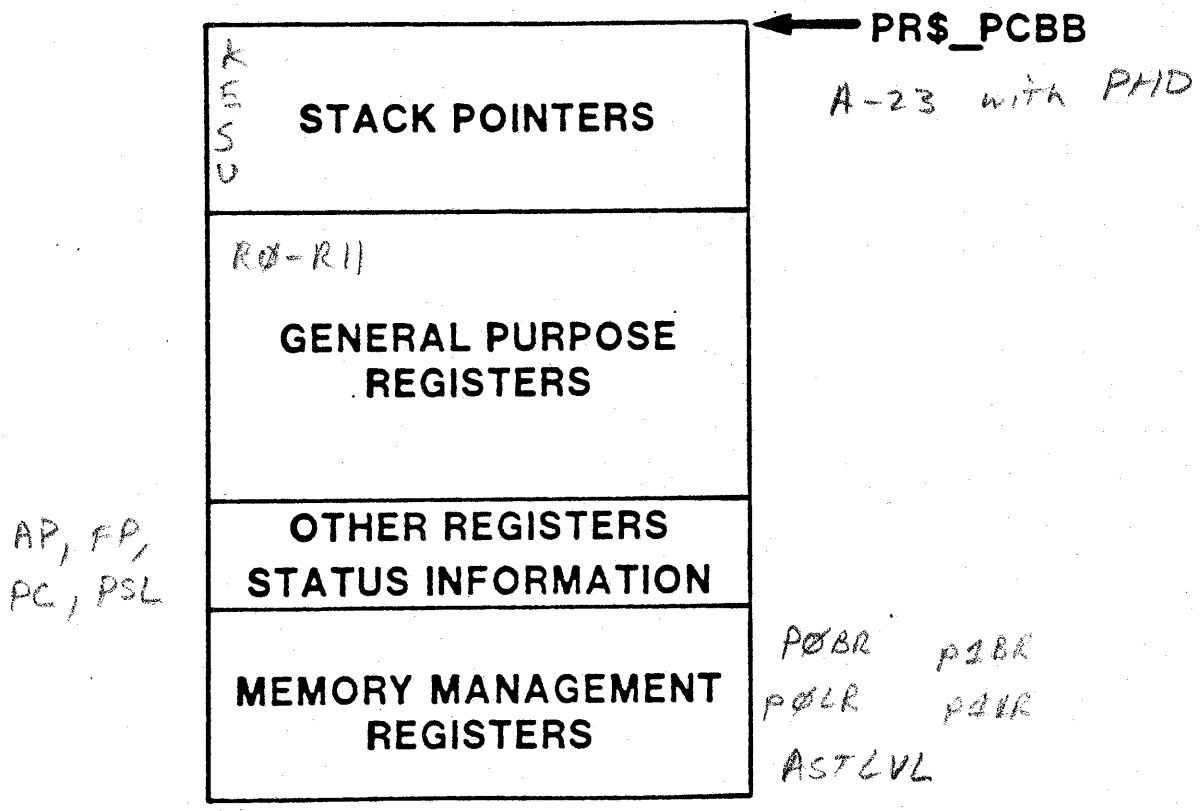


To go current process MTPR, LDPCTX

Remove current process MTPR, SVPCTX

D-13 TO D-15 IOSM

HARDWARE PROCESS CONTROL BLOCK



SYSTEM COMPONENTS

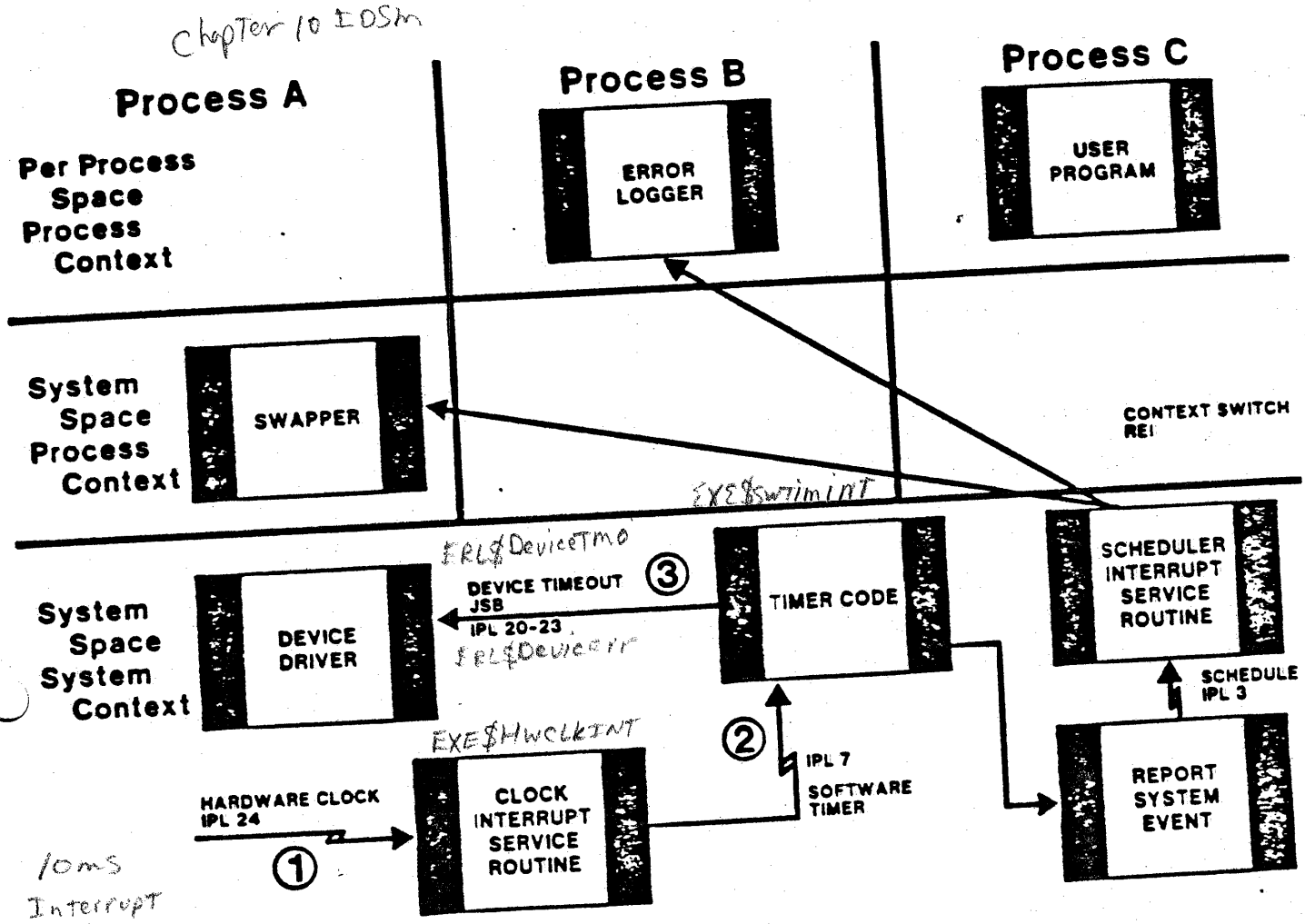
OBJECTIVES

* for each selected component, describe:
  - o primary function
  - o implementation
  - o access mode
  - o which address space it resides in
* methods of communication with each component

READINGS

* IDSM Chapter 1,16,20,27

HARDWARE CLOCK INTERRUPT



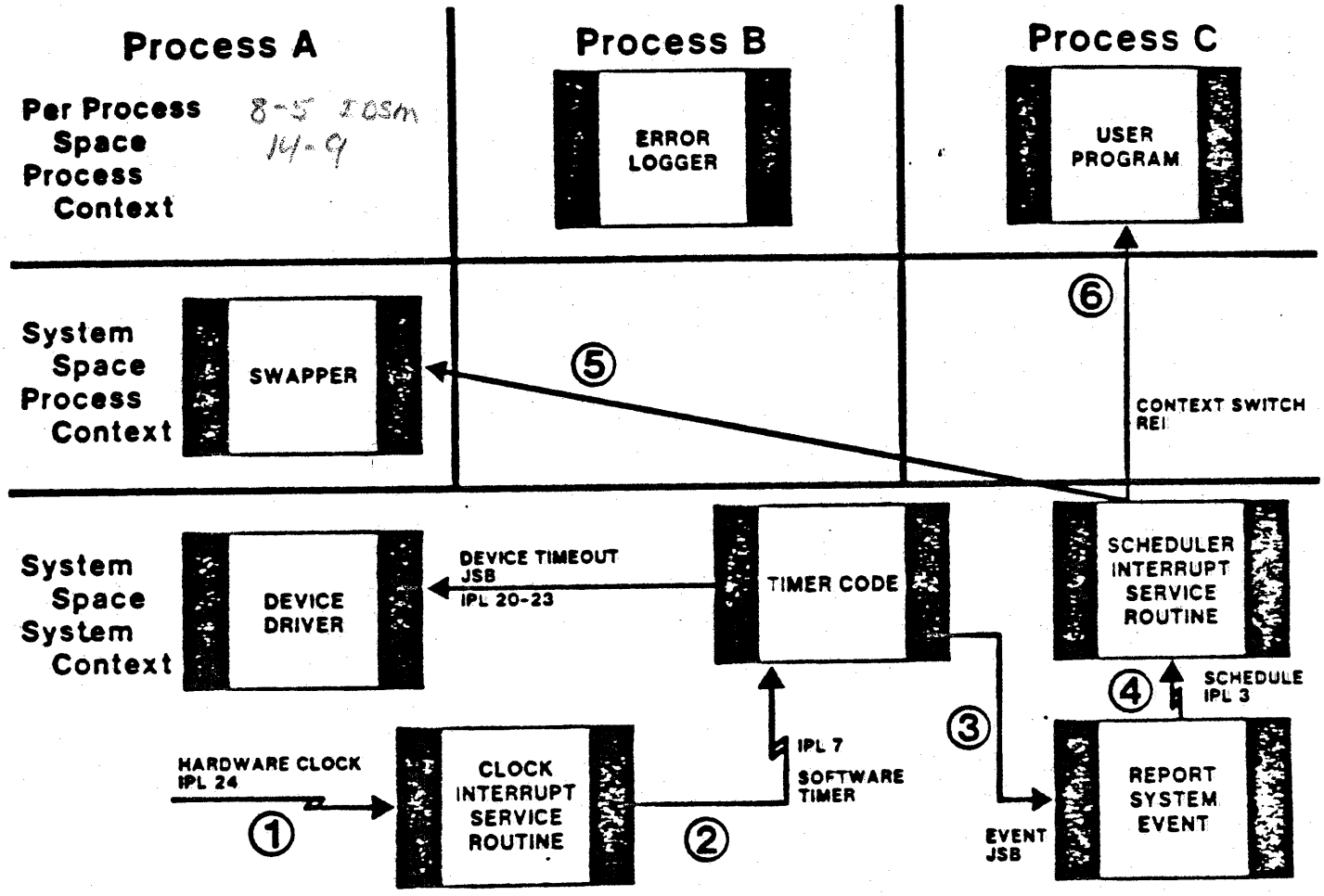
① - update System Time (+10ms)
EXE$GQ_SYSTIME
- update Quantum for current process (DECR by 10ms)

Quantum → Time slice period (# of 10ms increments)

↓
Check to see if value = 0 → yes → IPL = 2
- Check Timer Queue to see if any Events are DUE → -OR- → yes → IPL = 2

③ $SetImr
$Wake
$SchDwk
Builds TQE
- Repeating System Subroutine

SOFTWARE TIMER INTERRUPT



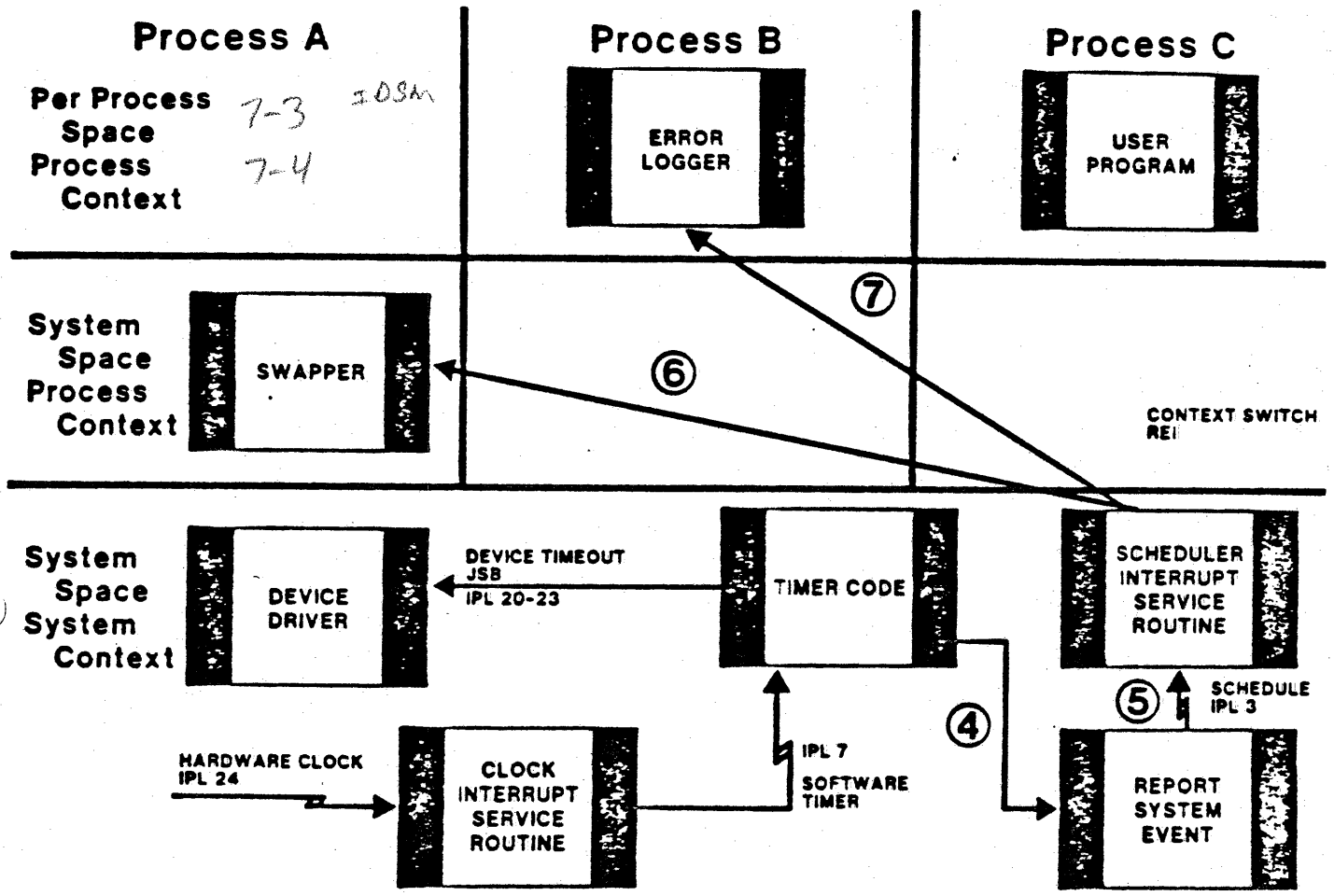
(3) Quantum END - System Event

R.S.E - Report System Event

SCH$QEND → Real time process
- Reset Quantum PHD$M_Quant
- no ws adjustment
- no priority adjustment
- clear InQuan BIT PCB$V_INQuan in PCB$L_STS

- computable = continues!

ERROR LOGGER



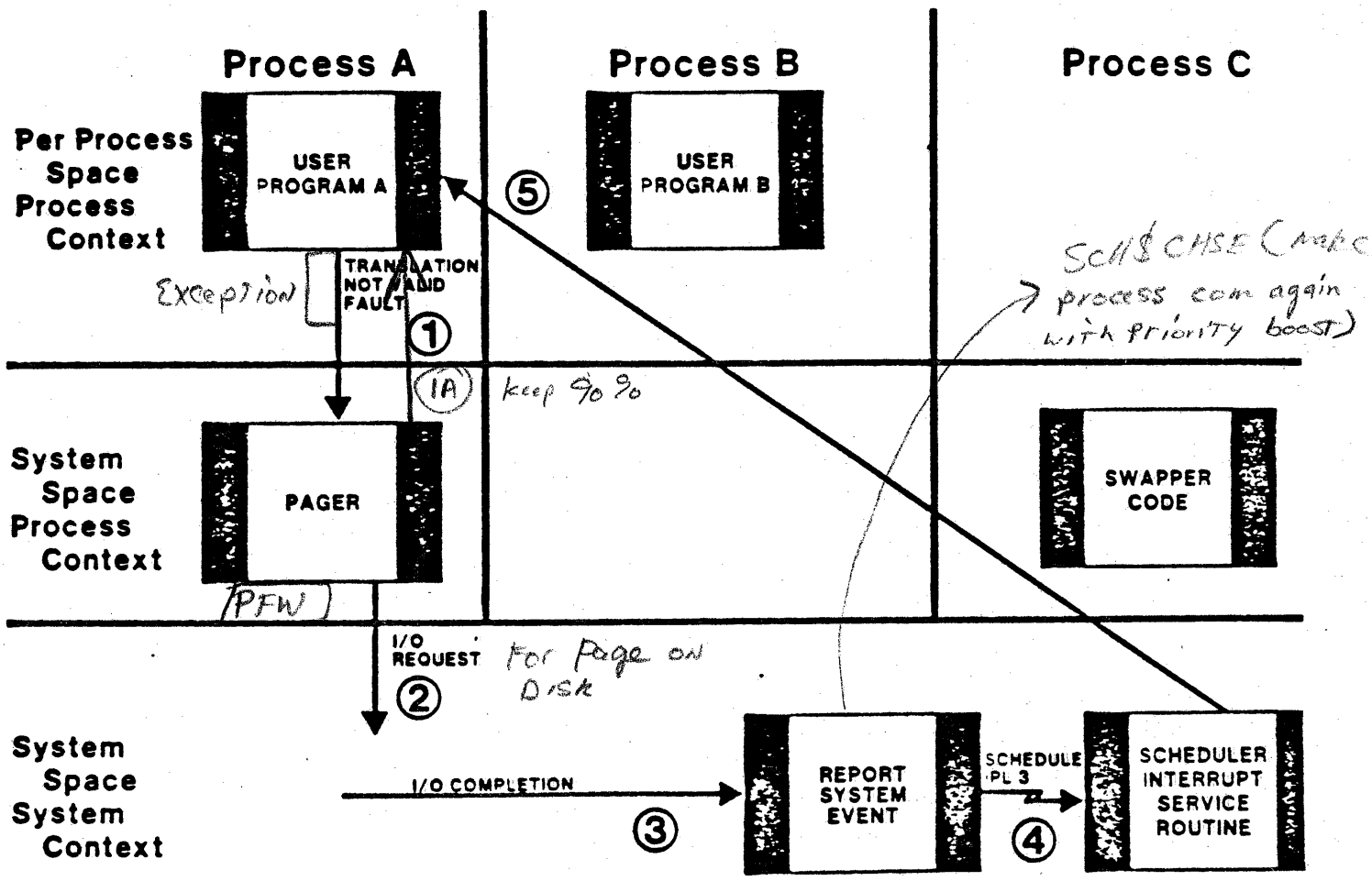
④ Call ERL$wake every second

- I/O ERROR ?
- I/O TIMEOUT ?
- ERROR log buffer(s) ? (TWO of them)
- 10 mins been performed to buffer(s) ?
- Has 30 seconds elapsed ?

Buffer(s) ←

SYS$ERRORLOG: ERRLOG.SYS

PAGER



① Look on FPL (free page list)] "cheap"
MPL (mod. page list)] Page fault (micro seconds)

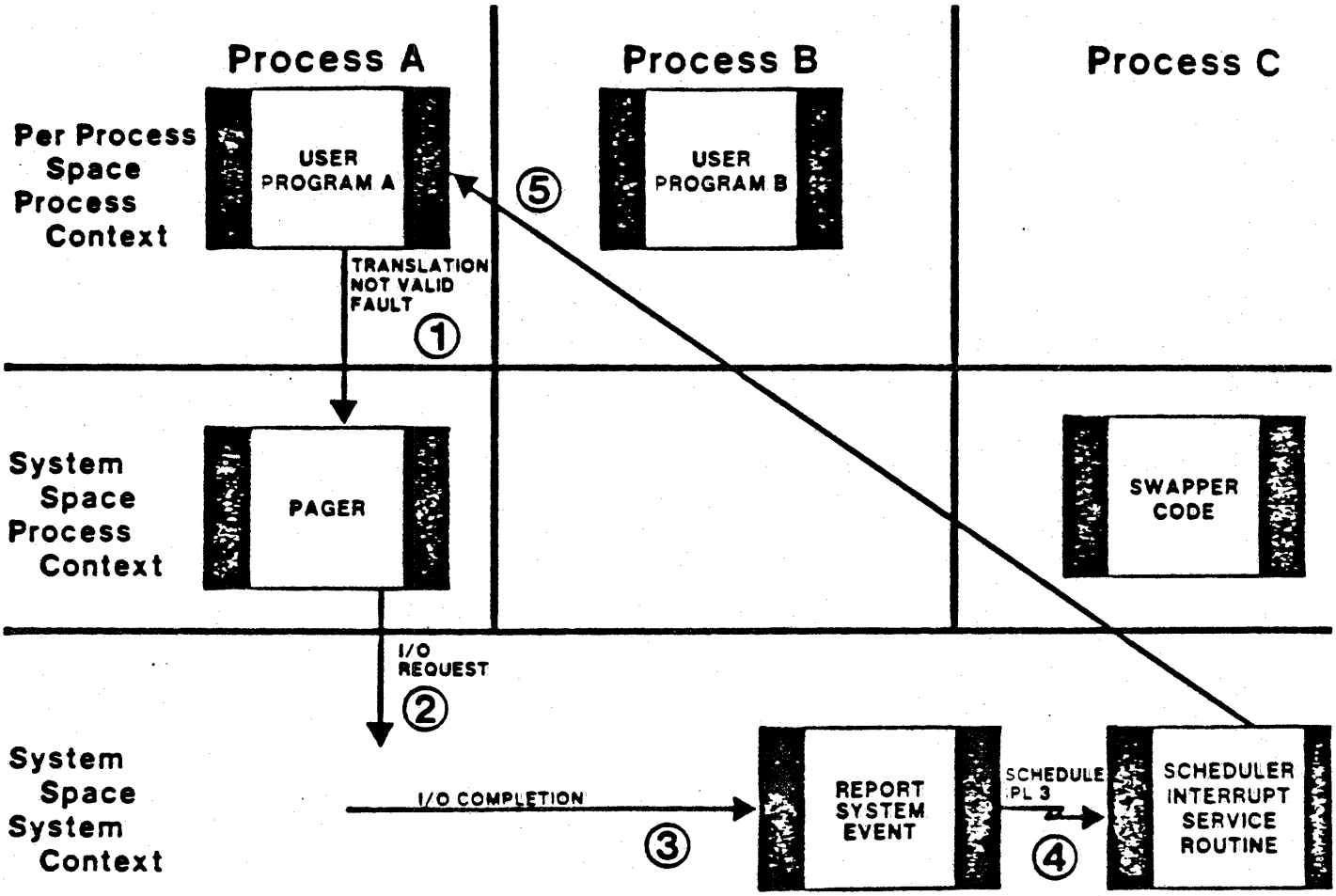
② need a disk access (hard fault)
10% (10's of ms)

monitor PAGE

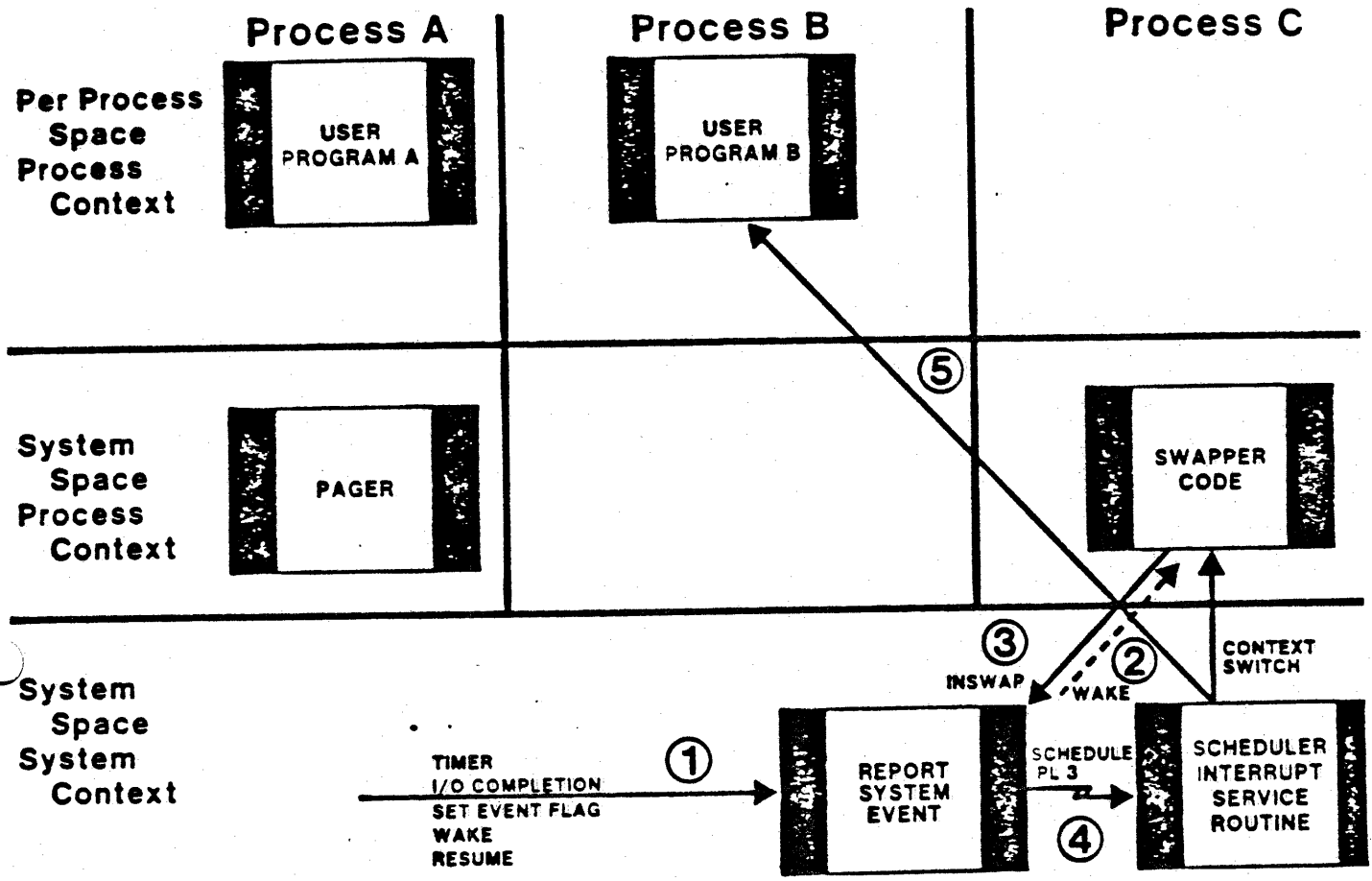
$$\frac{\text{Page Read I/O rate}}{\text{Page Fault Rate}} = \text{Percentage of Hard faults} \neq 100\%$$

$$\frac{\text{PMS\$GL\_PReadIO}}{\text{PMS\$GL\_faults}} = \%\ \text{Hard FLTS} \leq 10\%$$

Cause: WS to small, Badly sized FPL/MPL (too small) Lack Poor Code


Process A
Process B
Process C
Per Process Space Process Context
USER PROGRAM A
USER PROGRAM B
TRANSLATION NOT VALID FAULT
1
5
System Space Process Context
PAGER
SWAPPER CODE
I/O REQUEST
2
System Space System Context
I/O COMPLETION
3
REPORT SYSTEM EVENT
SCHEDULE IPL 3
4
SCHEDULER INTERRUPT SERVICE ROUTINE

SYSTEM EVENT REPORTING



$SetImr
$QIO

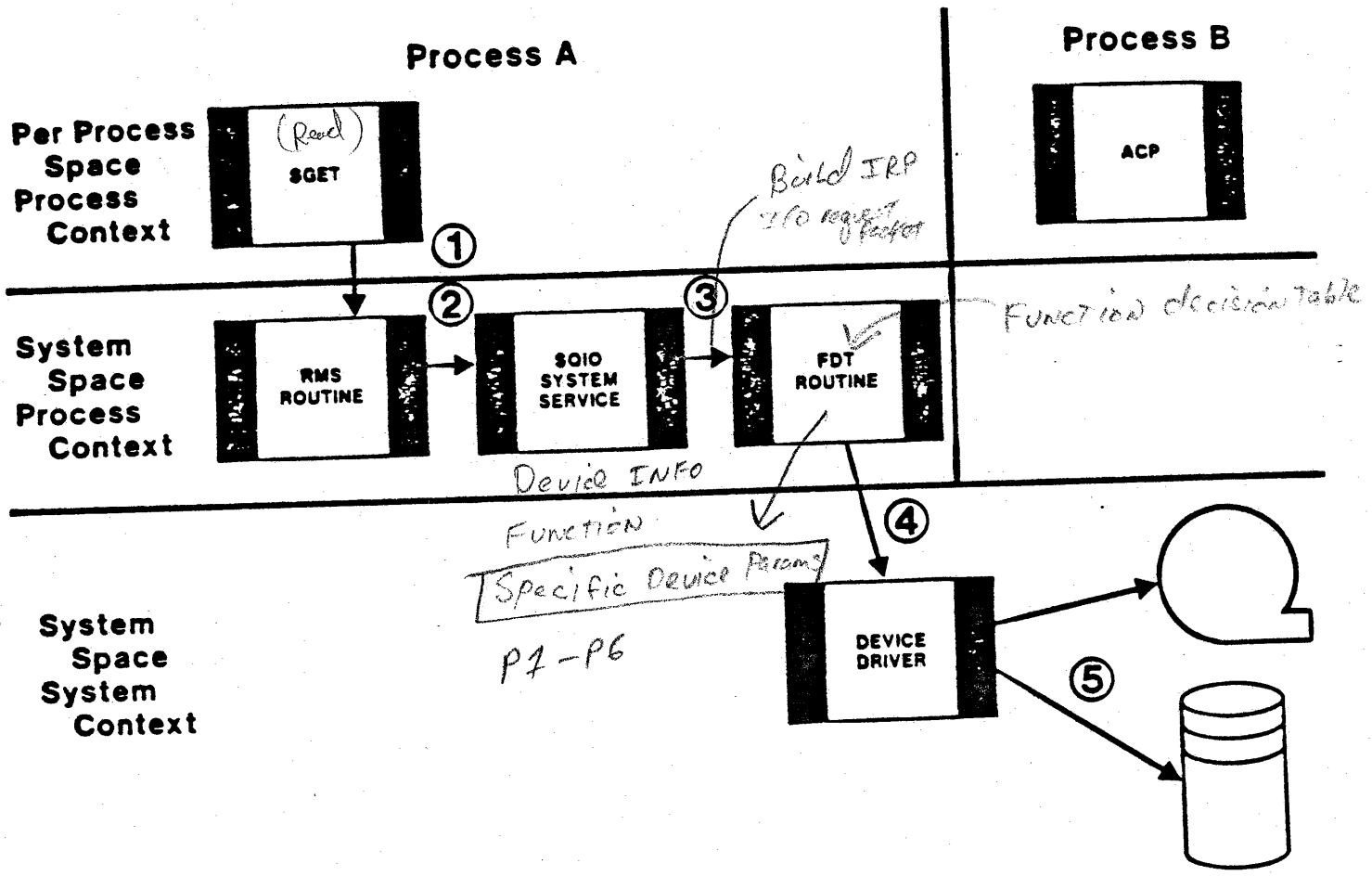
$QIOw)
$SNQ(w)
$GetJPI
$GetDVI
$GetSyI
$SetEF

LIB$

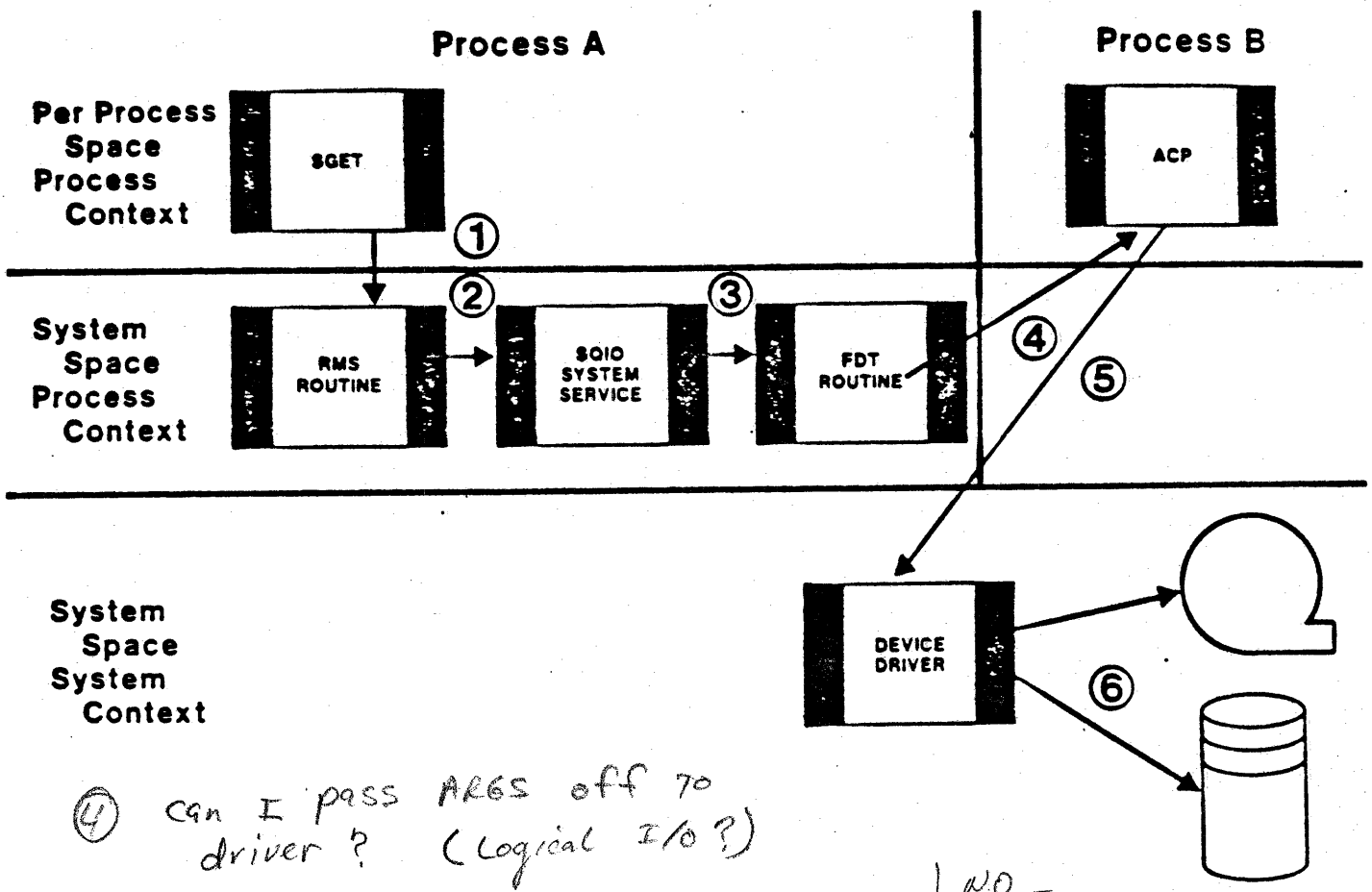
HIBO $Wake or $SchDwk
SUSPO $Resume

SCH$CHSE
(Com + Boost)

RMS DATA TRANSFER



ACCESSING DISK AND TAPE VIA AN ACP



④ Can I pass ARGS off to driver? (Logical I/O?)

↓ Yes

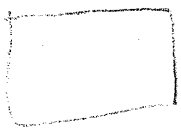
Jmp G^EXE$QIODRVPKT

↓ NO - Need ACP

⑤
- protection checks
- Virt – Logical Translation
- mapping pointers

Jmp G^EXE$QIOACPPKT

WCB (Window Control Block)

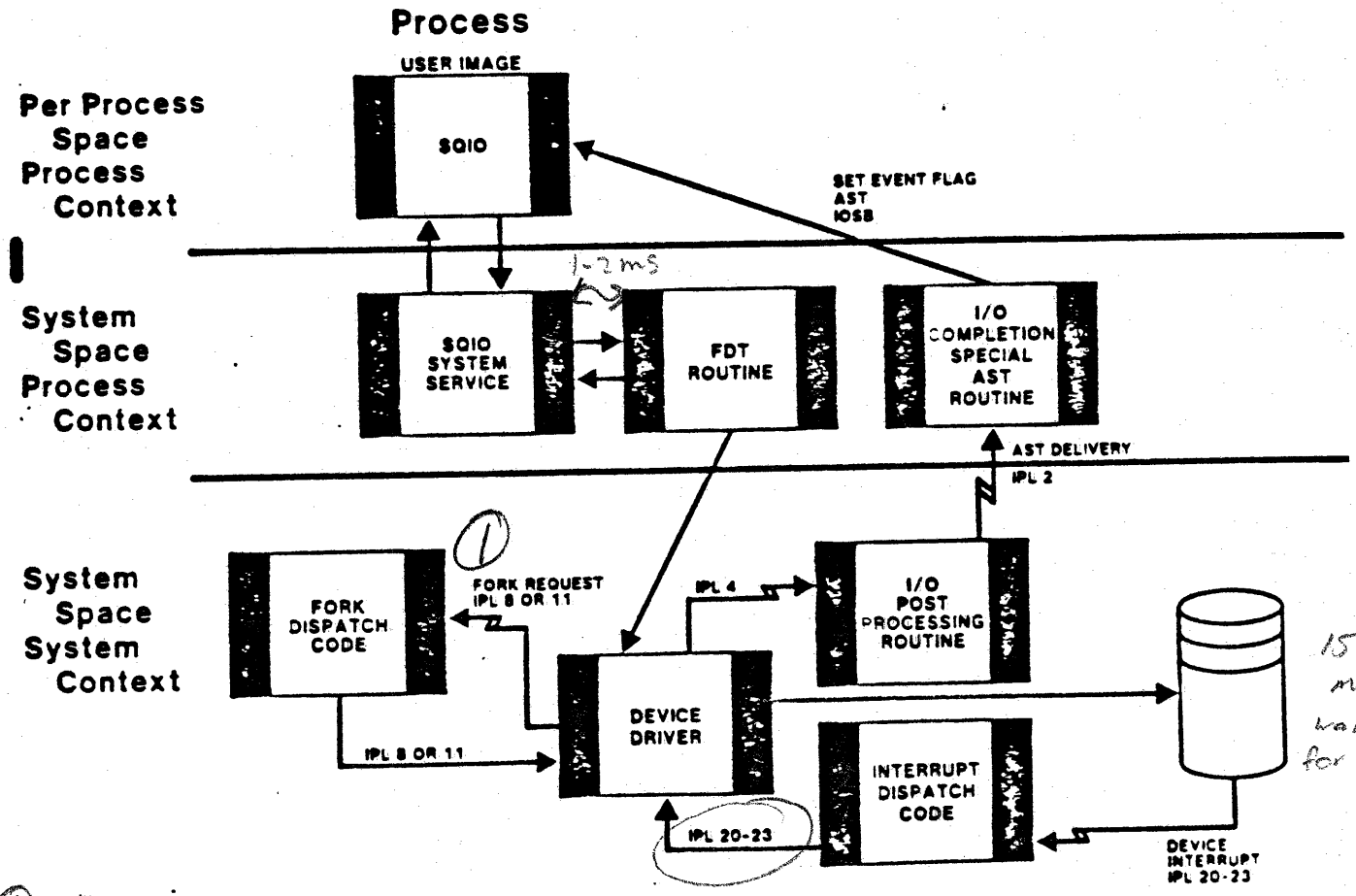


7 Long fragmentation Pointers

WINDOW TURN – MUST re-invoke ACP TO GET new pointers

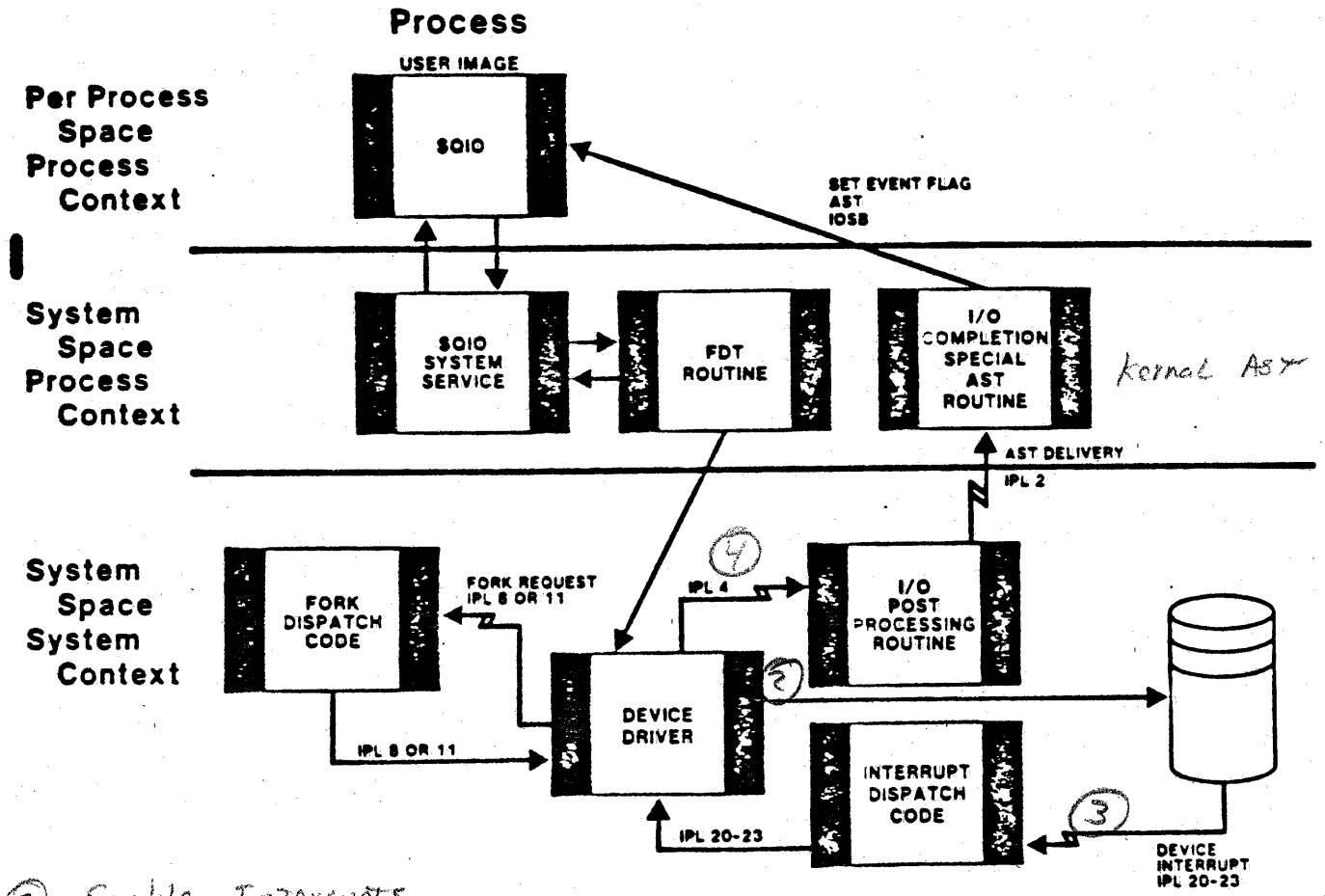
* monitor FCP IF >1 then must Backup/Restore PMS $ GL_TURN per second

QIO SEQUENCE



① Fork process
(dynamic process that "lives" in SØ space)
(IN UCB for device)
PC, R3, R4, R5

Chapter 16 IOSB



(2) Enable Interrupts
- WFIKPCH

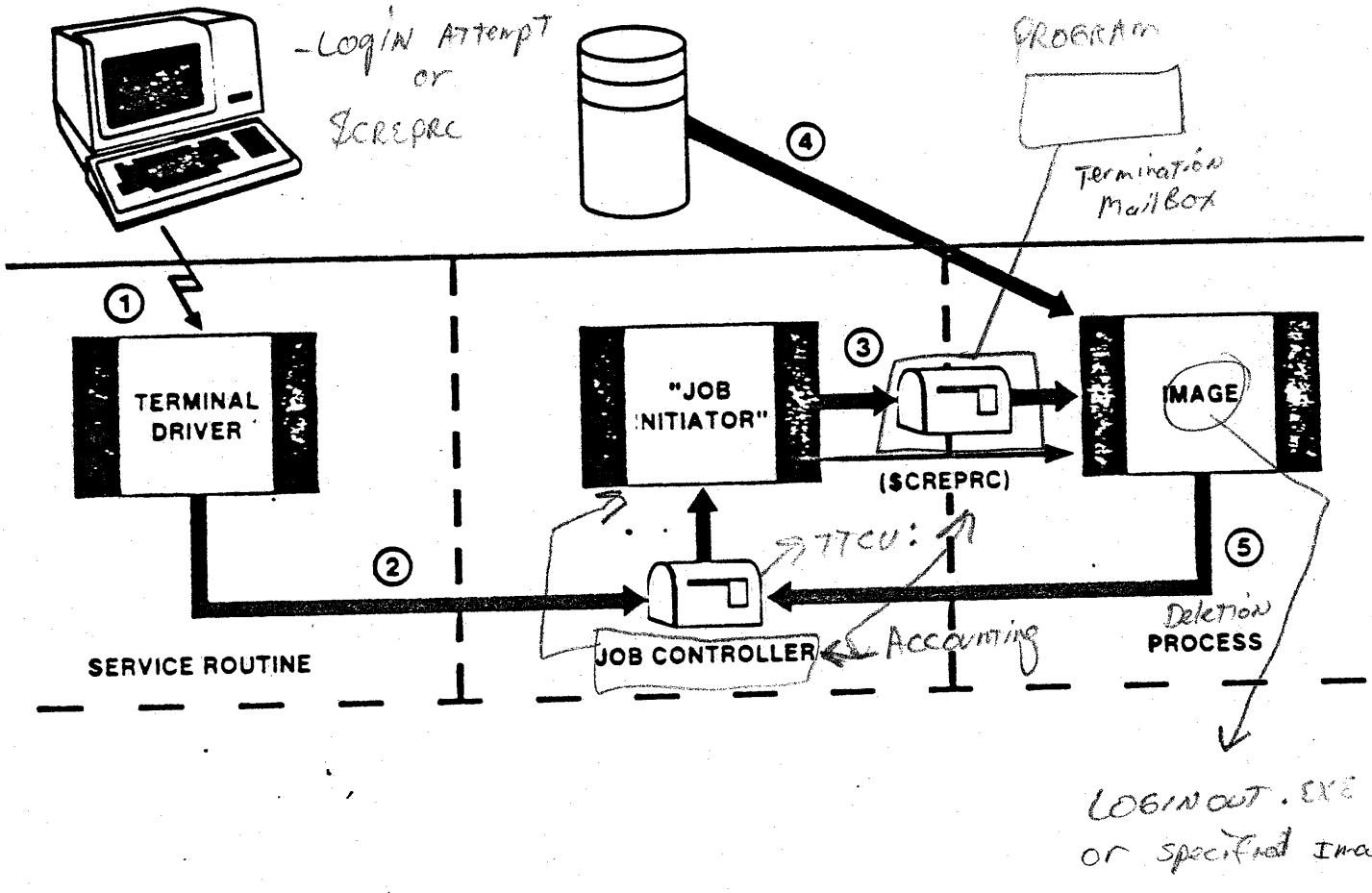
(3) Vector Through SCB (System Control Block)
To ISR (Interrupt Service Routine)

(4) REQ com (Request Complete)

Use $GETDVI
F$GETDVI

Opens 15-25 OP/sec
>25 Tomany QIO's
Try to balance disk QIO's
per Disk

TERMINAL INPUT



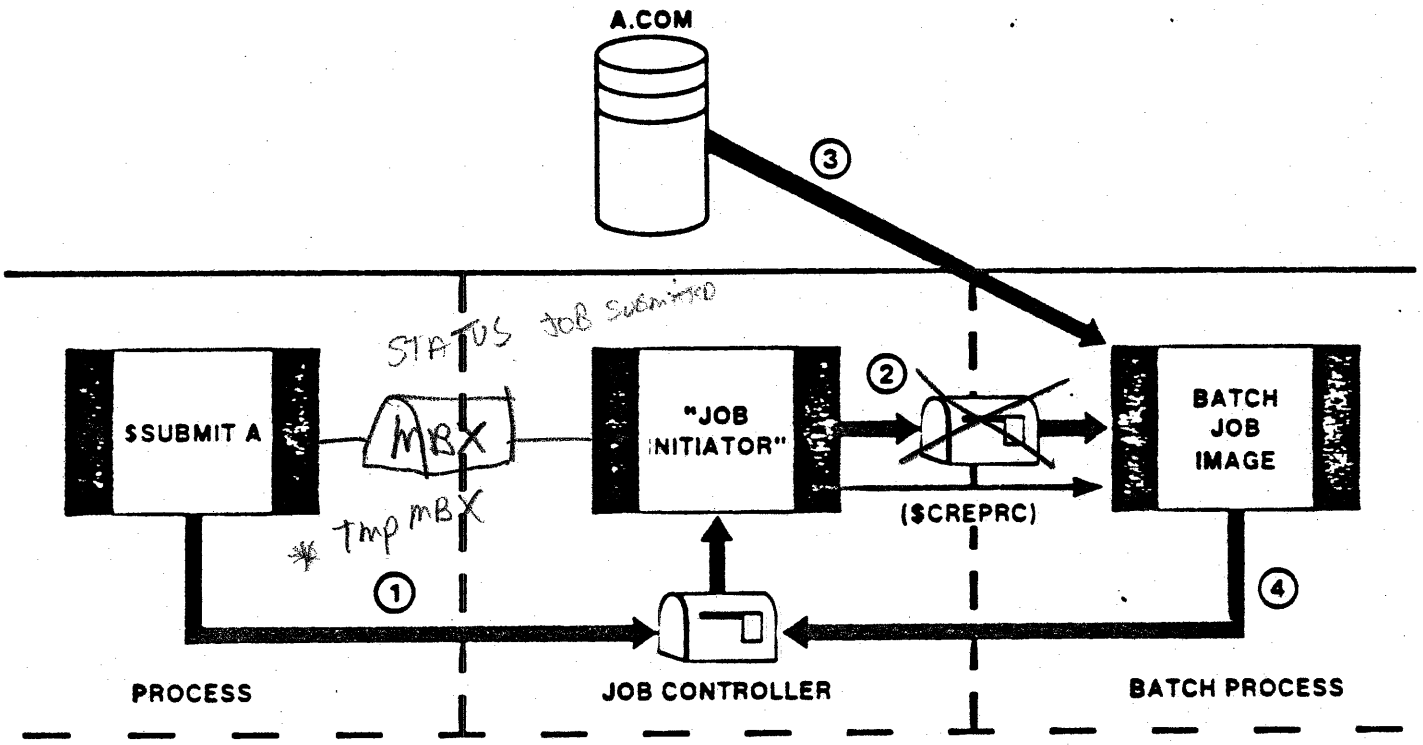
Termination MBX – Sys. Serv. Ref manual on $CREPRC

```
        .TITLE  NEWPROC         Program to create a process with DCL
        External symbols

        $IODEF
        $PRCDEF

        .PSECT  DATA,NOEXE
MAGE:   .ASCID  /SYS$SYSTEM:LOGINOUT.EXE/       ;Image to run
AME:    .ASCID  /SON_OF_PHIL/                   ;Name of subprocess
T:      .ASCID  /SYS$OUTPUT/                    ;Output for subproce
ERM_DESC:
        .WORD   5
        .BYTE   14,1
        .ADDRESS        TERM
ERM:    .BLKB   5
ROMPT:  .ASCII  /Enter name of terminal to  be used: /
ROMPT_LEN=.-PROMPT
LAG:    .LONG   PRC$V_LOGIN                     ;Flag set on creation
HAN:    .WORD   0

        Code

        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  NEWPROC,        ^M<>
        $ASSIGN_S       DEVNAM=TT,CHAN=CHAN     ;Get channel for I/O
        BLBC    R0,10$                          ;Check status
        $QIOW_S         CHAN=CHAN,-
                        FUNC=#IO$_READPROMPT,-
                        P1=TERM,P2=#5,-
                        P5=#PROMPT,P6=#PROMPT_LEN       ;QIO for ter
        BLBC    R0,10$                          ;Check status
        $CREPRC_S       IMAGE=IMAGE,INPUT=TERM_DESC,-
                        OUTPUT=TERM_DESC,PRCNAM=NAME,-
                        BASPRI=#4,STSFLG=FLAG
        BLBC    R0,10$                          ;Check status
        RET
0$:
        PUSHL   R0
        CALLS   #1,G^LIB$STOP                   ;Bomb with messages
        RET

        .END    NEWPROC
```

```
        .TITLE  TERMMBX         Program that uses termination mailbo

;       External symbols
;
        $ACCDEF
        $DVIDEF
        $IODEF

        .PSECT  DATA,NOEXE
ACC_BUF:
        .BLKB   84                          ;Length of termination buffe
DEV_BUF:
        .WORD   4
        .WORD   DVI$_UNIT
        .ADDRESS        MBX_NUMBER
        .LONG   0
        .LONG   0                           ;List for $GETDVI informatio
MBX_NUMBER:
        .WORD   0                           ;Storage for mbx number
NAME:   .ASCID  /Mailbox/                   ;Termination mailbox name
CHAN:   .WORD   0                           ;Storage for channel number
IMAGE:  .ASCID  /SUBPROG2.EXE/              ;Image to run in subprocess
PID:    .LONG   0                           ;Storage for PID number
CPU_TIME:
        .LONG   0                           ;Storage for CPU time used
CPU_CNV:
        .WORD   8
        .BYTE   14,1
        .ADDRESS        CPU_ASCII           ;Descriptor for CPU cnv.
CPU_OUTPUT:
        .WORD   STRING2_LEN + 8
        .BYTE   14,1
        .ADDRESS        STRING2
STRING2:
        .ASCII  /CPU time for subprocess was /
STRING2_LEN=.-STRING2
CPU_ASCII:
        .BLKB   8                           ;Storage for CPU output
PID_CNV:
        .WORD   8
        .BYTE   14,1
        .ADDRESS        PID_ASCII           ;Descriptor for PID cnv.
PID_OUTPUT:
        .WORD   STRING1_LEN + 8
        .BYTE   14,1
        .ADDRESS        STRING1
STRING1:
        .ASCII  /PID of subprocess was /
STRING1_LEN=.-STRING1
PID_ASCII:
        .BLKB   8                           ;Storage for PID output
```

```
;       Code
;
        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  TERMMBX,        ^M<>
        $CREMBX_S       CHAN=CHAN,-
                        LOGNAM=NAME     ;Create mailbox
        BLBC    R0,5$                   ;Check status
        $GETDVI_S       CHAN=CHAN,-
                        ITMLST=DEV_BUF  ;Get mbx unit number
        BLBC    R0,5$                   ;Check status
        $CREPRC_S       IMAGE=IMAGE,-
                        BASPRI=#4,  -
                        MBXUNT=MBX_NUMBER       ;Create subproc.
        BLBC    R0,5$                   ;Check status
        $QIOW_S         CHAN=CHAN,-
                        FUNC=#IO$_READVBLK,-
                        P1=ACC_BUF,-
                        P2=#84                  ;Read mailbox
        BLBC    R0,5$                   ;Check status
        BLBS    R0,6$
5$:
        PUSHL   R0
        CALLS   #1,G^LIB$STOP           ;Exit with errors
        RET
6$:
        MOVL    ACC_BUF+ACC$L_PID,PID           ;Get PID number
        MOVL    ACC_BUF+ACC$L_CPUTIM,CPU_TIME   ;Get CPU time
        PUSHAL  PID_CNV
        PUSHAL  PID
        CALLS   #2,G^OTS$CVT_L_TZ       ;Convert PID to ASCII
        BLBC    R0,10$
        PUSHAL  PID_OUTPUT              ;Descriptor for output
        CALLS   #1,G^LIB$PUT_OUTPUT     ;And display it
        BLBC    R0,10$
        PUSHAL  CPU_CNV                 ;Descriptor
        PUSHAL  CPU_TIME                ;Storage
        CALLS   #2,G^OTS$CVT_L_TI       ;Convert CPU to ASCII
        BLBC    R0,10$
        PUSHAL  CPU_OUTPUT              ;Descriptor for output
        CALLS   #1,G^LIB$PUT_OUTPUT     ;And display it
        BLBC    R0,10$
        RET                             ;All finished
10$:
        PUSHL   R0                      ;Put error code on stack
        CALLS   #1,G^LIB$STOP           ;And exit with messages
        RET

        .END    TERMMBX
```

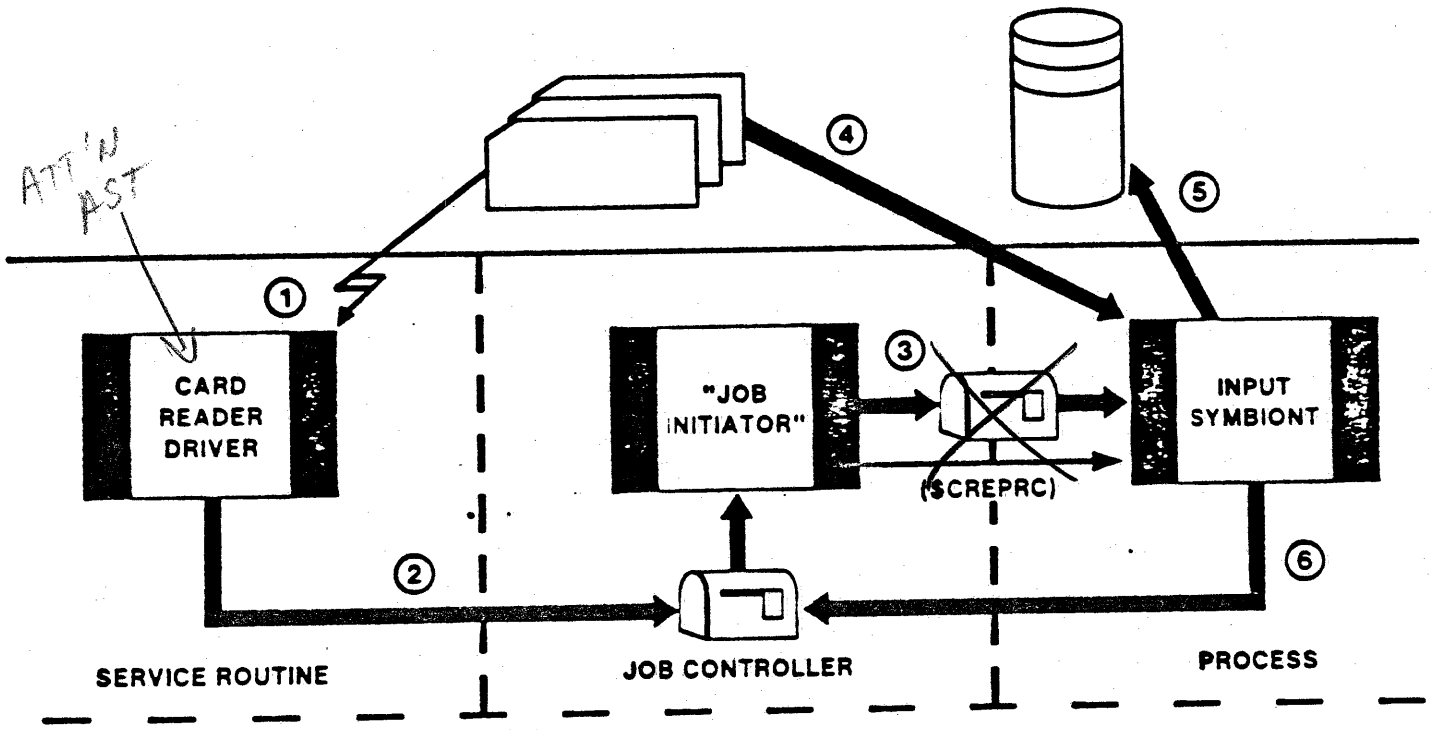
BATCH JOBS



SYS$OUTPUT
SYS$ERROR
→ A.LOG

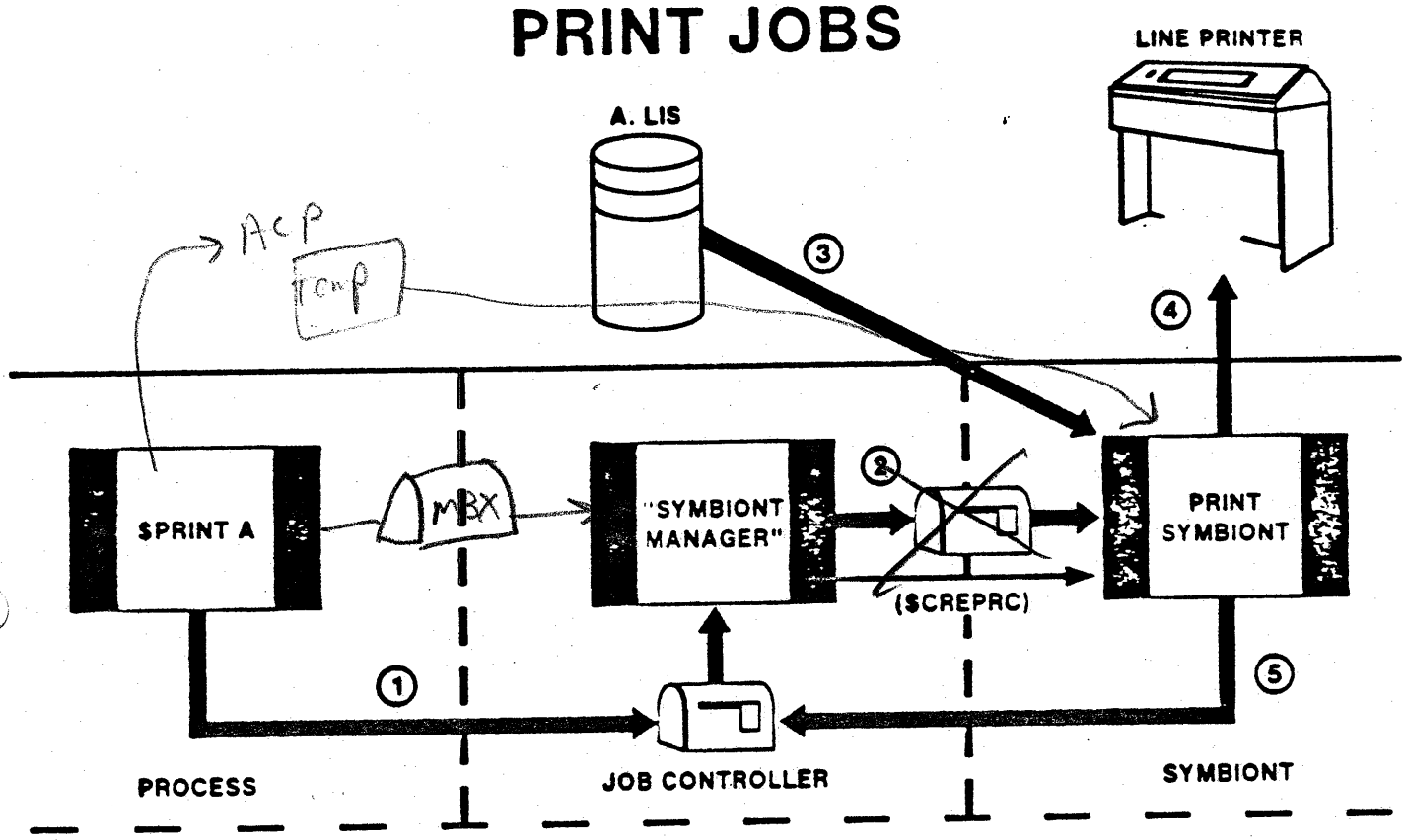
SYS$INPUT
SYS$COMMAND
→ A.COM

CARD INPUT



SYS$INPUT - CARD Deck

PRINT JOBS



START/Queue creates print Symbiont subprocess

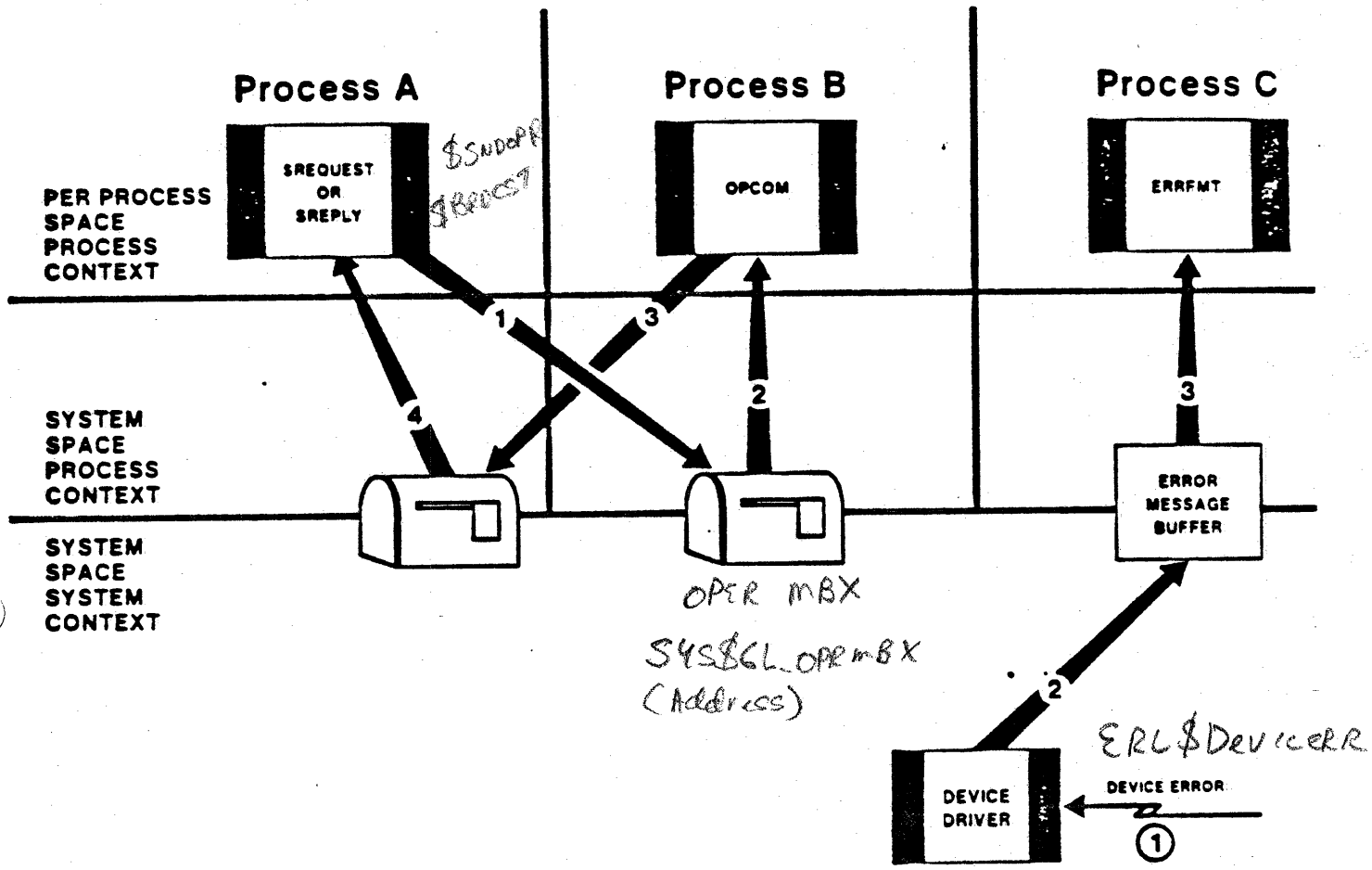
Tmp mbx needed to print

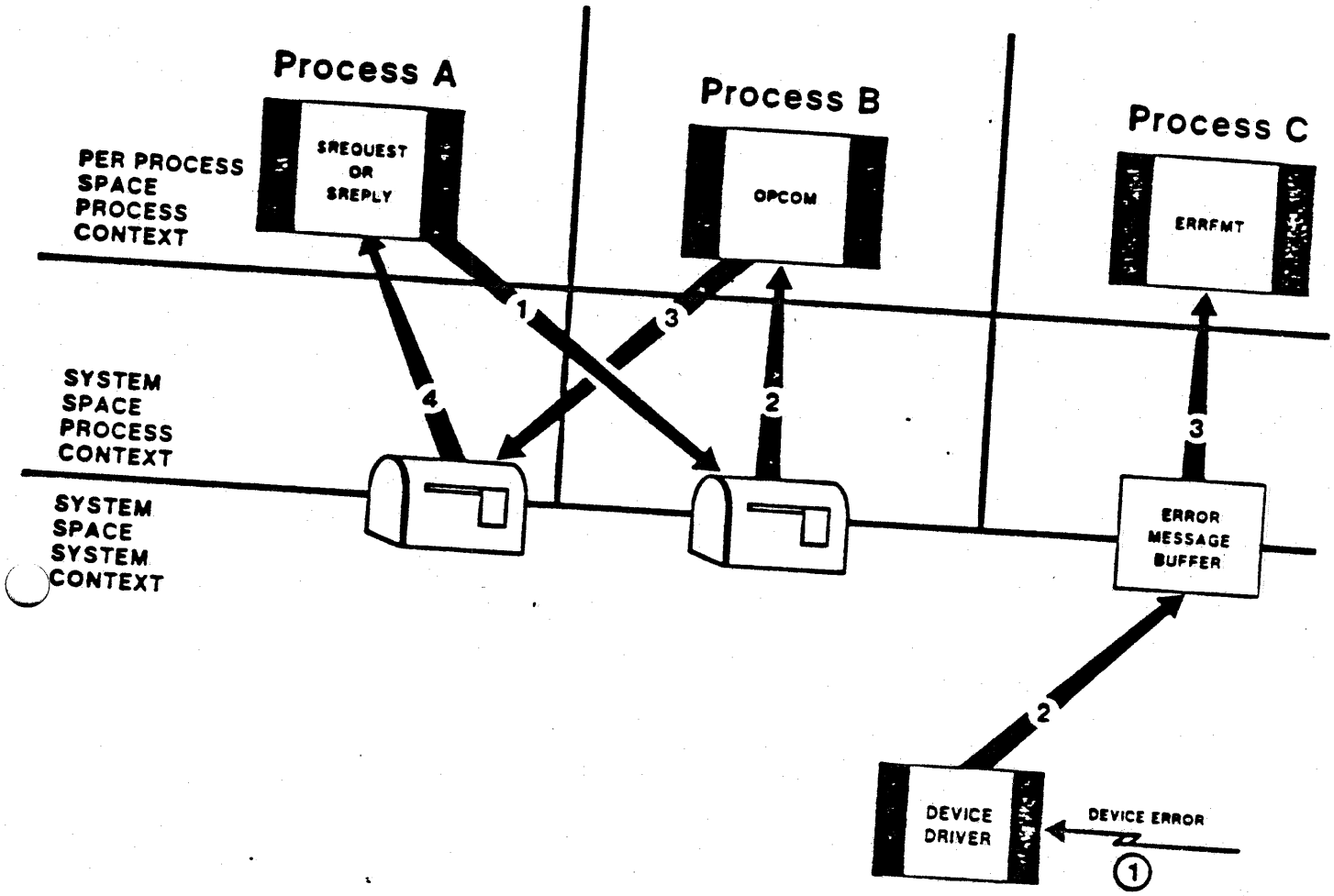
Spool Lineprinters — Set Device/Spool LPA0: Disk.

↙

Disk Quotas checked

ERROR LOGGER OPCOM




Process A
Process B
Process C
PER PROCESS SPACE PROCESS CONTEXT
SYSTEM SPACE PROCESS CONTEXT
SYSTEM SPACE SYSTEM CONTEXT
SREQUEST OR SREPLY
OPCOM
ERRFMT
ERROR MESSAGE BUFFER
DEVICE DRIVER
DEVICE ERROR
1
2
3
4
2
3
2
1

VAX/VMS OPERATING SYSTEM INTERNALS

PROGRAMMING LAB EXERCISES

NOTE: The labs presented with the OSI course have been developed by instructors in Rolling Meadows and Santa Clara training facilities. It was felt that the coverage of the material in the OSI course would be complemented by students writing code to access some of the data structures presented during the week. It is hoped that this technique will give the student a better 'feel' for the material covered.

Solutions have been provided for each lab. Find the lab problems that interest you and attack them.

You will find the lab problems "broken up" in each day by difficulty criterion; "easy", "medium", or "difficult". You should attempt to perform the "easy" or "medium" labs for each day.

Note that for each problem, two solutions are provided; (1) a "template" containing the complete solution EXCEPT for the kernel/elevated IPL code; and (2) a fully complete solution. This strategy provides you with the choice of completing the template and running it, and/or write the code from scratch.

As a matter of courtesy to your group and others to follow, please do not modify any of the files in [INTERNALS...]. As you progress to each lab, copy the file(s) into your [OSIn] directory.

Since we do not have a lot of lab time, use your time wisely to get a feel for the system. Do not be afraid to crash the system, but analyze the crash dump if you do crash it.

Good luck !

ADDENDUM

We have tried to interface high level languages to the lab problems. In the directory [INTERNALS.HIGHLEVEL], you will find the lab problems "redone", by combining FORTRAN and MACRO. The problems are named to correspond with the problem number (example: problem 1 LAB1.FOR (main code) LAB1.MAR (subroutine)). Note that while we haven't done all of the labs yet, you may wish to look at some of these programs for future reference/ideas.

MONDAY - TUESDAY

1) Write a privileged program to output, and optionally to change, your ACCOUNT NAME (Hint: account name is stored in your P1 space).

EASY

Solutions: [INTERNALS.MONTUES]ACCNTTEMP.MAR (template)
[INTERNALS.MONTUES]ACCOUNT.MAR (complete)

2) Write a privileged program to output the addresses of your PCB and PHD.

EASY

Solutions: [INTERNALS.MONTUES]PCBTEMP.MAR (template)
[INTERNALS.MONTUES]PCB.MAR (complete)

3) Write a privileged program to output the contents of your PCB and JIB (use a HEX longword dump format).

MEDIUM

Solutions: [INTERNALS.MONTUES]PCBJIBTMP.MAR (template)
[INTERNALS.MONTUES]PCBJIB.MAR (complete)

4) Write a privileged shareable image (user-written system service), to return the contents of the System Identification Register.

DIFFICULT

Solutions: Using the template in
[INTERNALS.MONTUES]:USSDISP.MAR
USSTEST.MAR

[INTERNALS.MONTUES]LAB4.MAR
[INTERNALS.MONTUES]LAB4TEST.MAR

(NOTE: due to the nature of this problem, NO template is is provided; only the COMPLETE solutions).

(Hint for #4: Copy [INTERNALS.MONTUES]USS*.* to directory.
Edit USSDISP.MAR to include service call.
Assemble USSDISP.MAR.
Use USSLNK.COM to link dispatcher/service code.
Edit USSTEST.MAR.
Assemble USSTEST.MAR.
Copy USS.EXE to SYS$SHARE.
INSTALL SYS$SHARE:USS.EXE/SHARE/PROTECT
Use USSTSTLNK.COM to link test code with your system service.

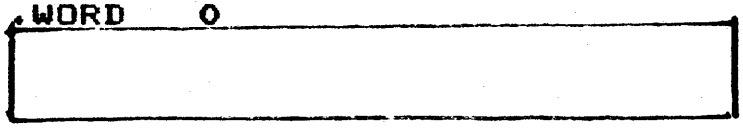
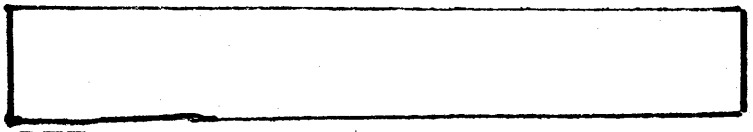
```
        .TITLE  ACCNTTEMP
;
;       Program to change P1 control information (account name)
;
        .LIBRARY        /MAR$EXTRA:MACROLIB/

        .PSECT  NONSHARED_DATA  PIC,NOEXE,LONG

MESS1:  .ASCID  /Account name: /
PROMPT: .ASCID  /Enter account name (1-8) characters: /
ACC_NAME:                               ;Descriptor for
        .WORD   8                       ;account name
        .BYTE   14,1
        .ADDRESS        ACC_BUF
ACC_BUF:
        .BLKB   8

ARG_LIST:                               ;Argument list
        .LONG   1                       ;for CHMK routines
        .ADDRESS        ACC_BUF
BUFFER: .WORD   80                      ;Descriptor for
        .BYTE   14,1                    ;string concats
        .ADDRESS        BUF
BUF:    .BLKB   80
LENGTH: .BLKW   1                       ;Storage for prompt

        .PSECT  CODE            PIC,SHR,NOWRT,LONG
        .ENTRY  ACCNTTEMP,      ^M<>
        $CMKRNL_S       routin=10$,-
                        arglst=ARG_LIST

        CONCAT2 BUFFER,MESS1,ACC_NAME   ;Put strings together
        DISPLAY BUFFER                  ;And show it
        PUSHAW  LENGTH                  ;Prompt for "new"
        PUSHAL  PROMPT                  ;account name
        PUSHAL  ACC_NAME
        CALLS   #3,G^LIB$GET_INPUT
        CHECK_STATUS

        MOVW    LENGTH,ACC_NAME         ;Get exact length
        $CMKRNL_S       routin=20$,-    ;And change it
                        arglst=ARG_LIST

        RET

10$:    .WORD   0
        [redacted]                      ;Get account name
                                        ;Successful status
        RET

20$:    .WORD   0
        [redacted]                      ;Change name
                                        ;Successful status
        RET

        .END    ACCNTTEMP
```

```
;*****************************************************************
;       Program name:   PCBTEMP 01/04/83        V 1.0
;
;       Author:         Phil Lovecchio
;                       Senior Educational Specialist
;                       Digital Equipment Corporation
;                       Rolling Meadows Training Facility
;
;       Function:       Get PCB and PHD addresses for process
;
;       Privileges:     CMKRNL
;
;       Inputs:         None
;
;       Outputs:        PCB and PHD addresses
;
;       Side effects:   Hopefully none
;
;
;*****************************************************************
;
        .TITLE  PCBTEMP List addresses of PCB and PHD
        .PSECT  DATA,NOEXE
;Program to list out the addresses of the PCB and PHD.  Note
;the use of conversion routines and run-time library routines
;in the code
;
;*****************************************************************
;
;External symbols
;
        $PCBDEF
;
;Generate values for PCB address
;
PCB_VAL::                               ;Value of PCB address
        .LONG
PHD_VAL::                               ;Value of PHD address
        .LONG
PCB_OUTPUT:                             ;Descriptor for output
        .WORD   PCB_LEN + 8             ;Length is header plus 8
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        STRING1
STRING1:
        .ASCII  /Address of PCB is: /   ;Header for PCB address
PCB_LEN=.-STRING1
PCB_ASCII:                              ;ASCII PCB address
        .BLKB   8
PCB_CNV_DESC:                           ;Conversion descriptor
        .WORD   8                       ;Eight characters
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        PCB_ASCII
;*****************************************************************
;
;Generate same values for PHD address
;
PHD_OUTPUT:                             ;Descriptor for PHD output
        .WORD   PHD_LEN + 8             ;Length is header plus 8
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        STRING2
STRING2:                                ;Header for PHD output
        .ASCII  /Address of PHD is: /
PHD_LEN=.-STRING2
```

```
PHD_ASCII:                                          ;ASCII PHD address
        .BLKB   8                                   ;Maximum is eight characters
PHD_CNV_DESC:                                       ;Conversion descriptor
        .WORD   8
        .BYTE   14,1                                ;Type=text  Class=scalar
        .ADDRESS        PHD_ASCII
;
;Code
;
        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  PCBTEMP,        ^M<>
        $CMKRNL_S       ROUTIN=10$                  ;Change mode
        BLBC    R0,20$                              ;Check for errors
        PUSHAL  PCB_CNV_DESC                        ;Descriptor for conversion
        PUSHAL  PCB_VAL                             ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ                   ;Convert hex to ASCII
        BLBC    R0,20$                              ;Check for errors
        PUSHAL  PCB_OUTPUT                          ;Output for PCB address
        CALLS   #1,G^LIB$PUT_OUTPUT                 ;Display it
        BLBC    R0,20$                              ;Check for errors
        PUSHAL  PHD_CNV_DESC                        ;Descriptor to set conversio
        PUSHAL  PHD_VAL                             ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ                   ;Convert hex to ASCII
        BLBC    R0,20$                              ;Check for errors
        PUSHAL  PHD_OUTPUT                          ;Output for PHD address
        CALLS   #1,G^LIB$PUT_OUTPUT                 ;Display it
        BLBC    R0,20$                              ;Check for errors
        RET                                         ;All done

10$:    .WORD   0                                   ;PCB address is in R4
        [redacted]                                  ;Get PCB address
                                                    ;Offset to set PHD
        RET                                         ;Back to main code

20$:    PUSHL   R0                                  ;Get error code for routine
        CALLS   #1,G^LIB$STOP                       ;Bomb with error message
        RET                                         ;Exit with status

        .END    PCBTEMP
```

```
;*****************************************************************************
;
;       Program name:   PCBJIBTMP
;
;       Author:         Phil Lovecchio
;                       Senior Educational Specialist
;                       Digital Equipment Corporation
;                       Rolling Meadows Training Facility
;
;       Function:       List out contents of PCB and JIB for process
;
;       Privileges:     CMKRNL
;
;       Input:          None
;
;       Outputs:        Contents of PCB and JIB
;
;       Side effects:   None as known
;
;*****************************************************************************
;
        .TITLE  PCBJIBTMP       List contents of PCB and JIB
        .PSECT  DATA,NOEXE,WRT,NOSHR
;Program to list out the contents of the PCB and JIB.  Note the
;use of conversion and run-time library output routines.
;*****************************************************************************
;
;External symbols
;
        $PCBDEF
        $JIBDEF
;
;Generate values for PCB address
;
PCB_VALUES:                             ;Storage for PCB values
        .BLKB   PCB$K_LENGTH
PCB_ASC_DESC:                           ;Descriptor for PCB values
        .WORD   8                       ;Size eight bytes max
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        PCB_ASC_VALUE
PCB_ASC_VALUE:                          ;Destination for conversion
        .BLKB   8
;*****************************************************************************
JIB_VALUES:                             ;Storage for JIB values
        .BLKB   JIB$K_LENGTH
JIB_ASC_DESC:                           ;Descriptor for JIB values
        .WORD   8
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        JIB_ASC_VALUE
JIB_ASC_VALUE:                          ;Destination for conversion
        .BLKB   8
;*****************************************************************************
PCB_VAL:                                ;Binary value of PCB address
        .LONG
JIB_VAL:                                ;Binary value of JIB address
        .LONG
;*****************************************************************************
PCB_OUTPUT:                             ;Descriptor for output strin
        .WORD   PCB_LEN + 8             ;Length is header plus 8
        .BYTE   14,1                    ;Type=text  Class=scalar
        .ADDRESS        STRING1
STRING1:
        .ASCII  /Address of PCB is: /   ;Header for PCB address outp
```

```
PCB_LEN=.-STRING1
PCB_ASCII:                                      ;ASCII PCB address
        .BLKB   8
PCB_CNV_DESC:                                   ;Conversion descriptor
        .WORD   8                               ;Maximum of ten characters
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        PCB_ASCII
;*****************************************************************************
;
;Generate same values for JIB address
;
JIB_OUTPUT:                                     ;Descriptor for JIB output
        .WORD   JIB_LEN + 8                     ;Length is header plus 8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        STRING2
STRING2:                                        ;Header for JIB output
        .ASCII  /Address of JIB is: /
JIB_LEN=.-STRING2
JIB_ASCII:                                      ;ASCII JIB address
        .BLKB   8
JIB_CNV_DESC:                                   ;Conversion descriptor
        .WORD   8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        JIB_ASCII
STRING3:
        .ASCID  /Contents of PCB/               ;Header for PCB output
STRING4:
        .ASCID  /Contents of JIB/               ;Header for JIB output
;
;Code
;
        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  PCBJIBTMP,      ^M<>
        $CMKRNL_S       ROUTIN=10$              ;PCB address is in R4
        PUSHAL  PCB_CNV_DESC                    ;Descriptor for conversion
        PUSHAL  PCB_VAL                         ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert hex to ASCII
        PUSHAL  PCB_OUTPUT                      ;Output for PCB address
        CALLS   #1,G^LIB$PUT_OUTPUT             ;Display it
;
;Use autoincrement to convert and display PCB values
;
        MOVL    #PCB$K_LENGTH/4,R3              ;Counter for displays
        MOVAL   PCB_VALUES,R2                   ;PCB value address
        PUSHAL  STRING3                         ;Descriptor of value
        CALLS   #1,G^LIB$PUT_OUTPUT             ;And display it
5$:     PUSHAL  PCB_ASC_DESC                    ;Descriptor for value
        PUSHAL  (R2)+                           ;Get next value
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert to ASCII
        PUSHAL  PCB_ASC_DESC
        CALLS   #1,G^LIB$PUT_OUTPUT             ;And display it
        SOBGTR  R3,5$                           ;Done ? ; if so, now JIB
;
;Now do the same for the JIB
;
        PUSHAL  JIB_CNV_DESC                    ;Descriptor for conversion
        PUSHAL  JIB_VAL                         ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert hex to ASCII
        PUSHAL  JIB_OUTPUT
        CALLS   #1,G^LIB$PUT_OUTPUT
;
;Use autoincrement to convert and display JIB values
;
        MOVL    #JIB$K_LENGTH/4,R3              ;Same as for PCB values
```

```
        CALLS   #1,G^LIB$PUT_OUTPUT
5$:     PUSHAL  JIB_ASC_DESC
        PUSHAL  (R2)+
        CALLS   #2,G^OTS$CVT_L_TZ
        PUSHAL  JIB_ASC_DESC
        CALLS   #1,G^LIB$PUT_OUTPUT
        SOBGTR  R3,6$
        RET                                     ;All done

10$:    .WORD   0                               ;PCB address in R4
        
                                                ;Get PCB address
                                                ;Offset to JIB address
;
;U   sizes of PCB and JIB to set contents
;
        
                                                ;PCB values
                                                ;JIB values
        RET                                     ;Back to main code

20$:    PUSHL   R0                              ;Get error code for routine
        CALLS   #1,G^LIB$STOP                   ;Bomb with error message
        RET                                     ;Exit with status

        .END    PCBJIBTMP
```

(LAB4)

```
        .TITLE  USER_SYS_DISP - Example of user system service dispatcher
        .IDENT  'V02-000'
;*****************************************************************************

; Facility: Example of User Written System Services
;++
; Abstract:
;       This module contains an example dispatcher for user written
;       system services along with several sample services and a user
;       rundown example.  It is a template intend to serve as the starting
;       point for implementing a privileged shareable image containing your
;       own services.  When used as a template, the definitions and code
;       for the sample services should be removed.
;
; Overview:
;       User written system services are contained in privileged shareable
;       images that are linked into user program images in exactly the
;       same fashion as any shareable image.  The creation and installation
;       of a privileged, shareable image is slightly different from that
;       of an ordinary shareable image.  These differences are:
;
;               1. A vector defining the entry points and providing other
;                  control information to the image activator.  This vector
;                  is a the lowest address in an image section with the VEC
;                  attribute.
;
;               2. The shareable image is linked with the /PROTECT option
;                  that marks all of the image sections so that they will
;                  protected and given EXEC mode ownership by the image
;                  activator.
;
;               3. The shareable image MUST be installed /SHARE /PROTECT
;                  with the INSTALL utility in order for the image activator
;                  to connect the privileged shareable image to the change mode
;                  dispatchers.
;
;       A privileged shareable image implementing user written system services
;       is comprised of the following major components:
;
;               1. A transfer vector containing all of the entry points and
;                  collecting them at the lowest virtual address in the shareable
;                  image.  This formalism enables revision of the shareable
;                  image without necessitating the relinking of images that
;                  use it.
;
;               2. A Privileged Library Vector in a PSECT with the VEC attribute
;                  that describes the entry points for dispatching EXEC and
;                  KERNEL mode services along with validation information.
;
;               3. A dispatcher for kernel mode services.  This code will
;                  be called by the VMS change mode dispatcher when it
;                  fails to recognize a kernel mode service request.
;
;               4. A dispatcher for executive mode services.  This code will
;                  be called by the VMS change mode dispatcher when it fails
;                  to recognize an executive mode service request.
;
;               5. Service routines to perform the various services.
;
;       The first four components are contained in this template and are
;       most easily implemented in MACRO, while the service routines can
```

```
        .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
        .BYTE   NARG                    ; Define number of required arguments

        .PSECT  USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
        .WORD   2+NAME-ECASE_BASE       ; Make entry in exec mode CASE table
        .ENDC                           ;
        .ENDM   DEFINE_SERVICE          ;

        Equated Symbols


        $PHDDEF                         ; Define process header offsets
        $PLVDEF                         ; Define PLV offsets and values
        $PRDEF                          ; Define processor register numbers

        Initialize counters for change mode dispatching codes

ERNEL_COUNTER=0                         ; Kernel code counter
XEC_COUNTER=0                           ; Exec code counter


        Own Storage

        .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
ERNEL_NARG:                             ; Base of byte table containing the
                                        ;  number of required arguments.
        .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
XEC_NARG:                               ; Base of byte table containing the
                                        ;  number of required arguments.
        .PAGE
        .SBTTL  Transfer Vector and Service Definitions
++
 T  use of transfer vectors to effect entry to the user written system services
 enables some updating of the shareable image containing them without necessitatin
 a re-link of all programs that call them.  The PSECT containinng the transfer
 vector will be positioned at the lowest virtual address in the shareable image
 and so long as the transfer vector is not re-ordered, programs linked with
 one version of the shareable image will continue to work with the next.

 Thus as additional services are added to a privileged shareable image, their
 definitions should be added to the end of the following list to ensure that
 programs using previous versions of it will not need to be re-linked.
 To completely avoid relinking existing programs the size of the privileged
 shareable image must not change so some padding will be required to provide
 the opportunity for future growth.
--
        DEFINE_SERVICE  USER_GET_SID,1,KERNEL   ; Service to get value system
                                                ;  identification register
 The base values used to generate the dispatching codes should be negative for
 user services and must be chosen to avoid overlap with any other privileged
 shareable images that will be used concurrently.  Their definition is
 deferred to this point in the assembly to cause their use in the preceding
 macro calls to be forward references that guarantee the size of the change
 mode instructions to be four bytes.  This satisfies an assumption that is
 made by for services that have to wait and be retried.  The PC for retrying
 the change mode instruction that invokes the service is assumed to be 4 bytes
 l   than that saved in the change mode exception frame.  Of course, the particul
 service routine determines whether this is possible.

ODE_BASE=-1024                          ; Base CHMK code value for these services
ODE_BASE=-1024                          ; Base CHME code value for these services
        .PAGE
        .SBTTL  Change Mode Dispatcher Vector Block
```

```
3G_LEN=.-MSG
        .PAGE
        .SBTTL  Get System ID Register Value
++
 Functional Description:
        This routine reads the content of the system identification
        processor register and stores the resulting value at the
        specified address.

 Input Parameters:
        04(AP) - Address to return identification value
        R4 - Address of current PCB

 Output Parameters:
        R0 - Completion Status Code
 Necessary Symbols:
        $PRVDEF                         ; Define privilege values

--
        .ENTRY  USER_GET_SID,^M<R2,R3,R4>
        MOVL    G^SCH$GL_CURPCB,R4      ;Get address of PCB
        IFNPRIV CMKRNL,ERROR,PCBREG=R4
        MOVL    4(AP),R1                ; Get address to store time of day register
        IFNOWRT #4,(R1),10$             ; Branch if not writable
        MFPR    #PR$_SID,(R1)           ; Return current time of day register
        MOVL    #SS$_NORMAL,R0          ; Set normal completion status
        RET                             ;  and return

0$:     MOVZWL  #SS$_ACCVIO,R0          ; Indicate access violation
        RET                             ;

RROR:   MOVZWL  #SS$_NOPRIV,R0          ; Indicate no privilege
        RET

        .END
```

(LAB4TEST)

```
        .TITLE  USSTEST
 Facility: Example of User Written System Services
++
 Abstract:
        This module contains an example of a program that invokes a sample
        user-written system service that is contained in a privilesed
        shareable library.  The module USSDISP contains the sample service
        and associated dispatching code being invoked by this simple test
        program.  For lab purposes, the module USSDISP is named LAB4.MAR.
--
 Link Command File:  to be used when linking the test image with USS

        $ !
        $ !     Link Command file for USSTEST
        $ !
        $ LINK USSTEST/MAP/FULL,SYS$INPUT/OPTIONS
        !
        !       Options file for USSTEST
                USS.EXE/SHARE

--
        .PSECT  DATA,NOEXE
UF:     .LONG   0                               ; Location to receive SID contents
UF_DESC:                                        ; Descriptor for conversion
        .WORD   8
        .BYTE   14,1                            ; Type=test  class=scalar
        .ADDRESS        BUF_ASCII
UF_ASCII:                                       ; Location for converted values
        .BLKB   8

        .PAGE
        .SBTTL  Sample invocation of user written system service
++
 Functional Description:
        This routine shows an invocation of the example user system service that
        will read the contents of the system identification register.

        As can be seen by this example, the privileged nature of the code used
        to implement the readings of the SID is not visible to the caller.
        For coding convenience and better maintainability, the code can be
        generated by macros patterned on the standard VMS system service macros.

--
        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  USSTEST,^M<>                    ; Entry mask and definition
        PUSHAL  BUF                             ; Location to receive id value
        CALLS   #1,G^USER_GET_SID               ; Invoke routine in privileged library
                                                ;  to get value from id register
        BLBS    R0,10$                          ; Check status from return value
        PUSHL   R0                              ; Put error code on stack
        CALLS   #1,G^LIB$STOP                   ; and display it
        RET                                     ; And exit
10$:    PUSHAL  BUF_DESC                        ; Descriptor for conversion value
        PUSHAL  BUF                             ; Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ               ; And convert it
        PUSHAL  BUF_DESC                        ; Descriptor for ASCII value
        CALLS   #1,G^LIB$PUT_OUTPUT             ; And display it
        RET                                     ; Finished

        .END    USSTEST
```

5) Write a privileged program to monitor the size of non-paged "pool". It should run every 5 seconds, and broadcast a message to all users to logout if the size of pool is less than 300 * MAXPROCESSCNT.

MEDIUM

Solution: [INTERNALS.WEDNESDAY]POOLTEMP.MAR (template)
[INTERNALS.WEDNESDAY]POOLMON.MAR (complete)
[INTERNALS.WEDNESDAY]POOLCTRL.MAR (runs the monitor)

6) Modify Lab Problem #2 to get the PCB address for any process on the system (given it's PID).

DIFFICULT

Solution: [INTERNALS.WEDNESDAY]GETPCBTMP.MAR (template)
[INTERNALS.WEDNESDAY]GETPCB.MAR (complete)

7) Modify Lab Problem #3 to output the PCB and JIB contents for any process on the system (given it's PID).

DIFFICULT

Solution: [INTERNALS.WEDNESDAY]DMPPCBTMP.MAR (template)
[INTERNALS.WEDNESDAY]DUMPPCB.MAR (complete)

```
;*****************************************************************************
;
;       Program name:   Poolmonitor (template)
;
;       Author:         Phil Lovecchio
;                       Senior Educational Specialist
;                       Digital Equipment Corporation
;                       Rolling Meadows Training Facility
;
;       Function:       Add up the size of the "pieces" of variable
;                       and send out a message if size is not suffic
;
;       Privileges:     CMKRNL
;
;       Input:          None
;
;       Output:         Message (sometimes), and the size of pool
;
;       Other:          Controlled by process created by LAB4CTRL
;
;       Side effects:   None
;
;*****************************************************************************
;
        .TITLE POOLTEMP             Non-paged pool monitor
;+      This program will calculate the size of non-paged dynamic
;       memory and decide if the value is sufficient to allow
;       continuation of user processes.
;-
        .PSECT  DATA,NOEXE
;
;External symbols
;
        $BRDCSTDEF
MIN_POOL:
        .LONG   0                               ;Storage for pool calculatic
MESSAGE:
        .ASCID /Pool is low--please log out !/  ;Warning message
POOL_SIZE:
        .LONG   0                               ;Storage for sum
POOL_OUTPUT:
        .WORD   POOL_STRING_LEN + 8             ;Size = header + 8
        .BYTE   14,1                            ;Type=text Class=scalar
        .ADDRESS        POOL_STRING
POOL_STRING:
        .ASCII  /Size of pool is: /             ;Header for pool size messa
POOL_STRING_LEN=.-POOL_STRING
POOL_ASCII:
        .BLKB   8                               ;ASCII conversion storage
POOL_CNV_DESC:
        .WORD   8                               ;Size = 8 characters max
        .BYTE   14,1                            ;Type=text Class=scalar
        .ADDRESS        POOL_ASCII
```

```
;
;       Code
;
        .PSECT  CODE,NOWRT,EXE,SHR
        .ENTRY  POOLTEMP,       ^M<>    ;Null entry mask
;
;Change mode to kernel to set at pool values
;
        CLRL    POOL_SIZE               ;Get location into memory
        CLRL    MIN_POOL                ;Get location into memory
        $CMKRNL_S       ROUTIN=10$      ;Get into kernel mode
        PUSHAL  POOL_CNV_DESC           ;Descriptor for conversion
        PUSHAL  POOL_SIZE               ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TI       ;And convert it
        PUSHAL  POOL_OUTPUT             ;Descriptor for output
        CALLS   #1,G^LIB$PUT_OUTPUT     ;And display it
        MULL2   #300,MIN_POOL           ;Minimum size of pool
;
;       NOTE:   Change the above value for testing purposes
;
        CMPL    POOL_SIZE,MIN_POOL      ;See if they are equal
        BLSS    5$                      ;Branch if pool is too low
        $HIBER_S                        ;Hibernate till next wake
        RET
5$:     $BRDCST_S       MSGBUF=MESSAGE  ;Message to all users
        $HIBER_S                        ;Hibernate till next wake
        RET
10$:    .WORD   ^M<R8>                  ;Kernel routine
                                        ;MAXPROCESSCNT
                                        ;IPL necessary to view pool
                                        ;Raise IPL, and get next adc
                                        ;First piece of pool
                                        ;Size of piece and add it
                                        ;See if value in R8 is 0
                                        ;Find next piece if not eq C
                                        ;Enable ints (IPL=0)
        RET

        .END    POOLTEMP
```

```
        .TITLE  POOLCONTROL     Controls pool monitor from POOLMONIT
;
;External symbols
;
        $PRCDEF
        $PCBDEF
;
;+      Program to control the pool monitor.  A subprocess is created
;       (change to detached later) to run the image every 5 secs,
;       comparing the value of variable pool to a calculated value.
;       If the size of pool is too low, a message will be displayed
;       to all users.
;-
        .PSECT  DATA,NOEXE
IMAGE:  .ASCID  /POOLMON.EXE/                           ;Pool monitor image
NAME:   .ASCID  /Pool_Monitor/                          ;Name of subprocess
BASEPRI:
        .LONG   4                                       ;Subprocess basepri
STATUS: .LONG   PRC$M_HIBER                             ;Hibernate flag
PID:    .LONG   0                                       ;Pid of subprocess
;
;Arguments for scheduled wakeup
;
DELTA:  .ASCID  /0 00:00:05.00/                         ;Scheduling interval
TIME:   .BLKQ   1                                       ;Conversion storage
MESSAGE:
        .ASCID  /Pool monitor is running/               ;Success message
;
;       Code
;
        .PSECT  CODE,NOWRT,EXE,SHR
        .ENTRY  POOLMONCTRL,    ^M<>
        $BINTIM_S       TIMBUF=DELTA, -
                        TIMADR=TIME                     ;Convert offset time
        BLBC            R0,10$                          ;Check status
        $CREPRC_S       PIDADR=PID,IMAGE=IMAGE,-
                        PRCNAM=NAME,BASPRI=#BASEPRI,-
                        STSFLG=STATUS                   ;Create the monitor
        BLBC            R0,10$                          ;Check status
        $SCHDWK_S       PIDADR=PID,DAYTIM=TIME,-
                        REPTIM=TIME                     ;Schedule wakeups
        BLBC            R0,10$                          ;Check status
        PUSHAL          MESSAGE
        CALLS           #1,G^LIB$PUT_OUTPUT             ;Display success
        BLBC            R0,10$                          ;Check status
        RET

10$:
        PUSHL           R0                              ;Get on stack
        CALLS           #1,G^LIB$STOP                   ;Bomb with messages

        .END    POOLMONCTRL
$
```

```
        .TITLE  GETPCBTMP                       Use system values to set PCB

;       External symbols
;
        $PCBDEF
        $IODEF

        .PSECT  DATA,NOEXE
TERM:   .ASCID  /SYS$OUTPUT/                    ;Output for messages
CHAN:   .WORD   0                               ;Channel storage
PID_DESC:
        .WORD   8                               ;Descriptor for conversion
        .BYTE   14,1
        .ADDRESS        PID_INPUT
PID_INPUT:
        .BLKB   8                               ;Storage for PID input
PID_VAL:
        .LONG   0                               ;Storage for conversion
PROMPT:
        .ASCII  /Enter PID of process: /

PROMPT_LEN=.-PROMPT
;*************************************************************************
;Values for PCB conversions
;
PCB_VAL:
        .LONG   0
PCB_OUTPUT:
        .WORD   STRING_LEN + 8
        .BYTE   14,1
        .ADDRESS        STRING
STRING:
        .ASCII  /Address of PCB is: /
STRING_LEN=.-STRING
PCB_ASCII:
        .BLKB   8
PCB_CNV_DESC:
        .WORD   8
        .BYTE   14,1
        .ADDRESS        PCB_ASCII
ISTAT:  .BLKQ   1
ARGS:   .LONG   1
        .ADDRESS        PID_VAL
```

```
;       Code
;
        .PSECT  CODE,NOWRT,SHR,EXE
        .ENTRY  GETPCBTMP,      ^M<>

        $ASSIGN_S       DEVNAM=TERM, -
                        CHAN=CHAN                       ;Get I/O channel
        $QIOW_S         CHAN=CHAN,    -
                        FUNC=#IO$_READPROMPT, -
                        IOSB=ISTAT,   -
                        P1=PID_INPUT,P2=#8,   -
                        P5=#PROMPT,P6=#PROMPT_LEN
        BLBS    R0,1$
        PUSHL   R0
        CALLS   #1,G^LIB$STOP
        RET
1$:
        MOVW    ISTAT+2,PID_DESC
        PUSHAL  PID_VAL
        PUSHAL  PID_DESC
        CALLS   #2,G^OTS$CVT_TZ_L
        BLBS    R0,2$
        PUSHL   R0
        CALLS   #1,G^LIB$STOP
        RET
2$:
        CLRL    PCB_VAL
        $CMKRNL_S       ROUTIN=10$, ARGLST=ARGS

        BLBS    R0,5$
        PUSHL   R0
        CALLS   #1,G^LIB$STOP
        RET
5$:     PUSHAL  PCB_CNV_DESC
        PUSHAL  PCB_VAL
        CALLS   #2,G^OTS$CVT_L_TZ
        PUSHAL  PCB_OUTPUT
        CALLS   #1,G^LIB$PUT_OUTPUT
        RET

10$:    .WORD   ^M<R2,R3,R4>
        +-------------------------------------+
        |                                     |
        |                                     |
        |                                     |
        +-------------------------------------+
                                        ;Elevate IPL
                                        ;Get PID from list
                                        ;Get PCB address
                                        ;Move to location
                                        ;Allow interrupts
        RET

        .END    GETPCBTMP
$
```

```
;*****************************************************************************
;
;       Program name:   DMPPCBTMP
;
;       Author:         Phil Lovecchio
;                       Senior Educational Specialist
;                       Digital Equipment Corporation
;                       Rolling Meadows Training Facility
;
;       Function:       List out contents of PCB and JIB for process
;
;       Privileges:     CMKRNL
;
;       Input:          None
;
;       Outputs:        Contents of PCB and JIB
;
;       Side effects:   None as known
;
;*****************************************************************************
;
        .TITLE  DMPPCBTMP                       List contents of PCB and JIB
        .LIBRARY        /MAR$EXTRA:MACROLIB/
        .PSECT  DATA,NOEXE,WRT,NOSHR
;Program to list out the contents of the PCB and JIB.  Note the
;use of conversion and run-time library output routines.
;*****************************************************************************
;
;External symbols
;
        $PCBDEF
        $JIBDEF
        $IODEF
;
;       Storage for PID
;
TERM:   .ASCID  /SYS$OUTPUT/                    ;Output for messages
CHAN:   .WORD   0                               ;Channel storage
PID_DESC:
        .WORD   8                               ;Descriptor for conversion
        .BYTE   14,1
        .ADDRESS        PID_INPUT
PID_INPUT:
        .BLKB   8                               ;Storage for PID input
PID_VAL:
        .LONG   0                               ;Storage for conversion
PROMPT:
        .ASCII  /Enter PID of process: /

PROMPT_LEN=.-PROMPT
;
;Generate values for PCB address
;
PCB_VALUES:                                     ;Storage for PCB values
        .BLKB   PCB$K_LENGTH
PCB_ASC_DESC:                                   ;Descriptor for PCB values
        .WORD   8                               ;Size eight bytes max
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        PCB_ASC_VALUE
PCB_ASC_VALUE:                                  ;Destination for conversion
        .BLKB   8
```

```
;*******************************************************************
JIB_VALUES:                                     ;Storage for JIB values
        .BLKB   JIB$K_LENGTH
JIB_ASC_DESC:                                   ;Descriptor for JIB values
        .WORD   8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        JIB_ASC_VALUE
JIB_ASC_VALUE:                                  ;Destination for conversion
        .BLKB   8
;*******************************************************************
PCB_VAL:                                        ;Binary value of PCB address
        .LONG
JIB_VAL:                                        ;Binary value of JIB address
        .LONG
;*******************************************************************
PCB_OUTPUT:                                     ;Descriptor for output strin
        .WORD   PCB_LEN + 8                     ;Length is header plus 8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        STRING1
STRING1:
        .ASCII  /Address of PCB is: /           ;Header for PCB address outp
PCB_LEN=.-STRING1
PCB_ASCII:                                      ;ASCII PCB address
        .BLKB   8
PCB_CNV_DESC:                                   ;Conversion descriptor
        .WORD   8                               ;Maximum of ten characters
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        PCB_ASCII
;*******************************************************************
;Generate same values for JIB address
;
JIB_OUTPUT:                                     ;Descriptor for JIB output
        .WORD   JIB_LEN + 8                     ;Length is header plus 8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        STRING2
STRING2:                                        ;Header for JIB output
        .ASCII  /Address of JIB is: /
JIB_LEN=.-STRING2
JIB_ASCII:                                      ;ASCII JIB address
        .BLKB   8
JIB_CNV_DESC:                                   ;Conversion descriptor
        .WORD   8
        .BYTE   14,1                            ;Type=text  Class=scalar
        .ADDRESS        JIB_ASCII
STRING3:
        .ASCID  /Contents of PCB/               ;Header for PCB output
STRING4:
        .ASCID  /Contents of JIB/               ;Header for JIB output
ARGS:   .LONG   1                               ;Arg list for kernel mode
        .ADDRESS        PID_VAL                 ;For converted PID
ISTAT:  .BLKQ   1                               ;IOSB for QIO
;*******************************************************************
        .PSECT  CODE,EXE,NOWRT,SHR
        .ENTRY  DMPPCBTMP,      ^M<>
        $ASSIGN_S       DEVNAM=TERM, -
                        CHAN=CHAN                       ;Get I/O channel
        CHECK_STATUS
        $QIOW_S         CHAN=CHAN,    -                 ;Prompt for PID
                        FUNC=#IO$_READPROMPT, -
                        IOSB=ISTAT,   -
                        P1=PID_INPUT,P2=#8,   -
                        P5=#PROMPT,P6=#PROMPT_LEN
```

```
        CHECK_STATUS
        MOVW    ISTAT+2,PID_DESC                ;Exact length PID
        PUSHAL  PID_VAL
        PUSHAL  PID_DESC
        CALLS   #2,G^OTS$CVT_TZ_L               ;Convert to binary
        CHECK_STATUS
        CLRL    PCB_VAL
        CLRL    JIB_VAL
        $CMKRNL_S       ROUTIN=10$,-
                        ARGLST=ARGS             ;Change mode with args

        PUSHAL  PCB_CNV_DESC                    ;Descriptor for conversion
        PUSHAL  PCB_VAL                         ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert hex to ASCII
        DISPLAY PCB_OUTPUT                      ;Output for PCB address
;
;Use autoincrement to convert and display PCB values
;
        MOVL    #PCB$K_LENGTH/4,R3              ;Counter for displays
        MOVAL   PCB_VALUES,R2                   ;PCB value address
        DISPLAY STRING3                         ;Descriptor of value
5$:     PUSHAL  PCB_ASC_DESC                    ;Descriptor for value
        PUSHAL  (R2)+                           ;Get next value
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert to ASCII
        DISPLAY PCB_ASC_DESC
        SOBGTR  R3,5$                           ;Done ? ; if so, now JIB
;
;Now do the same for the JIB
;
        PUSHAL  JIB_CNV_DESC                    ;Descriptor for conversion
        PUSHAL  JIB_VAL                         ;Value to be converted
        CALLS   #2,G^OTS$CVT_L_TZ               ;Convert hex to ASCII
        DISPLAY JIB_OUTPUT
;
;Use autoincrement to convert and display JIB values
;
        MOVL    #JIB$K_LENGTH/4,R3              ;Same as for PCB values
        MOVAL   JIB_VALUES,R2
        DISPLAY STRING4
6$:     PUSHAL  JIB_ASC_DESC
        PUSHAL  (R2)+
        CALLS   #2,G^OTS$CVT_L_TZ
        DISPLAY JIB_ASC_DESC
        SOBGTR  R3,6$
        RET                                     ;All done

10$:    .WORD   0                               ;Null entry mask
        |                                       ;IPL to synch
        |                                       ;Get PID value
        |                                       ;Use PID to set PCB
        |                                       ;Move into location
        |                                       ;Get JIB address
;       |
;Use sizes of PCB and JIB to set contents
;       |
        |
        MOVC3                                   ;PCB values
        MOVC3                                   ;JIB values
        |                                       ;Set IPL down again
        MOVL    #SS$_NORMAL,R0                  ;Normal status
        RET                                     ;Back to main code

        .END    NMPRCPTMP
```

(8) MWAIT is a mysterious state. Write an MWHAT program to display the reason that a process has gone into MWAIT state.

MEDIUM

Solutions: [INTERNALS.THURSDAY]MWHATTEMP.MAR (template)
[INTERNALS.THURSDAY]MWHAT.MAR (complete)

Program that "causes" MWAIT for mailbox writes is in [INTERNALS.THURSDAY]:

MAILS.FOR

(9) Write a privileged program to delete a process that is in MWAIT state (if possible). Also, have the program check to make sure that the process is REALLY in MWAIT.

MEDIUM

Solutions: [INTERNALS.THURSDAY]MWAITTEMP.MAR (template)
[INTERNALS.THURSDAY]MWAITKILL.MAR (complete)

(You may wish to run MAILS, then MWHAT, then MWAITKILL)

(10) Write a privileged program to get a PCB address using the system subroutine EXE$NAMPID. You will find the routine in the module SYSPCTRL in your listings. Note the required arguments. The difference in this lab is that there is no template. Try writing this code from scratch.

DIFFICULT

Solutions: [INTERNALS.THURSDAY]PCBTEMP2.MAR (template)
[INTERNALS.THURSDAY]GETPCB2.MAR (complete)

(11) Write a privileged program to dump out the contents of your working set list entries (WSLEs); and also to dump out the actual PFNs that are currently in use by your process.

NOTE: Due to the complexity of this exercise, you may wish to just examine the code and RUN the program).

VERY DIFFICULT

Solution: [INTERNALS.THURSDAY]WSLIST.MAR

```
;*****************************************************************
;
;       Program name:     mwhattemp       12/15/82  V 1.0
;
;       Author:           Bruce Ellis
;                         Principal Educational Specialist
;                         Digital Equipment Corporation
;                         Santa Clara Training Facility
;
;       Function:         Determine what MWAIT state a process is in.
;
;
;       Privieges required:      CMKRNL
;
;       Inputs:           PID
;
;       Outputs:          None
;
;       Side effects:     None known of currently.
;                         (If you find any let me know!)
;
;       Modifications:    Phil Lovecchio          2/14/83
;                                                 4/18/83
;                         Added missing error messages
;                         Put proper labeling for movement to
;                         error routine (if process is not in MWAIT)
;
;*****************************************************************
;
        .title  mwhattemp
;*****************************************************************
;       Macro Definitions
;*****************************************************************
        .macro  check_status    ?lab
        blbs    r0,lab                  ;Error?
        pushl   r0                      ;Yes, then return error status
        calls   #1,g^lib$signal         ; and abort if severe error
lab:                                    ;otherwise carry on
        .endm   check_status
;*****************************************************************
;       Define System symbols
;*****************************************************************
        $dscdef                         ;Descriptor
        $pcbdef                         ;Process Control Block (software)
        $rsndef
;*****************************************************************
;       Define symbols
;*****************************************************************
max_PID_length=8
status_loc=4
;*****************************************************************
;       Non-shareable data
;*****************************************************************
        .psect  Scratch pic,noexe,wrt
error_not_mwait:
        .ascid  /The process is not in MWAIT; or is not in a fixable MWAIT/
prompt_for_pid:                         ;Prompt to user terminal
        .ascid  /Please enter PID: /
```

```
PID_desc:
        .word   max_PID_length          ;Ascii string descriptor used to
        .byte   dsc$k_dtype_t,dsc$k_class_s     ;store the Process Id
        .address        PID_characters
PID_characters:                         ;Characters used to make up the PID
        .blkb   max_PID_length
length_of_PID:                          ;Longword to hold actual length for
        .blkl   1                       ; redefinition of descriptor for PID
PID:    .blkl   1                       ;Binary representation of PID
mwhat_args:
        .long   1
        .address        mwhat_status
mwhat_status:   .blkl   1
;*****************************************************************
;       Assume that resource wait symbols are valid.
;*****************************************************************
        ASSUME  RSN$_ASTWAIT    EQ      1
        ASSUME  RSN$_MAILBOX    EQ      2
        ASSUME  RSN$_NPDYNMEM   EQ      3
        ASSUME  RSN$_PGFILE     EQ      4
        ASSUME  RSN$_PGDYNMEM   EQ      5
        ASSUME  RSN$_BRKTHRU    EQ      6
        ASSUME  RSN$_IACLOCK    EQ      7
        ASSUME  RSN$_JQUOTA     EQ      8
;*****************************************************************
;       Set up pointers to messages describing the mwait state.
;*****************************************************************
wh_t_message:
        .address        mwait_mutex
        .address        mwait_astwait
        .address        mwait_mailbox
        .address        mwait_npdynmem
        .address        mwait_pgfile
        .address        mwait_pgdynmem
        .address        mwait_brkthru
        .address        mwait_iaclock
        .address        mwait_jquota
mwait_mutex:
        .ascid  /The process is waiting on a mutex./
mwait_astwait:
        .ascid  /The process is in AST wait. (Waiting for system or/-
                / special kernal AST)/
mwait_mailbox:
        .ascid  /The process is waiting because mailbox is full./
mwait_npdynmem:
        .ascid  /The process is waiting because of Non-paged Dynamic Memory./
mwait_pgfile:
        .ascid  /The process is waiting because Page File is Full./
mwait_pgdynmem:
        .ascid  /The process is waiting because of Paged Dynamic Memory./
mwait_brkthru:
        .ascid  /The process is waiting because of Breakthrough. /-
                /(wait for broadcast)/
mwait_iaclock:
        .ascid  /The process is waiting because of Image Activation Lock./
mwait_jquota:
        .ascid  /The process is waiting because Job Pooled Quota./
```

```
;*****************************************************************
;       User mode code
;*****************************************************************
        .psect  mwhattemp_code  pic,exe,shr,nowrt
        .entry  mwhattemp,      ^M<>
        clrl    mwhat_status                ;Assume it is waiting on a mutex.
        pushal  length_of_PID               ;Prompt user for
        pushal  prompt_for_pid              ;       the process ID
        pushal  PID_desc                    ;and read it into an ascii descriptor
        calls   #3,g^lib$get_input          ;
        check_status                        ;Check to make sure that it was received
        movw    length_of_PID,PID_desc      ;Update the length in the descriptor
        pushal  PID                         ;Load the binary Process Id
        pushal  PID_desc                    ; as converted from the ascii
        calls   #2,g^ots$cvt_tz_l           ; descriptor repesentation
        check_status                        ;Conversin O. K. ?
        $cmkrnl_s       routin=take_care_of_pcb,arglst=mwhat_args
        check_status                                ;wait mode
        tstl    mwhat_status                ;Is the process really waiting on
        blss    not_in_mwait                ; a mutex?
;*****************************************************************
;*******       If not mutex wait state, then find out what state it is in.
;*****************************************************************
        movl    mwhat_status,r6             ;r6  is index into message
        cmpl    #rsn$_jquota,r6             ;Mwhat type out of range?
        blss    not_in_mwait                ;The process is not in mwait state
is_mutex:               ;***    temporary until code added for mutex type
        movl    what_message[r6],r6         ;r6 points to proper messsage text.
        pushl   r6                          ;Now display which mwait state it is
        calls   #1,g^lib$put_output         ; in.
exit:   ret                                 ;All done.
not_in_mwait:
        pushal  error_not_mwait             ;Display error message
        calls   #1,g^lib$put_output         ;
        brb     exit
;       See ya later alligator!
```

```
;*****************************************************************************
;       Kernel mode code
;*****************************************************************************
;
;       Recommended corrections to sloppiness:
;       1)      Make sure ipl$_synch code in Physical Memory
;       2)      Check PID sequence number for validity
;       Recommended polish : add code for process name.
;

        .entry  take_care_of_pcb,^m<r3> ;$cmkrnl should save r4 for us.
.enable lsb

        movl    4(ap),r3                ;Address of arg list for MWAIT reason
        movl    #ss$_normal,r0          ;Assume all is well
        extzv   #0,#16,8(r3),r1         ;Get PCB vector index from PID
        setipl  #8                      ;Block access to scheduler's data
        cmpl    r1,@#sch$gl_maxpix      ;PID index too large?
        bgtru   non_existant_process    ; Yes, return error.
        movl    @#sch$gl_pcbvec,r2      ;Get base of PCB vector.
        movl    (r2)[r1],r2             ;Get PCB address, using PID index.
        beql    non_existant_process    ;Is it the Null process?
        cmpl    pcb$l_pid(r2),8(r3)     ; Yes, return error.
        beql    non_existant_process    ;Is it Swapper process?
        movl    pcb$l_efwm(r2),r2       ; Yes, return error
        beql    non_existant_process    ;Got EFWM which has MWAIT reason
        movl    4(r2),r2                ;
10$:    setipl  #0                      ;Set ipl back to normal
        ret                             ;Back to user mode
non_existant_process:
        movl    #ss$_nonexpr,r0         ;Non-existent process
        brb     10$                     ;Return error status
        .end    mwhattemp
```

```
;*****************************************************************
;
;       Program name:    mwaittemp      12/15/82  V 1.0
;
;       Author:          Bruce Ellis
;                        Principal Educational Specialist
;                        Digital Equipment Corporation
;                        Santa Clara Training Facility
;
;       Function:        Delete a process in MWAIT state
;                        (When possible!)
;
;       Privieges required:     CMKRNL
;
;       Inputs:          PID
;
;       Outputs:         None
;
;       Side effects:    None known of currently.
;                        (If you find any let me know!)
;
;       Modified:        Phil Lovecchio   June 2, 1983
;                        Added $STATEDEF definitions to check
;                        and see if process is REALLY in MWAIT
;                        before killing it
;
;*****************************************************************
;
        .title  mwaittemp
;*****************************************************************
;       Macro Definitions
;*****************************************************************
        .macro  check_status    ?lab
        blbs    r0,lab                  ;Error?
        pushl   r0                      ;Yes, then return error status
        calls   #1,s^lib$signal         ; and abort if severe error
lab:                                    ;otherwise carry on
        .endm   check_status
;*****************************************************************
;       Define System symbols
;*****************************************************************
        $dscdef                         ;Descriptor
        $pcbdef                         ;Process Control Block (software)
        $statedef                       ;Scheduler state definitions
;*****************************************************************
;       Define System symbols
;*****************************************************************
max_PID_length=8
;*****************************************************************
;       Non-shareable data
;*****************************************************************
        .psect  Scratch pic,noexe,wrt
error_code:
        .ascid  /The process is NOT in MWAIT state !!/
prompt_for_pid:                         ;Prompt to user terminal
        .ascid  /Please enter PID: /
PID_desc:
        .word   max_PID_length          ;Ascii string descriptor used to
        .byte   dsc$k_dtype_t,dsc$k_class_s     ;store the Process Id
        .address        PID_characters
```

```
PID_characters:                         ;Characters used to make up the PID
        .blkb   max_PID_length
length_of_PID:                          ;Longword to hold actual length for
        .blkl   1                       ; redefinition of descriptor for PID
PID:    .blkl   1                       ;Binary representation of PID
;*****************************************************************
;       User mode code
;*****************************************************************
        .psect  mnowait_code    pic,exe,shr,nowrt
        .entry  mwaittemp,^m<>
        pushal  length_of_PID           ;Prompt user for
        pushal  prompt_for_pid          ;       the process ID
        pushal  PID_desc                ;and read it into an ascii descriptor
        calls   #3,g^lib$get_input      ;
        check_status                    ;Check to make sure that it was received
        movw    length_of_PID,PID_desc  ;Update the length in the descriptor
        pushal  PID                     ;Load the binary Process Id
        pushal  PID_desc                ; as converted from the ascii
        calls   #2,g^ots$cvt_tz_l       ; descriptor repesentation
        check_status                    ;Conversin O. K. ?
        $cmkrnl_s       routin=take_care_of_pcb ;Reset bit to disable Resource
        check_status                            ;wait mode
        cmpw    #50,r8                  ;See if 'error' code is in r8
        beql    no_mwait
        $delprc_s       pidadr=PID      ;Deliver an AST to rerun the MWAIT code
        check_status                    ;should abort MWAIT and let delete AST
        ret
no_mwait:
        pushal  error_code              ;Error message
        calls   #1,g^lib$put_output     ;And display it
        ret                             ;be delivered to process (Bye Bye)
```

```
;*****************************************************************************
;       Kernal mode code
;*****************************************************************************
;
;       Recommended corrections to sloppiness:
;       1)      Make sure ipl$_synch code in Physical Memory
;       2)      Check PID sequence number for validity
;       Recommended polish : add code for process name.
;

        .entry  take_care_of_pcb,^m<>   ;$cmkrnl should save r4 for us.
.enable lsb
                                        ;Assume all is well
                                        ;Get PCB vector index from PID
                                        ;Synchronize with the scheduler Data Base
                                        ;PID index too large?
                                        ; Yes, return error.
                                        ;Get base of PCB vector.
                                        ;Get PCB address, using PID index.
                                        ;Is it the Null process?
                                        ; Yes, return error.
                                        ;Is it Swapper process?
                                        ; Yes, return error
                                        ;See if process is in MWAIT
                                        ;To error routine
                                        ;Got PCB set NO
                                        ;resource wait mode
10$:    setipl  #0                      ;Set ipl back to normal
        ret                             ;Back to user mode
non_existant_process:
        movl    #ss$_nonexpr,r0         ;Non-existent process
        brb     10$                     ;Return error status
not_in_mwait:
        movl    #50,r8                  ;Dummy error code (Check in main code)
        brb     10$                     ;Return with "error" status
        .end    mwaittemp
```

```
        .TITLE  WSLIST
;
;       Program to list out WS entries and valid PFNs for
;       a process

        $PCBDEF
        $PHDDEF
        $PTEDEF
        $WSLDEF
        .PSECT  DATA    PIC,NOEXE,LONG
CONTROL:
        .ASCID  'Wsle !XL SVAPTE !XL PFN !XL PTE !XL!/'
term:   .ASCID  /SYS$OUTPUT/
term_chan:
        .blkw   1
pid_prom:
        .ascii  <10><13>/PID? /
        pid_prom1=.-pid_prom
promptn:
        .blkl   1
promptn1:
        .blkl   1
proc:   .blkb   20
procd:  .blkw   1
        .byte   14,1
        .address        proc
stat:   .blkw   4
out_buf:
        out_bufln=80
        .long   out_bufln
        .address        10$
10$:    .blkl   out_bufln
data_1: .address        db_1
        .address        de_1
db_1:
pid:    .long   0
args:   .long   2
        .long   0
        .long   0
pcnt:   .long   0
wsle:   .blkl   1024
svapte: .blkl   1024
ppfn:   .blkl   1024
de_1:
code_1:
        .address        cb_1
        .address        ce_1
        .macro  check_stat      ?lab
        blbs    r0,lab
        pushl   r0
        calls   #1,g^lib$stop
lab:
        .endm   check_stat
;*****************************************************************
;
;*****************************************************************
        .psect  code    pic,shr,nowrt,long
        .entry  wslist  ,^m<r2,r3,r4,r5,r6>
        $assign_s       devnam=term,chan=term_chan
```

```
        $qiow_s chan=term_chan,func=#io$_readprompt,-
                p1=proc,p2=#20,p5=prompn,p6=prompn1,-
                iosb=stat
        check_stat
        movw    stat+2,procd

        pushal  pid
        pushaq  procd
        calls   #2,g^ots$cvt_tz_l
        $lkwset_s       inadr=data_l
        check_stat
        $lkwset_s       inadr=code_l
        check_stat
        $cmkrnl_s       routin=wsl_pfn
        check_stat
        moval   wsle,r2
        moval   svapte,r3
        moval   ppfn,r4
        clrl    r5
typ_out:
        movab   out_buf,r6
        extzv   #7,#20,(r4),r9
        $fao_s  control,(r6),(r6),(r2),(r3),(r4),r9
        check_stat
        pushaq  (r6)
        calls   #1,g^lib$put_output
        tstl    (r2)+
        tstl    (r3)+
        tstl    (r4)+
        sobgtr  pcnt,typ_out
        ret
;*****************************************************************
;       Kernal mode code
;*****************************************************************

cb_l:
        .entry  wsl_pfn ,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
        setipl  #ipl$_synch
        tstl    pid
        beql    50$
        cmpw    pid,sch$gl_swppid
        beql    30$
40$:    cvtwl   pid,r6
        cmpw    r6,sch$gl_maxpix
        bgtru   65$
        movl    @sch$gl_pcbvec[r6],r4
        cmpl    r4,#sch$gl_nullpcb
        bneq    50$
30$:    movzwl  #ss$_nonexpr,r0
        ret
65$:    movzwl  #ss$_nomoreproc,r0
        ret
50$:    clrl    pcnt
        moval   svapte,r10
        moval   ppfn,r11
        moval   wsle,r9
        movl    pcb$l_phd(r4),r5
        movzwl  phd$w_wslist(r5),r7
        movzwl  phd$w_wslast(r5),r6

map_loop:
        movl    (r5)[r7],r2
        blbc    r2,notvalid
        movl    r2,(r9)+
```

```
        movl    r8,(r10)+
        extzv   #pte$v_pfn,#pte$s_pfn,r8,r0
        movl    r0,(r11)+
        incl    pcnt
notvalid:
        aobleq  r6,r7,map_loop
done:
        setipl  #0
        ret
ce_1:
        .end    wslist
```

SYSTEM MECHANISMS

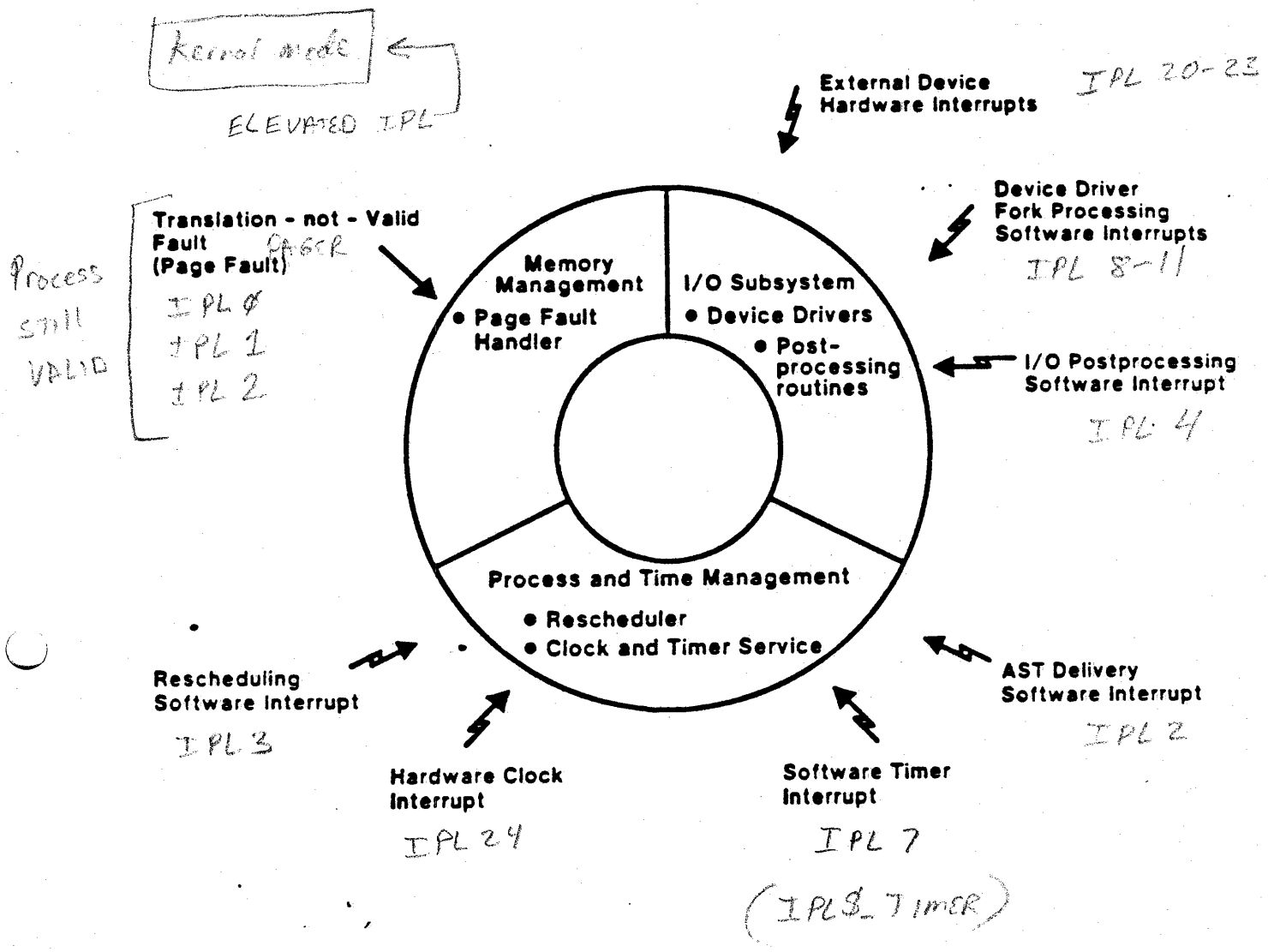
OBJECTIVES

* describe how various VAX/VMS protection, communication, and synchronization mechanisms are implemented, and the reason behind their use
* implement/write/use a user-written system service

READINGS

* IDSM Chapters 2,3,4,5,24,25
* (Optional) IDSM Chapters 6,10
  Architecture Handbook Chapter 12
  Hardware Handbook Chapter 12

ENTRY PATHS INTO THE VMS KERNEL



See $IPLDEF [A-10]

Any type of exception at IPL > 2 causes crash

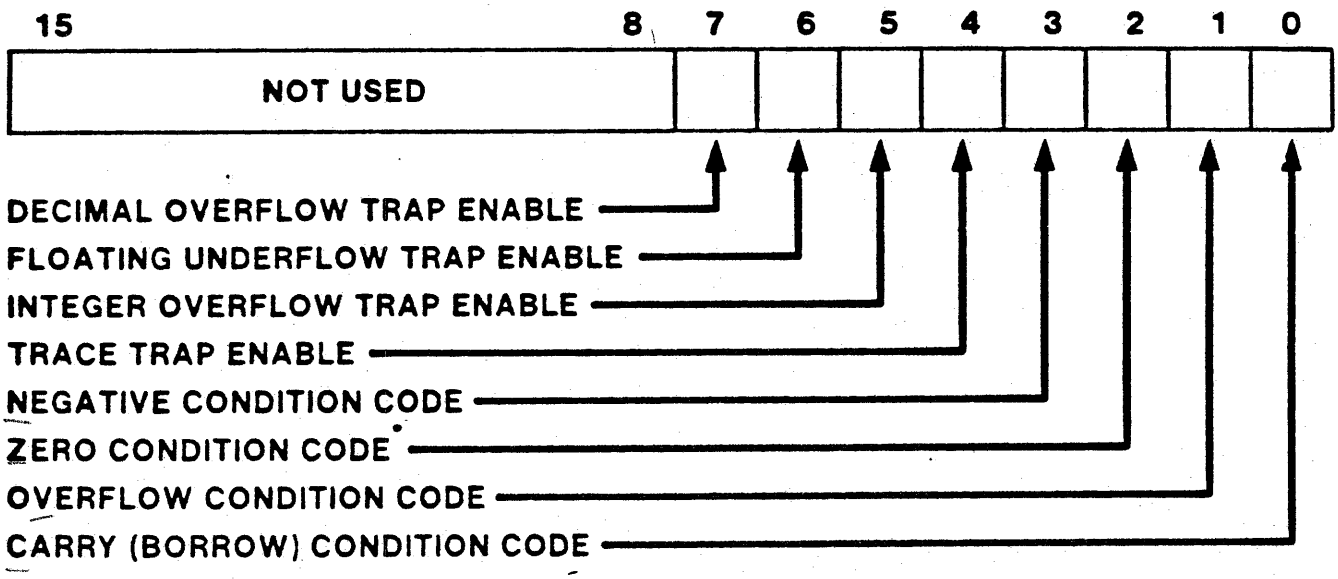
- Arithmetic Trap
- PF
- Call Sys Service

Kernal mode (IPL=0)

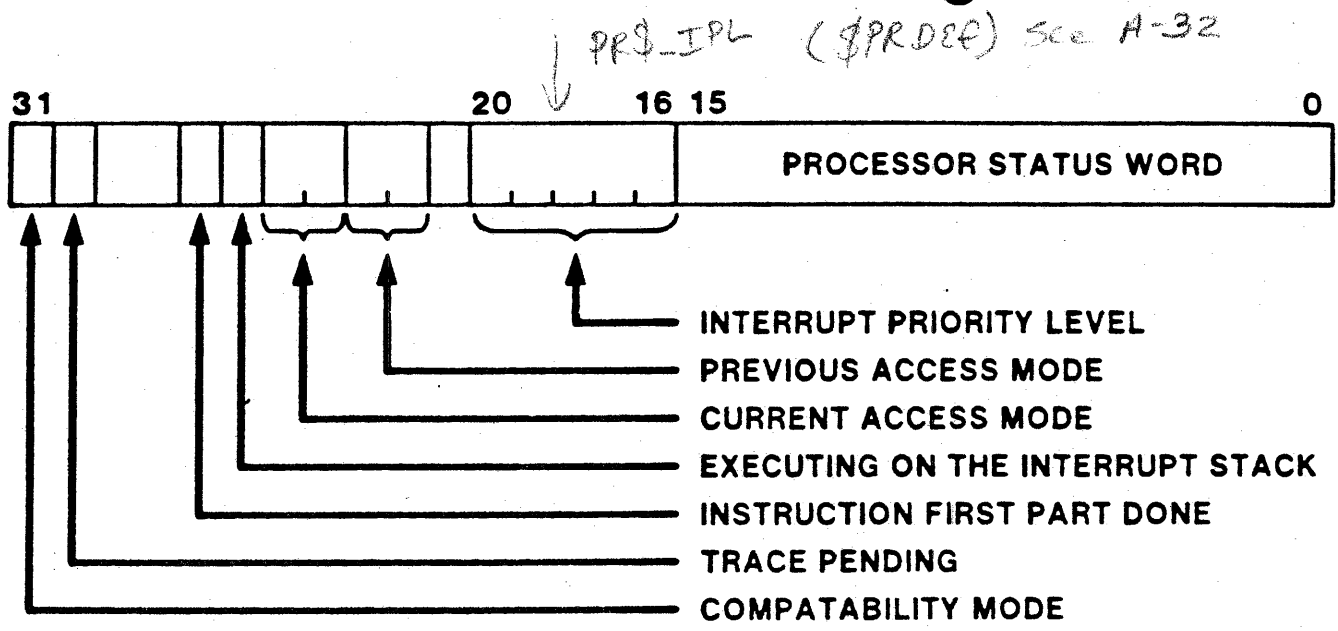
Don't Arithmetic Trap

Exception (Sys Srv Failure or access violation)

Processor Status Word



Processor Status Longword



USE / MTPR (Kernal mode) TO WRITE TO PRS_IPL

MOVPSL

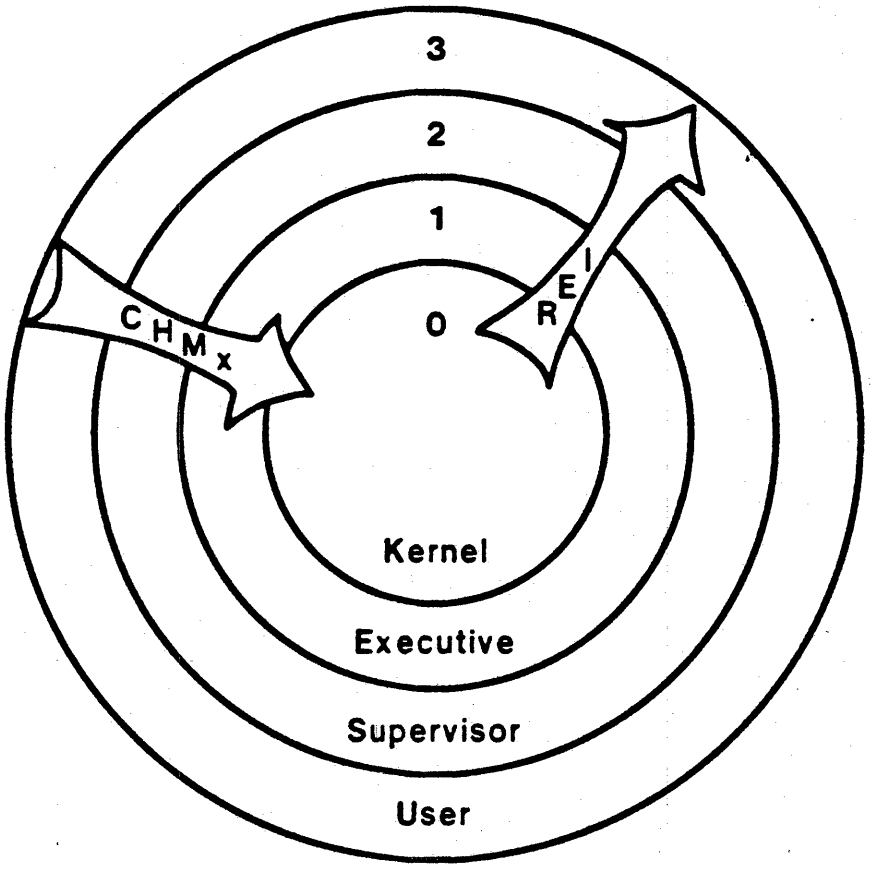
CHMK

all effect IPL

REI return from exception or interrupt

↓ IPL GOES TO Ø

ACCESS MODE TRANSITIONS



REI - • checks for pending AST's
• Checks for Illegal Transitions
• check for pending software interrupt (lower IPL)
• POPS PC, PSL PUR off stack

REI CONTEXTS

① CHMX . . .
:
REI

② Hardware Int ICR
:
REI

③ Software Int SISR
:
REI

④ PUSHL PSL
PUSHL PC
:
REI

(Privileged)
Compat mode

The REI Instruction

The REI instruction will result in a reserved operand fault if any one of the following operations is attempted:

1. Decreasing the access mode value (to a more privileged access mode). (This is a comparison of the current mode fields of both the present PSL and the saved PSL on the stack.)

2. Switching to the interrupt stack from one of the four perprocess stacks.

3. Leaving the processor on the interrupt stack in other than kernel access mode.

4. Leaving the processor on the interrupt stack at IPL Ø.

5. Leaving the processor at elevated IPL (IPL > Ø) and not in kernel access mode.

6. Restoring a PSL in which the previous mode field is more privileged than the current mode field (previous mode < current mode).

7. Raising IPL.

8. Setting any of the following bits -- PSL<29:28> or PSL<21>.

If the processor is being put into compatibility mode, the following checks are made:

1. The first-part-done bit must be clear.

2. The interrupt stack bit must be clear.

3. All three arithmetic trap enables (DV, IV, and FU) must be clear.

4. The current mode field of the saved PSL must be user access mode.

If all the preceding checks are performed without error, the REI microcode continues by:

1. saving the old stack pointer (SP register) in the appropriate processor register (KSP, ESP, SSP, or USP),

2. setting the trace pending bit in the new PSL if the trace pending bit in the old PSL is set,

3. moving the contents of the two temporaries (item 1 above) into the PC and PSL processor registers, and

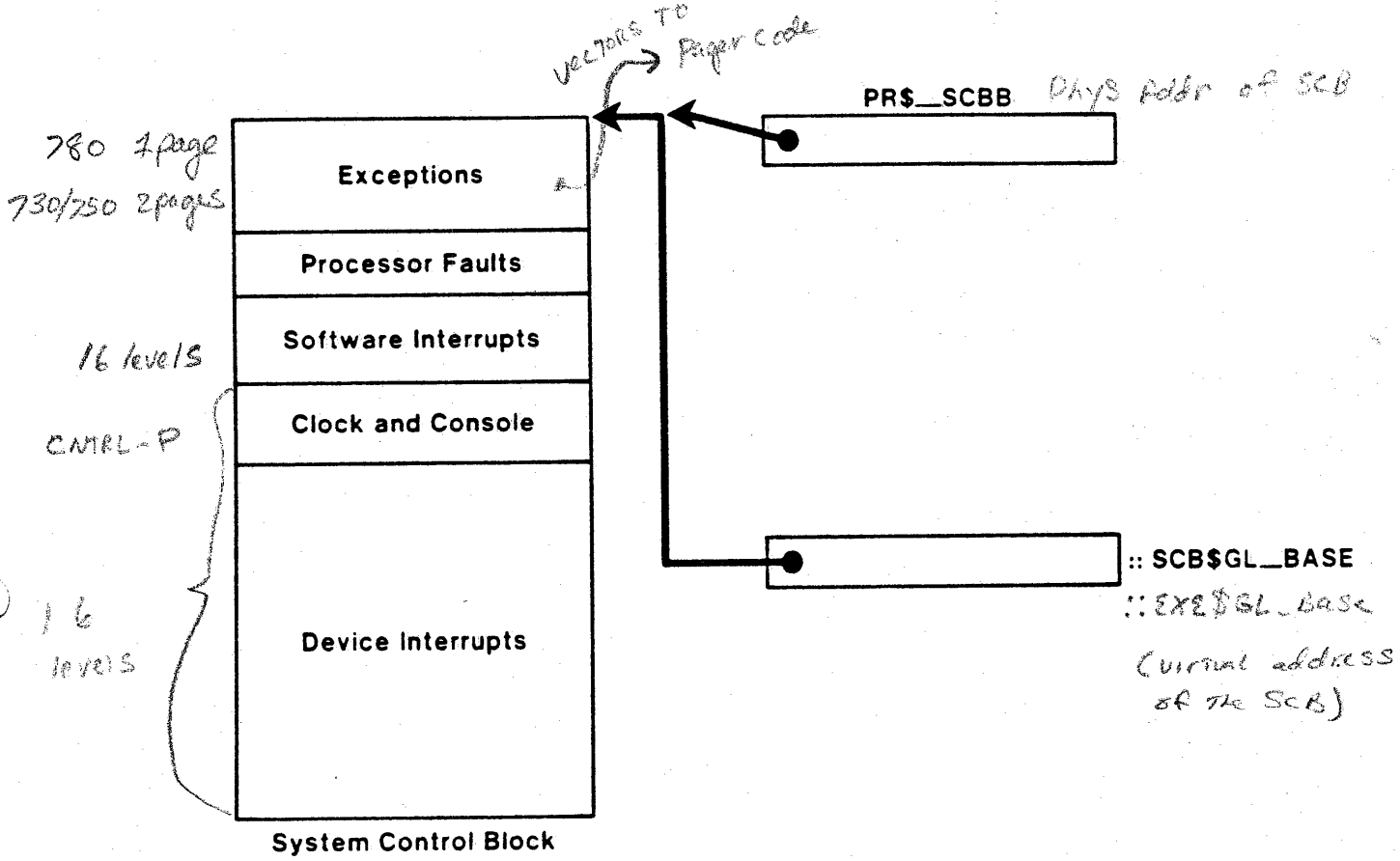
4. if the target stack is a perprocess stack:

    a. getting the new stack pointer from the corresponding processor register (KSP, ESP, SSP, or USP), and

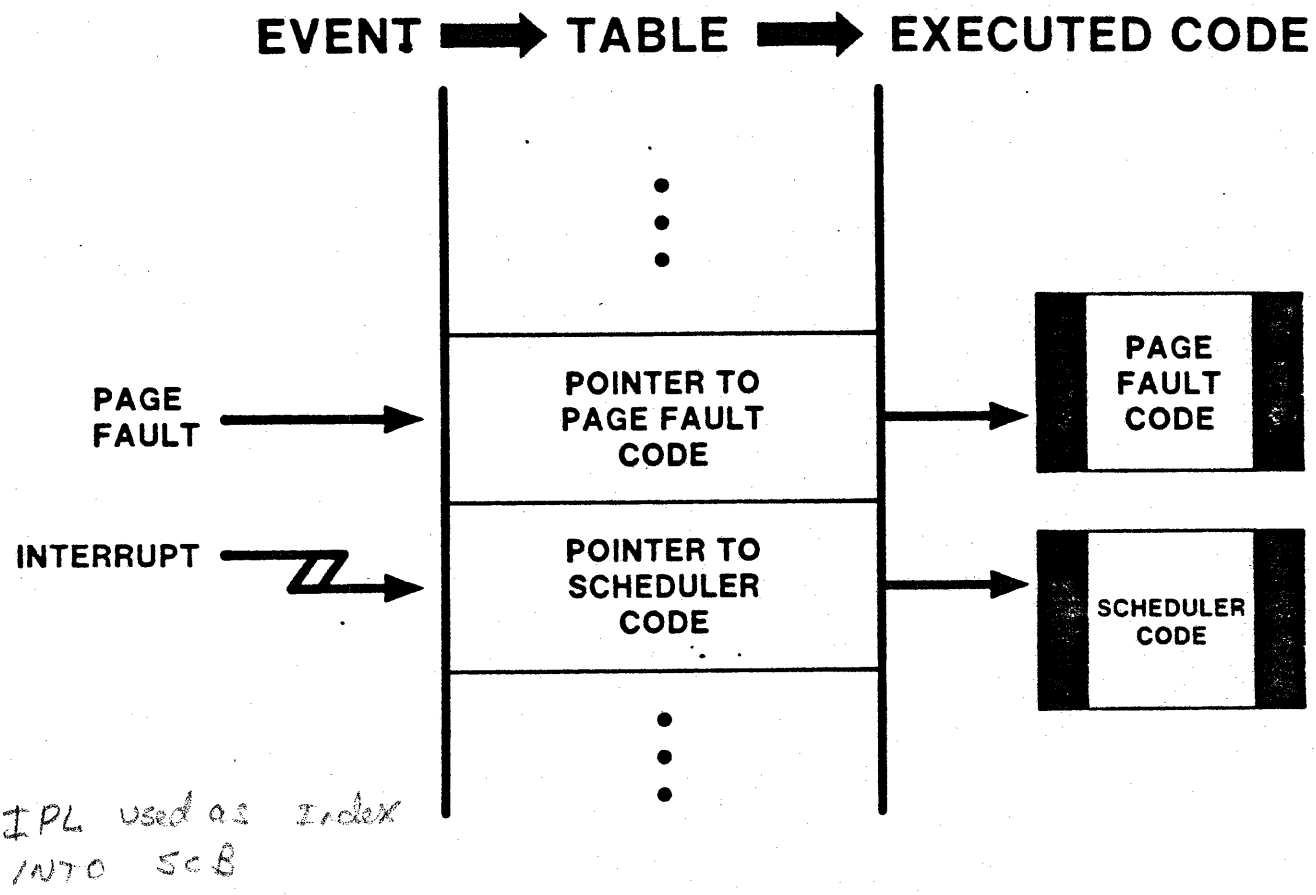
    b. checking for potential deliverability of pending ASTs.

HARDWARE/SOFTWARE INTERRUPTS AND THE SCB

System Control Block



INVOKING SYSTEM CODE



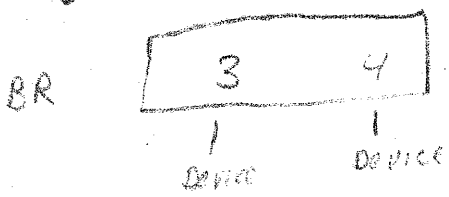
HARDWARE INTERRUPTS AND IPL

Chapter 6 6-7 Chart 6-5
Guide to writing a Device Driver

FUNCTION	VALUE (decimal)	NAME
Block all Interrupts	31	IPL$_POWER
Clock Interrupts	24	IPL$_HWCLK
Device Interrupts	20-23	UCB$B_DIPL*
Driver Fork Levels	8-11	UCB$B_FIPL*

*** Offset into Device's Unit Control Block**

UNIBUS



BR+16 = IPL → UCB$B_DIPL

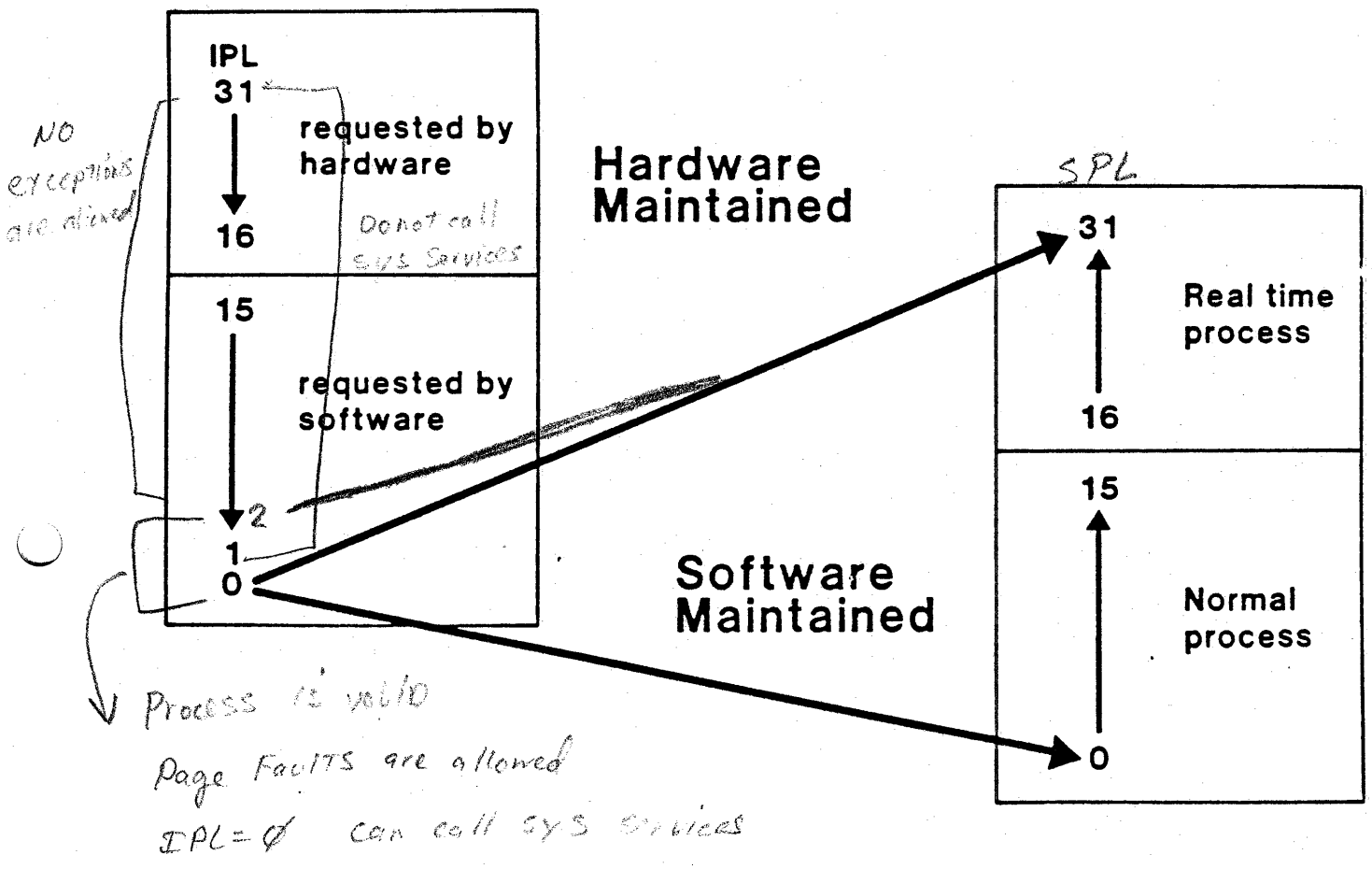
Closer to CPU higher Interrupt

SOFTWARE INTERRUPTS AND IPL

Chapter 7 of IDSM

IPL	USE
15-12	Unused
11	IPL=11 Fork Dispatching
10	(IPL=10 Fork Dispatching)
9	(IPL=9 Fork Dispatching)
8	IPL=8 Fork Dispatching
7	Software Timer Service Routine
6	IPL=6 Fork Dispatching
5	Used to Enter XDELTA
4	I/O Post Processing
3	Rescheduling Interrupt
2	AST Delivery Interrupt
1	Unused
Ø	Process Context

TWO TYPES OF PRIORITY



BLOCKING OTHER INTERRUPTS

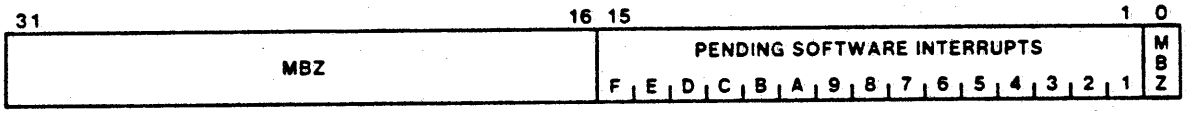
WHAT TO BLOCK	RAISE IPL TO (decimal)	NAME
All Interrupts	31	IPL$_POWER
Clock Interrupts	24	IPL$_HWCLK
Device Interrupts	20-23	UCB$B_DIPL*
Access to Scheduler's Data Structures	7	IPL$_SYNCH IPL$_TIMER
Delivery of ASTs (Prevent Process Deletion)	IPL ≥ 2	IPL$_ASTDEL

*** Offset into Device's Unit Control Block**

IPL$_TIMER - Invokes Software Timer ISR (Interrupt)

Synch - Stop's Scheduler during Timer

PR$_SIRR **Software Interrupt Request Register (Write Only)**

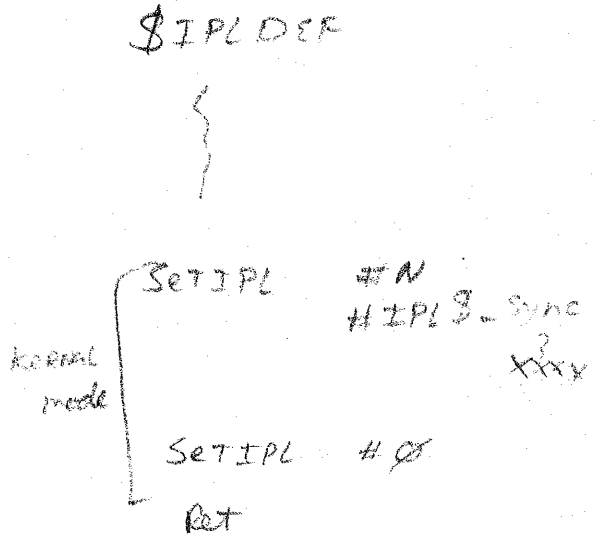


PR$_SISR **Software Interrupt Summary Register (Read/Write)**

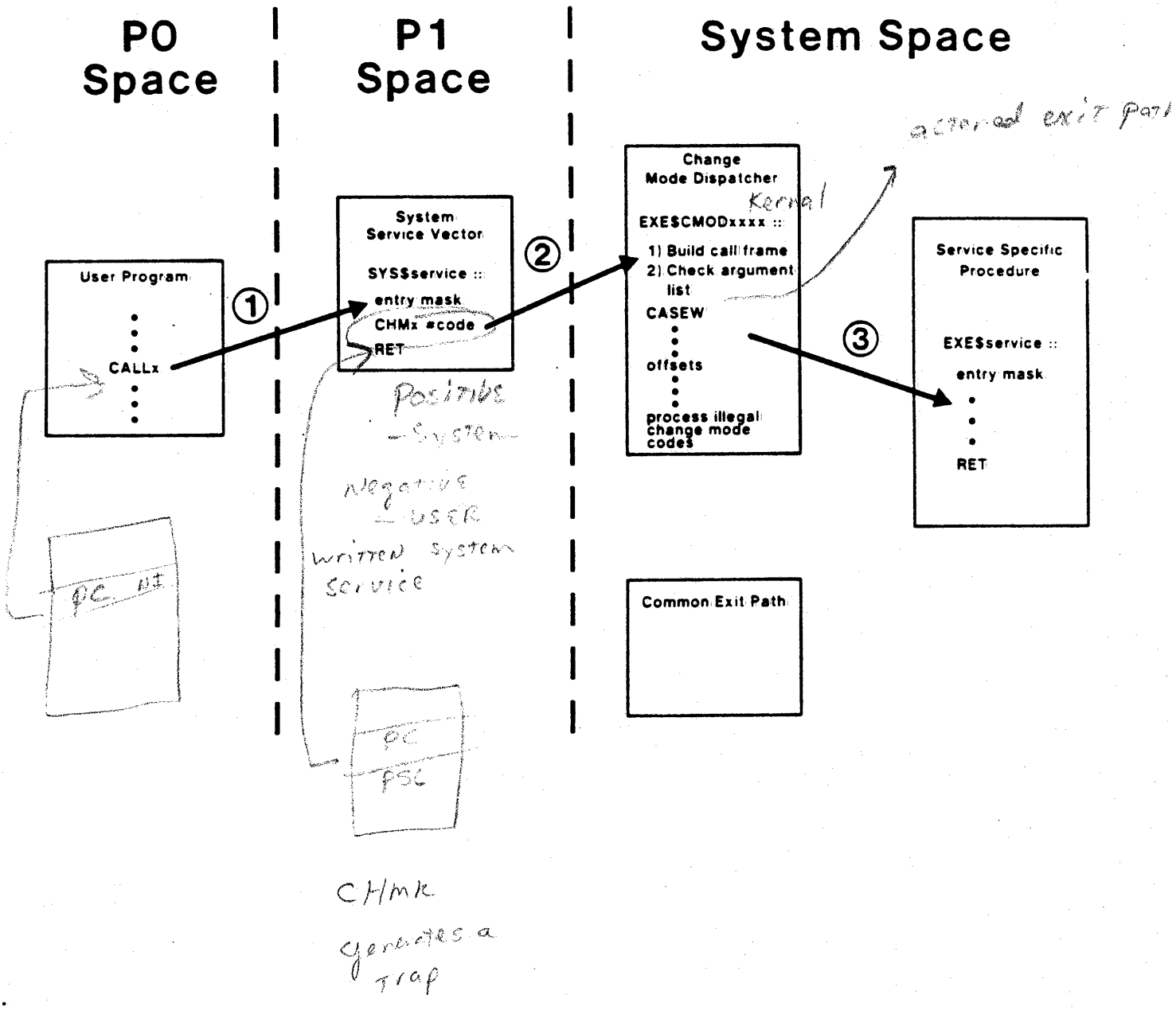
Commonly Used System Macros

IPL Control Macros

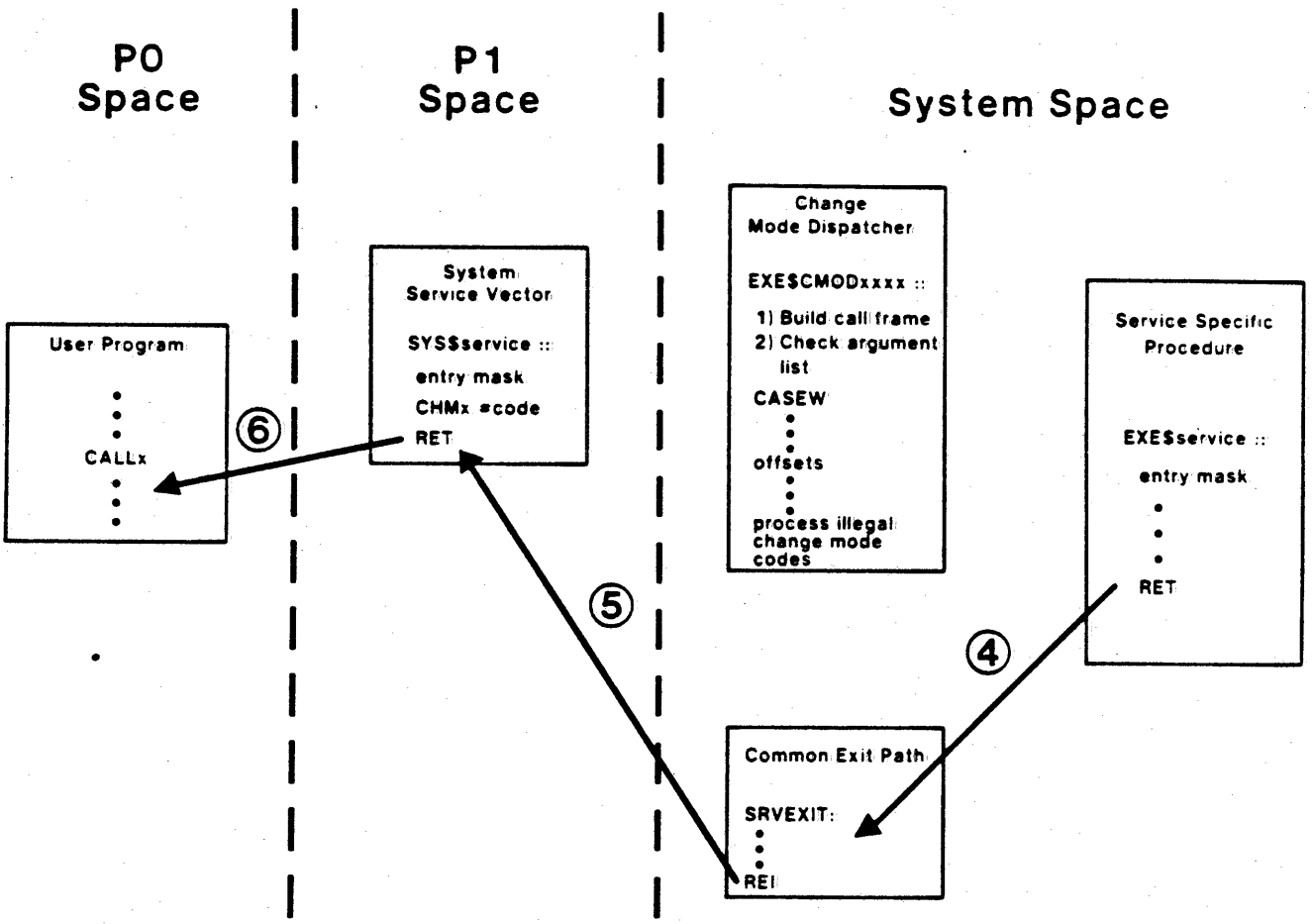
```
.MACRO  SETIPL IPL
        .IF NB  IPL
        MTPR    IPL,S^#PR$_IPL
        .IFF
        MTPR    #31,S^#PR$_IPL
        .ENDC
.ENDM   SETIPL


.MACRO  DSBINT IPL,DST
        .IF B   DST
        MFPR    S^#PR$_IPL,-(SP)
        .IFF
        MFPR    S^#PR$_IPL,DST
        .ENDC
        .IF B   IPL
        MTPR    #31,S^#PR$_IPL
        .IFF
        MTPR    IPL,S^#PR$_IPL
        .ENDC
.ENDM   DSBINT


.MACRO  ENBINT SRC
        .IF B   SRC
        MTPR    (SP)+,S^#PR$_IPL
        .IFF
        MTPR    SRC,S^#PR$_IPL
        .ENDC
.ENDM   ENBINT


.MACRO  SOFTINT IPL
        MTPR    IPL,S^#PR$_SIRR
.ENDM   SOFTINT
```



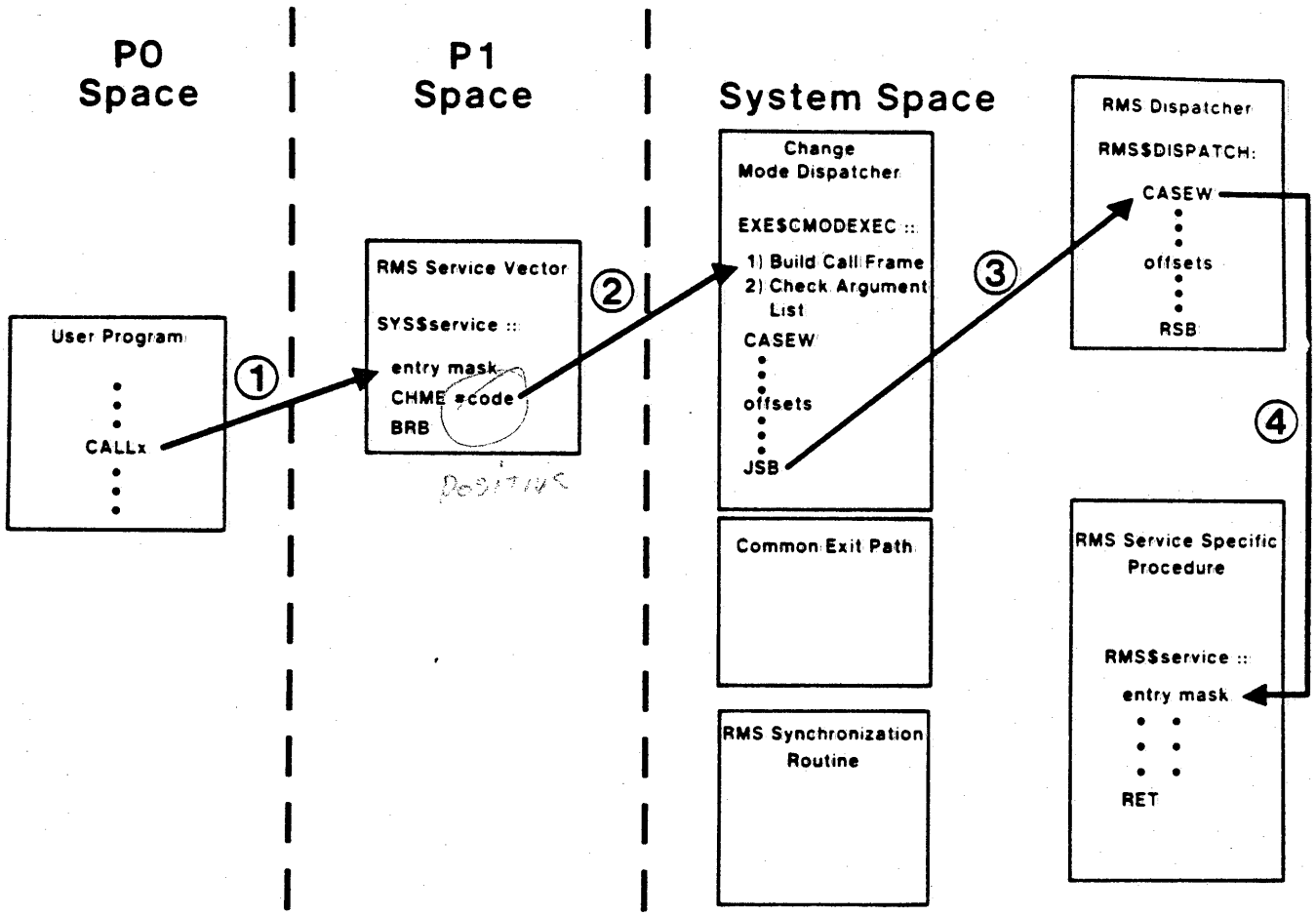
PATH TO SYSTEM SERVICES



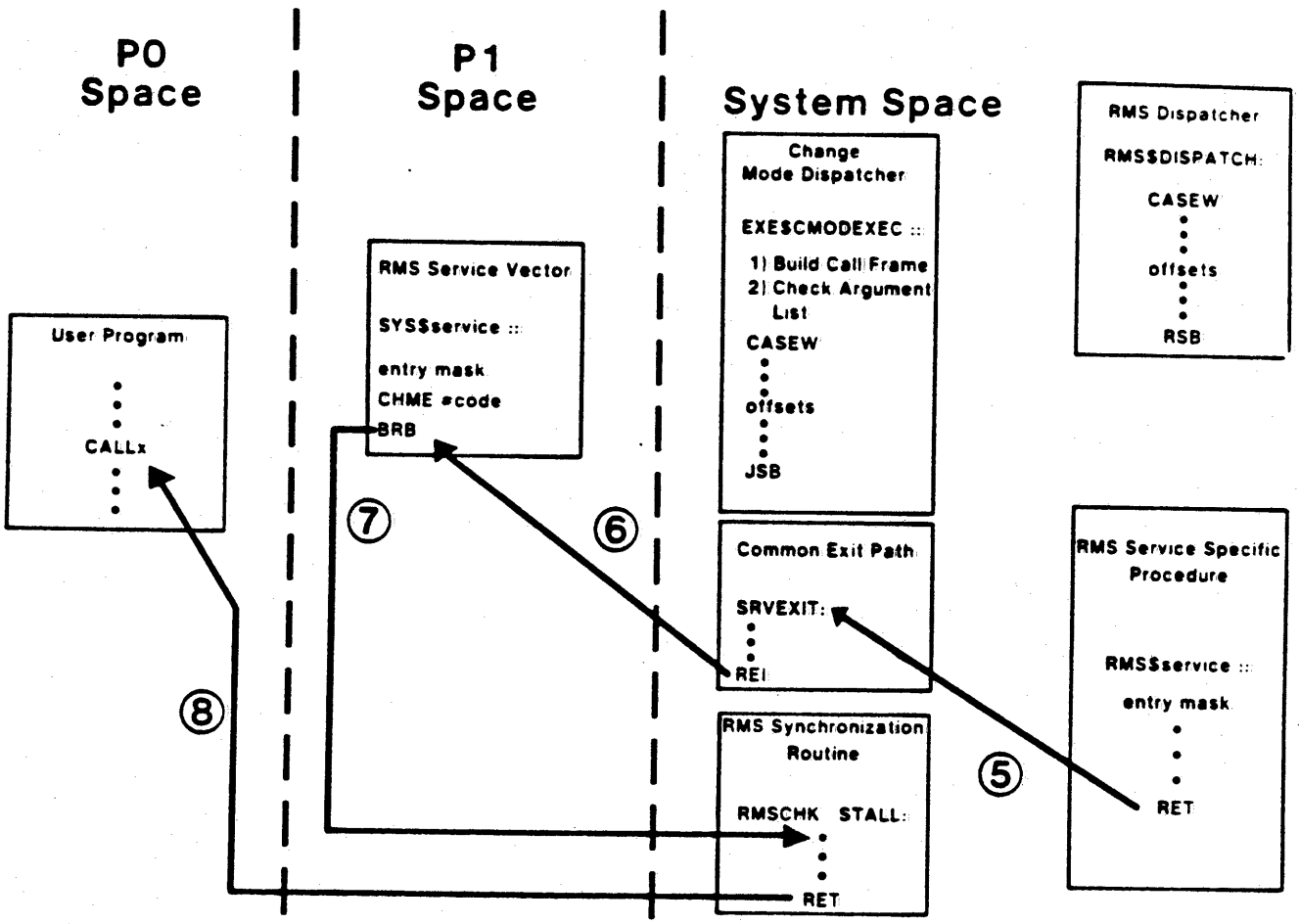
RETURN FROM SYSTEM SERVICES



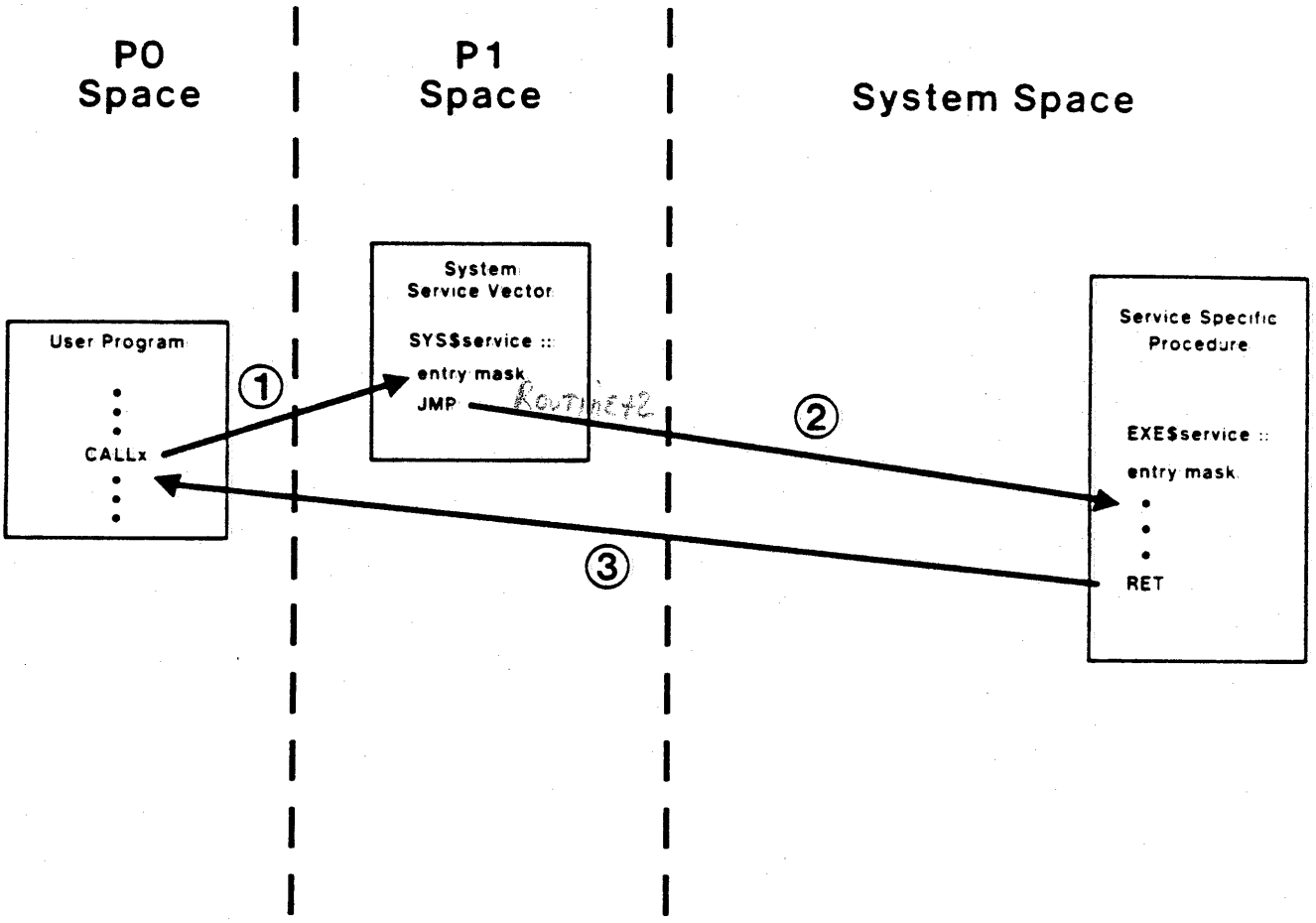
PATH TO RMS


P0 Space
P1 Space
System Space
User Program
CALLx
RMS Service Vector
SYS$service ::
entry mask
CHME #code
BRB
Change Mode Dispatcher
EXE$CMODEXEC ::
1) Build Call Frame
2) Check Argument List
CASEW
offsets
JSB
Common Exit Path
RMS Synchronization Routine
RMS Dispatcher
RMS$DISPATCH:
CASEW
offsets
RSB
RMS Service Specific Procedure
RMS$service ::
entry mask
RET
1
2
3
4

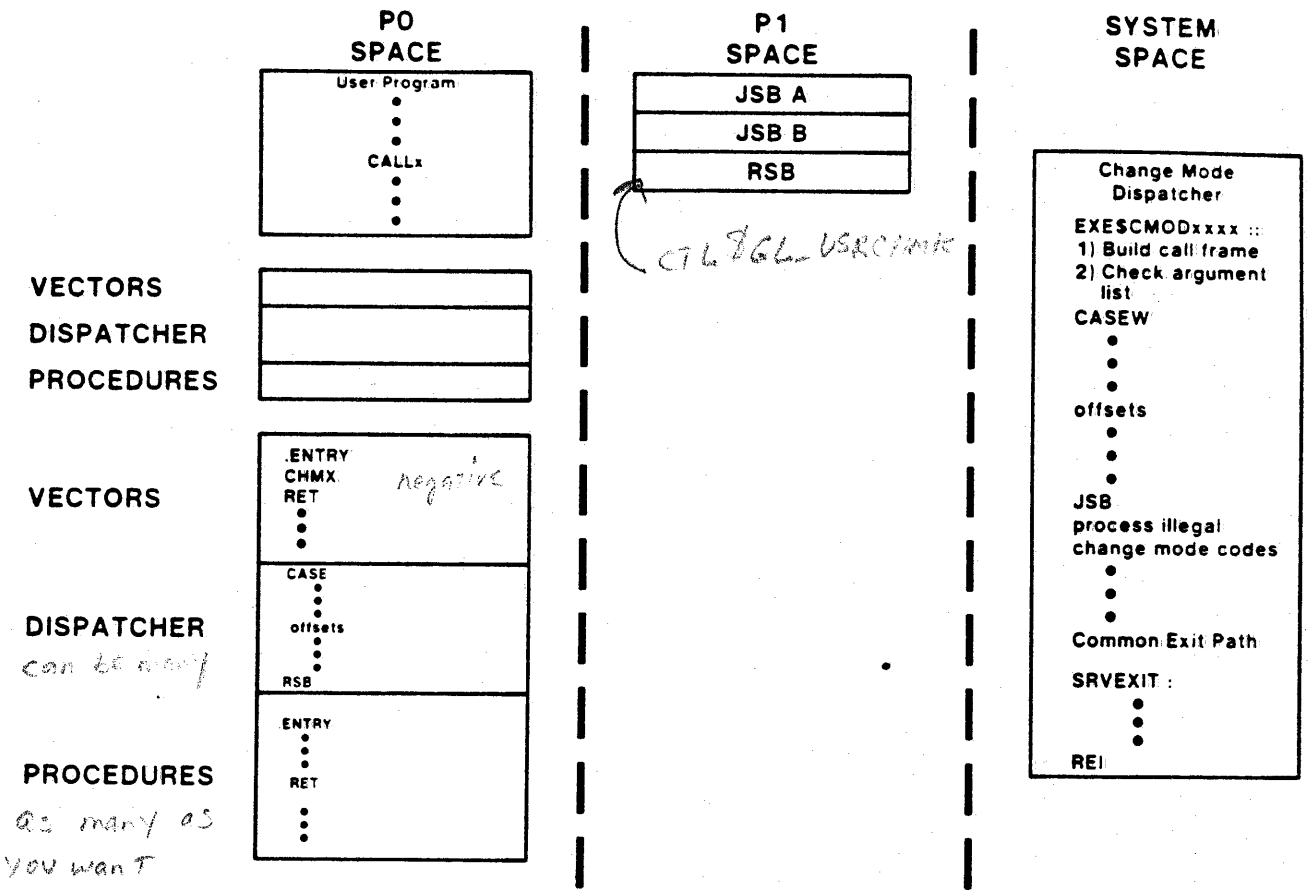
RETURN FROM RMS

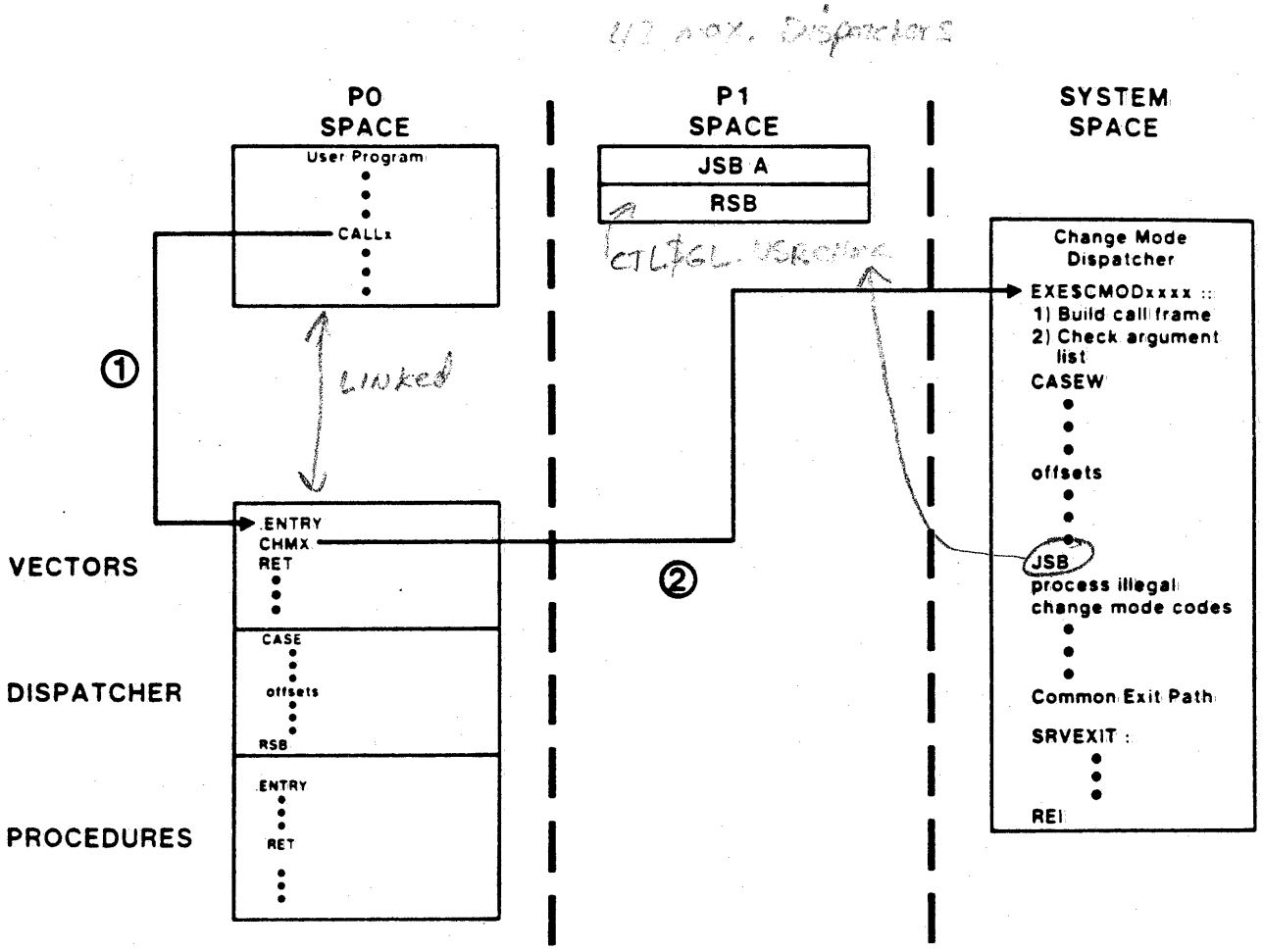


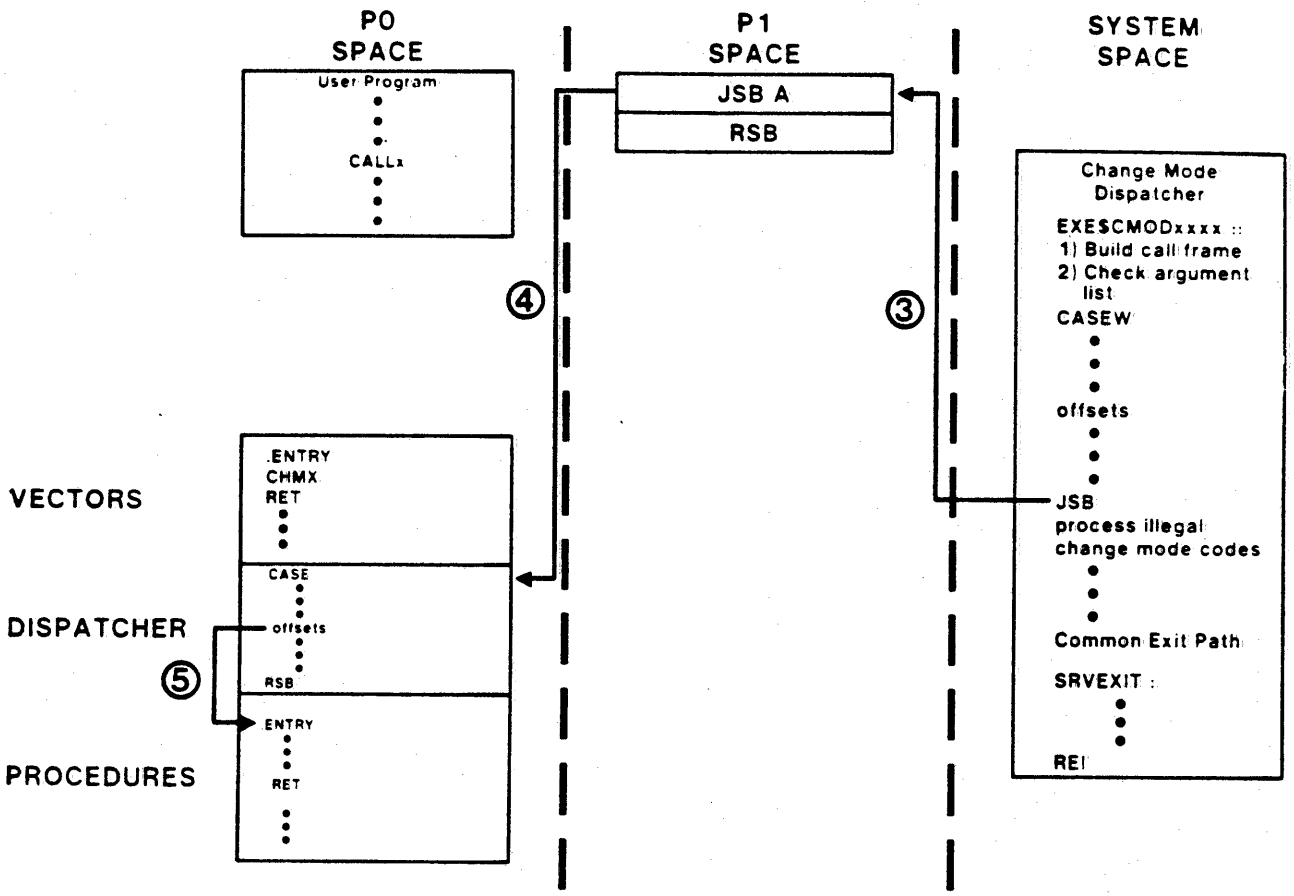
NON-PRIVILEGED SYSTEM SERVICE $FAO



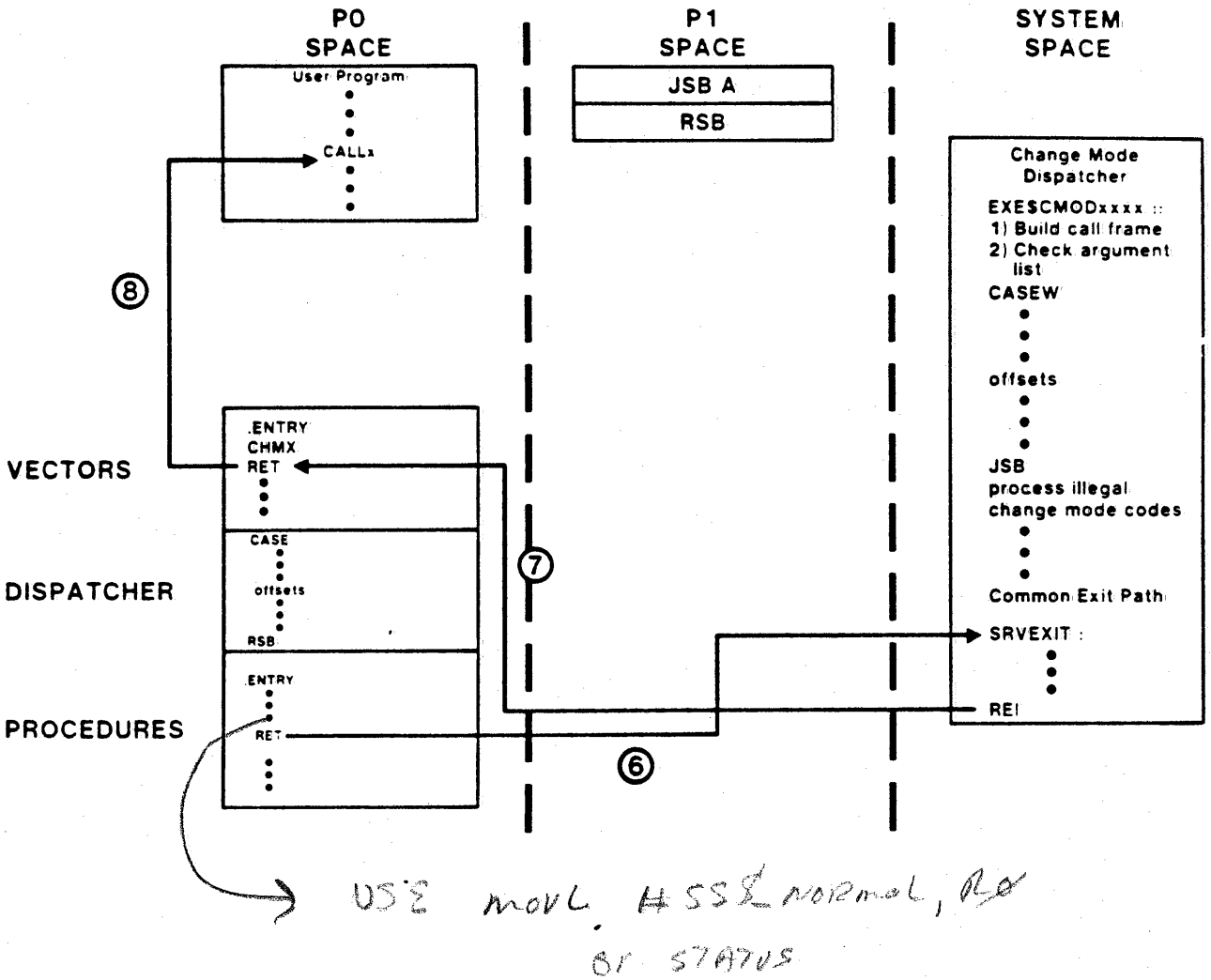
PATH TO USER-WRITTEN SYSTEM SERVICE




P0 SPACE
P1 SPACE
SYSTEM SPACE
User Program
CALLx
JSB A
RSB
Change Mode Dispatcher
EXESCMODxxxx :::
1) Build call frame
2) Check argument list
CASEW
offsets
JSB
process illegal change mode codes
Common Exit Path
SRVEXIT :
REI
Linked
①
②
VECTORS
DISPATCHER
PROCEDURES
.ENTRY
CHMX
RET
CASE
offsets
RSB
.ENTRY
RET


P0
SPACE
User Program
CALLx
P1
SPACE
JSB A
RSB
SYSTEM
SPACE
Change Mode
Dispatcher
EXE$CMODxxxx ::
1) Build call frame
2) Check argument
list
CASEW
offsets
JSB
process illegal
change mode codes
Common Exit Path
SRVEXIT :
REI
④
③
VECTORS
.ENTRY
CHMX
RET
DISPATCHER
CASE
offsets
RSB
⑤
PROCEDURES
ENTRY
RET

RETURN FROM USER-WRITTEN SYSTEM SERVICE



Commonly Used System Macros

Argument Probing Macros

Kernal mode

```
.MACRO   IFRD SIZ,ADR,DEST,MODE=#0
         PROBER  MODE,SIZ,ADR
         BNEQ    DEST
.ENDM    IFRD


.MACRO   IFNORD  SIZ,ADR,DEST,MODE=#0
         PROBER  MODE,SIZ,ADR
         BEQL    DEST
.ENDM    IFNORD


.MACRO   IFWRT   SIZ,ADR,DEST,MODE=#0
         PROBEW  MODE,SIZ,ADR
         BNEQ    DEST
.ENDM    IFWRT


.MACRO   IFNOWRT SIZ,ADR,DEST,MODE=#0
         PROBEW  MODE,SIZ,ADR
         BEQL    DEST
.ENDM    IFNOWRT
```

```
                                                    Sys$Examples:USSDISP.MAR
        .TITLE  USER_SYS_DISP - Example of user system service dispatcher
        .IDENT  'V02-000'
;
;*****************************************************************************
;*                                                                           *
;*  COPYRIGHT (c) 1980                                                       *
;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
;*                                                                           *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER     *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
;*  TRANSFERRED.                                                             *
;*                                                                           *
;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;*  CORPORATION.                                                             *
;*                                                                           *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;*                                                                           *
;*****************************************************************************
;
;
; Facility: Example of User Written System Services
;++
; Abstract:
;       This module contains an example dispatcher for user written
;       system services along with several sample services and a user
;       rundown example.  It is a template intend to serve as the starting
;       point for implementing a privileged shareable image containing your
;       own services.  When used as a template, the definitions and code
;       for the sample services should be removed.
;
; Overview:
;       User written system services are contained in privileged shareable
;       images that are linked into user program images in exactly the
;       same fashion as any shareable image.  The creation and installation
;       of a privileged, shareable image is slightly different from that
;       of an ordinary shareable image.  These differences are:
;
;               1. A vector defining the entry points and providing other
;                  control information to the image activator.  This vector
;                  is a the lowest address in an image section with the VEC
;                  attribute.
;
;               2. The shareable image is linked with the /PROTECT option
;                  that marks all of the image sections so that they will
;                  protected and given EXEC mode ownership by the image
;                  activator.
;
;               3. The shareable image MUST be installed /SHARE /PROTECT
;                  with the INSTALL utility in order for the image activator
;                  to connect the privileged shareable image to the change mode
;                  dispatchers.                                        (1)
;
;       A privileged shareable image implementing user written system services
;       is comprised of the following major components:
;
;               1. A transfer vector containing all of the entry points and
;                  collecting them at the lowest virtual address in the shareable
```

image without necessitating the relinking of images that use it.

2. A Privileged Library Vector in a PSECT with the VEC attribute that describes the entry points for dispatching EXEC and KERNEL mode services along with validation information.

3. A dispatcher for kernel mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize a kernel mode service request.

4. A dispatcher for executive mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize an executive mode service request.

5. Service routines to perform the various services.

The first four components are contained in this template and are most easily implemented in MACRO, while the service routines can be implemented in BLISS or MACRO. Other languages may be usable but are not recommended -- particularly if they require runtime support routines or are extravagant in their use of stack or are unable to generate PIC code.

This example is position-independent (PIC) and it is good practice to implement shareable images this way whenever possible.

;--

; Link Command File Example:

```
        $!
        $!      Command file to link User System Service example.
        $!
        $ LINK/PROTECT/NOSYSSHR/SHARE=USS/MAP=USS/FULL SYS$INPUT/OPTIONS
        !
        !       Options file for the link of User System Service example.
        !
                SYS$SYSTEM:SYS.STB/SELECTIVE
        !
        !       Create a separate cluster for the transfer vector.
        !
        CLUSTER=TRANSFER_VECTOR,,,SYS$DISK:[]USSDISP
        !
        GSMATCH=LEQUAL,1,1
```

;--

```
        .PAGE
        .SBTTL  Declarations and Equates

        Include Files

        .LIBRARY "SYS$LIBRARY:LIB.MLB"  ; Macro library for system structure
                                        ;  definitions
```

Macro Definitions

DEFINE_SERVICE - A macro to make the appropriate entries in several different PSECTs required to define an EXEC or KERNEL mode service. These include the transfer vector, the case table for dispatching, and a table containing the number of required arguments.

```
        DEFINE_SERVICE Name,Number_of_Arguments,Mode
```

```
        .PSECT  $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
        .ALIGN  QUAD                    ; Align entry points for speed and style
        .TRANSFER       NAME            ; Define name as universal symbol for entry
        .MASK   NAME                    ; Use entry mask defined in main routine
        .IF     IDN MODE,KERNEL
(1)     CHMK    #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and execute
        RET                             ; Return
(2)     KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter

        .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
        .BYTE   NARG                    ; Define number of required arguments

        .PSECT  USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
        .WORD   2+NAME-KCASE_BASE       ; Make entry in kernel mode CASE table

        .IFF
        CHME    #<ECODE_BASE+EXEC_COUNTER> ; Change to executive mode and execute
        RET                             ; Return
(3)     EXEC_COUNTER=EXEC_COUNTER+1     ; Advance counter

        .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
        .BYTE   NARG                    ; Define number of required arguments

(4)     .PSECT  USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
        .WORD   2+NAME-ECASE_BASE       ; Make entry in exec mode CASE table
        .ENDC                           ;
        .ENDM   DEFINE_SERVICE          ;

        Equated Symbols


        $PHDDEF                         ; Define process header offsets
        $PLVDEF                         ; Define PLV offsets and values
        $PRDEF                          ; Define processor register numbers

        Initialize counters for change mode dispatching codes

KERNEL_COUNTER=0                        ; Kernel code counter
EXEC_COUNTER=0  (5)                     ; Exec code counter


        Own Storage

        .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
KERNEL_NARG:                            ; Base of byte table containing the
                                        ;  number of required arguments.
        .PSECT  EXEC_NARG,BYTE,NOWRT,EXE,PIC
EXEC_NARG:                              ; Base of byte table containing the
                                        ;  number of required arguments.
        .PAGE
        .SBTTL  Transfer Vector and Service Definitions
;++
; The use of transfer vectors to effect entry to the user written system services
; enables some updating of the shareable image containing them without necessitati
; a re-link of all programs that call them.  The PSECT containinng the transfer
; vector will be positioned at the lowest virtual address in the shareable image
; and so long as the transfer vector is not re-ordered, programs linked with
; one version of the shareable image will continue to work with the next.
;
; Thus as additional services are added to a privileged shareable image, their
; definitions should be added to the end of the following list to ensure that
; programs using previous versions of it will not need to be re-linked.
; To completely avoid relinking existing programs the size of the privileged
; shareable image must not change so some padding will be required to provide
; the opportunity for future growth
```

```
DEFINE_SERVICE  USER_GET_TODR,1,KERNEL  ; Service to set value of time
                                        ;  of day register
DEFINE_SERVICE  USER_SET_PFC,2,KERNEL   ; Service to set value of process
                                        ;  default pagefault cluster
DEFINE_SERVICE  USER_NULL,0,EXEC        ; Null exec service
;
; The base values used to generate the dispatching codes should be negative for
; user services and must be chosen to avoid overlap with any other privileged
; shareable images that will be used concurrently.  Their definition is
; deferred to this point in the assembly to cause their use in the preceding
; macro calls to be forward references that guarantee the size of the change
; mode instructions to be four bytes.  This satisfies an assumption that is
; made by for services that have to wait and be retried.  The PC for retrying
; the change mode instruction that invokes the service is assumed to be 4 bytes
; less than that saved in the change mode exception frame.  Of course, the particul
; service routine determines whether this is possible.
;
KCODE_BASE=-1024                        ; Base CHMK code value for these services
ECODE_BASE=-1024                        ; Base CHME code value for these services
        .PAGE
        .SBTTL  Change Mode Dispatcher Vector Block
;++
; This vector is used by the image activator to connect the privileged shareable
; image to the VMS change mode dispatcher.  The offsets in the vector are self-
; relative to enable the construction of position independent images.  The system
; version number will be used by the image activator to verify that this shareable
; image was linked with the symbol table for the current system.
;
;                       Change Mode Vector Format
;
;           +-----------------------------------------+
;           !              Vector Type Code           !   PLV$L_TYPE
;           !              (PLV$C_TYP_CMOD)           !
;           +-----------------------------------------+
;           !           System Version Number         !   PLV$L_VERSION
;           !              (SYS$K_VERSION)            !
;           +-----------------------------------------+
;           !      Kernel Mode Dispatcher Offset      !   PLV$L_KERNEL
;           !                                         !
;           +-----------------------------------------+
;           !         Exec Mode Entry Offset          !   PLV$L_EXEC
;           !                                         !
;           +-----------------------------------------+
;           !       User Rundown Service Offset       !   PLV$L_USRUNDWN
;           !                                         !
;           +-----------------------------------------+
;           !                Reserved                 !
;           !                                         !
;           +-----------------------------------------+
;           !          RMS Dispatcher Offset          !   PLV$L_RMS
;           !                                         !
;           +-----------------------------------------+
;           !             Address Check               !   PLV$L_CHECK
;           !                                         !
;           +-----------------------------------------+
;
;
        .PSECT  USER_SERVICES,PAGE,VEC,PIC,NOWRT,EXE

        .LONG   PLV$C_TYP_CMOD          ; Set type of vector to change mode dispat
        .LONG   SYS$K_VERSION           ; Identify system version
        .LONG   KERNEL_DISPATCH-.       ; Offset to kernel mode dispatcher
        .LONG   EXEC_DISPATCH-.         ; Offset to executive mode dispatcher
        .LONG   USER_RUNDOWN-.          ; Offset to user rundown service
```

Update

We have recently discovered that the dispatcher in the user written system service template presents a real gotcha to users writing system service code in higher level languages. I refer to the template that is distributed with VMS in the file [SYSHLP.EXAMPLES]USSDISP.MAR.

The problem is that most higher level languages (such as BLISS) assume (as rightly they should) that FP and SP are equal upon entry to a procedure. After all, this is how the CALL instruction leaves them. Unfortunately, the mechanism used by VMS to dispatch to user-written or loadable system services, in conjunction with the dispatching code in the system service template, does not.

After the bottom level stack frame is built by the VMS change mode dispatcher, two JSB's are executed in the process of reaching the user's service dispatcher: one JSB to call into the vector, and another in the vector to call into the service dispatcher. Thus, there are two longwords on the stack between FP and SP.

The BLISS compiler assumes that FP and SP are equal on entry to a procedure, and will address stack locals using either register, whichever happens to be more convenient (i.e., allows a shorter offset). The current compiler uses SP if there is no advantage to either. This problem is particularly insidious in that a routine which works at present could cease working if modified (by adding more locals to the stack), or even just recompiled with some future compiler. Other languages may have similar problems; I am not sufficiently familiar with them to describe them in detail.

This problem can be fixed with a simple change to the user system service dispatcher. What follows is the code in the kernel mode service dispatcher; an identical piece of code exists for dispatching exec mode services.

```
KACCVIO:
        MOVZWL  #SS$_ACCVIO,R0          ; Kernel access violation
        RET                             ; Set access violation status code
                                        ;  and return
KINSFARG:
        MOVZWL  #SS$_INSFARG,R0         ; Kernel insufficient arguments.
        RET                             ; Set status code and
                                        ;  return
KNOTME: RSB                             ; RSB to forward request

KERNEL_DISPATCH::                       ; Entry to dispatcher
        MOVAB   W^-KCODE_BASE(R0),R1    ; Normalize dispatch code value

    BLSS    KNOTME                  ; Branch if code value too low
    CMPW    R1,#KERNEL_COUNTER      ; Check high limit
    BGEQU   KNOTME                  ; Branch if out of range
;
; The dispatch code has now been verified as being handled by this dispatcher,
; now the argument list will be probed and the required number of arguments
; verified.
;
        MOVZBL  W^KERNEL_NARG[R1],R1    ; Get required argument count
        MOVAL   @#4[R1],R1              ; Compute byte count including arg count
        IFNORD  R1,(AP),KACCVIO         ; Branch if arglist not readable
        CMPB    (AP),W^<KERNEL_NARG-KCODE_BASE>[R0] ; Check for required number
        BLSSU   KINSFARG                ;  of arguments
        CASEW   R0,-                    ; Case on change mode
                -                       ; argument value
                #KCODE_BASE,-           ; Base value
                #<KERNEL_COUNTER-1>     ; Limit value (number of entries)
KCASE_BASE:                             ; Case table base address for DEFINE_SERVICE
;
;       Case table entries are made in the PSECT USER_KERNEL_DISP1 by
;       invocations of the DEFINE_SERVICE macro.  The three PSECTS,
;       USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-time.
;
        .PSECT  USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
        RSB                             ; Return to reject out of
                                        ; range value
```

Because the change mode code has already been range checked by the time the CASEW instruction is executed, the RSB at the end of the dispatcher is never reached and is extraneous. Therefore, the JSB return addresses on the stack are no longer necessary at this point. As a result, the problem can be fixed by inserting the instruction

```
MOVL    FP,SP
```

immediately before the CASEW instruction to reset the stack, and changing the RSB at the end into a

```
BUG_CHECK IVSSRVRQST,FATAL      ; Invalid system service request
```

The system service dispatch template shipped with the next release of VMS will contain this modification. Obviously, however, any existing loadable system services contain their own copies of this code, and should be modified by their maintainers if they are written in higher level languages. (It might be a good idea to modify the dispatcher even if the service code is written in MACRO, since someone making a future change to the module might also make the obvious assumption that SP and FP are equal on entry.)

```
        .LONG   0                       ; No RMS dispatcher
        .LONG   0                       ; Address check - PIC image
        .PAGE
        .SBTTL  Kernel Mode Dispatcher
;++
; Input Parameters:
;
;       (SP) - Return address if bad change mode value
;
;        R0  - Change mode argument value.
;
;        R4  - Current PCB Address. (Therefore R4 must be specified in all
;                register save masks for kernel routines.)
;
;        AP  - Argument pointer existing when the change
;              mode instruction was executed.
;
;        FP  - Address of minimal call frame to exit
;              the change mode dispatcher and return to
;              the original mode.
;--
        .PSECT  USER_KERNEL_DISP0,BYTE,NOWRT,EXE,PIC
KACCVIO:                                ; Kernel access violation
        MOVZWL  #SS$_ACCVIO,R0          ; Set access violation status code
        RET                             ;  and return
KINSFARG:                               ; Kernel insufficient arguments.
        MOVZWL  #SS$_INSFARG,R0         ; Set status code and
        RET                             ;  return
KNOTME: RSB                             ; RSB to forward request

KERNEL_DISPATCH::                       ; Entry to dispatcher
        MOVAB   W^-KCODE_BASE(R0),R1    ; Normalize dispatch code value
        BLSS    KNOTME                  ; Branch if code value too low
        CMPW    R1,#KERNEL_COUNTER      ; Check high limit
        BGEQU   KNOTME                  ; Branch if out of range
;
; The dispatch code has now been verified as being handled by this dispatcher,
; now the argument list will be probed and the required number of arguments
; verified.
;
        MOVZBL  W^KERNEL_NARG[R1],R1    ; Get required argument count
        MOVAL   @#4[R1],R1              ; Compute byte count including arg count
        IFNORD  R1,(AP),KACCVIO         ; Branch if arglist not readable
        CMPB    (AP),W^<KERNEL_NARG-KCODE_BASE>[R0] ; Check for required number
        BLSSU   KINSFARG                ;  of arguments
        CASEW   R0,-                    ; Case on change mode
                -                       ; argument value
                #KCODE_BASE,-           ; Base value
                #<KERNEL_COUNTER-1>     ; Limit value (number of entries)
KCASE_BASE:                             ; Case table base address for DEFINE_SERVI
;
;       Case table entries are made in the PSECT USER_KERNEL_DISP1 by
;       invocations of the DEFINE_SERVICE macro.  The three PSECTS,
;       USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-time.
;
        .PSECT  USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
        RSB                             ; Return to reject out of
                                        ; range value
        .PAGE
        .SBTTL  Executive Mode Dispatcher
;++
; Input Parameters:
;
;       (SP) - Return address if bad change mode value
;
```

```
;
;        AP  - Argument pointer existing when the change
;              mode instruction was executed.
;
;        FP  - Address of minimal call frame to exit
;              the change mode dispatcher and return to
;              the original mode.
;
        .PSECT  USER_EXEC_DISP0,BYTE,NOWRT,EXE,PIC
EACCVIO:                                ; Exec access violation
        MOVZWL  #SS$_ACCVIO,R0          ; Set access violation status code
        RET                             ;  and return
EINSFARG:                               ; Exec insufficient arguments.
        MOVZWL  #SS$_INSFARG,R0         ; Set status code and
        RET                             ;  return
ENOTME: RSB                             ; RSB to forward request

EXEC_DISPATCH::                         ; Entry to dispatcher
        MOVAB   W^-ECODE_BASE(R0),R1    ; Normalize dispatch code value
        BLSS    ENOTME                  ; Branch if code value too low
        CMPW    R1,#EXEC_COUNTER        ; Check high limit
        BGEQU   ENOTME                  ; Branch if out of range
;
; The dispatch code has now been verified as being handled by this dispatcher,
; now the argument list will be probed and the required number of arguments
; verified.
;
        MOVZBL  W^EXEC_NARG[R1],R1      ; Get required argument count
        MOVAL   @#4[R1],R1              ; Compute byte count including arg count
        IFNORD  R1,(AP),EACCVIO         ; Branch if arglist not readable
        CMPB    (AP),W^<EXEC_NARG-ECODE_BASE>[R0] ; Check for required number
        BLSSU   EINSFARG                ;  of arguments
        CASEW   R0,-                    ; Case on change mode
                -                       ; argument value
                #ECODE_BASE,-           ; Base value
                #<EXEC_COUNTER-1>       ; Limit value (number of entries)
ECASE_BASE:                             ; Case table base address for DEFINE_SERVI
;
;       Case table entries are made in the PSECT USER_EXEC_DISP1 by
;       invocations of the DEFINE_SERVICE macro.  The three PSECTS,
;       USER_EXEC_DISP0,1,2 will be abutted in lexical order at link-time.
;
        .PSECT  USER_EXEC_DISP2,BYTE,NOWRT,EXE,PIC
        RSB                             ; Return to reject out of
                                        ; range value
        .PAGE
        .SBTTL  User Rundown Service
;++
; Functional description:
;       This service is invoked from within the kernel mode system service
;       that performs image rundown.  It is invoked before any system
;       rundown functions (i.e. deassign channels, release memory) are
;       performed.  User code should not invoked any RMS services or RTL
;       routines, must not signal any exceptions.  User code can invoke
;       most system services except those that use RMS (e.g. $PUTMSG).
;
; Calling sequence:
;       JSB     USER_RUNDOWN
;       Entered at IPL=0 and must leave at IPL=0.
;
; Input Parameters:
;        R4  - Current PCB Address. (Therefore R4 must be specified in all
;                register save masks for kernel routines.)
;
;        R7  - Access mode parameter to $RUNDWN maximized with previous mode
```

```
;               fragment pointer existing when the $RUNDWN system
;               service was invoked.
;
;       4(AP) - Access mode parameter to $RUNDWN
;
;--
        .PSECT  USER_CODE,BYTE,NOWRT,EXE,PIC

USER_RUNDOWN::                          ; Entry point for service
        PUSHL   R2                      ; Save a register
        PUSHAB  B^SYSOUT                ; Set up address of descriptor
        PUSHL   S^#SYS_LEN              ; Set up length
        MOVAL   -(SP), R2               ; Grab some temporary storage
        $ASSIGN_S 4(R2), (R2)           ; Assign a channel to operator console
        BLBC    R0, 10$                 ; Error
        $OUTPUT   (R2), S^#MSG_LEN, B^MSG       ; Print the message on operator cor
        $DASSGN_S (R2)                  ; Get rid of the channel
10$:    ADDL2   #12, SP                 ; Clean up
        MOVL    (SP)+, R2               ; Restore register
        RSB
;
SYSOUT: .ASCII  /_OPA0:/
SYS_LEN=.-SYSOUT
MSG:    .ASCII  /*** Image exiting ***/
MSG_LEN=.-MSG
        .PAGE
        .SBTTL  Get Time of Day Register Value
;++
; Functional Description:
;       This routine reads the content of the hardware time of day
;       processor register and stores the resulting value at the
;       specified address.
;
; Input Parameters:
;       04(AP) - Address to return time of day value   (8) longword
;       R4 - Address of current PCB
;
; Output Parameters:
;       R0 - Completion Status Code
;--
        .ENTRY  USER_GET_TODR,^M<R2,R3,R4>
(9) ->  MOVL    4(AP),R1                ; Get address to store time of day register
        IFNOWRT #4,(R1),10$             ; Branch if not writable
(10) -> MFPR    #PR$_TODR,(R1)          ; Return current time of day register
        MOVL    #SS$_NORMAL,R0          ; Set normal completion status
        RET                             ;  and return

10$:    MOVZWL  #SS$_ACCVIO,R0          ; Indicate access violation
        RET                             ;
        .PAGE
        .SBTTL  Set Page Fault Cluster Factor
;++
; Functional Description:
;       This routine sets the page fault cluster to the specified value
;       and returns the previous value.
;
; Input Parameters:
;       04(AP) - New value for Page Fault Cluster factor
;       08(AP) - Address to return previous value
;                (0 means none)
;       R4 - PCB address of current process
;
; Output Parameters:
;       R0 - Completion Status code
;--
```

```
        MOVL    @#CTL$GL_PCB,R5         ; Get address of process header
        MOVL    8(AP),R1                ; Get address to store previous value
        BEQL    10$                     ; Branch if none
        IFNOWRT #4,(R1),30$             ; Branch if not writable
        MOVZBL  PHD$B_DFPFC(R5),(R1)    ; Return current value
10$:    MOVB    4(AP),R0                ; Get new value for PFC
        CMPB    R0,#127                 ; Check for legal value
        BLEQU   20$                     ; Branch if legal
        MOVB    #127,R0                 ; Set to maximum value
20$:    MOVB    R0,PHD$B_DFPFC(R5)      ; Set new value into PHD
        MOVL    #SS$_NORMAL,R0          ; Set normal completion status
        RET                             ;  and return

30$:    MOVZWL  #SS$_ACCVIO,R0          ; Indicate access violation
        RET                             ;

        .PAGE
        .SBTTL  Null Service
;++
; Functional Description:
;
; Input Parameters:
;
; Output Parameters:
;
;--

        .ENTRY  USER_NULL,^M<>          ; Entry definition
        MOVZWL  #SS$_NORMAL,R0          ; Set normal completion status
        RET                             ;  and return

        .END
```

```
        .TITLE  USSTEST
        .IDENT  'V02-000'
;
;*****************************************************************************
;*                                                                           *
;*  COPYRIGHT (c) 1980                                                       *
;*  BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.                         *
;*                                                                           *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER     *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
;*  TRANSFERRED.                                                             *
;*                                                                           *
;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;*  CORPORATION.                                                             *
;*                                                                           *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;*                                                                           *
;*****************************************************************************
;
;
; Facility: Example of User Written System Services
;++
; Abstract:
;       This module contains an example of a program that invokes a sample
;       user-written system service that is contained in a privileged
;       shareable library.  The module USSDISP contains the sample service
;       and associated dispatching code being invoked by this simple test
;       program.
;--
; Link Command File:
;
;       $ !
;       $ !     Link Command file for USSTEST
;       $ !
;       $ LINK USSTEST/MAP/FULL,SYS$INPUT/OPTIONS
;       !
;       !       Options file for USSTEST
;               USS.EXE/SHARE
;
;--
BUF:    .LONG   0                       ; Location to receive TODR contents
        .PAGE
        .SBTTL  Sample invocation of user written system service
;++
; Functional Description:
;       This routine shows an invocation of the example user system service that
;       will read the contents of the time of day register.
;
;       As can be seen by this example, the privileged nature of the code used
;       to implement the readings of the TODR is not visible to the caller.
;       For coding convenience and better maintainability, the code can be
;       generated by macros patterned on the standard VMS system service macros.
;
;--


        .ENTRY  USSTEST,^M<>            ; Entry mask and definition
```

(1) must be in SYS$share

(2)

```
;  return value
;Invoke routine in privileged library
;  to set value from Time-of-day register
;
;
        CALLS   #1,G^USER_GET_TODR
        RET
        .END    USSTEST
```

3

should SUBS R2,#0$

PUSH R6

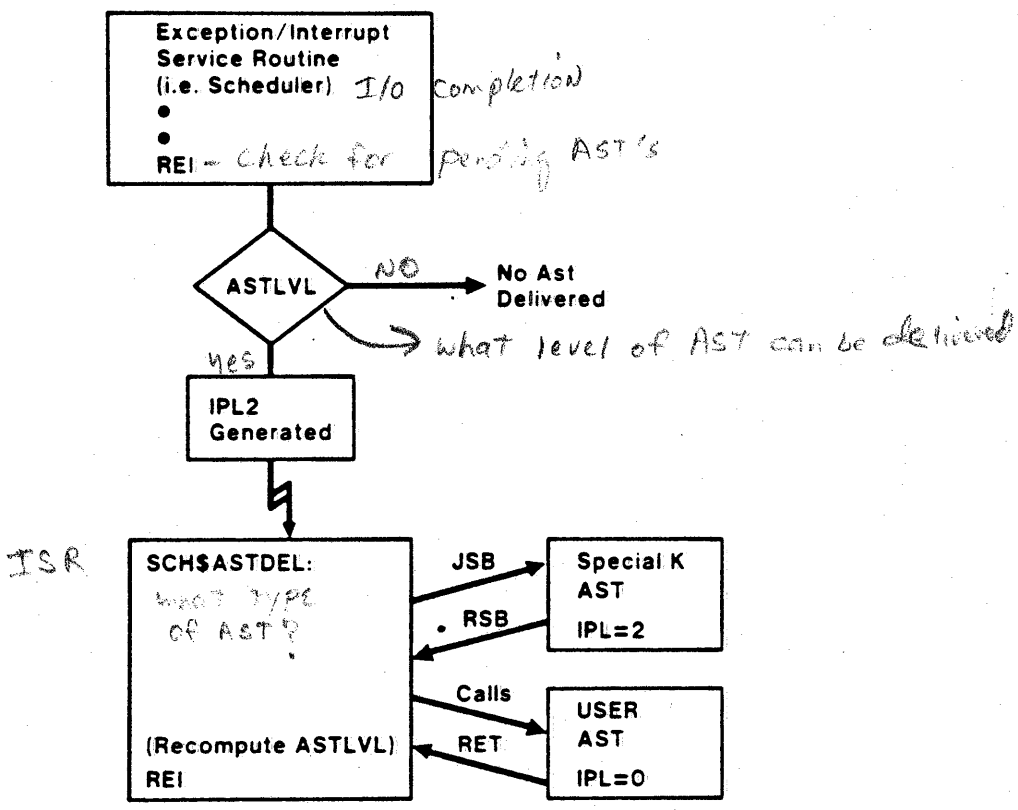
calls #2, G^LIB$stop

Commonly Used System Macros

Privilege Checking Macros

```
.MACRO  IFPRIV PRIV,DEST,PCBREG=R4
        .IF DIF <PRIV>,<R1>
        .IF DIF <PRIV>,<R2>
        BBS     #PRV$V_'PRIV,@PCB$L_PHD(PCBREG),DEST
        .IFF
        BBS     PRIV,@PCB$L_PHD(PCBREG),DEST
        .ENDC
        .IFF
        BBS     PRIV,@PCB$L_PHD(PCBREG),DEST
        .ENDC
.ENDM   IFPRIV


.MACRO  IFNPRIV PRIV,DEST,PCBREG=R4
        .IF DIF <PRIV>,<R1>
        .IF DIF <PRIV>,<R2>
        BBC     #PRV$V_'PRIV,@PCB$L_PHD(PCBREG),DEST
        .IFF
        BBC     PRIV,@PCB$L_PHD(PCBREG),DEST
        .ENDC
        .IFF
        BBC     PRIV,@PCB$L_PHD(PCBREG),DEST
        .ENDC
.ENDM   IFNPRIV
```

AST Delivery

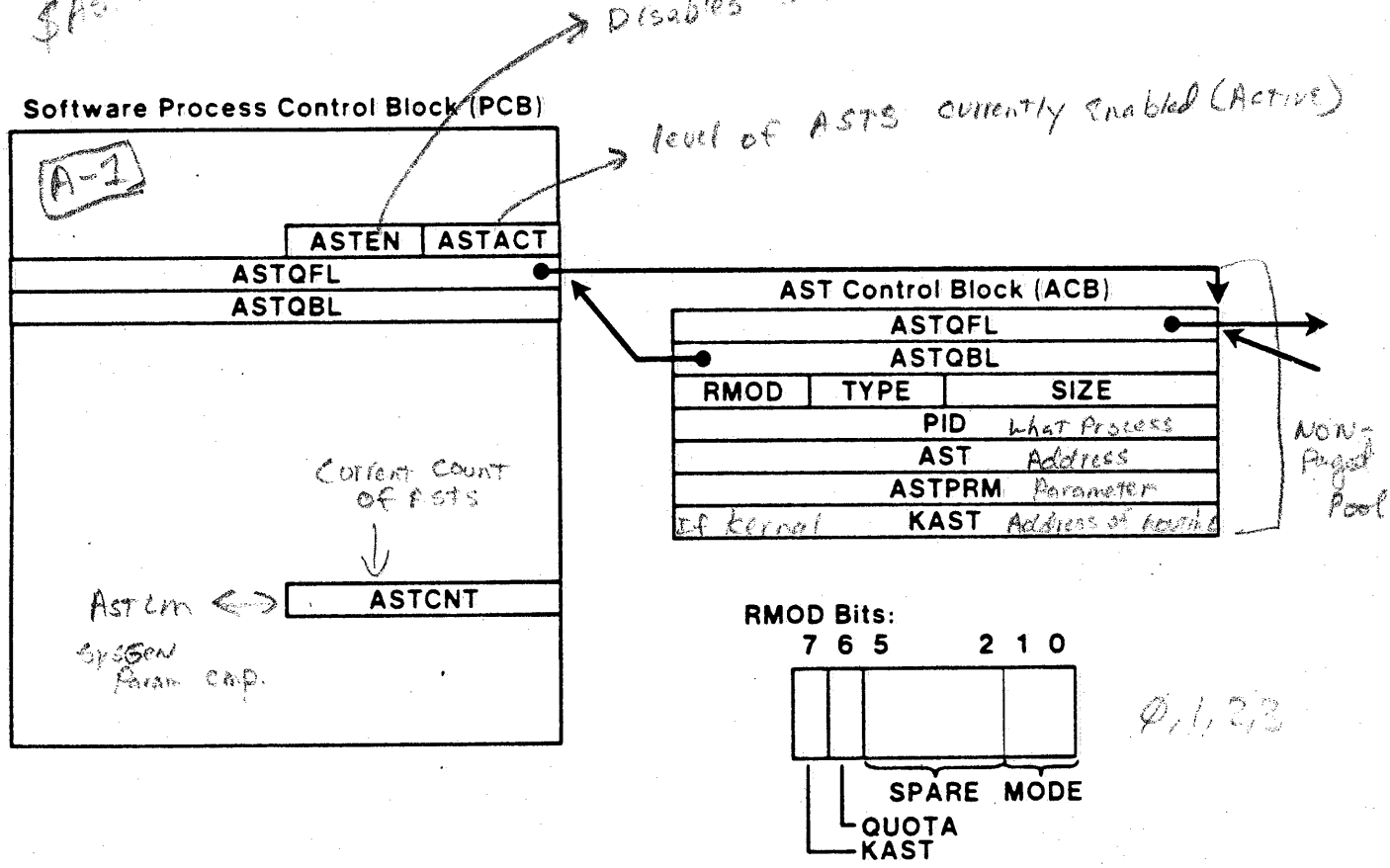


Chapter 5 ...

AST DATA STRUCTURE

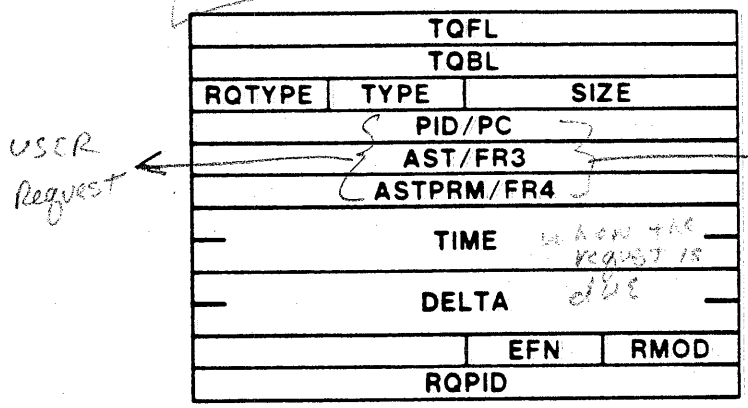
$Persdef

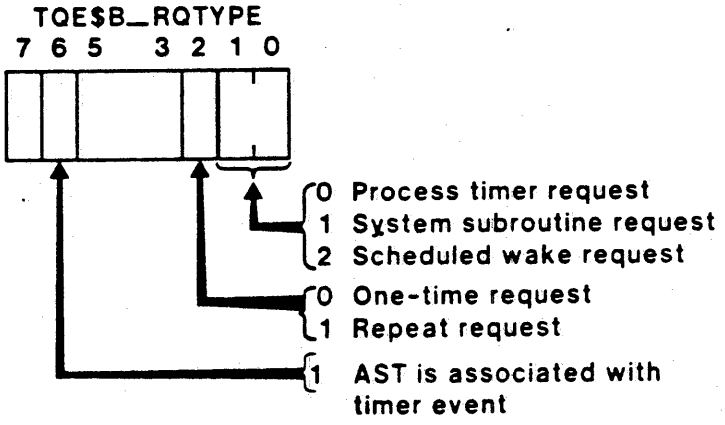
$Setast



Example in RSE

TIMER QUEUE ELEMENT

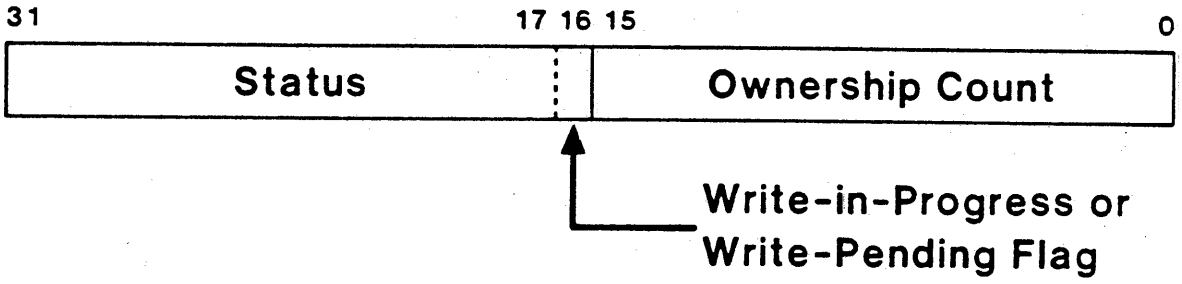




PROCESS SYNCHRONIZATION MUTEX

Mutual exclusion Semphore

- protect access to Data Structures
- "lock" for either Read only (1 writer)
  or write only (multiple writers)



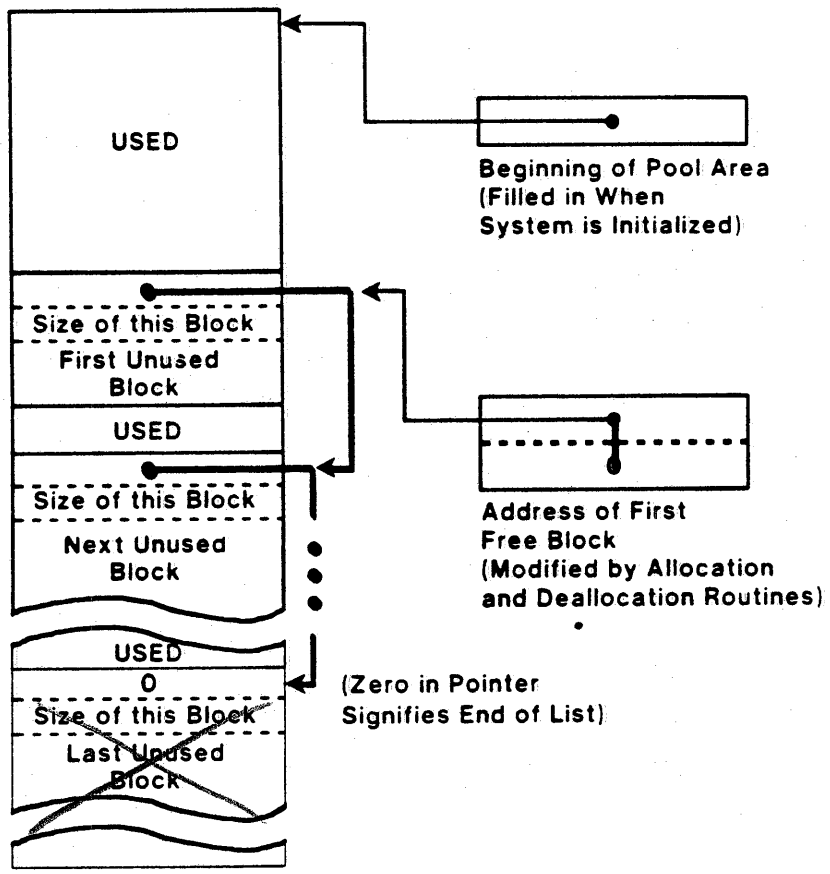
R0 = address of MULTEX
EXE$GL_CEBMTX

R4 = your PCB address (Set when in kernel mode automatically)

JSB G^SCH$LockR
-OR-
G^SCH$LockW

JSB G^SCH$UNLock

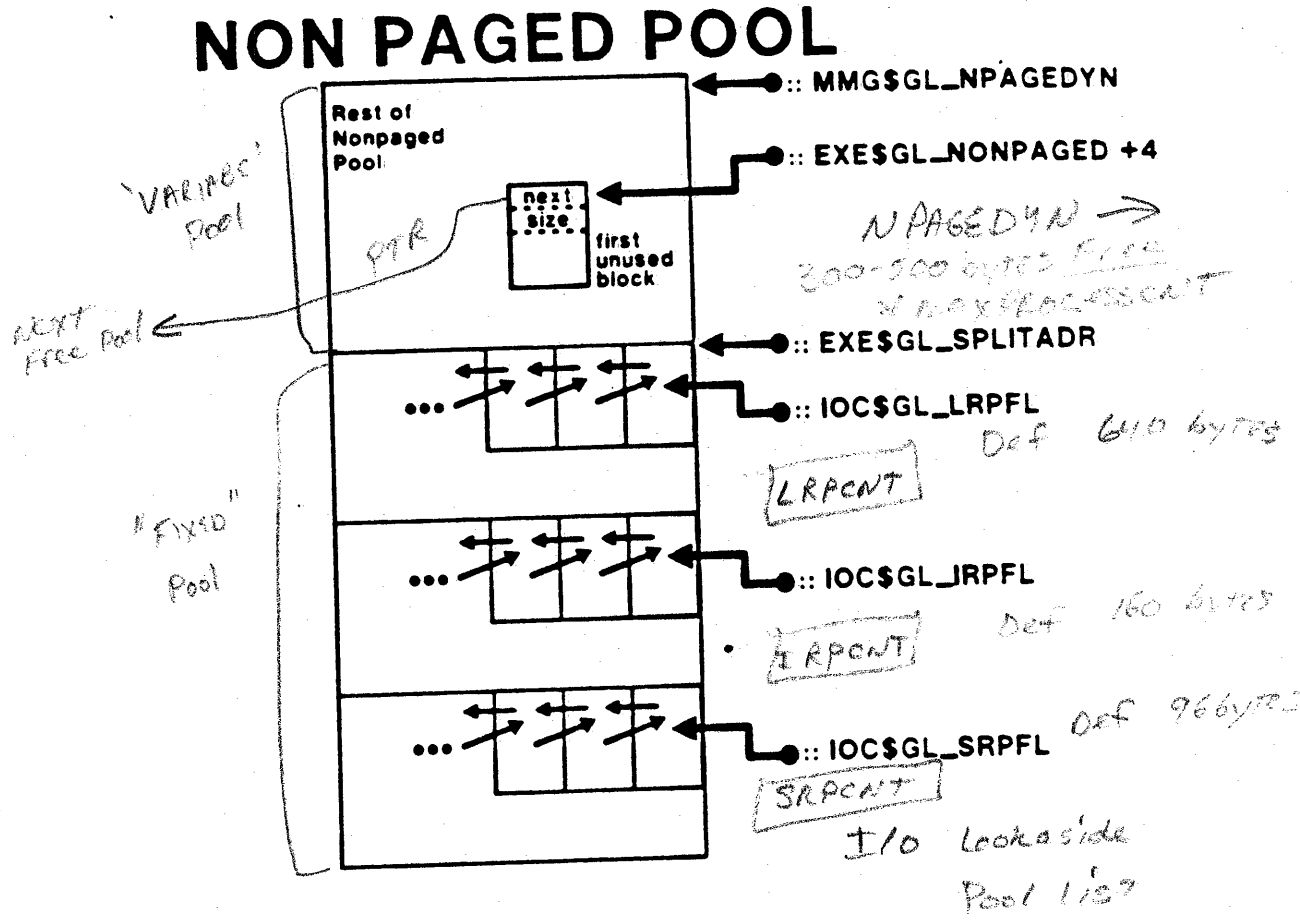
DYNAMIC MEMORY (PAGED POOL)



Sho mon/pool/full

25-8 IDSM

NON-DYNAMIC MEMORY



Fixed Pool used for: PCB, JIB, TQE, [illegible], DECnet

'Variable' pool for: Drivers

Taken away from WS

Virtual Pool ~ 3X Normal parameter

NPAGVIR LRPCOUNTV
IRPCOUNTV
SRPCOUNTV

```
                                        0000     1          $CEBDEF
                                    00000000     2          .PSECT  DATA,NOEXE
                                        0000     3
                                        0000     4 CEB_DESC:
                                   0000 0000     5          .WORD   0
                                   01 0E 0002     6          .BYTE   14,1
                                0000000C' 0004     7          .ADDRESS        CEB_NAME
                                        0008     8 CEB_NAME_LEN:
                                00000000  0008     9          .LONG   0
                                        000C    10 CEB_NAME:
                                0000001B  000C    11          .BLKB   15
                                        001B    12 HEADER:
20 20 20 20 45 4D 41 4E 5F 42 45 43  001B    13          .ASCII  /CEB_NAME          OWNER           PROT      PERM/<10>
20 20 52 45 4E 57 4F 20 20 20 20 20  0027
54 4F 52 50 20 20 20 20 20 20 20 20  0033
      0A 4D 52 45 50 20 20 20 20 20  003F
20 20 20 2D 2D 2D 2D 2D 2D 2D 2D 0D  0049    14          .ASCII  <13>/--------          -----           ----      ----/
20 2D 2D 2D 2D 2D 20 20 20 20 20 20  0055
2D 2D 2D 20 20 20 20 20 20 20 20 20  0061
      2D 2D 2D 2D 20 20 20 20 20 2D  006D
                                0000005C  0077    15 HEADER_LENGTH=.-HEADER
                                        0077    16 PROT_ON_OFF:
                                   00  0077    17          .BYTE   0
                                        0078    18 PERM_CEB:
                                   00  0078    19          .BYTE   0
                                        0079    20 UIC_OWNER:
                                0000  0079    21          .WORD   0
                                        007B    22 UIC_GROUP:
                                0000  007B    23          .WORD   0
                                   00  007D    24 DONE:    .BYTE   0
                                        007E    25          .SAVE_PSECT
                                00000000  0026    26          .PSECT  SCRATCH NOSHR,WRT,NOEXE
                                        0000    27 GCI_ARGS:
                                00000002  0000    28          .LONG   2
                                00000000  0004    29          .LONG   0               ;0->FIRST TIME THROUGH AND WILD CARD, CEB PTR
                                00000000  0008    30          .LONG   0
                                0000007E  0031    31          .RESTORE_PSECT
                                        007E    32 TERM_CHANNEL:
                                0000  007E    33          .WORD   0
                                        0080    34 TERMINAL:
55 4F 24 53 59 53 00000088'010E0000'  0080    35          .ASCID  /SYS$OUTPUT/
                        54 55 50 54  008E
                                00000000  0036    36          .PSECT  CODE,EXE,NOWRT,SHR
                                0000  0000    37          .ENTRY  LOC_CEB,^M< >
                                        0002    38          $ASSIGN_S       CHAN=TERM_CHANNEL,DEVNAM=TERMINAL
                                        0017    39          $QIOW_S         CHAN=TERM_CHANNEL,FUNC=#IO$_WRITEVBLK,P1=HEADER,-
                                        0017    40                          P2=#HEADER_LENGTH
            0000007D'EF   01    90  0040    41          MOVB            #1,DONE                 ;Note that we have not finished
                                        0047    42 10$:
                                        0047    43          $CMKRNL_S       ROUTIN=GET_CEB_INFO,ARGLST=GCI_ARGS     ;Get into kernal for
ceb's
                                        005A    44 ;Format and display ceb info
              2B 0000007D'EF   E9    005A    45          BLBC    DONE,20$
                                        0061    46          $OUTPUT         CHAN=TERM_CHANNEL,BUFFER=CEB_NAME,-
                                        0061    47                          LENGTH=CEB_NAME_LEN
                                   BB   11  008A    48          BRB     10$
                                        04  008C    49 20$:     RET
                                   003C  008D    50          .ENTRY  GET_CEB_INFO,^M<R2,R3,R4,R5>
```

```
                50    00000000'GF   DE  008F    51          MOVAL   G^EXE$GL_CEBMTX,R0       ;Set up mutex on ceb structures
                      00000000'GF   16  0096    52          JSB     G^SCH$LOCKR              ;Lock for read access (assume R4->PCB)
                            04 AC   D5  009C    53          TSTL    4(AP)                    ;B
                               08   12  009F    54          BNEQ    GOT_CURRENT
            04 AC     00000000'GF   D0  00A1    55          MOVL    G^SCH$GQ_CEBHD,4(AP)     ;4(AP)-> TO NEXT CEB
                                    00A9    56 GOT_CURRENT:
                      52    04 AC   D0  00A9    57          MOVL    4(AP),R2
                   04 AC    04 BC   D0  00AD    58          MOVL    @4(AP),4(AP)
            00000004'GF     04 AC   D1  00B2    59          CMPL    4(AP),G^SCH$GQ_CEBHD+4   ;POINTS TO LAST CEB?
                               08   12  00BA    60          BNEQ    MORE_CEBS
                   0000007D'EF      94  00BC    61          CLRB    DONE
                               15   11  00C2    62          BRB     NO_MORE_CEBS
                                    00C4    63 MORE_CEBS:
            00000008'EF    28 A2    9A  00C4    64          MOVZBL  CEB$T_EFCNAM(R2),CEB_NAME_LEN
         29 A2      00000008'EF    28  00CC    65          MOVC3   CEB_NAME_LEN,CEB$T_EFCNAM+1(R2),CEB_NAME
                    0000000C'EF        00D4
                                    00D9    66 NO_MORE_CEBS:
                50    00000000'GF   DE  00D9    67          MOVAL   G^EXE$GL_CEBMTX,R0
                54    00000000'GF   D0  00E0    68          MOVL    G^SCH$GL_CURPCB,R4
                      00000000'GF   16  00E7    69          JSB     G^SCH$UNLOCK             ;Free up mutex on ceb
                                    04  00ED    70          RET
                                        00EE    71          .END    LOC_CEB
```

```
$$T1            =  00000001
CEB$B_CREATPORT    0000001E
CEB$B_DELETPORT    0000001F
CEB$B_LOCK         0000001C
CEB$B_PROCCNT      0000001D
CEB$B_STS          0000000B
CEB$B_TYPE         0000000A
CEB$C_LENGTH       00000038
CEB$C_SLAVLNG      00000044
CEB$K_LENGTH       00000038
CEB$K_SLAVLNG      00000044
CEB$L_CEBBL        00000004
CEB$L_CEBFL        00000000
CEB$L_EFC          00000010
CEB$L_MASTER       00000040
CEB$L_PID          0000000C
CEB$L_SHB          00000038
CEB$L_UIC          00000020
CEB$L_VASLAVE1     00000038
CEB$L_WQBL-        00000018
CEB$L_WQFL         00000014
CEB$T_EFCNAM       00000028
CEB$W_GRP          00000022
CEB$W_INDX         0000003C
CEB$W_PROT         00000024
CEB$W_REFC         00000026
CEB$W_SIZE         00000008
CEB$W_STATE        0000001E
CEB$W_WQCNT        0000001C
CEB_DESC           00000000 R      03
CEB_NAME           0000000C R      03
CEB_NAME_LEN       00000008 R      03
DONE               0000007D R      03
EXE$GL_CEBMTX      ********   X    05
GCI_ARGS           00000000 R      04
GET_CEB_INFO       0000008D RG     05
GOT_CURRENT        000000A9 R      05
HEADER             0000001B R      03
HEADER_LENGTH   =  0000005C
IO$_WRITEVBLK   =  00000030
LOC_CEB            00000000 RG     05
MORE_CEBS          000000C4 R      05
NO_MORE_CEBS       000000D9 R      05
PERM_CEB           00000078 R      03
PROT_ON_OFF        00000077 R      03
SCH$GL_CURPCB      ********   X    05
SCH$GQ_CEBHD       ********   X    05
SCH$LOCKR          ********   X    05
SCH$UNLOCK         ********   X    05
SYS$ASSIGN         ********  GX    05
SYS$CMKRNL         ********  GX    05
SYS$QIOW           ********  GX    05
TERMINAL           00000080 R      03
TERM_CHANNEL       0000007E R      03
UIC_GROUP          0000007B R      03
UIC_OWNER          00000079 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000044 (68.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000092 (146.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
SCRATCH	0000000C (12.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
CODE	000000EE (238.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.08	00:00:00.18
Command processing	39	00:00:00.34	00:00:00.72
Pass 1	261	00:00:06.10	00:00:06.99
Symbol table sort	1	00:00:00.84	00:00:00.85
Pass 2	65	00:00:00.99	00:00:01.09
Symbol table output	7	00:00:00.10	00:00:00.10
Psect synopsis output	5	00:00:00.05	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	410	00:00:08.54	00:00:10.01

The working set limit was 430 pages.
16560 bytes (33 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 334 non-local and 2 local symbols.
71 source lines were read in Pass 1, producing 24 object records in Pass 2.
17 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	12
TOTALS (all libraries)	13

497 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

```
/LIS/ENABL=SUPP CEB+SYS$LIBRARY:LIB/LIB
$
$
```

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
.MAIN.	0	396	SYS$SYSDEVICE:[ELLIS]CEB.OBJ;2	4-DEC-1982 19:11	VAX-11 Macro V03-00
SYS	V03-003	0	SYS$SYSROOT:[SYSEXE]SYS.STB;1	27-APR-1982 03:46	VAX-11 Linker V03-16

Program Section Synopsis

Psect Name	Module Name	Base	End	Length	Align	Attributes
DATA		00000200	00000291	00000092 (146.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	.MAIN.	00000200	00000291	00000092 (146.)	BYTE 0	
SCRATCH		00000292	0000029D	0000000C (12.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
	.MAIN.	00000292	0000029D	0000000C (12.)	BYTE 0	
CODE		00000400	000004ED	000000EE (238.)	BYTE 0	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC
	.MAIN.	00000400	000004ED	000000EE (238.)	BYTE 0	

Symbols By Name

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
EXE$GL_CEBMTX	800028C4						
GET_CEB_INFO	0000048D-R						
LOC_CEB	00000400-R						
SCH$GL_CURPCB	8000210C						
SCH$GQ_CEBHD	80002124						
SCH$LOCKR	80009802						
SCH$UNLOCK	80009870						
SYS$ASSIGN	7FFEDE50						
SYS$CMKRNL	7FFEDE90						
SYS$K_VERSION	594A3158						
SYS$QIOW	7FFEDE00						

Key for special characters above:

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

```
                                +-----------------+
                                ! Image Synopsis !
                                +-----------------+

Virtual memory allocated:                00000200 000007FF 00000600 (1536. bytes, 3. pages)
Stack size:                                    20. pages
Image header virtual block limits:              1.        1. (    1. block)
Image binary virtual block limits:              2.        3. (    2. blocks)
Image name and identification:           CEB 0
Number of files:                                4.
Number of modules:                              3.
Number of program sections:                     8.
Number of global symbols:                      11.
Number of image sections:                       4.
User transfer address:                   00000400
Debugger transfer address:               7FFEDF68
Image type:                              EXECUTABLE.
Map format:                              DEFAULT in file SYS$SYSDEVICE:[ELLIS]CEB.MAP;1
Estimated map length:                    15. blocks
                              +---------------------+
                              ! Link Run Statistics !
                              +---------------------+

Performance Indicators                   Page Faults   CPU Time        Elapsed Time
----------------------                   -----------   --------        ------------
    Command processing:                          25   00:00:00.09     00:00:00.15
    Pass 1:                                     125   00:00:01.54     00:00:02.38
    Allocation/Relocation:                       25   00:00:00.10     00:00:00.22
    Pass 2:                                      55   00:00:00.29     00:00:00.77
    Map data after object module synopsis:       24   00:00:00.14     00:00:00.13
    Symbol table output:                          3   00:00:00.01     00:00:00.16
Total run values:                               257   00:00:02.17     00:00:03.81

Using a working set limited to 415 pages and 33 pages of data storage (excluding image)

Total number object records read (both passes):   190
    of which 18 were in libraries and 2 were DEBUG data records containing 139 bytes
123 bytes of DEBUG data were written,starting at VBN 5 with 1 blocks allocated

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

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

/MAP CEB,SYS$SYSTEM:SYS.STB/SELE
$
```

```
$ analyze/system
VAX/VMS System analyzer

SDA> format^U
SDA> read cebdef.obj
SDA> format @sch$gq_cebhd
801153C0    CEB$L_CEBFL             80002124
801153C4    CEB$L_CEBBL             80002124
801153C8    CEB$W_SIZE                  0040
801153CA    CEB$B_TYPE                08
801153CB    CEB$B_STS               02
801153CC    CEB$L_PID               00020045
801153D0    CEB$L_EFC               00000000
801153D4    CEB$L_WQFL              801153D4
801153D8    CEB$L_WQBL              801153D4
801153DC    CEB$B_LOCK                    00
            CEB$W_WQCNT
801153DD    CEB$B_PROCCNT               00
801153DE    CEB$B_CREATPORT           03
            CEB$W_STATE
801153DF    CEB$B_DELETPORT         00
801153E0    CEB$L_UIC               00010006
            CEB$W_GRP
801153E4    CEB$W_PROT                  0000
801153E6    CEB$W_REFC              0000
801153E8    CEB$T_EFCNAM            'BERT'
801153ED                            000000
801153F0                            00000000
801153F4                            00000000
801153F8    CEB$L_SHB               00010004
            CEB$L_VASLAVE1
801153FC    CEB$W_INDX                  A000
801153FE                            0000
80115400    CEB$L_MASTER            0000008F

SDA> ^Z
$
```

— or — mcR SDA
>* Dump file

Exam System Symbols only

> Show Process

> EX PHS$GL_Faults

> Format address /Type = PCB ← data structure

> Read SYS$SYSTEM:SYSDEF.STB
(has all of $XXXX symbols)

> Show Process /Index = PID (last 4 digits)

```
$
$
$
$ r ceb
CEB_NAME        OWNER           PROT     PERM
--------        -----           ----     ----
BERT
$
```

```
                     0000   100         .TITLE  MAKETQE -- Inserts TQE into timer queue
                     0000   200         .IDENT  /V01/
                     0000   300 ;++
                     0000   400 ;
                     0000   500 ; ABSTRACT:
                     0000   600 ;
                     0000   700 ;       This program places a segment of code into nonpaged pool,
                     0000   800 ;       and then establishes a TQE which invokes that routine
                     0000   900 ;       every tenth of a second.
                     0000  1000 ;
                     0000  1100 ; SIDE EFFECTS:
                     0000  1200 ;
                     0000  1300 ;       Non-paged pool is used to hold the TQE, and the code that
                     0000  1400 ;       executes.
                     0000  1500 ;
                     0000  1600 ; PROGRAMMER:
                     0000  1700 ;
                     0000  1800 ;       Vik Muiznieks   15-MAY-1980
                     0000  1900 ;
                     0000  2000 ;--
                     0000  2100 ;
                     0000  2200 ;       External symbols
                     0000  2300 ;
                     0000  2400         $IPLDEF                                 ; IPL definitions
                     0000  2500         $TQEDEF                                 ; TQE definitions
                     0000  2600 ;
                     0000  2700 ;       Local symbols
                     0000  2800 ;
0000000C             0000  2900 HEADER = 12                                     ; size of header
00000078             0000  3000 DYN_C_MY_TYPE = 120                             ; my block type
                     0000  3100                                                 ; NOTE: Type must be < 128
                     0000  3200                                                 ; else EXESDEANONPAGED will
                     0000  3300                                                 ; bugcheck (in STOPTQE).
                     0000  3400 ;
                     0000  3500 ;       Local storage
                     0000  3600 ;
000F4240             0000  3700 DELTA:  .LONG   10000*100                       ; delta repeat time
00000000             0004  3800         .LONG   0                               ; of .1 seconds
                     0008  3900 ;
                     0008  4000 ;       Program entry point
                     0008  4100 ;
                0000 0008  4200 START:  .WORD   0                              ; null entry mask
                     000A  4300         $CMKRNL_S  ROUTIN=10$                   ; enter kernel mode
                  04 0019  4400         RET                                     ; all done
                     001A  4500
                003C 001A  4600 10$:    .WORD   ^M<R2,R3,R4,R5>                 ; save registers used
                     001C  4700         .ENABL  LSB                             ; enable local symbol block
                     001C  4800 ;
                     001C  4900 ;       Allocate pool to hold code.  Code must be placed in system
                     001C  5000 ;       space so that it can execute in ANY process context.  HEADER extra
                     001C  5100 ;       bytes will be allocated for a header (since the code block may
                     001C  5200 ;       later be deleted by running program STOPTQE).  The program will
                     001C  5300 ;       use the first word in the third longword to store the size of
                     001C  5400 ;       the block.  Normally the system uses the first two longwords
                     001C  5500 ;       for forward and backward links.  In this case, the first
                     001C  5600 ;       longword will be incremented each time the routine specified
                     001C  5700 ;       by the TQE executes.  The second longword will not be used.
```

```
                                001C  5800 ;       Note that IPL is raised to IPL$_ASTDEL before the block of pool
                                001C  5900 ;       is allocated.  This is done so that the process can not be
                                001C  6000 ;       deleted while it has the address of the block in a register
                                001C  6100 ;       (and no other record of the block is maintained elsewhere in
                                001C  6200 ;       the system).
                                001C  6300 ;
          51   00000017'8F    D0  001C  6400       MOVL    #COPY_LEN+HEADER,R1              ; size of pool needed
                               0023  6500       SETIPL  #IPL$_ASTDEL             ; so process not deleted
               00000000'GF     16  0026  6600       JSB     G^EXE$ALONONPAGED        ; allocate pool
                               002C  6700 ;
                               002C  6800 ;       The above routine destroys R0-R3, and returns in R2 the
                               002C  6900 ;       address of the allocated block of pool.
                               002C  7000 ;
                    09 50      E8  002C  7100       BLBS    R0,20$                   ; proceed if no error
                               002F  7200       SETIPL  #0                       ; lower IPL before exiting
               50    0000'8F   3C  0032  7300       MOVZWL  #SS$_INSFMEM,R0          ; indicate error
                               04  0037  7400       RET                              ; return error code
          0000005D'EF    52    D0  0038  7500 20$:    MOVL    R2,UPDATE                ; save address of block
                         82    7C  003F  7600       CLRQ    (R2)+                    ; clear location to be updated
                               0041  7700                                            ; point R2 to 3rd longword
                      82   51    B0  0041  7800       MOVW    R1,(R2)+                 ; fill in size field
                   82    78 8F    9B  0044  7900       MOVZBW  #DYN_C_MY_TYPE,(R2)+     ; fill in type field and
                               0048  8000                                            ; point R2 to start of code
                         52    DD  0048  8100       PUSHL   R2                       ; save address of code
     62 00000056'EF    000B'8F   28  004A  8200       MOVC3   #COPY_LEN,COPY_START,(R2)        ; copy code to buffer
                               0054  8300                                            ; NOTE -- R0-R5 altered
                         08    11  0054  8400       BRB     30$                      ; continue
                               0056  8500 ;
                               0056  8600 ;       This is the code that executes every .1 seconds in response to
                               0056  8700 ;       the TQE.  The timer interrupt service routine transfers control
                               0056  8800 ;       to the code with a JSB instruction at IPL$_TIMER (7).  Note that
                               0056  8900 ;       the code must be PIC (position independent) since it is being COPIED
                               005b  9000 ;       to the system buffer (and executes at arbitrary system addresses).
                               0056  9100 ;
                               0056  9200 COPY_START:                                ; start of code to be
                               005b  9300                                            ; copied into pool
               0000005D'FF     D6  0056  9400       INCL    @UPDATE                  ; This is where the
                               005C  9500                                            ; routine could do
                               005C  9600                                            ; useful work
                               05  005C  9700       RSB                              ; return control to
                               005D  9800                                            ; timer interrupt
                               005D  9900                                            ; service routine
                   00000000    005D 10000 UPDATE:  .LONG   0                        ; will hold address of
                               0061 10100                                            ; location to be incremented
                   0000000B    0061 10200 COPY_LEN = . - COPY_START                  ; size of copied code
                               0061 10300 ;
                               0061 10400 ;       Allocate a TQE.  Note that the routine allocates the TQE at
                               0061 10500 ;       IPL$_SYNCH, but returns control at IPL$_ASTDEL (so process
                               0061 10600 ;       cannot be deleted before it can deallocate pool used for TQE).
                               0061 10700 ;       The routine destroys R0-R4, and returns the address of the TQE
                               0061 10800 ;       block in R2.
                               0061 10900 ;
               00000000'GF     16  0061 11000 30$:    JSB     G^EXE$ALLOCTQE           ; allocate TQE block
                    13 50      E8  0067 11100       BLBS    R0,40$                   ; continue if no error
                   50    8E    D0  006A 11200       MOVL    (SP)+,R0                 ; else, get code address
                               006D 11300                                            ; and clean up stack
                   50    0C    C2  006D 11400       SUBL    #HEADER,R0               ; account for header
```

```
                00000000'GF   16  0070 11500          JSB     G^EXE$DEANONPAGED               ; deallocate code block
             50    0000'8F   3C  0076 11600          MOVZWL  #SS$_NOSLOT,R0                  ; return error code
                       31   11  0078 11700          BRB     50$                             ; and exit
                            007D 11800 ;
                            007D 11900 ;        Initialize TQE and insert TQE into queue (using system routine).
                            007D 12000 ;        The routine expects the TQE address in R5.  It copies the
                            007D 12100 ;        due time into the TQE, and inserts the TQE in the queue at
                            007D 12200 ;        the appropriate point.  Since the current time is passed
                            007D 12300 ;        (in R0 and R1) as the due time, the TQE should be placed
                            007D 12400 ;        at the head of the queue, and delivered after the next
                            007D 12500 ;        timer interrupt.
                            007D 12600 ;
                            007D 12700 ;        The address of the TQE is also stored in a global location
                            007D 12800 ;        in the executive reserved for site-specific use.
                            007D 12900 ;
                 0B A2    05   90  007D 13000 40$:    MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R2)  ; indicate system sub.
                                  0081 13100                                                  ; and repeat request
              20 A2    FF7B CF   7D  0081 13200          MOVQ    DELTA,TQE$Q_DELTA(R2)           ; set repeat time=.1 sec
                 0C A2    8E   D0  0087 13300          MOVL    (SP)+,TQE$L_FPC(R2)             ; starting address of code;
                                  0088 13400                                                  ; also cleans up stack
             00000000'GF   52   D0  008B 13500          MOVL    R2,G^EXE$GL_SITESPEC            ; save TQE address for
                                  0092 13600                                                  ; program that will
                                  0092 13700                                                  ; cancel TQE request
                                  0092 13800          ASSUME  IPL$_SYNCH EQ IPL$_TIMER
                                  0092 13900 LOCK_START:
                                  0092 14000
                                  0092 14100          SETIPL  #SYNCH                          ; accessing system data base
             50    00000000'GF   7D  0099 14200          MOVQ    G^EXE$GQ_SYSTIME,R0             ; get current abs. time
                      55   52   D0  00A0 14300          MOVL    R2,R5                           ; copy TQE address for
                00000000'GF   16  00A3 14400          JSB     G^EXE$INSTIMQ                   ; queuing routine
             50    0000'8F   3C  00A9 14500          MOVZWL  #SS$_NORMAL,R0                  ; set success status
                                  00AC 14600 50$:     SETIPL  #0                              ; lower IPL
                            04    00B1 14700          RET                                     ; all done
                                  00B2 14800          .DSABL  LSB                             ; disable local symbol block
                                  00B2 14900 ;
                                  00B2 15000 ;        By placing the SYNCH label after the code that must execute
                                  00B2 15100 ;        at IPL$_SYNCH, the page with the SETIPL SYNCH instruction and
                                  00B2 15200 ;        the page with the SYNCH label are guaranteed to be in the
                                  00B2 15300 ;        process's working set.  Since the code will not span more
                                  00B2 15400 ;        than 2 pages, there is no way to have a page fault above IPL 2,
                                  00B2 15500 ;        even though the pages have not been locked into the working
                                  00B2 15600 ;        set (with the $LKWSET system service).
                                  00B2 15700 ;
                       00000007   00B2 15800 SYNCH:   .LONG   IPL$_SYNCH
                                  00B6 15900 LOCK_END:
                                  00B6 16000          ASSUME LOCK_END-LOCK_START LE 512
                                  00B6 16100
                                  00B6 16200          .END    START
```

```
$IT...          = 00000003
COPY_LEN        = 00000008
COPY_START        00000056 R      01
DELTA             00000000 R      01
DYN_C_MY_TYPE   = 00000078
EXE$ALLOCTQE      ********    X   01
EXE$ALONONPAGED   ********    X   01
EXE$DEANONPAGED   ********    X   01
EXE$GL_SITESPEC   ********    X   01
EXE$GQ_SYSTIME    ********    X   01
EXE$INSTIMQ       ********    X   01
GBL...          = 00000000
HEADER          = 0000000C
IPL$_ASTDEL     = 00000002
IPL$_HWCLK      = 00000018
IPL$_IOPOST     = 00000004
IPL$_MAILBOX    = 0000000B
IPL$_POWER      = 0000001F
IPL$_QUEUEAST   = 00000006
IPL$_SCHED      = 00000003
IPL$_SYNCH      = 00000007
IPL$_TIMER      = 00000007
LOCK_END          000000B6 R      01
LOCK_START        00000092 R      01
PR$_IPL           ********    X   01
SS$_INSFMEM       ********    X   01
SS$_NORMAL        ********    X   01
SS$_NOSLOT        ********    X   01
START             00000008 R      01
SYNCH             000000B2 R      01
SYS$CMKRNL        ********  GX    01
TQE$B_EFN         00000029
TQE$B_RMOD        00000028
TQE$B_RQTYPE      00000008
TQE$B_TYPE        0000000A
TQE$C_LENGTH      00000030
TQE$C_SSREPT    = 00000005
TQE$C_SSSNGL    = 00000001
TQE$C_TMSNGL    = 00000000
TQE$C_WKREPT    = 00000006
TQE$C_WKSNGL    = 00000002
TQE$K_LENGTH      00000030
TQE$L_AST         00000010
TQE$L_ASTPRM      00000014
TQE$L_FPC         0000000C
TQE$L_FR3         00000010
TQE$L_FR4         00000014
TQE$L_PID         0000000C
TQE$L_RQPID       0000002C
TQE$L_TQBL        00000004
TQE$L_TQFL        00000000
TQE$M_REPEAT    = 00000004
TQE$Q_DELTA       00000020
TQE$Q_TIME        00000018
TQE$V_REPEAT    = 00000002
TQE$W_SIZE        00000008
UPDATE            0000005D R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000086 (182.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000030 (48.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.08	00:00:00.49
Command processing	45	00:00:00.26	00:00:00.92
Pass 1	1390	00:00:02.83	00:00:07.83
Symbol table sort	5	00:00:00.04	00:00:00.07
Pass 2	134	00:00:00.57	00:00:01.35
Symbol table output	9	00:00:00.06	00:00:00.06
Psect synopsis output	12	00:00:00.04	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1628	00:00:03.88	00:00:10.93

The working set limit was 150 pages.
6992 bytes (14 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 57 non-local and 5 local symbols.
162 source lines were read in Pass 1, producing 12 object records in Pass 2.
15 pages of virtual memory were used to define 15 macros.

Macro library statistics

Macro library name	Macros defined
_DRA1:[SYSLIB]LIB.MLB;1	15
_DRA1:[SYSLIB]STARLET.MLB;1	0
TOTALS (all libraries)	15

184 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

/LIS MAKETQE+SYS$LIBRARY:LIB/LIB

```
                        ;0176   259          .SBTTL  INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                        ;0176   260 ;+
                        ;0176   261 ; EXESINSTIMQ - INSERT ENTRY IN TIME DEPENDENT SCHEDULER QUEUE
                        ;0176   262 ;
                        ;0176   263 ; THIS ROUTINE IS CALLED TO INSERT AN ENTRY IN THE TIME DEPENDENT SCHEDULER
                        ;0176   264 ; QUEUE. THE ENTRY IS THREADED INTO THE QUEUE ACCORDING TO ITS DUE TIME.
                        ;0176   265 ; THE QUEUE IS ORDERED SUCH THAT THE MOST IMMINENT ENTRIES ARE AT THE FRONT
                        ;0176   266 ; OF THE QUEUE.
                        ;0176   267 ;
                        ;0176   268 ; INPUTS:
                        ;0176   269 ;
                        ;017A   270 ;       R0 = LOW ORDER PART OF EXPIRATION TIME.
                        ;0176   271 ;       R1 = HIGH ORDER PART OF EXPIRATION TIME.
                        ;0176   272 ;       R5 = ADDRESS OF ENTRY TO INSERT IN TIME QUEUE.
                        ;0176   273 ;
                        ;0176   274 ;       IPL MUST BE IPL$_TIMER OR ABOVE.
                        ;0176   275 ;
                        ;0176   276 ; OUTPUTS:
                        ;0176   277 ;
                        ;0176   278 ;       SPECIFIED ENTRY IS INSERTED INTO THE TIME DEPENDENT SCHEDULER QUEUE
                        ;0176   279 ;       ACCORDING TO ITS DUE TIME.
                        ;0176   280 ;-
                        ;0176   281
                        ;0000   282          .PSECT
                        ;0000   283 EXESINSTIMQ::                               ;INSERT ENTRY IN TIME QUEUE
  10 A5   50   7D       0002    284          MOVQ    R0,TQESQ_TIME(R5)          ;SET ABSOLUTE DUE TIME
53   0000'CF   DE       0006    285          MOVAL   W^EXESGL_TQFL,R3           ;GET ADDRESS OF TIME QUEUE LISTHEAD
     52   53   D0       0009    286          MOVL    R3,R2                      ;COPY ADDRESS OF TIME QUEUE LISTHEAD
  52   04 A2   D0       000C    287 10$:     MOVL    TQESL_TQBL(R2),R2          ;GET ADDRESS OF NEXT ENTRY
     52   53   D1       0010    288          CMPL    R3,R2                      ;END OF QUEUE?
          0E   13       0013    289          BEQL    20$                        ;IF EQL YES
  1C A2   51   D1       0015    290          CMPL    R1,TQESQ_TIME+4(R2)        ;COMPARE HIGH ORDER PARTS OF TIME
          F1   1F       0019    291          BLSSU   10$                        ;IF LSSU NEW ENTRY MORE IMMINENT
          06   1A       001B    292          BGTRU   20$                        ;IF GTRU NEW ENTRY LESS IMMINENT
  18 A2   50   D1       001D    293          CMPL    R0,TQESQ_TIME(R2)          ;COMPARE LOW ORDER PART OF TIME
          E9   1F       0021    294          BLSSU   10$                        ;IF LSSU NEW ENTRY MORE IMMINENT
     62   65   0E       0023    295 20$:     INSQUE  TQESL_TQFL(R5),TQESL_TQFL(R2) ;INSERT NEW ENTRY IN TIME QUEUE
                        0026    296          RSB                                ;
```

```
                                0000   100          .TITLE  STOPTQE -- Removes TQE from timer queue
                                0000   200          .IDENT  /V01/
                                0000   300 ;++
                                0000   400 ;
                                0000   500 ; ABSTRACT:
                                0000   600 ;
                                0000   700 ;       This program displays the contents of the location being updated
                                0000   800 ;       by the routine specified in a TQE (thrice).  It then cancels the
                                0000   900 ;       TQE request, and deallocates the block of pool being used to
                                0000  1000 ;       contain the TQE routine.
                                0000  1100 ;
                                0000  1200 ; SIDE EFFECTS:
                                0000  1300 ;
                                0000  1400 ;       Non-paged pool is returned to the system.
                                0000  1500 ;
                                0000  1600 ; PROGRAMMER:
                                0000  1700 ;
                                0000  1800 ;       Vik Muiznieks   15-MAY-1980
                                0000  1900 ;
                                0000  2000 ;--
                                0000  2100 ;
                                0000  2200 ;       External symbols
                                0000  2300 ;
                                0000  2400          $IPLDEF                                 ; IPL definitions
                                0000  2500          $TQEDEF                                 ; TQE definitions
                                0000  2600 ;
                                0000  2700 ;       Local symbols
                                0000  2800 ;
                         0000000C  0000  2900 HEADER = 12                                   ; header size for code block
                         00000003  0000  3000 LOOP_CNT = 3                                  ; loop counter
                                0000  3100 ;
                                0000  3200 ;       Local storage
                                0000  3300 ;
                         000001A3' 0000  3400 LKWSET: .LONG   START_LOCK                    ; starting address
                         000001C1' 0004  3500         .LONG   END_LOCK                      ; ending address
                              0000 0008  3600 TTCHAN: .WORD                                 ; TT channel
              54 54 00000012'010E0000' 000A  3700 TT:     .ASCID  /TT/                      ; descriptor for terminal
                         00000014' 0014  3800 CTR:    .LONG   STR_END - STRING              ; $FAO control string
                         00000042' 0018  3900         .LONG   STRING                        ; descriptor
                         0000001E' 001C  4000 CTR1:   .LONG   STR1_END - STR                ; $FAO control string
                         00000024' 0020  4100         .LONG   STR                           ; descriptor
45 58 45 20 6E 69 20 65 75 6C 61 56  0024  4200 STR:    .ASCII  *Value in EXE$GL_SITESPEC = !XL*; converts to hexadecimal
43 45 50 53 45 54 49 53 5F 4C 47 24  0030
                  4C 58 21 20 3D 20  003C
                                0042  4300 STR1_END:
65 69 66 20 6E 69 20 65 75 6C 61 56  0042  4400 STRING: .ASCII  *Value in field = !XL*         ; converts to hexadecimal
            4C 58 21 20 3D 20 64 6C  004E
                                0056  4500 STR_END:
                         00000000  0056  4600 FAOLEN: .LONG                                 ; $FAO output length
                         00000023  005A  4700 OUT:    .LONG   35                            ; Output string desc.
                         00000062  005E  4800         .LONG   BUFF
                         00000085  0062  4900 BUFF:   .BLKB   35                            ; Actual output string
                                0085  5000 BAD_MESSAGE:                                    ; used in case MAKETQE
67 6F 72 70 20 45 51 54 45 4B 41 4D  0085  5100         .ASCII  /MAKETQE program has not been run./ ; not yet run
20 74 6F 6E 20 73 61 68 20 6D 61 72  0091
          2E 6E 75 72 20 6E 65 65 62  009D
                         00000021  00A6  5200 BAD_SIZE = . - BAD_MESSAGE
```

```
                        00A6  5300 ;
                        00A6  5400 ;       Entry point for routine
                        00A6  5500 ;
                  0000  00A6  5600 START:  .WORD   0                              ; null entry mask
                        00A6  5700         $CMKRNL_S       ROUTIN=10$           ; enter kernel mode
                        00B7  5800 ;       Note that most of the work being done in kernel mode by this
                        00B7  5900 ;       example really could be done in user mode.  There is not much
                        00B7  6000 ;       need to enter kernel mode before label START_LOCK.
                     04 00B7  6100         RET                                  ; all done
                  007C  00B8  6200 10$:    .WORD   ^M<R2,R3,R4,R5,R6>           ; save registers used
                        00BA  6300         $LKWSET_S INADR=LKWSET               ; lock pages in working set
               01 50  E8  00C9  6400         BLBS    R0,15$                    ; proceed on success
                     04  00CC  6500         RET                               ; stop on error
                  00CD  6600 15$:    $ASSIGN_S DEVNAM=TT,CHAN=TTCHAN           ; get channel to terminal
               36 50  E9  00DE  6700         BLBC    R0,25$                    ; exit on error
52    00000000'GF  D0  00E1  6800 20$:    MOVL    G^EXE$GL_SITESPEC,R2         ; get TQE address
                        00E8  6900                                              ; if negative, system address
                  30  19  00E8  7000         BLSS    30$                       ; stop if not negative
                        00EA  7100         $OUTPUT CHAN=TTCHAN,LENGTH=$BAD_SIZE,BUFFER=BAD_MESSAGE
                        010A  7200         $DASSGN_S CHAN=TTCHAN                ; deassign terminal channel
                     04 0116  7300         RET                                  ; all done
               00B9  31  0117  7400 25$:    BRW     ERROR                        ; solve BLBC byte displacement
            56   0C A2  D0  011A  7500 30$:    MOVL    TQE$L_FPC(R2),R6          ; get code address
               56    0C  C2  011E  7600         SUBL2   #HEADER,R6               ; point to update location
               54    03  9A  0121  7700         MOVZBL  #LOOP_CNT,R4             ; set loop count
                        0124  7800         $FAO_S  CTRSTR=CTR1,OUTLEN=FAOLEN,-   ; format EXE$GL_SITESPEC
                        0124  7900                 OUTBUF=OUT,P1=R2              ; for debugging
               DB 50  E9  0139  8000         BLBC    R0,25$                    ; test for errors
                        013C  8100         $OUTPUT CHAN=TTCHAN,LENGTH=FAOLEN,BUFFER=BUFF ; print value
               B5 50  E9  015F  8200         BLBC    R0,25$                    ; test for errors
                        0162  8300 40$:    $FAO_S  CTRSTR=CTR,OUTLEN=FAOLEN,-    ; format counter which
                        0162  8400                 OUTBUF=OUT,P1=(R6)            ; changes every .1 seconds
               9D 50  E9  0177  8500         BLBC    R0,25$                    ; check for error
                        017A  8600         $OUTPUT CHAN=TTCHAN,LENGTH=FAOLEN,BUFFER=BUFF ; display counter
               33 50  E9  019D  8700         BLBC    R0,ERROR                  ; check for error
               BF 54  F5  01A0  8800         SOBGTR  R4,40$                    ; loop a few times
                        01A3  8900 START_LOCK:                                  ; code must be locked in
                        01A3  9000                                              ; working set so no page
                        01A3  9100                                              ; faults above IPL 2
                        01A3  9200         SETIPL  #IPL$_SYNCH                  ; raise IPL to synch
            50    62  0F  01A6  9300         REMQUE  (R2),R0                   ; remove TQE from queue
      00000000'GF     16  01A9  9400         JSB     G^EXE$DEANONPAGED         ; deallocate TQE
            50    56  D0  01AF  9500         MOVL    R6,R0                     ; get address of code block
      00000000'GF     16  01B2  9600         JSB     G^EXE$DEANONPAGED         ; deallocate code block
      00000000'GF     D4  01B8  9700         CLRL    G^EXE$GL_SITESPEC         ; clean-up location so this
                        01BE  9800                                              ; program cannot be rerun
                        01BE  9900                                              ; until MAKETQE rerun
                        01BE  10000        SETIPL  #0                           ; enable interrupts
                        01C1  10100 END_LOCK:                                   ; end of locked down code
                        01C1  10200        $DASSGN_S CHAN=TTCHAN                ; deassign terminal channel
         50   0000'8F  3C  01CD  10300        MOVZWL  #SS$_NORMAL,R0            ; return success status
                     04 01D2  10400         RET                                  ; all done
               56    50  D0  01D3  10500 ERROR:  MOVL    R0,R6                    ; save exit status code
                        01D6  10600        $DASSGN_S CHAN=TTCHAN                ; deassign terminal channel
               50    56  D0  01E2  10700        MOVL    R6,R0                    ; restore exit status code
                     04 01E5  10800         RET                                  ; all done
                        01E6  10900         .END    START
```

```
$$T1            =  00000001
$$T2            =  00000004
BAD_MESSAGE        00000085 R      01
BAD_SIZE        =  00000021
BIT...          =  0000000D
BUFF               00000062 R      01
CTR                00000014 R      01
CTR1               0000001C R      01
END_LOCK           000001C1 R      01
ERROR              000001D3 R      01
EXE$DEANONPAGED    ********   X    01
EXE$GL_SITESPEC    ********   X    01
FAOLEN             00000056 R      01
GBL...          =  00000000
HEADER          =  0000000C
IO$M_ABORT      =  00000100
IO$M_ACCESS     =  00000040
IO$M_ATTNAST    =  00000100
IO$M_BINARY     =  00000040
IO$M_CANCTRLO   =  00000040
IO$M_CECYL      =  00000400
IO$M_COMMOD     =  00000040
IO$M_CREATE     =  00000080
IO$M_CTRLCAST   =  00000100
IO$M_CTRLYAST   =  00000080
IO$M_CVTLOW     =  00000100
IO$M_CYCLE      =  00001000
IO$M_DATACHECK  =  00004000
IO$M_DATAPATH   =  00000400
IO$M_DELDATA    =  00000040
IO$M_DELETE     =  00000100
IO$M_DIAGNOSTIC=  00000100
IO$M_DMOUNT     =  00000400
IO$M_DSABLMBX   =  00000400
IO$M_ENABLMBX   =  00000080
IO$M_FCODE      =  0000003F
IO$M_HANGUP     =  00000200
IO$M_INHERLOG   =  00000800
IO$M_INHEXTGAP  =  00001000
IO$M_INHRETRY   =  00008000
IO$M_INHSEEK    =  00001000
IO$M_INTERRUPT  =  00000040
IO$M_INTSKIP    =  00000100
IO$M_MOUNT      =  00000200
IO$M_MOVETRACKD=  00000080
IO$M_NOECHO     =  00000040
IO$M_NOFILTR    =  00000200
IO$M_NOFORMAT   =  00000100
IO$M_NOW        =  00000040
IO$M_NOWAIT     =  00000080
IO$M_OPPOSITE   =  00000200
IO$M_PACKED     =  00000080
IO$M_PURGE      =  00000800
IO$M_READATTN   =  00000080
IO$M_REFRESH    =  00002000
IO$M_RESET      =  00000080
IO$M_REVERSE    =  00000040
IO$M_SETEVF     =  00000040
IO$M_SETFNCT    =  00000200
IO$M_SHUTDOWN   =  00000080
IO$M_SKPSECINH  =  00000200
IO$M_STARTUP    =  00000040
IO$M_SWAP       =  00000400
IO$M_SYNCH      =  00000200
IO$M_TIMED      =  00000080
IO$M_TRMNOECHO  =  00001000
IO$M_TYPEAHDCNT=  00000040
IO$M_WORD       =  00000040
IO$M_WRTATTN    =  00000100
IO$S_FCODE      =  00000006
IO$V_ABORT      =  00000008
IO$V_ACCESS     =  00000006
IO$V_ATTNAST    =  00000008
IO$V_BINARY     =  00000006
IO$V_CANCTRLO   =  00000006
IO$V_CECYL      =  0000000A
IO$V_COMMOD     =  00000006
IO$V_CREATE     =  00000007
IO$V_CTRLCAST   =  00000008
IO$V_CTRLYAST   =  00000007
IO$V_CVTLOW     =  00000008
IO$V_CYCLE      =  0000000C
IO$V_DATACHECK  =  0000000E
IO$V_DATAPATH   =  0000000A
IO$V_DELDATA    =  00000006
IO$V_DELETE     =  00000008
IO$V_DIAGNOSTIC=  00000008
IO$V_DMOUNT     =  0000000A
IO$V_DSABLMBX   =  0000000A
IO$V_ENABLMBX   =  00000007
IO$V_FCODE      =  00000000
IO$V_HANGUP     =  00000009
IO$V_INHERLOG   =  00000008
IO$V_INHEXTGAP  =  0000000C
IO$V_INHRETRY   =  0000000F
IO$V_INHSEEK    =  0000000C
IO$V_INTERRUPT  =  00000006
IO$V_INTSKIP    =  00000008
IO$V_MOUNT      =  00000009
IO$V_MOVETRACKD=  00000007
IO$V_NOECHO     =  00000006
IO$V_NOFILTR    =  00000009
IO$V_NOFORMAT   =  00000008
IO$V_NOW        =  00000006
IO$V_NOWAIT     =  00000007
IO$V_OPPOSITE   =  00000009
IO$V_PACKED     =  00000007
IO$V_PURGE      =  00000008
IO$V_READATTN   =  00000007
IO$V_REFRESH    =  0000000D
IO$V_RESET      =  00000007
IO$V_REVERSE    =  00000006
IO$V_SETEVF     =  00000006
IO$V_SETFNCT    =  00000009
IO$V_SHUTDOWN   =  00000007
IO$V_SKPSECINH  =  00000009
IO$V_STARTUP    =  00000006
IO$V_SWAP       =  0000000A
IO$V_SYNCH      =  00000009
IO$V_TIMED      =  00000007
IO$V_TRMNOECHO  =  0000000C
IO$V_TYPEAHDCNT=  00000006
IO$V_WORD       =  00000006
IO$V_WRTATTN    =  00000008
IO$_ACCESS      =  00000032
IO$_ACPCONTROL  =  00000038
IO$_CLEAN       =  0000001E
IO$_CONINTREAD  =  0000003C
IO$_CONINTWRITE=  0000003D
IO$_CREATE      =  00000033
IO$_DEACCESS    =  00000034
IO$_DELETE      =  00000035
IO$_DIAGNOSE    =  0000001D
IO$_DRVCLR      =  00000004
IO$_ERASETAPE   =  00000006
IO$_FORMAT      =  0000001E
IO$_INITIALIZE  =  00000004
IO$_LOADMCODE   =  00000001
IO$_LOGICAL     =  0000002F
IO$_MODIFY      =  00000036
IO$_MOUNT       =  00000039
IO$_NOP         =  00000000
IO$_OFFSET      =  00000006
IO$_PACKACK     =  00000008
IO$_PHYSICAL    =  0000001F
IO$_QSTOP       =  00000007
IO$_READHEAD    =  0000000E
IO$_READLBLK    =  00000021
IO$_READPBLK    =  0000000C
IO$_READPRESET  =  00000019
IO$_READPROMPT  =  00000037
IO$_READTRACKD  =  00000010
IO$_READVBLK    =  00000031
IO$_RECAL       =  00000003
IO$_RELEASE     =  00000005
IO$_REREADN     =  00000016
IO$_REREADP     =  00000017
IO$_RETCENTER   =  00000007
IO$_REWIND      =  00000024
IO$_REWINDOFF   =  00000022
IO$_SEARCH      =  00000009
IO$_SEEK        =  00000002
IO$_SENSECHAR   =  0000001B
IO$_SENSEMODE   =  00000027
IO$_SETCHAR     =  0000001A
IO$_SETCLOCK    =  00000037
IO$_SETCLOCKP   =  00000005
IO$_SETMODE     =  00000023
IO$_SKIPFILE    =  00000025
IO$_SKIPRECORD  =  00000026
IO$_SPACEFILE   =  00000002
```

```
IO$_SPACERECORD= 00000009
IO$_STARTDATA  = 00000038
IO$_STARTDATAP = 00000006
IO$_STARTNPROC = 00000002
IO$_STARTSPNDL = 00000019
IO$_STOP       = 00000003
IO$_TTYREADALL = 0000003A
IO$_TTYREADPALL= 0000003B
IO$_UNLOAD     = 00000001
IO$_VIRTUAL    = 0000003F
IO$_WRITECHECK = 0000000A
IO$_WRITECHECKH= 00000018
IO$_WRITEHEAD  = 0000000D
IO$_WRITELBLK  = 00000020
IO$_WRITEMARK  = 0000001C
IO$_WRITEOF    = 00000028
IO$_WRITEPBLK  = 00000008
IO$_WRITERET   = 00000018
IO$_WRITETRACKD= 0000000F
IO$_WRITEVBLK  = 00000030
IO$_WRTTMKR    = 0000001D
IPL$_ASTDEL    = 00000002
IPL$_HWCLK     = 00000018
IPL$_IOPOST    = 00000004
IPL$_MAILBOX   = 0000000B
IPL$_POWER     = 0000001F
IPL$_QUEUEAST  = 00000006
IPL$_SCHED     = 00000003
IPL$_SYNCH     = 00000007
IPL$_TIMER     = 00000007
LKWSET           00000000 R      01
LOOP_CNT       = 00000003
OUT              0000005A R      01
PR$_IPL          ******** X     01
SS$_NORMAL       ******** X     01
START            000000A6 R      01
START_LOCK       000001A3 R      01
STR              00000024 R      01
STR1_END         00000042 R      01
STRING           00000042 R      01
STR_END          00000056 R      01
SYS$ASSIGN       ******** GX    01
SYS$CMKRNL       ******** GX    01
SYS$DASSGN       ******** GX    01
SYS$FAO          ******** GX    01
SYS$LKWSET       ******** GX    01
SYS$QIOW         ******** GX    01
TQE$B_EFN        00000029
TQE$B_RMOD       00000028
TQE$B_RQTYPE     0000000B
TQE$B_TYPE       0000000A
TQE$C_LENGTH     00000030
TQE$C_SSREPT   = 00000005
TQE$C_SSSNGL   = 00000001
TQE$C_TMSNGL   = 00000000
TQE$C_WKREPT   = 00000006
TQE$C_WKSNGL   = 00000002
TQE$K_LENGTH     00000030
TQE$L_AST        00000010
TQE$L_ASTPRM     00000014
TQE$L_FPC        0000000C
TQE$L_FR3        00000010
TQE$L_FR4        00000014
TQE$L_PID        0000000C
TQE$L_RQPID      0000002C
TQE$L_TQBL       00000004
TQE$L_TQFL       00000000
TQE$M_REPEAT   = 00000004
TQE$Q_DELTA      00000020
TQE$Q_TIME       00000018
TQE$V_REPEAT   = 00000002
TQE$W_SIZE       00000008
TT               0000000A R      01
TTCHAN           00000008 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	000001E6 (486.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000030 (48.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	22	00:00:00.03	00:00:00.16
Command processing	30	00:00:00.20	00:00:00.83
Pass 1	2099	00:00:09.76	00:00:11.79
Symbol table sort	14	00:00:00.29	00:00:00.31
Pass 2	247	00:00:01.49	00:00:02.08
Symbol table output	34	00:00:00.16	00:00:00.16
Psect synopsis output	4	00:00:00.03	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2461	00:00:11.97	00:00:15.39

The working set limit was 150 pages.
39571 bytes (78 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 245 non-local and 6 local symbols.
109 source lines were read in Pass 1, producing 13 object records in Pass 2.
22 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
_DRA1:[SYSLIB]LIB.MLB;1	21
_DRA1:[SYSLIB]STARLET.MLB;1	0
TOTALS (all libraries)	21

430 GETS were required to define 21 macros.

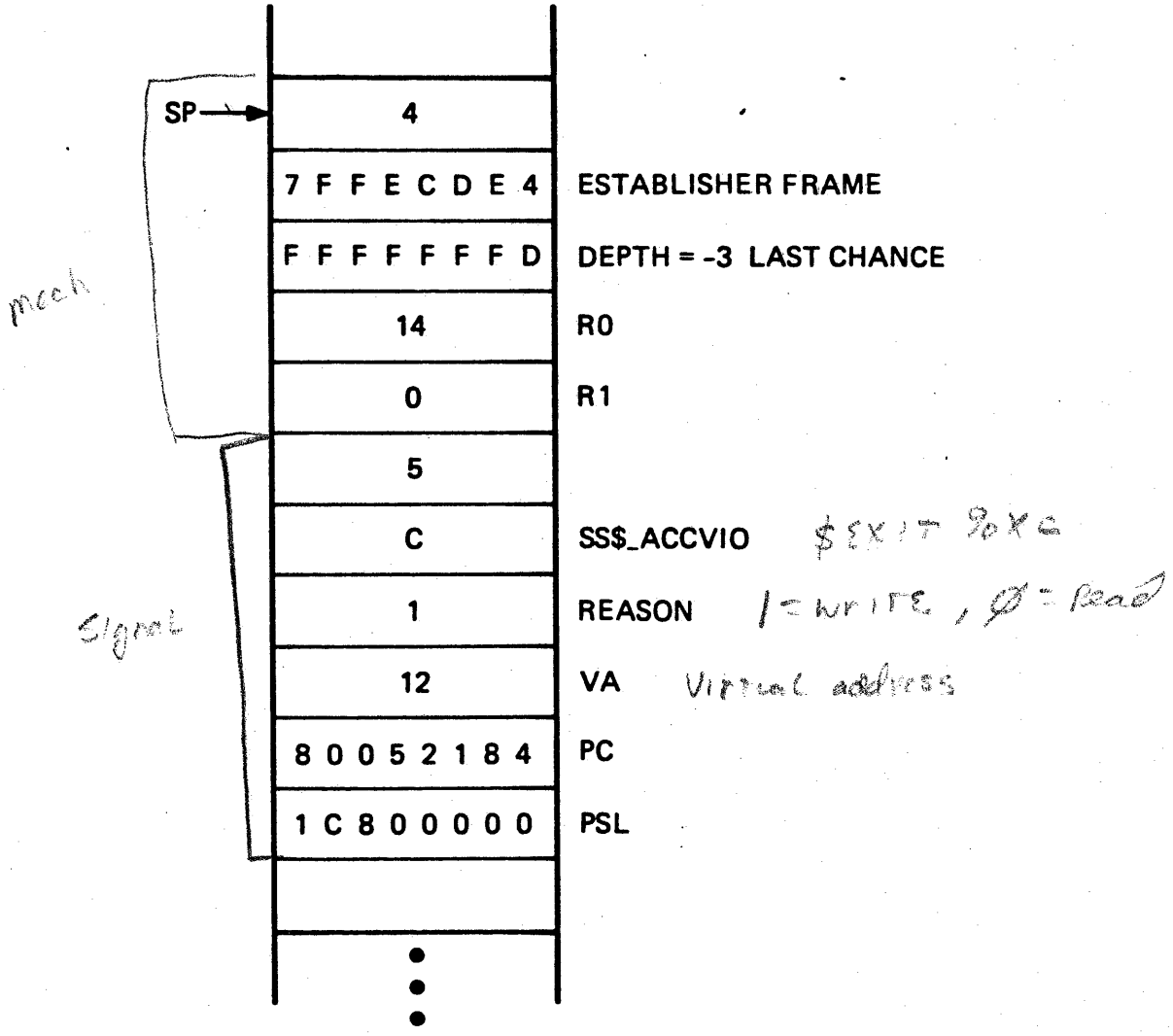
There were no errors, warnings or information messages.

/LIS STOPTQE+SYS$LIBRARY:LIB/LIB

DEBUGGING

SAMPLE STACKS AFTER BUGCHECKS

Access Violation

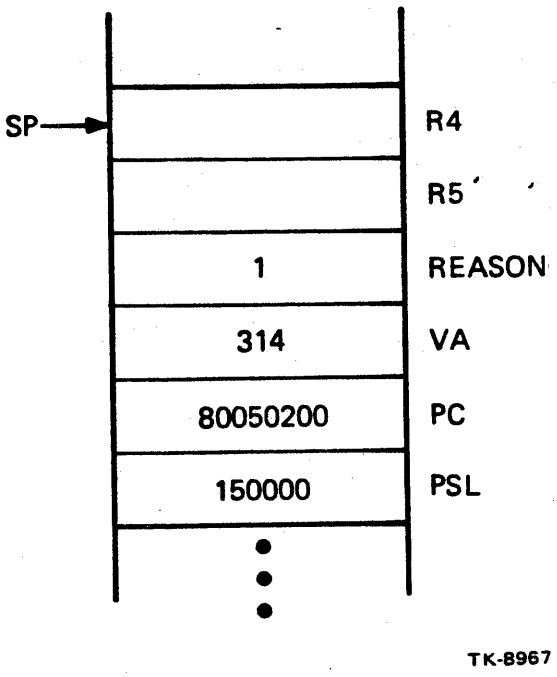


Stack After Access Violation Bugcheck

Probable Causes:

- Blown register
- Incorrect data structure field
- Improper synchronization

Page Fault Above IPL 2

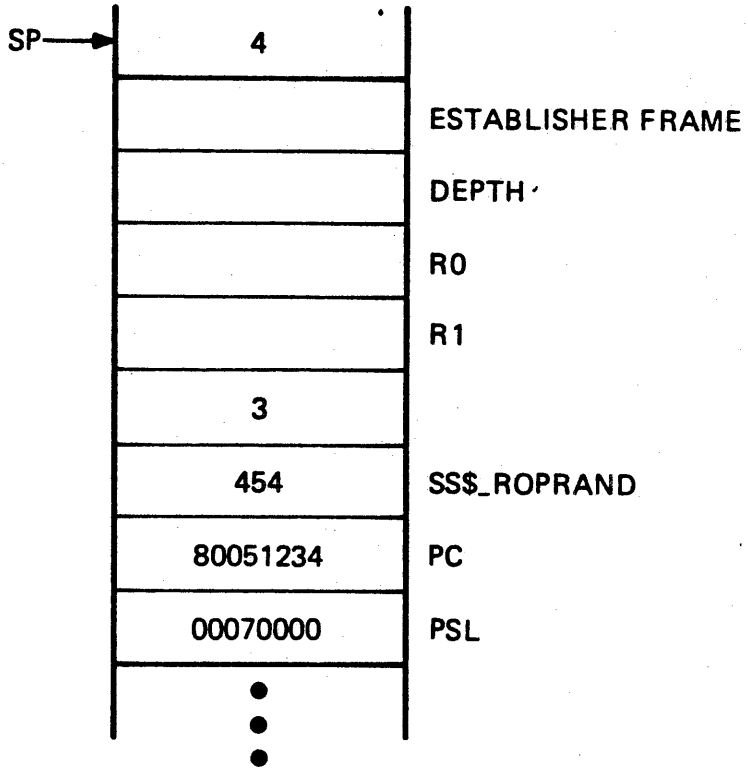


Stack After Page Fault Above IPL-2

Probable Causes:

- Blown register in fork interrupt routine
- Improper start I/O routine design

Reserved Operand Fault



Stack After Reserved Operand Fault

Probable Causes:

- REI failure
  - IPL problems (allocate memory at wrong IPL)
  - Blown stack
- RET failure

SAMPLE CRASH ANALYSIS 1

Most system crashes are caused by access violations, typically as the result of referencing an illegal address. Program CRASHER is an example of a kernel mode program that references address 20 (decimal). This causes an access violation which crashes the system because:

* the linker makes the first page, with addresses 0-511 decimal, a no access page, and
* the default contents of the primary exception vector for kernel mode is the address of a procedure that issues a fatal bugcheck (SSVEXCEPT). This bugcheck simply indicates that an exception occurred while in executive or kernel mode.

This example shows how the bugcheck output (which also is in the SDA dump file) can be used to locate the instruction that caused a system crash. In this example, the signal and mechanism arrays are used to locate the offending instruction.

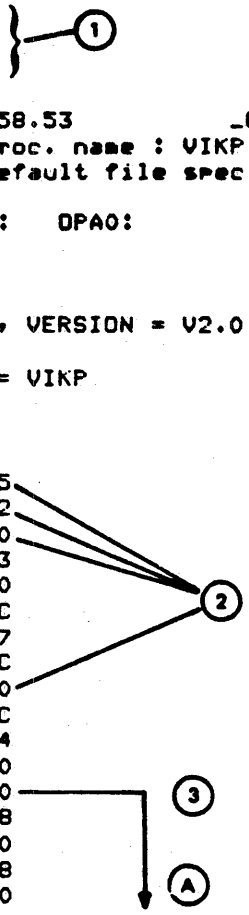
You will also be asked to "debug" a program that is crashing the system, by using the crash output to locate the offending instruction.

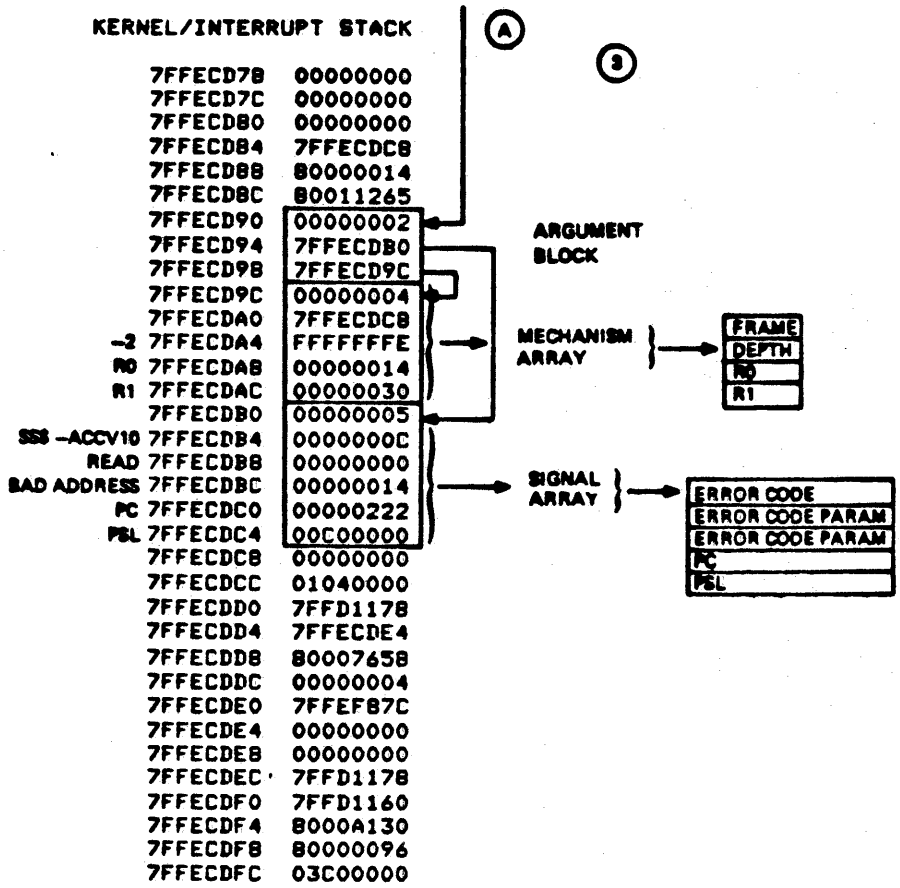
```
                        0000    100         .TITLE  CRASHER -- This program crashes the system
                        0000    200         .IDENT  /V01/
                        0000    300 ;++
                        0000    400 ;
                        0000    500 ; ABSTRACT:
                        0000    600 ;
                        0000    700 ;       Any error in kernel mode causes the system to crash,  In this
                        0000    800 ;       case, the program simply accesses address 20, to which the
                        0000    900 ;       program has no access.
                        0000   1000 ;
                        0000   1100 ; SIDE EFFECTS:
                        0000   1200 ;
                        0000   1300 ;       The system crashes!!
                        0000   1400 ;
                        0000   1500 ; PROGRAMMER:
                        0000   1600 ;
                        0000   1700 ;       Vik Muiznieks   MAY-21-1980
                        0000   1800 ;
                        0000   1900 ;--
                        0000   2000 ;
                        0000   2100 ;       Program entry point
                        0000   2200 ;
                0000    0000   2300 START:  .WORD   0                       ; null entry mask
                        0002   2400         $CMKRNL_S ROUTIN=10$            ; enter kernel mode
                  04    0011   2500         RET                             ; will never get here
                0104    0012   2600 10$:    .WORD   ^M<R2,R8>               ; save two registers
        50    14  9A    0014   2700         MOVZBL  #20,R0                  ; place bad address in R0
        51    30  9A    0017   2800         MOVZBL  #48,R1                  ; store dummy values
    52    40  8F  9A    001A   2900         MOVZBL  #64,R2                  ; to see on BUGCHECK
    58    50  8F  9A    001E   3000         MOVZBL  #80,R8                  ; output
        58    60  D0    0022   3100         MOVL    (R0),R8                 ; this instruction should
                        0025   3200                                         ; crash the system
50      0000'8F   3C    0025   3300         MOVZWL  #SS$_NORMAL,R0          ; if get here, return success
                  04    002A   3400         RET                             ; code
                        002B   3500         .END    START
```

PC = 2222 First 200 NOT USED

```
CRASHER                         -- This program crashes the system      25-JUL-1980 12:10:51  VAX-11 Macro V02.45          Page   2
Symbol table                                                            23-JUN-1980 11:03:05  _DRA0:[VIK]CRASHER.MAR;2      (1)

SS$_NORMAL          ********   X   01
START               00000000 R     01
SYS$CMKRNL          ********  GX   01


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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
. BLANK .                       0000002B  (   43.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE


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

Phase                       Page faults     CPU Time        Elapsed Time
-----                       -----------     --------        ------------
Initialization                       22     00:00:00.06     00:00:00.18
Command processing                   21     00:00:00.18     00:00:00.38
Pass 1                              381     00:00:00.53     00:00:01.16
Symbol table sort                     3     00:00:00.00     00:00:00.00
Pass 2                               79     00:00:00.17     00:00:00.32
Symbol table output                   2     00:00:00.01     00:00:00.01
Psect synopsis output                 5     00:00:00.02     00:00:00.06
Cross-reference output                0     00:00:00.00     00:00:00.00
Assembler run totals                521     00:00:00.98     00:00:02.17

The working set limit was 150 pages.
674 bytes (2 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 3 non-local and 1 local symbols.
35 source lines were read in Pass 1, producing 10 object records in Pass 2.
2 pages of virtual memory were used to define 2 macros.


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

Macro library name                    Macros defined
------------------                    --------------
_DRA1:[SYSLIB]STARLET.MLB;1                      2

13 GETS were required to define 2 macros.

There were no errors, warnings or information messages.

/LIST CRASHER
```

```
$ MACRO/LIST CRASHER
$ LINK CRASHER
$ SHOW PROCESS

  25-JUL-1980 12:10:58.53          _OPA0:       User : VIKP
  Pid : 001F000D   Proc. name : VIKP            UIC  : [311,007]
  Priority :   4   Default file spec. :    DRA0:[VIK]

  Devices allocated :   OPA0:
$ RUN CRASHER

**** FATAL BUG CHECK, VERSION = V2.0 SSRVEXCEPT, Unexpected system service exception

    CURRENT PROCESS = VIKP

    REGISTER DUMP

        R0 = 7FFEFE35
        R1 = 8000A122
        R2 = 00000040
        R3 = 7FFD8D73
        R4 = 8007AC00
        R5 = 7FFEC48C
        R6 = 31000407
        R7 = 7FFEF99C
        R8 = 00000050
        R9 = 7FFEF98C
        R10= 7FFEF9A4
        R11= 7FFEA210
       *AP = 7FFECD90
        FP = 7FFECD78
        SP = 7FFECD70
        PC = 8000A128
        PSL= 00000000
```





EXEC STACK

PROCESS PRIVILEGES

800B2200 1FFFFFFF
800B2204 00000000

IMAGE NAME = _DRA0:[VIK]CRASHER.EXE;6 ④

HALT INST EXECUTED
HALTED AT 800072E7

(BOOTING) ⑤
CPU HALTED
INIT SEQ DONE
HALT INST EXECUTED
HALTED AT 200034F9

G 0000000E 00000200
LOAD DONE, 00002000 BYTES LOADED

```
$ COPY SYS$SYSTEM:SYSDUMP.DMP DRA0:[VIK]CRASHER.DMP ——(6)
$ RUN SYS$SYSTEM:SDA
Enter name of dump file > DRA0:[VIK]CRASHER.DMP ——(7)
VAX/VMS System dump analyzer

Dump taken on 25-JUL-1980 12:11:10.57
SSRVEXCEPT, Unexpected system service exception

SDA> SHOW CRASH ——(8)
Time of system crash: 25-JUL-1980 12:11:10.57


Version of system: VAX/VMS VERSION V2.0


Reason for BUGCHECK exception: SSRVEXCEPT, Unexpected system service exception


Process currently executing: VIKP


Current image file name: _DRA0:[VIK]CRASHER.EXE;6


Current IPL: 0  (decimal)



General registers:

        R0  = 7FFEFE35   R1  = 8000A122   R2  = 00000040   R3  = 7FFD8D73
        R4  = 8007AC00   R5  = 7FFEC48C   R6  = 31000407   R7  = 7FFEF99C
        R8  = 00000050   R9  = 7FFEF98C   R10 = 7FFEF9A4   R11 = 7FFEA210
        AP  = 7FFECD90   FP  = 7FFECD78   SP  = 7FFECD78   PC  = 8000A128
        PSL = 00000000



Processor registers:

        P0BR   = 800B2C00      PCBB   = 00037074      ACCS   = 00000000
        P0LR   = 00000002      SCBB   = 000BCE00      SBIFS  = 00040000
        P1BR   = 7F8B6C00      ASTLVL = 00000004      SBISC  = 00000000
        P1LR   = 001FFE75      SISR   = 00180000      SBIMT  = 00200200
        SBR    = 000BDE00      ICCS   = 800000C1      SBIER  = 00008002
        SLR    = 00000880      ICR    = FFFFF30A      SBITA  = 20000001
                               TODR   = 7A59266F      SBIS   = 00000000

        ISP    = 80088400
        KSP    = 7FFECD78
        ESP    = 7FFEDE00
        SSP    = 7FFEF878
        USP    = 7FFD1160
SDA>
```

```
SDA> SHOW STACK                                                    (8)
Current operating stack (KERNEL):

                    7FFECD58  7FFEF98C
                    7FFECD5C  7FFEF9A4
                    7FFECD60  7FFEA210      CTL$AG_CLIDATA
                    7FFECD64  7FFECD90      CTL$GL_KSTKBAS+590
                    7FFECD68  7FFECD78      CTL$GL_KSTKBAS+578
                    7FFECD6C  7FFECD70      CTL$GL_KSTKBAS+570
                    7FFECD70  8000A128      EXE$EXCPTN+006
                    7FFECD74  00000000

              SP => 7FFECD78  00000000
                    7FFECD7C  00000000
                    7FFECD80  00000000
                    7FFECD84  7FFECDC8      CTL$GL_KSTKBAS+5C8
                    7FFECD88  80000014      SYS$CALL_HANDL+004
                    7FFECD8C  80011265      EXE$REFLECT+14E
          AP ->     7FFECD90  00000002
                    7FFECD94  7FFECDB0      CTL$GL_KSTKBAS+5B0
                    7FFECD98  7FFECD9C      CTL$GL_KSTKBAS+59C
                    7FFECD9C  00000004
          MECHSAV ->7FFECDA0  7FFECDC8      CTL$GL_KSTKBAS+5C8
                    7FFECDA4  FFFFFFFE
                    7FFECDA8  00000014
                    7FFECDAC  00000030
          SIGNAL -> 7FFECDB0  00000005      ARGS
                    7FFECDB4  0000000C      SS$_ACCVIO
                    7FFECDB8  00000000      READ
                    7FFECDBC  00000014      BAD ADDRESS
                    7FFECBC0  00000222      PC
                    7FFECDC4  00C00000      PSL$M_PRVMOD
                    7FFECDC8  00000000
                    7FFECDCC  01040000
                    7FFECDD0  7FFD1178
                    7FFECDD4  7FFECDE4      CTL$GL_KSTKBAS+5E4
                    7FFECDD8  80007658      EXE$CMKRNL+018
                    7FFECDDC  00000004
                    7FFECDE0  7FFEF87C
                    7FFECDE4  00000000
                    7FFECDE8  00000000
                    7FFECDEC  7FFD1178
                    7FFECDF0  7FFD1160
                    7FFECDF4  8000A130      EXE$EXCPTN+00E
                    7FFECDF8  80000096      SYS$CMKRNL+006
                    7FFECDFC  03C00000

SDA> EXIT
$
```

SDA will match addresses with system routine entry point

CRASH ANALYSIS 2

Please use the following bugcheck output and corresponding assembler listing to determine the reason for the system crash, and also to locate the offending instruction and possible "fix" for the problem.

```
$ analyze/crash sys$system:
VAX/VMS System dump analyzer

Dump taken on  4-DEC-1982 19:51:58.88
SSRVEXCEPT, Unexpected system service exception

SDA> show crash
Time of system crash:  4-DEC-1982 19:51:58.88


Version of system: VAX/VMS VERSION V3.0


Reason for BUGCHECK exception: SSRVEXCEPT, Unexpected system service exception


Process currently executing: ELLIS


Current image file name: SYS$SYSDEVICE:[ELLIS]CEBCRASH.EXE;1


Current IPL: 2  (decimal)


General registers:

        R0  = 00000000   R1  = 8000CFEA   R2  = 00000000   R3  = 00000210
        R4  = 00000000   R5  = 00000000   R6  = 31000208   R7  = 7FFED988
        R8  = 7FFED570   R9  = 7FFED778   R10 = 7FFEDDD4   R11 = 7FFE7C90
        AP  = 7FFEAD84   FP  = 7FFEAD6C   SP  = 7FFEAD6C   PC  = 8000CFF0
        PSL = 00020000


Processor registers:

        P0BR    = 80141600      PCBB    = 0015B674      ACCS    = 00000000
        P0LR    = 00000004      SCBB    = 001B9C00      TBDR    = 00000000
        P1BR    = 7F949600      ASTLVL  = 00000004      CADR    = 00000000
        P1LR    = 001FFDA1      SISR    = 00000000      MCESR   = 0000000C
        SBR     = 001BAE00      ICCS    = 800000C1      CAER    = 00000000
        SLR     = 00001480      ICR     = FFFFF01B      CMIERR  = 00080113
                                TODR    = BDF9D7C3

        ISP     = 8012C200
        KSP     = 7FFEAD6C
        ESP     = 7FFEBE00
        SSP     = 7FFED56C
        USP     = 7FFB6960
SDA> sho stack
Current operating stack (KERNEL):

                7FFEAD4C  7FFED778
                7FFEAD50  7FFEDDD4
                7FFEAD54  7FFE7C90      CTL$AG_CLIDATA+080
```

ok AST can't be delivered, process can't be deleted.

```
                7FFEAD60  7FFEAD64      CTL$GL_KSTKBAS+564
                7FFEAD64  8000CFF0      EXE$EXCEPTN+006
                7FFEAD68  00020000

          SP => 7FFEAD6C  00000000
                7FFEAD70  00000000
                7FFEAD74  00000292      BUG$_NOMULTBK+002
                7FFEAD78  7FFEADC0      CTL$GL_KSTKBAS+5C0
                7FFEAD7C  80000014      SYS$CALL_HANDL+004
                7FFEAD80  800130D1      EXE$SRCHANDLER+0C5
                7FFEAD84  00000002
                7FFEAD88  7FFEADA8      CTL$GL_KSTKBAS+5A8
                7FFEAD8C  7FFEAD90      CTL$GL_KSTKBAS+590
                7FFEAD90  00000004
                7FFEAD94  7FFB6960
                7FFEAD98  FFFFFFFD      VA$M_VPG+1FD
                7FFEAD9C  00000000
                7FFEADA0  801153ED
                7FFEADA4  000008F8      SS$_ENDOFFILE+088
                7FFEADA8  00000005
                7FFEADAC  0000000C
                7FFEADB0  00000000
                7FFEADB4  00000022
                7FFEADB8  000004D9      BUG$_EXTCACHIV+001
                7FFEADBC  00C20004
                7FFEADC0  00000000
                7FFEADC4  003C0000
                7FFEADC8  7FFB6978
                7FFEADCC  7FFEADE4      CTL$GL_KSTKBAS+5E4
                7FFEADD0  80008089      EXE$CMKRNL+00D
                7FFEADD4  00000004
                7FFEADD8  7FFC513B
                7FFEADDC  800ED660
                7FFEADE0  7FFEA490      MMG$IMGHDRBUF+090
                7FFEADE4  00000000
                7FFEADE8  00000000
                7FFEADEC  7FFB6978
                7FFEADF0  7FFB6960
                7FFEADF4  8000CFE6      EXE$CMODEXEC+18E
                7FFEADF8  7FFEDE96      SYS$CMKRNL+006
                7FFEADFC  03C00000
SDA> ex/inst 4d9
BUG$_EXTCACHIV+001:  MOVW    22(R2),BUG$_NOAQBACP+003
SDA>
```

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
.MAIN.	0	404	SYS$SYSDEVICE:[ELLIS]CEBCRASH.OBJ;1	4-DEC-1982 19:51	VAX-11 Macro V03-00
SYS	V03-003	0	SYS$SYSROOT:[SYSEXE]SYS.STB;1	27-APR-1982 03:46	VAX-11 Linker V03-16

Program Section Synopsis

Psect Name	Module Name	Base	End	Length	Align	Attributes
DATA	.MAIN.	00000200	00000291	00000092 (146.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
SCRATCH	.MAIN.	00000292	0000029D	0000000C (12.)	BYTE 0	NOPIC,USR,CON,REL,LCL,NOSHR,NOEXE, RD, WRT,NOVEC
CODE	.MAIN.	00000400	000004F5	000000F6 (246.)	BYTE 0	NOPIC,USR,CON,REL,LCL, SHR, EXE, RD,NOWRT,NOVEC

Symbols By Name

Symbol	Value	Symbol	Value	Symbol	Value	Symbol	Value
EXE$GL_CEBMTX	800028C4						
GET_CEB_INFO	0000048D-R						
LOC_CEB	00000400-R						
SCH$GL_CURPCB	8000210C						
SCH$GQ_CEBHD	80002124						
SCH$LOCKR	80009802						
SCH$UNLOCK	80009870						
SYS$ASSIGN	7FFEDE50						
SYS$CMKRNL	7FFEDE90						
SYS$K_VERSION	594A3158						
SYS$QIOW	7FFEDE00						

PC = 4D9
- Base
PC in Psect

409
- 400
D9 . PC in code Psect

Key for special characters above:

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

```
                                +----------------+
                                ! Image Synopsis !
                                +----------------+

Virtual memory allocated:                00000200 000007FF 00000600 (1536. bytes, 3. pages)
Stack size:                                    20. pages
Image header virtual block limits:              1.        1. (    1. block)
Image binary virtual block limits:              2.        3. (    2. blocks)
Image name and identification:           CEBCRASH 0
Number of files:                                4.
Number of modules:                              3.
Number of program sections:                     8.
Number of global symbols:                      11.
Number of image sections:                       4.
User transfer address:                   00000400
Debugger transfer address:               7FFEDF68
Image type:                              EXECUTABLE.
Map format:                              DEFAULT in file SYS$SYSDEVICE:[ELLIS]CEBCRASH.MAP;1
Estimated map length:                    15. blocks
                              +---------------------+
                              ! Link Run Statistics !
                              +---------------------+

Performance Indicators                   Page Faults   CPU Time        Elapsed Time
----------------------                   -----------   --------        ------------
    Command processing:                           25   00:00:00.11     00:00:00.15
    Pass 1:                                      120   00:00:01.54     00:00:02.34
    Allocation/Relocation:                        25   00:00:00.09     00:00:00.21
    Pass 2:                                       54   00:00:00.30     00:00:00.70
    Map data after object module synopsis:        25   00:00:00.13     00:00:00.13
    Symbol table output:                           3   00:00:00.02     00:00:00.16
Total run values:                                252   00:00:02.19     00:00:03.69

Using a working set limited to 415 pages and 33 pages of data storage (excluding image)

Total number object records read (both passes):   190
    of which 18 were in libraries and 2 were DEBUG data records containing 139 bytes
123 bytes of DEBUG data were written,starting at VBN 5 with 1 blocks allocated

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

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

/MAP CEBCRASH,SYS$SYSTEM:SYS.STB/SEL
$
```

```
                                        0000     1          $CEBDEF
                                    00000000     2          .PSECT  DATA,NOEXE
                                        0000     3
                                        0000     4  CEB_DESC:
                                   0000 0000     5          .WORD   0
                                  01 0E 0002     6          .BYTE   14,1
                                0000000C' 0004     7          .ADDRESS        CEB_NAME
                                        0008     8  CEB_NAME_LEN:
                                00000000  0008     9          .LONG   0
                                        000C    10  CEB_NAME:
                                0000001B  000C    11          .BLKB   15
                                        001B    12  HEADER:
20 20 20 20 45 4D 41 4E 5F 42 45 43     001B    13          .ASCII  /CEB_NAME          OWNER           PROT      PERM/<10>
20 20 52 45 4E 57 4F 20 20 20 20 20     0027
54 4F 52 50 20 20 20 20 20 20 20 20     0033
      0A 4D 52 45 50 20 20 20 20 20     003F
20 20 20 2D 2D 2D 2D 2D 2D 2D 0D        0049    14          .ASCII  <13>/--------          -----           ----      ----/
20 2D 2D 2D 2D 2D 20 20 20 20 20 20     0055
2D 2D 2D 20 20 20 20 20 20 20 20 20     0061
      2D 2D 2D 2D 20 20 20 20 20 2D     006D
                                0000005C  0077    15  HEADER_LENGTH=.-HEADER
                                        0077    16  PROT_ON_OFF:
                                     00 0077    17          .BYTE   0
                                        0078    18  PERM_CEB:
                                     00 0078    19          .BYTE   0
                                        0079    20  UIC_OWNER:
                                   0000 0079    21          .WORD   0
                                        007B    22  UIC_GROUP:
                                   0000 007B    23          .WORD   0
                                     00 007D    24  DONE:   .BYTE   0
                                        007E    25          .SAVE_PSECT
                                    00000000    26          .PSECT  SCRATCH NOSHR,WRT,NOEXE
                                        0000    27  GCI_ARGS:
                                00000002  0000    28          .LONG   2
                                00000000  0004    29          .LONG   0               ;0->FIRST TIME THROUGH AND WILD CARD, CEB PTR
                                00000000  0008    30          .LONG   0
                                    0000007E    31          .RESTORE_PSECT
                                        007E    32  TERM_CHANNEL:
                                   0000 007E    33          .WORD   0
                                        0080    34  TERMINAL:
55 4F 24 53 59 53 00000088'010E0000'    0080    35          .ASCID  /SYS$OUTPUT/
                        54 55 50 54     008E
                                    00000000    36          .PSECT  CODE,EXE,NOWRT,SHR
                                   0000 0000    37          .ENTRY  LOC_CEB,^M< >
                                        0002    38          $ASSIGN_S       CHAN=TERM_CHANNEL,DEVNAM=TERMINAL
                                        0017    39          $QIOW_S         CHAN=TERM_CHANNEL,FUNC=#IO$_WRITEVBLK,P1=HEADER,-
                                        0017    40                          P2=#HEADER_LENGTH
              0000007D'EF   01    90    0040    41          MOVB            #1,DONE                 ;Note that we have not finished
                                        0047    42  10$:
                                        0047    43          $CMKRNL_S       ROUTIN=GET_CEB_INFO,ARGLST=GCI_ARGS     ;Get into kernal for
ceb's
                                        005A    44  ;Format and display ceb info
                2B 0000007D'EF    E9    005A    45          BLBC    DONE,20$
                                        0061    46          $OUTPUT         CHAN=TERM_CHANNEL,BUFFER=CEB_NAME,-
                                        0061    47                          LENGTH=CEB_NAME_LEN
                                  BB    11    008A    48          BRB     10$
                                     04 008C    49  20$:    RET
                                   003C 008D    50          .ENTRY  GET_CEB_INFO,^M<R2,R3,R4,R5>
```

```
                50   00000000'GF   DE   008F    51          MOVAL   G^EXE$GL_CEBMTX,R0       ;Set up mutex on ceb structures
                     00000000'GF   16   0096    52          JSB     G^SCH$LOCKR              ;Lock for read access (assume R4->PCB)
                           04 AC   D5   009C    53          TSTL    4(AP)                    ;B
                              08   12   009F    54          BNEQ    GOT_CURRENT
            04 AC    00000000'GF   D0   00A1    55          MOVL    G^SCH$GQ_CEBHD,4(AP)     ;4(AP)-> TO NEXT CEB
                                   00A9    56 GOT_CURRENT:
                         52   04 AC   D0   00A9    57          MOVL    4(AP),R2
                      04 AC   04 BC   D0   00AD    58          MOVL    @4(AP),4(AP)
            00000004'GF   04 AC   D1   00B2    59          CMPL    4(AP),G^SCH$GQ_CEBHD+4   ;POINTS TO LAST CEB?
                              08   12   00BA    60          BNEQ    MORE_CEBS
                     0000007D'EF   94   00BC    61          CLRB    DONE
                              1D   11   00C2    62          BRB     NO_MORE_CEBS
                                   00C4    63 MORE_CEBS:
            00000008'EF   28 A2   9A   00C4    64          MOVZBL  CEB$T_EFCNAM(R2),CEB_NAME_LEN
            29 A2    00000008'EF   28   00CC    65          MOVC3   CEB_NAME_LEN,CEB$T_EFCNAM+1(R2),CEB_NAME
                     0000000C'EF        00D4
Error here  ->      0000007B'EF   22 A2   B0   00D9    66          MOVW    CEB$W_GRP(R2),UIC_GROUP
                                   00E1    67 NO_MORE_CEBS:
                50   00000000'GF   DE   00E1    68          MOVAL   G^EXE$GL_CEBMTX,R0
                54   00000000'GF   D0   00E8    69          MOVL    G^SCH$GL_CURPCB,R4
                     00000000'GF   16   00EF    70          JSB     G^SCH$UNLOCK             ;Free up mutex on ceb
                                   04   00F5    71          RET
                                        00F6    72          .END    LOC_CEB
```

Cause: R0 - R5 wiped out
use PUSHR
POPR

.MAIN.
Symbol table

```
$$T1            =  00000001
CEB$B_CREATPORT    0000001E
CEB$B_DELETPORT    0000001F
CEB$B_LOCK         0000001C
CEB$B_PROCCNT      0000001D
CEB$B_STS          0000000B
CEB$B_TYPE         0000000A
CEB$C_LENGTH       00000038
CEB$C_SLAVLNG      00000044
CEB$K_LENGTH       00000038
CEB$K_SLAVLNG      00000044
CEB$L_CEBBL        00000004
CEB$L_CEBFL        00000000
CEB$L_EFC          00000010
CEB$L_MASTER       00000040
CEB$L_PID          0000000C
CEB$L_SHB          00000038
CEB$L_UIC          00000020
CEB$L_VASLAVE1     00000038
CEB$L_WQBL         00000018
CEB$L_WQFL         00000014
CEB$T_EFCNAM       00000028
CEB$W_GRP          00000022
CEB$W_INDX         0000003C
CEB$W_PROT         00000024
CEB$W_REFC         00000026
CEB$W_SIZE         00000008
CEB$W_STATE        0000001E
CEB$W_WQCNT        0000001C
CEB_DESC           00000000 R      03
CEB_NAME           0000000C R      03
CEB_NAME_LEN       00000008 R      03
DONE               0000007D R      03
EXE$GL_CEBMTX      ********   X    05
GCI_ARGS           00000000 R      04
GET_CEB_INFO       0000008D RG     05
GOT_CURRENT        000000A9 R      05
HEADER             0000001B R      03
HEADER_LENGTH   =  0000005C
IO$_WRITEVBLK   =  00000030
LOC_CEB            00000000 RG     05
MORE_CEBS          000000C4 R      05
NO_MORE_CEBS       000000E1 R      05
PERM_CEB           00000078 R      03
PROT_ON_OFF        00000077 R      03
SCH$GL_CURPCB      ********   X    05
SCH$GQ_CEBHD       ********   X    05
SCH$LOCKR          ********   X    05
SCH$UNLOCK         ********   X    05
SYS$ASSIGN         ********  GX    05
SYS$CMKRNL         ********  GX    05
SYS$QIOW           ********  GX    05
TERMINAL           00000080 R      03
TERM_CHANNEL       0000007E R      03
UIC_GROUP          0000007B R      03
UIC_OWNR           00000079 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000044 (68.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DATA	00000092 (146.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
SCRATCH	0000000C (12.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
CODE	000000F6 (246.)	05 (5.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.07	00:00:00.18
Command processing	40	00:00:00.35	00:00:00.72
Pass 1	238	00:00:06.11	00:00:07.00
Symbol table sort	1	00:00:00.85	00:00:00.85
Pass 2	67	00:00:00.98	00:00:01.09
Symbol table output	8	00:00:00.10	00:00:00.10
Psect synopsis output	4	00:00:00.05	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	393	00:00:08.54	00:00:10.01

The working set limit was 425 pages.
16596 bytes (33 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 334 non-local and 2 local symbols.
72 source lines were read in Pass 1, producing 24 object records in Pass 2.
17 pages of virtual memory were used to define 16 macros.

Macro library statistics

Macro library name	Macros defined
SYS$SYSROOT:[SYSLIB]LIB.MLB;1	1
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	12
TOTALS (all libraries)	13

497 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

/LIS/ENABLE=SUPP CEBCRASH+SYS$LIBRARY:LIB/LIB
$ type cebcrash.map

SCHEDULING

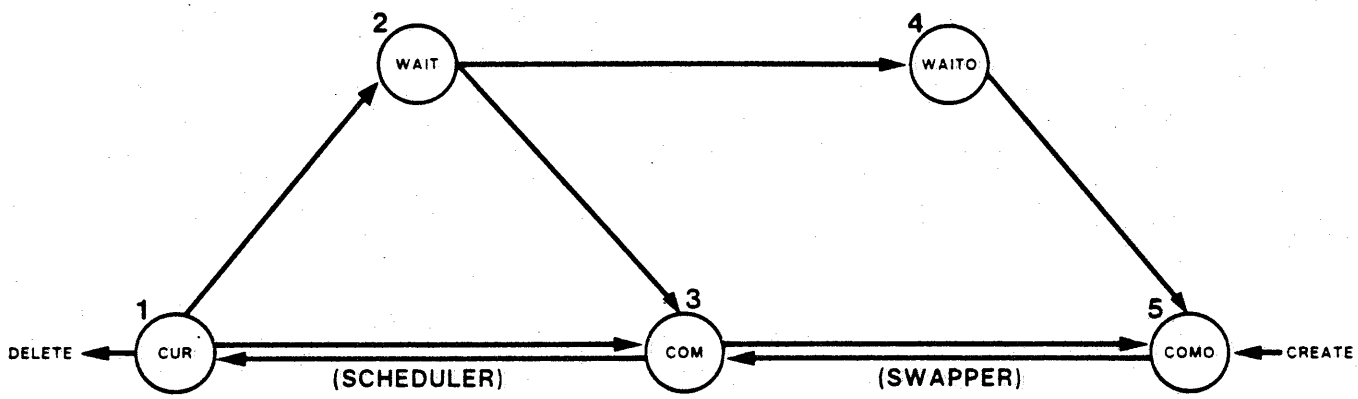
OBJECTIVES

* for selected process states, describe the properties of a process in the state
* for selected process states, describe how processes enter and leave the state
* describe the operations of the scheduler (look at code)
* see the effect of priority on the system (software)

READINGS

* IDSM Chapter 8,14.2,
* (Optional) IDSM Chapter 4,5,9,10,24

PROCESS STATES OVERVIEW



8-2 IDSM

D-24 IDSM

$STATEDEF

PROCESS WAIT STATES

SCH$C_ Name
LEF
CUR
HIB

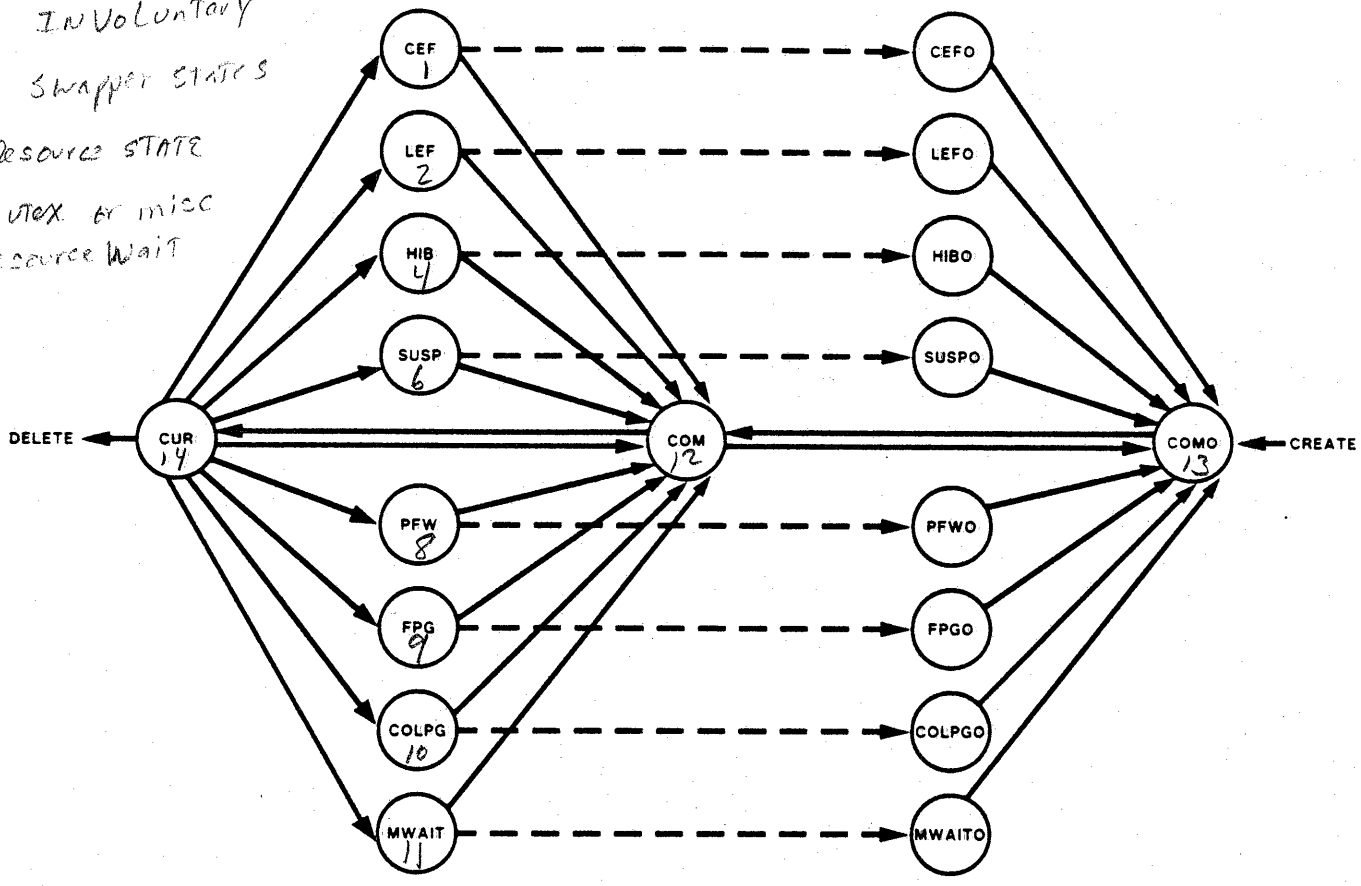
1-5 Voluntary

6-9 Involuntary

8-9 Swapper states

10 Resource STATE

11 mutex or misc Resource Wait

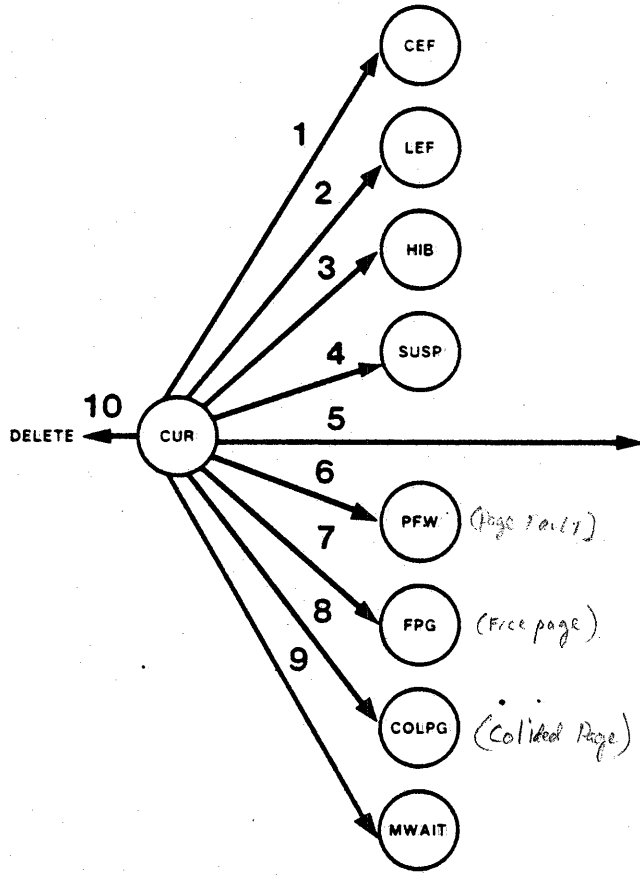


14 Process STATES

1 Process that is CUR

all others are in a wait state

WAYS TO LEAVE CURRENT STATE



① $ASCEFC
$WAITFR

② RMS $SETIMR (any service that allows a flag to be set on completion)
$QIO $GETJPI

③ $HIBER

④ $SUSPND

⑤ Pre-Emption
Quantum-END

⑥ Resolve a PF from Disk

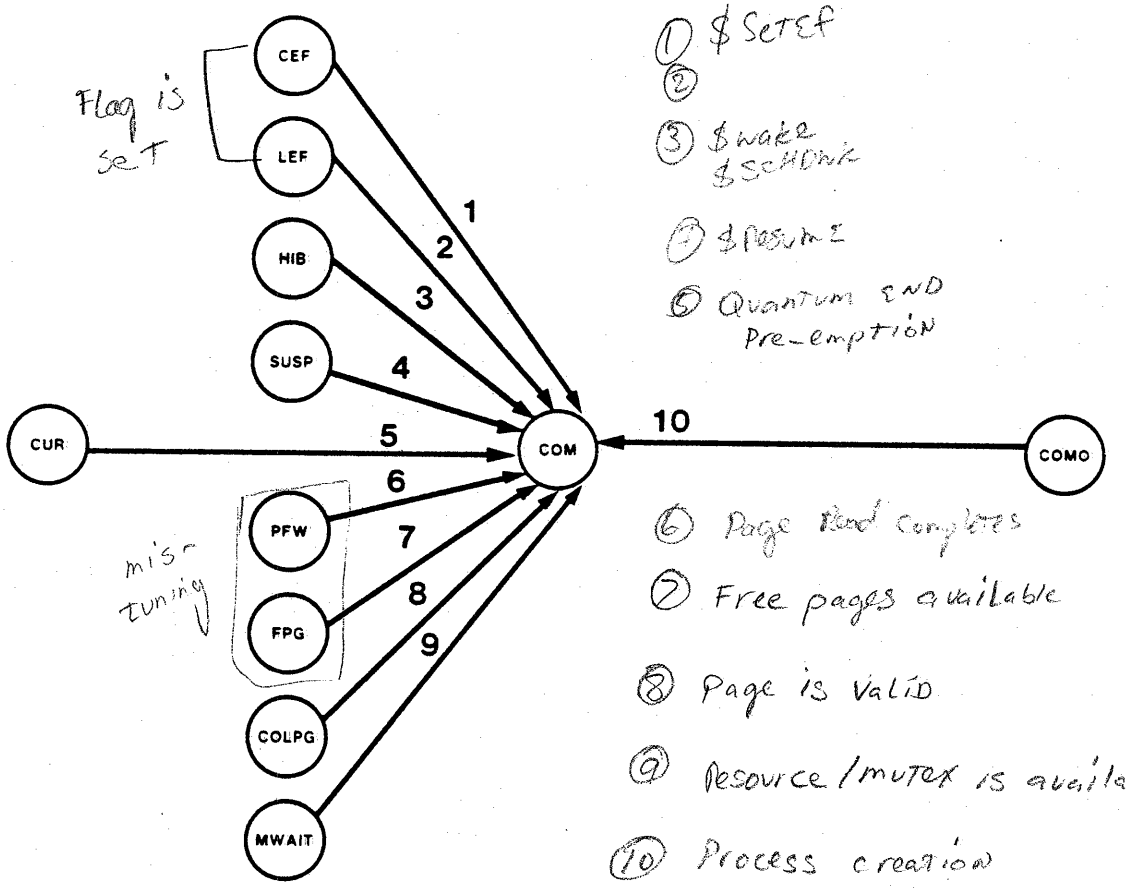
⑦ FPL Depleted

⑧ Fault on shared page that has already been faulted on

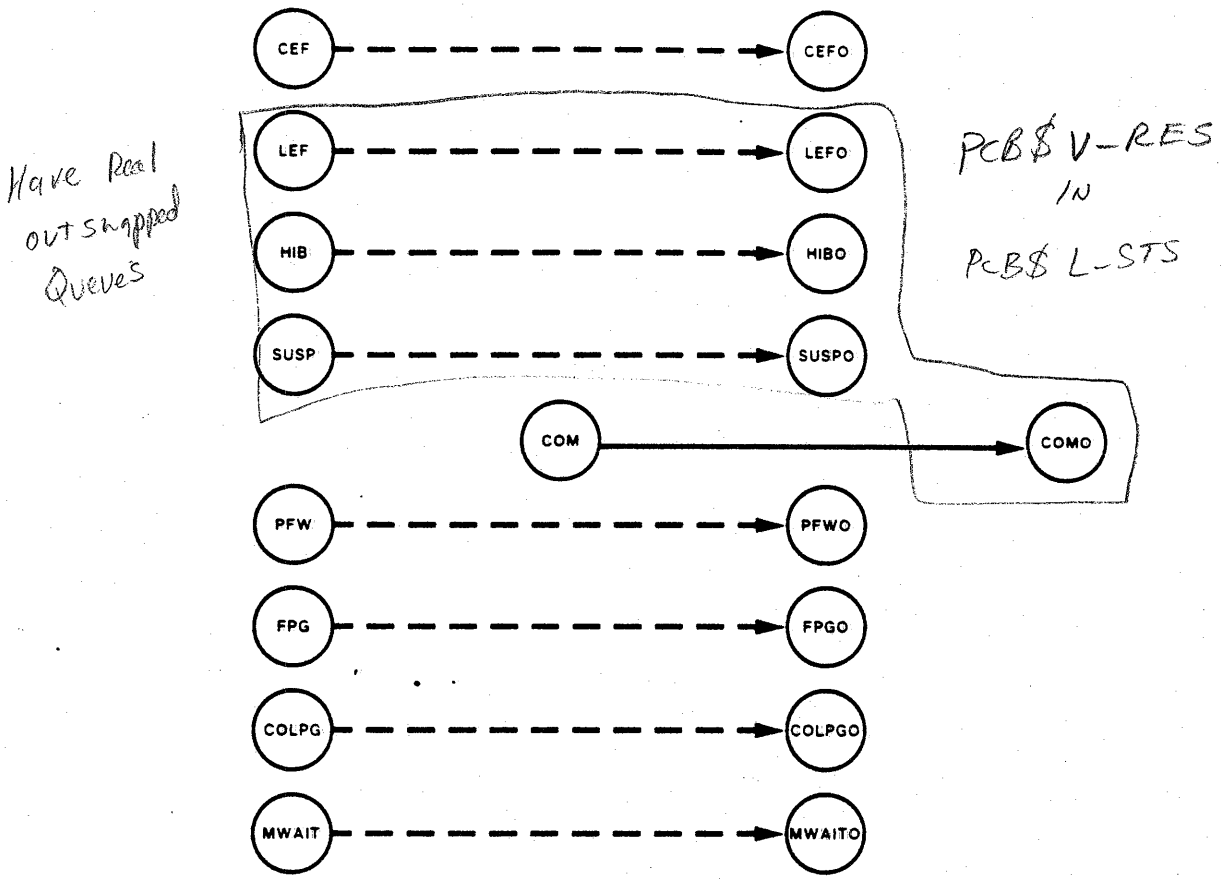
⑨ Waiting on mutex
- resource is NOT available (depletion)
  - AST
  - IRP
  - Buff I/O
  - Non-Paged Pool

Can't get out of These (Non-paged pool
BRKThru
IACLock

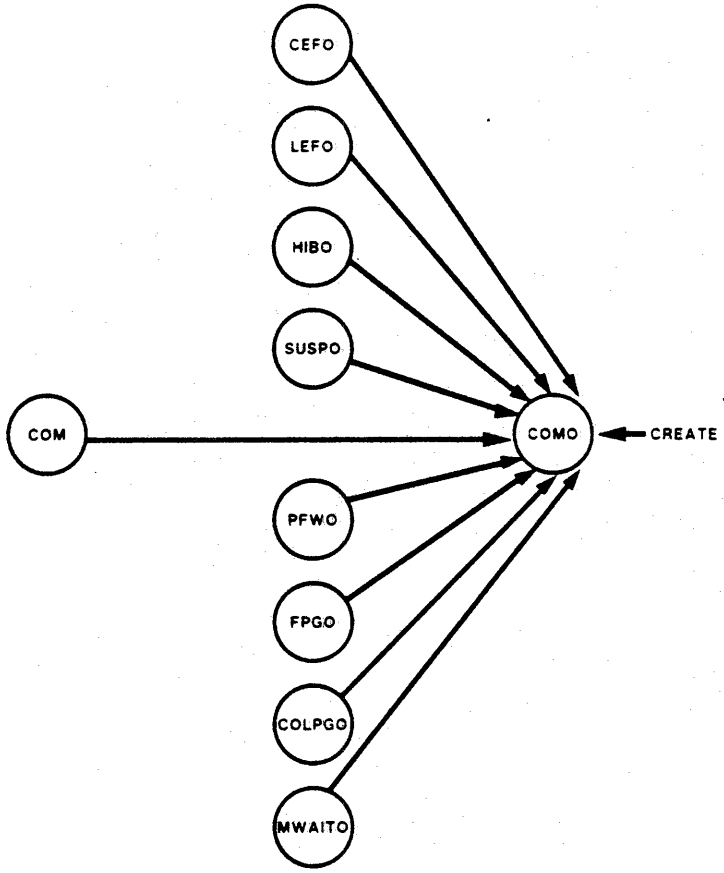
WAYS TO BECOME COMPUTABLE INSWAPPED (COM)



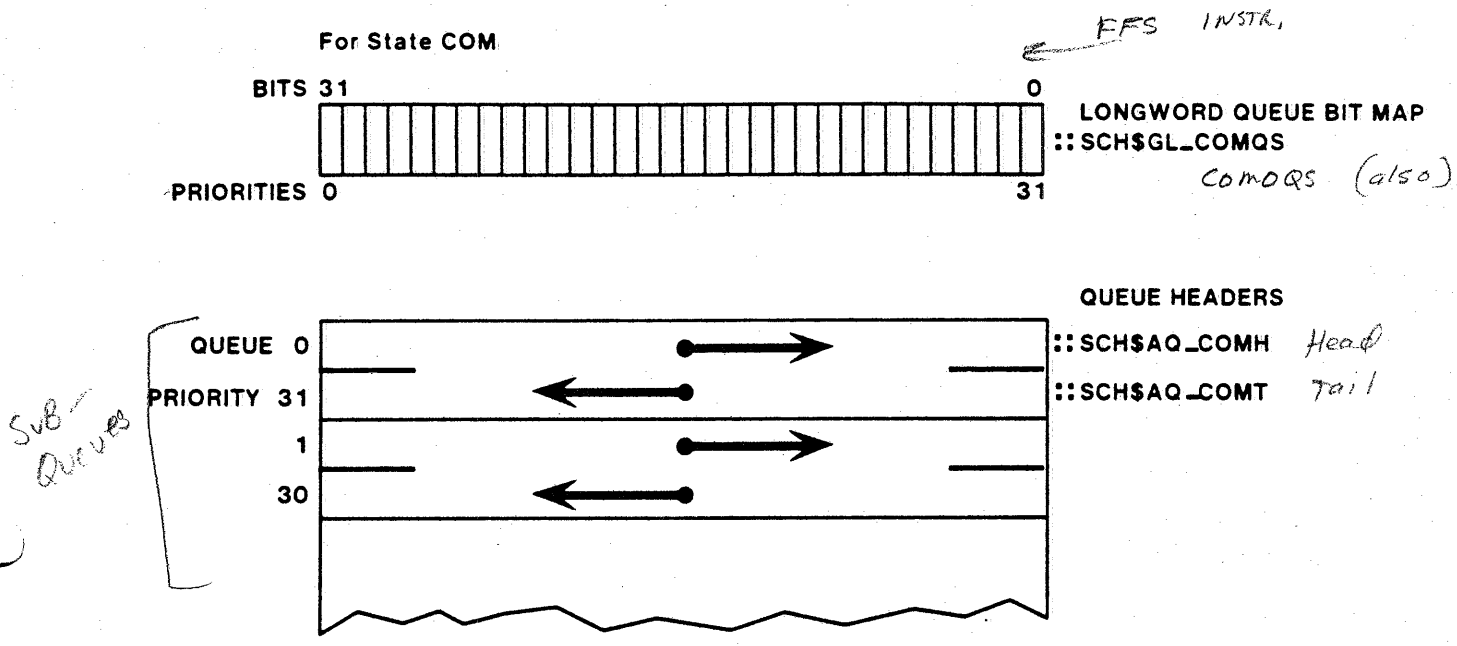
INSWAPPED TO OUTSWAPPED TRANSITIONS



WAYS TO BECOME COMPUTABLE OUTSWAPPED (COMO)



IMPLEMENTATION OF COM AND COMO STATES (PRIORITIES)



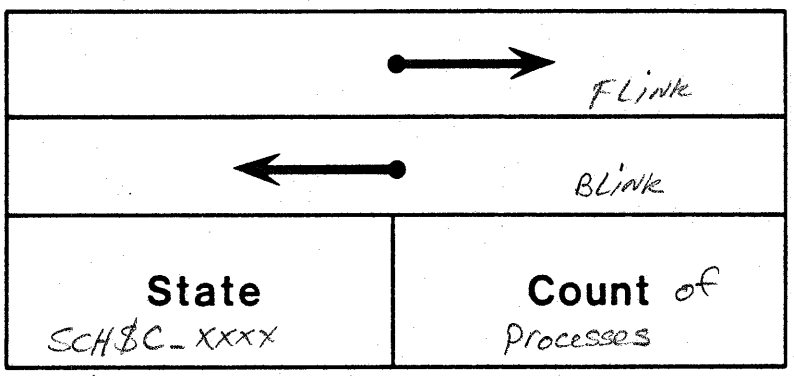
Com/como have 32 subqueues (listheads)

Priority is stored as 31 - spl

Ex. Null process = 0 stored as 31 - 0 = 31

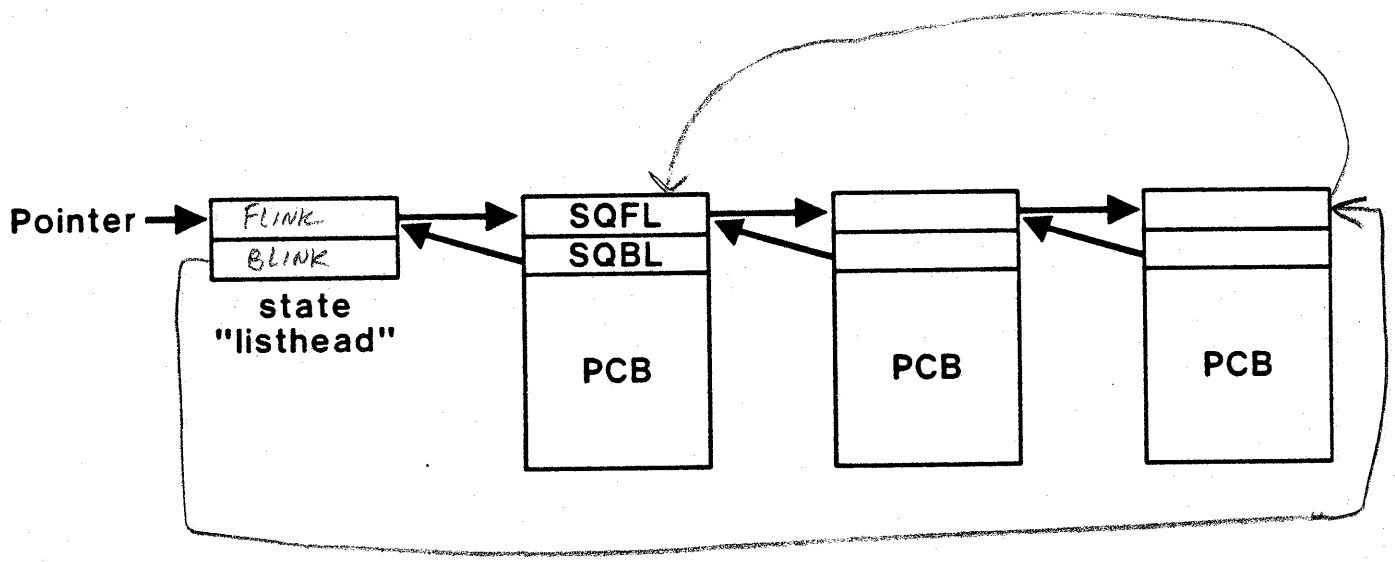
A-45

WAIT STATE LISTHEAD



INSQUE
REMQUE
> updates double-link queues

IMPLEMENTATION OF STATES BY QUEUES

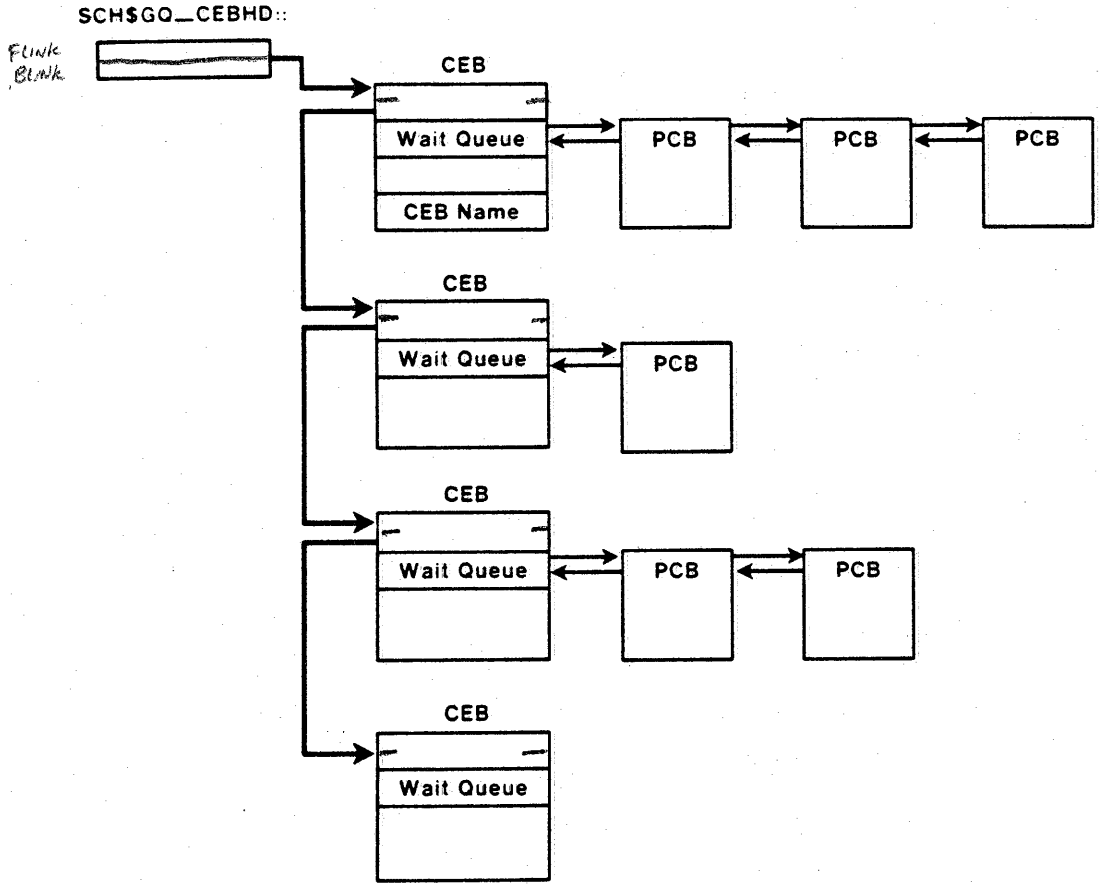


If FLINK = BLINK Then Done

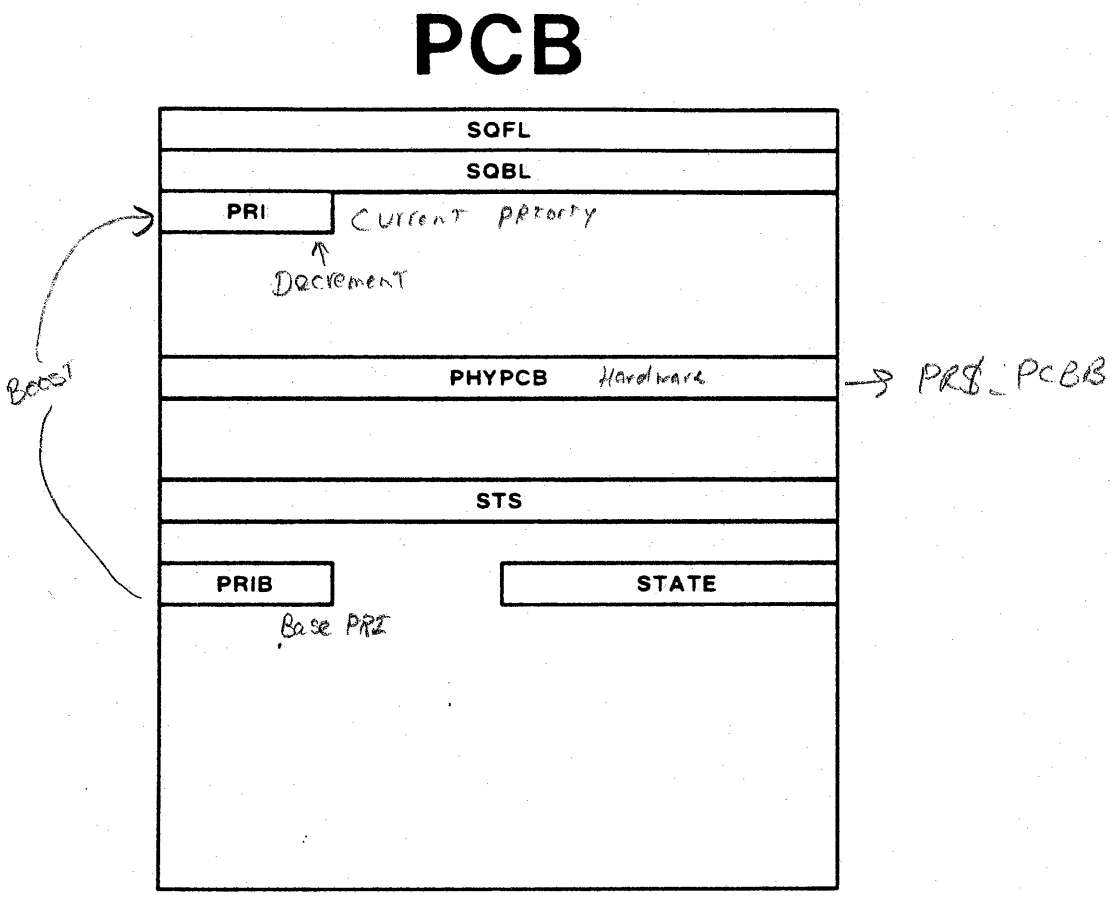
IMPLEMENTATION OF CEF STATE

9-4 IDSM

Data structure on A-4

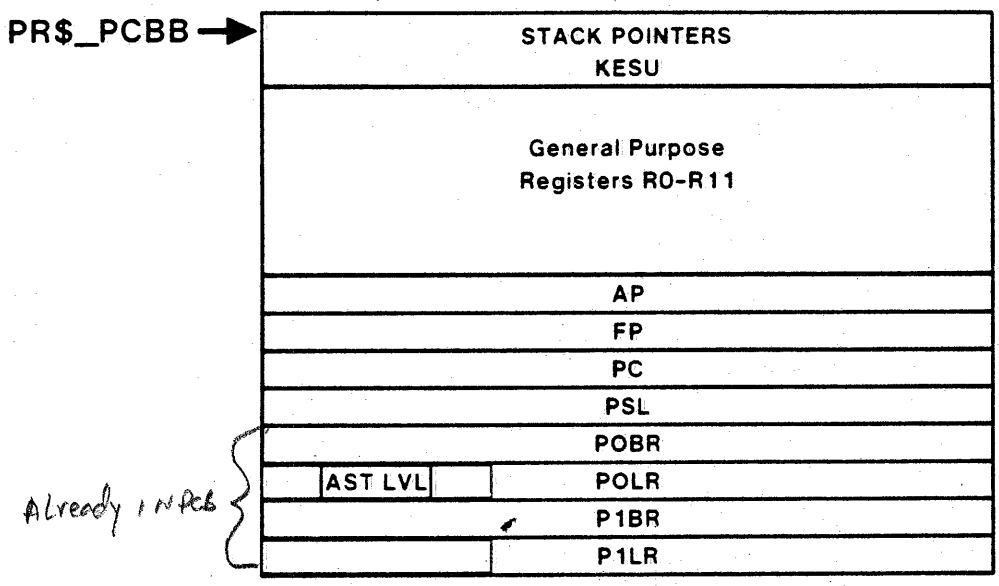


SCHEDULING FIELDS IN SOFTWARE PCB



8-26 TO 8-28 IDSM

HARDWARE PCB



SVPCTX

- copies SPs
  GPRs
  AP, FP, PC, PSL
- Switches TO INT STACK +
  Re-schedule

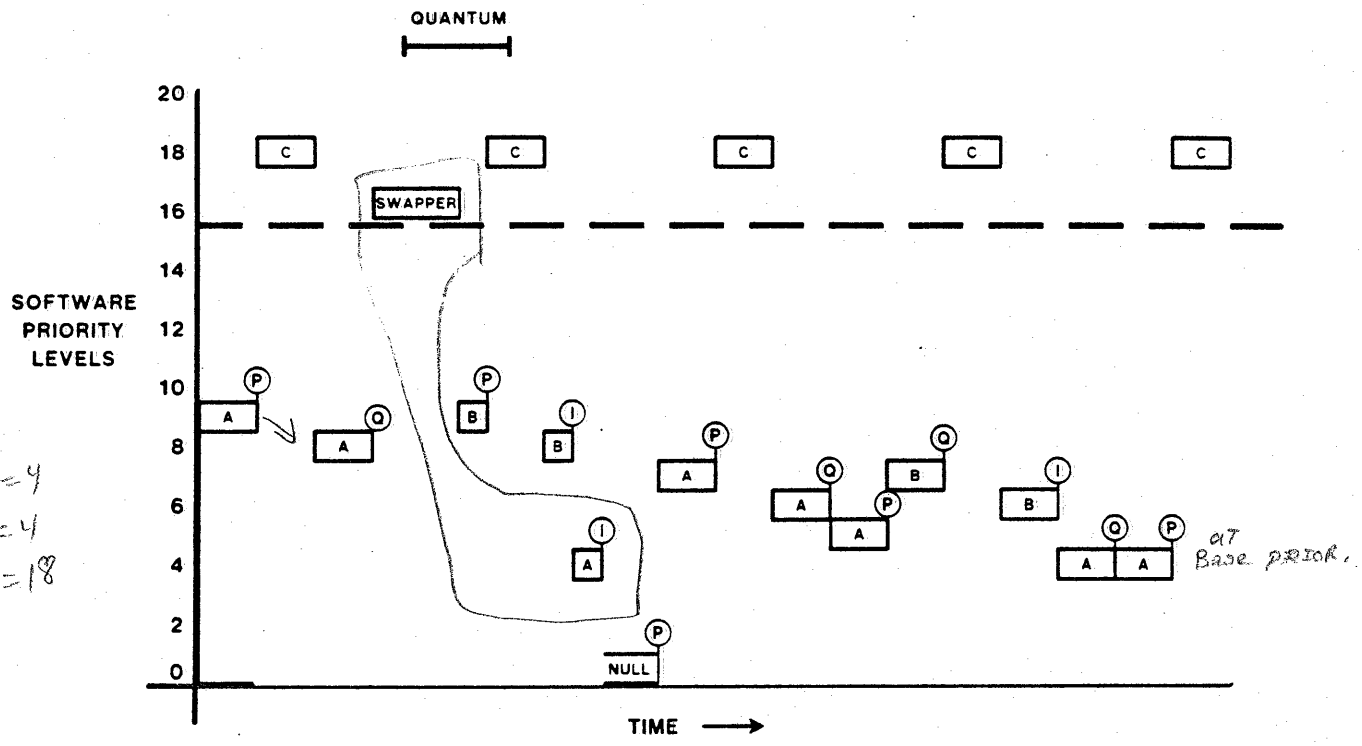
LDPCTX

Load PCB$L_PHYDCB INTO
PR$_PCBB

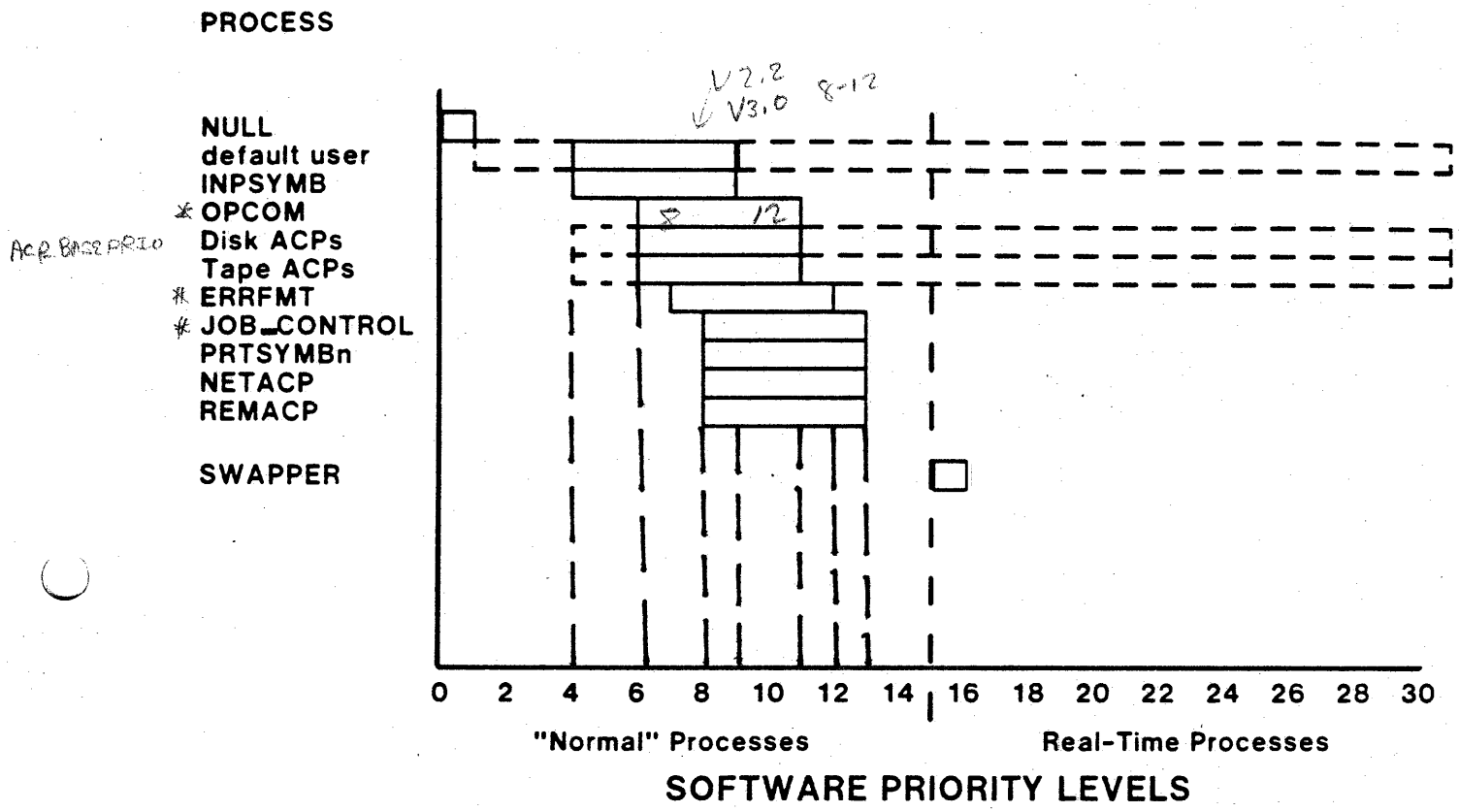
- Restores SP's
- Restore GPRs
- Restore ASTLVL + P_x BRs
  P_xLRs
- PUTS PC & PSL ON
  INTR. STACK
  & REI POPS THem

SOFTWARE PRIORITIES AND PRIORITY ADJUSTMENT

8-6 8-7 IDSm



SOFTWARE PRIORITY LEVELS OF SYSTEM PROCESSES



SOFTWARE PRIORITY LEVELS

ACP BASEPRIO

* SYS$SYSTEM:STARTUP.COM

Stay away from PRIOR 0, 10-15

At 14 could block ACP Disk access

At 16 could block Swapper (contention)
Use 17 or higher

At 31 can't get in and kill it

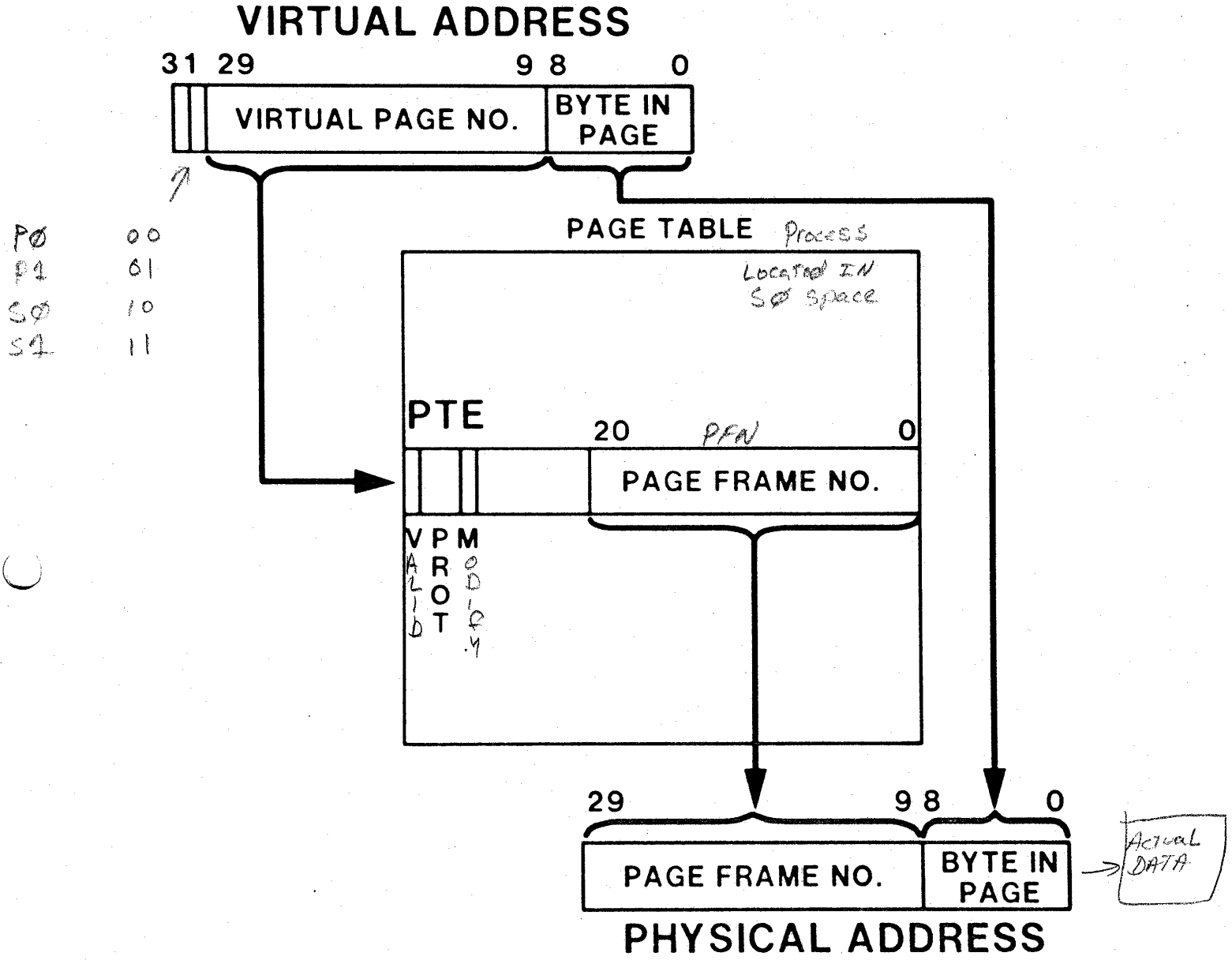
PAGING

OBJECTIVES

* describe the effects of changing working set size, creating/deleting virtual address space, and creating/mapping global sections
* considerations for programming
* SYSGEN parameters that may affect paging
* given an initial set of conditions, work through the changes in status and locations of pages/process states

READINGS

* IDSM Chapters 11,12,13
* (Optional) IDSM Chapter 18

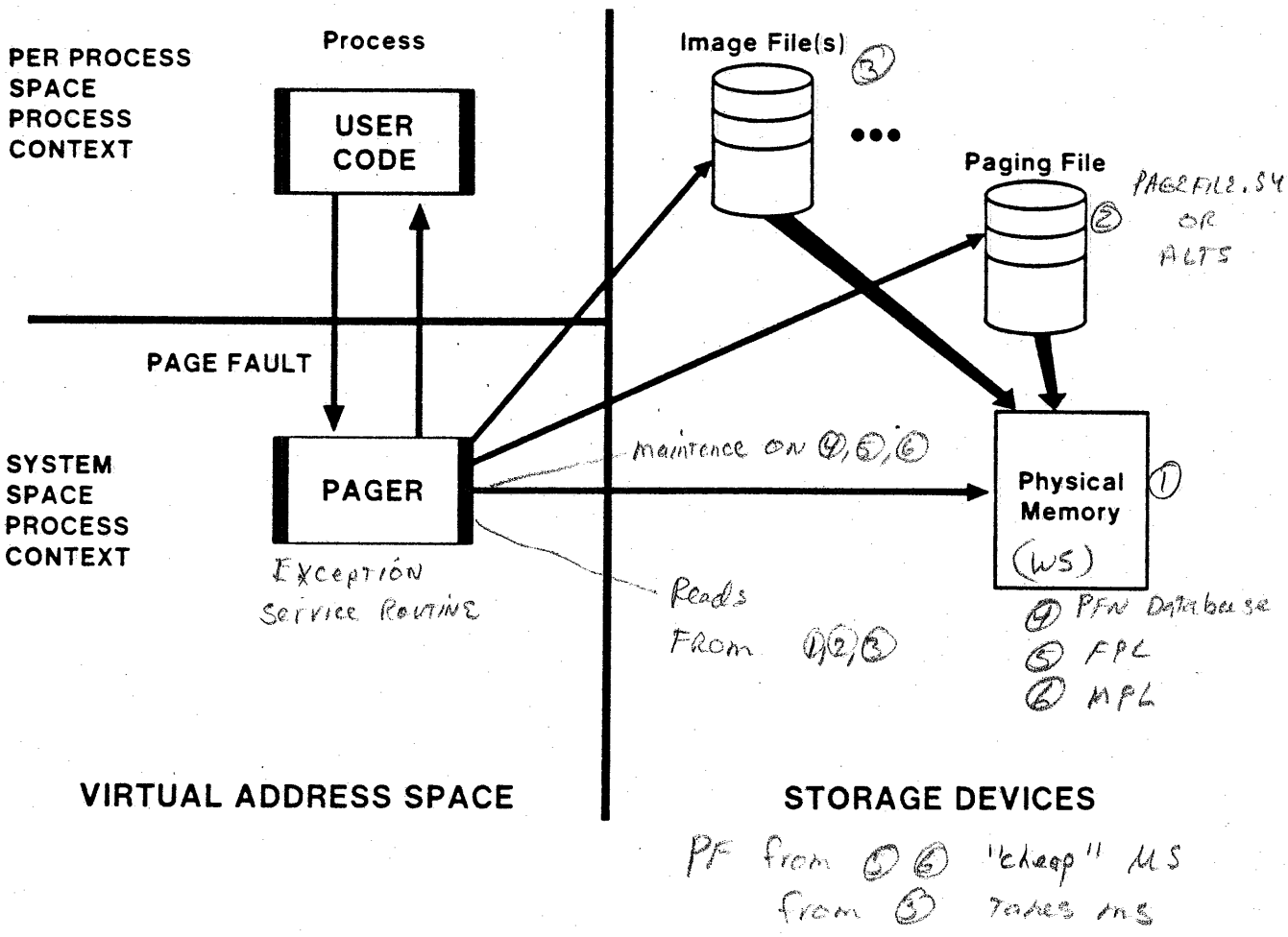


Address translation is the hardware operation of converting a virtual address into a physical address for actual execution of an instruction. The conversion, or mapping, information is located in an entry in the appropriate page table.

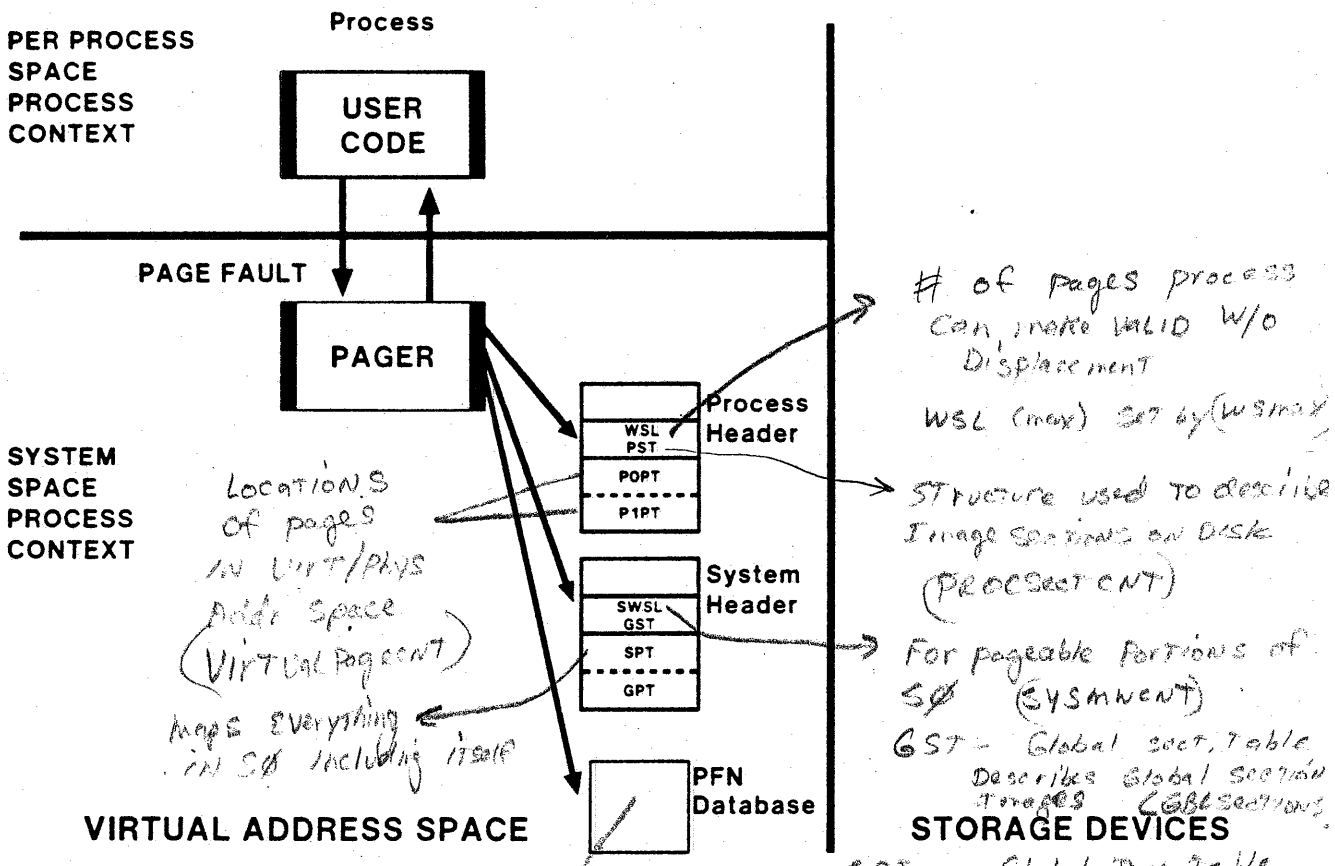
There is Process Page Table System Page Table

map each page of S0 space

RESOLVING PAGE FAULTS



DATA STRUCTURES USED BY THE PAGER



$ MONITOR Page

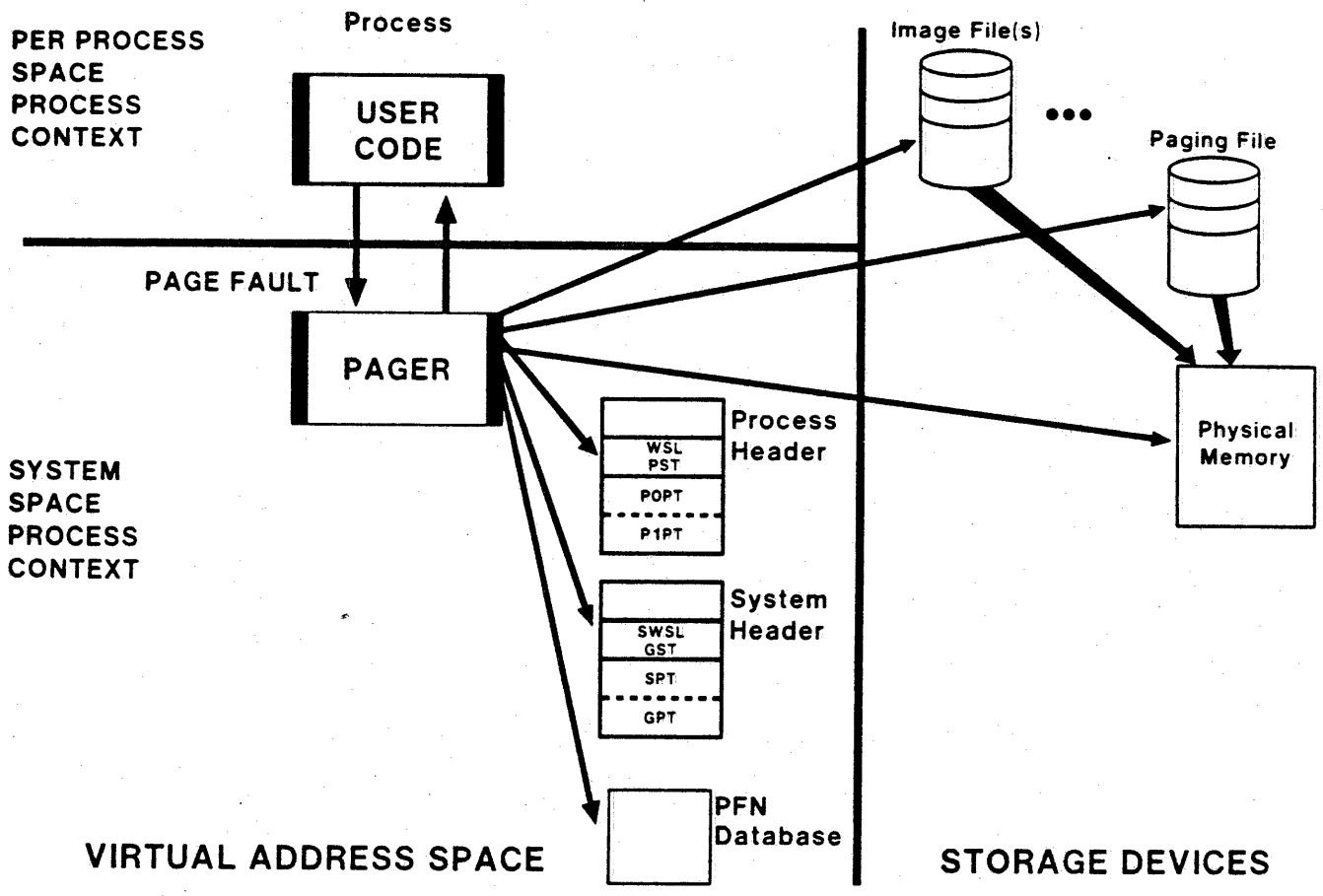
System Fault Rate ≤ 1-3/sec

MMG $GL_SYSPHD (Address of System header) in R3

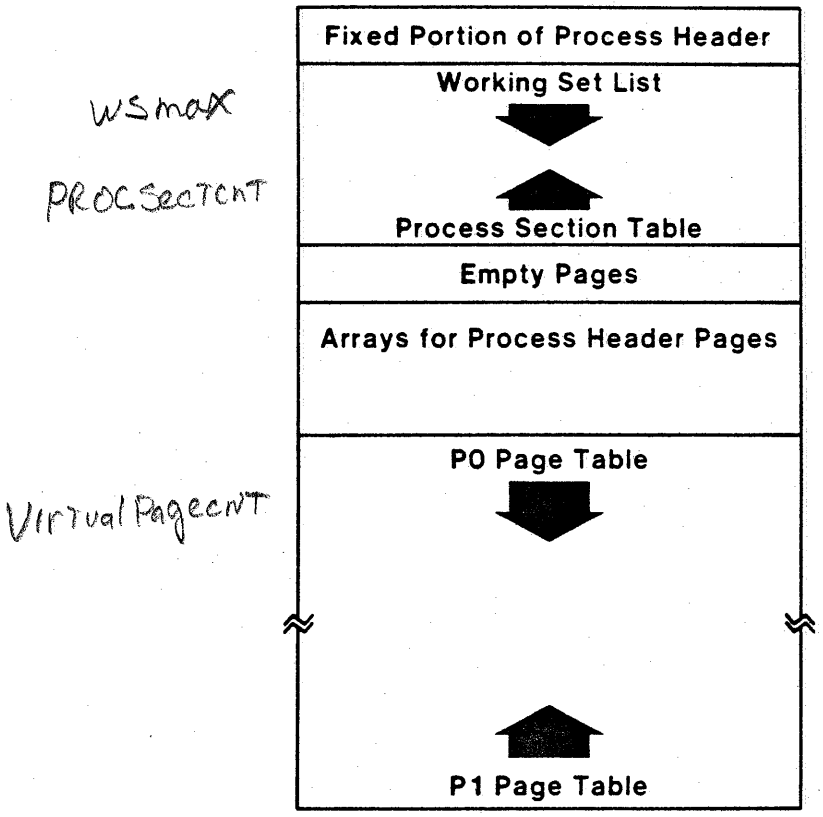
Look at PHD$L_PAGEFLTS(R3)

= # of sys page faults

SUMMARY OF THE PAGER

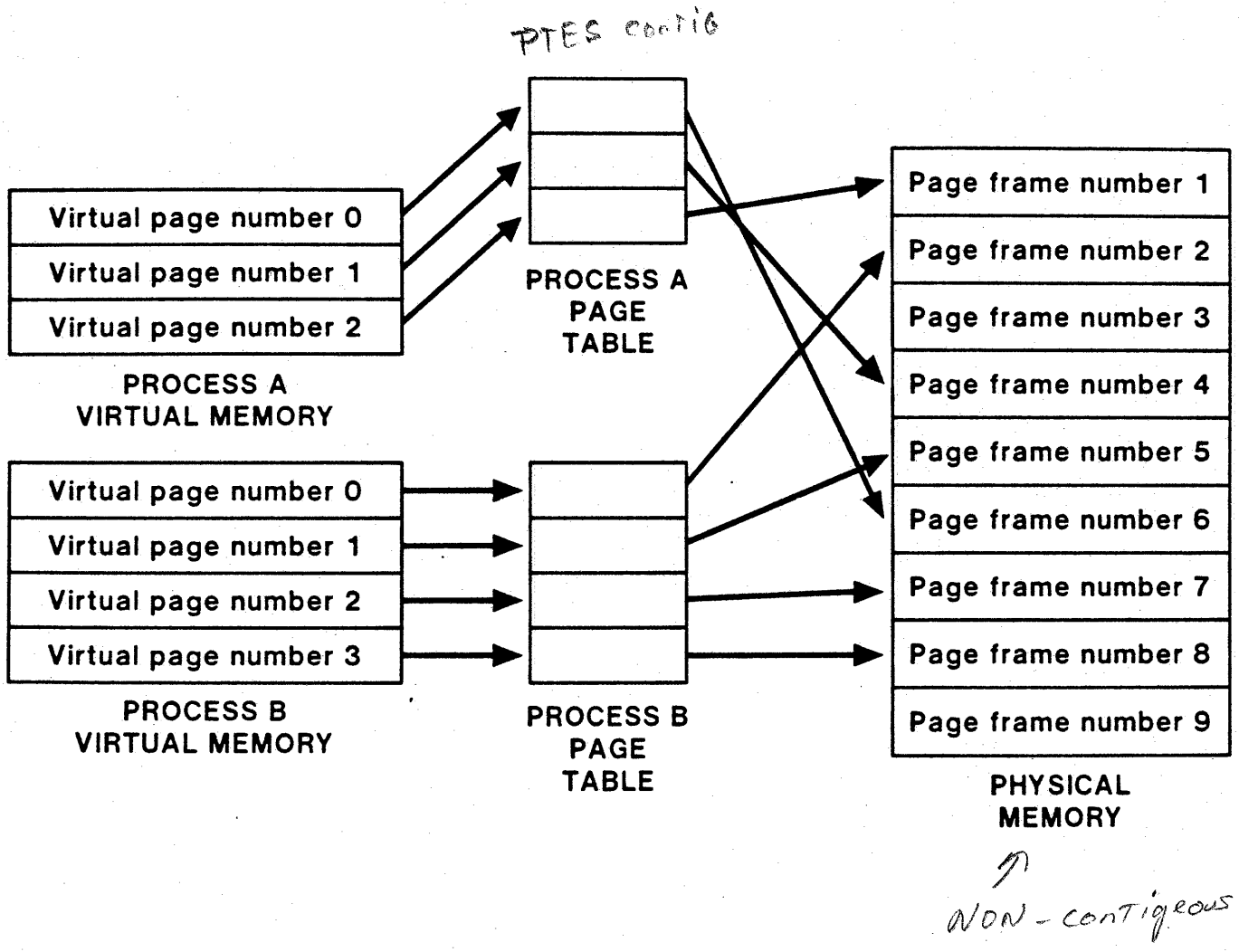


PROCESS HEADER



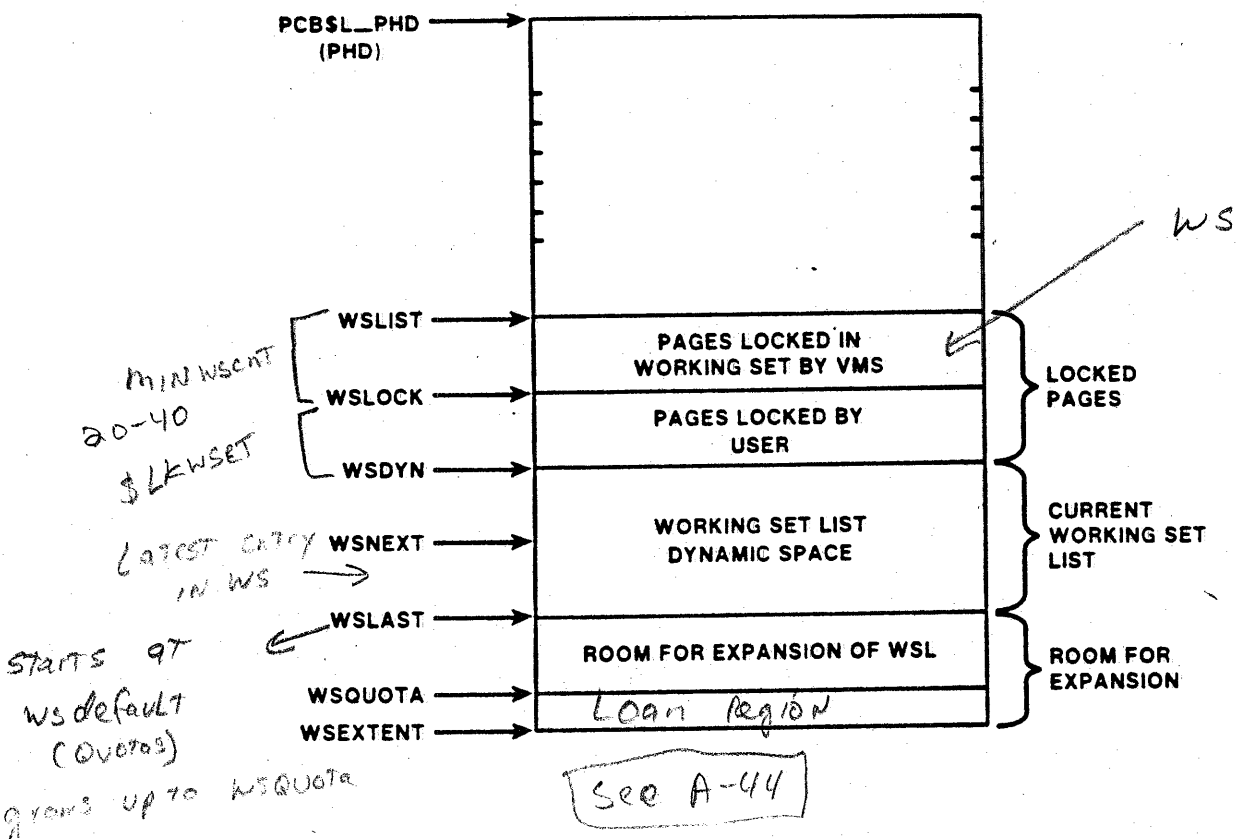
11-2 to 11-7 IDSM

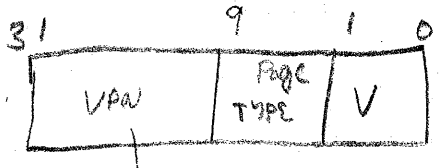
RELATIONSHIP OF VIRTUAL TO PHYSICAL MEMORY



11-7 to 11-11 IDSM.

WORKING SET LIST - Describes valid Pages

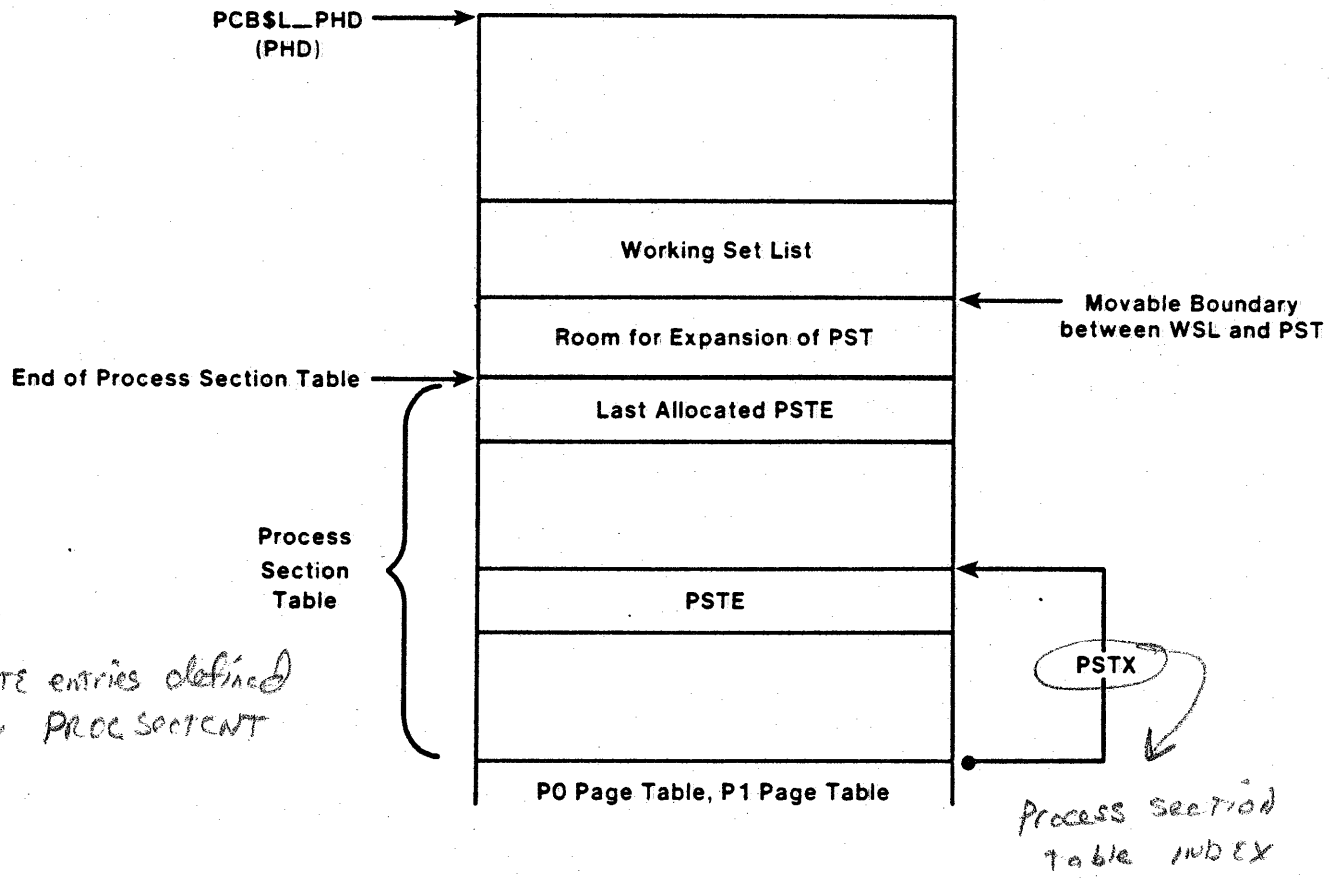




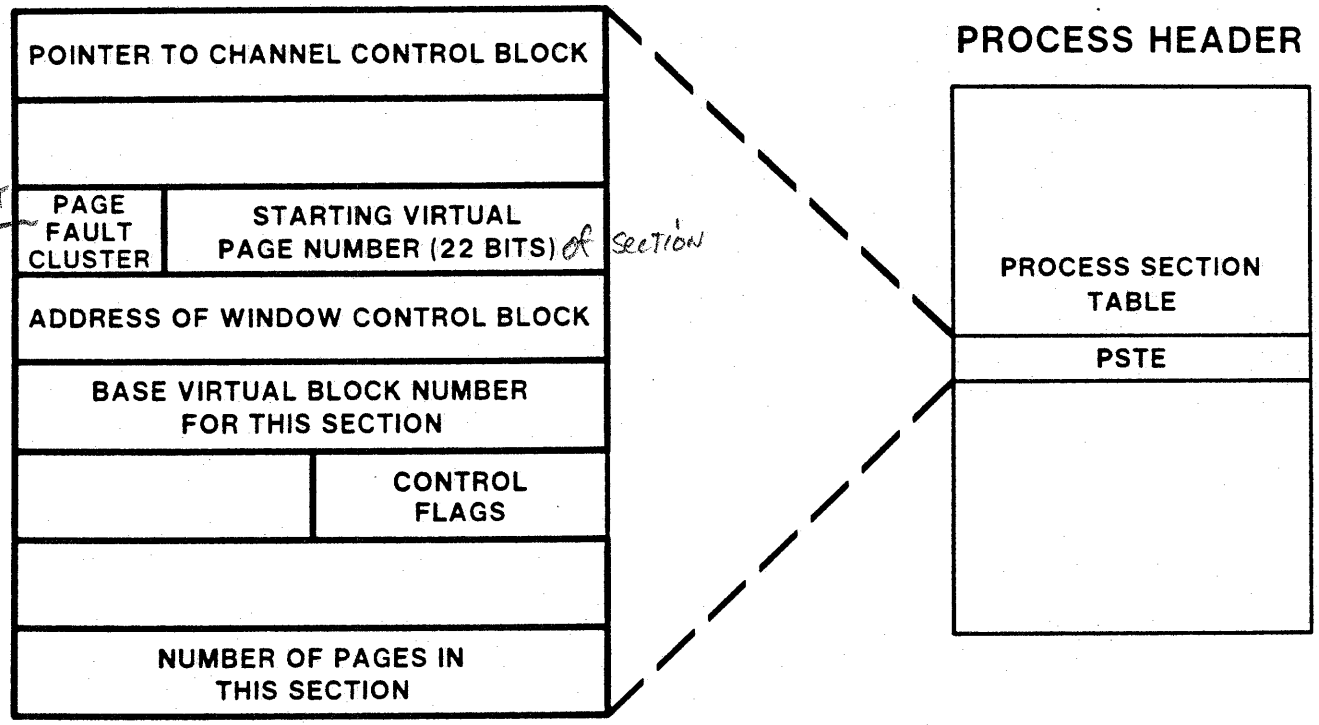
+ Base of page table
= Process page table entry
if not valid Then go into system page table

11-11 TO 11-13 IDSM

PROCESS SECTION TABLE



PROCESS SECTION TABLE ENTRY



PFDefault usually 64 better 16-32

VMS uses pre-fetching

Same PTE

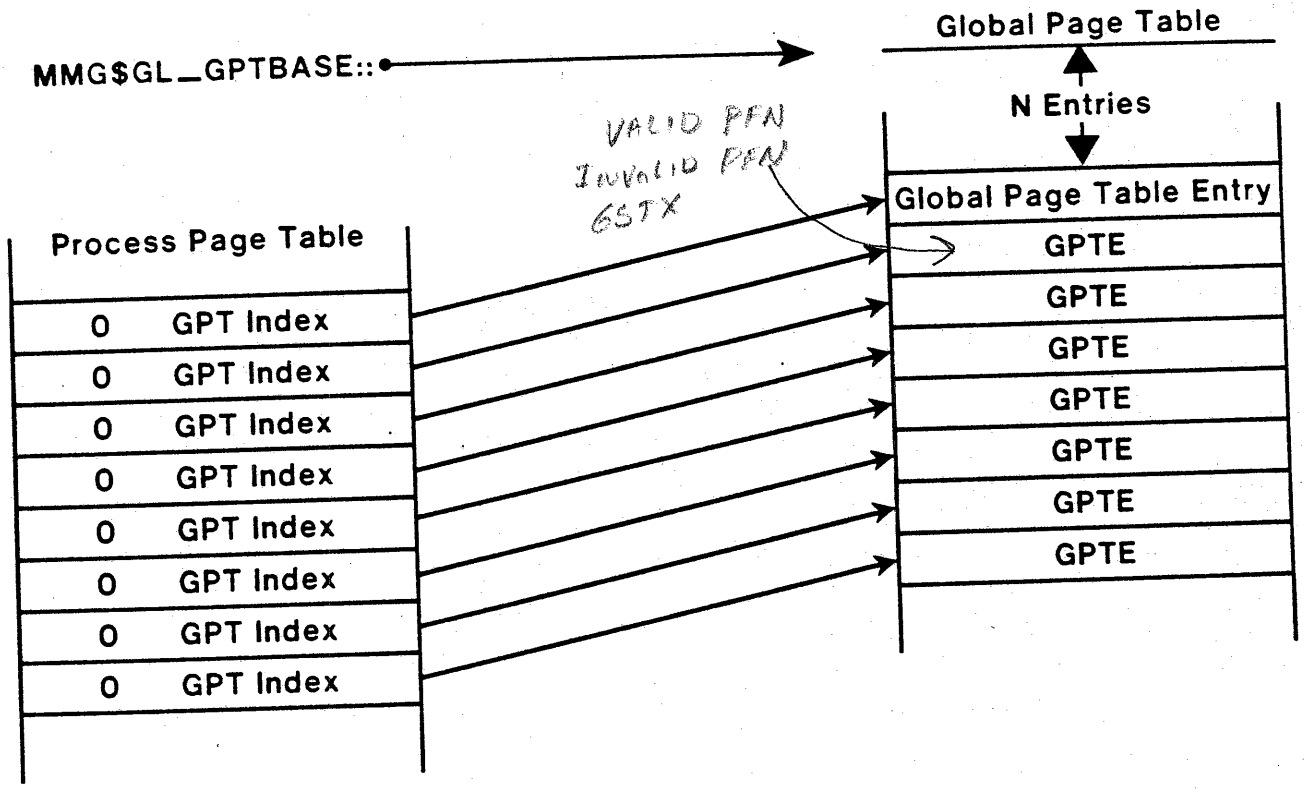
PTE
PTE
PTE
PTE
PTE
Read cluster

STOP Clusters
- No more WSLEs
- No more phy. Pages avail.
- PT page NOT valid (page table)
- max. cluster size
- Different INDEX

Same OSTY
Same GPTX
Same Page File INDEX

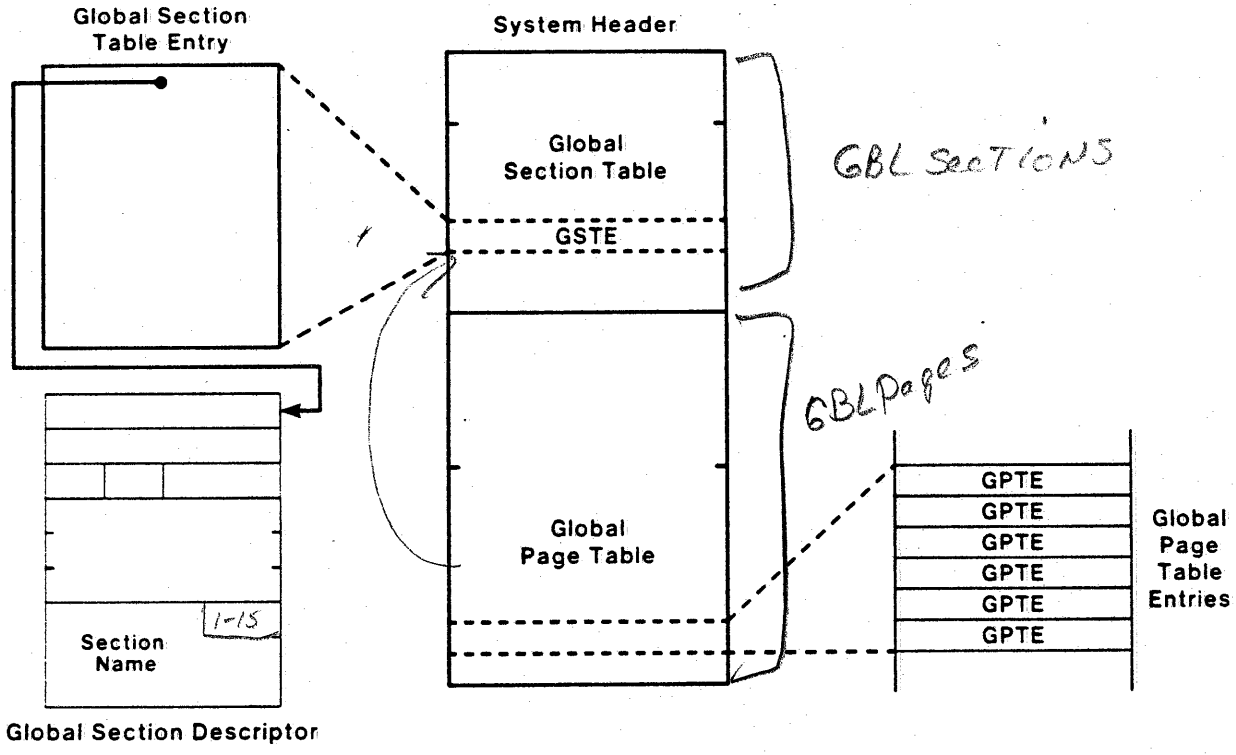
GLOBAL PAGES

PROCESS PTEs MAP TO GLOBAL PTEs

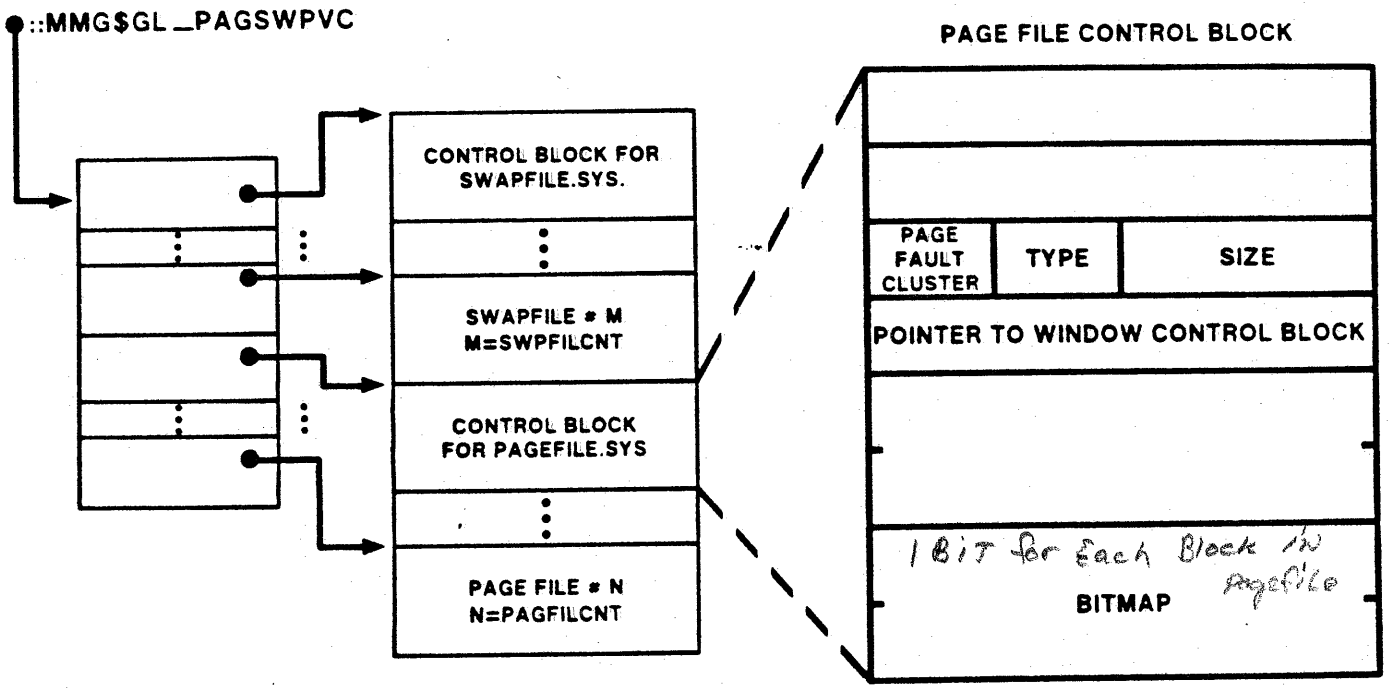


RELATIONSHIP AMONG GLOBAL SECTION DATA STRUCTURES

$CRMPSC or INSTALL



PAGE FILE CONTROL BLOCK



PFN ARRAYS

(1) BAK - • WHERE PAGE SHOULD GO IF IT IS BE USED

• VALUE THAT IS PUT BACK INTO THE PTE

• STORES ORIGINAL PTE CONTENTS

• POSSIBLE VALUES:

- PFLINK
- PSTX
- GSTX

(2) PTE - SVA OF PTE THAT MAPS A PHYSICAL PAGE

(3) WSLX - • INDEX INTO PROCESS WS (OR SYSTEM) FOR VALID PAGES

• NOT USED FOR GLOBAL PAGES

• OTHER USE IF PAGE IS NOT ON FRL/MDL

(3) BLINK - • PAGES ON FRL/MDL ARE DOUBLY LINKED; BUT LINK CANNOT BE ON THE PAGE

• PHYSICAL PAGE NUMBER OF

③ FLINK • PREDECESSOR "LINK" AS DESCRIBED FOR BLINK

③ SHRCNT • FOR GLOBAL PAGES (NUMBER OF PROCESSES SHARING A PAGE)

- ACTUALLY # OF PROCESS PTEs MAPPED TO A PARTICULAR GLOBAL PAGE

④ REFCNT - REASONS WHY A PAGE SHOULD NOT BE PLACED ON FPL/MPL

• IF REFCNT = Ø, PAGE PLACED ON FPL/MPL DEPENDING ON SAVED MODIFY BIT (FROM PTE), IN STATE ARRAY

• AFFECTED BY SHRCNT

• REASONS:

- PAGE IN WS
- SHRCNT > Ø
- LOCKED PAGE FOR DIRECT I/O

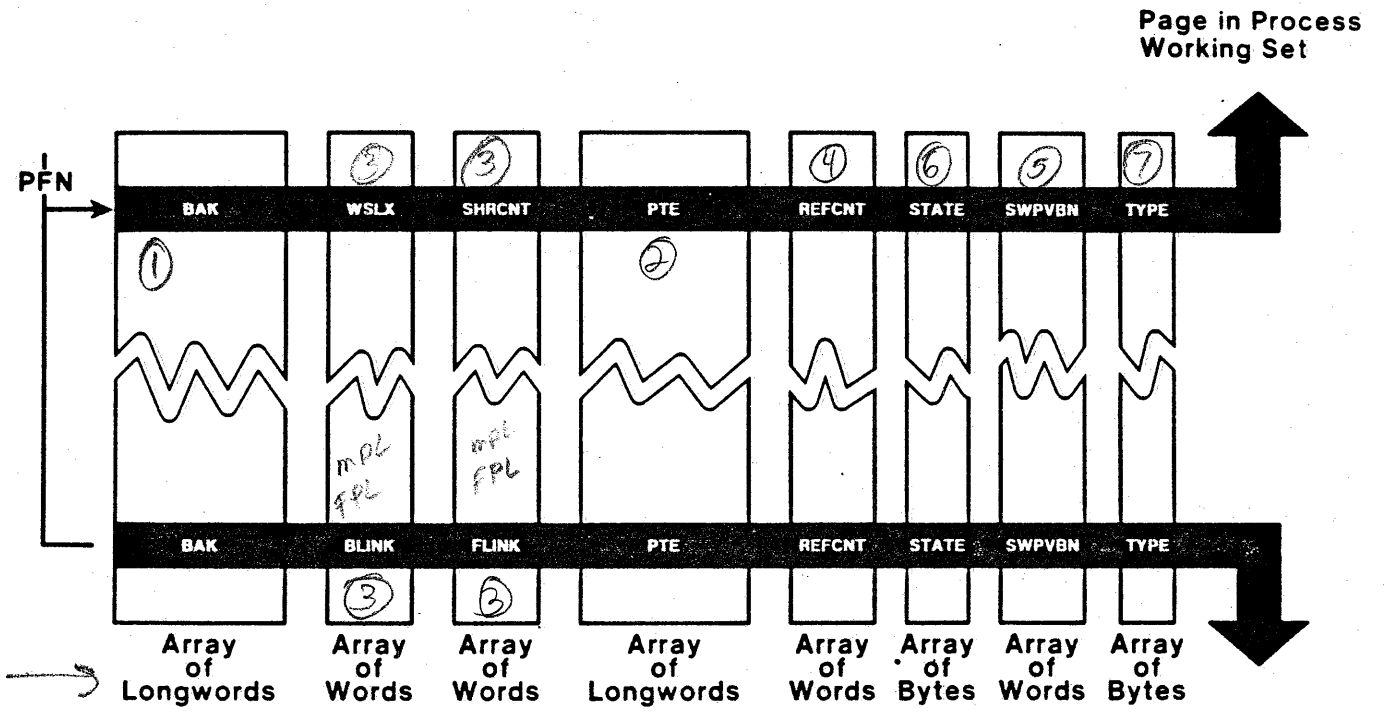
(5) SWPVBN - • SWAP FILE VBN
• SUPPORTS PROCESS OUTSWAP WITH I/O IN PROGRESS
• RECORDS VBN IN SWAPFILE FOR PAGE THAT IS LOCKED DOWN
• MDW DIVERTS PAGE FROM "NORMAL" BACKING STORE TO SWAP FILE BLOCK

(6) STATE - • SPECIFIES ACTIVITY/LOCATION OF EACH PHYSICAL PAGE
• UPPER BIT INDICATES FPL/MPL WHEN PAGE IS RELEASED (MODIFY BIT FROM PTE)

(7) TYPE - • DIFFERENT TYPES OF VALID PAGES
• PAGER/SWAPPER ACTIONS DEPEND ON TYPE OF PAGE
TYPES ON 11-18 IDSM

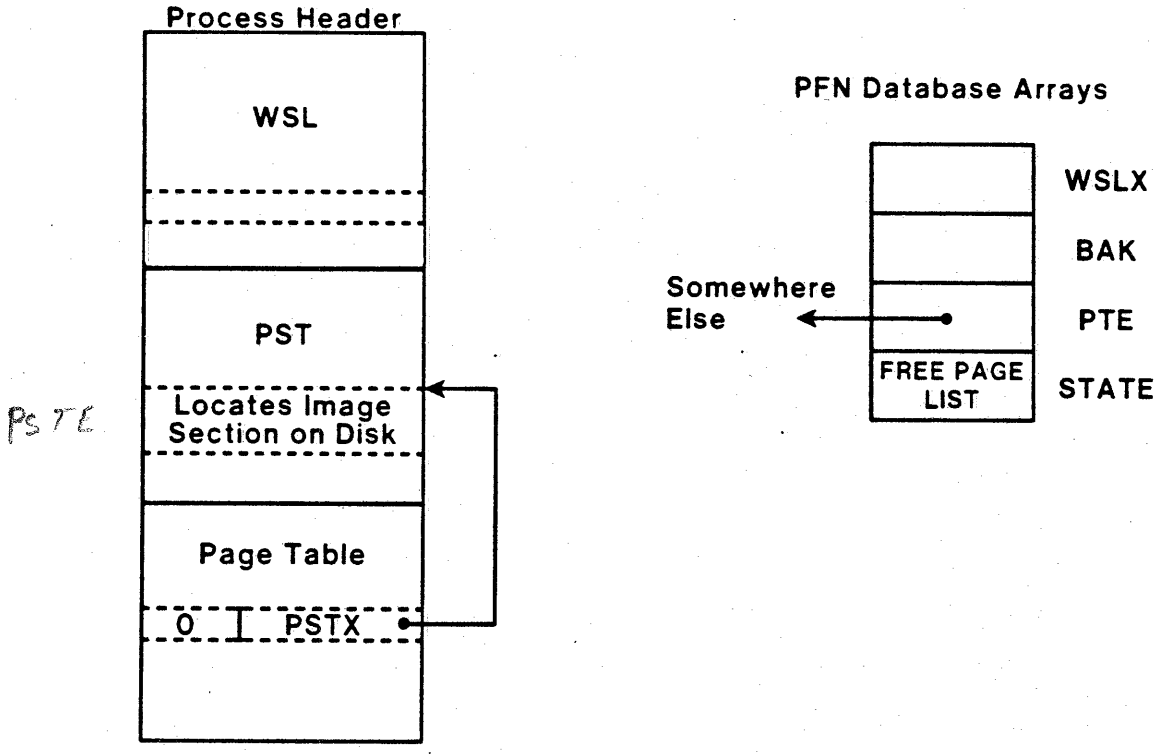
PFN DATABASE

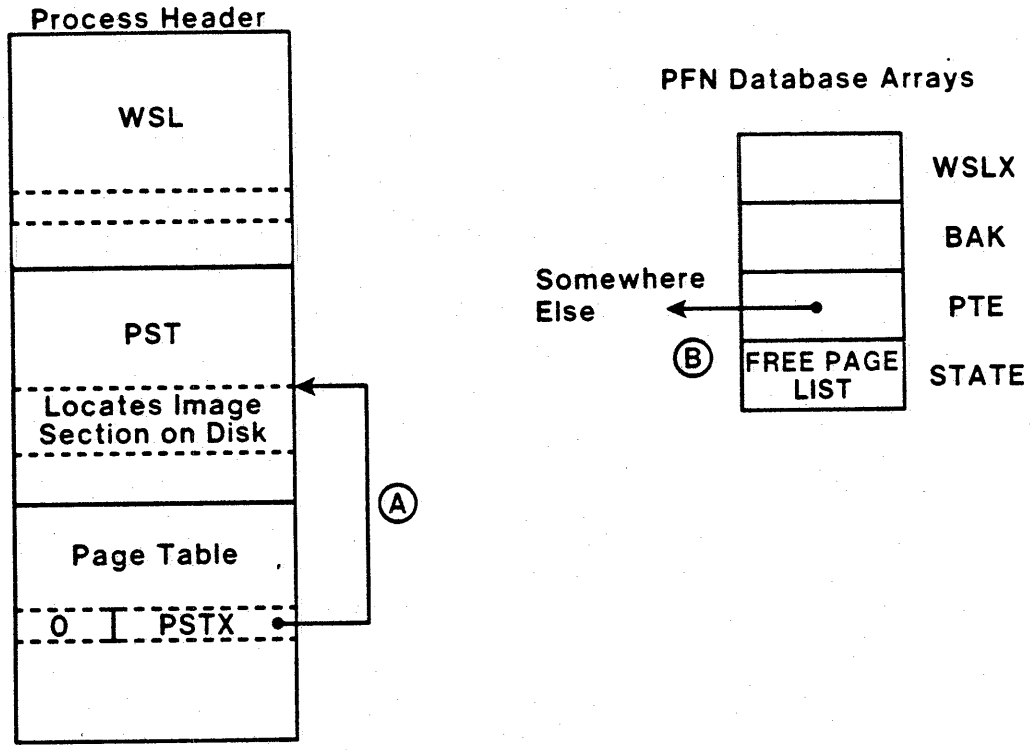
$PFNDEF



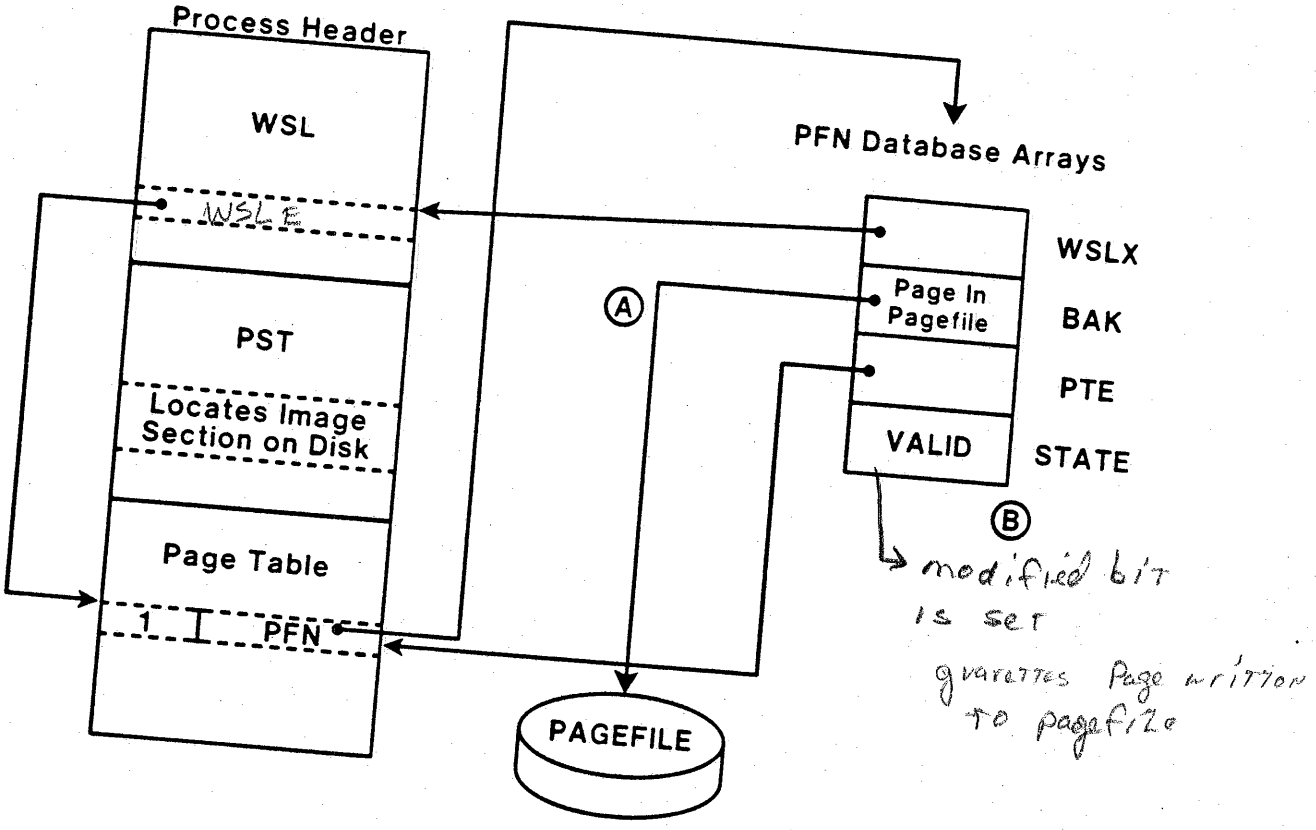
Total of 18 bytes per page

INITIAL STATUS OF PROCESS COPY-ON-REFERENCE PAGE

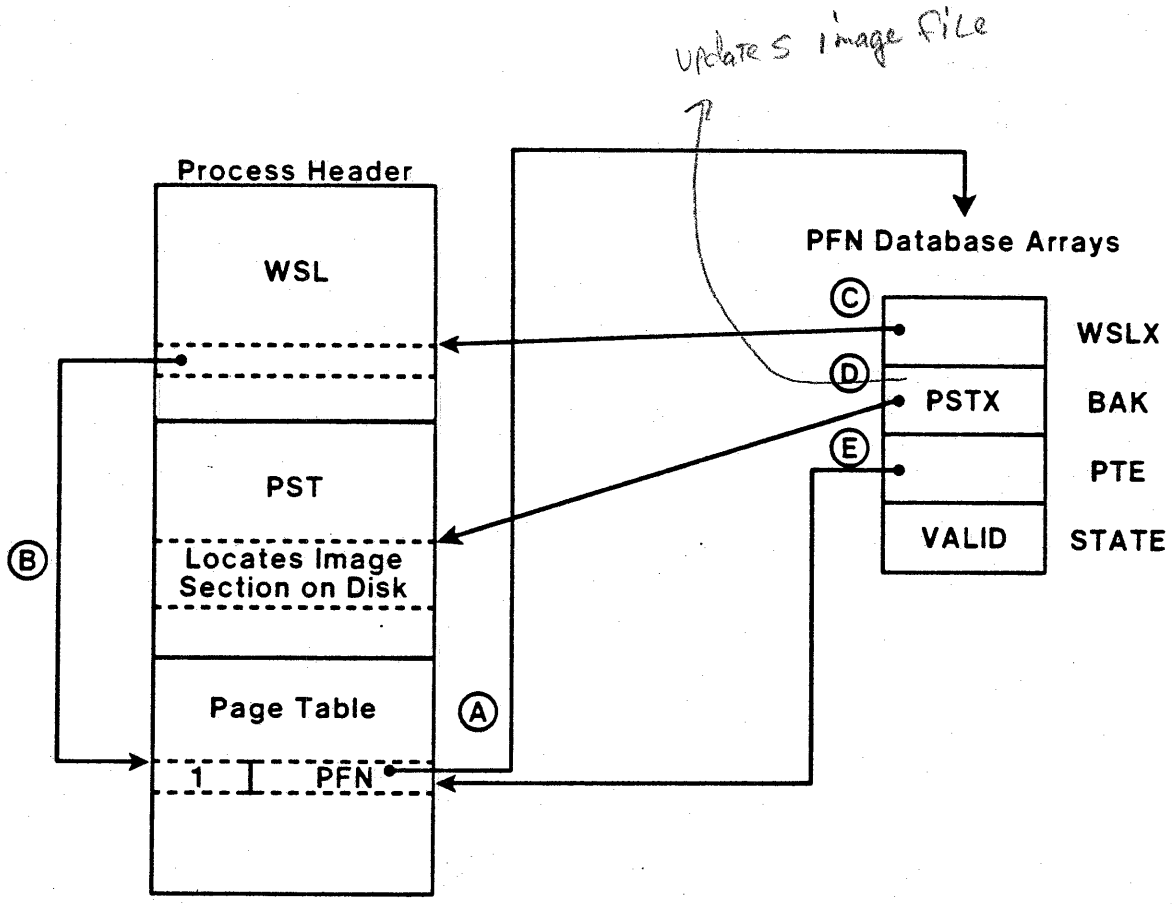



Process Header
WSL
PST
Locates Image Section on Disk
Page Table
0
PSTX
A
PFN Database Arrays
WSLX
BAK
PTE
STATE
Somewhere Else
B
FREE PAGE LIST

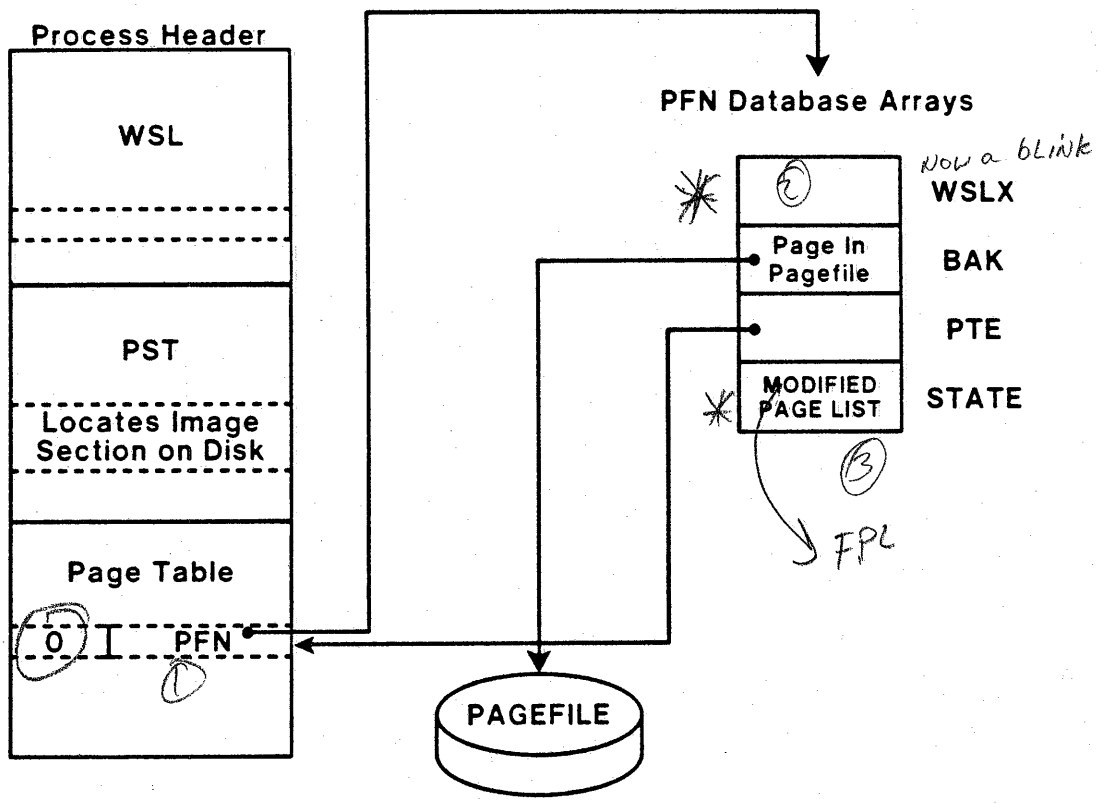
ADDING PROCESS COPY-ON-REFERENCE PAGE TO WORKING SET



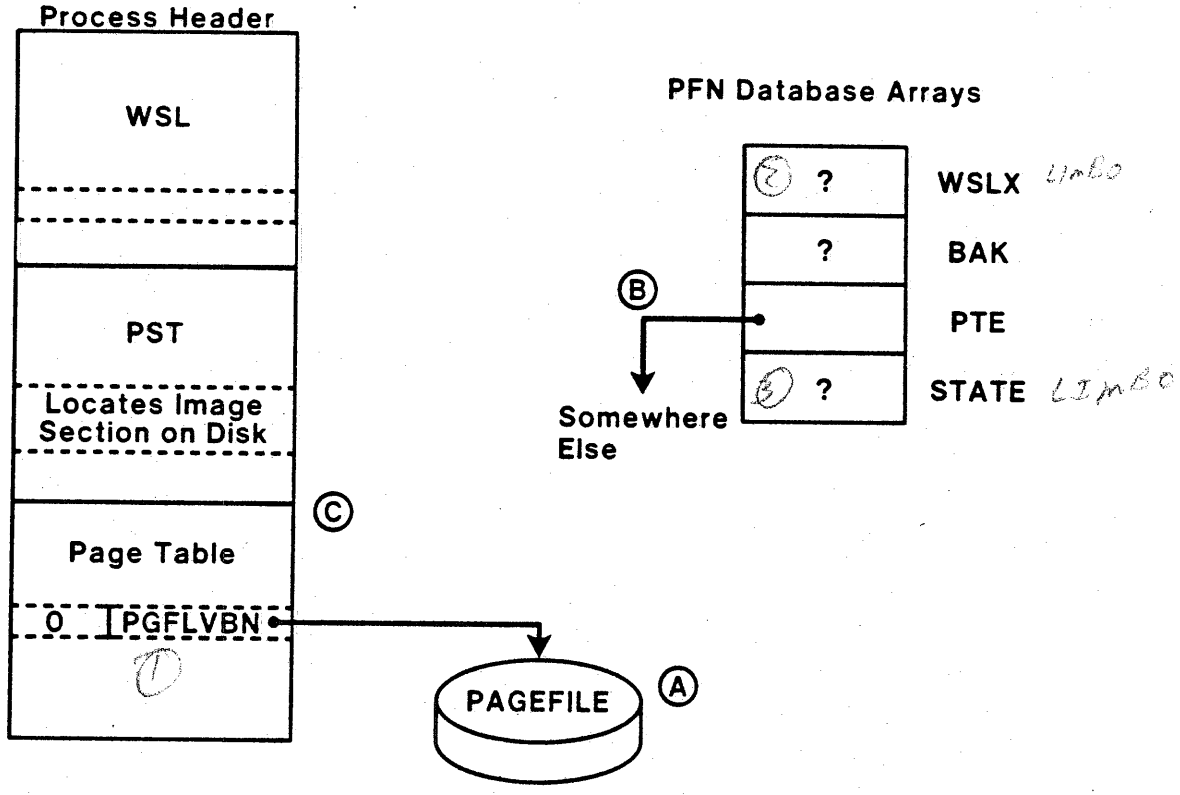
ADDING PROCESS READ/WRITE SECTION TO WORKING SET



REMOVING PROCESS CRF SECTION PAGE FROM WORKING SET

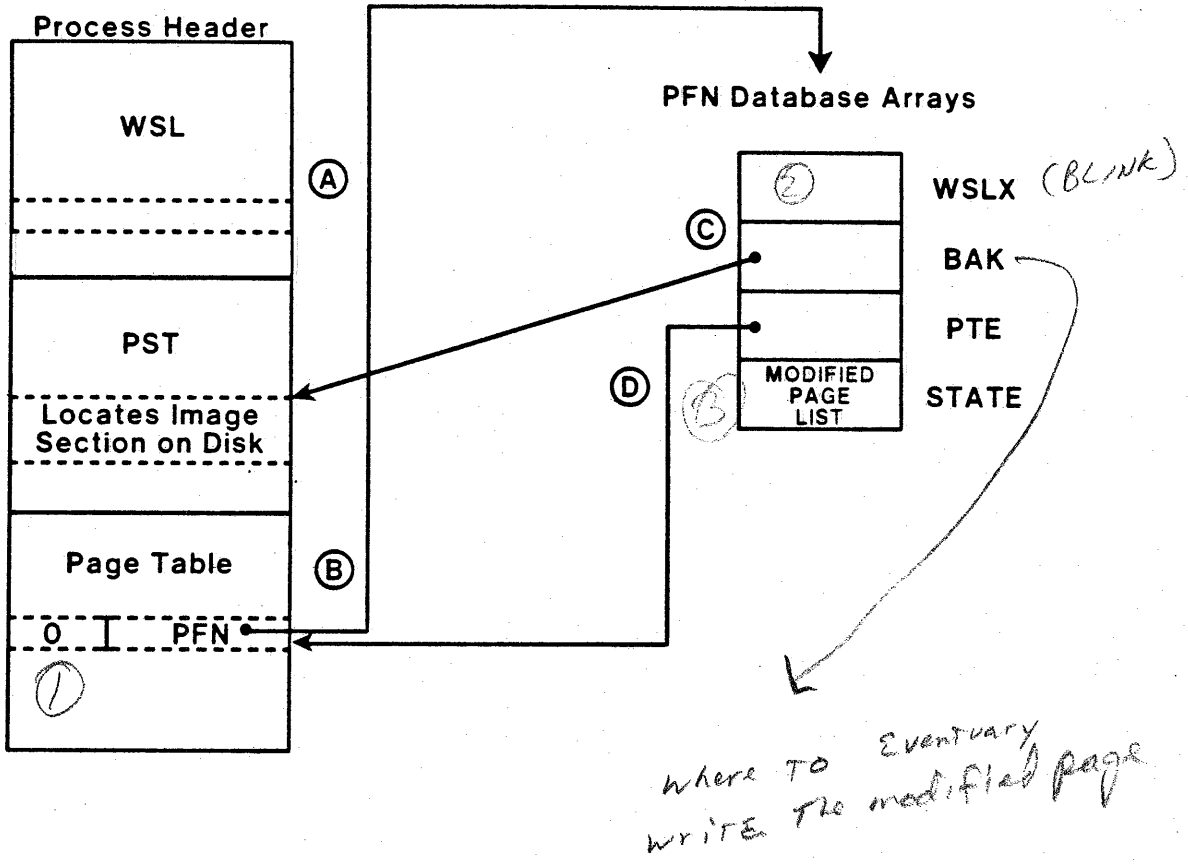


REMOVING PROCESS CRF PAGE FROM THE MPL (TO FPL)

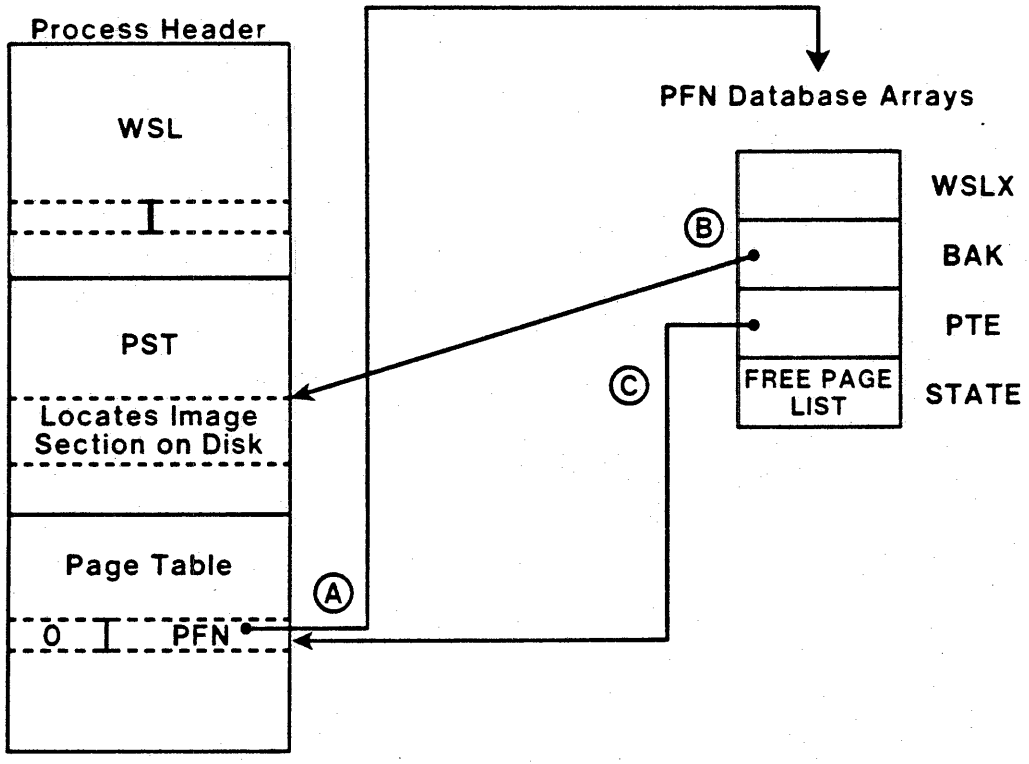


REMOVING PROCESS READ/WRITE SECTION FROM WORKING SET

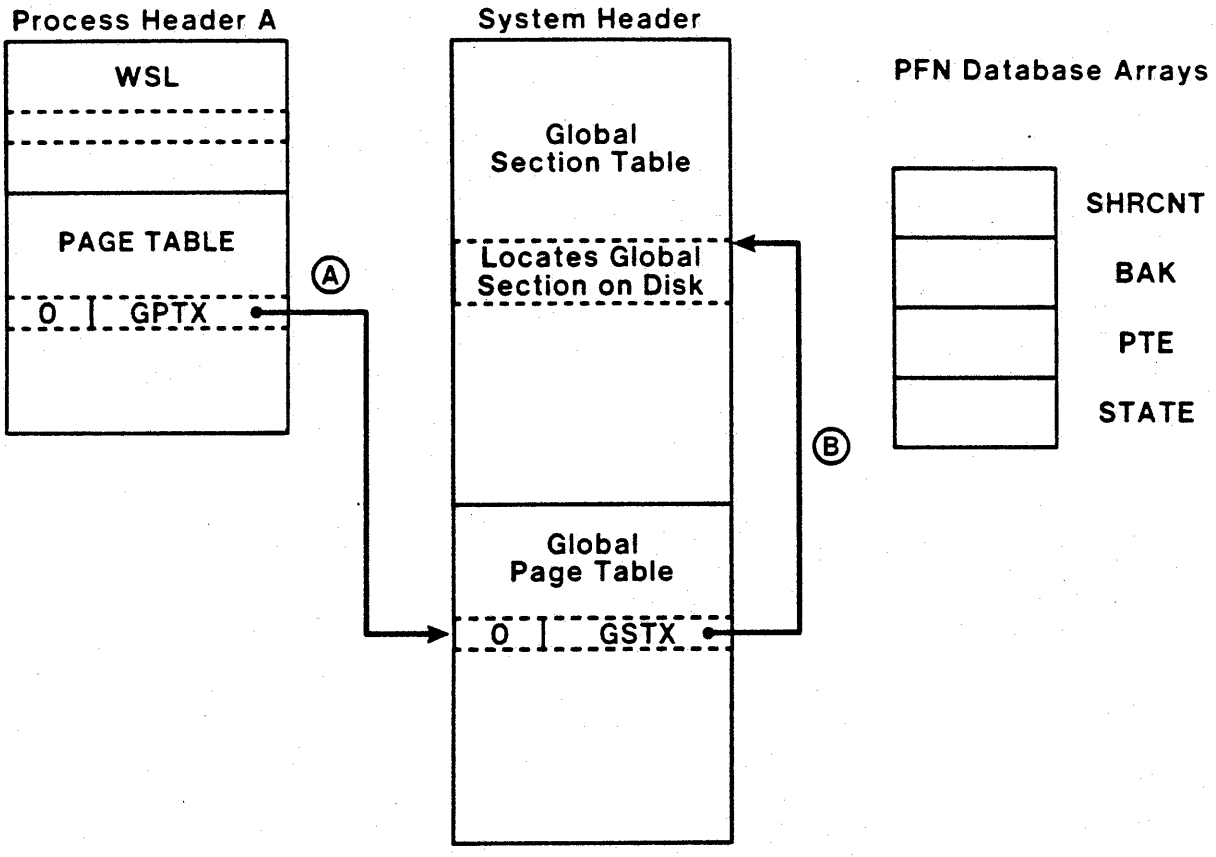
(MODIFIED PAGES)



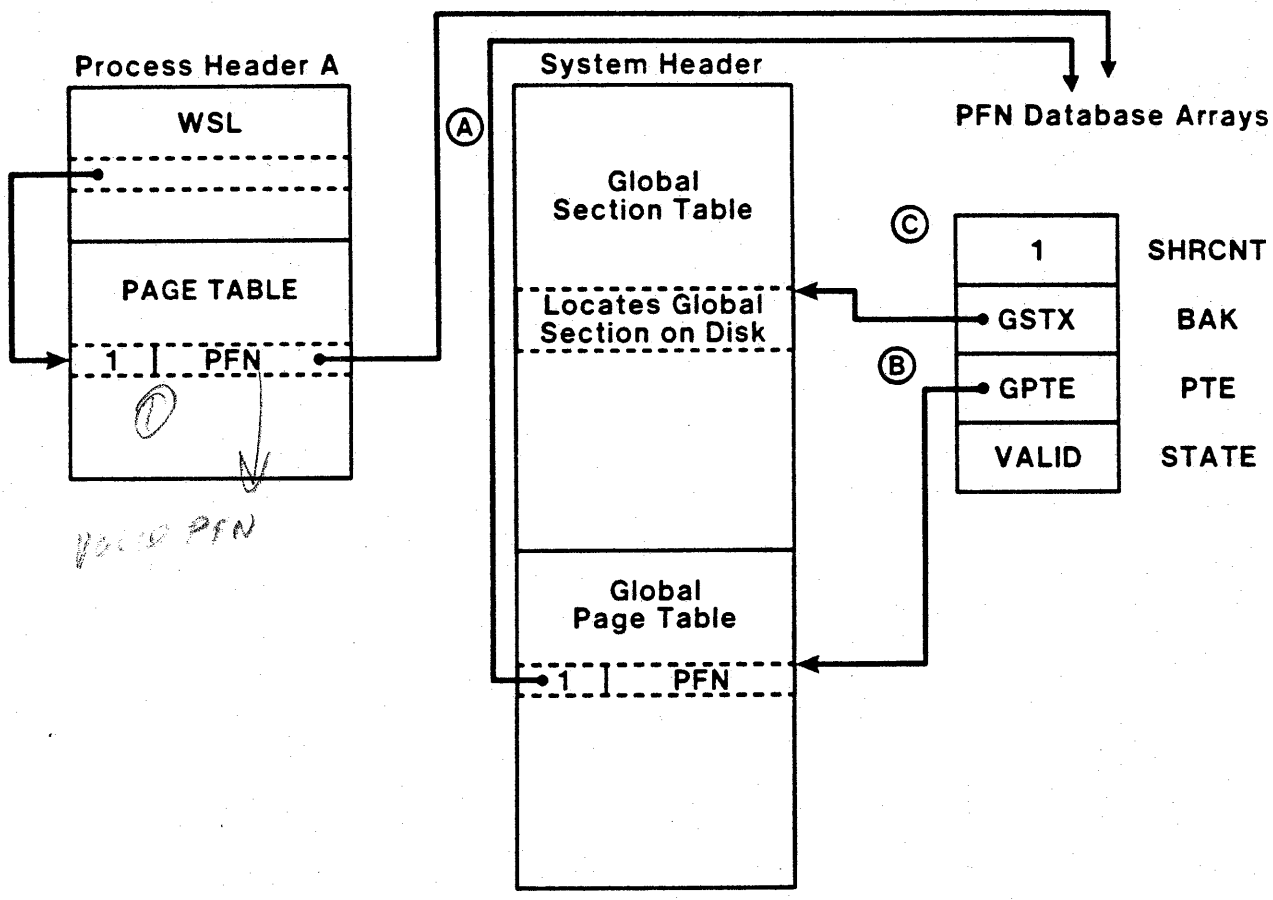
MOVING PAGE FROM MPL TO FPL

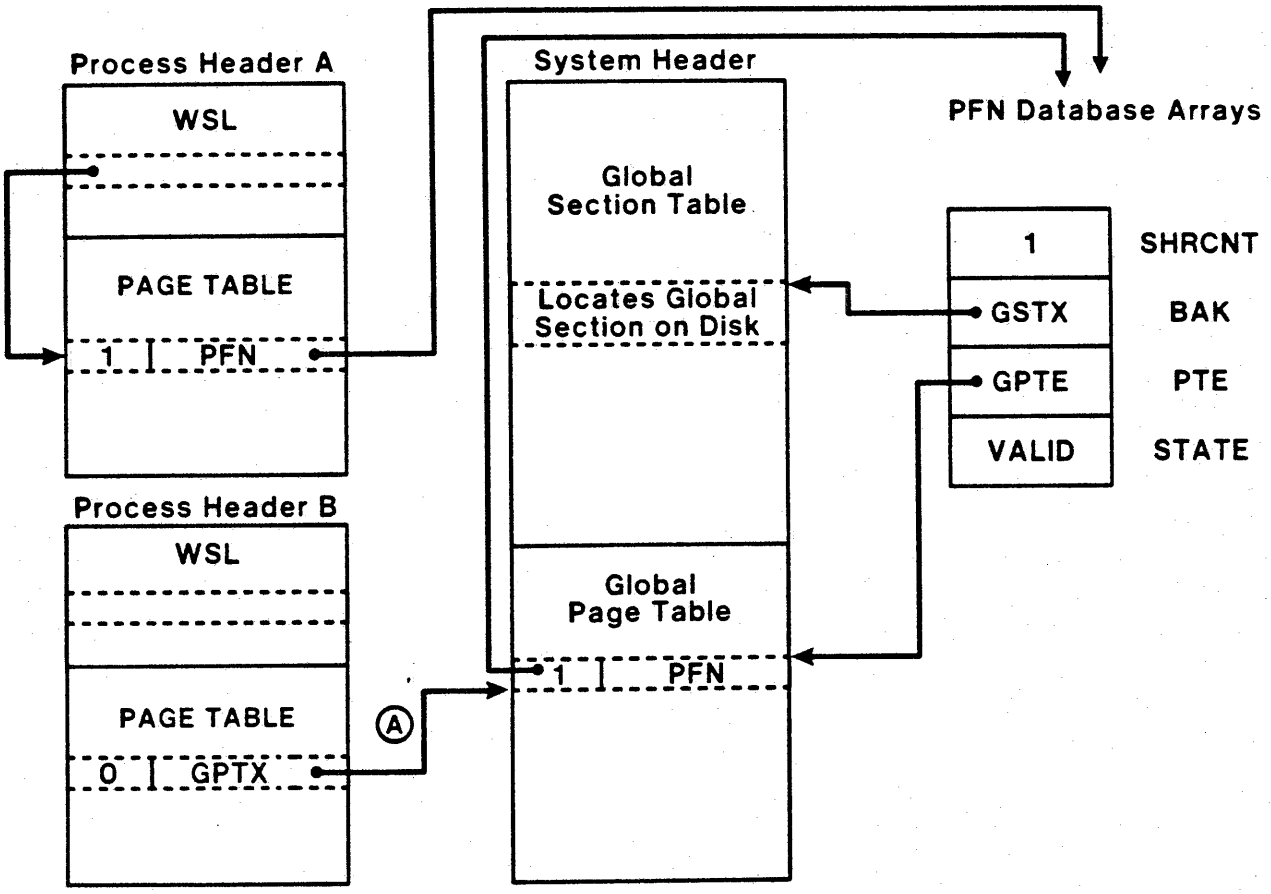


INITIAL STATE OF GLOBAL READ/WRITE SECTION PAGE

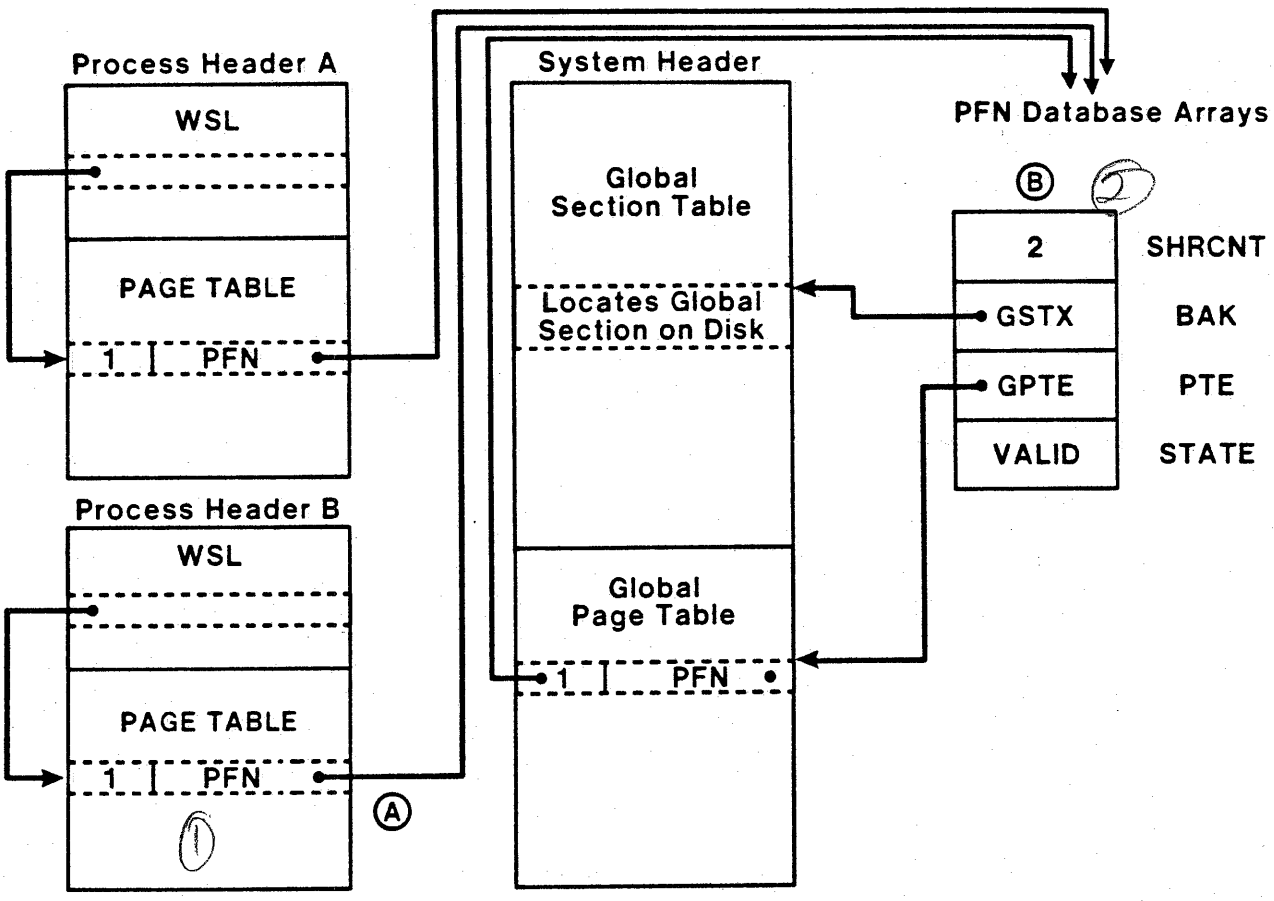


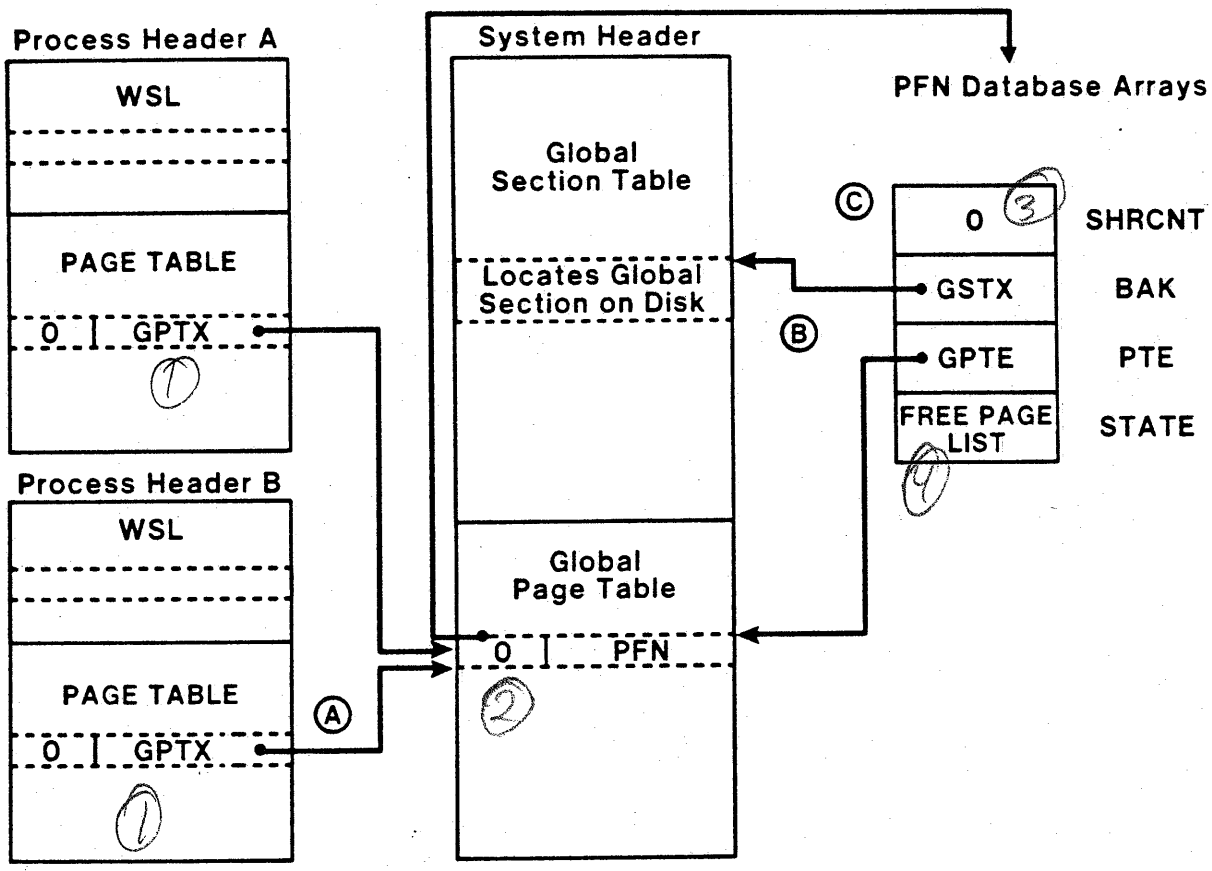
ADDING GLOBAL READ/WRITE SECTION PAGE TO WORKING SET




Process Header A
WSL
PAGE TABLE
1
PFN
Process Header B
WSL
PAGE TABLE
0
GPTX
A
System Header
Global Section Table
Locates Global Section on Disk
Global Page Table
1
PFN
PFN Database Arrays
1
SHRCNT
GSTX
BAK
GPTE
PTE
VALID
STATE

ADDING GLOBAL READ/WRITE SECTION PAGE TO SECOND WORKING SET




Process Header A
WSL
PAGE TABLE
0
GPTX
1
Process Header B
WSL
PAGE TABLE
0
GPTX
1
A
System Header
Global Section Table
Locates Global Section on Disk
B
Global Page Table
0
PFN
2
PFN Database Arrays
C
0
3
SHRCNT
GSTX
BAK
GPTE
PTE
FREE PAGE LIST
STATE
4

REMOVING GLOBAL READ/WRITE SECTION PAGE FROM FPL

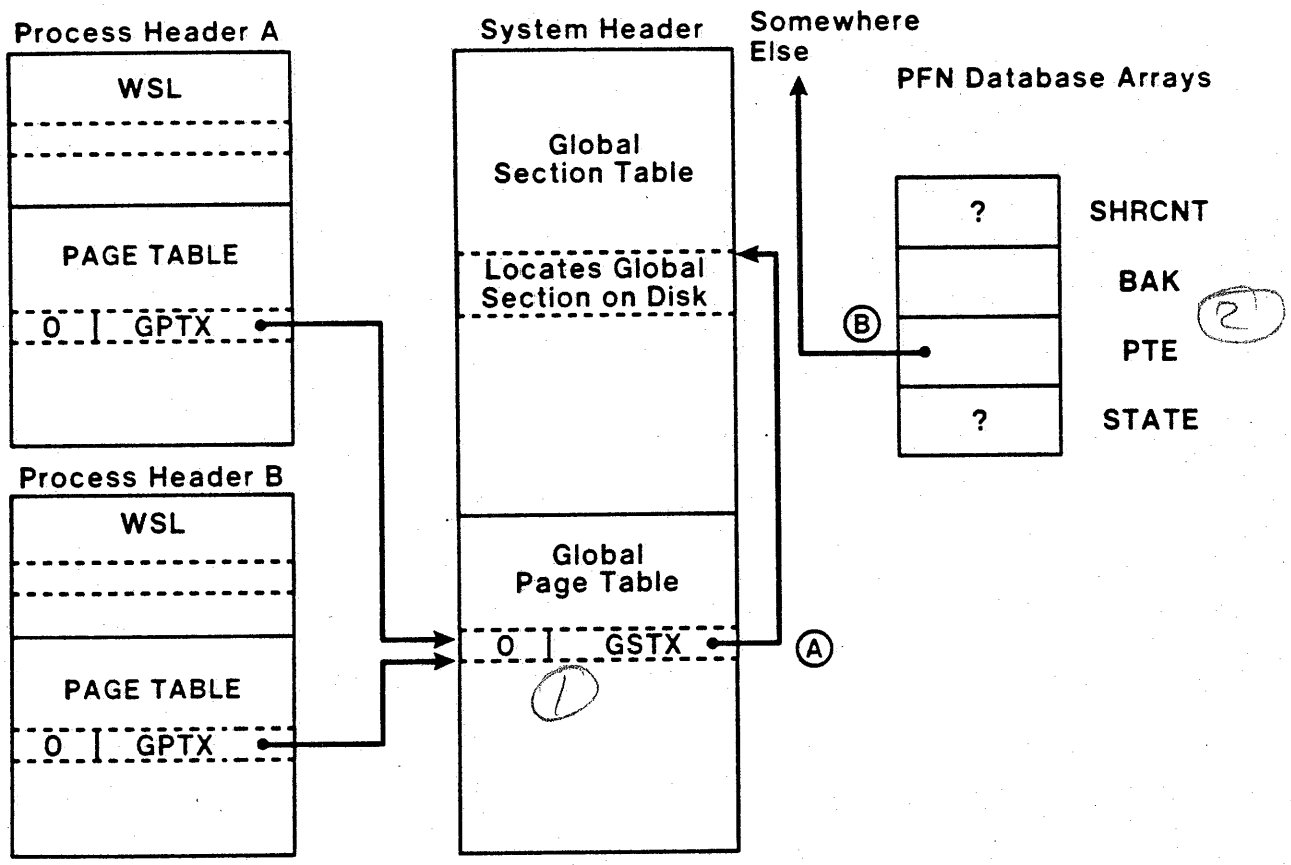


IMAGE ACTIVATION

OBJECTIVES

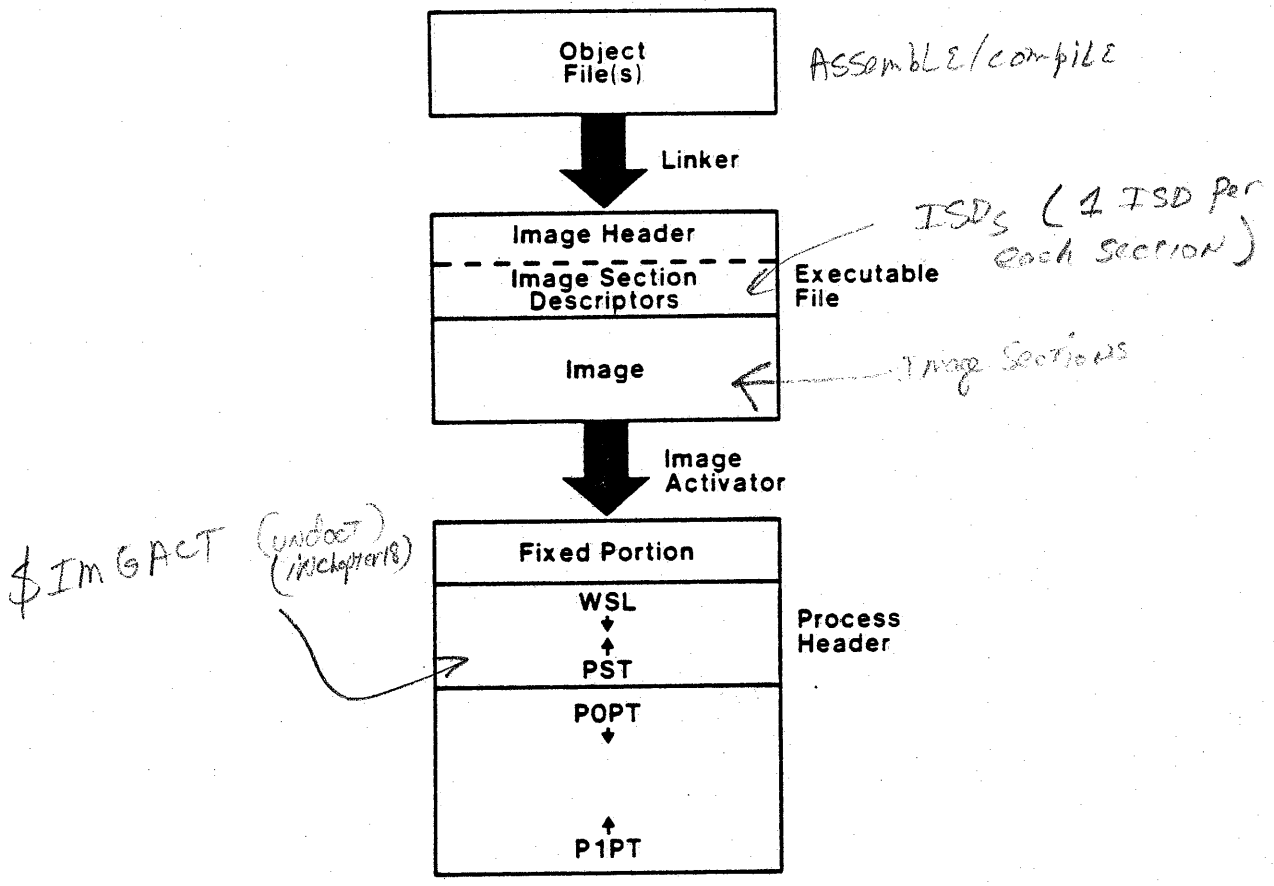
* discuss the steps involved in image activation
* discuss SYSGEN parameters and linker options that affect images

READINGS

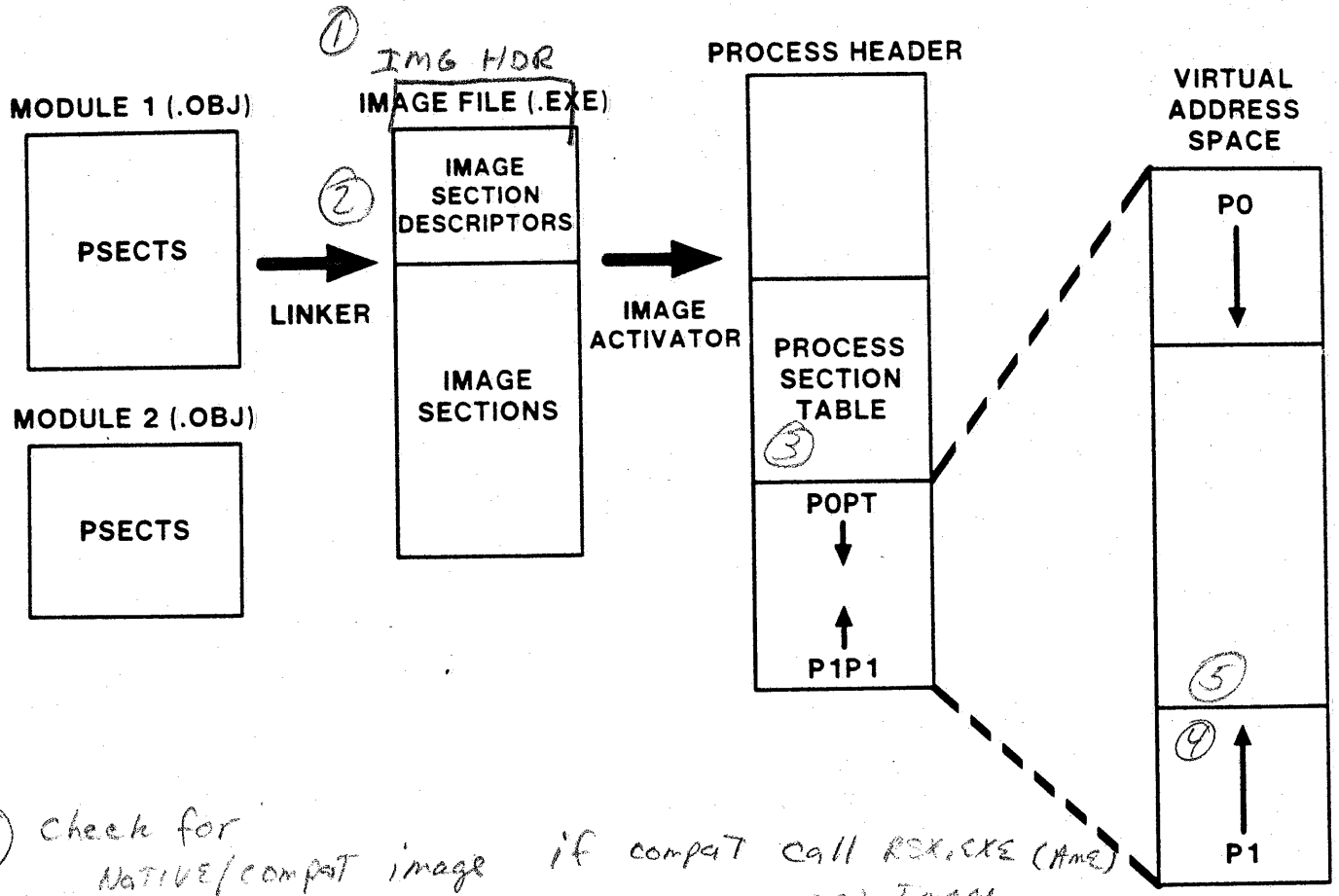
* IDSM Chapter 18

18-3 to 18-4 IDSM

STEPS IN IMAGE ACTIVATION



SEQUENCE OF EVENTS OVERVIEW



(1) Check for NATIVE/compat image if compat call RSX.EXE (name)
- map image
- set exit PSL/REI
- field all exceptions and Directives

(2) Read each ISD's and set PTE's for pages

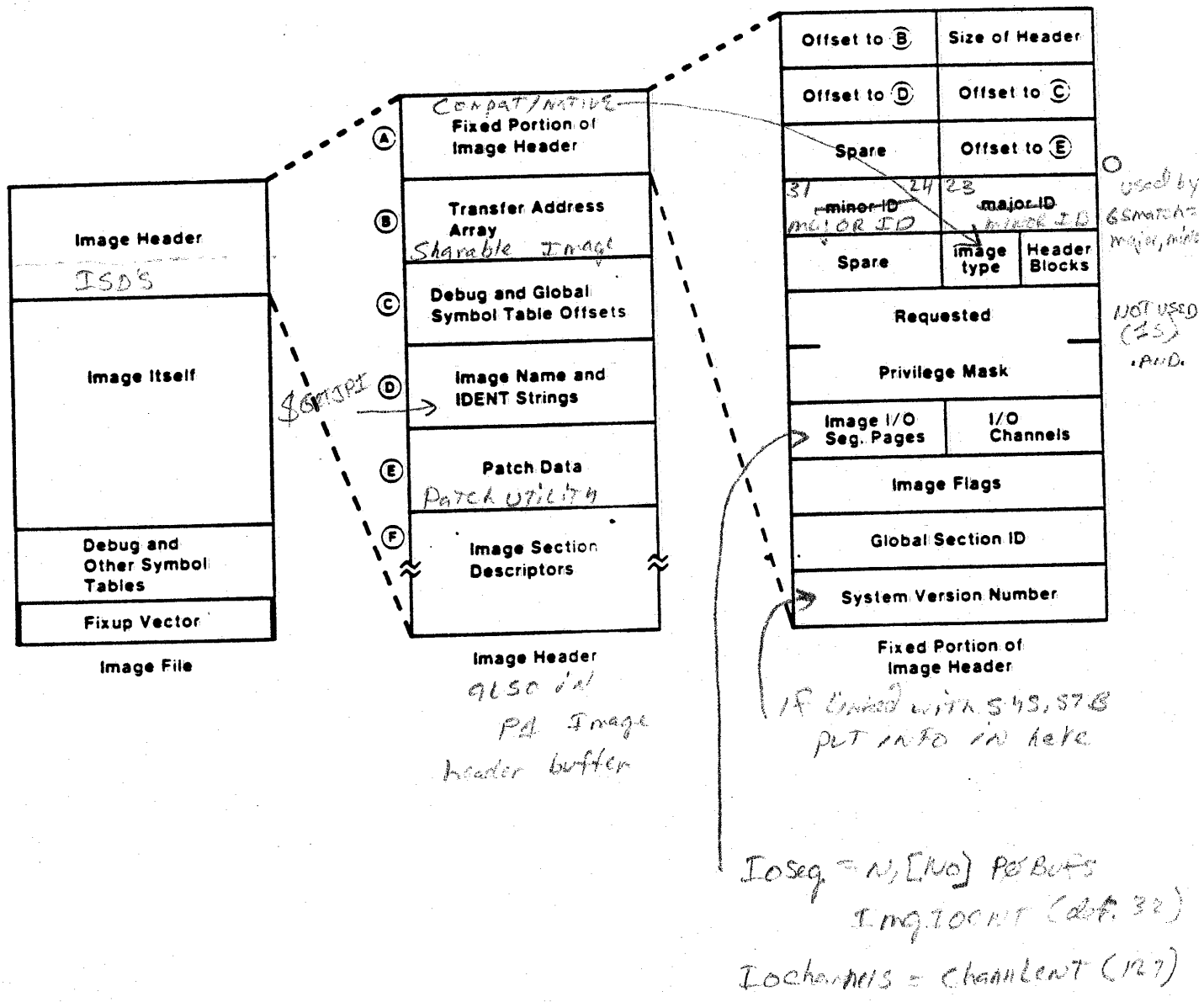
(3) SETUP PSTEs (from ISDs)
if DZRO Section found
call $CRETVA

(4) Fill in P1 Image I/O Segment
(RMS information)

(5) "Create" User stack + set USP. (User stack PTR.)

(6) Check to see if Image was installed with privs
- If so, "Enhance" Process Privs (PHD .OR. Image mask)

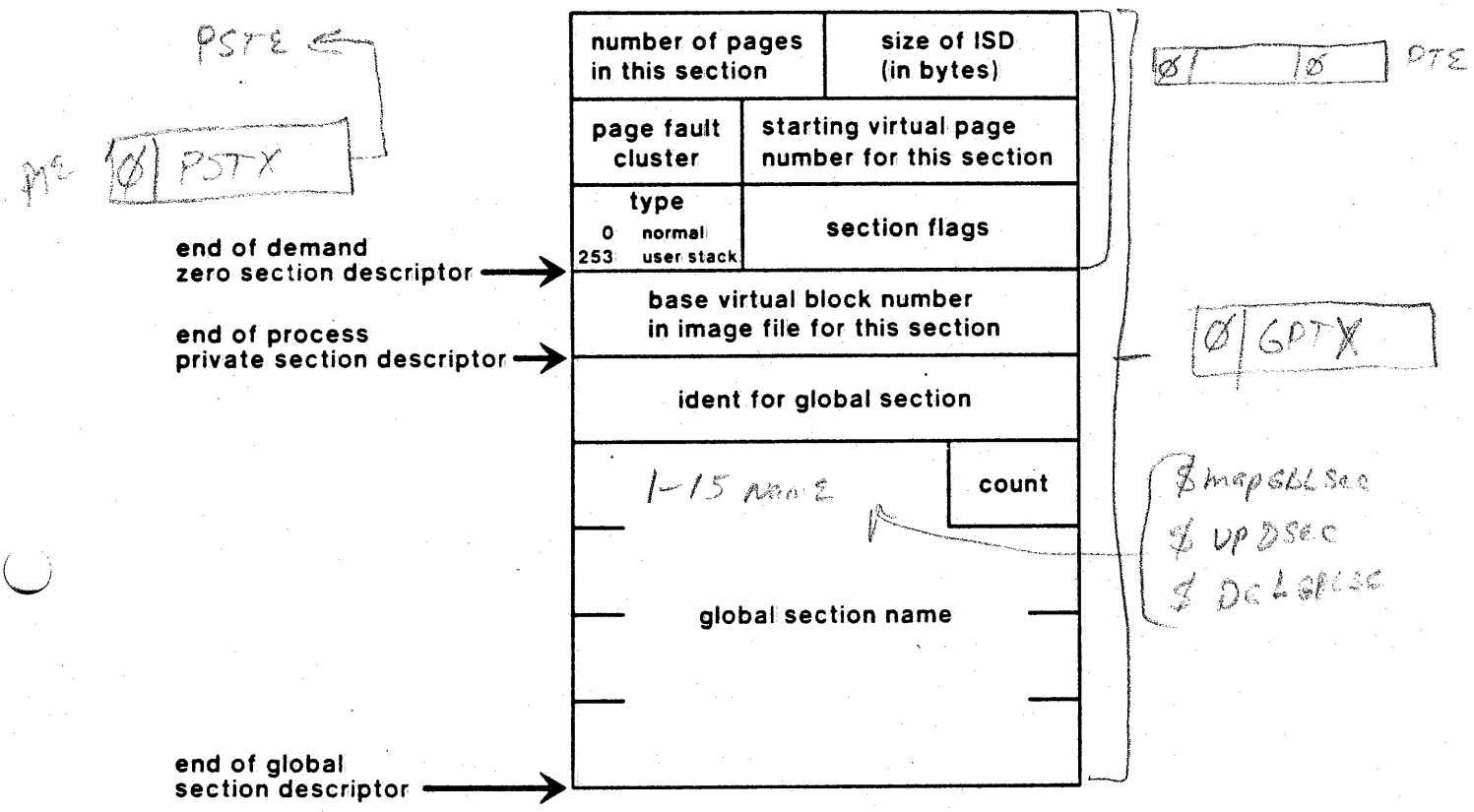
CONTENTS OF THE IMAGE HEADER



GSD A-7

ISD A-11

IMAGE SECTION DESCRIPTOR



$IMGACT → PCEMRSC

SET UP GSTX IN GPTE

IMAGE HEADER

Fixed Header Information

image format major id: 02, minor id: 04
header block count: 1
image type: executable (IHD$K_EXE) → Sharable
I/O channel count: default
I/O page count: default
linker flags:

(0)	IHD$V_LNKDEBUG	0
(1)	IHD$V_LNKNOTFR	0
(2)	IHD$V_NOPOBUFS	0
(3)	IHD$V_PICIMG	1
(4)	IHD$V_POIMAGE	0

system version: X1JY ← Blank if not linked

Image Activation Information

first transfer address: %X'7FFEDF68'
second transfer address: %X'00000400'
third transfer address: %X'00000000' ← debugger

Global Symbol Table & Debug Symbol Table Information

debug symbol table VBN: 5, block count: 1
global symbol table VBN: 0, record count: 0

Image Identification Information

image name: "CEB"
image file identification: "0"
link date/time: 4-DEC-1982 18:40:58.41
linker identification: "03-16"

Patch Information

There are no patches at this time.

Image Section Descriptors (ISD)

1) image section descriptor (16 bytes).
page count: 1
base virtual address: %X'00000200' (P0 space)
page fault cluster size: default
ISD flags:

(0)	ISD$V_GBL	0
(1)	ISD$V_CRF	1
(2)	ISD$V_DZRO	0
(3)	ISD$V_WRT	1
(7)	ISD$V_LASTCLU	1
(8)	ISD$V_COPYALWAY	0
(9)	ISD$V_BASED	0
(10)	ISD$V_FIXUPVEC	0
(17)	ISD$V_VECTOR	0

DATA RSECT

```
                (18) ISD$V_PROTECT     0
            section type: ISD$K_NORMAL
            base VBN: 2

    2)  image section descriptor (16 bytes)
            page count: 1
            base virtual address: %X'00000400' (P0 space)
            page fault cluster size: default
            ISD flags:
                    (0)  ISD$V_GBL         0
                    (1)  ISD$V_CRF         0
                    (2)  ISD$V_DZRO        0
                    (3)  ISD$V_WRT         0
                    (7)  ISD$V_LASTCLU     1
                    (8)  ISD$V_COPYALWAY   0
                    (9)  ISD$V_BASED       0
                    (10) ISD$V_FIXUPVEC    0
                    (17) ISD$V_VECTOR      0
                    (18) ISD$V_PROTECT     0
            section type: ISD$K_NORMAL
            base VBN: 3

    3)  image section descriptor (16 bytes)
            page count: 1
            base virtual address: %X'00000600' (P0 space)
            page fault cluster size: default
            ISD flags:
                    (0)  ISD$V_GBL         0
                    (1)  ISD$V_CRF         1
                    (2)  ISD$V_DZRO        0
                    (3)  ISD$V_WRT         1
                    (7)  ISD$V_LASTCLU     0
                    (8)  ISD$V_COPYALWAY   0
                    (9)  ISD$V_BASED       0
                    (10) ISD$V_FIXUPVEC    1
                    (17) ISD$V_VECTOR      0
                    (18) ISD$V_PROTECT     0
            section type: ISD$K_NORMAL
            base VBN: 4

    4)  image section descriptor (12 bytes)
            page count: 20
            base virtual address: %X'3FFFD800' (P1 space)
            page fault cluster size: default
            ISD flags:
                    (0)  ISD$V_GBL         0
                    (1)  ISD$V_CRF         0
                    (2)  ISD$V_DZRO        1
                    (3)  ISD$V_WRT         1
                    (7)  ISD$V_LASTCLU     1
                    (8)  ISD$V_COPYALWAY   0
                    (9)  ISD$V_BASED       0
                    (10) ISD$V_FIXUPVEC    0
                    (17) ISD$V_VECTOR      0
                    (18) ISD$V_PROTECT     0
```

CODE

USER STACK

section type: ISD$K_USRSTACK

```
IMAGE ACTIVATOR FIXUP SECTION

        Fixed Information

                Flags:
                        (0)  IAF$V_SHR          0
                shareable image count: 1
                extra image count: 0

        Shareable Image List

                0)  this image

        Protection Change Fixups (relative to %X'00000200')

                address: %X'00000400', page count: 1
                protection: PRT$C_UREW


The analysis uncovered NO errors.

ANALYZE/IMAGE CEB.EXE
$
```

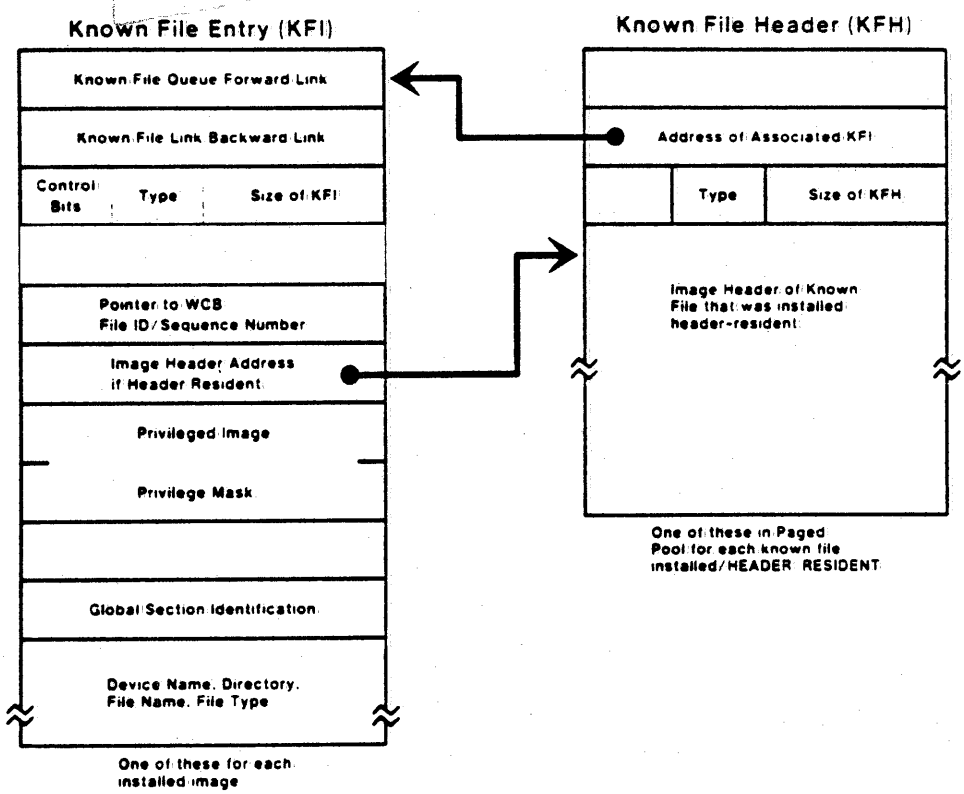
```
00:  ***
01:
02:
03:
04:
05:
06:
07:
08:
09:
0A:
0B:
0C:
0D:
0E:
0F:
10:
11:
12:
13:
14:
15:
16:
  PC: 7FFEE128  State: LEF     SYS$SYSDEVICE:[ELLIS]CEB.EXE;10
```

* = Valid page
L = Locked
G = Global

KNOWN FILE LISTS

A-16

A-15



PHD Mask, OR,

$CRMPSL → OPEN ← Shared

Header Resident

Directory Info Perm Avail.

Privileged

Protected (Limit in kernel mode)

SWAPPING

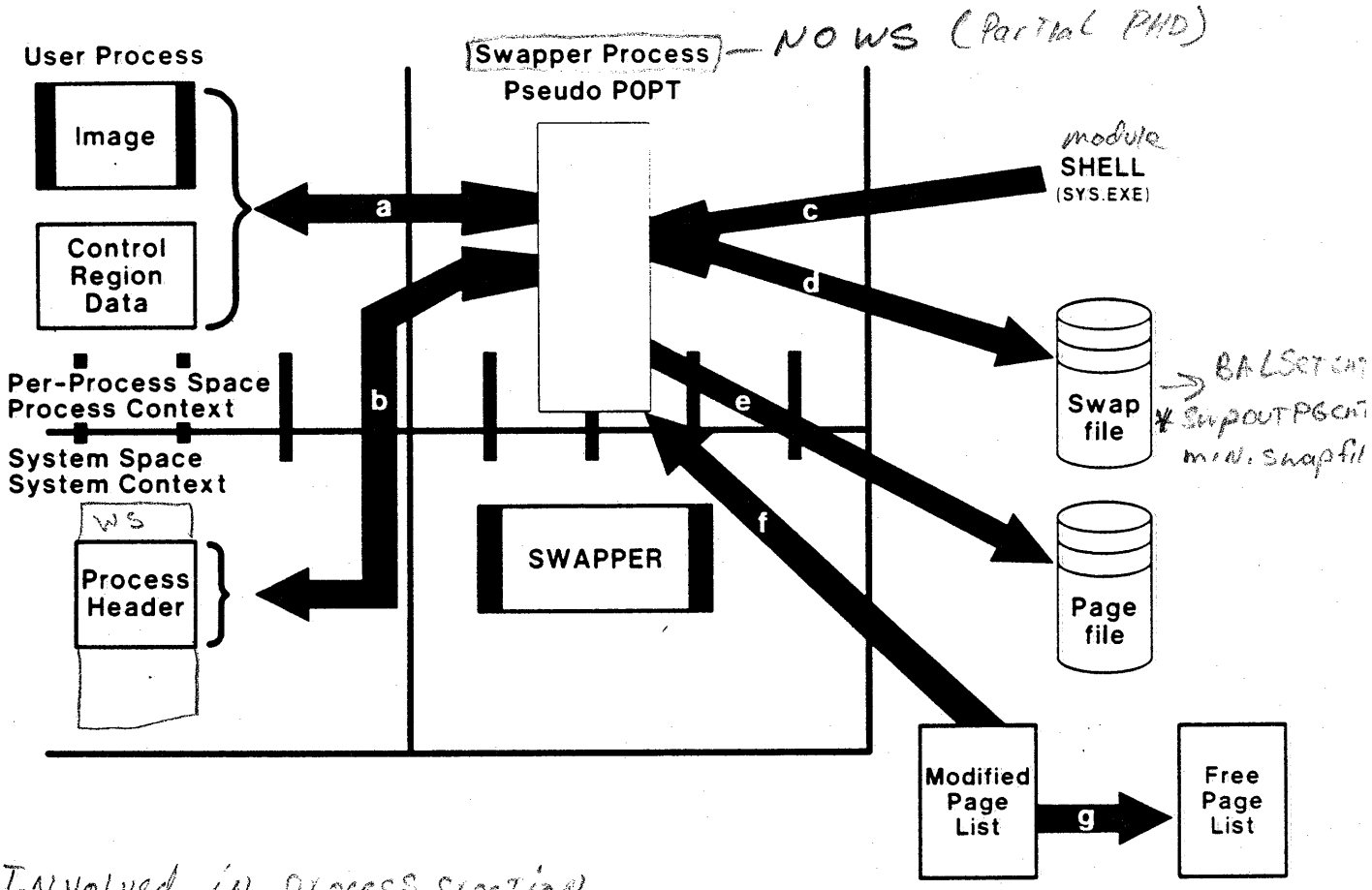
OBJECTIVES

* describe the swapping operations
* learn steps in creation of alternate swap file(s)
* see differences in swapper's use of PFN database from pager's

READINGS

* IDSM Chapter 14

OVERVIEW OF SWAPPER FUNCTIONS



Involved in process creation

C-a-b

Large Swapfile =
MAXprocesscnt * wsmax

Small swap file =
BalSetCnt * SwpOutPgCnt

OUT Swap

Need memory → ① JUST b(WS) - TO EITHER D, E

Need balance SLOT → ② b(rest PHD) - TO either D, E

IN Swap

① rebuild PTs/WSL

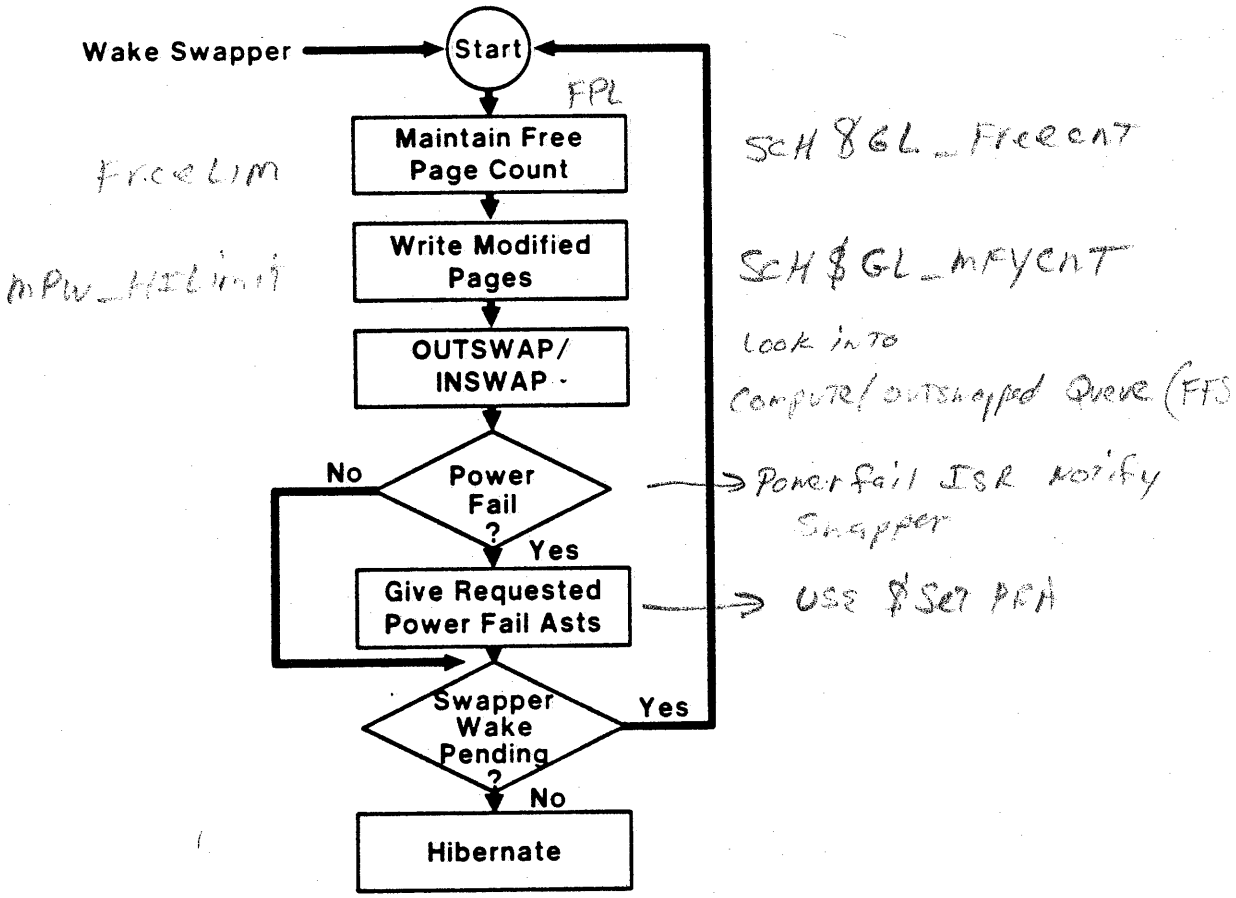
② rebuild PHD/PTs

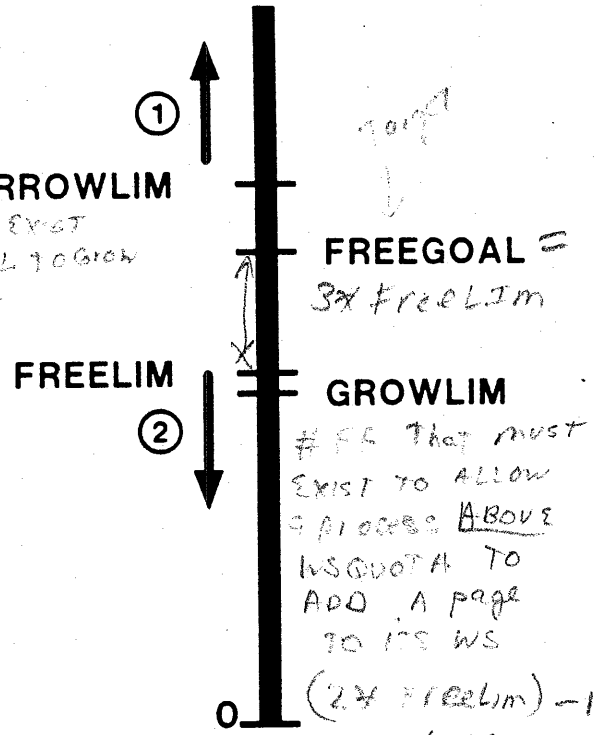
Small Page file

BalSetCnt * avg Proc Size

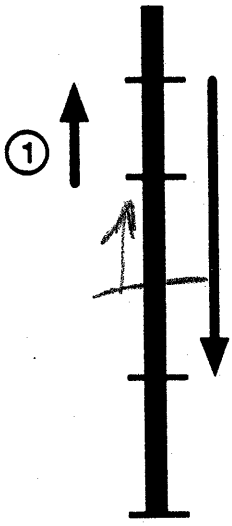
USE Autogen Page file =

SWAPPER CODE MAIN LOOP

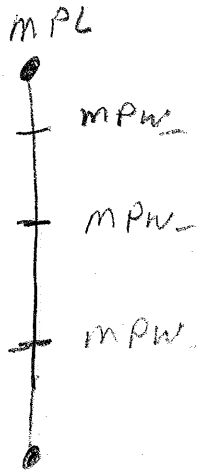




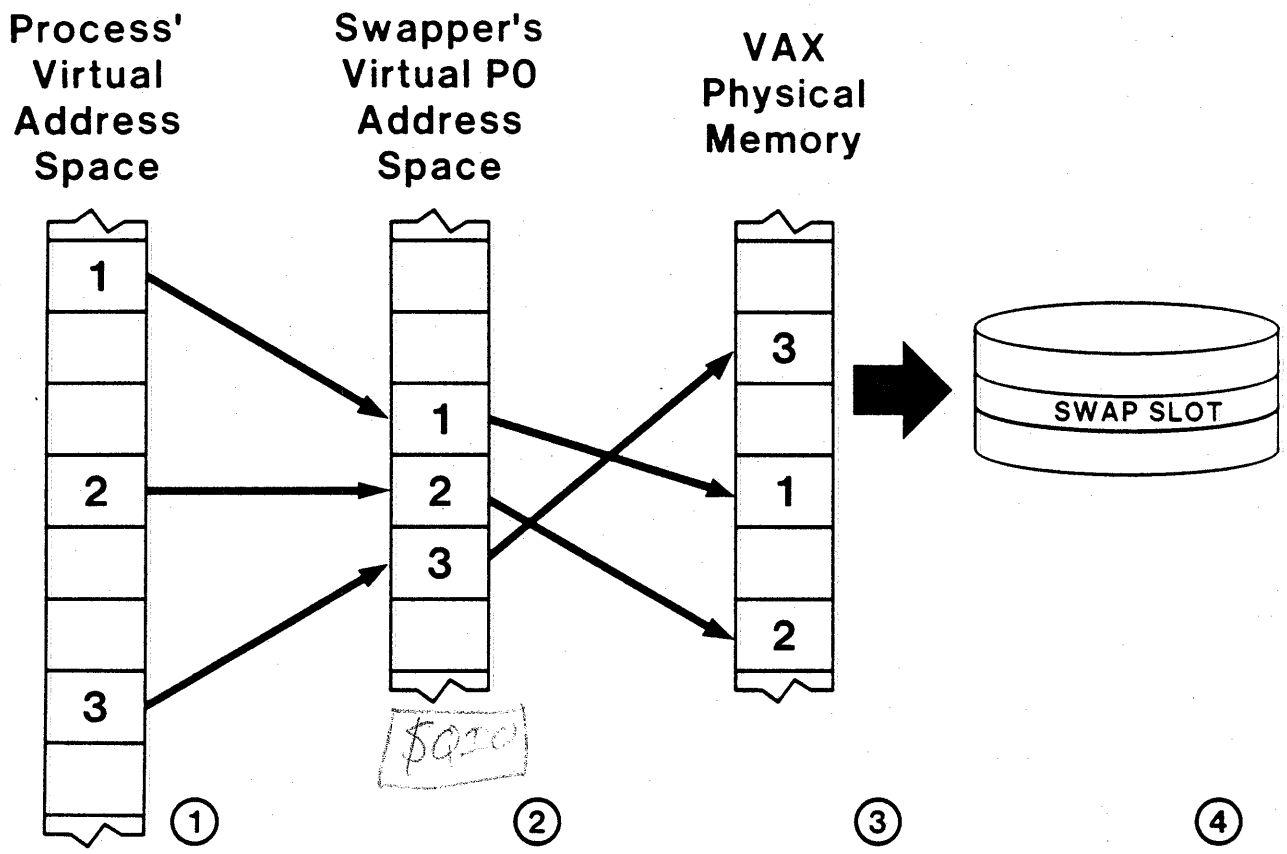
Number of Pages on Free Page List



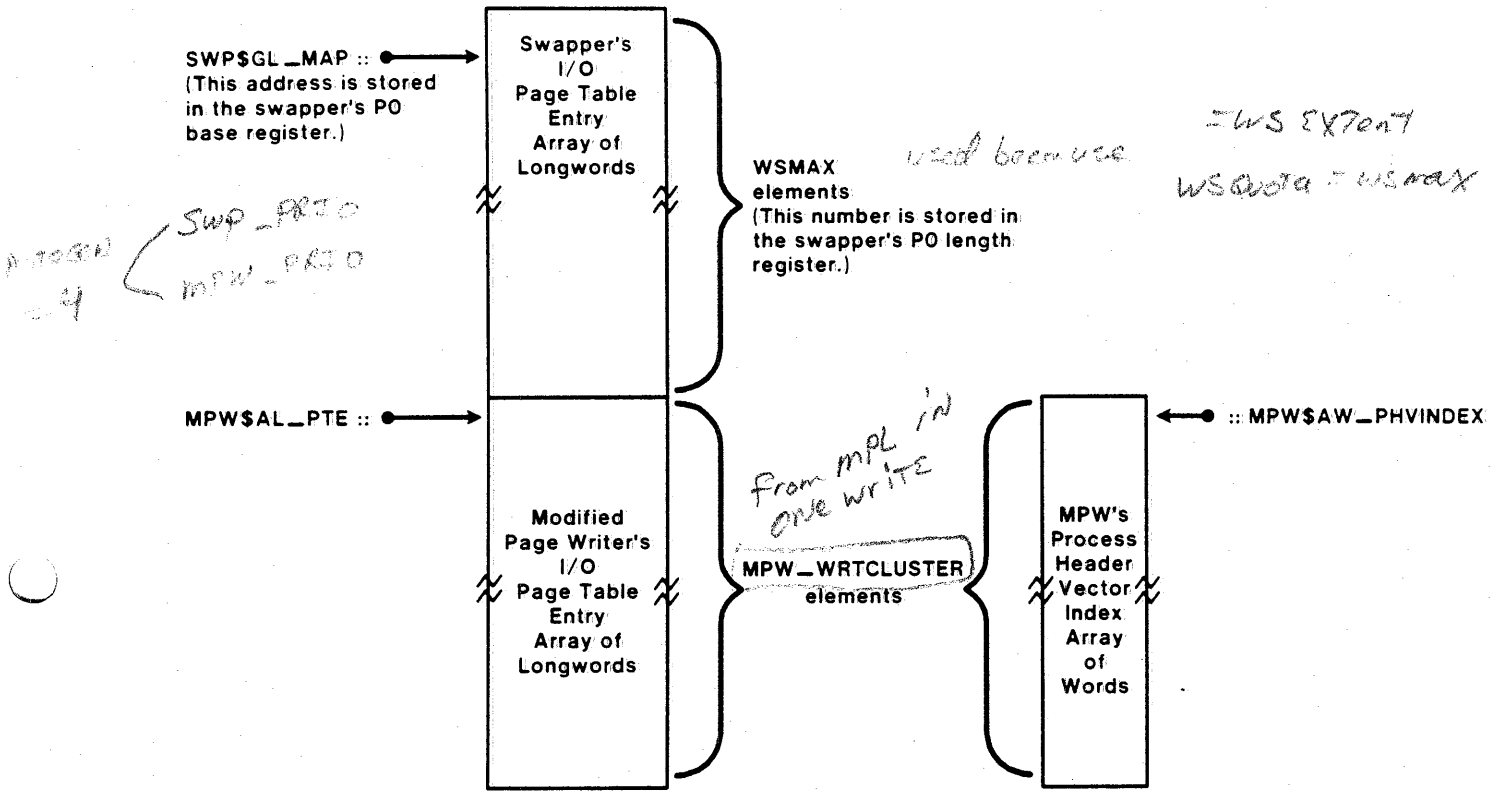
Number of Pages in Working Set



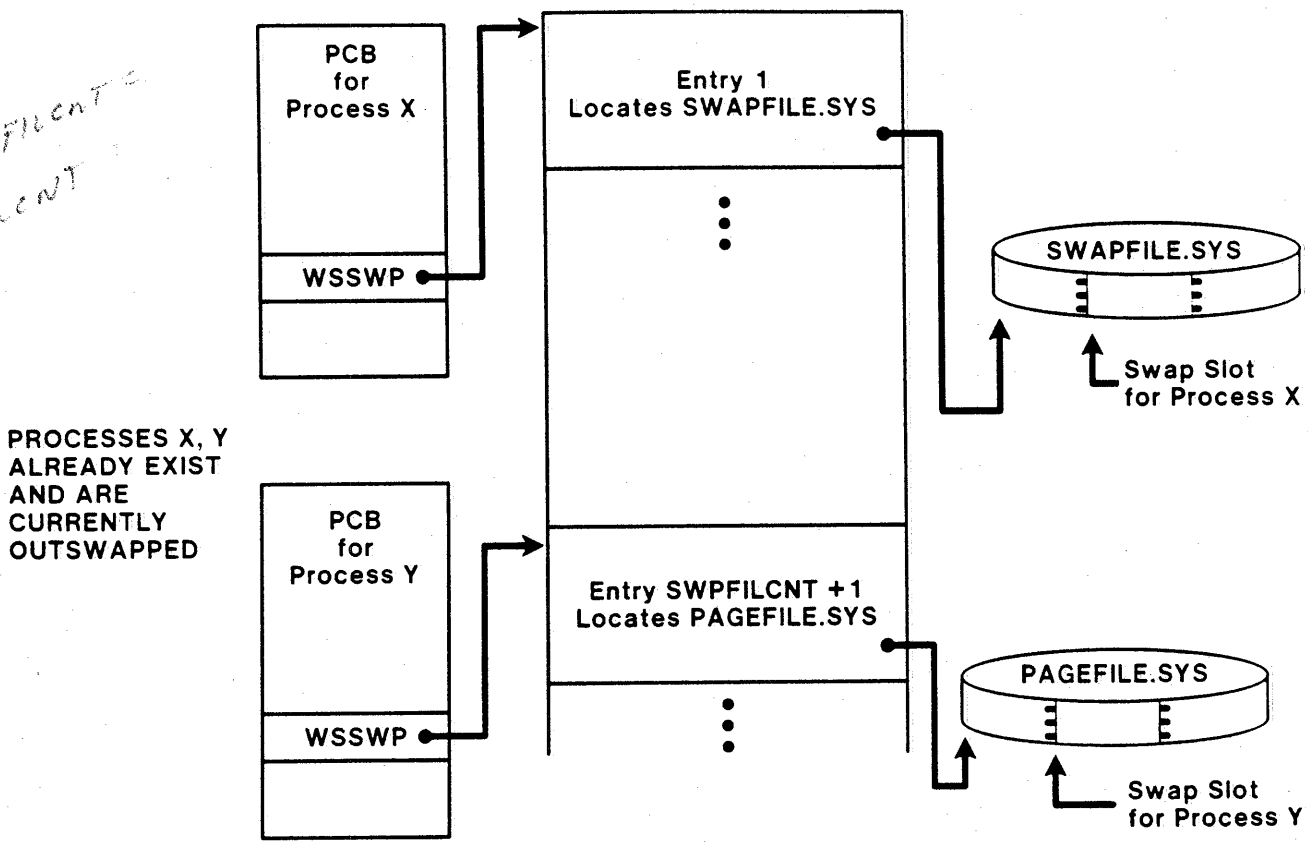
HOW SWAPPER'S P0 PAGE TABLE IS USED TO SPEED SWAP I/O



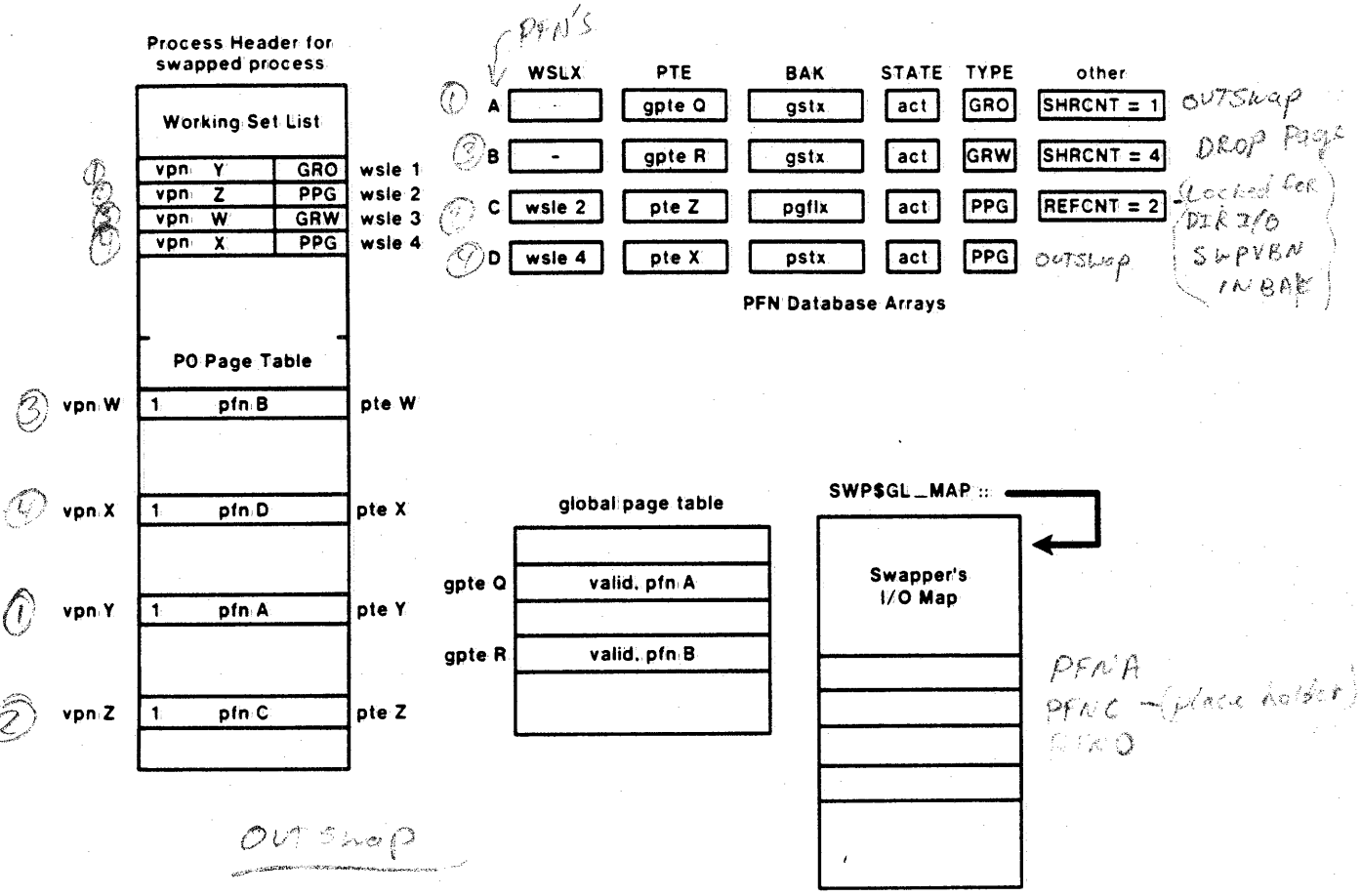
SWAPPER'S PSEUDO PAGE TABLES



LOCATING DISK FILES FOR SWAPPING



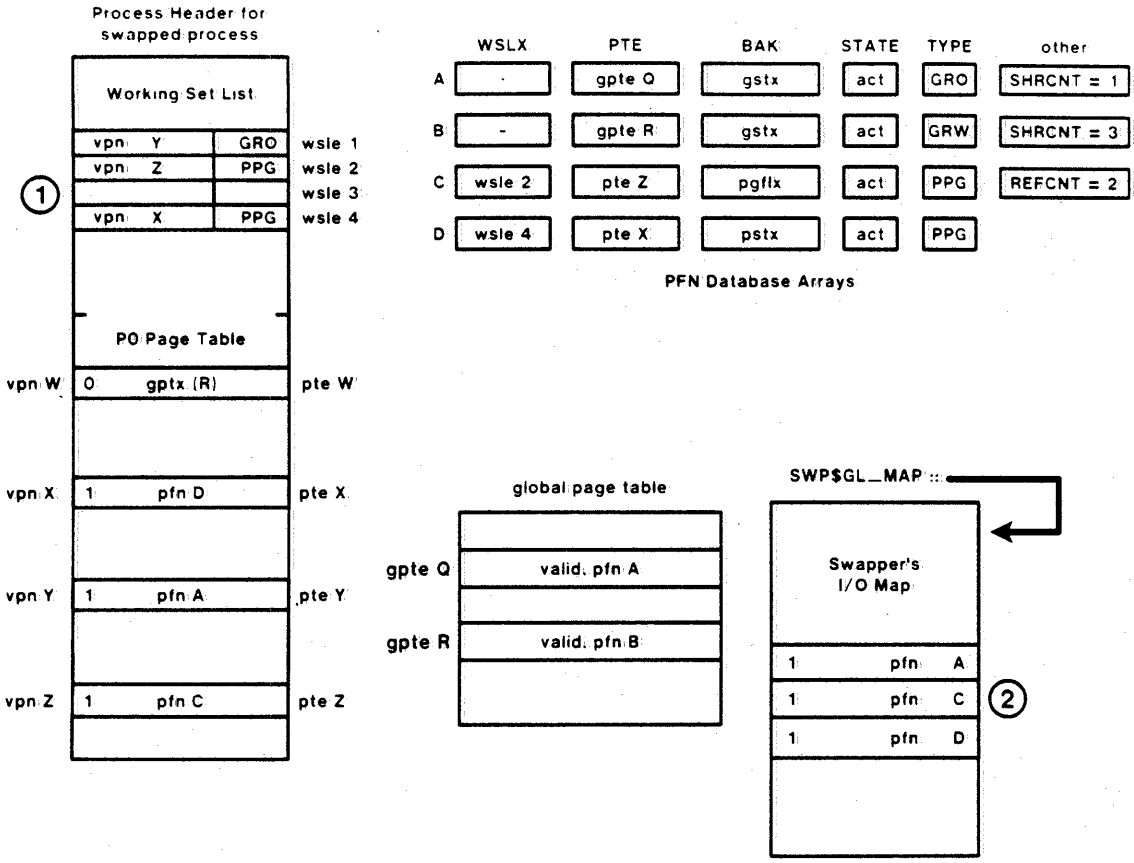
OUTSWAP WORKING SET LIST BEFORE OUTSWAP SCAN



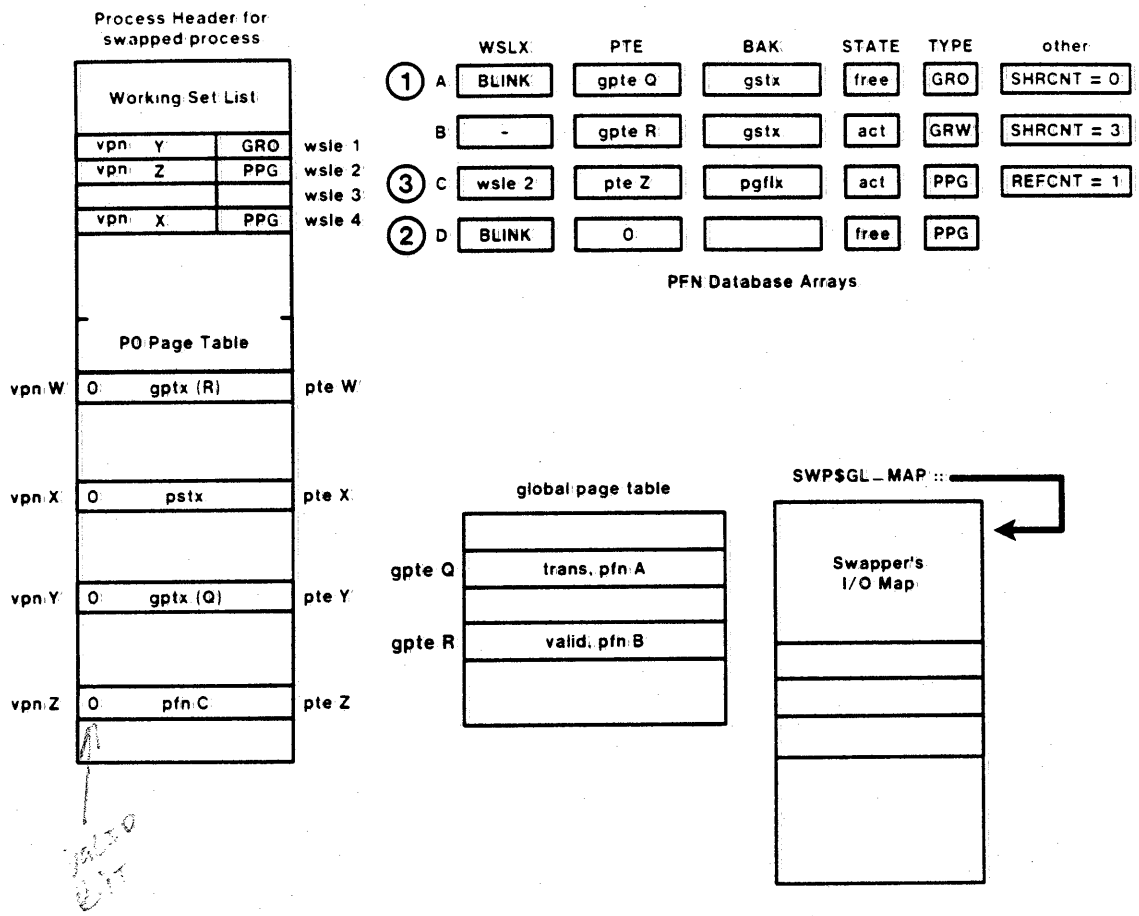
OUTSwap

PAGE	VALID	ACTION
PROCESS	Yes	OUTSwap
	Yes, DIR I/O	PUT SWPVBN IN PFN BAK array (DECR REFCNT)
GLOBAL (Read)	Yes	SHRCNT = 1 → OUTSwap (DECR SHRCNT)
(R/W)	Yes	SHRCNT > 1 → Drop page from WS
	NO	Drop page from WS (DECR SHRCNT)

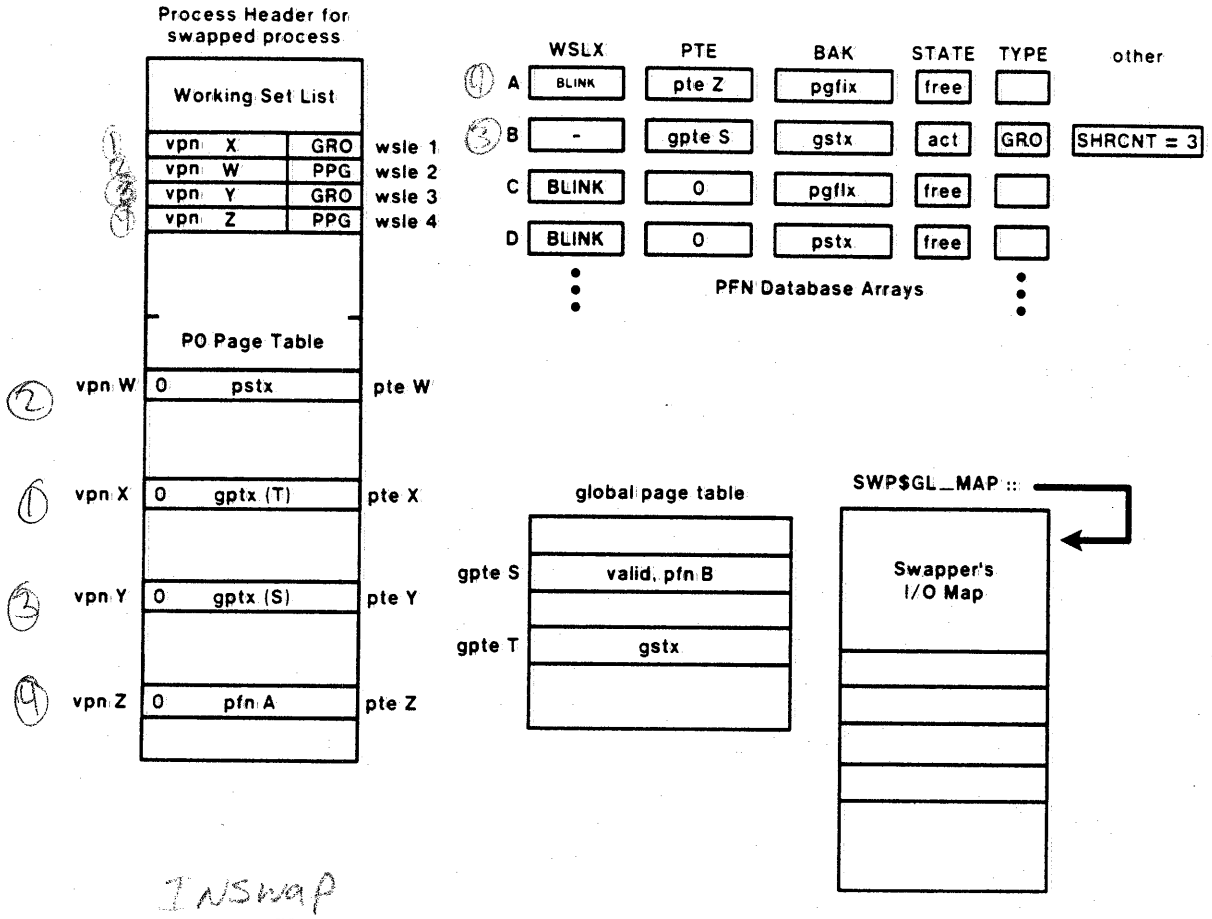
OUTSWAP WORKING SET LIST AFTER OUTSWAP SCAN



OUTSWAP PROCESS PAGE TABLES AFTER SWAPPER WRITE



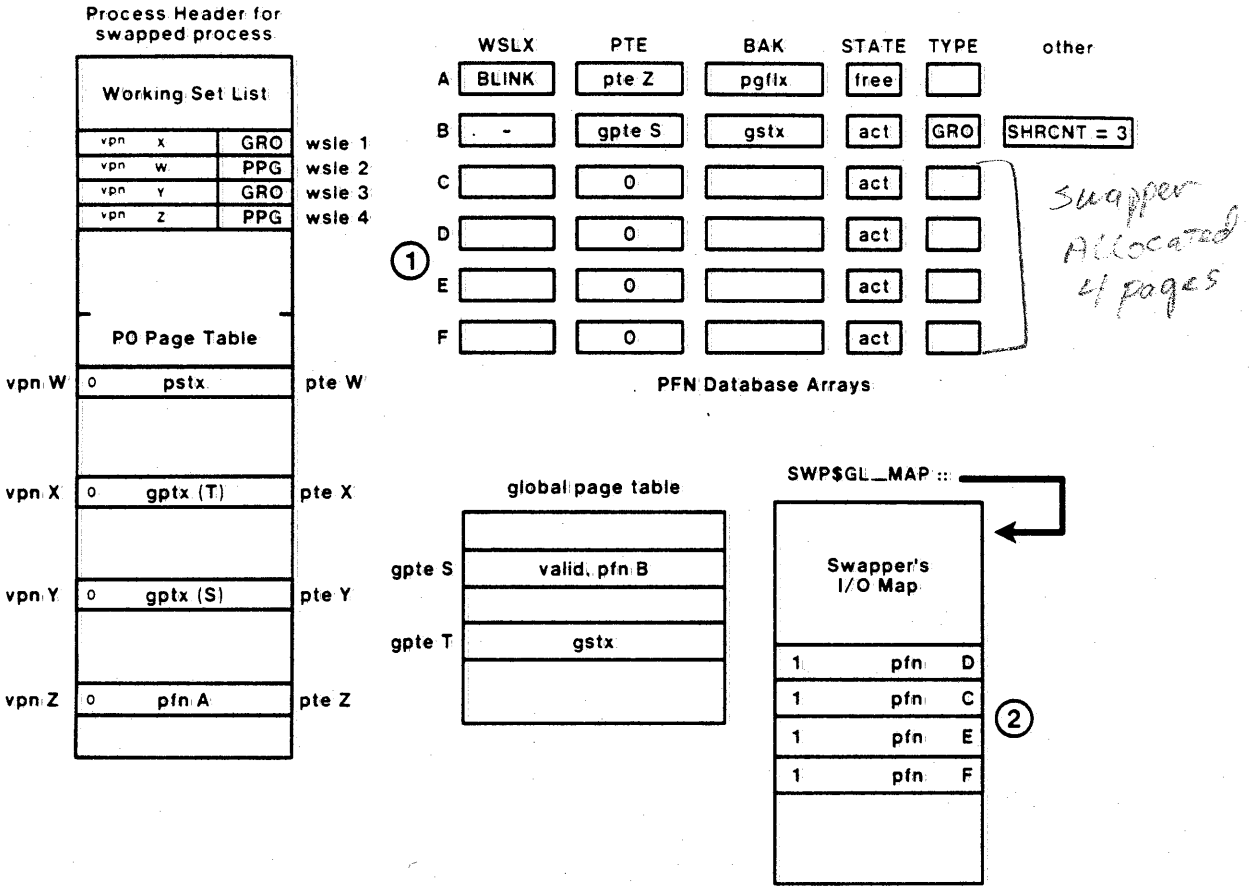
INSWAP WORKING SET LIST AND SWAPPER MAP BEFORE PHYSICAL PAGE ALLOCATION



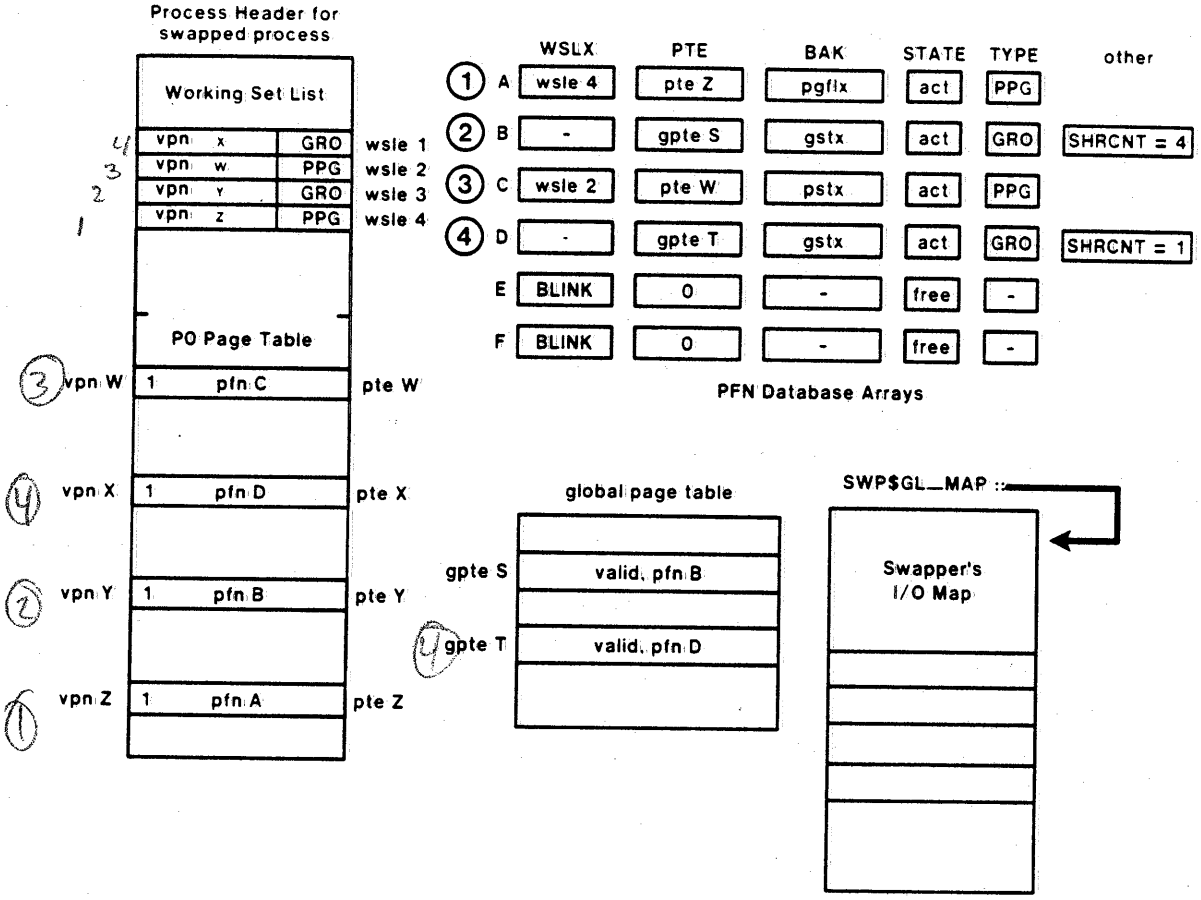
INSwap

PTE	Action
VALID	Never out swapped
Transition PFN (RFL, FPL)	• Fault on Transition Page • Drop Duplicate
GPTX (Contents of GPTE)	• VALID PFN – update PTE, WSLE, Drop Duplicate • Transition PFN – update PTE, WSLE, Drop Duplicate • GSTX – Fault in Page, keep Page – update GPTE + PTE

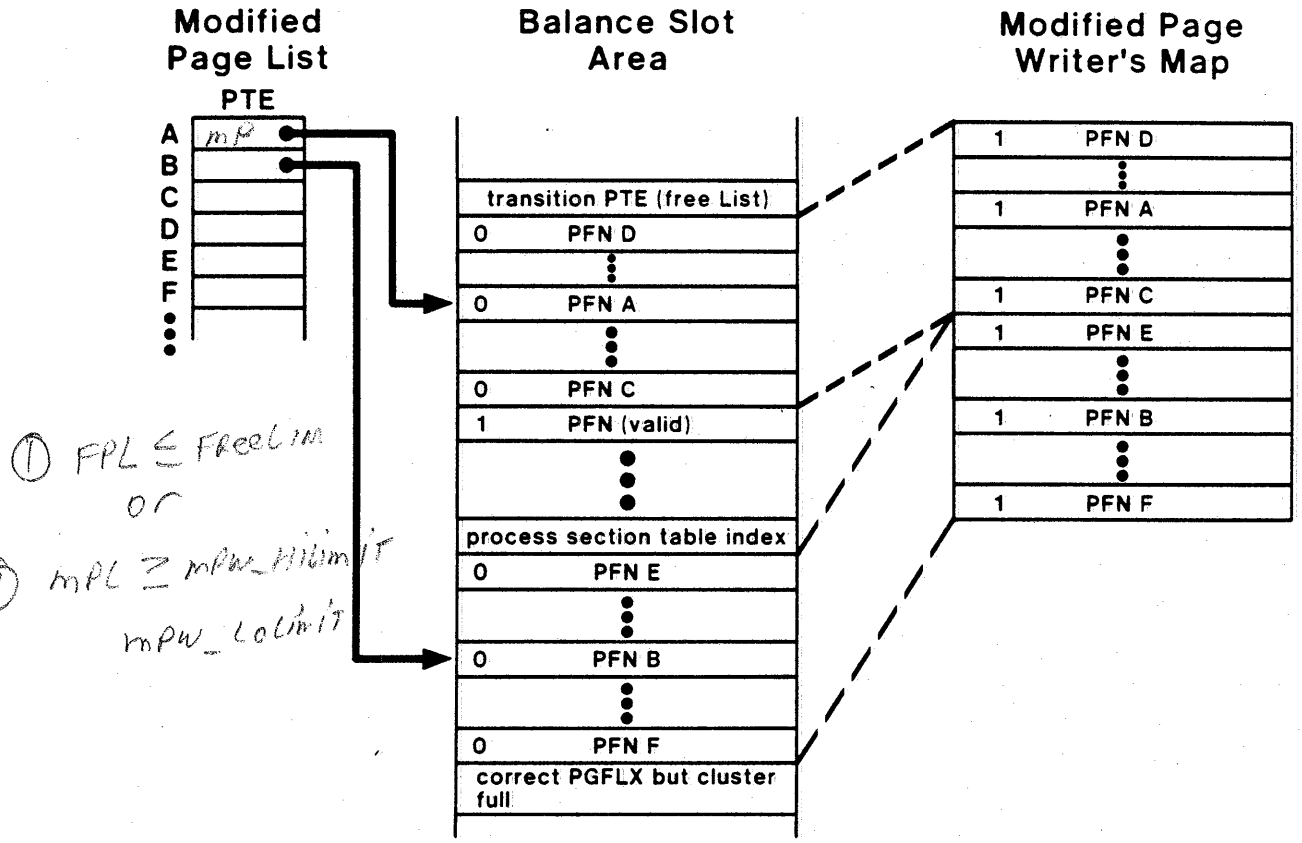
INSWAP WORKING SET LIST AND SWAPPER MAP AFTER ALLOCATION



INSWAP WORKING SET LIST AND REBUILT PAGE TABLES



HOW MODIFIED PAGE WRITER GATHERS PAGES



① FPL ≤ FreeLim
or
② mPL ≥ mPW_HiLimit
mPW_LoLimit

Build mini-clusters into
1 cluster that is ≤ mPW_WRTCLUSTR
Gather Pages!
until -
FPL PTE
VALID PFN
PSTX
mPW_WRTCluster
Pages is reached

IF mPL = mPW_WaitLimit
Current process is put in wait
Set to ≥ mpwLimit
IF < Then Trouble

PROCESS CREATION/DELETION

OBJECTIVES

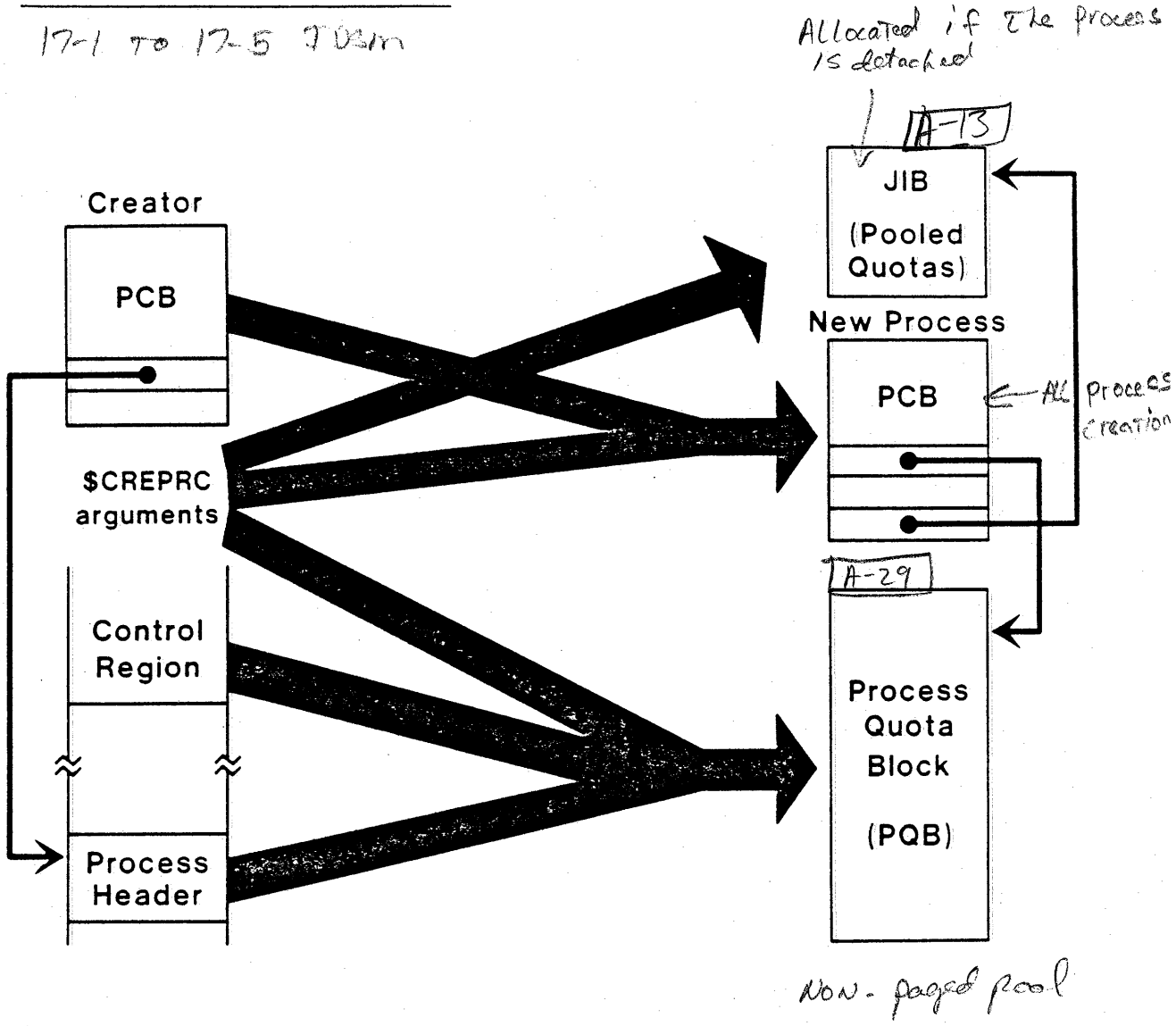
* list and explain several differences between simple processes, interactive processes, and batch processes
* show steps in process creation/deletion
* show primary loop of command language interpreters

READINGS

* IDSM . Chapters 17,18,19,20

CREATION OF PCB, JIB, PQB

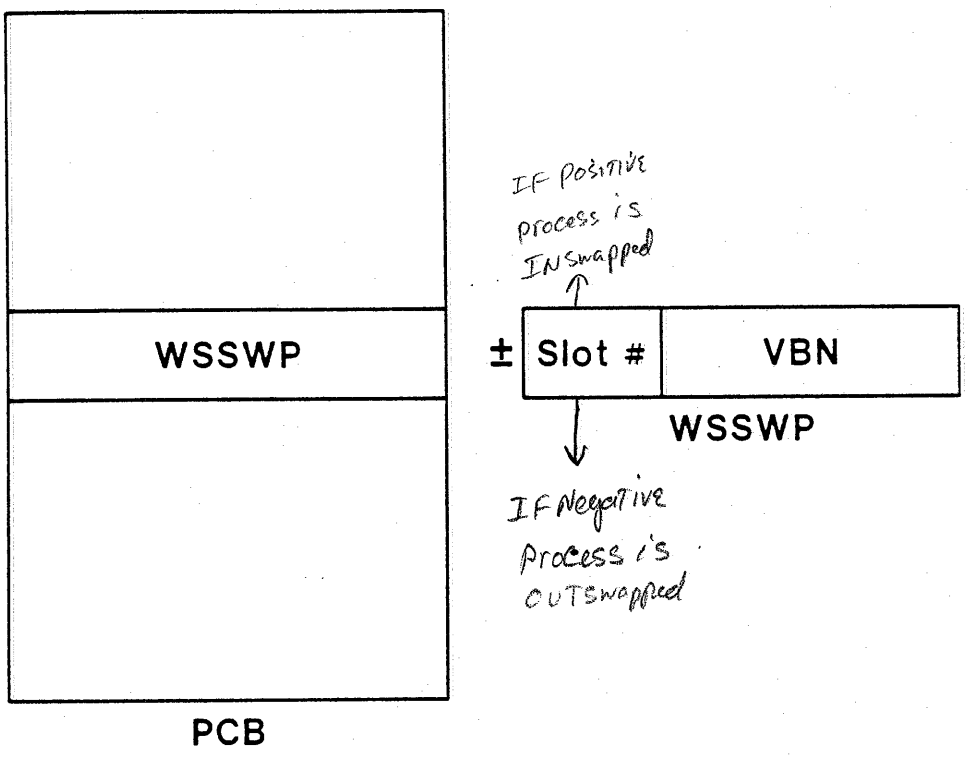
17-1 to 17-5 TDSM



SWAPPER'S ROLE IN PROCESS CREATION

17-14 to 17-15 IDSM

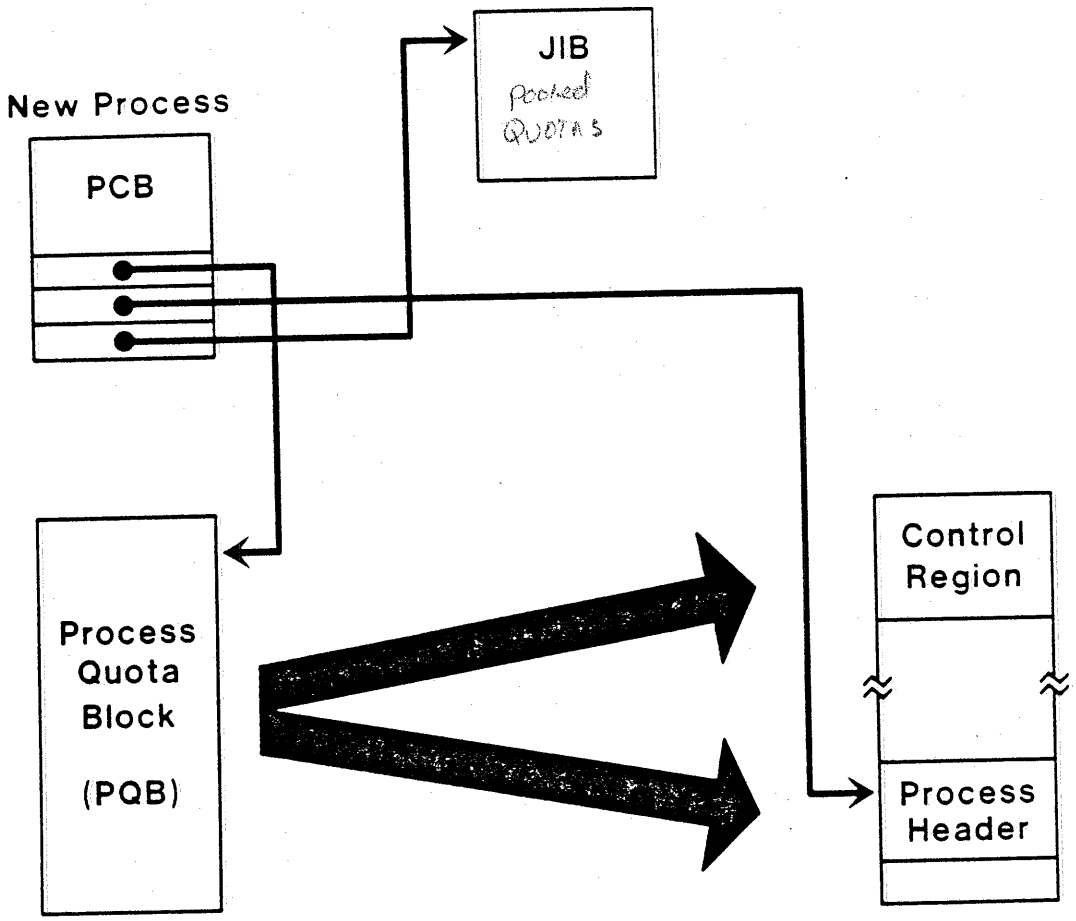
(1) configuration of PHD
- PT
- PST
- nSLEs



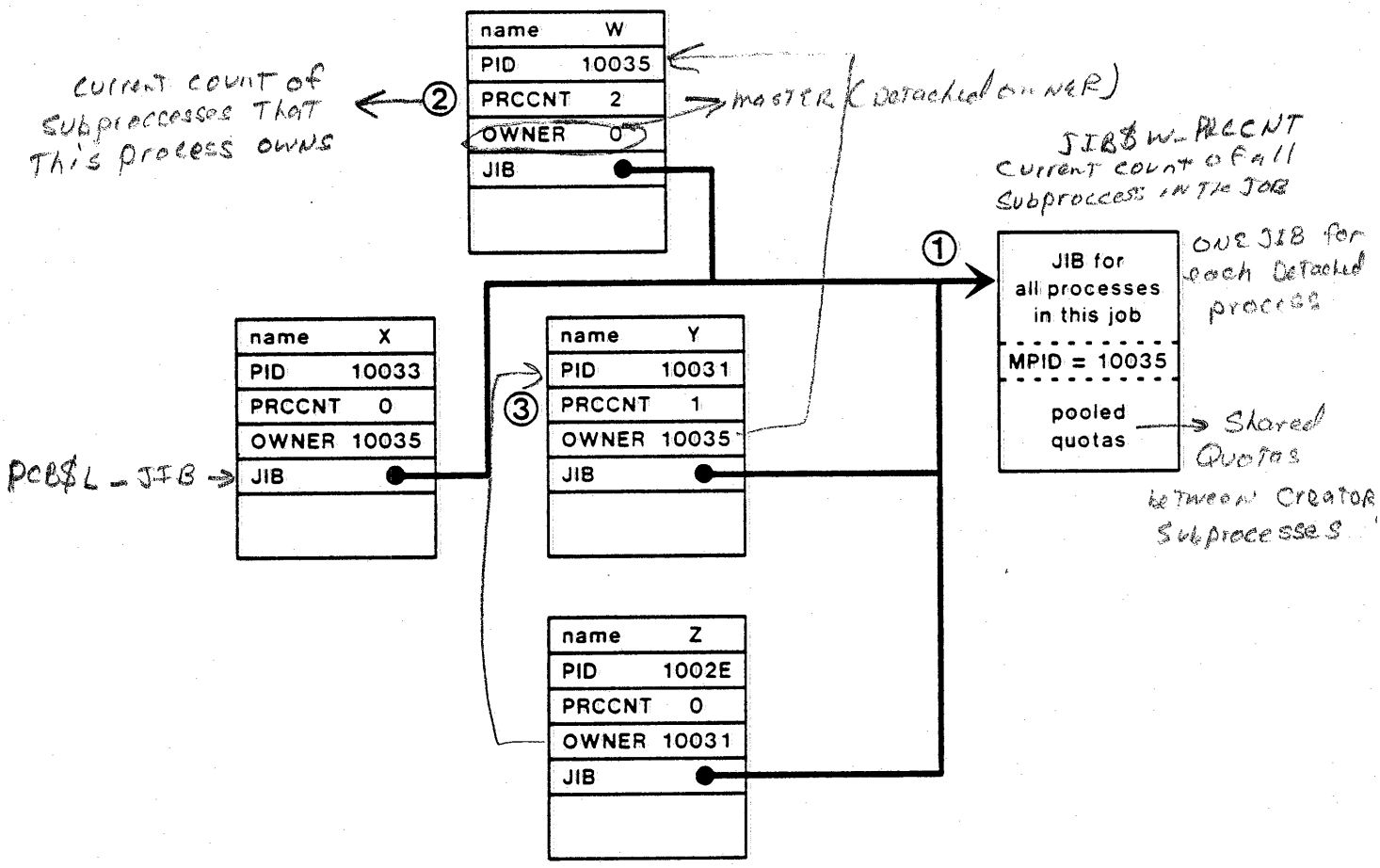
PROCSTRT'S ROLE IN PROCESS CREATION

DATA STRUCTURES USED AT PROCESS CREATION

17-16 TO 17-19 IDSm

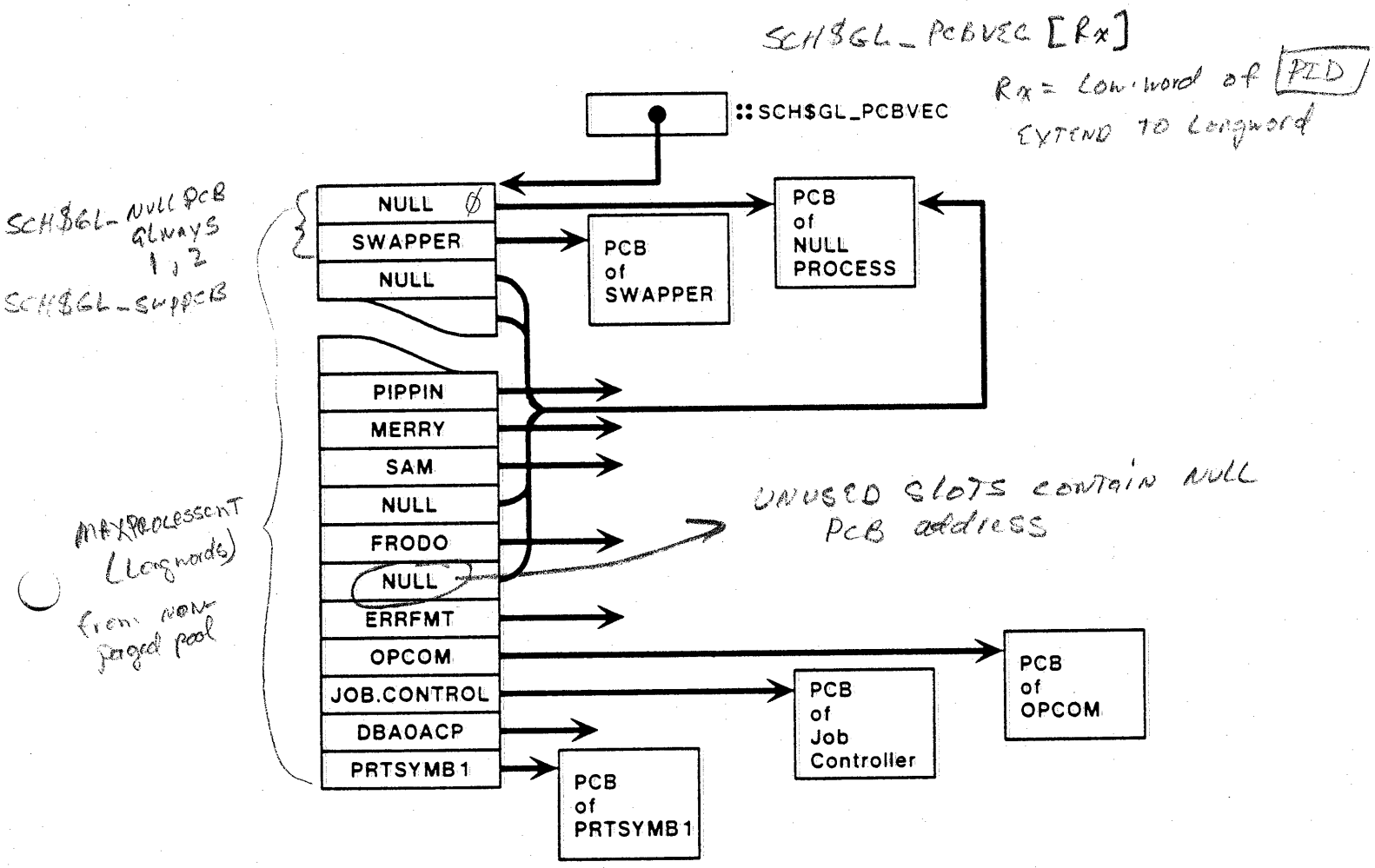


PCB AND JIB RELATIONSHIP

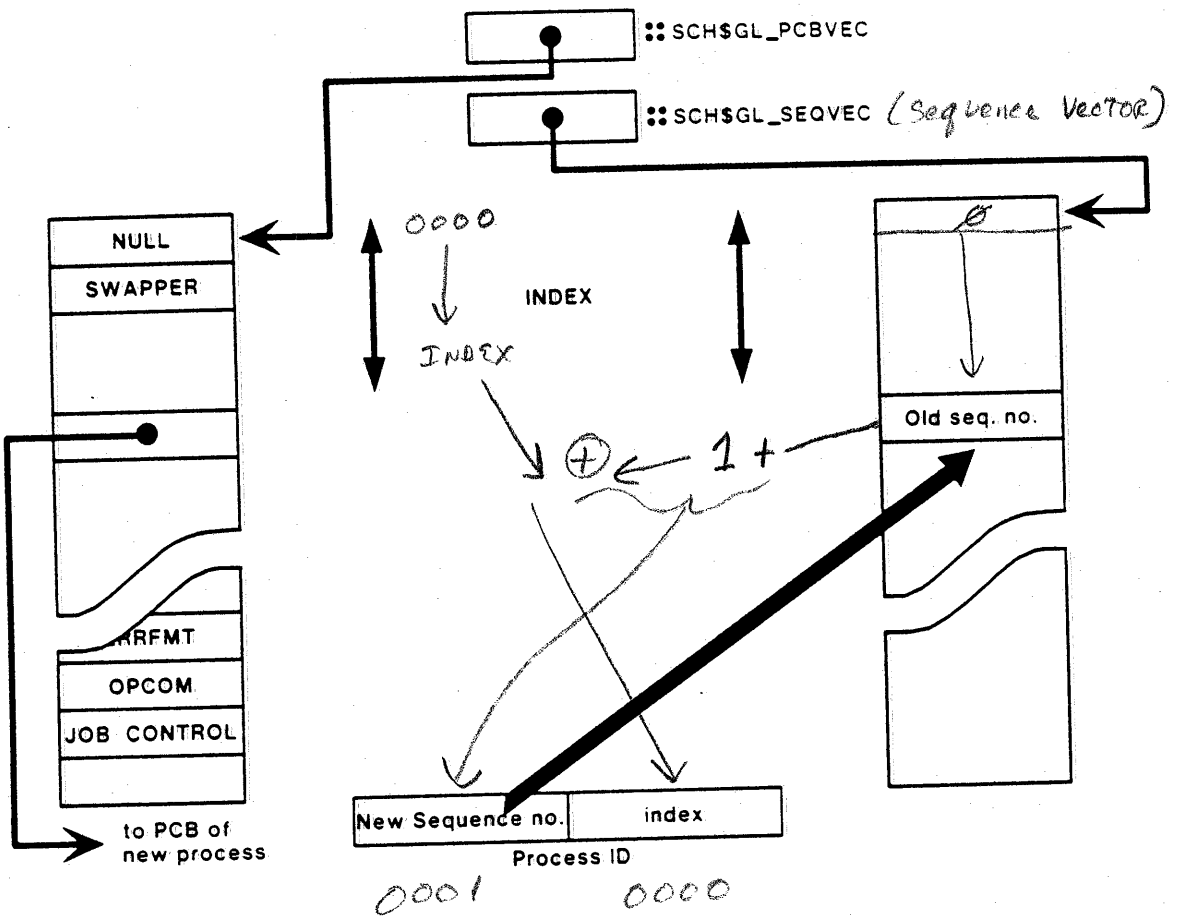


17-9 to 17-12 IDSM

PCB VECTOR



PID, PCB SEQUENCE VECTORS



SCH$GL_MAXPIX
Maximum PID INDEX

IS PID INDEX > MAXPIX

↑

0 - 32767 (POSITIVE)

PROCESS DELETION

$DELPRC

↓

EXE$NAM PID (get PCB)

↓

PCB$L_DELPEN in PCB$L_STS

IF THIS BIT IS SET CANNOT Delete process

Dismiss AST

name	OTG
PID	10035
PRCCNT	2 (MWAIT)
OWNER	0

name	BERT
PID	10033
PRCCNT	0
OWNER	10035

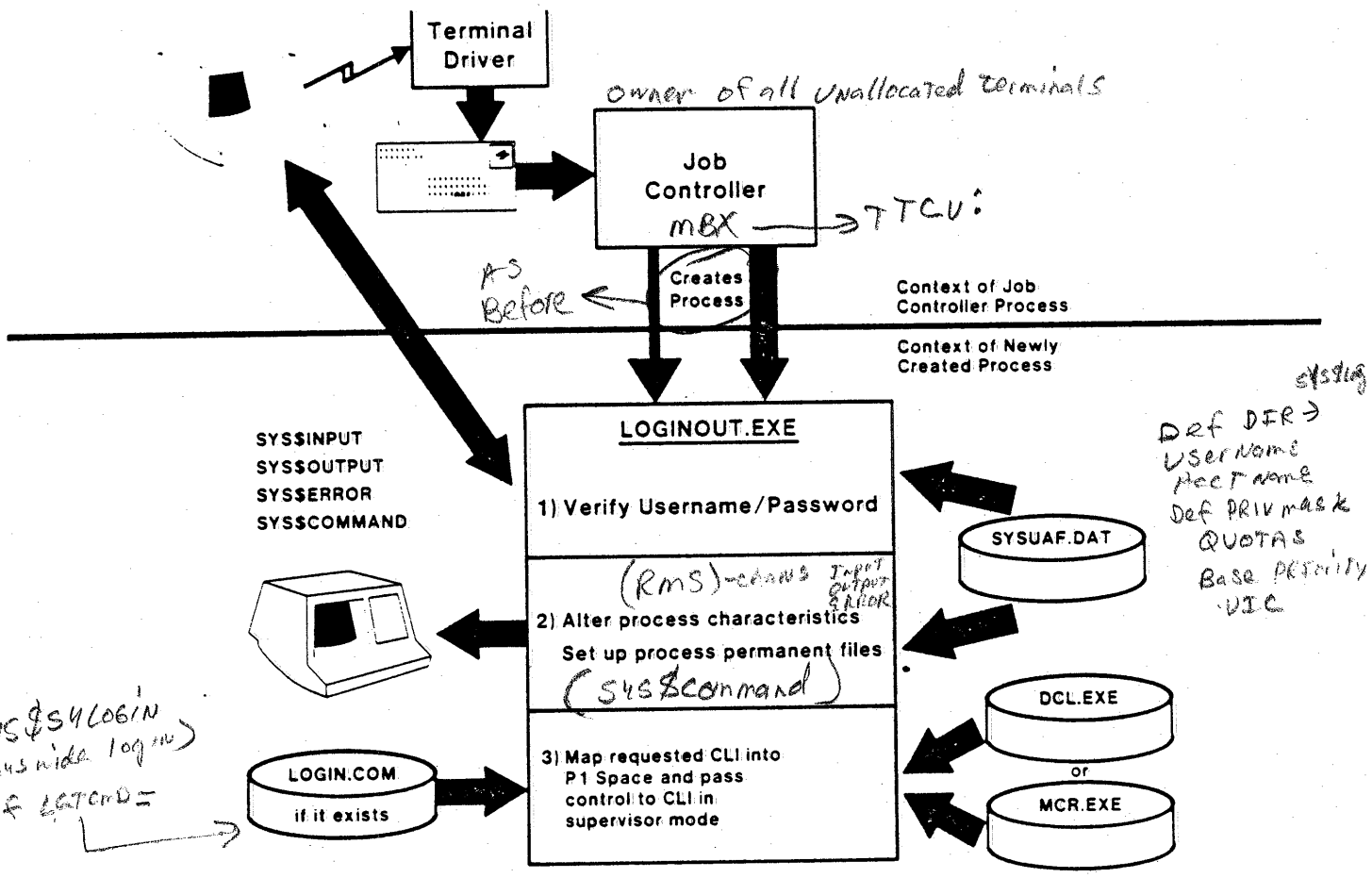
name	ERNIE
PID	10031
PRCCNT	0
OWNER	10035

RetQuota → check for subprocesses

- return quotas/resources Allocation
- Termination message to MBX
- put ACC INFO on STACK for IOCTL
- delete VIA space
- put NULL PCB in slot
- put NULL PCB in SCH$GL_CURPCB
- wake swapper to remove PID
- deallocate PCB/JIB to pool
- Jmp TO SCH$SCHED

INITIATING INTERACTIVE JOB (PROCESS)

20-5 to 20-7 IDSM



ProcessName TTCU:

UIC [10,40]

Image SYS$SYSTEM:LOGINOUT.EXE

SYS$INPUT
SYS$OUTPUT
SYS$ERROR
] TTCU:

Base PRIority SYSGEN DEFPRI (4) (reflects BATCH too!)

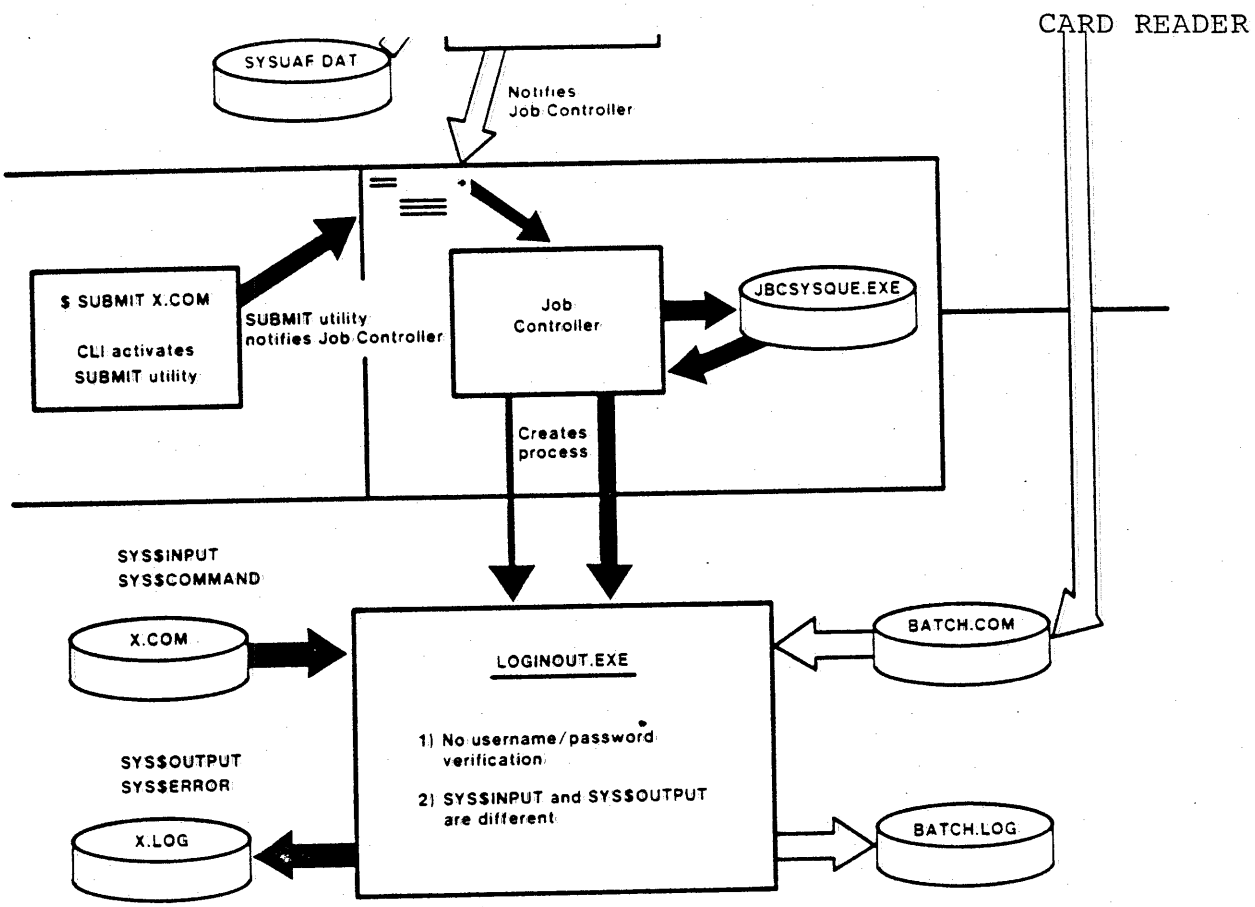
PRIVS ALL

ATTempt USERNAME → ProcessName (UNIQUE Process name IN SAME UIC Group)
(Look in RUBS)

INITIATING BATCH PROCESS USING $SUBMIT

20-3 IDSM

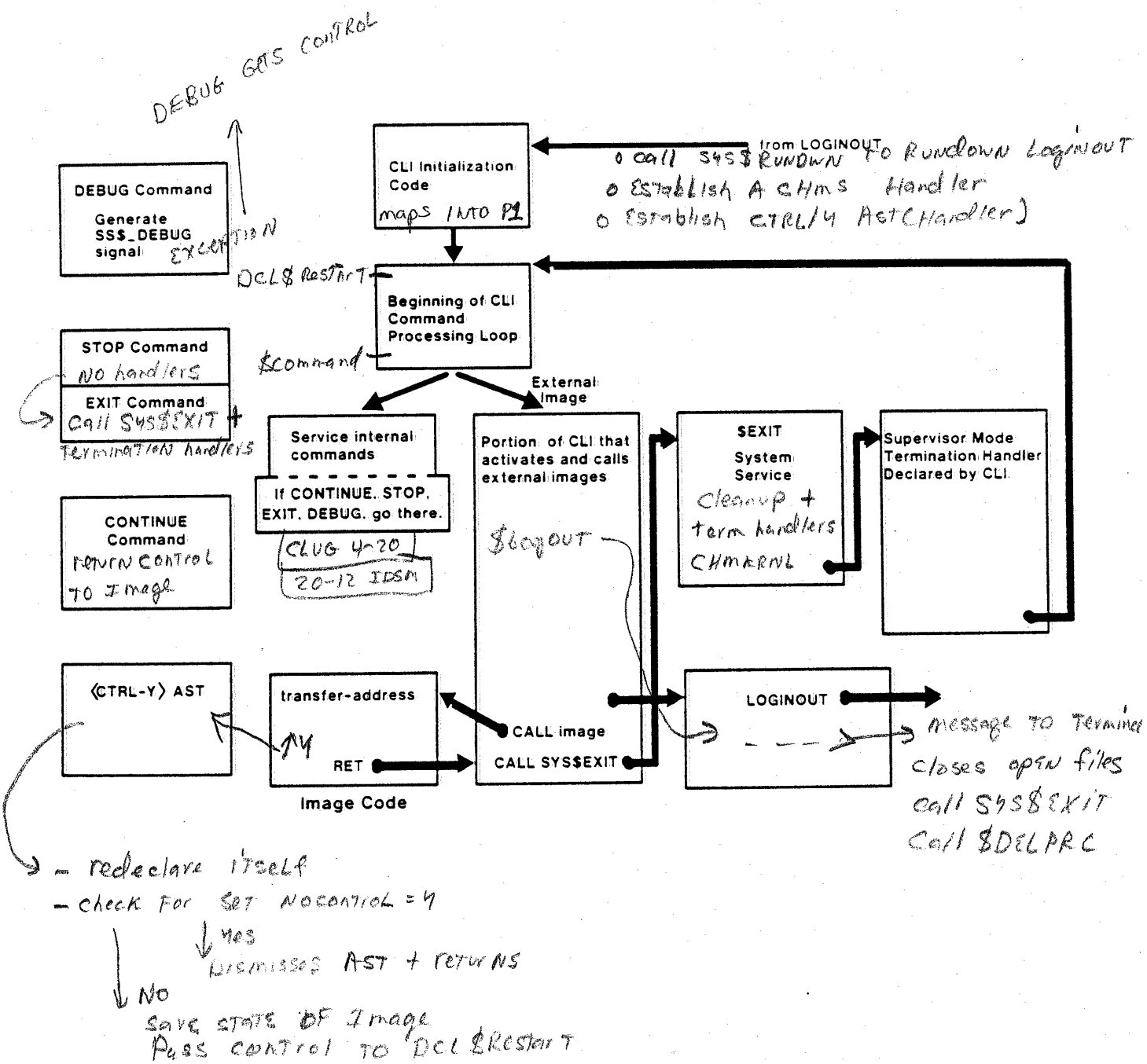
$SUBMIT command-proc



Process Name — JOB nnn

20-9 to 20-16 IDSM

COMMAND LANGUAGE INTERPRETER OPERATION (DCL)



SYSTEM INITIALIZATION

OBJECTIVES

* describe in general terms the sequence of operations in:
  - o initial bootstrap
  - o powerfail recovery
  - o startup for different VAX processors
* features like stand-alone backup and diagnostic boot differences from normal bootstraps

READINGS

* IDSM Chapters 7,21,22,23

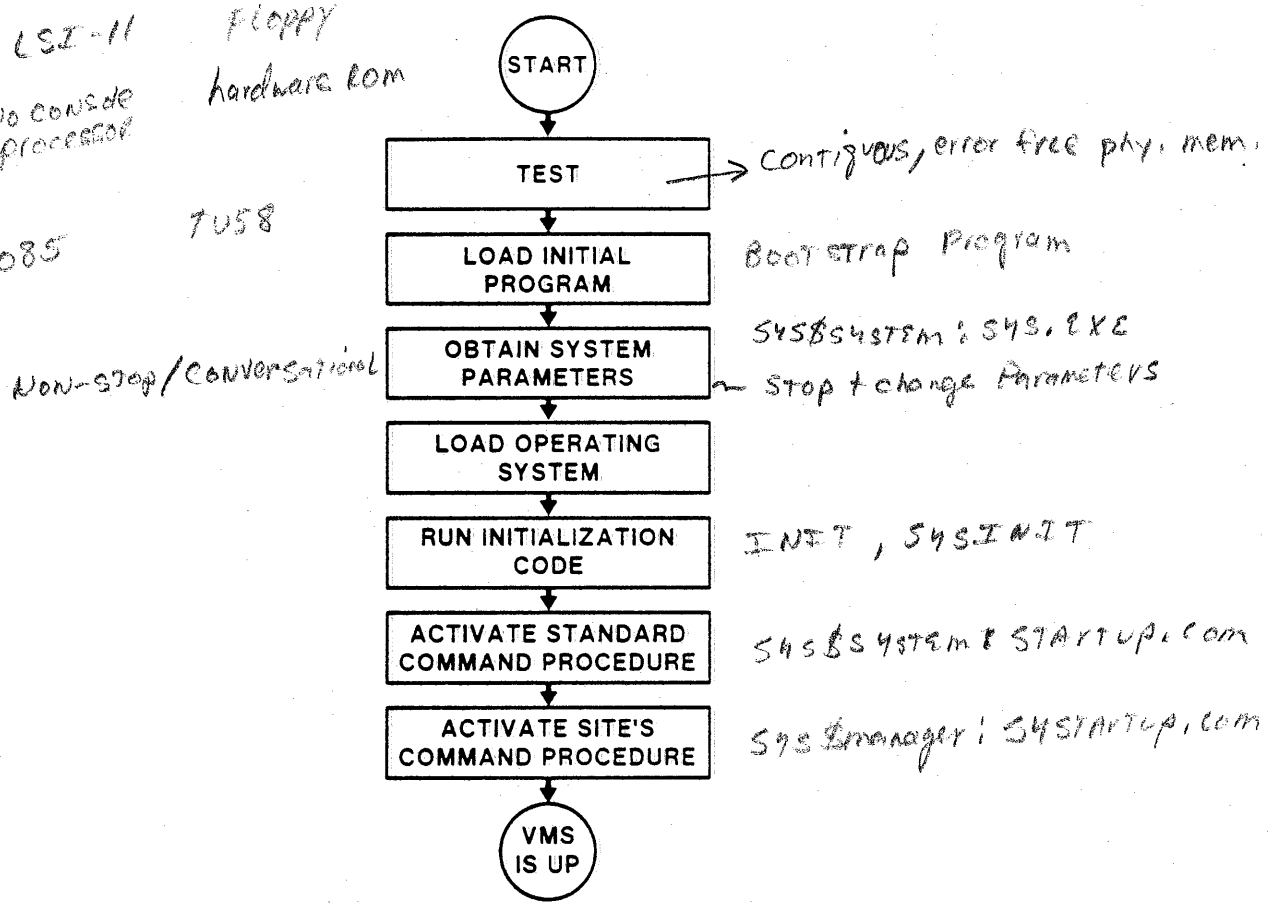
SYSTEM INITIALIZATION SEQUENCE

CONSOLE

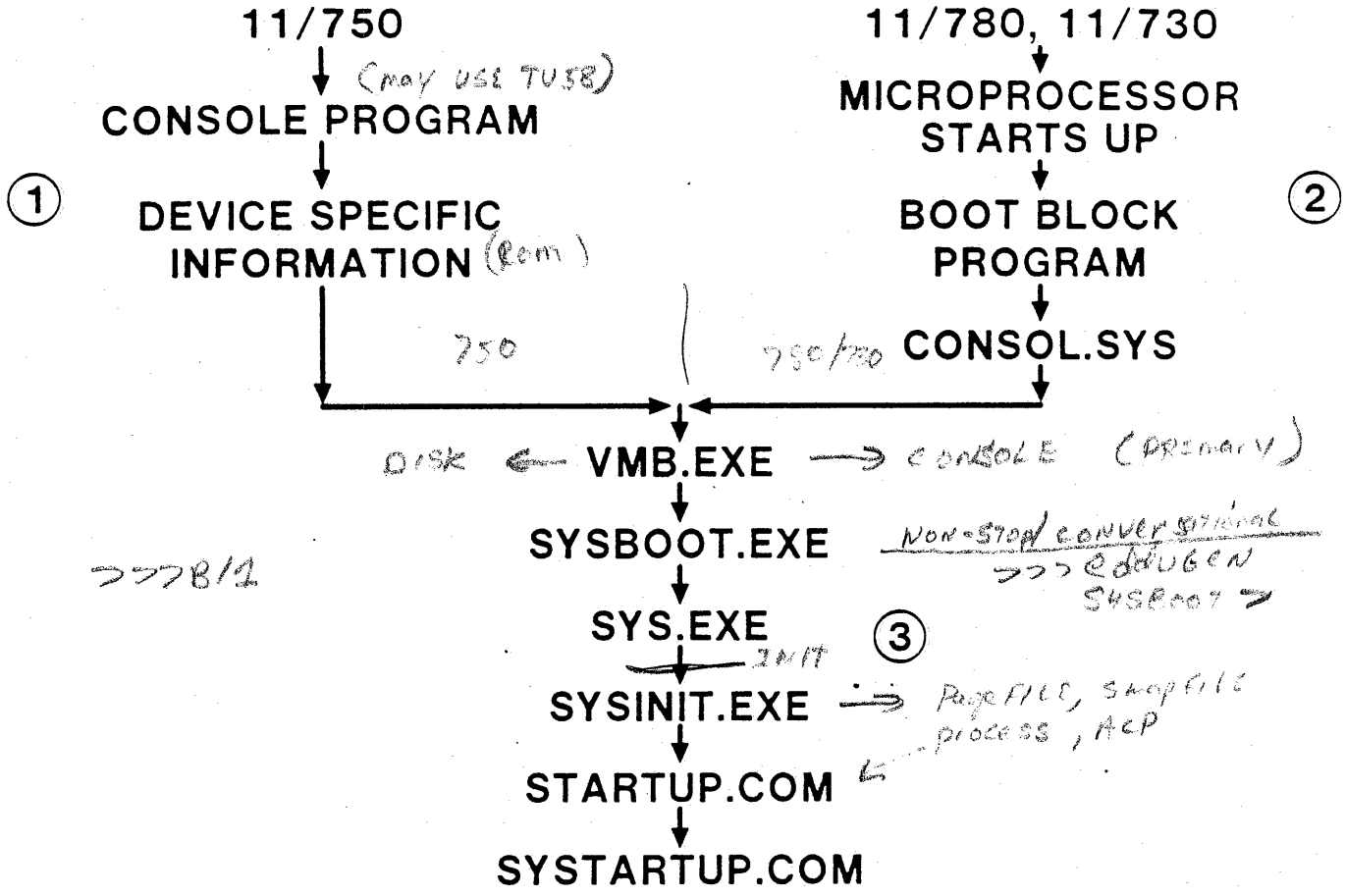
780 LSI-11 Floppy

250 No console processor hardware rom

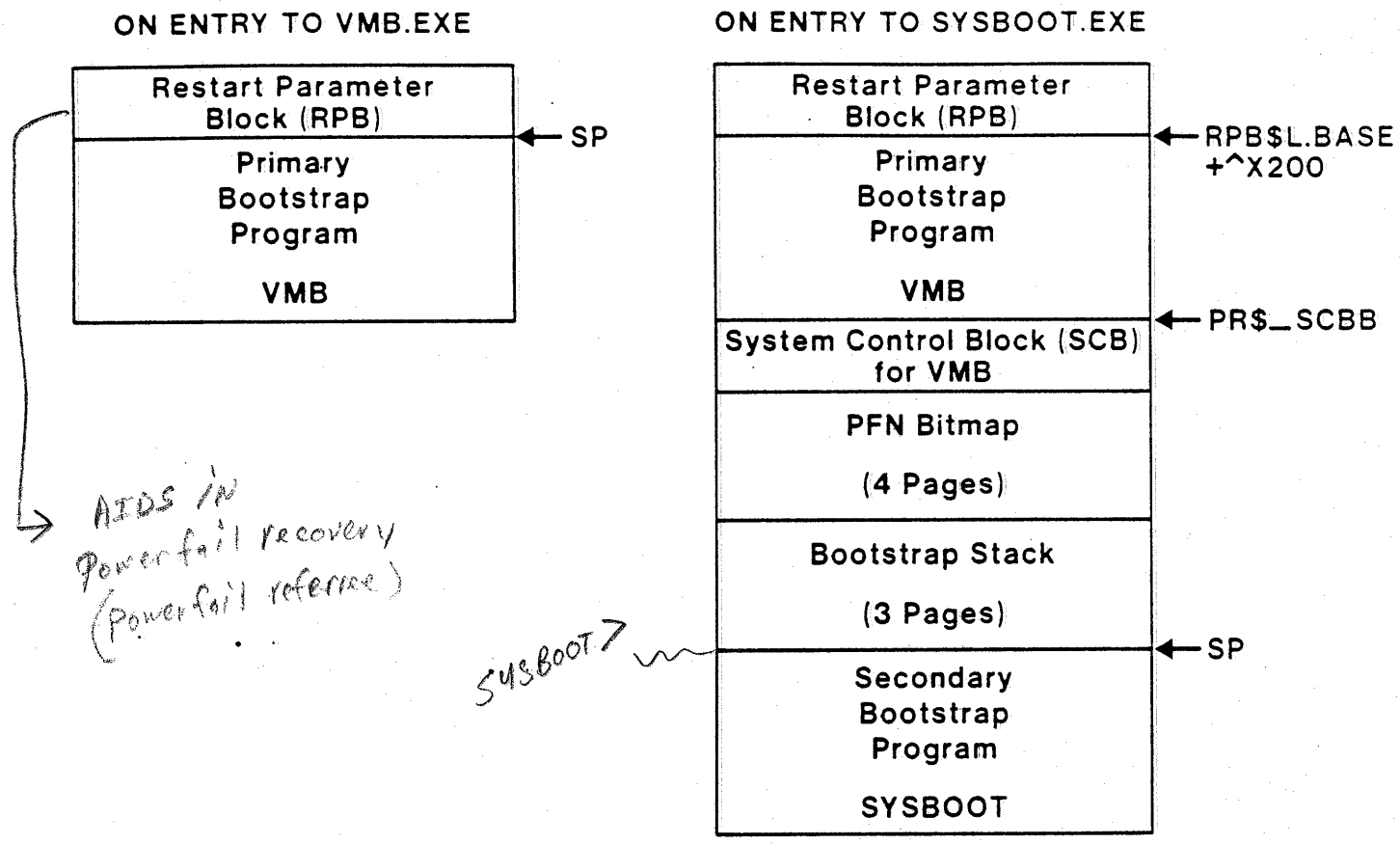
730 8085 7058



VAX PROCESSOR DIFFERENCES IN BOOTSTRAP

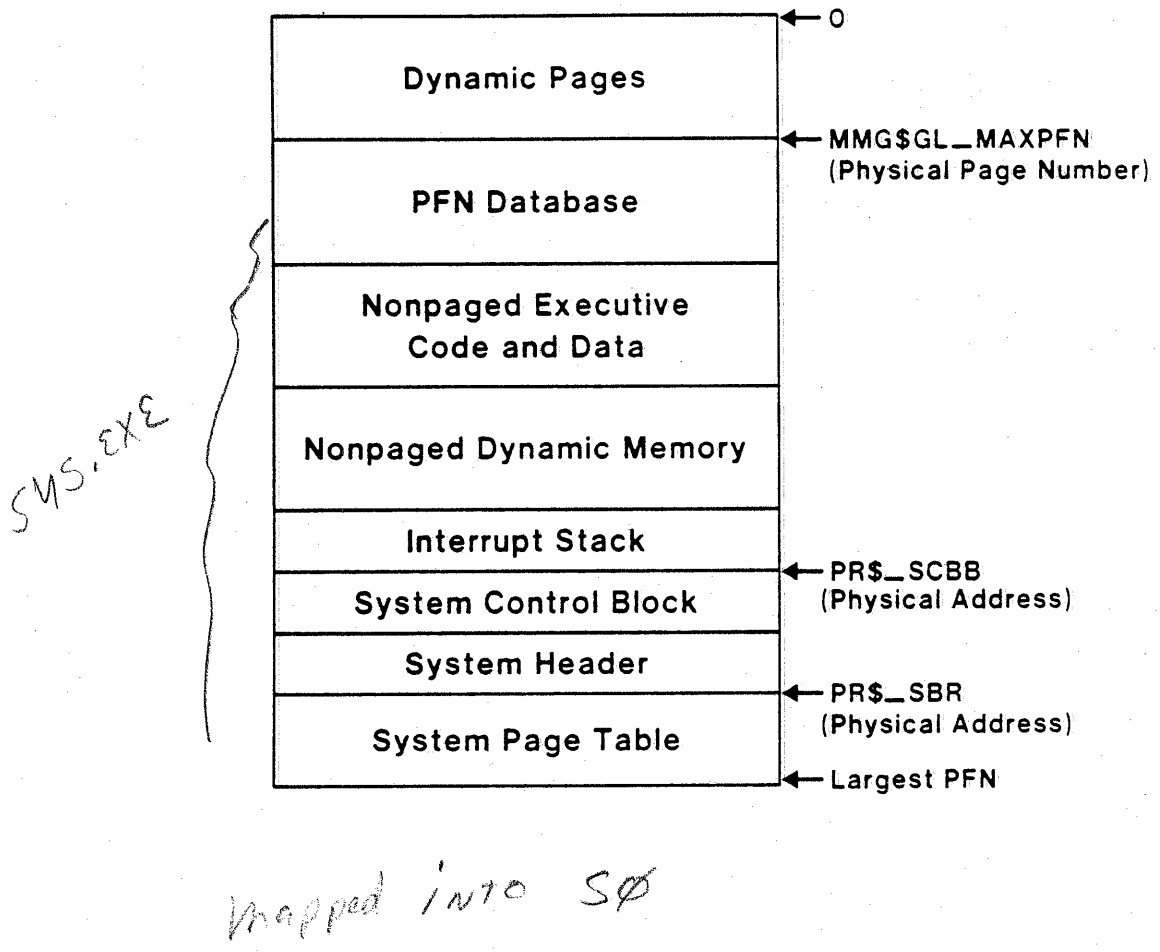


PHYSICAL MEMORY DURING INITIALIZATION

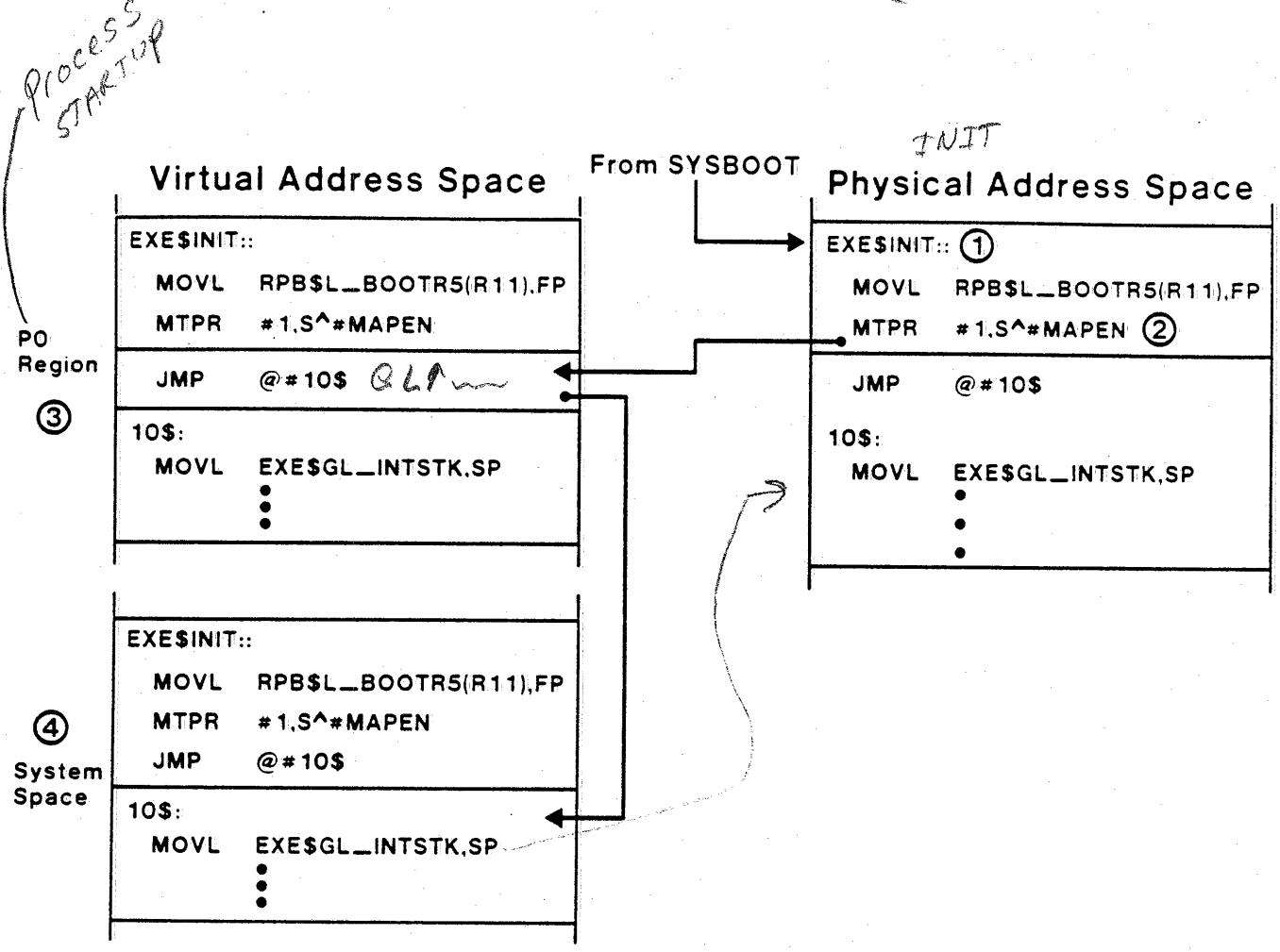


PHYSICAL MEMORY LAYOUT

AFTER SYSBOOT

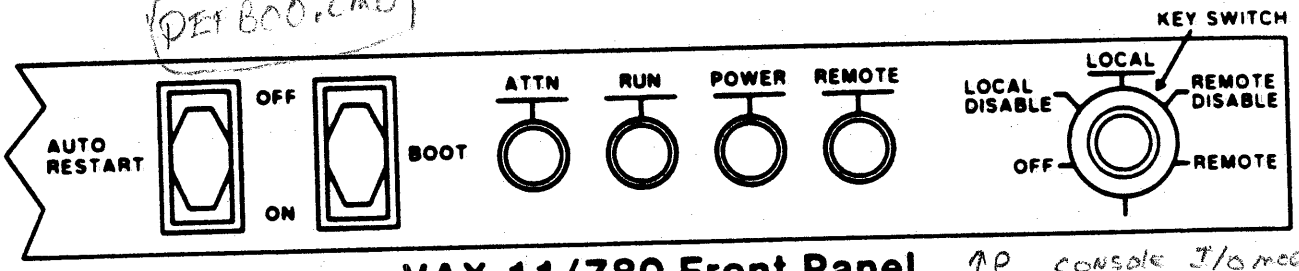


TURNING ON MEMORY MANAGEMENT (INIT)



VAX FRONT PANELS

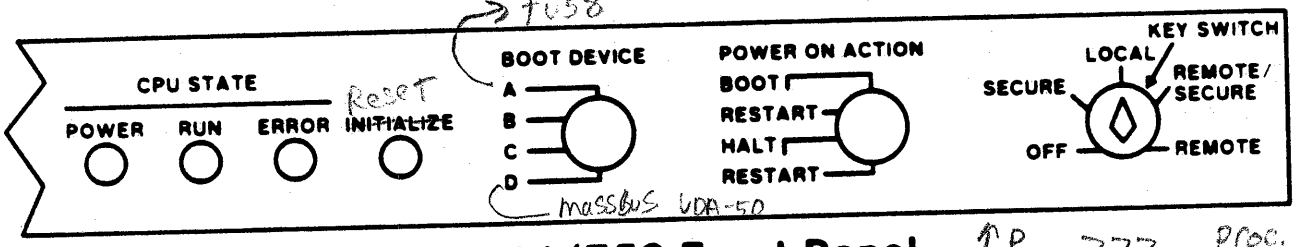
DET BOO.CMD



VAX 11/780 Front Panel

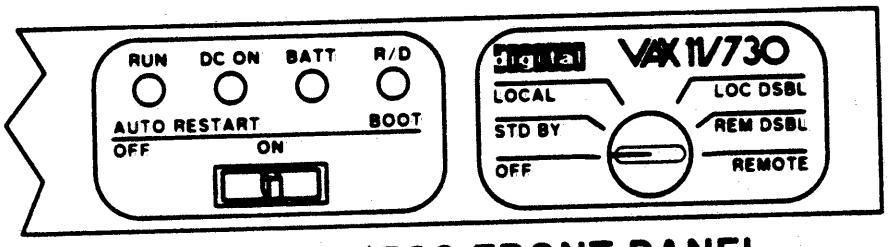
↑P console I/O mode
>>>C Can continue

TU58



VAX 11/750 Front Panel

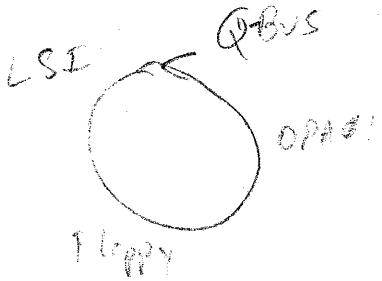
↑P >>> proc. stops
>>>C can continue
if BATT. Backup



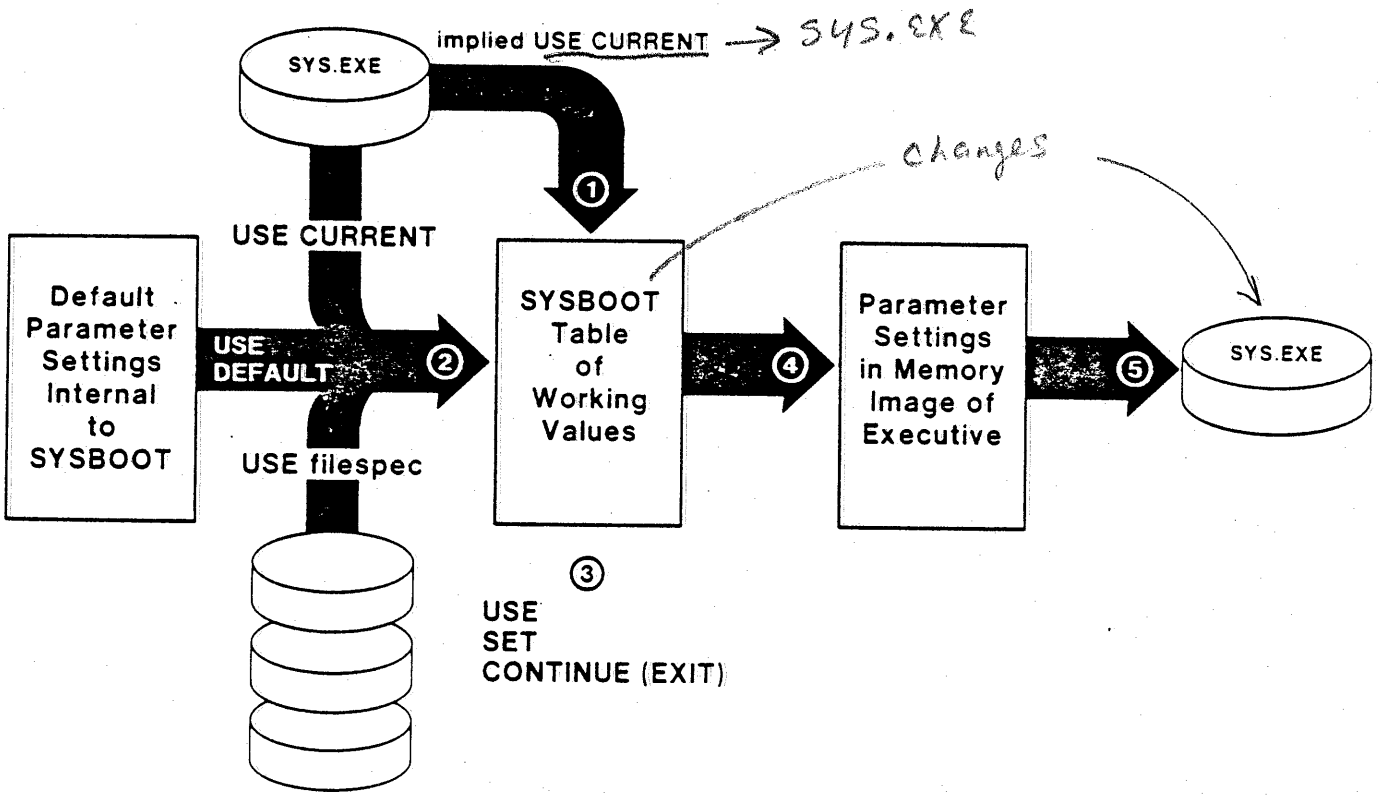
VAX 11/730 FRONT PANEL

↑P proc. stops

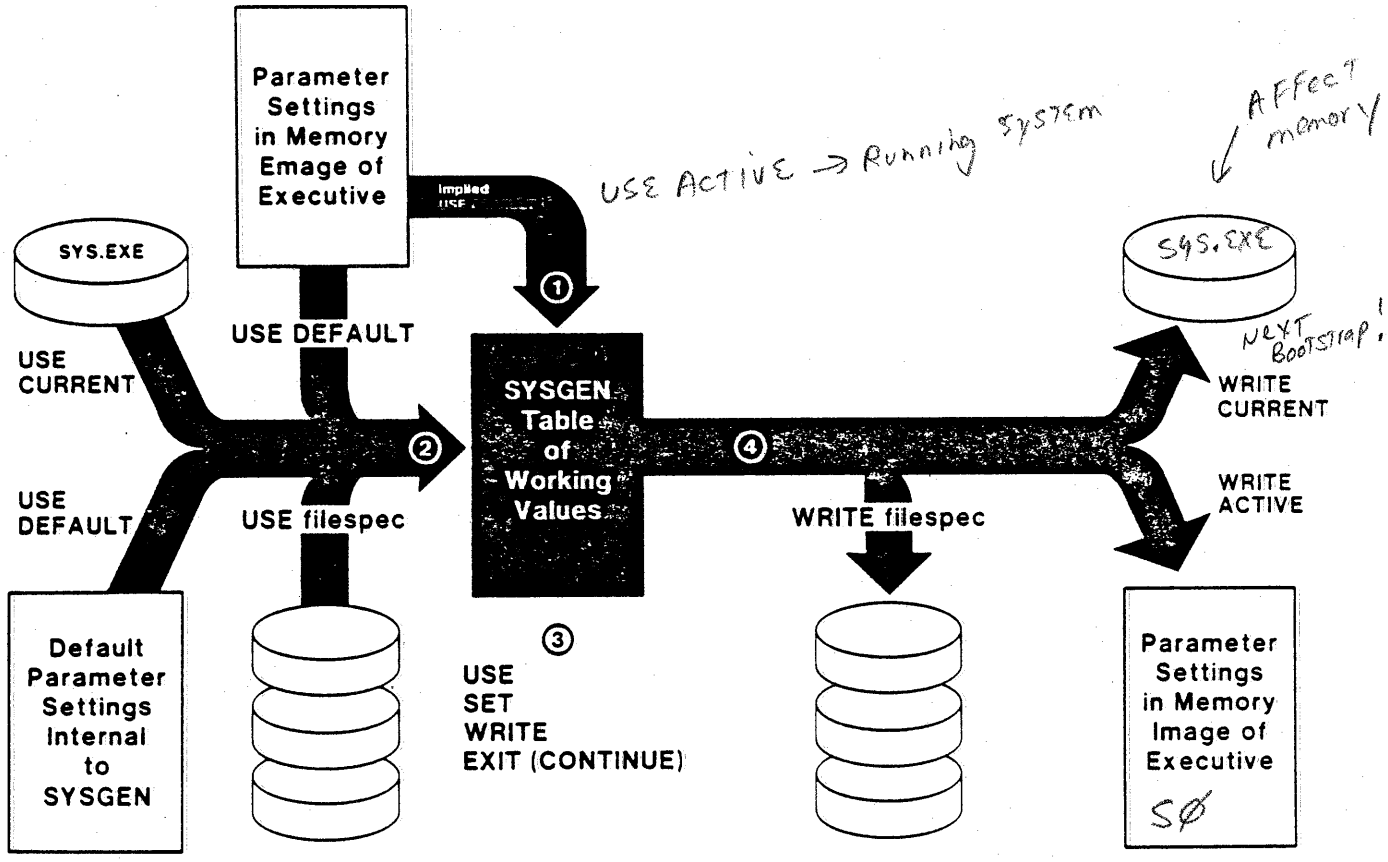
Figure 9-11 VAX Front Panels



CHANGING OF SYSTEM PARAMETERS WITH SYSBOOT



CHANGING OF SYSTEM PARAMETERS WITH SYSGEN



VAX/VMS INTERNALS AND DATA STRUCTURES

DATA STRUCTURES HANDOUT

APPENDIX A

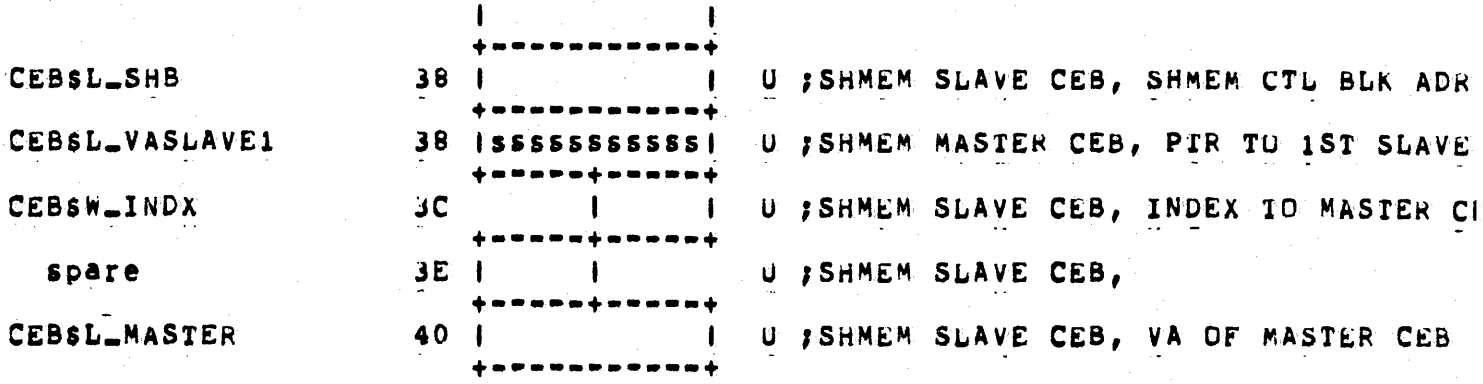
digital

```
;+
; AST CONTROL BLOCK DEFINITIONS
;
; AST CONTROL BLOCKS EXIST AS SEPARATE STRUCTURES AND AS SUBSTRUCTURES
; WITHIN LARGER CONTROL BLOCKS SUCH AS I/O REQUEST PACKETS AND TIMER
; QUEUE ENTRIES.
;
;-

        $ACBDEF


                                +------------+
ACB$L_ASTQFL                 0  |            |   U ;AST QUEUE FORWARD LINK
                                +------------+
ACB$L_ASTQBL                 4  |            |   U ;AST QUEUE BACKWARD LINK
                                +------+-----+
ACB$W_SIZE                   8       |     |   U ;STRUCTURE SIZE IN BYTES
                                +---+--+-----+
ACB$B_TYPE                   A   |   |        U ;STRUCTURE TYPE CODE
                                +---+---+
ACB$B_RMOD                   B  |   |         U ;REQUEST ACCESS MODE
                                +---+---------+
ACB$L_PID                    C  |            |   U ;PROCESS ID OF REQUEST
                                +------------+
ACB$L_AST                   10  |            |   U ;AST ROUTINE ADDRESS
                                +------------+
ACB$L_ASTPRM                14  |            |   U ;AST PARAMETER
                                +------------+
ACB$L_KAST                  18  |            |   U ;INTERNAL KERNEL MODE XFER ADDRESS
                                +------------+


ACB$B_RMOD                                          ;REQUEST ACCESS MODE
     0  ACB$V_MODE          2  no mask   U ;MODE FOR FINAL DELIVERY
     2    spare              2  no mask   U ;SPARE
     4  ACB$V_PKAST          1       10   U ;PIGGY BACK SPECIAL KERNEL AST
     5  ACB$V_NODELETE       1       20   U ;DON'T DELETE ACB ON DELIVERY
     6  ACB$V_QUOTA          1       40   U ;ACCOUNT FOR QUOTA
     7  ACB$V_KAST           1       80   U ;SPECIAL KERNEL AST
```

```
;+
; BOO - Boot Control Block
;
; A boot control block is produced by SYSBOOT and placed in non-paged
; pool.  It is pointed to by the cell EXE$GL_BOOTCB and contains
; the mapping information for SYS.EXE, SYSDUMP.DMP, SYSPARAM portion
; of SYS.EXE, and non-resident BUGCHECK code.
;-
        $BOODEF
; from starting VBN to end of executable image
; from starting VBN to end of file
                          +-----------+
BOO$L_CHECKSUM         0  |           |  U ; Checksum
                          +-----------+
BOO$L_PARAM_MAP        4  |           |  U ; Address of map for SYSPARAM
                          +-----+-----+
BOO$W_SIZE             8        |     |  U ; Size of fixed portion of BOOTCB
                             +--+-----+
BOO$B_TYPE             A     |  |        U ; Type of control block
                          +--+--+
BOO$B_SUBTYP           B  |  |           U ; Sub-type
                          +--+---------+
BOO$L_SYS_VBN          C  |           |  U ; SYS.EXE starting VBN
                          +-----------+
BOO$L_SYS_SIZE        10  |           |  U ; SYS.EXE size in blocks
                          +-----------+
BOO$L_SYS_MAP         14  |           |  U ; Adr of map for SYS.EXE
                          +-----------+
BOO$L_DMP_VBN         18  |           |  U ; Starting VBN for dump file
                          +-----------+
BOO$L_DMP_SIZE        1C  |           |  U ; Size in blocks of dump file
                          +-----------+
BOO$L_DMP_MAP         20  |           |  U ; Adr of map for SYSDUMP.DMP
                          +-----------+
BOO$L_BUG_MAP         24  |           |  U ; Adr of map for non-resident BUGCHI
                          +-----------+
```

```
;+
; CCB - CHANNEL CONTROL BLOCK
;
; THERE IS ONE CHANNEL CONTROL BLOCK FOR EACH SOFTWARE CHANNEL THAT A
; PROCESS MAY INITIATE I/O REQUESTS ON. THE NUMBER OF SUCH I/O CHANNELS
; IS DETERMINED BY THE FIXED NUMBER ASSIGNED TO A PROCESS PLUS ANY
; ADDITIONAL CHANNELS REQUIRED BY THE IMAGE CURRENTLY BEING EXECUTED
; BY THE PROCESS.
;
; **** WARNING ****
;       THE CHANNEL CONTROL BLOCK IS ASSUMED TO BE FOUR LONG WORDS
; THROUGHOUT THE EXEC.  ITS SIZE MAY BE CHANGED BUT ONLY BY POWERS OF 2.
;-

        $CCBDEF

                                  +-------------+
CCB$L_UCB                      0  |             |   U ;ADDRESS OF ASSIGNED DEVICE UCB
                                  +-------------+
CCB$L_WIND                     4  |             |   U ;ADDRESS OF WINDOW BLOCK
                                  +---------+---+
CCB$B_STS                      8  |         |   |   U ;CHANNEL STATUS
                                  +---+---+---+
CCB$B_AMOD                     9  |   |   |       U ;ACCESS MODE THAT ASSIGNED CHANNEL
                                  +---+---+---+
CCB$W_IOC                      A  |       |       U ;NUMBER OF OUTSTANDING I/O REQUESTS
                                  +-------+-------+
CCB$L_DIRP                     C  |             |   U ;DEACCESS I/O REQUEST PACKET ADDRESS
                                  +-------------+

CCB$B_STS                                             ;CHANNEL STATUS
     0  CCB$V_AMB                 1          1   U ;   MAILBOX ASSOCIATED WITH CHANNEL
```

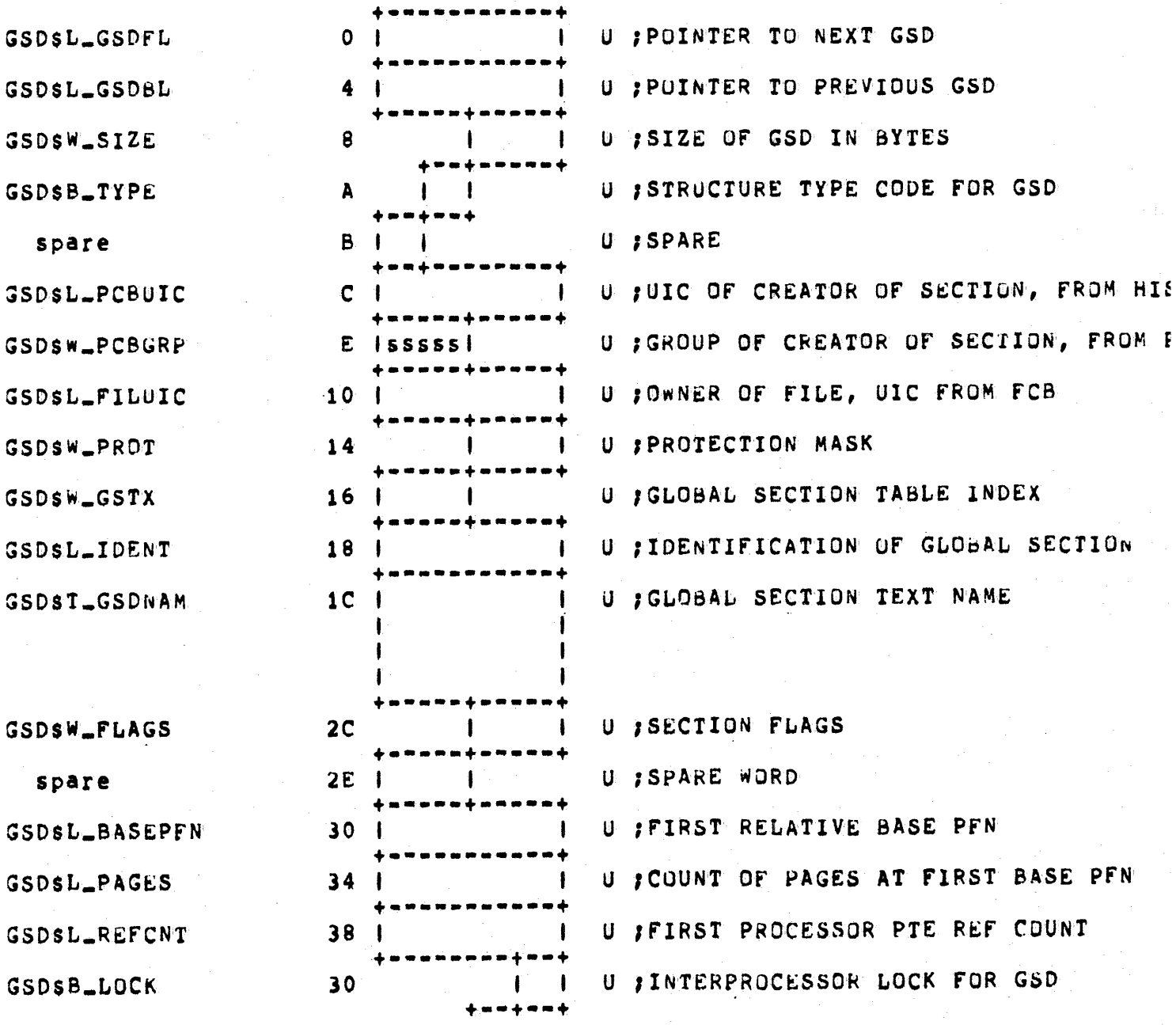
```
;+
; COMMON EVENT BLOCK
;-
        $CEBDEF

;SHMEM FIELDS IN THIS WORD
;SHMEM FIELDS IN THIS WORD

;
; THE FOLLOWING FIELDS ARE DEFINED FOR SHARED MEMORY COMMON EVENT BLOCKS.
; CEB$L_SHB, CEB$W_INDX, AND CEB$L_MASTER ARE CONTAINED IN THE SLAVE CEB WHIL
; CEB$L_VASLAVE1 IS THE OFFSET IN THE MASTER CEB TO THE FIRST SLAVE CEB.
;
;SHMEM MASTER CEB, FIELDS IN NEXT N
; LONGWORDS ARE PROCESSOR REFCNTS
; (ONE WORD FOR EACH PROCESSOR)
; (OFFSET IS COMPUTED AT RUN-TIME)

                                +-----------+
CEB$L_CEBFL              0      |           |   U ;POINTER TO NEXT COMMON EVENT BLOCK
                                +-----------+
CEB$L_CEBBL              4      |           |   U ;POINTER TO PREVIOUS COMMON EVENT BL
                                +-----+-----+
CEB$W_SIZE               8            |     |   U ;SIZE OF COMMON EVENT BLOCK IN BYTES
                                   +--+-----+
CEB$B_TYPE               A        |  |           U ;STRUCTURE TYPE CODE FOR CEB
                                +--+--+
CEB$B_STS                B      |  |              U ;STATUS FLAGS FOR CEB
                                +--+---------+
CEB$L_PID                C      |            |   U ;PID OF CREATOR
                                +------------+
CEB$L_EFC               10      |            |   U ;EVENT FLAGS (32 BIT VECTOR)
                                +------------+
CEB$L_WQFL              14      |            |   U ;HEAD OF WAIT QUEUE
                                +------------+
CEB$L_WQBL              18      |            |   U ;TAIL OF WAIT QUEUE
                                +-----+-----+
CEB$W_WQCNT             1C            |     |   U ;WAIT QUEUE COUNT(LENGTH)
                                      +--+--+
CEB$B_LOCK              1C               |ss|   U ;SHMEM MASTER CEB, # OF PORT OWNING
                                      +--+--+
CEB$B_PROCCNT           1D            |ss|      U ;SHMEM MASTER CEB, MAX # OF PROCESSO
                                +-----+--+
CEB$W_STATE             1E      |     |          U ;CEF WAIT STATE NUMBER
                                +--+--+
CEB$B_CREATPORT         1E        |ss|           U ;SHMEM MASTER CEB, # OF CREATOR PORT
                                +--+--+
CEB$B_DELETPORT         1F      |ss|             U ;SHMEM MASTER CEB, # OF DELETER PORT
                                +--+---------+
CEB$L_UIC               20      |            |   U ;USER IDENT OF CEB CREATOR
                                +-----+-----+
CEB$W_GRP               22      |sssss|          U ;GROUP NUMBER OF OWNER
                                +-----+-----+
CEB$W_PROT              24            |     |   U ;PROTECTION MASK
                                +-----+-----+
CEB$W_REFC              26      |     |          U ;REFERENCE COUNT FOR CEB
                                +-----+-----+
CEB$T_EFCNAM            28      |            |   U ;EVENT CLUSTER TEXT NAME
                                |            |
                                |            |
```



```
CEB$L_SHB                38 |           |  U ;SHMEM SLAVE CEB, SHMEM CTL BLK ADR
                            +-----------+
CEB$L_VASLAVE1           38 |ssssssssss|  U ;SHMEM MASTER CEB, PTR TO 1ST SLAVE
                            +-----+-----+
CEB$W_INDX               3C |     |     |  U ;SHMEM SLAVE CEB, INDEX TO MASTER CE
                            +-----+-----+
  spare                  3E |     |     |  U ;SHMEM SLAVE CEB,
                            +-----+-----+
CEB$L_MASTER             40 |           |  U ;SHMEM SLAVE CEB, VA OF MASTER CEB
                            +-----------+

CEB$L_CEBFL                                    ;POINTER TO NEXT COMMON EVENT BLOCK
     0  CEB$V_VALID          1          1  U ;SHMEM MASTER CEB, SET IF VALID ENTR
     1  CEB$V_LOCKED         1          2  U ;SHMEM MASTER CEB, SET IF ENTRY LOCK
     2  CEB$V_REFCNTLCK      1          4  U ;SHMEM MASTER CEB, LOCKED FOR REFCN

CEB$B_STS                                      ;STATUS FLAGS FOR CEB
     0  CEB$V_NOQUOTA        1  no mask  U ;NO QUOTA UPDATE
     1  CEB$V_PERM           1  no mask  U ;PERMANENT CLUSTER
```

```
;+
; FKB - FORK BLOCK
;
; A FORK BLOCK DESCRIBES THE CONTEXT OF A FORK PROCESS. EACH UNIT CONTROL
; BLOCK CONTAINS A FORK BLOCK AS ITS FIRST SIX LONGWORDS.
;-

        $FKBDEF

                                +------------+
FKB$L_FQFL               0      |            |   U ;FORK QUEUE FORWARD LINK
                                +------------+
FKB$L_FQBL               4      |            |   U ;FORK QUEUE BACKWARD LINK
                                +------+-----+
FKB$W_SIZE               8             |     |   U ;SIZE OF FKB IN BYTES
                                   +---+-----+
FKB$B_TYPE               A         |   |         U ;STRUCTURE TYPE OF FKB
                                +--+---+
FKB$B_FIPL               B      |  |            U ;FORK INTERRUPT PRIORITY LEVEL
                                +--+---------+
FKB$L_FPC                C      |            |   U ;FORK PC
                                +------------+
FKB$L_FR3               10      |            |   U ;FORK R3
                                +------------+
FKB$L_FR4               14      |            |   U ;FORK R4
                                +------------+
```

```
;+
; GLOBAL SECTION DESCRIPTOR BLOCK
;-
        $GSDDEF


;
; THE FOLLOWING FIELDS ARE ONLY FOUND IN EXTENDED GSD'S.  THESE ARE USED
; WHENEVER A GSD IS NEEDED WITHOUT A SECTION TABLE ENTRY, I.E., FOR SHARED
; MEMORY AND FOR PAGES MAPPED BY PFN.
;

;
; THE FOLLOWING FIELDS ARE CONTAINED ONLY IN SHARED MEMORY GSD'S.  THE LENGTH
; GSD$C_SHMGSDLNG, IS ONLY THE CONSTANT SIZE OF THE GSD.  IN ADDITION, THERE
; ONE LONGWORD FOR EACH PROCESSOR AND TWO LONGWORDS FOR EACH BASE PFN-SIZE PA
;
;FIRST PFN/PAGE COUNT PAIR
```



```
GSD$B_PROCCNT          31        |  |      U ;NUMBER OF PROCESSOR REF. COUNTS IN
                               +--+--+
GSD$B_CREATPORT        32        |  |      U ;PORT # FOR CREATOR PROCESSOR
                            +--+--+
GSD$B_DELETPORT        33  |  |           U ;PORT # FOR DELETOR PROCESSOR
                           +--+---------+
GSD$L_BASPFN1          34  |            |  U ;FIRST BASE PFN FOR SECTION PAGES
                           +------------+
GSD$L_BASCNT1          38  |            |  U ;CNT OF SECTION PAGES AT FIRST BASE
                           +------------+
  spare                3C  |            |  U ;    NOTE:  THE NUMBER OF QUADWORDS
                           |            |
                           +------------+
  spare                44  |            |  U ;    RESERVED FOR PFN/PAGE COUNTS PA
                           |            |
                           +------------+
  spare                4C  |            |  U ;    MUST BE EQUAL TO GSD$C_PFNBASMA
                           |            |
                           +------------+
GSD$L_PTECNT1          54  |            |  U ;PTE COUNT FOR FIRST PROCESSOR
                           +------------+

GSD$L_GSDFL                                    ;POINTER TO NEXT GSD
     0  GSD$V_VALID          1          1  U ;SH MEM GSD FLAG, SET IF VALID ENT
     1  GSD$V_LOCKED         1          2  U ;SH MEM GSD FLAG, SET IF ENTRY LOCK
     2  GSD$V_DELPEND        1          4  U ;SH MEM GSD FLAG, GS MARKED FOR DEL
     3  GSD$V_INITFAIL       1          8  U ;SH MEM GSD FLAG, SET WHEN GS INIT
     4  GSD$V_DUPGSD         1         10  U ;SH MEM GSD FLAG, DUPLICATE GSD FOU
```

```
;+
; IAF - IMAGE ACTIVATOR FIXUP SECTION
;
; THE IMAGE ACTIVATOR FIXUP SECTION IS AN IMAGE SECTION THAT IS CREATED
; BY THE LINKER AND USED BY THE IMAGE ACTIVATOR TO MODIFY THE IMAGE AS
; IT IS ACTIVATED.  THIS IS DONE TO MAINTAIN THE POSITION INDEPENDENCE
; OF EXIERNAL REFERENCES.
;-

        $IAFDEF

                                        +
IAF$L_IAFLINK           0               |   U ; Link for image activator use
                                        +
IAF$L_FIXUPLNK          0               |   U ; Link for shareable image fixups
                                        +
IAF$L_SIZE              0               |   U ; Size of fixed part of IAF
                                        +
IAF$L_FLAGS             0               |   U ; Flags
                                        +
IAF$L_G_FIXOFF          0               |   U ; Offset to g^ address data
                                        +
IAF$L_DOTADROFF         0               |   U ; Offset to .address fixup data
                                        +
IAF$L_CHGPRIOFF         0               |   U ; Offset to isect change prot. data
                                        +
IAF$L_SHLSTOFF          0               |   U ; Offset to shareable image list
                                        +
IAF$L_SHRIMGCNT         0               |   U ; Number of shareable images in shl
                                        +
IAF$L_SHLEXTRA          0               |   U ; Number of extra shareable images
                                        +
IAF$L_PERMCTX           0               |   U ; Permanent sharable image context
                      +-------------+
  spare                 0 |             |   U ; Spare
                      +-------------+
  spare                 4 |             |   U ; Spare
                      +-------------+
  spare                 8 |             |   U ; Spare
                      +-------------+
  spare                 C |             |   U ; Spare
                      +-------------+
  spare                10 |             |   U ; Spare
                      +-------------+
  spare                14 |             |   U ; Spare
                      +-------------+

IAF$L_FLAGS                                   ; Flags
     0  IAF$V_SHR          1  no mask    U ; This is in a shareable image
```

```
;+
; TEMPORARY PROCESSOR PRIORITY LEVEL DEFINITIONS
;-
        $IPLDEF
```

IPL$_ASTDEL	2	AST DELIVERY INTERRUPT
IPL$_SCHED	3	RESCHEDULING INTERRUPT
IPL$_IOPOST	4	I/O POSTPROCESSING INTERRUPT
IPL$_QUEUEAST	6	FORK LEVEL USED FOR AST QUEUING
IPL$_SYNCH	7	SYSTEM WIDE SYNCHRONIZATION LEVEL
IPL$_TIMER	7	SOFTWARE TIMER INTERRUPT
IPL$_MAILBOX	11	FORK IPL FOR MAILBOX INTERRUPT
IPL$_HWCLK	24	HARDWARE CLOCK INTERRUPT
IPL$_POWER	31	BLOCK POWER FAIL INTERRUPT

```
;+
; IMAGE SECTION DESCRIPTOR DEFINITIONS
;-

        $ISDDEF


;+
; MATCH CONTROL VIELD VALUES
;-

;+
; ISD TYPE FIELD DEFINITIONS
;-

;NO SPECIAL ACTION REQUIRED

                                  +------+
ISD$W_SIZE              0         |      |   U ;SIZE IN BYTES OF THIS ISD
                        +------+------+
ISD$W_PAGCNT            2 |      |          U ;# OF PAGES DESCRIBED BY THIS ISD
                        +------+------+
ISD$L_VPNPFC            4 |             |   U ;VPN & PFC VIELDS
                        +---+---------+
ISD$B_PFC               7 |ss|          |   U ;PAGE FAULT CLUSTER
                        +---+---------+
ISD$L_FLAGS             8 |             |   U ;FLAGS AND ISD TYPE
                        +---+---------+
ISD$B_TYPE              B |ss|          |   U ;ISD TYPE CODE
                        +---+---------+
ISD$L_VBN               C |             |   U ;BASE VIRTUAL BLOCK NUMBER
                        +-------------+
ISD$L_IDENT            10 |             |   U ;IDENT FOR GLOBAL SECTION
                        +-------------+
ISD$T_GBLNAM           14 |             |   U ;GLOBAL NAME COUNTED STRING
                          |             |
                          |             |
                          |             |
                          +-------------+

ISD$L_VPNPFC                                    ;VPN & PFC VIELDS
     0  ISD$V_VPN          21  no mask   U ; STARTING VIRTUAL PAGE NUMBER
    15  ISD$V_P1            1  no mask   U ; P1 SPACE
    16  ISD$V_SYSTEM        1  no mask   U ; SYSTEM SPACE
    17    spare             1  no mask   U ; SPARE
    18  ISD$V_PFC           8  no mask   U ; PAGE FAULT CLUSTER
     0  ISD$V_VPG          23  no mask   U ; VIRTUAL PAGE INCLUDING P1 & S

ISD$L_FLAGS                                     ;FLAGS AND ISD TYPE
     0  ISD$V_GBL           1         1   U ; GLOBAL
     1  ISD$V_CRF           1         2   U ; COPY ON REFERENCE
     2  ISD$V_DZRO          1         4   U ; DEMAND ZERO PAGE
     3  ISD$V_WRT           1         8   U ; WRITABLE
     4  ISD$V_MATCHCTL      3        70   U ; IDENT MATCH CONTROL FIELD
     7  ISD$V_LASTCLU       1        80   U ; ISD IS PART OF LAST P0 SPACE CLUS
     8  ISD$V_COPYALWAY     1       100   U ; COPY ALWAYS FROM USER IMAGE
     9  ISD$V_BASED         1       200   U ; ISECT IS BASED
     A  ISD$V_FIXUPVEC      1       400   U ; ISECT IS FIXUP SECTION
     B    spare             6     1F800   U ; UNUSED, RESERVED FOR FUTURE USE
```

```
11   ISDsV_VECTOR        1     20000   U ; VECTOR CONTAINED IN IMAGE SECTION
12   ISDsV_PROTECT       1     40000   U ; IMAGE SECTION IS PROTECTED
13     spare             5    F80000   U ; UNUSED, RESERVED FOR FUTURE USE
```

```
;+
; Job Information Block - Structure containing common context for a set
;                         of related processes.
;-
        $JIBDEF
; Primary day hours, from 0:00 to 23:00
; Off day access hours, 0:00 to 23:00
; See pdayflags
                          +------------+
JIB$Q_PRIV              0 |            |   U ; Privilege mask
                          |            |
                          +------------+
JIB$L_ARB               0 |ssssssssss|   U ; Access rights block
                          +------+-----+
JIB$W_SIZE              8        |     |   U ; Size of structure in bytes
                            +--+-----+
JIB$B_TYPE              A   |  |         U ; Structure type code
                          +--+--+
JIB$B_DAYTYPES          B |  |            U ; Set bits 0-6 flag non-prime days
                          +--+---------+
JIB$L_UICLIST           C |            |   U ; Address of UIC list
                          +------------+
JIB$L_UIC               C |ssssssssss|   U ; UIC **** temp until UIC list ****
                          +------------+
JIB$L_BYTCNT           10 |            |   U ; Buffered I/O byte count avail
                          +------------+
JIB$L_BYTLM            14 |            |   U ; Original value for Byte count
                          +------------+
JIB$L_PBYTCNT          18 |            |   U ; Paged pool byte count remaining
                          +------------+
JIB$L_PBYTLIM          1C |            |   U ; Paged pool byte limit
                          +------+-----+
JIB$W_FILCNT           20        |     |   U ; Open File count remaining
                          +------+-----+
JIB$W_FILLM            22 |      |        U ; Open file limit
                          +------+-----+
JIB$W_TQCNT            24        |     |   U ; Timer queue entry count remaining
                          +------+-----+
JIB$W_TQLM             26 |      |        U ; Timer queue entry limit
                          +------+-----+
JIB$L_PGFLQUOTA        28 |            |   U ; Paging file quota
                          +------------+
JIB$L_PGFLCNT          2C |            |   U ; Paging file limit
                          +------------+
JIB$L_CPULIM           30 |            |   U ; CPU time quota remaining
                          +------+-----+
JIB$W_PRCCNT           34        |     |   U ; Count of subprocesses existing
                          +------+-----+
JIB$W_PRCLIM           36 |      |        U ; Limit on number of subprocesses
                          +------+-----+
JIB$W_SHRFCNT          38        |     |   U ; Shared file block count remaing
                          +------+-----+
JIB$W_SHRFLIM          3A |      |        U ; Shared file count limit
                          +------+-----+
JIB$W_ENQCNT           3C        |     |   U ; Enqueue count avail
                          +------+-----+
JIB$W_ENQLM            3E |      |        U ; Enqueue limit
                          +------+-----+
JIB$L_MPID             40 |            |   U ; PID of master process
                          +------------+
```

```
JIB$L_JLNAMFL          44 |           |  U ; Forward link for job-wide logical
                          +-----------+
JIB$L_JLNAMBL          48 |           |  U ; Back link for job-wide logical na
                          +-----------+
JIB$T_USERNAME         4C |           |  U ; User name for easy access
                          |           |
                          |           |
                          +-----------+
JIB$T_ACCOUNT          58 |           |  U ; Account name for resident access
                          |           |
                          +-----------+
JIB$L_PDAYHOURS        60 |           |  U ; Field describing primary day acce
                          +--+--------+
JIB$B_PDAYFLAGS        63 |ss|           U ; Flags associated with primary day
                          +--+--------+
JIB$L_ODAYHOURS        64 |           |  U ; Field describing off day access
                          +--+--------+
JIB$B_ODAYFLAGS        67 |ss|           U ; Flags associated with off day usa
                          +--+

JIB$B_PDAYFLAGS                                 ; Flags associated with primary day
    **  JIB$V_DISDIALUP     1  no mask  U ; Set disallows dial-in use of acco
     1  JIB$V_DISNETWOR     1  no mask  U ; Set disallows network use of acc
     2    spare             4  no mask  U ; SYSTEM SPACE
     6  JIB$V_TERMRIA       1  no mask  U ; Flags terminal as a remote termin
     7  JIB$V_TERMDIAL      1  no mask  U ; Flags terminal as a dial up
```

```
;
; KNOWN FILE IMAGE HEADER DEFINITIONS
;

        $KFHDEF


;
; THE REMAINDER OF THIS STRUCTURE CONTAINS THE IMAGE HEADER OF THE
; SPECIFIED KNOWN FILE.  THE LOCATION KFI$L_IMGHDR IN THE KNOWN FILE
; ENTRY POINTS KFH$C_LENGTH INTO THIS STRUCTURE, I.E AT THE IMAGE HEADER
; ITSELF.
;
                          +-------------+
KFH$L_BUFEND            0 |             |   U ;ADDRESS OF END OF KNOWN FILE HEADE
                          +-------------+
KFH$L_KFIADR            4 |             |   U ;ADDRESS OF ASSOCIATED KNOWN FILE E
                          +------+------+
KFH$W_SIZE              8        |      |   U ;SIZE OF DYNAMIC STRUCTURE
                             +---+------+
KFH$B_TYPE              A    |   |          U ;DYNAMIC STRUCTURE TYPE
                          +--+---+
  spare                 B |   |             U ;SPARE BYTE
                          +---+
```

```
;
; KNOWN FILE ENTRY DEFINITIONS
;
        $KFIDEF


                                +-----------+
KFI$L_KFIQFL            0       |           |   U ;KNOWN FILE QUEUE FORWARD LINK
                                +-----------+
KFI$L_KFIQBL            4       |           |   U ;KNOWN FILE QUEUE BACK LINK
                                +-----+-----+
KFI$W_SIZE              8       |     |     |   U ;SIZE OF BLOCK
                              +---+-----+
KFI$B_TYPE              A     |   |           U ;STRUCTURE TYPE
                              +---+---+
KFI$B_KFICTL            B     |   |           U ;CONTROL BITS
                              +---+     +---+
KFI$B_DEVUCB            C             |   |   U ;DEVICE UCB OFFSET
                                      +---+
KFI$B_DEVNAM            C             |ss |   U ;NAME THE ABOVE CONSISTENTLY
                                  +---+---+
KFI$B_DIRNAM            D         |   |       U ;DIRECTORY NAME STRING OFFSET
                              +---+---+
KFI$B_FILNAM            E     |   |           U ;FILE NAME STRING OFFSET
                              +---+---+
KFI$B_TYPNAM            F     |   |           U ;FILE TYPE STRING OFFSET
                              +---+   +-------+
KFI$W_REFCNT           10         |   |       U ;REFERENCE COUNT
                              +---+-----+
KFI$B_KFIQNUM          12     |   |           U ;KFIQ NUMBER (INDEX INTO VECTOR OF
                              +---+---+
KFI$B_KFISEQ           13     |   |           U ;KNOWN FILE ENTRY SEQUENCE NUMBER
                              +---+   +-------+
KFI$W_FLAGS            14         |   |       U ;FLAGS WORD
                                +-----+-------+
KFI$W_GBLSECCNT        16       |     |       U ;GLOBAL SECTION COUNT IF SHARED
                                +-----+-------+
KFI$L_USECNT           18       |             U ;USAGE COUNTER
                                +-------------+
KFI$L_WINDOW           1C       |             U ;WCB ADDRESS IF OPEN
                                +-----+-------+
KFI$W_FID              1C         |sssss|     U ;FILE ID
                                  +-----+
KFI$W_FID_NUM          1C         |sssss|     U ;FILE NUMBER FIELD OF FILE ID
                                +-----+-------+
KFI$W_FID_SEQ          1E       |sssss|       U ;FILE SEQUENCE NUMBER FIELD OF FILE
                                +-----+-------+
KFI$L_IMGHDR           20       |             U ;IMAGE HEADER ADDRESS IF RESIDENT
                                +-----+-------+
KFI$W_FID_RVN          20         |sssss|     U ;RELATIVE VOLUME NUMBER FIELD OF FI
                                +-----+-------+
KFI$Q_PROCPRIV         24       |             U ;PROCESS PRIVILEGE MASK
                                |             |
                                +-------+---+-+
KFI$B_MATCHCTL         2C               |   |   U ;GLOBAL SECTION MATCH CONTROL
                                      +---+---+
  spare                2D           |   |       U ;SPARE BYTE
                                +-----+---+
KFI$W_AMECOD           2E       |     |         U ;IMAGE HEADER CODE SPECIFYING AME
                                +-----+-------+
```

```
KFI$L_IDENT             30 |            |   U ;GLOBAL SECTION IDENT VALUE
                           +------------+

KFI$B_KFICTL                                    ;CONTROL BITS
     0  KFI$V_KFIHD          1          1   U ;KNOWN FILE HEADER BLOCK
     1  KFI$V_FILIDOPEN      1          2   U ;OPEN BY FILE ID IF SET
     2  KFI$V_DONOTOPEN      1          4   U ;DO NOT OPEN THE FILE IF SET
     3    spare              3         38   U ;SPARE
     6  KFI$V_NOREPLACE      1         40   U ;DELETE AND DO NOT REPLACE ENTRY
     7  KFI$V_MARKDEL        1         80   U ;ENTRY IS TO BE DELETED

KFI$B_KFISEQ                                    ;KNOWN FILE ENTRY SEQUENCE NUMBER
     0  KFI$V_KFISEQ         2          3   U ;SEQUENCE NUMBER FIELD

KFI$W_FLAGS                                     ;FLAGS WORD
     0  KFI$V_KP_OPEN        1          1   U ;KEEP THE IMAGE FILE OPEN
     1  KFI$V_KP_RESHDR      1          2   U ;MAKE IMAGE HEADER RESIDENT
     2  KFI$V_KP_SHARED      1          4   U ;MAKE IMAGE SHARED
     3  KFI$V_PROTECT        1          8   U ;KNOWN FILE WAS INSTALLED PROTECTED
     4    spare              2         30   U ;SPARE BITS
     6  KFI$V_LIM            1         40   U ;LINKABLE IMAGE
     7  KFI$V_PROCPRIV       1         80   U ;USE PROCESS PRIVILEGE MASK
     8  KFI$V_IS_RESHDR      1        100   U ;IMAGE HEADER BLOCK IS RESIDENT
     9  KFI$V_IS_SHARED      1        200   U ;IMAGE IS SHARED
     A    spare              4       3C00   U ;SPARE BITS
     E  KFI$V_SHMIDENT       1       4000   U ;SHARED MEMORY IDENT ALREADY SET
     F  KFI$V_COMPATMOD      1       8000   U ;IMAGE IS COMPATABILITY MODE
```

```
;+
; PCB DEFINITIONS
;-

        $PCBDEF

;FOR INTERACTIVE JOBS
;(PROCESS CREATION ONLY)

PCB$L_SQFL          0   U ;STATE QUEUE FORWARD LINK
PCB$L_SQBL          4   U ;STATE QUEUE BACKWARD LINK
PCB$W_SIZE          8   U ;SIZE IN BYTES
PCB$B_TYPE          A   U ;STRUCTURE TYPE CODE FOR PCB
PCB$B_PRI           B   U ;PROCESS CURRENT PRIORITY
PCB$B_ASTACT        C   U ;ACCESS MODES WITH ACTIVE ASTS
PCB$B_ASTEN         D   U ;ACCESS MODES WITH ASTS ENABLED
PCB$W_MTXCNT        E   U ;COUNT OF MUTEX SEMAPHORES OWNED
PCB$L_ASTQFL       10   U ;AST QUEUE FORWARD LINK(HEAD)
PCB$L_ASTQBL       14   U ;AST QUEUE BACK LINK(TAIL)
PCB$L_PHYPCB       18   U ;PHYSICAL ADDRESS OF HW PCB
PCB$L_OWNER        1C   U ;PID OF CREATOR
PCB$L_WSSWP        20   U ;SWAP FILE DISK ADDRESS
PCB$L_STS          24   U ;PROCESS STATUS FLAGS
PCB$L_WTIME        28   U ;TIME AT START OF WAIT
PCB$B_PRISAV       28   U ;SAVED CURRENT PRIORITY
PCB$B_PRIBSAV      29   U ;SAVE BASE PRIORITY
PCB$W_STATE        2C   U ;PROCESS STATE
PCB$B_WEFC         2E   U ;WAITING FF CLUSTER NUMBER
PCB$B_PRIB         2F   U ;BASE PRIORITY
PCB$W_APTCNT       30   U ;ACTIVE PAGE TABLE COUNT
PCB$W_TMBU         32   U ;TERMINATION MAILBOX UNIT NO.
PCB$W_GPGCNT       34   U ;GLOBAL PAGE COUNT IN WS
PCB$W_PPGCNT       36   U ;PROCESS PAGE COUNT IN WS
PCB$W_ASTCNT       38   U ;AST COUNT REMAINING
```

```
PCB$W_BIOCNT        3A |      |        U ;BUFFERED I/O COUNT REMAINING
                       +------+------+
PCB$W_BIOLM         3C        |      |  U ;BUFFERED I/O LIMIT
                       +------+------+
PCB$W_DIOCNT        3E |      |        U ;DIRECT I/O COUNT REMAINING
                       +------+------+
PCB$W_DIOLM         40        |      |  U ;DIRECT I/O COUNT LIMIT
                       +------+------+
PCB$W_PRCCNT        42 |      |        U ;SUBPROCESS COUNT
                       +------+------+
PCB$T_TERMINAL      44 |             |  U ;TERMINAL DEVICE NAME STRING
                       |             |
                       +-------------+
PCB$L_PQB           4C |             |  U ;POINTER TO PROCESS QUOTA BLOCK
                       +-------------+
PCB$L_EFWM          4C |ssssssssssss|  U ;EVENT FLAG WAIT MASK
                       +-------------+
PCB$L_EFCS          50 |             |  U ;LOCAL EVENT FLAG CLUSTER,SYSTEM
                       +-------------+
PCB$L_EFCU          54 |             |  U ;LOCAL EVENT FLAG CLUSTER, USER
                       +-------------+
PCB$L_EFC2P         58 |             |  U ;POINTER TO GLOBAL CLUSTER #2
                       +-------------+
PCB$L_EFC3P         5C |             |  U ;POINTER TO GLOBAL CLUSTER #3
                       +-------------+
PCB$L_PID           60 |             |  U ;PROCESS IDENTIFICATION
                       +-------------+
PCB$L_PHD           64 |             |  U ;PROCESS HEADER ADDRESS
                       +-------------+
PCB$T_LNAME         68 |             |  U ;LOGICAL NAME OF PROCESS
                       |             |
                       |             |
                       |             |
                       +-------------+
PCB$L_JIB           78 |             |  U ;ADDRESS OF JOB INFORMATION BLOCK
                       +-------------+
PCB$G_PRIV          7C |             |  U ;CURRENT PRIVILEGE MASK
                       |             |
                       +-------------+
PCB$L_ARB           84 |             |  U ;ADDRESS OF ACCESS RIGHTS BLOCK
                       +-------------+
PCB$L_UIC           88 |             |  U ;LOGON UIC OF PROCESS
                       +------+------+
PCB$W_MEM           88        |sssss|  U ;MEMBER NUMBER IN UIC
                       +------+------+
PCB$W_GRP           8A |sssss|        U ;GROUP NUMBER IN UIC
                       +------+------+
PCB$L_LOCKQFL       8C |             |  U ;LOCK QUEUE FORWARD LINK
                       +-------------+
PCB$L_LOCKQBL       90 |             |  U ;LOCK QUEUE BACKWARD LINK
                       +-------------+
PCB$L_DLCKPRI       94 |             |  U ;DEADLOCK RESOLUTION PRIORITY
                       +-------------+
PCB$L_IPAST         98 |             |  U ;VECTOR OF MODE BITS FOR IPASTS
                       +-------------+

PCB$L_STS                                  ;PROCESS STATUS FLAGS
     0  PCB$V_RES          1  no mask  U ; RESIDENT, IN BALANCE SET
     1  PCB$V_DELPEN       1  no mask  U ; DELETE PENDING
     2  PCB$V_FORCPEN      1  no mask  U ; FORCE EXIT PENDING
     3  PCB$V_INQUAN       1  no mask  U ; INITIAL QUANTUM IN PROGRESS
```

```
 4   PCB$V_PSWAPM        1   no mask   U ; PROCESS SWAP MODE (1=NOSWAP)
 5   PCB$V_RESPEN        1   no mask   U ; RESUME PENDING, SKIP SUSPEND
 6   PCB$V_SSFEXC        1   no mask   U ; SYSTEM SERVICE EXCEPTION ENABLE (K
 7   PCB$V_SSFEXCE       1   no mask   U ; SYSTEM SERVICE EXCEPTION ENABLE (E
 8   PCB$V_SSFEXCS       1   no mask   U ; SYSTEM SERVICE EXCEPTION ENABLE (S
 9   PCB$V_SSFEXCU       1   no mask   U ; SYSTEM SERVICE EXCEPTION ENABLE (U
 A   PCB$V_SSRWAIT       1   no mask   U ; SYSTEM SERVICE RESOURCE WAIT DISAB
 B   PCB$V_SUSPEN        1   no mask   U ; SUSPEND PENDING
 C   PCB$V_WAKEPEN       1   no mask   U ; WAKE PENDING, SKIP HIBERNATE
 D   PCB$V_WALL          1   no mask   U ; WAIT FOR ALL EVENTS IN MASK
 E   PCB$V_BATCH         1   no mask   U ; PROCESS IS A BATCH JOB
 F   PCB$V_NOACNT        1   no mask   U ; NO ACCOUNTING FOR PROCESS
10   PCB$V_SWPVBN        1   no mask   U ; WRITE FOR SWP VBN IN PROGRESS
11   PCB$V_ASTPEN        1   no mask   U ; AST PENDING
12   PCB$V_PHDRES        1   no mask   U ; PROCESS HEADER RESIDENT
13   PCB$V_HIBER         1   no mask   U ; HIBERNATE AFTER INITIAL IMAGE ACTI
14   PCB$V_LOGIN         1   no mask   U ; LOGIN WITHOUT READING AUTH FILE
15   PCB$V_NETWRK        1   no mask   U ; NETWORK CONNECTED JOB
16   PCB$V_PWRAST        1   no mask   U ; POWER FAIL AST
17   PCB$V_NODELET       1   no mask   U ; NO DELETE
18   PCB$V_DISAWS        1   no mask   U ; 1=DISABLE AUTOMATIC WS ADJUSTMENT
```

```
;+
; PAGE FILE CONTROL BLOCK
;-

        $PFLDEF

;
; ***** L_VBN, L_WINDOW, AND B_PFC MUST BE THE SAME OFFSET VALUES AS THE
; ***** EQUIVALENTLY NAMED OFFSETS IN $SECDEF
;
;BIT = 1 MEANS AVAILABLE
; BACKING STORE ADDRESS

                                +------------+
PFL$L_BITMAP              0     |            |   U ;ADDRESS OF START OF BIT MAP
                                +------------+
PFL$L_STARTBYTE           4     |            |   U ;STARTING BYTE OFFSET TO SCAN
                                +------+-----+
PFL$W_SIZE                8     |      |     |   U ;SIZE OF PAGE FILE CONTROL BLOCK
                                   +---+------+
PFL$B_TYPE                A        |   |      |   U ;PAGE FILE CONTROL BLOCK TYPE COD
                                +---+---+      |
PFL$B_PFC                 B     |   |          |   U ;PAGE FAULT CLUSTER FOR PAGE READS
                                +---+----------+
PFL$L_WINDOW              C     |            |   U ;WINDOW ADDRESS
                                +------------+
PFL$L_VBN                 10    |            |   U ;BASE VBN
                                +------------+
PFL$L_BITMAPSIZ           14    |            |   U ;SIZE IN BYTES OF PAGE FILE
                                +------------+
PFL$L_FREPAGCNT           18    |            |   U ;COUNT - 1 OF PAGES WHICH MAY BE ALI
                                +------------+
PFL$L_MAXVBN              1C    |            |   U ;MASK APPLIED TO PTE WITH PAGING FI
                                +------+-----+
PFL$W_ERRORCNT            20    |      |     |   U ;COUNT OF POTENTIALLY BAD PAGES
                                   +---+------+
PFL$B_ALLOCSIZ            22       |   |      |   U ;CURRENT ALLOCATION REQUEST SIZE
                                +---+---+      |
PFL$B_FLAGS               23    |   |          |   U ;FLAGS BYTE FOR THIS PAGE FILE
                                +---+----------+
PFL$L_BITMAPLOC           24    |            |   U ;BITMAP FOLLOWS PFL HEADER
                                +------------+

PFL$B_FLAGS                                          ;FLAGS BYTE FOR THIS PAGE FILE
      0  PFL$V_INITED          1          1   U ;THIS PAGE FILE IS USABLE
      1  PFL$V_PAGFILFUL       1          2   U ;REQUEST FOR PAGING SPACE HAS FAILE
      2  PFL$V_SWPFILFUL       1          4   U ;REQUEST FOR SWAPPING SPACE HAS FAI
      3    spare               4         78   U ;SPARE BITS FOR EXPANSION
      7  PFL$V_STOPPER         1         80   U ;RESERVED FOR ALL TIME (MUST NEVER
```

```
;+
; PFN DATA BASE DEFINITIONS
;-

        $PFNDEF
;
; VIELD DEFINITIONS IN PFN$AB_STATE
;
;
; ***** THE FOLLOWING SPARE BIT MUST BE USED FOR EXTENSION OF THE LOC FIELD
; ***** OR ALTERNATIVELY THE DELCON BIT MUST BE MOVED ADJACENT TO LOC
;
;
; VIELD DEFINITIONS IN PFN$AB_TYPE
;
;
; VIELD DEFINITIONS IN PFN$AL_BAK
;
;
; LOCATION VIELD VALUES
;
;WHEN REFCNT = 0 RELEASE PFN
;
; PAGE TYPE VIELD DEFINITIONS
;

     0  PFN$V_LOC           3          7  U ;LOCATION OF PAGE
     3    spare             1   no mask  U ;NOT IN USE
     4  PFN$V_DELCON        1         10  U ;DELETE PFN CONTENTS WHEN REF=0
     5    spare             2   no mask  U ;NOT IN USE
     7  PFN$V_MODIFY        1         80  U ;MODIFY BIT
     0  PFN$V_PAGTYP        3          7  U ;PAGE TYPE
     3    spare             1   no mask  U ;NOT IN USE
     4  PFN$V_COLLISION     1         10  U ;EMPTY COLLISION QUEUE WHEN PAGE RE
     5  PFN$V_BADPAG        1         20  U ;BAD PAGE BIT
     6  PFN$V_RPTEVT        1         40  U ;REPORT EVENT ON I/O COMPLETE
     0  PFN$V_BAK          23    7FFFFF  U ;BACKUP ADDRESS
    17  PFN$V_GBLBAK        1    800000  U ;GLOBAL BACKING STORE ADDRESS
    18  PFN$V_PGFLX         8  FF000000  U ;PAGE FILE INDEX
```

```
;+
; A PROCESS HEADER CONTAINS THE SWAPPABLE SCHEDULER AND
; MEMORY MANAGEMENT DATA BASES FOR A PROCESS IN THE
; BALANCE SET.
;-

        $PHDDEF

;
; WORKING SET LIST POINTERS - THESE CONTAIN LONG WORD OFFSETS FROM THE
; BEGINNING OF THE PROCESS HEADER.
;
;
; THE FOLLOWING THREE WORDS SPECIFY THE MAXIMUM AND INITIAL WORKING SET
; SIZES FOR THE PROCESS.  RATHER THAN CONTAINING THE COUNT OF PAGES
; THEY CONTAIN THE LONG WORD INDEX TO WHAT WOULD BE THE LAST WORKING
; SET LIST ENTRY.
;
;
; PROCESS SECTION TABLE DATA BASE
;       PSTBASOFF IS THE BYTE OFFSET (INTEGRAL # OF PAGES) FROM THE
; BEGINNING OF THE PROCESS HEADER TO THE 1ST LONG WORD BEYOND THE
; PROCESS SECTION TABLE.
;       THE WORDS, PSTLAST AND PSTFREE ARE SECTION TABLE INDICES WHICH
; ARE THE NEGATIVE LONG WORD INDEX FROM THE END OF THE SECTION TABLE TO
; THE SECTION TABLE ENTRY.
;
;FIRST LONG WORD NOT IN PST
;PST GROWS BACKWARDS FROM HERE
;ADR OF LAST PSTE ALLOCATED
;
; CREATE/DELETE PAGE CONTEXT
;
;***** MUST BE QUAD WORD AWAY FROM FREP1VA
;OF THE P0 AND P1 PAGE TABLES
;
; QUOTAS AND LIMITS
;
;LAST QUANTUM OVERFLOW

;PROCESS HEADER PAGES
;SAVE AREA

;
; THE NEXT TWO I/O COUNTERS MUST BE ADJACENT
;



;
; PAGE TABLE STATISTICS
;
; # OF LOCKED WSLE'S IN THIS PAGE TABLE
; # OF VALID WSLE'S IN THIS PAGE TABLE
; 1 OR MORE LOCKED WSLE
; 1 OR MORE VALID WSLE
; WHICH HAVE NON-ZERO PTE'S
; ABOVE REQUIRED WSFLUID MINIMUM
;
; HARDWARE PCB PORTION OF PROCESS HEADER
;
```

```
;  MUST RUN ON PRIMARY IN MP SYSTEM
;
;  END OF FIXED PORTION OF PROCESS HEADER
;

PHD$Q_PRIVMSK           0   |           |   U ;PRIVILEGE MASK
                            |           |
                            +-----+-----+
PHD$W_WSLIST            8   |     |     |   U ;1ST WORKING SET LIST ENTRY
                            +-----+-----+
PHD$W_WSAUTH            A   |     |     |   U ;AUTHORIZED WORKING SET SIZE
                            +-----+-----+
PHD$W_WSLOCK            C   |     |     |   U ;1ST LOCKED WORKING SET LIST ENTRY
                            +-----+-----+
PHD$W_WSDYN             E   |     |     |   U ;1ST DYNAMIC WORKING SET LIST ENTRY
                            +-----+-----+
PHD$W_WSNEXT           10   |     |     |   U ;LAST WSL ENTRY REPLACED
                            +-----+-----+
PHD$W_WSLAST           12   |     |     |   U ;LAST WSL ENTRY IN LIST
                            +-----+-----+
PHD$W_WSAUTHEXT        14   |     |     |   U ;AUTHORIZED WS EXTENT
                            +-----+-----+
PHD$W_WSEXTENT         16   |     |     |   U ;MAX WORKING SET SIZE AGAINST BORR_
                            +-----+-----+
PHD$W_WSQUOTA          18   |     |     |   U ;QUOTA ON WORKING SET SIZE
                            +-----+-----+
PHD$W_DFWSCNT          1A   |     |     |   U ;DEFAULT WORKING SET SIZE
                            +-----+-----+
PHD$L_PAGFIL           1C   |           |   U ;PAGING FILE INDEX, LONG WORD REF
                            +--+--------+
PHD$B_PAGFIL           1F   |ss|        |   U ;PAGING FILE INDEX, BYTE REFERENCE
                            +--+--------+
PHD$L_PSTBASOFF        20   |           |   U ;BYTE OFFSET TO BASE OF PST
                            +-----+-----+
PHD$W_PST              20   |sssss|     |   U ;*****TEMP*****
                            +-----+     |
PHD$W_PSTLAST          24   |     |     |   U ;END OF PROCESS SECTION TABLE
                            +-----+-----+
PHD$W_PSTFREE          26   |     |     |   U ;HEAD OF FREE PSTE LIST
                            +-----+-----+
PHD$L_FREPOVA          28   |           |   U ;1ST FREE VIRTUAL ADR AT END OF P0
                            +-----------+
PHD$L_FREPTECNT        2C   |           |   U ;CNT OF FREE PTE'S BETWEEN THE ENDS
                            +-----------+
PHD$L_FREP1VA          30   |           |   U ;1ST FREE VIRTUAL ADR AT END OF P1 S
                            +---------+-+-+
PHD$B_DFPFC            34             | | |   U ;DEFAULT PAGE FAULT CLUSTER
                                    +-+-+-+
PHD$B_PGTBPFC          35       |     | |     U ;PAGE TABLE CLUSTER FACTOR
                            +---+-----+-+
PHD$W_FLAGS            36   |     |     |   U ;FLAGS WORD
                            +-----+-----+
PHD$L_CPUTIM           38   |           |   U ;ACCUMULATED CPU TIME CHARGED
                            +-----+-----+
PHD$W_QUANT            3C   |     |     |   U ;ACCUMULATED CPU TIME SINCE
                            +-----+-----+
PHD$W_PRCLM            3E   |     |     |   U ;SUBPROCESS QUOTA
                            +-----+-----+
PHD$W_ASTLM            40   |     |     |   U ;AST LIMIT
                            +-----+-----+
```

```
PHD$W_PHVINDEX      42 |      |        U ;PROCESS HEADER VECTOR INDEX
                       +------+------+
PHD$W_BAK           44        |      |  U ;POINTER TO BACKUP ADDRESS VECTOR FO
                              +------+
PHD$W_WSLX          46 |      |        U ;POINTER TO WORKING SET LIST INDEX
                       +------+
PHD$W_PSTBASMAX     46 |sssss|        U ;Lw OFFSET TO TOP PST ADDRESS
                       +------+------+
PHD$L_PAGEFLTS      48 |             |  U ;COUNT OF PAGE FAULTS
                       +------+------+
PHD$W_WSSIZE        4C        |      |  U ;CURRENT ALLOWED WORKING SET SIZE
                              +------+
PHD$W_SWAPSIZE      4E |      |        U ;CURRENT SWAP BLOCK ALLOCAIION
                       +------+------+
PHD$L_DIOCNT        50 |             |  U ;DIRECT I/O COUNT
                       +------------+
PHD$L_BIOCNT        54 |             |  U ;BUFFERED I/O COUNT
                       +------------+
PHD$L_CPULIM        58 |             |  U ;LIMIT ON CPU TIME FOR PROCESS
                       +---------+---+
PHD$B_CPUMODE       5C            |   |  U ;ACCESS MODE TO NOTIFY ABOUT CPUTIME
                             +---+---+
PHD$B_AWSMODE       5D        |   |      U ;ACCESS MODE FLAG FOR AUTO WS AST
                       +------+---+
PHD$W_WAITIME       5E |      |        U ;TIME OF DAY OF WAIT REQUEST @6.5536
                       +------+------+
PHD$L_PIWSLELCK     60 |             |  U ; BYTE OFFSET TO BYTE ARRAY OF COUNT
                       +------------+
PHD$L_PIWSLEVAL     64 |             |  U ; BYTE OFFSET TO BYTE ARRAY OF COUNT
                       +------+------+
PHD$W_PICNTLCK      68        |      |  U ; COUNT OF PAGE TABLES CONIAINING
                              +------+
PHD$W_PICNTVAL      6A |      |        U ; COUNT OF PAGE TABLES CONTAINING
                       +------+------+
PHD$W_PICNTACT      6C        |      |  U ; COUNT OF ACTIVE PAGE TABLES
                              +------+
PHD$W_PICNTMAX      6E |      |        U ; MAX COUNT OF PAGE TABLES
                       +------+------+
PHD$W_WSFLUID       70        |      |  U ; GUARANTEED NUMBER OF FLUID WS PAGE
                              +------+
PHD$W_EXTDYNWS      72 |      |        U ; EXTRA DYNAMIC WORKING SET LIST ENT
                       +------+------+
PHD$L_PCB           74 |             |  U ;HARDWARE PCB
                       +------------+
PHD$L_KSP           74 |ssssssssssss|  U ;KERNEL STACK POINTER
                       +------------+
PHD$L_ESP           78 |             |  U ;EXEC STACK POINTER
                       +------------+
PHD$L_SSP           7C |             |  U ;SUPERVISOR STACK POINTER
                       +------------+
PHD$L_USP           80 |             |  U ;USER STACK POINTER
                       +------------+
PHD$L_R0            84 |             |  U ;R0
                       +------------+
PHD$L_R1            88 |             |  U ;R1
                       +------------+
PHD$L_R2            8C |             |  U ;R2
                       +------------+
PHD$L_R3            90 |             |  U ;R3
                       +------------+
PHD$L_R4            94 |             |  U ;R4
```

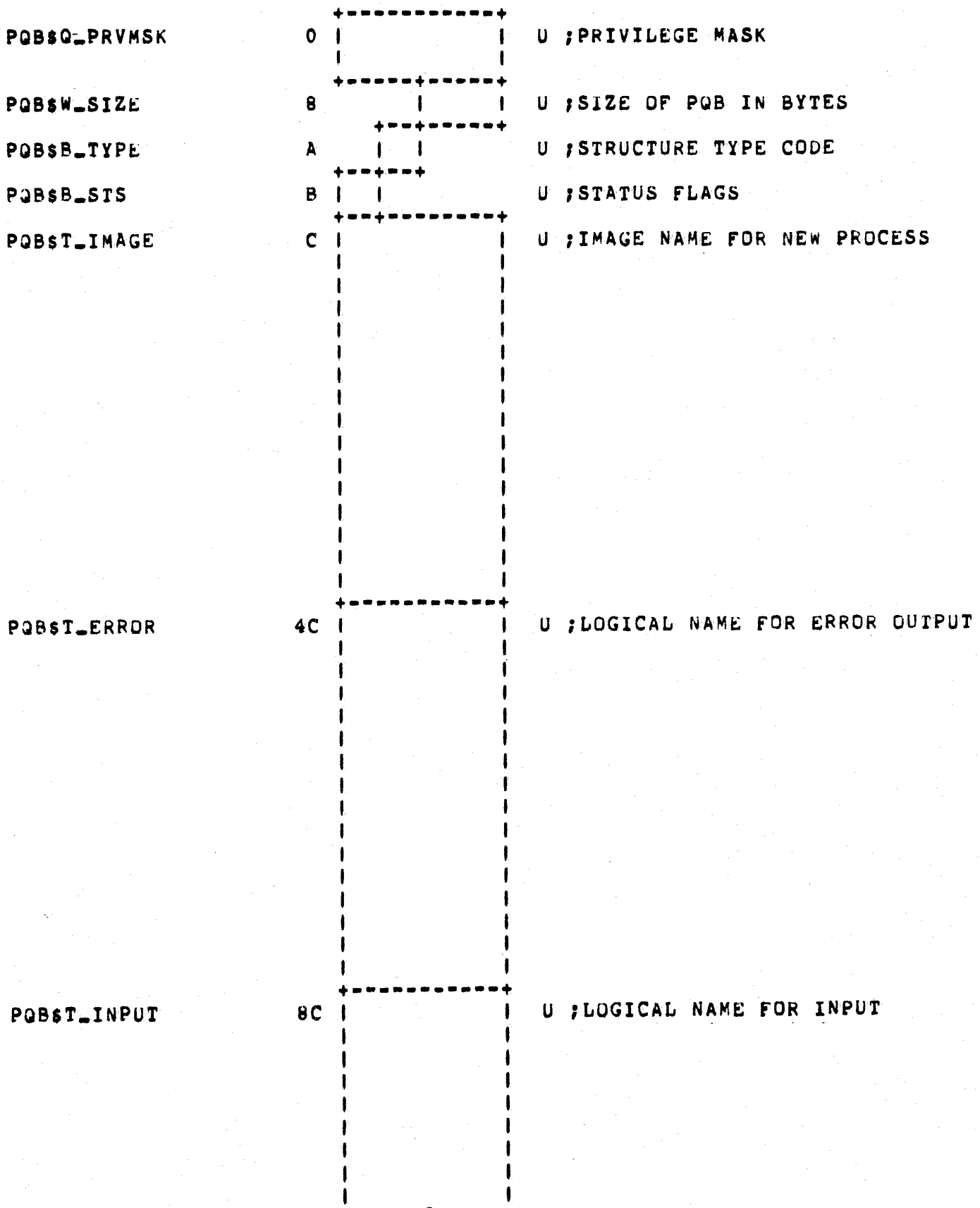
AJ2

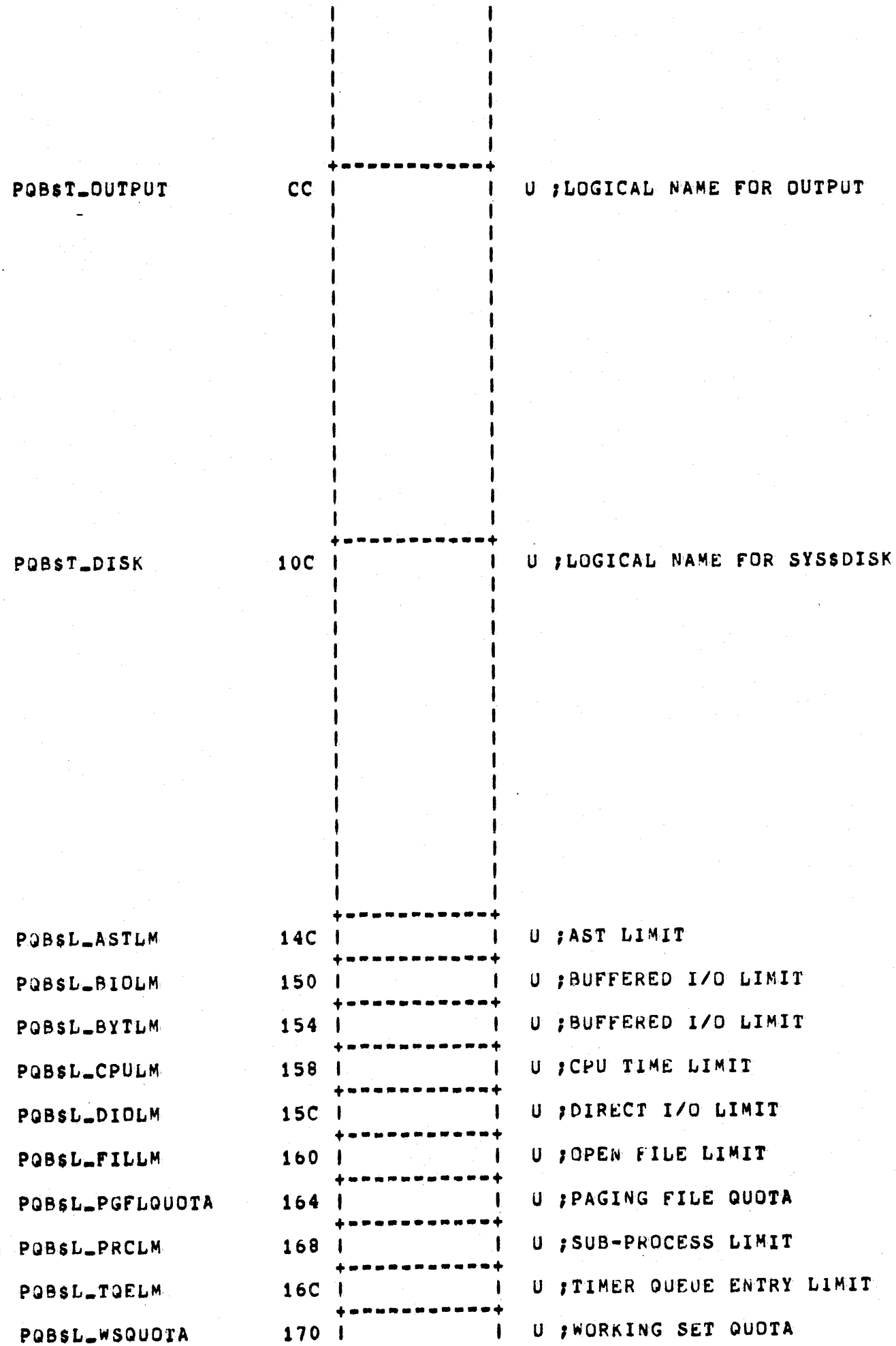
```
                        +-----------+
PHD$L_R5          98    |           |   U ;R5
                        +-----------+
PHD$L_R6          9C    |           |   U ;R6
                        +-----------+
PHD$L_R7          A0    |           |   U ;R7
                        +-----------+
PHD$L_R8          A4    |           |   U ;R8
                        +-----------+
PHD$L_R9          A8    |           |   U ;R9
                        +-----------+
PHD$L_R10         AC    |           |   U ;R10
                        +-----------+
PHD$L_R11         B0    |           |   U ;R11
                        +-----------+
PHD$L_R12         B4    |           |   U ;R12
                        +-----------+
PHD$L_R13         B8    |           |   U ;R13
                        +-----------+
PHD$L_PC          BC    |           |   U ;PC
                        +-----------+
PHD$L_PSL         C0    |           |   U ;PROGRAM STATUS LONGWORD
                        +-----------+
PHD$L_P0BR        C4    |           |   U ;P0 BASE REGISTER
                        +-----------+
PHD$L_P0LRASTL    C8    |           |   U ;P0LR, ASTLVL
                        +---+-------+
PHD$B_ASTLVL      CB    |ss|           U ;AST LEVEL SUBFIELD
                        +---+-------+
PHD$L_P1BR        CC    |           |   U ;P1 BASE REGISTER
                        +-----------+
PHD$L_P1LR        D0    |           |   U ;P1 LENGTH REGISTER
                        +-----+-----+
PHD$W_EMPTPG      D4          |     |   U ;COUNT OF EMPTY WORKING SET PAGES
                        +-----+-----+
PHD$W_RESPGCNT    D6    |     |         U ;RESIDENT PAGE COUNT
                        +-----+-----+
PHD$W_REQPGCNT    D8          |     |   U ;REQUIRED PAGE COUNT
                        +-----+-----+
PHD$W_CWSLX       DA    |     |         U ;CONTINUATION WSLX
                        +-----+-----+
PHD$Q_AUTHPRIV    DC    |           |   U ;AUTHORIZED PRIVILEGES MASK
                        |           |
                        +-----------+
PHD$Q_IMAGPRIV    E4    |           |   U ;INSTALLED IMAGE PRIVILEGES MASK
                        |           |
                        +-----------+
PHD$L_RESLSTH     EC    |           |   U ;POINTER TO RESOURCE LIST
                        +-----------+
PHD$L_IMGCNT      F0    |           |   U ;IMAGE COUNTER BUMPED BY SYSRUNDWN
                        +-----------+
PHD$L_PFLTRATE    F4    |           |   U ;PAGE FAULT RATE
                        +-----------+
PHD$L_PFLREF      F8    |           |   U ;PAGE FAULTS AT END OF LAST INTERVAL
                        +-----------+
PHD$L_TIMREF      FC    |           |   U ;TIME AT END OF LAST INTERVAL
                        +-----------+
PHD$L_MPINHIBIT   100   |           |   U ;COUNT OF REASONS WHY PROCESS
                        +-----------+
PHD$L_PGFLTIO     104   |           |   U ;COUNT OF PAGEFAULT I/O
                        +---------+-+
```

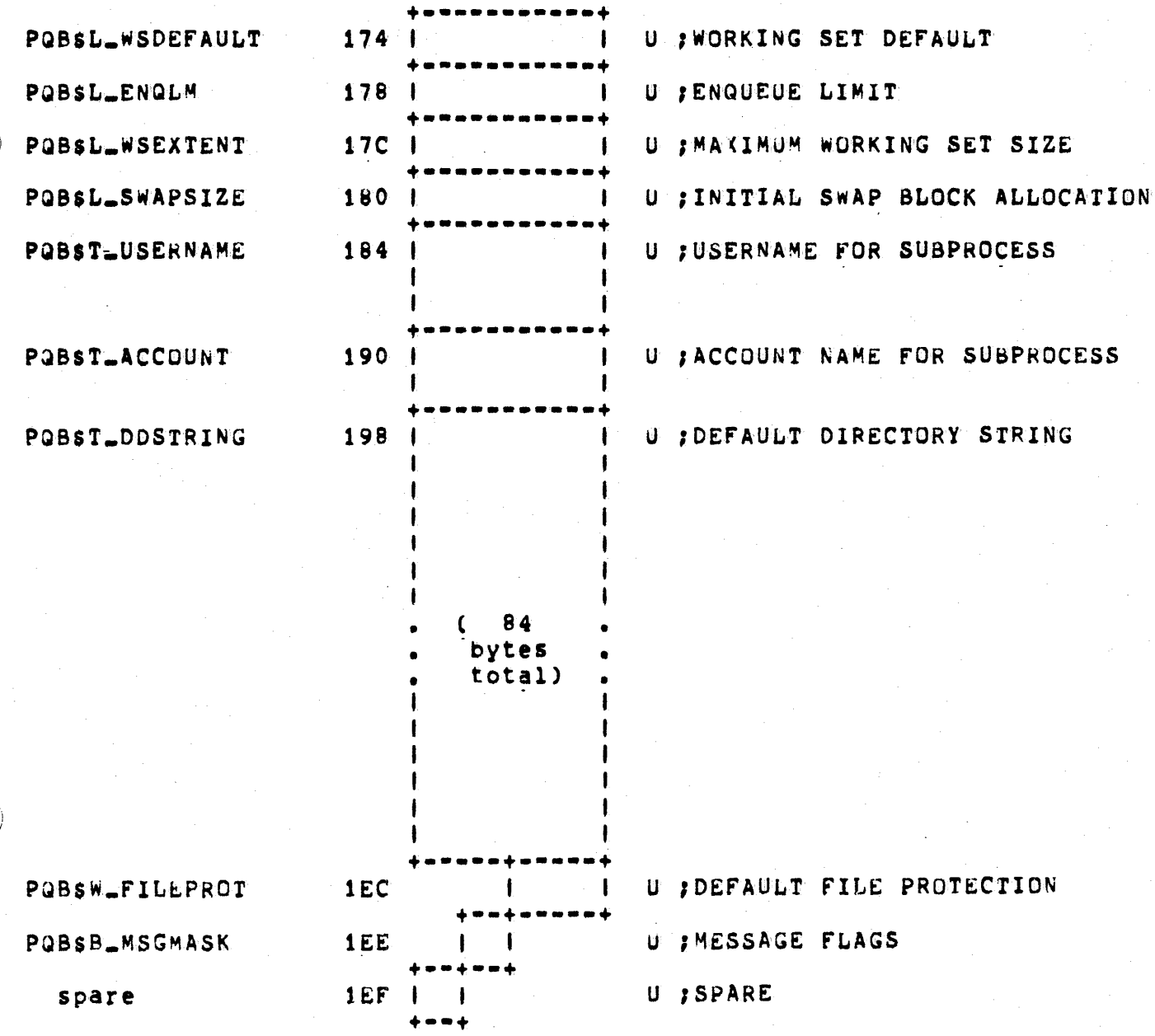
```
PHD$B_AUTHPRI       108          |  |   U ;INITIAL PROCESS PRIORITY
                              +--+--+
  spare             109       |  |      U ;SPARE
                    +---------+--+
  spare             10A |     |         U ;SPARE
                    +-----------+
PHD$L_EXTRACPU      10C |             |   U ;ACCUMULATED CPU TIME LIMIT EXTENSIO
                    +-------------+
  spare             110 |             |   U ;SPARE
                    +-------------+
  spare             114 |             |   U ;SPARE
                    +-------------+
  spare             118 |             |   U ;SPARE
                    +-------------+
  spare             11C |             |   U ;SPARE
                    +-------------+
  spare             120 |             |   U ;SPARE
                    +-------------+
  spare             124 |             |   U ;SPARE
                    +-------------+
  spare             128 |             |   U ;SPARE
                    +-------------+
  spare             12C |             |   U ;SPARE
                    +-------------+
  spare             130 |             |   U ;SPARE
                    +-------------+
PHD$L_SPARE         134 |             |   U ;SPARE
                    +-------------+
  spare             138 |             |   U ;SPARE
                    +-------------+
  spare             13C |             |   U ;SPARE
                    +-------------+
  spare             140 |             |   U ;SPARE
                    +-------------+
  spare             144 |             |   U ;SPARE
                    +-------------+
  spare             148 |             |   U ;SPARE
                    +-------------+
  spare             14C |             |   U ;SPARE
                    +-------------+
  spare             150 |             |   U ;SPARE
                    +-------------+
  spare             154 |             |   U ;SPARE
                    +-------------+
  spare             158 |             |   U ;SPARE
                    +-------------+
  spare             15C |             |   U ;SPARE
                    +-------------+
  spare             160 |             |   U ;SPARE
                    +-------------+
  spare             164 |             |   U ;SPARE
                    +-------------+
  spare             168 |             |   U ;SPARE
                    +-------------+
  spare             16C |             |   U ;SPARE
                    +-------------+
  spare             170 |             |   U ;SPARE
                    +-------------+
  spare             174 |             |   U ;SPARE
                    +-------------+
PHD$L_WSL           178 |             |   U ;FIRST WORKING SET LIST ENTRY
```

```
                +------------+

PHD$W_FLAGS                                             ;FLAGS WORD
      0   PHD$V_PFMFLG          1          1   U ;PAGE FAULT MONITORING ENABLED
      1   PHD$V_DALCSIX         1          2   U ;NEED TO DEALLOCATE SECTION INDICES
      2   PHD$V_WSPEAKCHK       1          4   U ;CHECK FOR NEW WORKING SET SIZE (PRO
      3   PHD$V_NOACCVIO        1          8   U ;SET AFTER INSWAP OF PROCESS HEADER
      4   PHD$V_IWSPEAKCK       1         10   U ;CHECK FOR NEW WORKING SET SIZE (IMA

PHD$L_POLRASTL                                          ;POLR, ASTLVL
      0   PHD$V_POLR           24  no mask   U ;P0 LENGTH REGISTER
     18   PHD$V_ASTLVL          8  no mask   U ; AST LEVEL
```

```
;+
; PROCESS QUOTA BLOCK DEFINITION
;-

        $PQBDEF
```




PQB$T_OUTPUT CC U ;LOGICAL NAME FOR OUTPUT
PQB$T_DISK 10C U ;LOGICAL NAME FOR SYS$DISK
PQB$L_ASTLM 14C U ;AST LIMIT
PQB$L_BIOLM 150 U ;BUFFERED I/O LIMIT
PQB$L_BYTLM 154 U ;BUFFERED I/O LIMIT
PQB$L_CPULM 158 U ;CPU TIME LIMIT
PQB$L_DIOLM 15C U ;DIRECT I/O LIMIT
PQB$L_FILLM 160 U ;OPEN FILE LIMIT
PQB$L_PGFLQUOTA 164 U ;PAGING FILE QUOTA
PQB$L_PRCLM 168 U ;SUB-PROCESS LIMIT
PQB$L_TQELM 16C U ;TIMER QUEUE ENTRY LIMIT
PQB$L_WSQUOTA 170 U ;WORKING SET QUOTA


PQB$L_WSDEFAULT 174 U ;WORKING SET DEFAULT
PQB$L_ENQLM 178 U ;ENQUEUE LIMIT
PQB$L_WSEXTENT 17C U ;MAXIMUM WORKING SET SIZE
PQB$L_SWAPSIZE 180 U ;INITIAL SWAP BLOCK ALLOCATION
PQB$T_USERNAME 184 U ;USERNAME FOR SUBPROCESS
PQB$T_ACCOUNT 190 U ;ACCOUNT NAME FOR SUBPROCESS
PQB$T_DDSTRING 198 U ;DEFAULT DIRECTORY STRING
(84 bytes total)
PQB$W_FILEPROT 1EC U ;DEFAULT FILE PROTECTION
PQB$B_MSGMASK 1EE U ;MESSAGE FLAGS
spare 1EF U ;SPARE

$PRDEF MACRO - SYMBOLIC NAMES FOR PROCESSOR REGISTERS

Symbolic Name	Register
PR$_KSP	Kernel stack pointer
PR$_ESP	Executive stack pointer
PR$_SSP	Supervisor stack pointer
PR$_USP	User stack pointer
PR$_ISP	Interrupt stack pointer
PR$_P0BR	P0 base register
PR$_P0LR	P0 limit register
PR$_P1BR	P1 base register
PR$_P1LR	P1 limit register
PR$_SBR	System base register
PR$_SLR	System limit register
PR$_PCBB	Process control block base register
PR$_SCBB	System control block base register
PR$_IPL	Interrupt priority level register
PR$_ASTLVL	AST level register
PR$_SIRR	Software interrupt request register
PR$_SISR	Software interrupt summary register
PR$_MAPEN	Mapping enable register
PR$_TBIA	Translation buffer invalidate all
PR$_TBIS	Translation buffer invalidate single
PR$_ICCS	Interval clock control status register
PR$_NICR	Interval clock next interval register
PR$_ICR	Interval clock interval count register
PR$_TODR	Time of day register
PR$_RXCS	Console receiver control status register
PR$_RXDB	Console receiver data buffer register
PR$_TXCS	Console transmit control status register
PR$_TXDB	Console transmit data buffer register
PR$_ACCS	Accelerator control status register
PR$_ACCR	Accelerator reserved
PR$_PME	Performance monitor enable
PR$_SID	System identification register
PR$_SID_TYP730	Processor is a VAX-11/730
PR$_SID_TYP750	Processor is a VAX-11/750
PR$_SID_TYP780	Processor is a VAX-11/780
PR$_WCSA	WCS address register
PR$_WCSD	WCS data register
PR$_SBIFS	SBI fault status register
PR$_SBIS	SBI silo register
PR$_SBISC	SBI comparator register
PR$_SBIMT	SBI maintenance register
PR$_SBIER	SBI error register
PR$_SBITA	SBI timeout address register
PR$_SBIQC	SBI quadword clear register

```
;+
; DEFINE PAGE TABLE ENTRY VIELDS AND VALUES
;-
        $PTEDEF
;
; VIELD DEFINITION FOR "VALID" PTE'S
;
;
; VIELD DEFINITIONS FOR VARIOUS INVALID FORMS OF PTE
;
; OVERLAYS MODIFY BIT
;  TO BE FORGOTTEN

;+
; PROTECTION FIELD DEFINITIONS
;-
;+
; OWNER FIELD DEFINITIONS
;-
     0  PTE$V_PFN          21    1FFFFF   U ; PAGE FRAME NUMBER
    15  PTE$V_WINDOW        1    200000   U ; WINDOW BIT
    16    spare             1   no mask   U ; RESERVED
    17  PTE$V_OWN           2   1800000   U ; MODE OF THE OWNER
    19    spare             1   no mask   U ; RESERVED
    1A  PTE$V_MODIFY        1   4000000   U ; MODIFY BIT
    1B  PTE$V_PROT          4  78000000   U ; PROTECTION
    1F  PTE$V_VALID         1  80000000   U ; VALID BIT
     0  PTE$V_STX          16   no mask   U ; SECTION TABLE INDEX
    10  PTE$V_CRF           1     10000   U ; COPY ON REFERENCE
    11  PTE$V_DZRO          1     20000   U ; DEMAND ZERO
    12  PTE$V_WRT           1     40000   U ; SECTION FILE IS ACCESSED FOR WRITI
    13    spare             3   no mask   U ; SPARE
    16  PTE$V_TYP0          1    400000   U ; LOW ORDER BIT OF PTE TYPE
    17    spare             2   no mask   U ; OWNER FIELD
    19    spare             1   no mask   U ; RESERVED
    1A  PTE$V_TYP1          1   4000000   U ; HIGH ORDER BIT OF PTE TYPE
     0  PTE$V_PGFLVB       22    3FFFFF   U ; PAGE FILE VBN
     0    spare            21   no mask   U ; SPACING
    15  PTE$V_CHKPNT        1    200000   U ; FORGET THAT THIS PAGE HAS A BACK P
     0  PTE$V_GPTX          22    3FFFFF   U ; GLOBAL PAGE TABLE INDEX
```

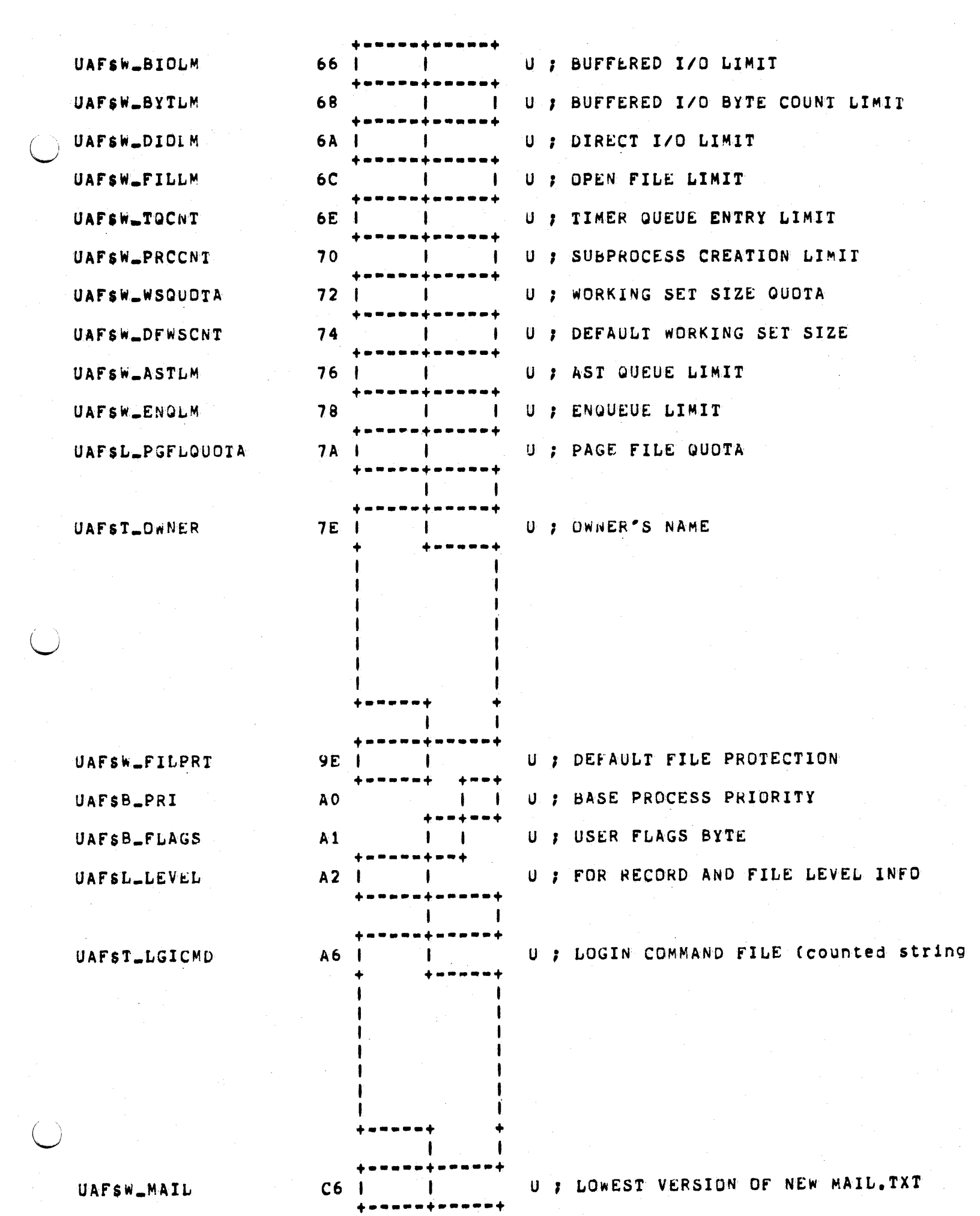
```
;+
; RESTART PARAMETER BLOCK DEFINITIONS
;-
        $RPBDEF

                          +-----------+
RPB$L_BASE             0  |           |   U ;PHYSICAL BASE ADDRESS OF 64K BLOCK
                          +-----------+
RPB$L_RESTART          4  |           |   U ;POINTER TO RESTART ROUTINE (PHYSICA
                          +-----------+
RPB$L_CHKSUM           8  |           |   U ;CHECKSUM OF BYTES 0-7F OF RESTART R
                          +-----------+
RPB$L_RSTRTFLG         C  |           |   U ;RESTART IN PROGRESS FLAG
                          +-----------+
RPB$L_HALTPC          10  |           |   U ;PC AT RESTART/HALT
                          +-----------+
RPB$L_HALTPSL         14  |           |   U ;PSL AT RESTART/HALT
                          +-----------+
RPB$L_HALTCODE        18  |           |   U ;CODE DESCRIBING RESTART REASON
                          +-----------+
RPB$L_BOOTR0          1C  |           |   U ;SAVED BOOT PARAMETER R0
                          +---------+---+
RPB$B_RODEVTYP        1C            |ss|   U ; DEVICE TYPE SUBFIELD
                              +---+---+
  spare               1D        |ss|       U ; RESERVED
                          +-----+---+
RPB$W_ROUBVEC         1E  |sssss|           U ; UNIBUS INT VECTOR SUBFIELD
                          +-----+-----+
RPB$L_BOOTR1          20  |           |   U ;SAVED BOOT PARAMETER R1
                          +-----------+
RPB$L_BOOTR2          24  |           |   U ;SAVED BOTT PARAMETER R2
                          +-----------+
RPB$L_BOOTR3          28  |           |   U ;SAVED BOOT PARAMETER R3
                          +-----------+
RPB$L_BOOTR4          2C  |           |   U ;SAVED BOOT PARAMETER R4
                          +-----------+
RPB$L_BOOTR5          30  |           |   U ;SAVED BOOT PARAMETER R5
                          +-----------+
RPB$L_IOVEC           34  |           |   U ;ADDRESS OF BOOTSTRAP QIO VECTOR
                          +-----------+
RPB$L_IOVECSZ         38  |           |   U ;SIZE OF BOOT QIO ROUTINE
                          +-----------+
RPB$L_FILLBN          3C  |           |   U ;LOGICAL BLOCK NUMBER OF BOOT FILE
                          +-----------+
RPB$L_FILSIZ          40  |           |   U ;SIZE OF BOOT FILE
                          +-----------+
RPB$Q_PFNMAP          44  |           |   U ;DESCRIPTOR FOR PFN BITMAP
                          |           |
                          +-----------+
RPB$L_PFNCNT          4C  |           |   U ;COUNT OF PHYSICAL PAGES
                          +-----------+
RPB$L_SVASPT          50  |           |   U ;SYSTEM VIRTUAL ADDRESS OF SPT
                          +-----------+
RPB$L_CSRPHY          54  |           |   U ;UBA DEVICE CSR ADDRESS (PHYSICAL)
                          +-----------+
RPB$L_CSRVIR          58  |           |   U ;UBA DEVICE CSR ADDRESS (VIRTUAL)
                          +-----------+
RPB$L_ADPPHY          5C  |           |   U ;ADAPTER CONFIGURATION REGISTER (PHY
                          +-----------+
RPB$L_ADPVIR          60  |           |   U ;ADAPTER CONFIGURATION REGISTER (VIR
```

```
RPB$W_UNIT          64                  U ;UNIT NUMBER
RPB$B_DEVTYP        66                  U ;DEVICE TYPE CODE
RPB$B_SLAVE         67                  U ;SLAVE UNIT NUMBER
RPB$T_FILE          68                  U ;BOOT FILE NAME (ASCIC)
RPB$B_CONFREG       90                  U ;ARRAY OF ADAPTER TYPES
RPB$B_HDRPGCNT      A0                  U ;COUNT OF HEADER PAGES
RPB$B_BOOTNDT       A1                  U ;NEXUS DEVICE TYPE OF BOOT ADAPTER
  spare             A2                  U ;SPARE
RPB$L_ISP           A4                  U ;PWR FAIL INTERRUPT STACK POINTER
RPB$L_PCBB          A8                  U ;PROCESS CONTROL BLOCK BASE
RPB$L_SBR           AC                  U ;SYSTEM BASE REGISTER
RPB$L_SCBB          B0                  U ;SYSTEM CONTROL BLOCK BASE
RPB$L_SISR          B4                  U ;SOFTWARE INTERRUPT SUMMARY REGISTEF
RPB$L_SLR           B8                  U ;SYSTEM LENGTH REGISTER
RPB$L_MEMDSC        BC                  U ;MEMORY DESCRIPT. - PAGCNT, TR, SA
RPB$L_BUGCHK        FC                  U ;BUGCHECK LOOP ADDRESSS FOR MP SECOI
RPB$B_WAIT          100                 U ;BUGCHECK LOOP CODE FOR MP SECONDARI
```

```
RPB$L_BOOTR1                                        ;SAVED BOOT PARAMETER R1
     0  RPB$V_NEXUS          4  no mask  U ;NEXUS OF SYSTEM DEVICE ADAPTER
     4  RPB$V_ABUS           2  no mask  U ;ABUS ADAPTER NUMBER OF SBIA

RPB$L_BOOTR5                                        ;SAVED BOOT PARAMETER R5
     0  RPB$V_CONV           1  no mask  U ; CONVERSATIONAL BOOTSTRAP
     1  RPB$V_DEBUG          1  no mask  U ; KEEP DEBUGGER CODE
     2  RPB$V_INIBPT         1  no mask  U ; INITIAL BREAKPOINT
     3  RPB$V_BBLOCK         1  no mask  U ; TRANSFER TO BOOTBLOCK
     4  RPB$V_DIAG           1  no mask  U ; BOOT DIAGNOSTIC FILE
     5  RPB$V_BOOBPT         1  no mask  U ; BOOTSTRAP BREAKPOINT
     6  RPB$V_HEADER         1  no mask  U ; USE START ADDRESS FROM IMAGE HEADE
     7  RPB$V_NOTEST         1  no mask  U ; FLAG TO INHIBIT MEMORY TESTING
     8  RPB$V_SOLICT         1  no mask  U ; SOLICIT BOOT FILE NAME
     9  RPB$V_HALT           1  no mask  U ; HALT BEFORE TRANSFER
     A  RPB$V_NOPFND         1  no mask  U ; INHIBIT PFN DELETION
     B  RPB$V_MPM            1       800 U ; MULTI-PROCESSOR BOOT, USE MA780 ON
     C  RPB$V_USEMPM         1      1000 U ; USE MA780 AS IF IT WERE LOCAL MEMC
     D  RPB$V_MEMTEST        1      2000 U ; USE STRICTER TEST TO VALIDATE MEMC
     E  RPB$V_FINDMEM        1      4000 U ; FIND SUFFICIENT MEMORY TO BOOT (>5
     F    spare             13  no mask  U
    1C  RPB$V_TOPSYS         4 F0000000  U ;SYSTEM DIRECTORY NUMBER

RPB$L_MEMDSC                                        ;MEMORY DESCRIPT. - PAGCNT, TR, BASE
     0  RPB$V_PAGCNT        24  no mask  U ; COUNT OF PAGES FOR THIS MEMORY
    18  RPB$V_TR             8  no mask  U ; TR NUMBER FOR THIS MEMORY
    20  RPB$V_BASEPFN       32  no mask  U ; BASE PFN FOR THIS MEMORY
```

```
;+
; TQE - TIME QUEUE ENTRY
;
; TIME QUEUE ENTRIES ARE UTILIZED TO SET TIMERS, WAKE UP PROCESSES, AND
; FOR INTERNAL SYSTEM SUBROUTINES.
;-

        $TQEDEF

;
; TIME QUEUE ENTRY REQUEST TYPE DEFINITIONS
;


                                +------------+
TQE$L_TQFL                  0   |            |   U ;TIME QUEUE FORWARD LINK
                                +------------+
TQE$L_TQBL                  4   |            |   U ;TIME QUEUE BACKWARD LINK
                                +------+-----+
TQE$W_SIZE                  8         |     |   U ;SIZE OF TQE IN BYTES
                                  +---+-----+
TQE$B_TYPE                  A     |   |         U ;STRUCTURE TYPE FOR TQE
                                +---+---+
TQE$B_RQTYPE                B   |   |           U ;TIME QUEUE ENTRY TYPE
                                +---+---------+
TQE$L_PID                   C   |             |   U ;TIMER OR WAKE REQUEST PROCESS ID
                                +-------------+
TQE$L_FPC                   C   |ssssssssssss|   U ;TIMER SUBROUTINE ADDRESS
                                +-------------+
TQE$L_AST                   10  |             |   U ;ADDRESS OF AST ROUTINE
                                +-------------+
TQE$L_FR3                   10  |ssssssssssss|   U ;TIMER SUBROUTINE SAVED R3
                                +-------------+
TQE$L_ASTPRM                14  |             |   U ;AST PARAMETER
                                +-------------+
TQE$L_FR4                   14  |ssssssssssss|   U ;TIMER SUBROUTINE SAVED R4
                                +-------------+
TQE$Q_TIME                  18  |             |   U ;ABSOLUTE EXPIRATION TIME
                                |             |
                                +-------------+
TQE$Q_DELTA                 20  |             |   U ;DELTA REPEAT TIME
                                |             |
                                +-------------+---+
TQE$B_RMOD                  28                |   U ;ACCESS MODE OF REQUEST
                                        +-----+---+
TQE$B_EFN                   29          |     |     U ;EVENT FLAG NUMBER AND EVENT GROUP
                                +-------+-----+
  spare                     2A  |       |           U ;SPARE WORD
                                +-------+-----+
TQE$L_RQPID                 2C  |             |   U ;REQUESTER PROCESS ID
                                +-------------+

TQE$B_RQTYPE                                        ;TIME QUEUE ENTRY TYPE
     0    spare                  2   no mask   U ; STARTING OFFSET
     2  TQE$V_REPEAT             1         4   U ; REPEAT REQUEST (1=YES)
     3  TQE$V_ABSOLUTE           1         8   U ; Absolute expiration time specifie
```

```
;++
; USER AUTHORIZATION FILE FORMAT
;--

        $UAFDEF
; HOURS, FROM 0:00 TO 23:00
; HOURS, 0:00 TO 23:00
; USAGE: SEE PDAYFLAGS
; 0 MEANS NO LIMIT
; 0 MEANS NO LIMIT
;                           (longword)
UAF$T_USERNAME          0   U ; USERNAME
UAF$Q_PWD               C   U ; HASHED PASSWORD
UAF$L_PWD               C   U ; 32 BIT SUBFIELD
UAF$T_ACCOUNT          14   U ; ACCOUNT NAME
UAF$L_UIC              1C   U ; USER ID CODE
UAF$W_MEM              1C   U ; MEMBER SUBFIELD
UAF$W_GRP              1E   U ; GROUP SUBFIELD
UAF$T_DEFDIR           20   U ; DEFAULT DIRECTORY (counted string)
UAF$T_DEFDEV           40   U ; DEFAULT DEVICE (counted string)
UAF$T_DEFCLI           50   U ; DEFAULT COMMAND INTERPRETER
UAF$B_ENCKYPT          59   U ; ENCRYPTION ALGORITHM
UAF$L_CPUTIM           5A   U ; CPU TIME QUOTA
UAF$Q_PRIV             5E   U ; PROCESS PRIVILEGE VECTOR
```


UAF$W_BIOLM 66 U ; BUFFERED I/O LIMIT
UAF$W_BYTLM 68 U ; BUFFERED I/O BYTE COUNT LIMIT
UAF$W_DIOLM 6A U ; DIRECT I/O LIMIT
UAF$W_FILLM 6C U ; OPEN FILE LIMIT
UAF$W_TQCNT 6E U ; TIMER QUEUE ENTRY LIMIT
UAF$W_PRCCNT 70 U ; SUBPROCESS CREATION LIMIT
UAF$W_WSQUOTA 72 U ; WORKING SET SIZE QUOTA
UAF$W_DFWSCNT 74 U ; DEFAULT WORKING SET SIZE
UAF$W_ASTLM 76 U ; AST QUEUE LIMIT
UAF$W_ENQLM 78 U ; ENQUEUE LIMIT
UAF$L_PGFLQUOTA 7A U ; PAGE FILE QUOTA
UAF$T_OWNER 7E U ; OWNER'S NAME
UAF$W_FILPRT 9E U ; DEFAULT FILE PROTECTION
UAF$B_PRI A0 U ; BASE PROCESS PRIORITY
UAF$B_FLAGS A1 U ; USER FLAGS BYTE
UAF$L_LEVEL A2 U ; FOR RECORD AND FILE LEVEL INFO
UAF$T_LGICMD A6 U ; LOGIN COMMAND FILE (counted string
UAF$W_MAIL C6 U ; LOWEST VERSION OF NEW MAIL.TXT

```
UAF$L_PBYTLM        C8 |           | U ; PAGED BUFFER I/O BYTE COUNT LIMIT
                       +-----+-----+
UAF$W_SHRFILLM      CC       |     | U ; SHARED FILE LIMIT
                       +-----+-----+
UAF$W_USRDATOFF     CE |     |       U ; OFFSET OF COUNTED STRING OF USER D
                       +-----+-----+
UAF$W_SALT          D0       |     | U ; RANDOM PASSWORD SALT
                       +-----+-----+
UAF$W_WSEXTENT      D2 |     |       U ; WORKING SET SIZE LIMIT
                       +-----+-----+
UAF$L_PDAYHOURS     D4 |           | U ; FIELD DESCRIBING PRIMARY DAY ACCES
                       +--+--------+
UAF$B_PDAYFLAGS     D7 |ss|          U ; FLAGS ASSOCIATED WITH PRIMARY DAYS
                       +--+--------+
UAF$L_SDAYHOURS     D8 |           | U ; FIELD DESCRIBING SECONDARY DAY ACC
                       +--+--------+
UAF$B_SDAYFLAGS     DB |ss|          U ; FLAGS ASSOCIATED WITH SECONDARY DA
                       +--+     +--+
UAF$B_MAXPROC       DC          |  | U ; MAXIMUM PROCESSES FOR UIC ALLOWED
                             +--+--+
UAF$B_MAXGRPROC     DD       |  |    U ; MAXIMUM PROCESSES FOR GROUP ALLOWE
                          +--+--+
UAF$B_PRIMEDAYS     DE    |  |       U ; BITS REPRESENTING PRIMARY DAYS
                       +--+--+
  spare             DF |  |          U ; UNUSED, 0
                       +--+--------+
UAF$L_BYTLM         E0 |           | U ; BUFFERED I/O BYTE COUNT LIMIT
                       +-----------+
  spare             E4 |           | U ; fixed length field expansion
                       |           |
                       +-----------+
  spare             EC |           | U
                       |           |
                       |           |
                       +-----------+
  spare             F8 |           | U
                       |           |
                       |           |
                       |           |
                       |           |
                       |           |
                       +-----------+
  spare            110 |           | U
                       |           |
                       |           |
                       +-----------+
  spare            11C |           | U ; variable length field region
                       |           |
                       |           |
                       |           |
                       |           |
                       |           |
                       |           |
                       .  ( 768    .
                       .   bytes   .
                       .   total)  .
                       |           |
                       |           |
                       |           |
                       |           |
                       |           |
```

```
UAF$B_FLAGS                                          ; USER FLAGS BYTE
     0  UAF$V_DISCTLY       1  no mask  U ; DON'T ALLOW USER CONTROL-Y
     1  UAF$V_DEFCLI        1  no mask  U ; ONLY ALLOW USER DEFAULT CLI
    4B  UAF$V_LOCKPWD       1  no mask  U ; DISABLE SET PASSWORD COMMAND
     3  UAF$V_CAPTIVE       1  no mask  U ; NO OVERIDES ALLOWED ON LOGIN-EG /D
     4  UAF$V_DISACNT       1  no mask  U ; DON'T ALLOW INTERACTIVE LOGIN
     5  UAF$V_DISWELCOM     1  no mask  U ; FLAG TO ALLOW LOGIN TO SKIP WELCOM
     6  UAF$V_DISMAIL       1  no mask  U ; FLAG TO ALLOW LOGIN TO SKIP NEW MA

UAF$B_PDAYFLAGS                                      ; FLAGS ASSOCIATED WITH PRIMARY DAYS
     0  UAF$V_DISDIALUP     1  no mask  U ; SET DISALLOWS DIAL-IN USE OF ACCOU
     1  UAF$V_DISNETWOR     1  no mask  U ; SET DISALLOWS NETWORK USE OF ACCOU

UAF$B_PRIMEDAYS                                      ; BITS REPRESENTING PRIMARY DAYS
     0  UAF$V_MONDAY        1  no mask  U ; BIT CLEAR MEANS THIS IS A PRIMARY
     1  UAF$V_TUESDAY       1  no mask  U ; BIT SET MEANS THIS IS AN OFF DAY
     2  UAF$V_WEDNESDAY     0  no mask  U ; HALT BEFORE TRANSFER
     2  UAF$V_THURSDAY      0  no mask  U ; INHIBIT PFN DELETION
     2  UAF$V_FRIDAY        0  no mask  U ; MULTI-PROCESSOR BOOT, USE MA780 ON
     2  UAF$V_SATURDAY      0  no mask  U ; USE MA780 AS IF IT WERE LOCAL ME 1C
     2  UAF$V_SUNDAY        0  no mask  U ; USE STRICTER TEST TO VALIDATE MEMC
```

```
;+
; WCB - WINDOW CONTROL BLOCK
;
; THERE IS A WINDOW CONTROL BLOCK FOF EACH FILE ACCESSED BY A PROCESS.
; IT CONTAINS MAPPING INFORMATION SUCH THAT A LARGE PERCENTAGE OF VIRTUAL
; FILE I/O CAN BE MAPPED FROM VIRTUAL TO LOGICAL BLOCK NUMBERS WITHOUT
; HAVING TO READ THE RESPECTIVE FILE HEADER.
;-

        $WCBDEF
; FILE COMPLETELY
; NOTE - THESE BITS TRACK THE BITS
; IN FIB$L_ACCTL
; FORMAT OF RETRIEVAL POINTER
; RETRIEVAL POINTER FORMAT
                          +-----------+
WCB$L_WLFL              0 |           |   U ; WINDOW LIST FORWARD LINK
                          +-----------+
WCB$L_WLBL              4 |           |   U ; WINDOW LIST BACKWARD LINK
                          +-----+-----+
WCB$W_SIZE              8       |     |   U ; SIZE OF WINDOW BLOCK IN BYTES
                          +---+-----+
WCB$B_TYPE              A   |   |         U ; STRUCTURE TYPE OF WCB
                          +---+---+
WCB$B_ACCESS            B |   |           U ; ACCESS CONTROL BYTE
                          +---+---------+
WCB$L_PID               C |           |   U ; PROCESS ID OF ACCESSOR PROCESS
                          +-----+-----+
WCB$W_REFCNT            E |sssss|         U ; REFERENCE COUNT FOR SHARED WINDOW
                          +-----+-----+
WCB$L_ORGUCB           10 |           |   U ; ADDRESS OF ORIGINAL UCB FROM CCB
                          +-----+-----+
WCB$W_ACON             14       |     |   U ; ACCESS CONTROL INFORMATION
                          +-----+-----+
WCB$W_NMAP             16 |     |         U ; NUMBER OF MAPPING POINTERS
                          +-----+-----+
WCB$L_FCB              18 |           |   U ; ADDRESS OF FCB
                          +-----------+
WCB$L_RVT              1C |           |   U ; ADDRESS OF RELATIVE VOLUME TABLE
                          +-----------+
WCB$L_LINK             20 |           |   U ; LINK TO NEXT WINDOW SEGMENT
                          +-----------+
WCB$L_READS            24 |           |   U ; COUNT OF READS PERFORMED
                          +-----------+
WCB$L_WRITES           28 |           |   U ; COUNT OF WRITES PERFORMED
                          +-----------+
WCB$L_STVBN            2C |           |   U ; STARTING VBN MAPPED BY WINDOW
                          +-----+-----+
WCB$W_P1_COUNT         30       |     |   U ; COUNT FIELD OF FIRST POINTER
                          +-----+-----+
WCB$L_P1_LBN           32 |     |         U ; LBN FIELD OF SECOND POINTER
                          +-----+-----+
                                |     |
                          +-----+-----+
WCB$W_P2_COUNT         36 |     |         U ; COUNT FIELD OF SECOND POINTER
                          +-----+-----+
WCB$L_P2_LBN           38 |           |   U ; LBN FIELD OF FIRST POINTER
                          +-----+-----+
WCB$W_COUNT             0       |     |   U ; COUNT FIELD
```

```
                                +-------+-------+
WCB$L_LBN                     2 |       |       |       U ; LBN FIELD
                                +-------+-------+
                                        |       |
                                        +-------+


WCB$B_ACCESS                                                ; ACCESS CONTROL BYTE
      0   WCB$V_READ              1          1    U ; READ ACCESS ALLOWED (1=YES)
      1   WCB$V_WRITE             1          2    U ; WRITE ACCESS ALLOWED (1=YES)
      2   WCB$V_NOTFCP            1          4    U ; FILE NOT ACCESSED BY FCP IF SET
      3   WCB$V_SHRWCB            1          8    U ; SHARED WINDOW
      4   WCB$V_OVERDRAWN         1         10    U ; FILE ACCESSOR HAS OVERDRAWN HIS QU
      5   WCB$V_COMPLETE          1         20    U ; SET WINDOW MAPS ENTIRE FILE
      6   WCB$V_CATHEDRAL         1         40    U ; LARGE, COMPLEX WINDOW (SIC) TO MAP
      7   WCB$V_EXPIRE            1         80    U ; FILE EXPIRATION DATE MAY NEED TO B

WCB$W_ACON                                                  ; ACCESS CONTROL INFORMATION
      0   WCB$V_NOWRITE           1   no mask    U ; NO OTHER WRITERS
      1   WCB$V_DLOCK             1   no mask    U ; ENABLE DEACCESS LOCK
      2     spare                 2   no mask    U ; UNUSED
      4   WCB$V_SPOOL             1   no mask    U ; SPOOL FILE ON CLOSE
      5   WCB$V_WRITECK           1   no mask    U ; ENABLE WRITE CHECK
      6   WCB$V_SEQONLY           1   no mask    U ; SEQUENTIAL ONLY ACCESS
      7     spare                 1   no mask    U ; SPARE
      8   WCB$V_WRITEAC           1   no mask    U ; WRITE ACCESS
      9   WCB$V_READCK            1   no mask    U ; ENABLE READ CHECK
      A   WCB$V_NOREAD            1   no mask    U ; NO OTHER READERS
      B   WCB$V_NOTRUNC           1   no mask    U ; NO TRUNCATES
```

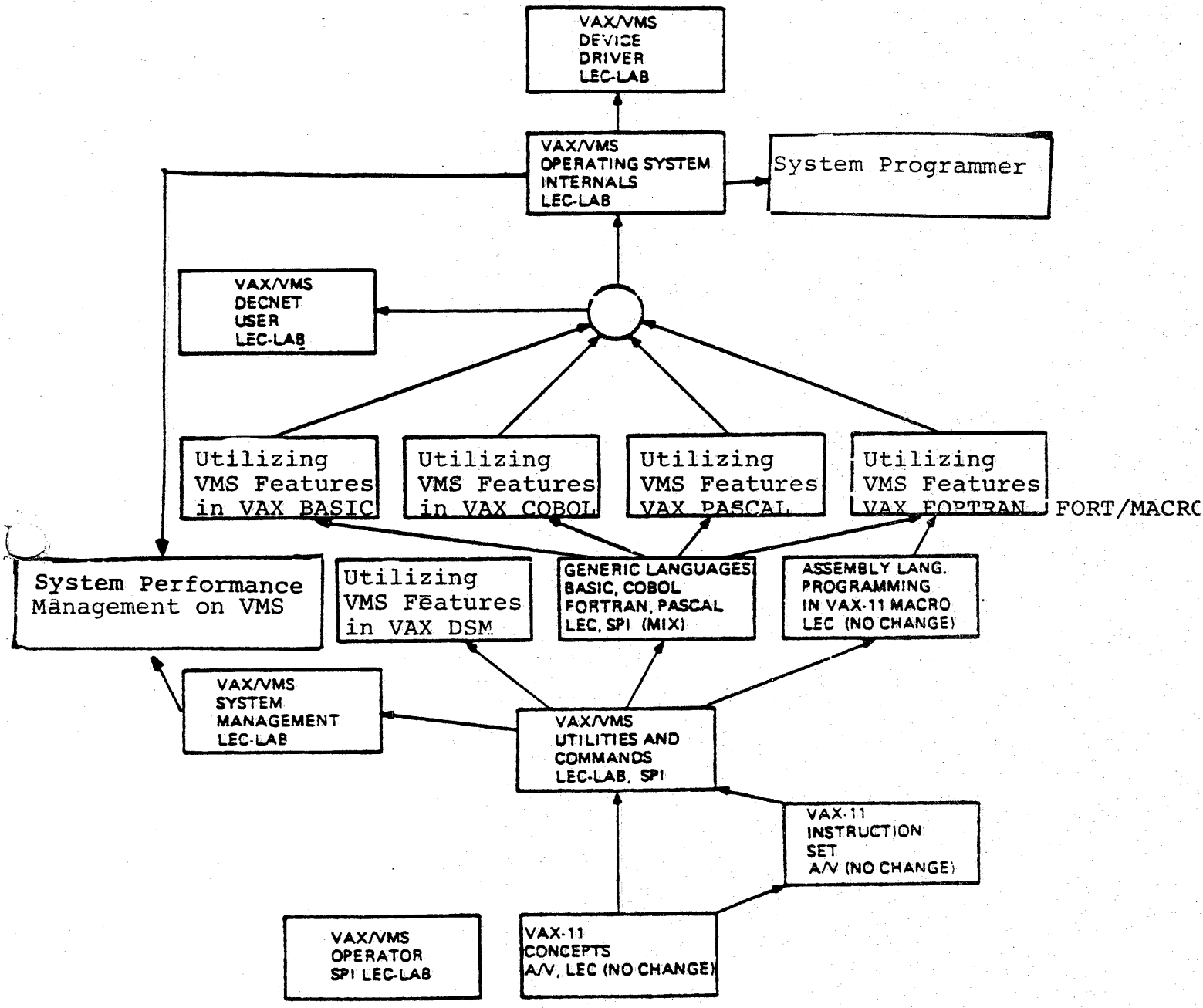
```
;+
; WORKING SET LIST DEFINITIONS
;-

        $WSLDEF

;THE FOLLOWING 5 BITS MUST BE IN ORDER
;THE PRECEDING 5 BITS MUST BE IN ORDER

;
; PAGE TYPE VIELD DEFINITIONS
;
     0  WSL$V_VALID          1          1  U ;WSL ENTRY VALID
     1  WSL$V_PAGTYP         3          E  U ;PAGE TYPE (SEE PFNDEF FOR VALUES)
     4  WSL$V_PFNLOCK        1         10  U ;PAGE FRAME LOCK
     5  WSL$V_WSLOCK         1         20  U ;WORKING SET LOCK
     6  WSL$V_GOODPAGE       1         40  U ;THIS PAGE SHOULD REMAIN IN WS ONE
     7    spare              1         80  U ;SPARE BIT
     8  WSL$V_MODIFY         1        100  U ;SAVED MODIFY BIT
```

```
;+
; WAIT QUEUE HEADER DEFINITIONS
;-

        $WQHDEF


                                +-------------+
WQH$L_WQFL              0       |             |   U ;HEAD OR FORWARD LINK
                                +-------------+
WQH$L_WQBL              4       |             |   U ;TAIL OR BACKWARD LINK
                                +------+------+
WQH$W_WQCNT             8       |      |      |   U ;WAIT QUEUE COUNT
                                +------+------+
WQH$W_WQSTATE           A       |      |          U ;STATE NUMBER FOR WAIT
                                +------+
```

CURRICULUM MAP FOR VAX/VMS

APPENDIX B

VAX/VMS CURRICULUM



Key

Lec = Lecture
Lec/Lab = Lecture and lab
SPI = Self-paced instruction
A/V = Audiovisual instruction