Internals of a VAX/VMS Process

September 1986

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Printed in U.S.A.

The manuscript for this book was prepared using Digital Standard Runoff. Book production was done by Educational Services Development and Publishing in Nashua, N.H.

The following are trademarks of Digital Equipment Corporation:

digital™
DATATRIEVE
DEC
DECmate
DECnet
DECset
DECsystem-10
DECSYSTEM-20
DECtape
DECUS
DECwriter
DIBOL
MASSBUS
PDP
P/OS
Professional
Rainbow
RSTS
RSX
UNIBUS
VAX
VMS
VT
Work Processor

INTERNALS OF A VAX/VMS PROCESS

A process is the environment within which programs execute on VMS. It is made up of a hardware context (e.g. general registers), software context (e.g. privileges, quotas), and virtual address space. This seminar provides an indepth look at the structure and function of VMS processes.

The discussion will include information on how a VMS process is implemented, scheduled and maintained. This will assist in using VMS processes more efficiently in applications.

In addition, this information will give system managers a better understanding of the resources required for processes, and how to manage those resources.

The seminar will reflect the most recent version of VMS.

Topics:

- The major internal data structures used by VMS to describe and control a process
- Tools provided by VMS for examination of processes
- Process scheduling
- Quantum end activities
- Examination of relevant VMS source code in MACRO32 (if there is sufficient class interest)

Information on image formation and activation will be included if time permits.

Prerequisites:
Experience with VMS at the DCL level. Fundamental knowledge of a process (received through programming, design or system management). Experience with data structure, definition and manipulation.

SEMINAR TOPICS

1. THE PROCESS

   - Process vs System Context
   - Process Data Structures
   - Virtual Address Space

2. PROCESS CREATION AND DELETION

   - Steps in Process Creation
   - Interactive vs Batch Jobs
   - Process Deletion

3. PROCESS SCHEDULING

   - Process States and Data Structures
   - Operating System Scheduling Code
   - Quantum End

TOPICS

I. Process vs. System Context

II. Process Data Structures Overview

A. Software context information

B. Hardware context information

III. Virtual Address Space Overview

A. SØ space (operating system code and data)

B. PØ space (user image code and data)

C. P1 space (command language interpreter, process data)

IV. SYSGEN Parameters Related to Process Characteristics

```
$ SHOW SYSTEM

VAX/VMS V4.0  on node MUSIC  30-NOV-1984 11:04:10.58   Uptime    0 13:44:50
  Pid    Process Name    State  Pri      I/O       CPU        Page flts Ph.Mem
00000080 NULL            COM      0        0   0 10:53:07.27          0      0
00000081 SWAPPER         HIB     16        0   0 00:03:17.21          0      0
00000085 ERRFMT          HIB      8      760   0 00:00:07.71         67     88
00000088 OPCOM           LEF      8      378   0 00:00:03.96       1184    135
00000089 JOB_CONTROL     LEF      8     2281   0 00:00:33.34        149    293
0000008B SYMBIONT_0001   HIB      6      448   0 00:00:31.56       2465     44
0000008C SYMBIONT_0002   HIB      6       19   0 00:00:00.66        235     45
0000018D MOZART          LEF      5      190   0 00:00:10.74       6552    150
0000008F NETACP          HIB      9     4892   0 00:01:37.36       3216   1500
00000090 EVL             HIB      5       32   0 00:00:00.82        253     44
00000091 REMACP          HIB      9       71   0 00:00:00.45         72     41
00000112 HUNT            CUR      4     3663   0 00:01:30.36      29430    200
00000213 CHOPIN          LEF      6      450   0 00:00:10.12       5111    300
00000214 BATCH_931       LEF      4      376   0 00:00:16.94       6077    260
00000115 STRAVINSKY      LEF      4     2404   0 00:00:29.46       8336    148
00000116 HAYDN           LEF      5      686   0 00:00:15.19       7480    400
00000117 COPELAND        LEF      6     4121   0 00:01:08.54      17151    154
00000118 SOUZA           LEF      6      819   0 00:00:13.84       2862    150
0000011B BEETHOVEN       LEF      6      719   0 00:00:11.59       4970    500
00000120 SALIERI         LEF      5      856   0 00:00:13.26       4855    500
000001BE MAIL_12         LEF      6      380   0 00:00:05.24       1014    215
```

PROCESS VS. SYSTEM CONTEXT

Process Context

- Software Context, including
  - Privileges
  - Quotas
  - Scheduling priority
  - IDs (user name, UIC, Process ID)
- Hardware Context, including
  - General Purpose Registers (RØ- R11, AP, FP, PC)
  - Stack pointers (4)
  - Processor Status Longword (PSL)
- Virtual Address Space
  - Program region (PØ)
  - Control region (P1)
  - System region (SØ)

System Context

- System virtual address space (SØ)
- The interrupt stack

VIRTUAL ADDRESS SPACE OVERVIEW

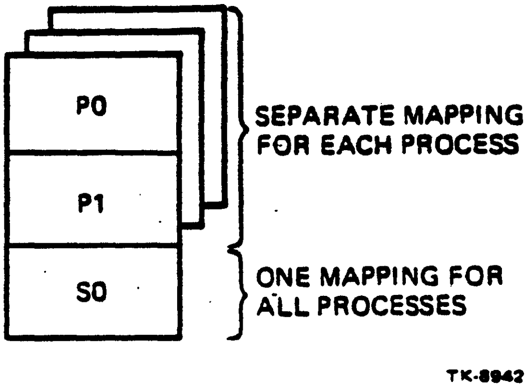

P0
P1
S0
SEPARATE MAPPING FOR EACH PROCESS
ONE MAPPING FOR ALL PROCESSES
TK-8942

Figure Virtual Address Space

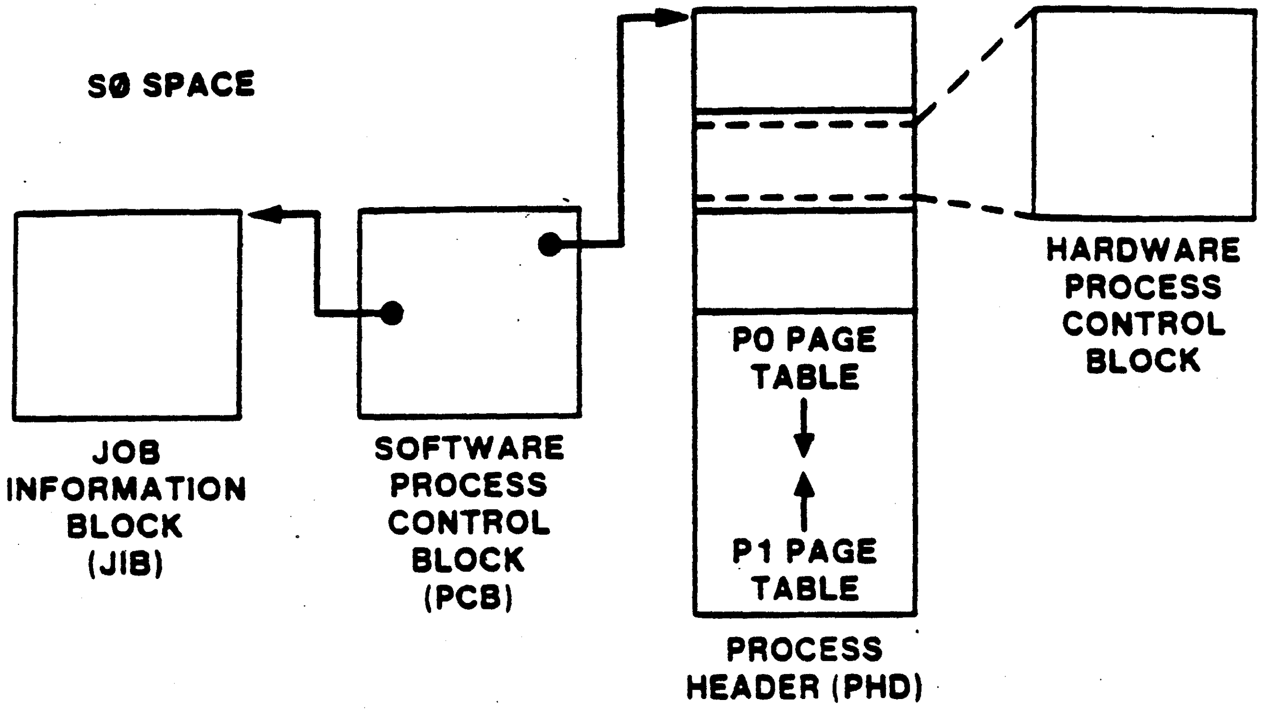
Process Virtual Address Space

P0 - Image, Run-Time Library, Debugger

P1 - Command Language Interpreter, stacks, file system XQP, I/O data areas

S0 - System services, Record Management Services, other executive code and data

PROCESS DATA STRUCTURES OVERVIEW

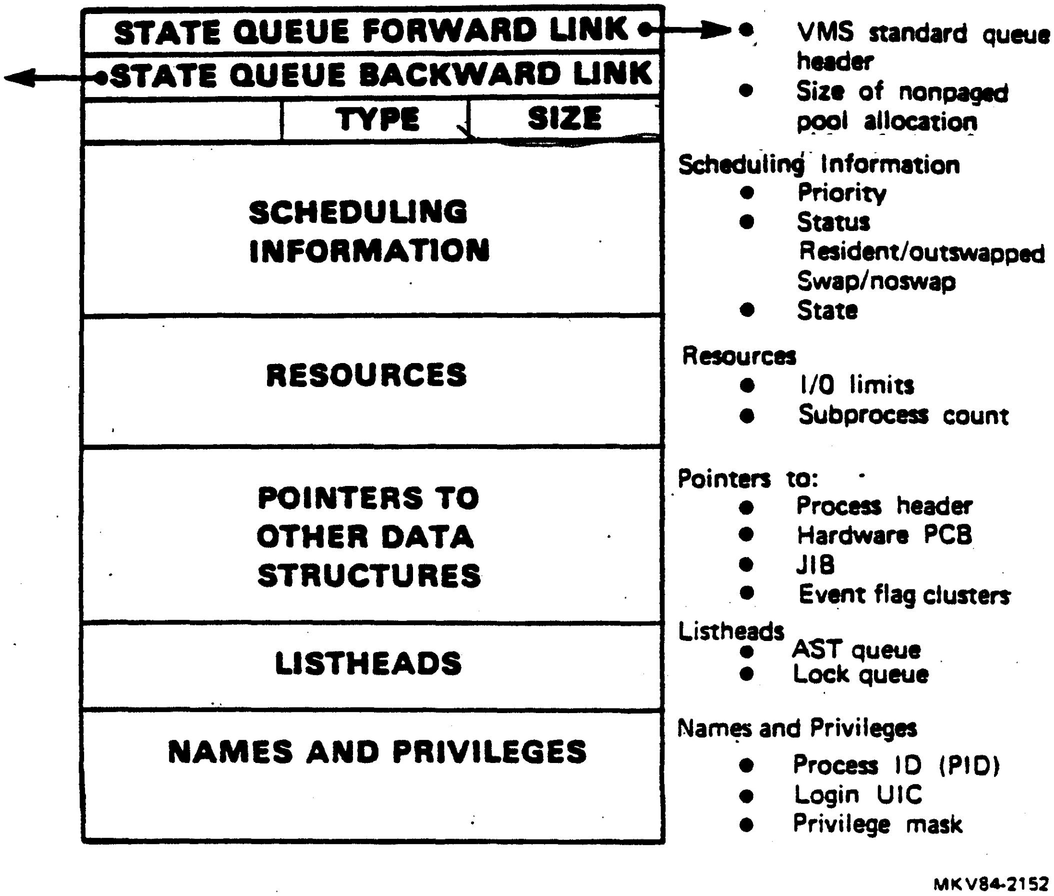


SØ SPACE
JOB INFORMATION BLOCK (JIB)
SOFTWARE PROCESS CONTROL BLOCK (PCB)
P0 PAGE TABLE
P1 PAGE TABLE
PROCESS HEADER (PHD)
HARDWARE PROCESS CONTROL BLOCK

Figure Process Data Structures

- Software Process Control Block (PCB)
  - Holds process-specific data that must always be available (for example, process state, priority).
  - Contains pointers to other process data structures
  - Not paged, not swapped
- Process Header (PHD)
  - Contains process memory management information
  - Contains hardware process control block
- Hardware Process Control Block
  - Contains saved hardware context
- Job Information Block (JIB)
  - Keeps track of resources for a detached process and all its subprocesses.

Software Process Control Block (PCB)



- VMS standard queue header
- Size of nonpaged pool allocation

Scheduling Information

- Priority
- Status
  Resident/outswapped
  Swap/noswap
- State

Resources

- I/O limits
- Subprocess count

Pointers to:

- Process header
- Hardware PCB
- JIB
- Event flag clusters

Listheads

- AST queue
- Lock queue

Names and Privileges

- Process ID (PID)
- Login UIC
- Privilege mask

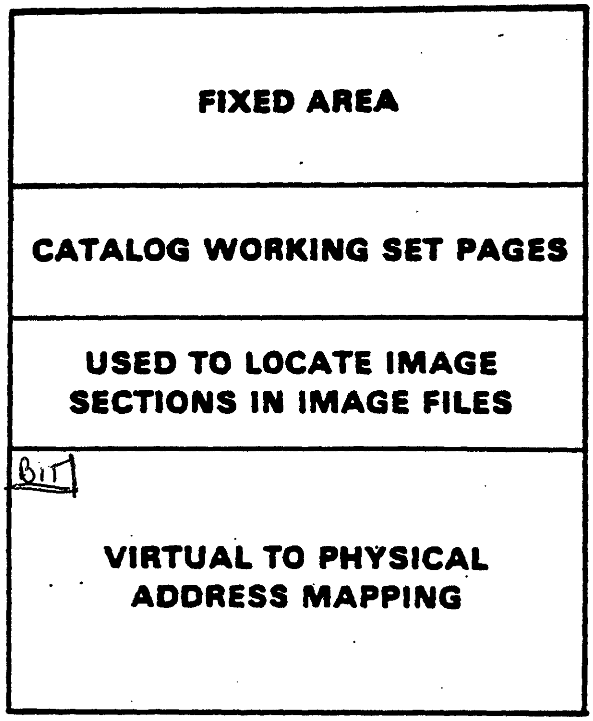
Figure / Software Process Control Block (PCB)

```
                            +---------------+
PCB$L_SQFL                  |               |     0 (    0)
                            +---------------+
PCB$L_SQBL                  |               |     4 (    4)
                            +-------+-------+
PCB$W_SIZE                          |       |     8 (    8)
                                +---+-------+
PCB$B_TYPE                      |   |             A (   10)
                            +---+---+
PCB$B_PRI                   |   |                 B (   11)
                            +---+       +---+
PCB$B_ASTACT                            |   |     C (   12)
                                    +---+---+
PCB$B_ASTEN                         |   |         D (   13)
                            +-------+---+
PCB$W_MTXCNT                |       |             E (   14)
                            +-------+-------+
PCB$L_ASTQFL                |               |    10 (   16)
                            +---------------+
PCB$L_ASTQBL                |               |    14 (   20)
                            +---------------+
PCB$L_PHYPCB                |               |    18 (   24)
                            +---------------+
PCB$L_OWNER                 |               |    1C (   28)
                            +---------------+
PCB$L_WSSWP                 |               |    20 (   32)
                            +---------------+
PCB$L_STS                   |               |    24 (   36)
                            +---------------+
PCB$L_WTIME                 |               |    28 (   40) PCB$B_PRISAV ...
                            +-------+-------+
PCB$W_STATE                         |       |    2C (   44)
                                +---+-------+
PCB$B_WEFC                      |   |            2E (   46)
                            +---+---+
PCB$B_PRIB                  |   |                2F (   47)
                            +---+   +-------+
PCB$W_APTCNT                        |       |    30 (   48)
                            +-------+-------+
PCB$W_TMBU                  |       |            32 (   50)
                            +-------+-------+
PCB$W_GPGCNT                        |       |    34 (   52)
                            +-------+-------+
PCB$W_PPGCNT                |       |            36 (   54)
                            +-------+-------+
PCB$W_ASTCNT                        |       |    38 (   56)
                            +-------+-------+
PCB$W_BIOCNT                |       |            3A (   58)
                            +-------+-------+
PCB$W_BIOLM                         |       |    3C (   60)
                            +-------+-------+
PCB$W_DIOCNT                |       |            3E (   62)
                            +-------+-------+
PCB$W_DIOLM                         |       |    40 (   64)
                            +-------+-------+
PCB$W_PRCCNT                |       |            42 (   66)
                            +-------+
                          +----------------+
PCB$T_TERMINAL            |                |   44 (   68)
                          |                |
                          +----------------+
PCB$L_PQB                 |                |   4C (   76) PCB$L_EFWM
                          +----------------+
PCB$L_EFCS                |                |   50 (   80)
                          +----------------+
PCB$L_EFCU                |                |   54 (   84)
                          +-------+--------+
PCB$W_PGFLCHAR                    |        |   58 (   88)
                              +---+--------+
PCB$B_PGFLINDEX               |   |            5A (   90)
                          +---+---+
unused                    |   |                5B (   91)
                          +---+------------+
PCB$L_SWAPSIZE            |                |   5C (   92)
                          +----------------+
PCB$L_EFC2P               |                |   58 (   88)
                          +----------------+
PCB$L_EFC3P               |                |   5C (   92)
                          +----------------+
PCB$L_PID                 |                |   60 (   96)
                          +----------------+
PCB$L_EPID                |                |   64 (  100)
                          +----------------+
PCB$L_EOWNER              |                |   68 (  104)
                          +----------------+
PCB$L_PHD                 |                |   6C (  108)
                          +----------------+
PCB$T_LNAME               |                |   70 (  112)
                          |                |
                          |                |
                          |                |
                          +----------------+
PCB$L_JIB                 |                |   80 (  128)
                          +----------------+
PCB$Q_PRIV                |                |   84 (  132)
                          |                |
                          +----------------+
PCB$L_ARB                 |                |   8C (  140)
                          +----------------+
unused                    |                |   90 (  144)
                          .                .
                          .                .
                          .                .
                          +----------------+
PCB$L_UIC                 |                |   BC (  188) PCB$W_MEM...
                          +----------------+
unused                    |                |   C0 (  192)
                          .                .
                          .                .
                          .                .
                          +----------------+
PCB$L_ACLFL               |                |   FC (  252)
                          +----------------+
PCB$L_ACLBL               |                |  100 (  256)
                          +----------------+
PCB$L_LOCKQFL             |                |  104 (  260)
                          +----------------+
PCB$L_LOCKQBL             |                |  108 (  264)
                          +----------------+
PCB$L_DLCKPRI            |                |  10C (   268)
                         +----------------+
PCB$L_IPAST              |                |  110 (   272)
                         +----------------+
PCB$L_DEFPROT            |                |  114 (   276)
                         +----------------+
PCB$L_WAITIME            |                |  118 (   280)
                         +----------------+
PCB$L_PMB                |                |  11C (   284)
                         +----------------+
```

Sysgen parameter are used to control the data structure.

THE PROCESS

Process Header (PHD) FOR CURRENT PROCESS



FIXED AREA

- Privilege mask
- Hardware process control block

CATALOG WORKING SET PAGES

- Working set list

LIST OF VIRTUAL PAGES. USED DURING SWAPING.
ALL PROCESS HAVE SAME SIZE PROCESS HI

USED TO LOCATE IMAGE SECTIONS IN IMAGE FILES

- Process section table

BIT

VIRTUAL TO PHYSICAL ADDRESS MAPPING

TRANSLATION TABLE.

- P0 page table
- P1 page table

MKV84-2153

Figure 2 Process Header (PHD)

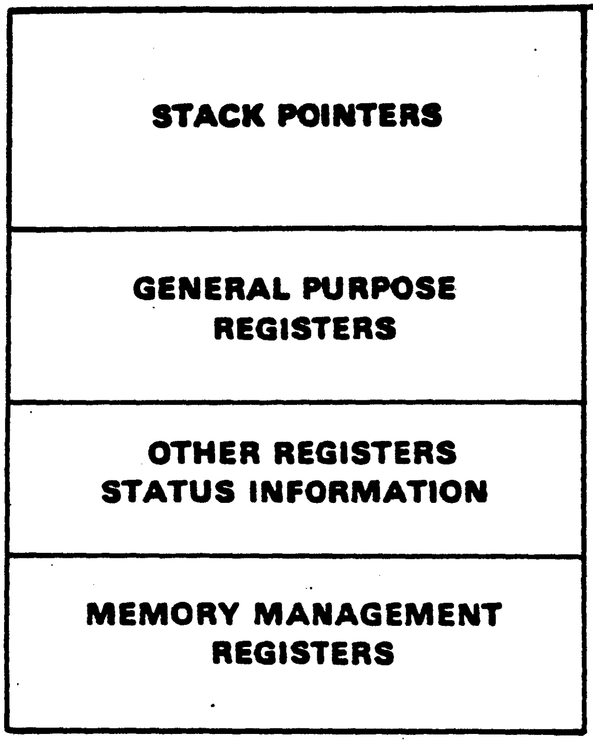
BIT = 1 MICRO CODE USES ~~TO~~ THE CONTENT OF REST OF THE LONGWORD TO GET THE PHYSICAL ADDRESS.

BIT = 0 MICRO CODE DOES NOT WANT TO DEAL WITH IT. PAGE FAULT OCCURS. MICRO CODE INVOKES THE PAGGER

THE REST OF THE WORD TO HAVE ~~AT~~ UP TO 5 DIFFERENT VALUE, where to find the Page, But if the Page is on Disk, then, the disk address will not fit here. ~~so~~ it tells it ~~you~~ to look in the PROCESS section table above it.

Hardware Process Control Block

Book Marks the Process.
Contains what was in the CPU, when Process was swapped.


STACK POINTERS
GENERAL PURPOSE REGISTERS
OTHER REGISTERS STATUS INFORMATION
MEMORY MANAGEMENT REGISTERS

PR$_PCBB

- Pointers to:
  - Kernel stack — O.S.
  - Executive stack — RMS
  - Supervisor stack — DCL.
  - User stack — IMAGE.

  Specific TO your Process. FOUND IN P1 space

  Depending on what operation is being done thats the stack you are using.
- R0, R1, ..., R11
- Argument Pointer (AP)
  Frame Pointer (FP)
  Program Counter (PC)
- Processor Status Longword (PSL)
- P0 base register
  P1 base register
  P0 length register
  P1 length register

  Memory Boundary FOR A PROCESS.
  CAUSES ACCESS VIOLATION if the ~~length is~~ boundary is crossed.

MKV84-2148

Figure 3 Hardware Process Control Block

- PR$_PCBB contains the physical address of the hardware PCB for the current process.

THIS IS one of the few times a actual Physical address is used by Process, "Context switching" "Fast" saving of your registers etc in the HPCB before being swapped.

Job Information Block

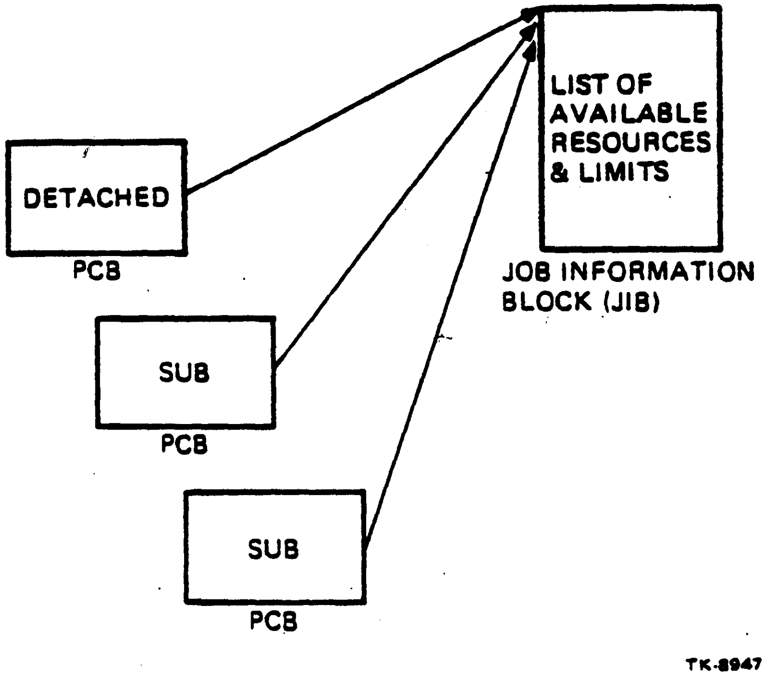
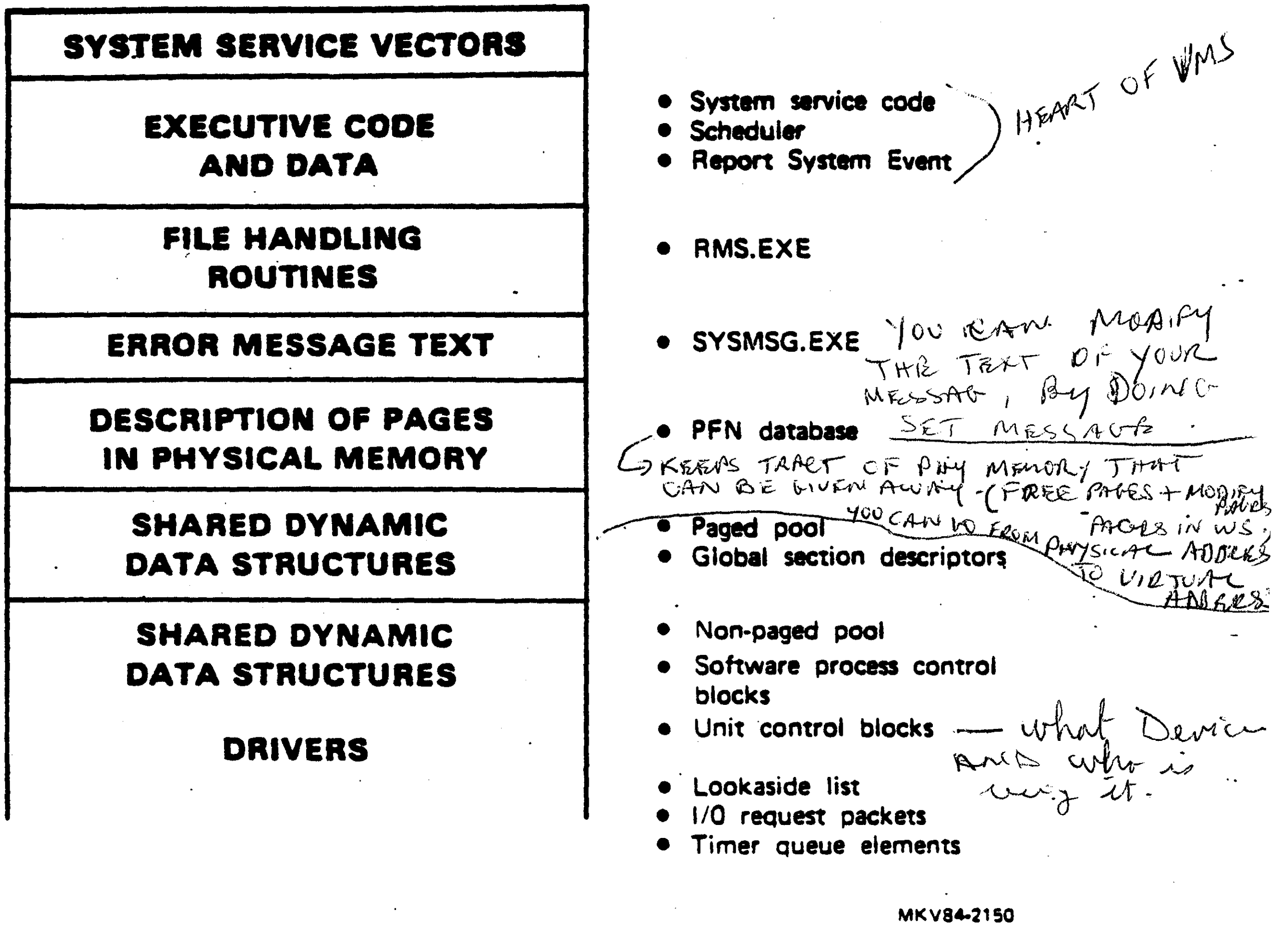


Figure 4 Job Information Block (JIB)

- Job consists of a detached process and its subprocesses.
- Job information block (JIB) keeps track of resources allotted to a job, such as:
  - Limit on number of subprocesses (PRCLIM)
  - Open File Limit (FILLM)

S0 Virtual Address Space is built when system is booting


SYSTEM SERVICE VECTORS
EXECUTIVE CODE AND DATA
FILE HANDLING ROUTINES
ERROR MESSAGE TEXT
DESCRIPTION OF PAGES IN PHYSICAL MEMORY
SHARED DYNAMIC DATA STRUCTURES
SHARED DYNAMIC DATA STRUCTURES
DRIVERS

- System service code
- Scheduler
- Report System Event

HEART OF VMS

- RMS.EXE
- SYSMSG.EXE

YOU CAN MODIFY THE TEXT OF YOUR MESSAGE, BY DOING SET MESSAGE.

- PFN database

KEEPS TRACT OF PHY MEMORY THAT CAN BE GIVEN AWAY -(FREE PAGES + MODIFY PAGES PAGES IN WS.

- Paged pool
- Global section descriptors

YOU CAN GO FROM PHYSICAL ADDRESS TO VIRTUAL ADDRES.

- Non-paged pool
- Software process control blocks
- Unit control blocks — what Device and who is using it.
- Lookaside list
- I/O request packets
- Timer queue elements

MKV84-2150

Figure 5 S0 Virtual Address Space - Low Addresses

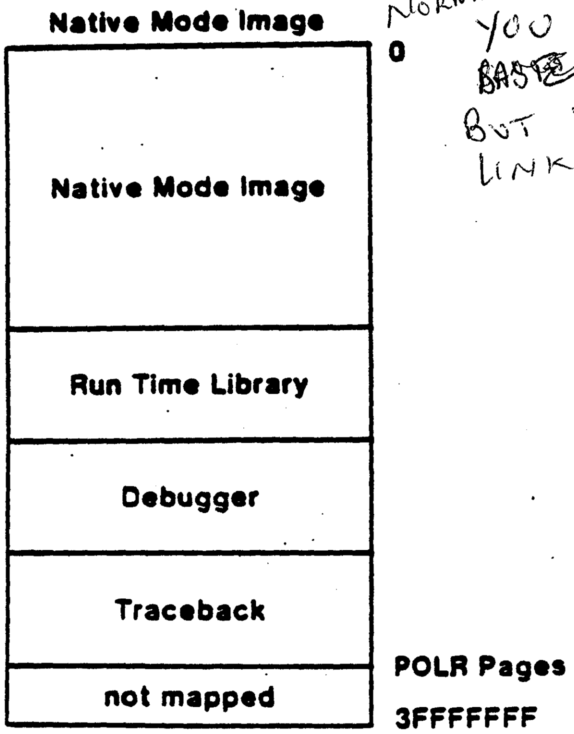
* WS - WORKING SET.

STACK USED WHEN INTERRUPTS OCCUR	● Interrupt stack
TABLE FOR VECTORING BY HARDWARE TO SERVICE ROUTINES	● System Control Block (SCB) when hardware has a problem it goes here: e.g. virtual address of the pager.
STORAGE FOR PROCESS HEADERS	● Balance slots Balance set count
LOCATIONS OF VALID SYSTEM VIRTUAL ADDRESSES DATA STRUCTURES USED TO LOCATE GLOBAL SECTIONS	● System header - System working set list - Global section table GLOBAL Section. and where they are on disk. same as the Psects. LIKE PROCESS Context
LOCATION OF EACH PAGE OF SYSTEM VIRTUAL ADDRESS SPACE	● System page table
LOCATIONS OF GLOBAL PAGES	● Global page table

MKV84-2149

Figure 6 S0 Virtual Address Space - High Addresses

P0 Virtual Address Space



NORMALLY YOU DON'T LINK WITH ~~BASE~~ ADDRESS OF 0. BUT YOU CAN SPECIFICALLY LINK IF YOU WANT.

Figure 7 PØ Virtual Address Space

P1 Virtual Address Space

VMS Sizes AND FORMATS the space but the info it contains is Process specific.

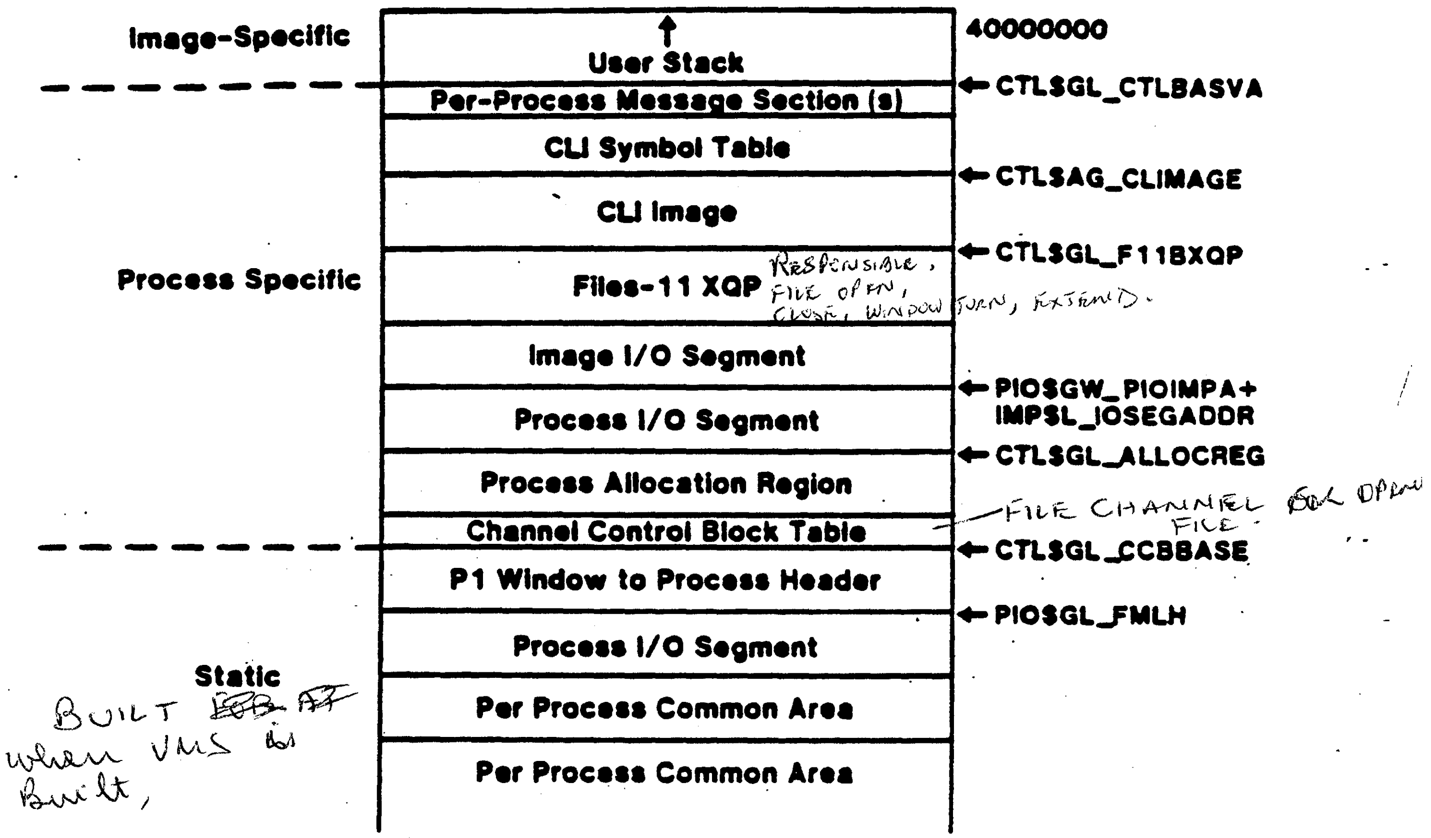
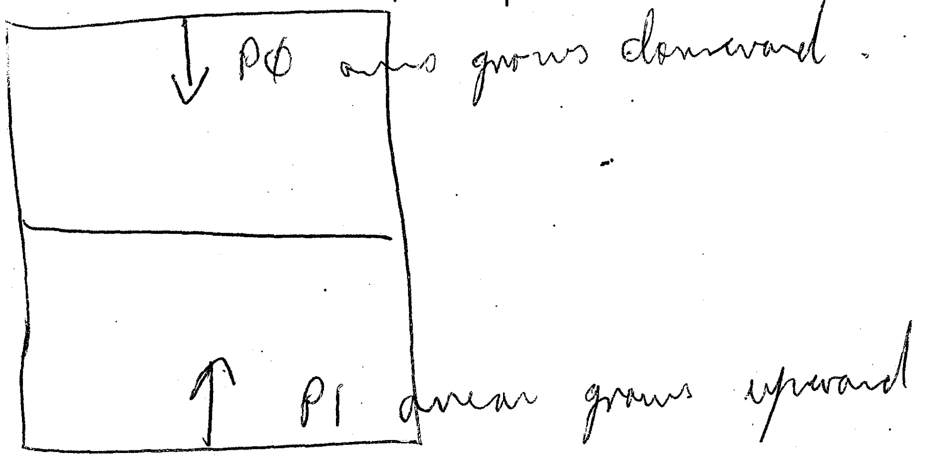


Figure 8 P1 Virtual Address Space - ~~High~~ Low Addresses

P1 space is built from high addresses toward low addresses.

WINDOW TURN — Systems has to get Pointer to the Fragmented File from disk.



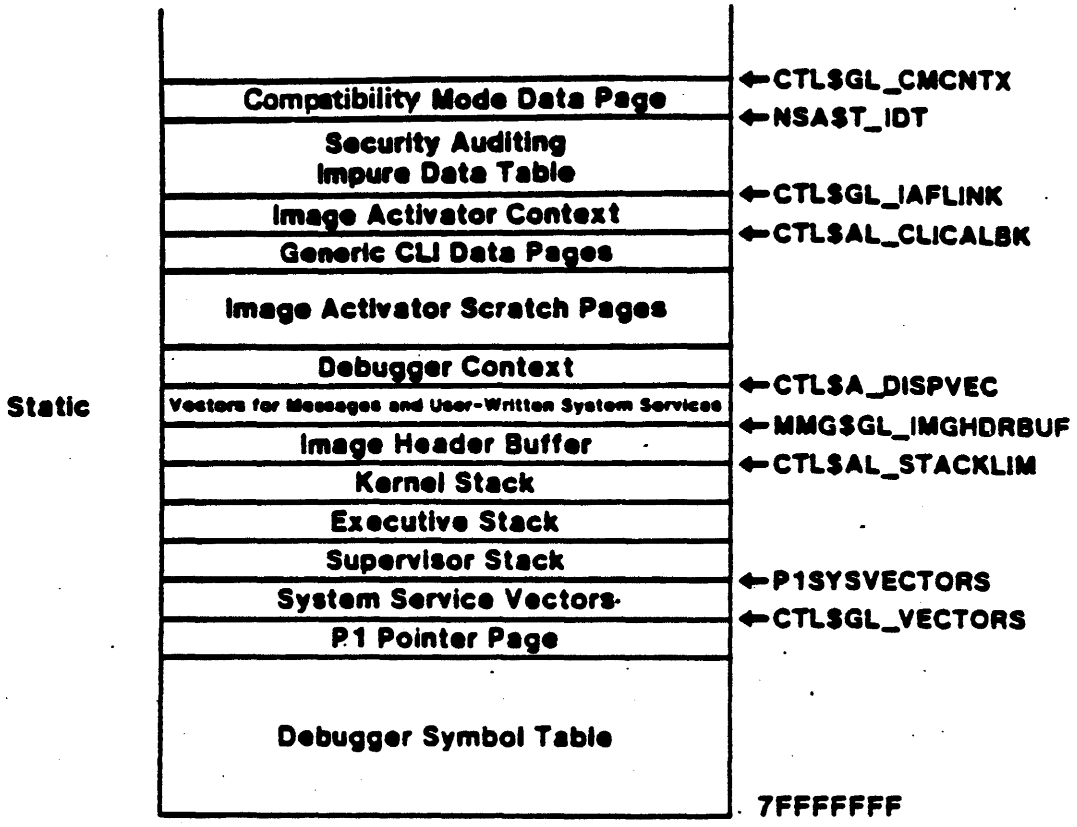


Figure 9 P1 Virtual Address Space - Low Addresses

Image-Specific - Deleted on image exit
Process-Specific - Changes according to SYSGEN parameters
and CLI used
Static - Does not change

System does not expect you to change the P1 space while system is doing work for you

Table 1 Function of P1 Space

Function	P1 Area
Images	Command Language Interpreter (DCL, MCR, user-written)
Symbol tables	Symbolic Debugger Command Language Interpreter
Pointers	System service vectors User-written system service vectors P1 window to process header (maps to PHD in S0 space) P1 pointer page (i.e., CTL$GL_CTLBASVA; addresses of exception vectors) Perprocess message vectors
Stacks	Kernel, executive, supervisor, user
RMS data	Image I/O segment Process I/O segment
File system code	Files-11 XQP
Error message text	Perprocess message section
Storage area	
● Data stays around between images	Perprocess Common Area (LIB$GET_COMMON)
● Logical names	Process allocation region
Other data areas	Generic CLI data pages Image activator scratch pages Image header buffer Compatibility mode data page (used by AME) Channel control block table (links process to device)

Table 2 SYSGEN Parameters Relevant to Process Structure

Function	Parameter
Size of the CLI symbol table	CLISYMTBL
Limit on use of process allocation region by images	CTLIMGLIM (*)
Number of pages in the process allocation region	CTLPAGES (*)
Default number of pages created by the image activator for the image I/O segment	IMGIOCNT (*)
Number of pages for the process I/O segment mapped by PROCSTRT	PIOPAGES (*)

(*) = special SYSGEN parameter

PROCESS CREATION AND DELETION

TOPICS

I. Process Creation

  A. Roles of operating system programs

  B. Creation of process data structures

II. Types of Processes

III. Initiating Jobs

  A. Interactive

  B. Batch

IV. Process Deletion

V. SYSGEN Parameters Relating to Process Creation and Deletion

TYPES OF PROCESSES

Table 3 Types of Processes

	Created By	Creating Code	Special Properties
Batch	Job Controller	SUBMIT, $SNDJBC, $CREPRC	- Deleted upon logout, or at end of command stream - No password check
Detached	Another process	RUN, $CREPRC	- Survives deletion of its creator - May be interactive or not
Network	Network ACP (result of DCL command with node name)	$CREPRC	- Deleted when no more logical links to service
Subprocess	Another process (the owner)	RUN, SPAWN, LIB$SPAWN, $CREPRC	- Cannot survive deletion of owner - Quotas are pooled with owner - May be interactive or not

- RUN and SPAWN call $CREPRC
- After system initialization
  - A process is created by another process
  - Process creation is done by $CREPRC
- An interactive process has:
  - PCB$V_INTER bit set in PCB$L_STS field
  - Non-file-oriented SYS$INPUT

Table 4 PCB Fields Defining Process Types

	PCB$V_BATCH	PCB$V_NETWRK	PCB$V_INTER	PCB$L_OWNER
Network	0	1	0	0
Batch	1	0	0	0
Detached	0	0	0 or 1	0
Subprocess	0	0	0 or 1	non-zero

- PCB$V_xxx symbols represent bits in PCB$L_STS longword
- These bits in the status longword
  - Are intended ONLY for use by the system (for example, the job controller or SPAWN)
  - Can be set using STSFLG argument to $CREPRC
- Interactive processes have the PCB$V_INTER bit set

Table 5 Restrictions on Process Creation

Quota/Limit	Meaning
MAXJOBS	Maximum number of interactive, detached, and batch processes a user may create
MAXDETACH	Maximum number of detached processes a process may create
PRCLM	Limit on number of subprocesses a process may create
Privilege	**Required for**
DETACH or CMKRNL	Creation of a detached process with a different UIC than the creator

$CREPRC

The Create Process service creates a subprocess or detached process on behalf of the calling process.

Format:

```
SYS$CREPRC  [pidadr] ,image [,input] [,output] [,error]
           [,prvadr] [,quota] [,prcnam] [,baspri] [,uic]
           [,mbxunt] [,stsflg]
```

Arguments:

pidadr

Process identification (PID) of the newly created process. The pidadr argument is the address of a longword into which $CREPRC writes the PID.

image

Name of the image to be activated in the newly created process. The image argument is the address of a character string descriptor pointing to the file specification of the image.

input

Equivalence name to be associated with the logical name SYS$INPUT in the logical name table of the created process. The input argument is the address of a character string descriptor pointing to the equivalence name string.

output

Equivalence name to be associated with the logical name SYS$OUTPUT in the logical name table of the created process. The output argument is the address of a character string descriptor pointing to the equivalence name string.

error

Equivalence name to be associated with the logical name SYS$ERROR in the logical name table of the created process. The error argument is the address of a character string descriptor pointing to the equivalence name string.

prvadr

Privileges to be given to the created process. The prvadr argument is the address of a quadword bit vector wherein each bit corresponds to a privilege; setting a bit gives the privilege.

quota

Process quotas to be established for the created process; these quotas limit the created process's use of system resources. The quota argument is the address of a list of quota descriptors, where each quota descriptor consists of a 1-byte quota name followed by a longword that specifies the desired value for that quota. The list of quota descriptors is terminated by the symbolic name PQL$_LISTEND.

prcnam

Process name to be assigned to the created process. The prcnam is the address of a character string descriptor pointing to a 1- to 15-character process name string.

baspri

Base priority to be assigned to the created process. The baspri argument is a longword value in the range 0 to 31, where 31 is the highest possible priority and 0 is the lowest. Normal priorities are in the range 0 through 15, and real-time priorities are in the range 16 through 31.

uic

User identification code (UIC) to be assigned to the created process. The uic argument is a longword value containing the UIC.

mbxunt

Unit number of a mailbox to receive a termination message when the created process is deleted. The mbxunt argument is a longword containing this number.

stsflg

Options selected for the created process. The stsflg argument is a longword bit vector wherein a bit corresponds to an option. Only bits 0 to 10 are used; bits 11 to 31 are reserved and must be 0.

PROCESS CREATION

Table 6 Steps in Process Creation and Deletion

Action	Code
Creating process	SYS$CREPRC
Inswap a process	SWAPPER
Process startup	PROCSTRT
Process deletion	SYS$DELPRC

Table 7 Three Contexts Used in Process Creation

Creator's Context	Swapper's Context	New Process's Context
$CREPRC	From SHELL	PC= EXE$PROCSTRT
• PCB	PHD filled in	PSL= K mode, IPL=2
• JIB	COMO --> COM	Sets up:
• PQB (temp)		- logical names (sys$input...) - Catch-all cond. handler
SW priority boost		- RMS dispatcher - XQP merged in - Image name moved to PHD
Process returned COMO		- Image activated

Creation of PCB, JIB, and PQB

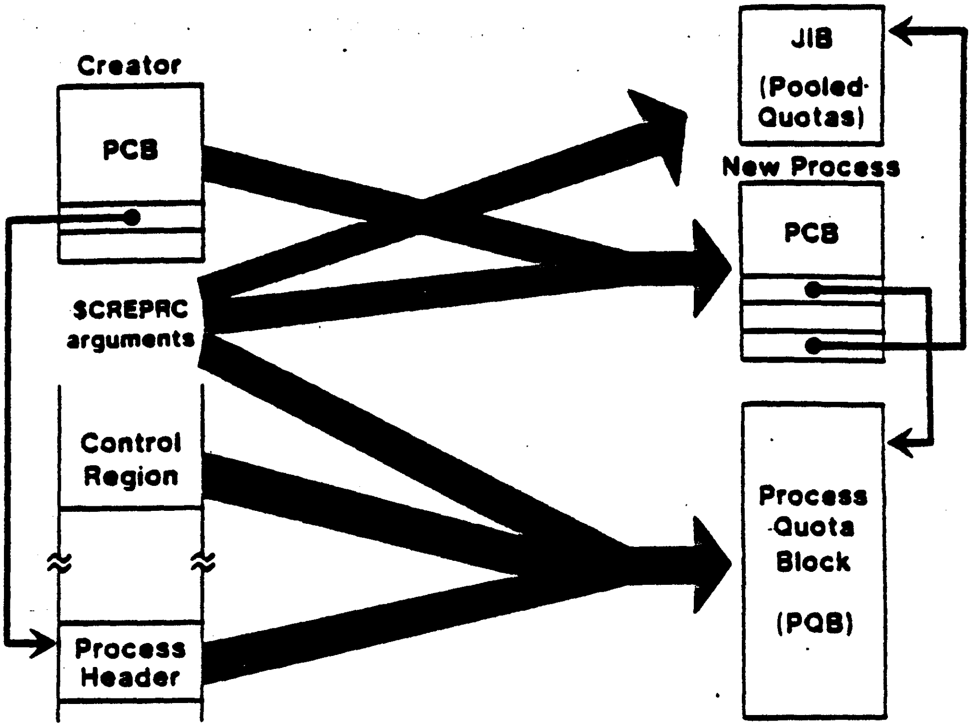


Figure 10 Creation of PCB, JIB and PQB

1. $CREPRC allocates new data structures

   - PCB
   - JIB (if new process is detached)
   - PQB (temporary)

2. These new data structures are filled from:

   - $CREPRC arguments
   - Creator's PCB
   - Creator's control region
   - Creator's process header
   - System defaults

*SYSGEN -

PQL_xxxx parameters

Relationships Between PCBs and JIB

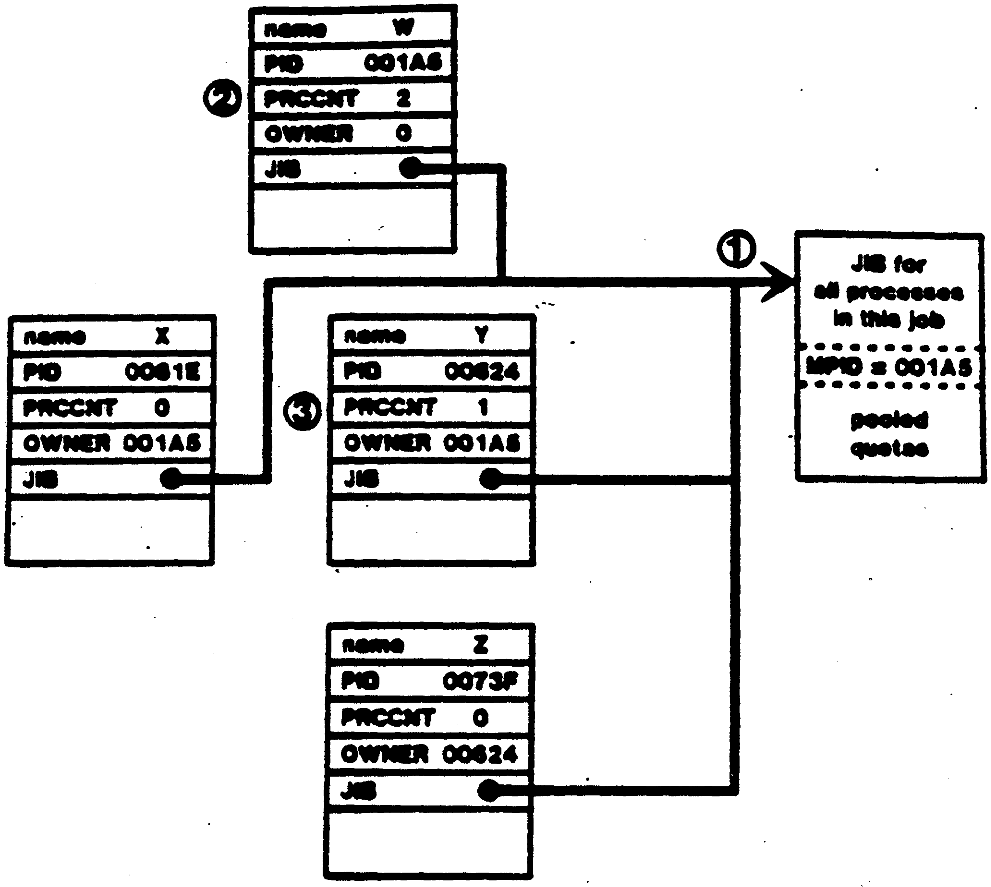


Figure 11 Relationships Between PCBs and JIB

1. All PCBs point to JIB

   W created X and Y

2. W's PRCCNT is 2

3. X and Y owner PID is W PID

   Y created Z

   No pointers from creator to subprocess

PCB Vector

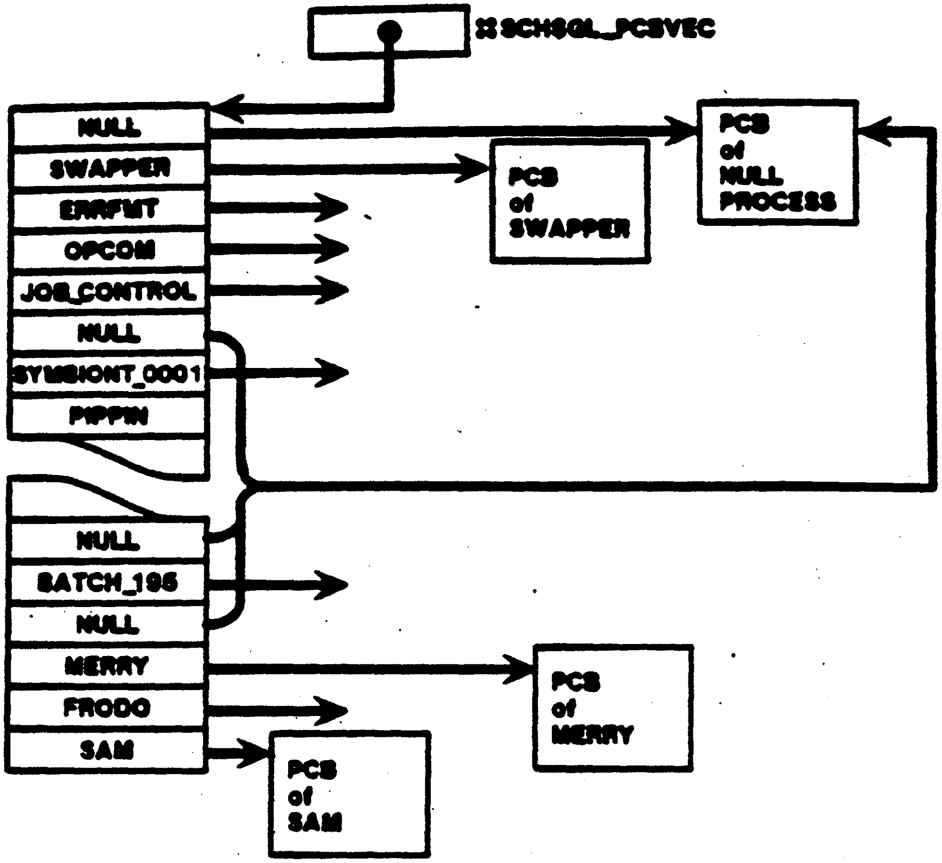


Figure 12 PCB Vector

- On process creation, search for unused vector
- Unused vectors point to Null's PCB
- Table of pointers to all PCBs
- Index into table is contained in PID
- SCH$GL_PCBVEC points to start of table

*SYSGEN -

MAXPROCESSCNT

- PID and PCB, Sequence Vectors

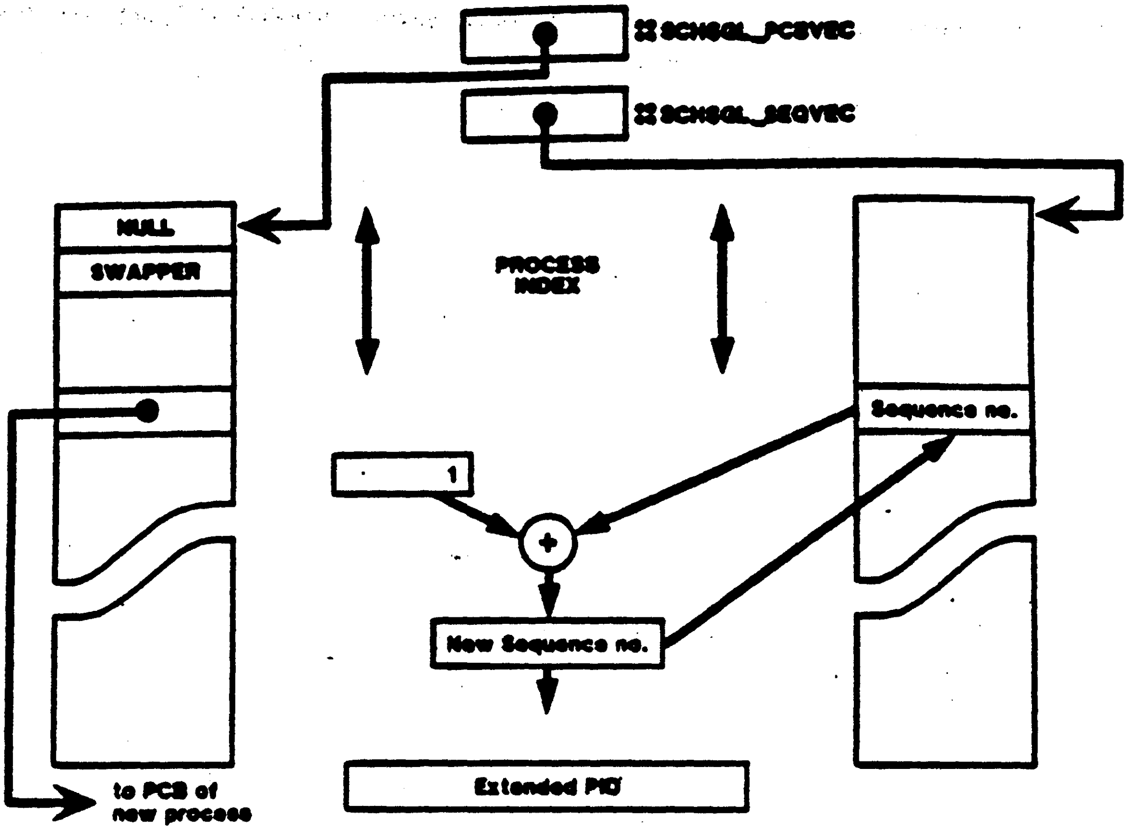


Figure 13 PID and PCB, Sequence Vectors

- Extended PID contains four parts:
  - Process index into PCB and sequence vectors
  - Process sequence number
  - Cluster node index
  - Node sequence number
- PID formed at process creation
- Sequence number incremented each time vector slot re-used
- SCH$GL_SEQVEC points to start of sequence vector

Process IDs

- There are actually two PIDs for a process
- Extended PID
  - Visible at the user level
  - Uniquely identifies a process on a single system, and on a VAXcluster
  - Displayed by VMS utilities and system services
  - Stored in PCB at offset PCB$L_EPID
  - Format is <u>very</u> subject to change
- Internal PID
  - Only visible through SDA, and in VMS source code
  - Stored in PCB at offset PCB$L_PID
  - Only contains process index and sequence number (original pre-v4 PID)
  - Used by most kernel-mode code
  - Some privileged data structures contain internal PIDs (for example TQEL_PID, ACBL_PID, and LKB$L_PID)
- Several routines available for manipulating PIDs

Table 8 Routines for Manipulating PIDs

Operation	Mechanism
Convert an extended PID to an internal PID	EXE$EPID_TO_IPID
Convert an internal PID to an extended PID	EXE$IPID_TO_EPID
Return the PCB address given an extended PID	EXE$EPID_TO_PCB
Return the PCB address given an internal PID	EXE$IPID_TO_PCB

Swapper's Role in Process Creation

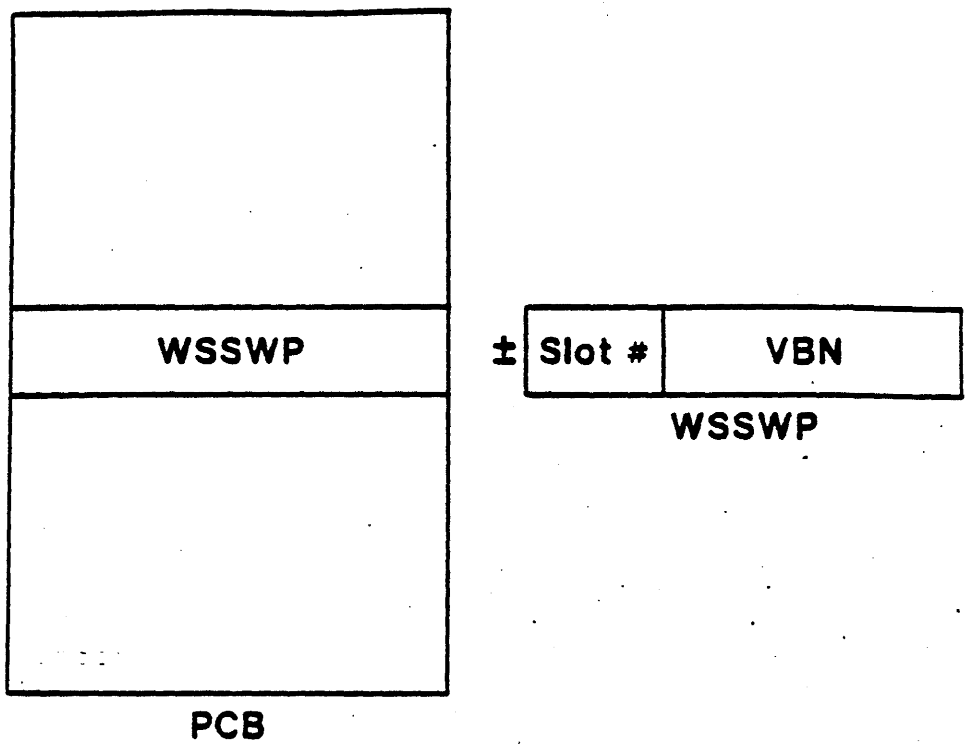


Figure 14 Swapper's Role in Process Creation

- For new process, WSSWP is less than or equal to zero
- WSSWP less than or equal to zero causes SHELL to be copied
- Swapper
  - Stores SYSGEN parameters in PHD
  - Initializes pointers, counters in PHD
  - Initializes system page table entries

PROCSTRT's Role in Process Creation

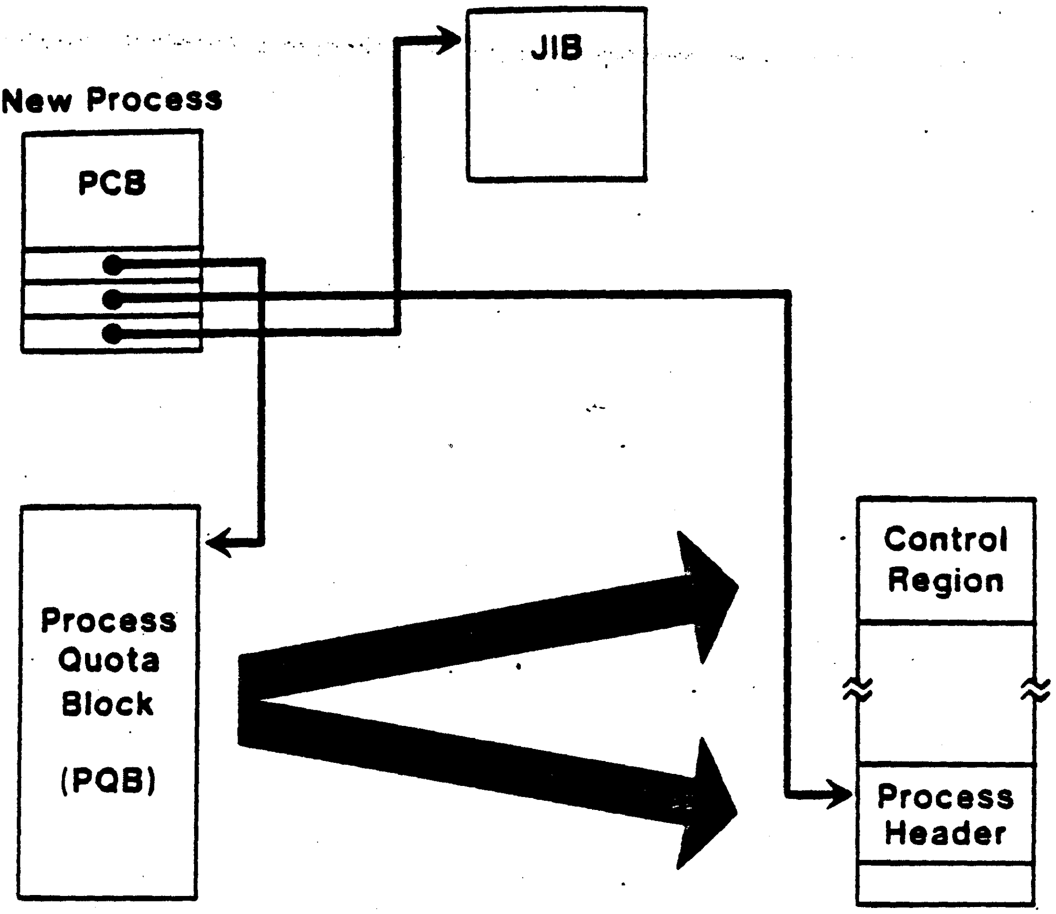
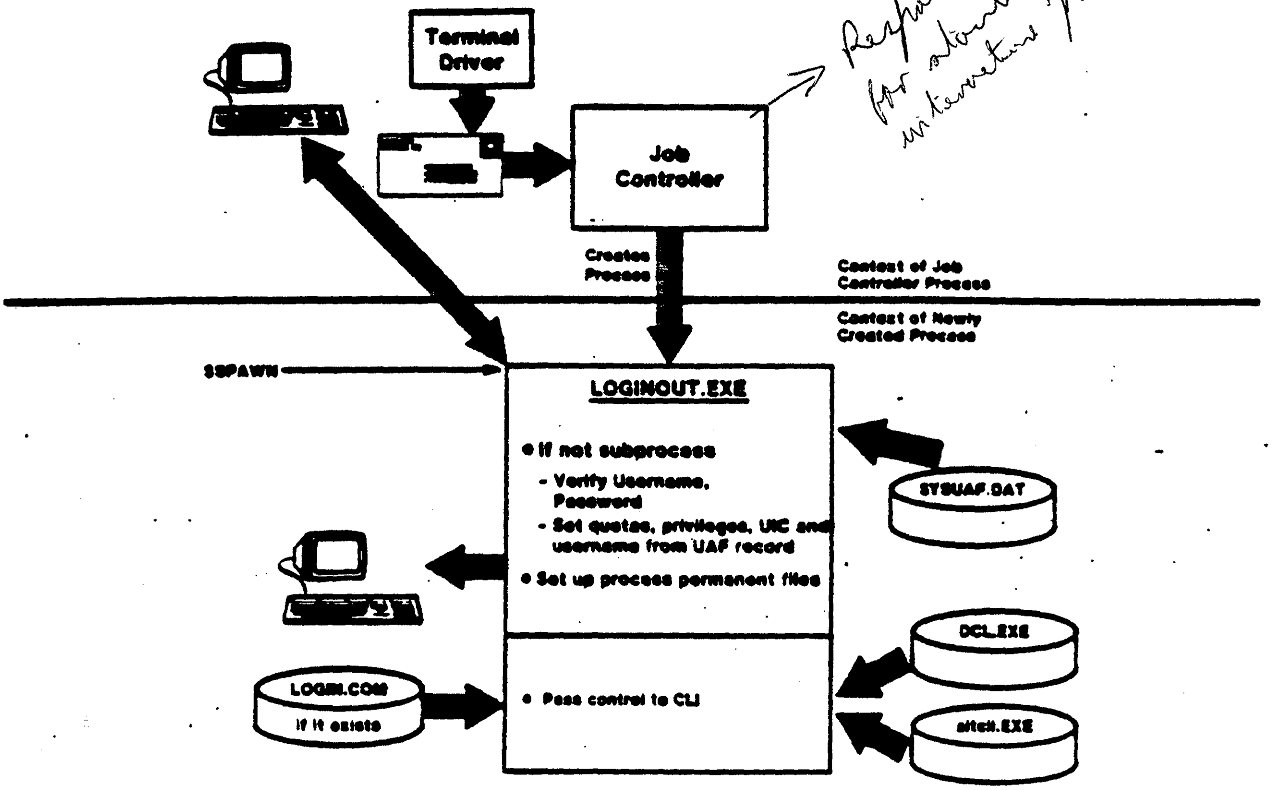


Figure 15 PROCSTRT's Role in Process Creation

- Hardware PCB defined in SHELL
- PC and IPL invoke PROCSTRT at IPL 2
- Code located in SYS.EXE
- Functions
  - PQB information moved to PHD and P1
  - Create logical name tables
  - Change to user mode, IPL 0
  - Map in F11BXQP
  - Call SYS$IMGACT
  - Call image at transfer vector

INITIATING JOBS

Initiating an Interactive Job


Terminal Driver
Job Controller
Creates Process
Context of Job Controller Process
Context of Newly Created Process
$SPAWN
LOGINOUT.EXE
• If not subprocess
- Verify Username, Password
- Set quotas, privileges, UIC and username from UAF record
• Set up process permanent files
• Pass control to CLI
SYSUAF.DAT
DCL.EXE
altcli.EXE
LOGIN.COM
if it exists

Responsible for starting all interactive processes

Figure 16 Initiating an Interactive Job

- Initiated by unsolicited input at a free terminal
    - Job controller notified by driver
    - Creates process with user name equal to terminal name
- LOGINOUT runs
- DCL mapped (or alternate CLI)
- SPAWN creates an interactive or non-interactive subprocess (no need to verify user name, etc.)

Initiating Job Using $SUBMIT

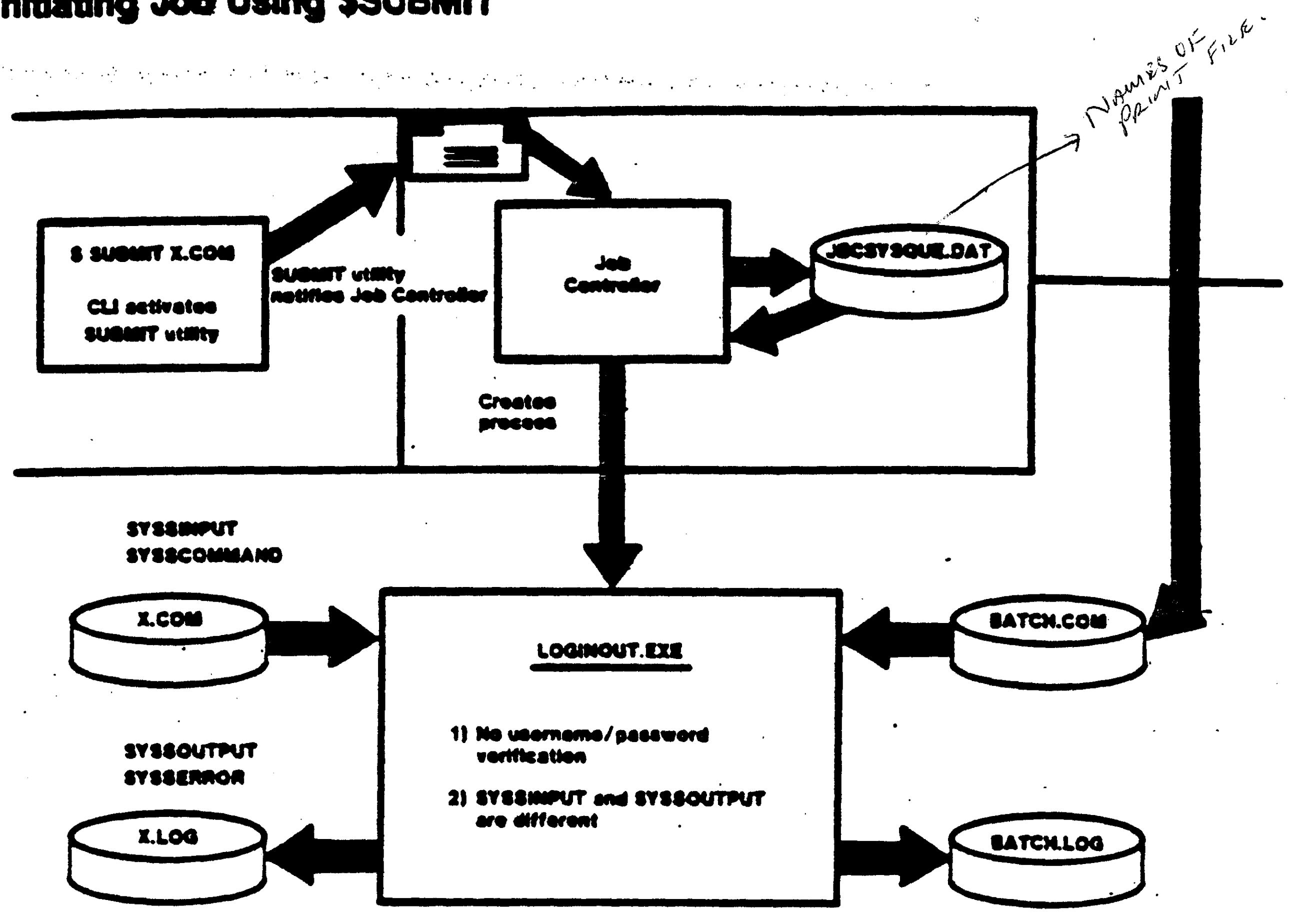


Figure Initiating Job Using $SUBMIT

o Similar to interactive process, except

  - Job controller notified by DCL ($SUBMIT)
  - User already validated
  - Files are assigned:

    SYS$INPUT to batch stream
    SYS$OUTPUT to log file

Initiating Job Through Card Reader

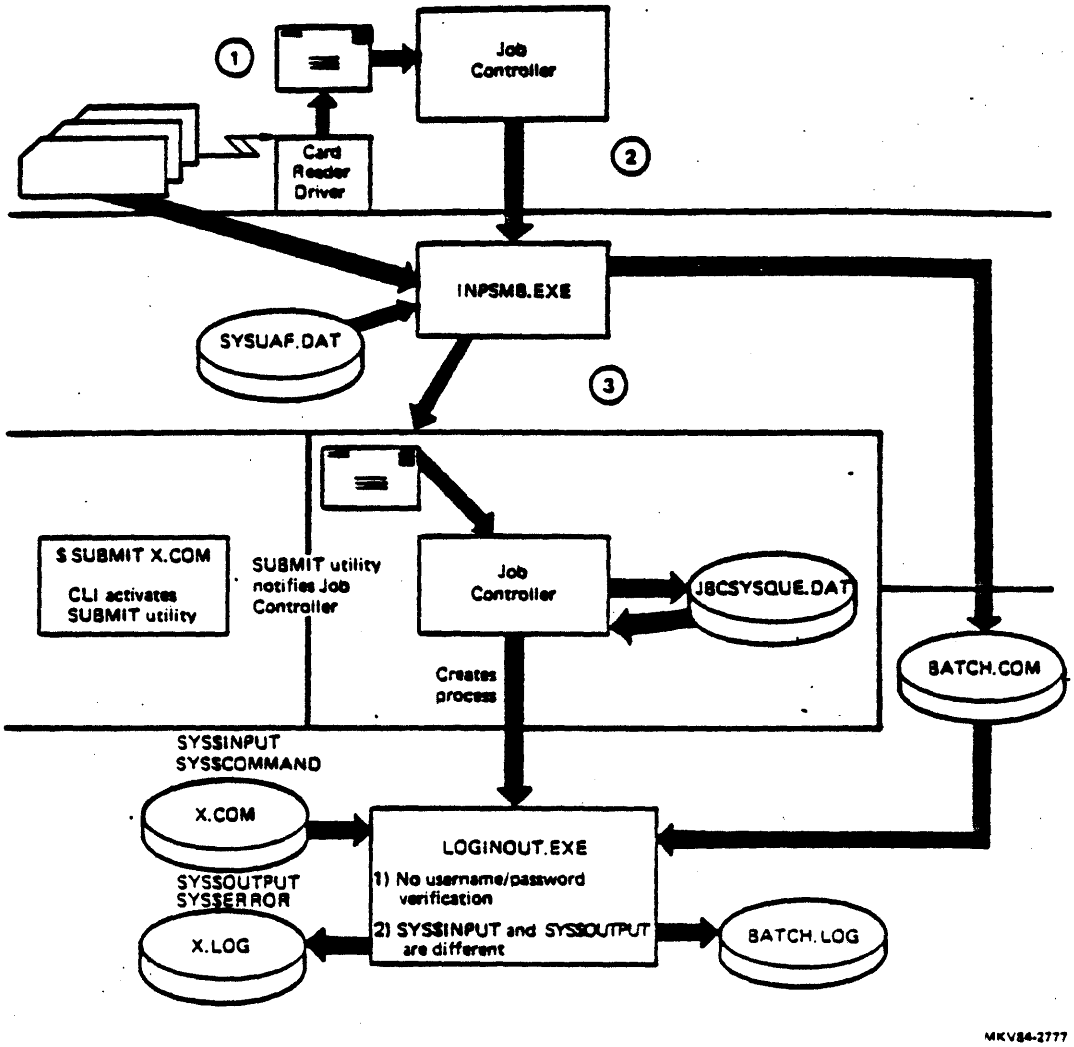


Figure Initiating Job Through Card Reader

1. Job controller notified by card reader driver

2. Job controller creates input symbiont process

   - User authorization
   - Read cards into command file
   - Submit as batch job

3. Same as for $SUBMIT

PROCESS DELETION

- After image runs and exits, process deleted
  - Unless running with a CLI
- All traces of process removed from system
- All system resources returned
- Accounting information passed to job controller
- For subprocess, all quotas and limits returned to creator
- Creator notified of deletion

Process Deletion Sequence

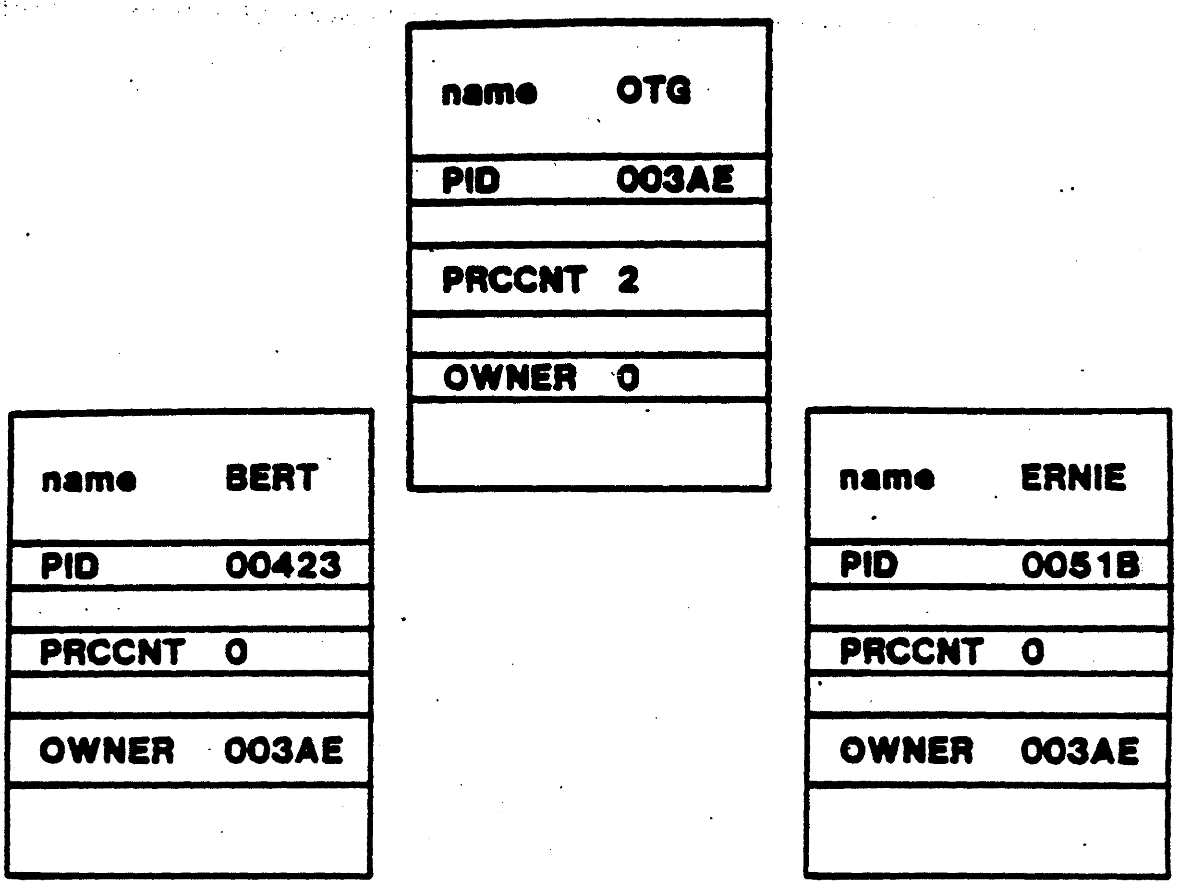


Figure Process Deletion

- Deleted by kernel AST while CURRENT
- Sequence
  - Delete any subprocesses
  - Accounting information to job controller
  - Call SYS$RUNDOWN
  - Delete P1 space
  - Free PCBVEC and SWAP slots, page file space
  - Decrement counts

    Balance set
    Total processes

  - Jump to SCH$SCHED

PROCESS CREATION AND DELETION

SUMMARY

Table Steps in Process Creation and Deletion

Action	Code
Creating process	SYS$CREPRC — Things been done in the context of another process.
Inswap a process	SWAPPER
Process startup	PROCSTRT
Process deletion	SYS$DELPRC

Table SYSGEN Parameters Relating to Process Creation and Deletion

Function	Parameter
Maximum number of processes allowed on the system	MAXPROCESSCNT
System default values for some process limits and quotas	PQL_Dxxx
System minimum values for some process limits and quotas	PQL_Mxxx

SCHEDULING

TOPICS

I. Process States

A. What they are (current, computable, wait)

B. How they are defined

C. How they are related

II. How Process States are Implemented in Data Structures

A. Queues

B. Process data structures

III. The Scheduler (SCHED.MAR)

IV. Boosting Software Priority of Normal Processes

V. Operating System Code that Implements Process State Changes

A. Context switch (SCHED.MAR)

B. Result of system event (RSE.MAR)

VI. Steps at Quantum End

A. Automatic working set adjustment

VII. Software Priority Levels of System Processes

- THE PROCESS STATES

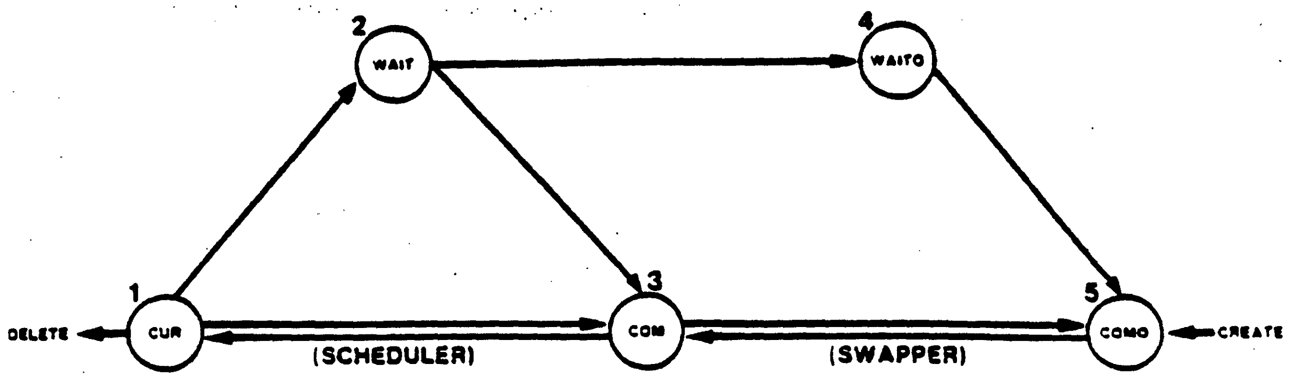


Figure Process States

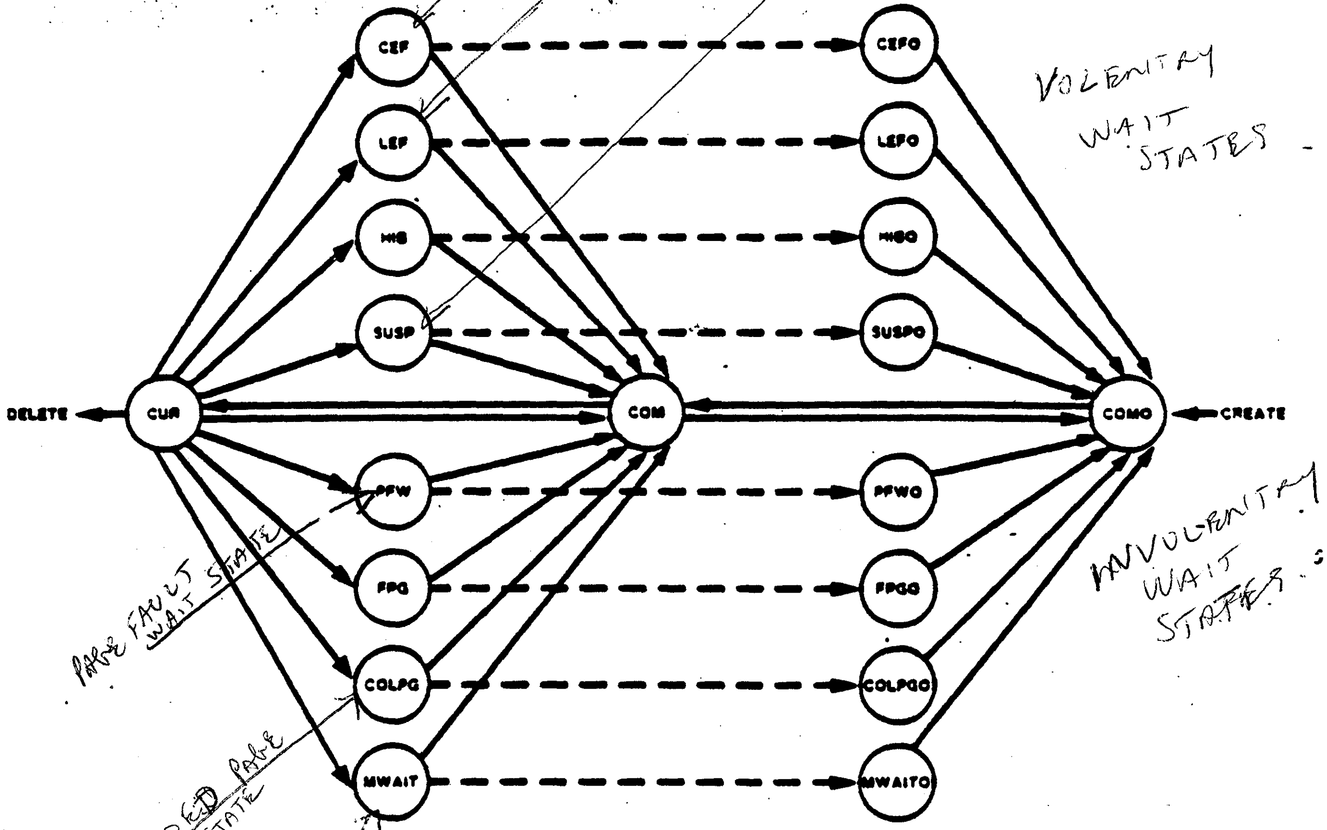
1. CURRENT - executing
2. WAIT - removed from execution to wait for event completion
3. COMPUTABLE - ready to execute
4. WAIT OUTSWAPPED
5. COMPUTABLE OUTSWAPPED

Process Wait States

COMMON EVENT FLAG: WATING ON A EVENT ~~FLAG~~ FROM ANOTHER PROCESS FROM ~~ANOTHER~~ SAME GROUP.

WAIT REQUESTED LOCALLY

CANNOT RECEIVE ASTS.



VOLENTARY WAIT STATES

INVOLENTARY WAIT ~~STATES~~ STATES.

PAGE FAULT WAIT STATE

COLLIDED PAGE WAIT STATE

Figure   Process Wait States

(MYSTRY) WAIT STATE
MUTEX WAIT STATE.
(SEMAPHORE) RW~~AIT~~

HOW PROCESS STATES ARE IMPLEMENTED

Queues

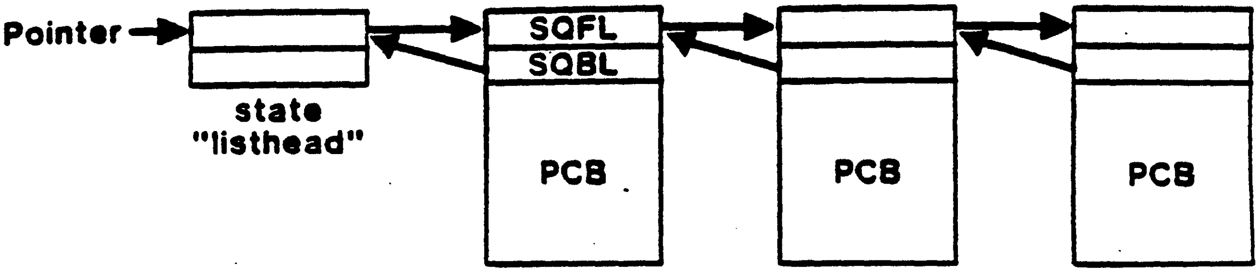


Figure A State Implemented by a Queue

- The state of a process is defined by:
  - The value in the PCB$W_STATE field
  - The PCB being in the corresponding state queue
- State queues are circular
- The current state is not implemented as a queue
  - Just a longword pointer (SCH$GL_CURPCB)
  - Queue structure not necessary because only one process in the current state
- VAX instructions for manipulating queues:
  - INSQUE new_entry, predecessor
  - REMQUE out_entry, return_address

Implementation of COM and COMO States

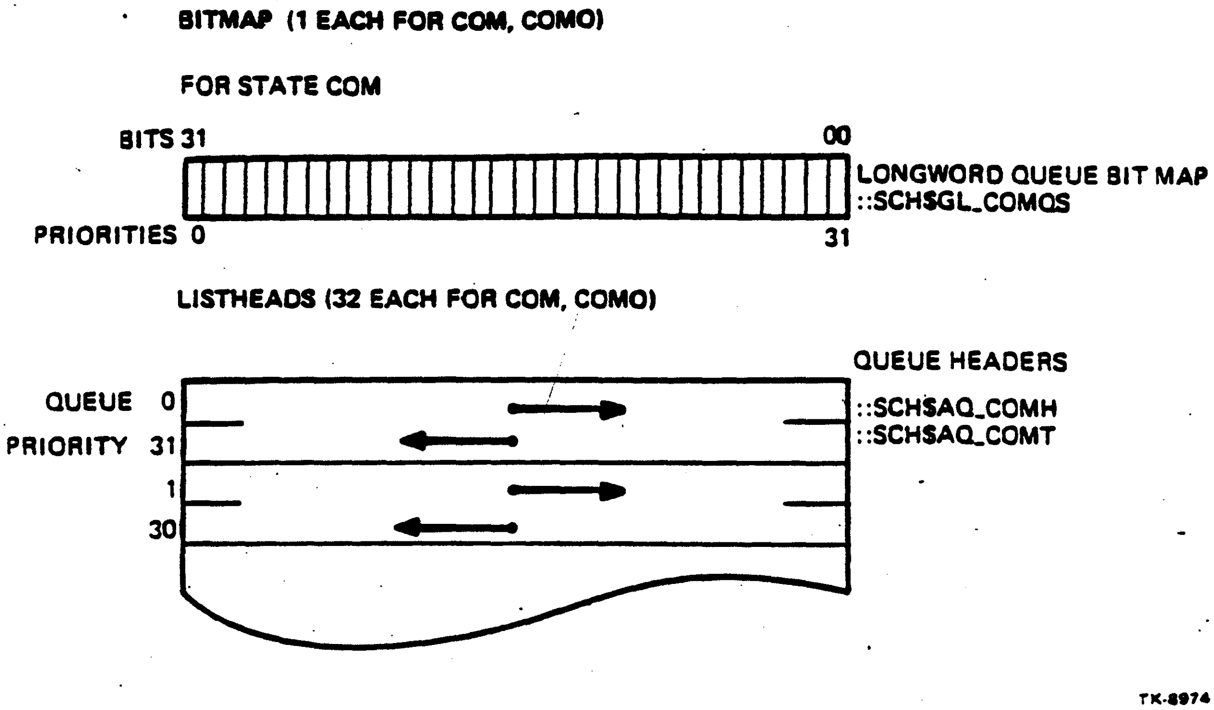


Figure Implementation of COM and COMO States

- COM state implemented as a collection of queues
- Designed to speed scheduler's search for highest-priority computable process
  - A queue for each software priority
  - Summary longword records nonempty COM queues
  - Internally, software priority stored as inverted value (as 31 minus priority)
- COMO state is implemented like COM state
  - 32 more queues
  - Another summary longword

Example of Computable Queues

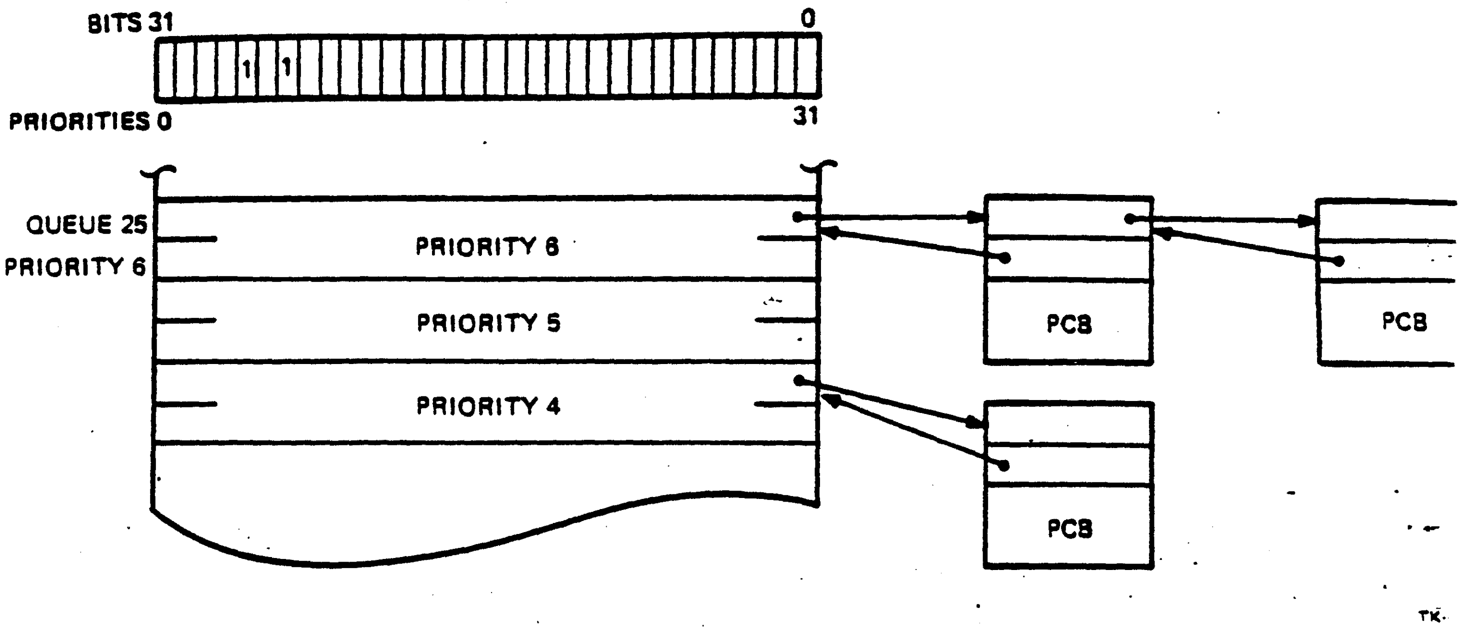


Figure Example of Computable Queues

- COM processes at priorities 4 and 6
  - Bit 25 in summary longword is set
  - Queue for priority 6 has entries
  - Bit 27 in summary longword is set
  - Queue for priority 4 has an entry

Implementation of Wait States

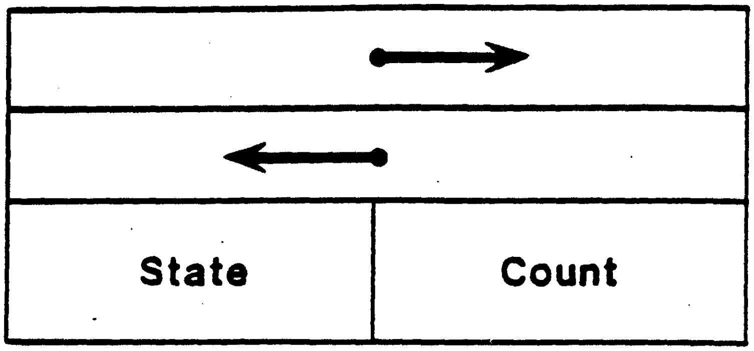


Figure Wait State Listhead

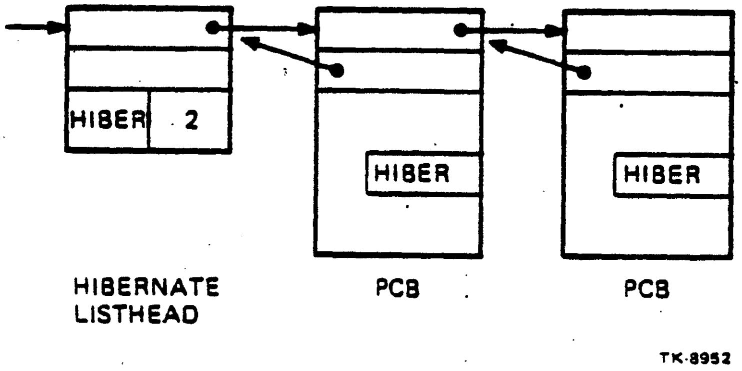


Figure Implementation of Wait States

Implementation of CEF State

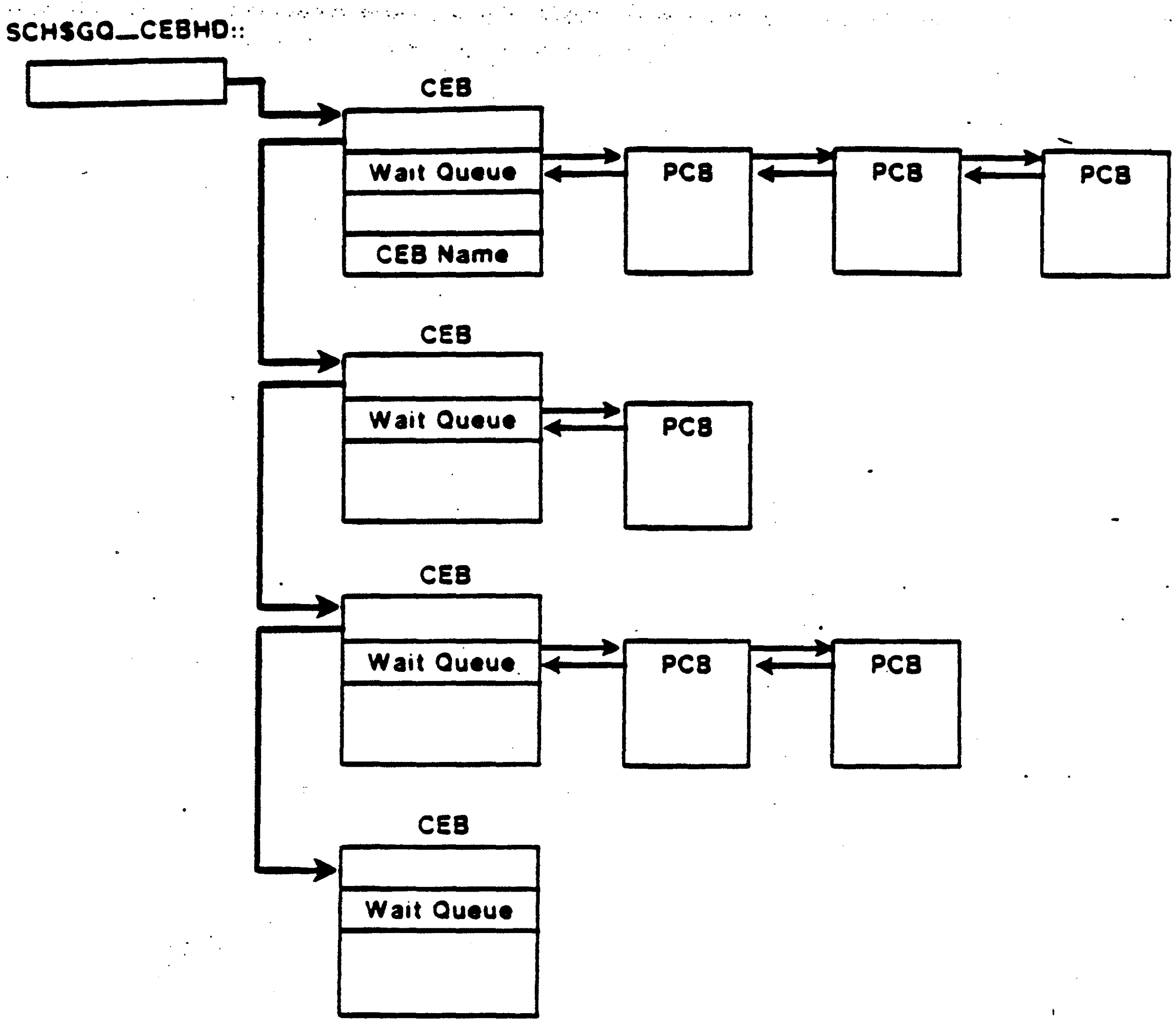


Figure Implementation of CEF State

- CEB created when event flag cluster created
- CEB contains the cluster, CEF state queue listhead, and other information about the cluster
- One CEF state queue for each CEF cluster

Summary of Scheduling States

- Current
  - Implemented with one longword pointer
  - Contains at most one process
- Computable and computable-outswapped
  - Each consists of a summary longword, and 32 queues
- Voluntary wait (LEF, LEFO, SUSP, SUSPO, HIB, HIBO)
  - One queue for each state
- Involutary wait (PFW, PFWO, FPG, FPGO, COLPG, COLPGO, MWAIT, MWAITO)
  - In four queues
  - Resident and outswapped in same queue (differentiate with resident bit in PCB$L_STS)
  - Usually not in these states very often

Process Data Structures Related to Scheduling

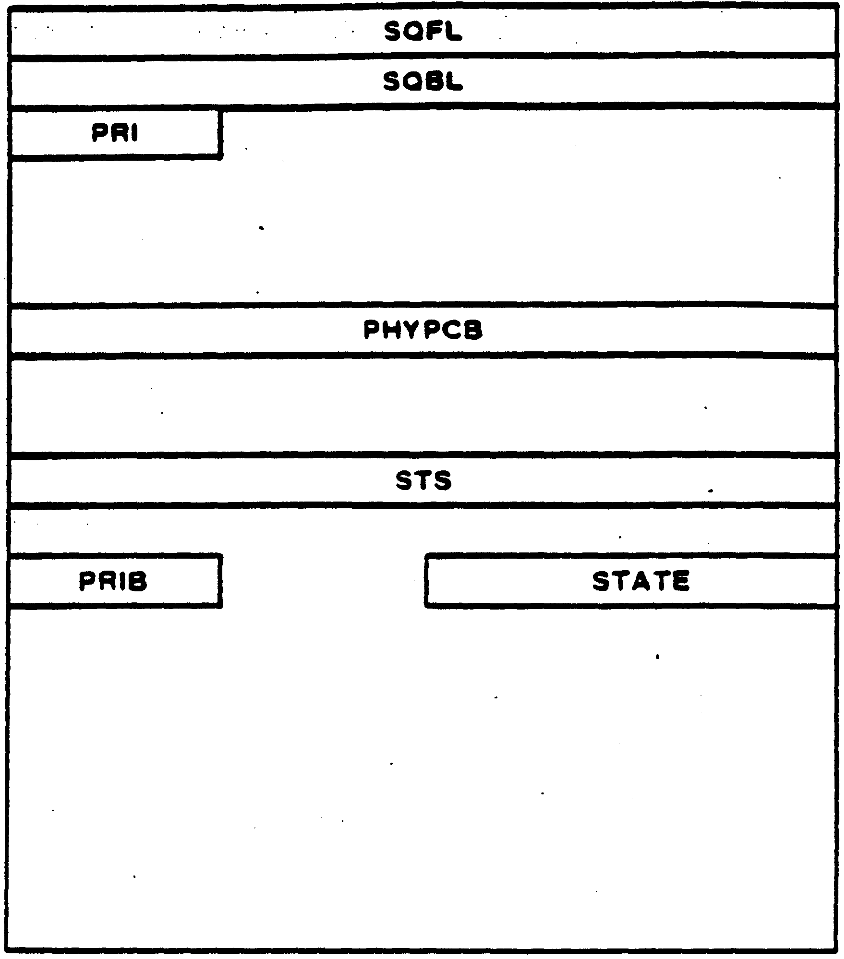


Figure Scheduling Fields in Software PCB

- SQFL, SQBL - state queue forward, backward links, link PCBs in a given state
- STATE - process state
- PRI - current software priority
- PRIB - base software priority
- PHYPCB - physical address of hardware PCB
- STS - process status

Saving and Restoring CPU Registers

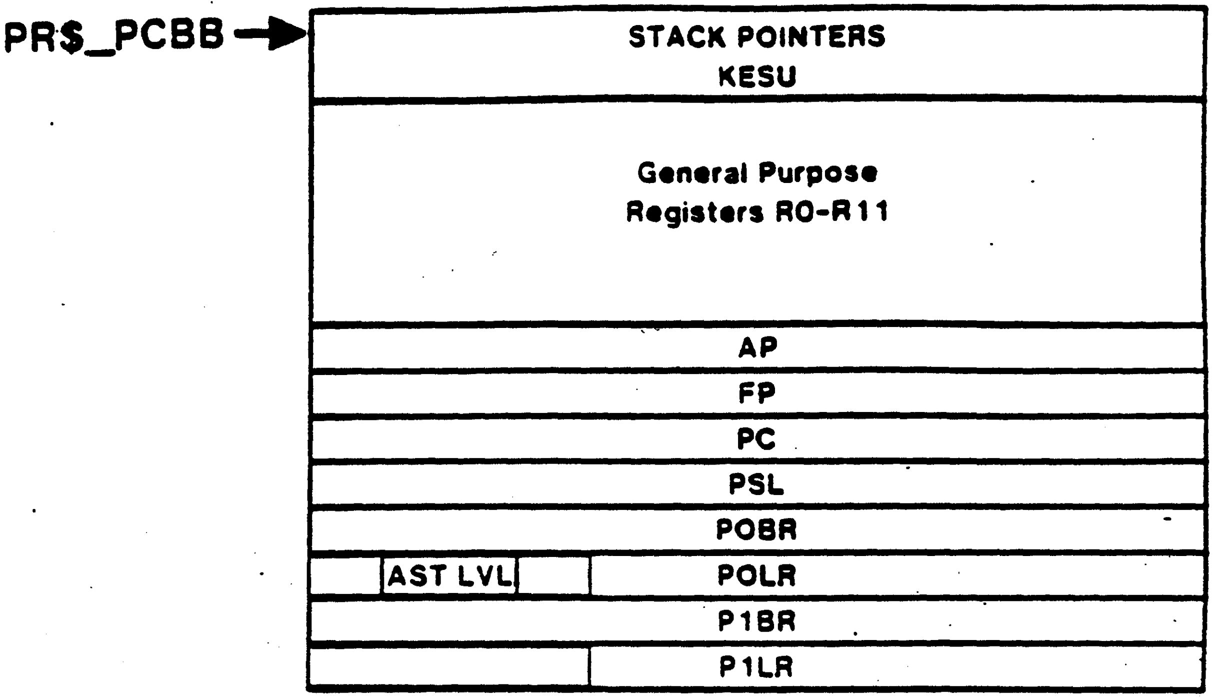


Figure Saving and Restoring CPU Registers

- Process-specific CPU registers saved/restored during context switch

- SVPCTX instruction

  - Copies registers to hardware PCB
  - Switches to Interrupt Stack
  - Does not save PØBR, PØLR, P1BR, P1LR, ASTLVL

- LDPCTX instruction

  - Restores registers (except PC, PSL) from hardware PCB
  - Pushes PC, PSL on kernel stack (REI removes them)

THE SCHEDULER (SCHED.MAR)

```
; SCH$RESCHED - RESCHEDULING INTERRUPT HANDLER
;
; THIS ROUTINE IS ENTERED VIA THE IPL 3 RESCHEDULING INTERRUPT.
; THE VECTOR FOR THIS INTERRUPT IS CODED TO CAUSE EXECUTION
; ON THE KERNEL STACK.
;
; ENVIRONMENT:     IPL=3 MODE=KERNEL IS=0
; INPUT:           00(SP)=PC AT RESCHEDULE INTERRUPT
;                  04(SP)=PSL AT INTERRUPT.
;--
     .ALIGN  LONG
MPH$RESCHED::                          ;MULTI-PROCESSING CODE HOOKS IN HERE
SCH$RESCHED::                          ;RESCHEDULE INTERRUPT HANDLER
     SETIPL  #IPL$_SYNCH               ;SYNCHRONIZE SCHEDULER WITH EVENT REPORTING
     SVPCTX                            ;SAVE CONTEXT OF PROCESS
     MOVL    L^SCH$GL_CURPCB,R1        ;GET ADDRESS OF CURRENT PCB
     MOVZBL  PCB$B_PRI(R1),R2          ;CURRENT PRIORITY
     BBSS    R2,L^SCH$GL_COMQS,10$     ;MARK QUEUE NON-EMPTY
10$: MOVW    #SCH$C_COM,PCB$W_STATE(R1) ;SET STATE TO RES COMPUTE
     MOVAQ   SCH$AQ_COMT[R2],R3        ;COMPUTE ADDRESS OF QUEUE
     INSQUE  (R1),@(R3)+               ;INSERT AT TAIL OF QUEUE
;+
; SCH$SCHED - SCHEDULE NEW PROCESS FOR EXECUTION
;
; THIS ROUTINE SELECTS THE HIGHEST PRIORITY EXECUTABLE PROCESS
; AND PLACES IT IN EXECUTION.
;-
MPH$SCHED::                            ;MULTI-PROCESSING CODE HOOKS IN HERE
SCH$SCHED::                            ;SCHEDULE FOR EXECUTION
     SETIPL  #IPL$_SYNCH               ;SYNCHRONIZE SCHEDULER WITH EVENT REPORTING
     FFS     #0,#32,L^SCH$GL_COMQS,R2         ;FIND FIRST FULL STATE
     BEQL    SCH$IDLE                  ;NO EXECUTABLE PROCESS??
     MOVAQ   SCH$AQ_COMH[R2],R3        ;COMPUTE QUEUE HEAD ADDRESS
     REMQUE  @(R3)+,R4                 ;GET HEAD OF QUEUE
     BVS     QEMPTY                    ;BR IF QUEUE WAS EMPTY (BUG CHECK)
     BNEQ    20$                       ;QUEUE NOT EMPTY
     BBCC    R2,L^SCH$GL_COMQS,20$     ;SET QUEUE EMPTY
20$:
     CMPB    #DYN$C_PCB,PCB$B_TYPE(R4)  ;MUST BE A PROCESS CONTROL BLOCK
     BNEQ    QEMPTY                    ;OTHERWISE FATAL ERROR
     MOVW    #SCH$C_CUR,PCB$W_STATE(R4)       ;SET STATE TO CURRENT
     MOVL    R4,L^SCH$GL_CURPCB        ;NOTE CURRENT PCB LOC
     CMPB    PCB$B_PRIB(R4),PCB$B_PRI(R4)     ;CHECK FOR BASE
                                              ;PRIORITY=CURRENT
     BEQL    30$                       ;YES, DONT FLOAT PRIORITY
     BBC     #4,PCB$B_PRI(R4),30$      ;DONT FLOAT REAL TIME PRIORITY
     INCB    PCB$B_PRI(R4)             ;MOVE TOWARD BASE PRIO
30$: MOVB    PCB$B_PRI(R4),L^SCH$GB_PRI       ;SET GLOBAL PRIORITY
     MTPR    PCB$L_PHYPCB(R4),#PR$_PCBB       ;SET PCB BASE PHYS ADDR
     LDPCTX                            ;RESTORE CONTEXT
     REI                               ;NORMAL RETURN

SCH$IDLE:                              ;NO ACTIVE, EXECUTABLE PROCESS
     SETIPL  #IPL$_SCHED               ;DROP IPL TO SCHEDULING LEVEL
     MOVB    #32,L^SCH$GB_PRI          ;SET PRIORITY TO -1(32) TO SIGNAL IDLE
     BRB     SCH$SCHED                 ;AND TRY AGAIN

QEMPTY:      BUG_CHECK QUEUEMPTY,FATAL        ;SCHEDULING QUEUE EMPTY

     .END
```

Example   The Scheduler (SCHED.MAR)

BOOSTING SOFTWARE PRIORITY OF NORMAL PROCESSES

- Usually normal interactive process has base priority 4
- To help interactive processes compete with compute-bound processes
  - Boosts applied upon certain events (I/O completion, resource available)
  - Different boosts for different events
  - Current priority equals greater of:
    - Current priority
    - Base priority plus boost
  - Lowering of priority
    - Each time process scheduled, decrement priority (until reach base priority)
    - Return to base priority at quantum end if COMO process exists
  - Not allowed to boost above normal priority range (Ø-15)

Example of Process Scheduling

Table 1 Initial Conditions for Scheduling Example

Process	Type	Base Priority	Priority	State
Swapper	System	16	16	HIB
Null	Compute Bound	0	0	COM
A	Compute Bound	4	9	CUR
B	I/O Bound	4	10	COMO
C	Real-Time	18	18	HIB

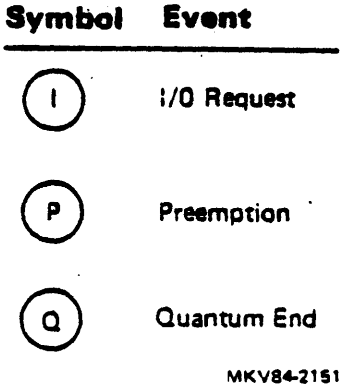


Figure 15 Scheduling Example Symbols

SCHEDULING

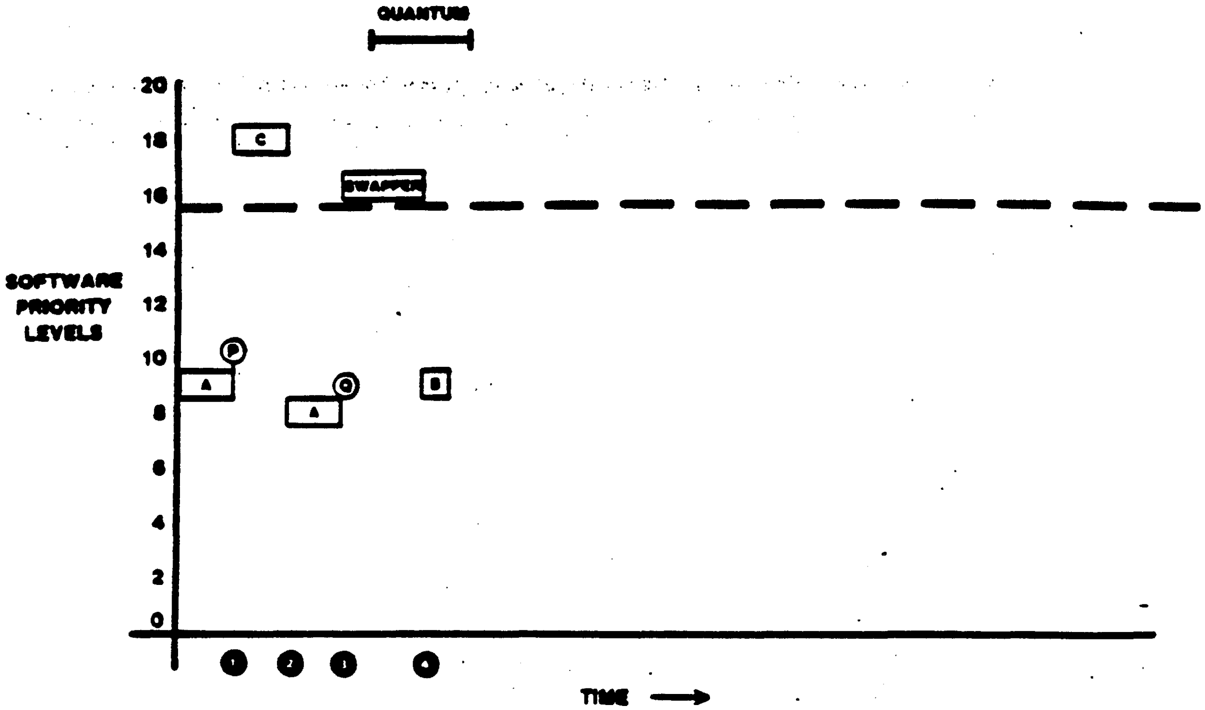


Figure Example of Process Scheduling - Part 1

1. Process C becomes computable. Process A is preempted.

2. C hibernates. A executes again, one priority level lower.

3. A experiences quantum end and is rescheduled at its base priority. B is computable outswapped.

4. The swapper process executes to inswap B. B is scheduled for execution.

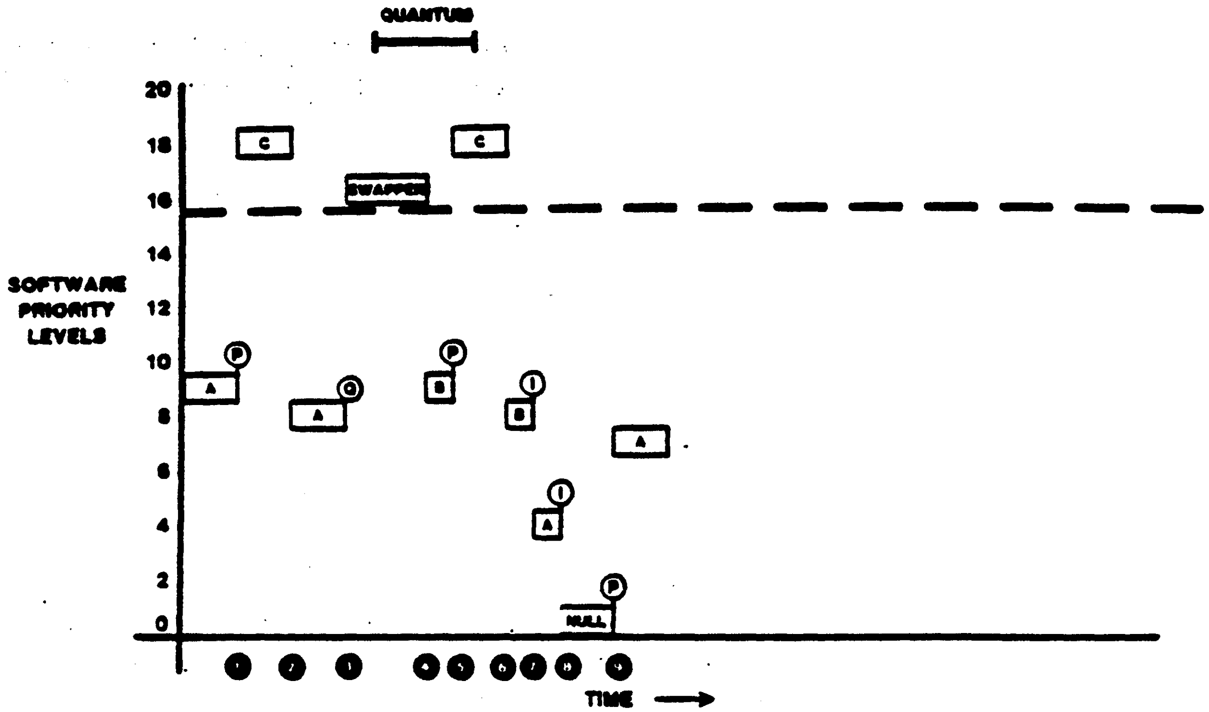


Figure   Example of Process Scheduling - Part 2

5. B is preempted by C.

6. B executes again, one priority level lower.

7. B requests an I/O operation (not terminal I/O). A executes at its base priority.

8. A requests a terminal output operation. The null process executes.

9. A executes following I/O completion at its base priority plus 3. (The applied boost was 4.)

SCHEDULING

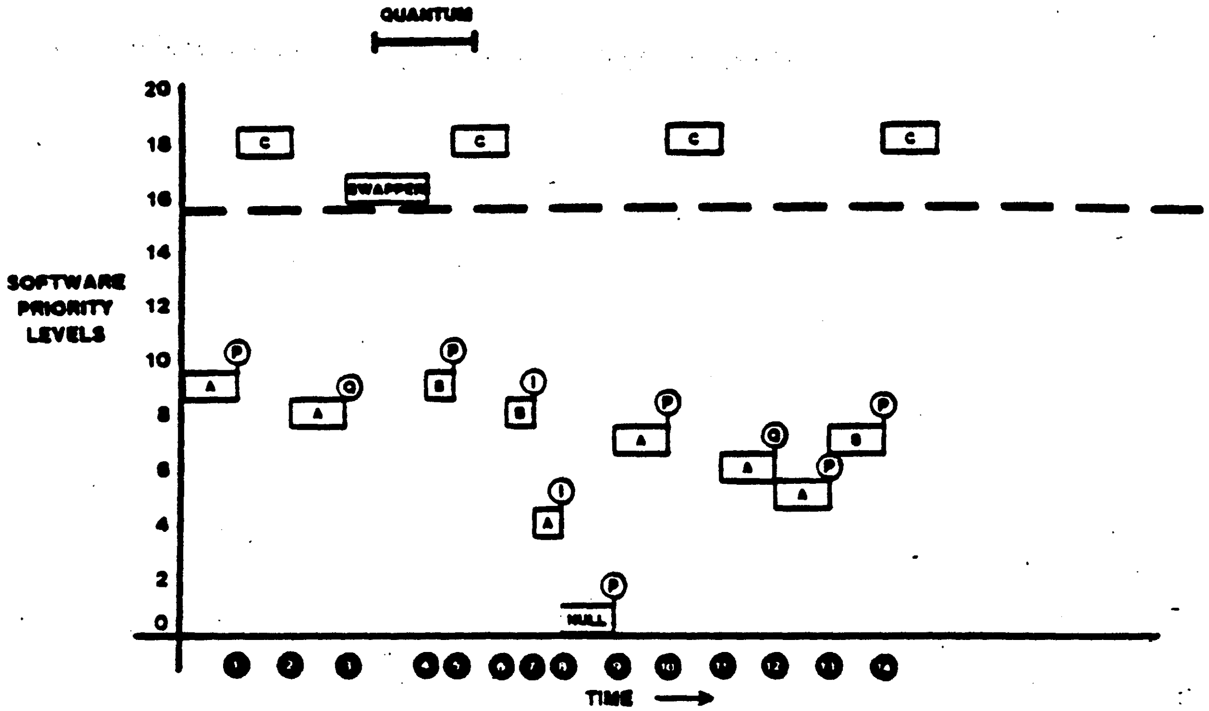


Figure · Example of Process Scheduling - Part 3

10. **A** is preempted by **C**.

11. **A** executes again, one priority level lower.

12. **A** experiences quantum end and is rescheduled at one priority level lower.

13. **A** is preempted by **B**. A priority boost of 2 is not applied to **B** because the result would be less than the current priority.

14. **B** is preempted by **C**.

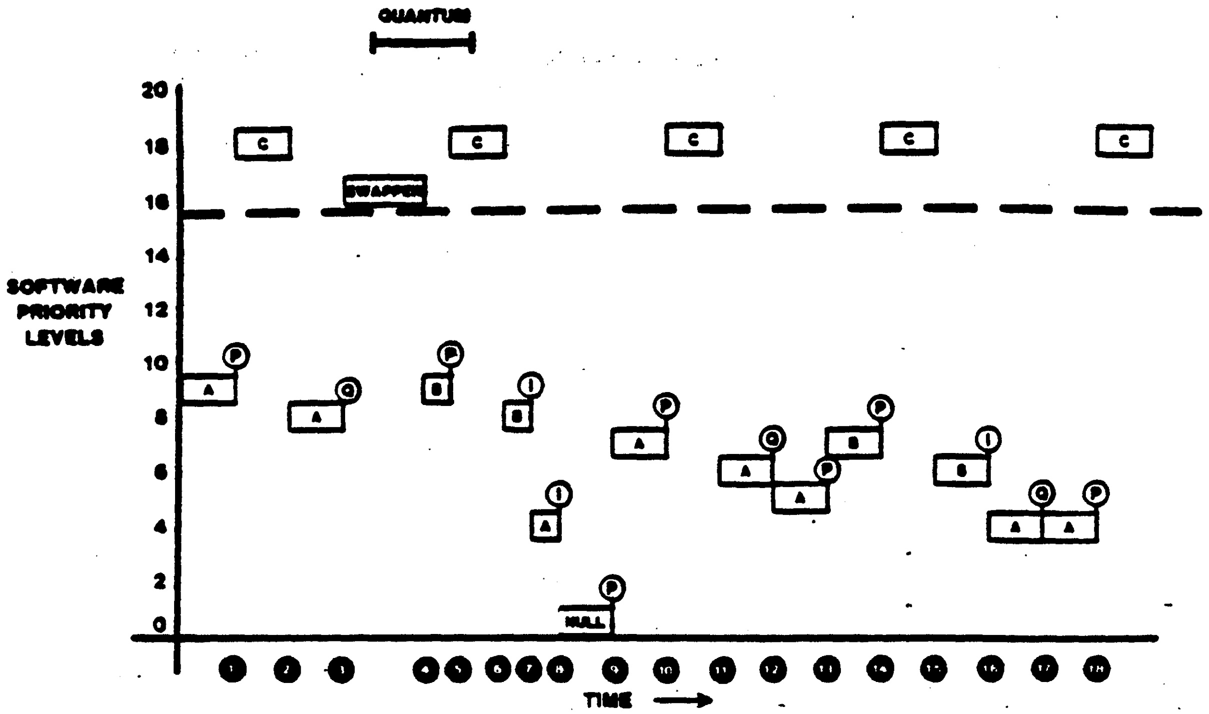


Figure Example of Process Scheduling - Part 4

15. B executes again, one priority level lower.

16. B requests an I/O operation. A executes at its base priority.

17. A experiences quantum end and is rescheduled at the same priority (its base priority).

18. A is preempted by C.

IMPLEMENTATION OF PROCESS STATE CHANGES

Table Operating System Code for Scheduling Functions

Function	Module	Routines
Change between CUR and COM	SCHED.MAR	SCH$RESCHED SCH$SCHED
Move between resident and outswapped	SWAPPER.MAR	SWAPSCHED INSWAP OUTSWAP
Move in and out of wait states	RSE.MAR	SCH$RSE SCH$UNWAIT (and others)
Quantum end processing	RSE.MAR	SCH$QEND

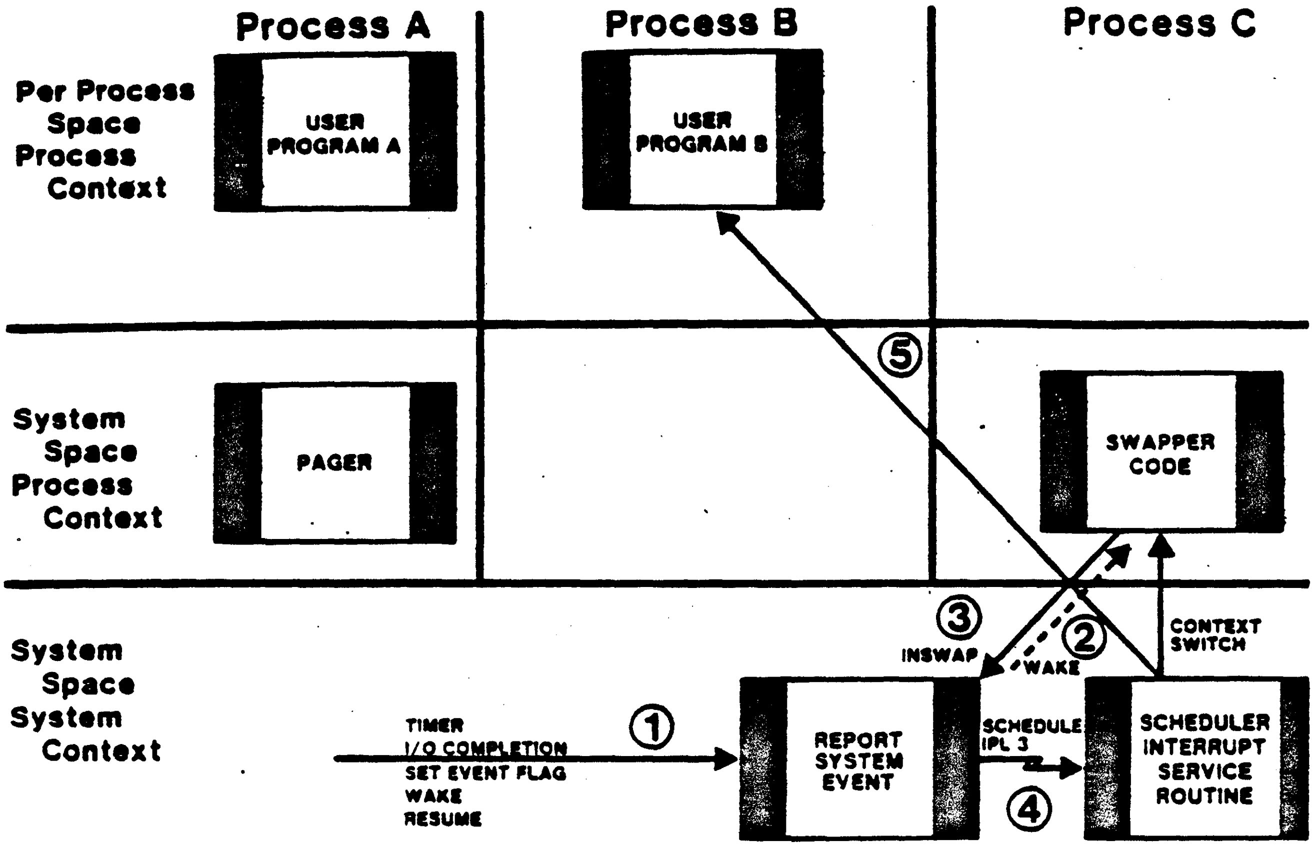


Figure Interaction of Scheduling Components

Report System Event Component (RSE.MAR)

1. System events cause transitions between process states.

2. These transitions are accomplished by the code in RSE.MAR.

3. Inputs to RSE

   a. PCB address

   b. Event number (number for WAKE, CEF SET, and so on)

4. RSE flow

   a. Event checked for significance (for example, WAKE only if in HIBER state).

   b. PCB removed from wait queue and wait queue header count decremented.

   c. PCB inserted on COM or COMO state queue after priority adjustment, and summary bit set.

   d. Swapper process can be awakened (if PCB was inserted on COMO queue).

   e. Scheduler interrupt at IPL 3 requested if the new computable process has software priority greater than that of current process.

STEPS AT QUANTUM END

Real-Time Process

1. Reset PHD$B_QUANT to full quantum value.
2. Clear initial quantum bit PCB$V_INQUAN in PCB$L_STS.

Normal Process

1. Reset PHD$B_QUANT to full quantum value.
2. Clear initial quantum bit PCB$V_INQUAN in PCB$L_STS.
3. If any outswapped process computable, set current software priority PCB$B_PRI to base priority PCB$B_PRIB.
4. If SWAPPER needed, wake SWAPPER.
5. If CPU limit imposed, and limit has expired, queue AST to process for process deletion.
6. If not, then calculate automatic working set adjustment.
7. Request scheduling interrupt at IPL 3.

Automatic Working Set Adjustment

- Goal: optimal working set size
  - Large enough to allow good program performance
  - Small enough to optimize overall memory usage
- Adjustment calculated at quantum end
  - If high paging rate, want to increase working set size
  - If low paging rate, may want to decrease working set size (take back some physical memory)
- Usually gives large increases, small decreases
- Only affects the list size, not the number of entries in use
- No adjustment done for real-time processes
- Can disable adjustment for normal processes
  - Perprocess: `$ SET WORKING_SET/NOADJUST`
  - System-wide: `SYSGEN> SET WSINC 0`

Automatic Working Set Adjustment

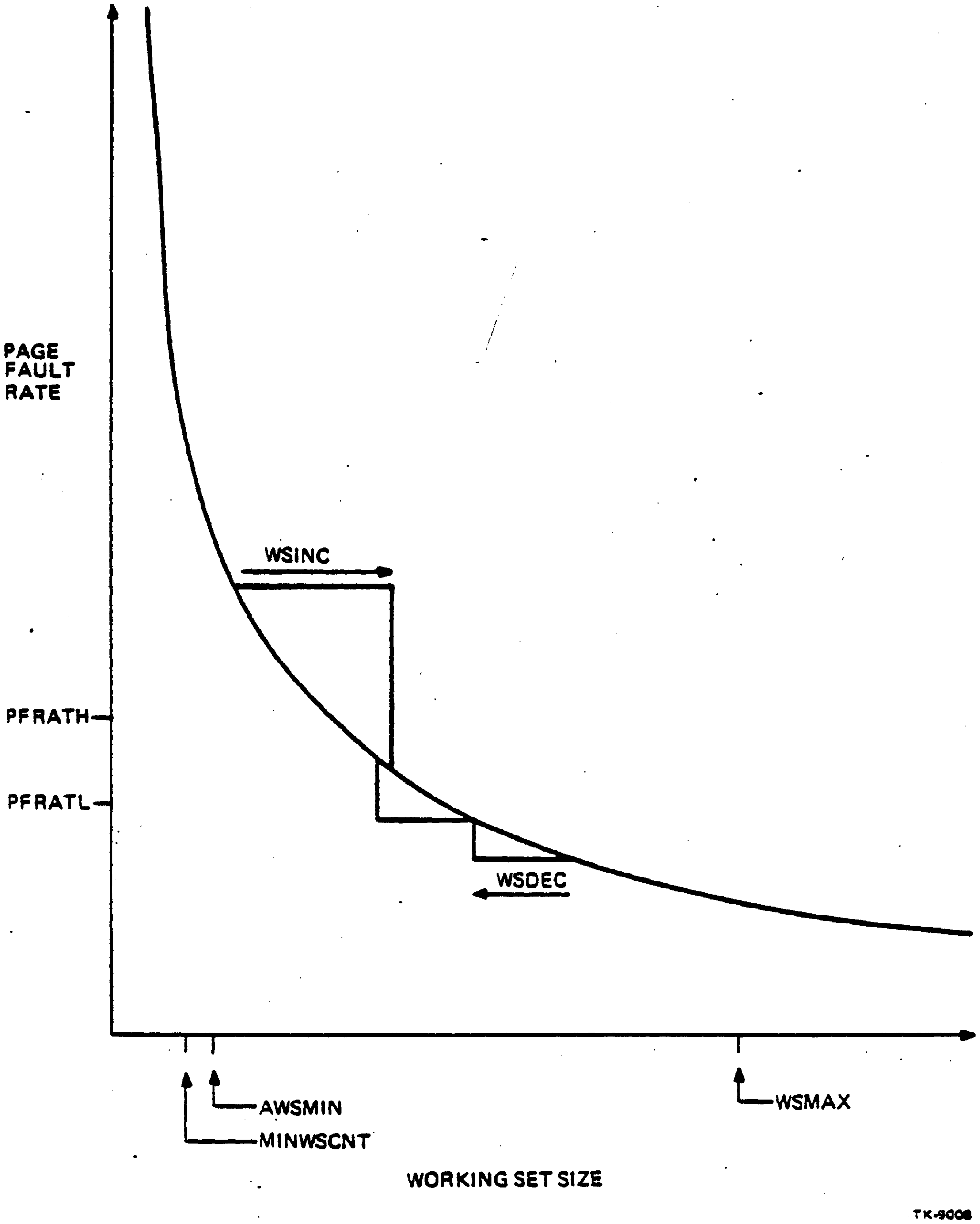


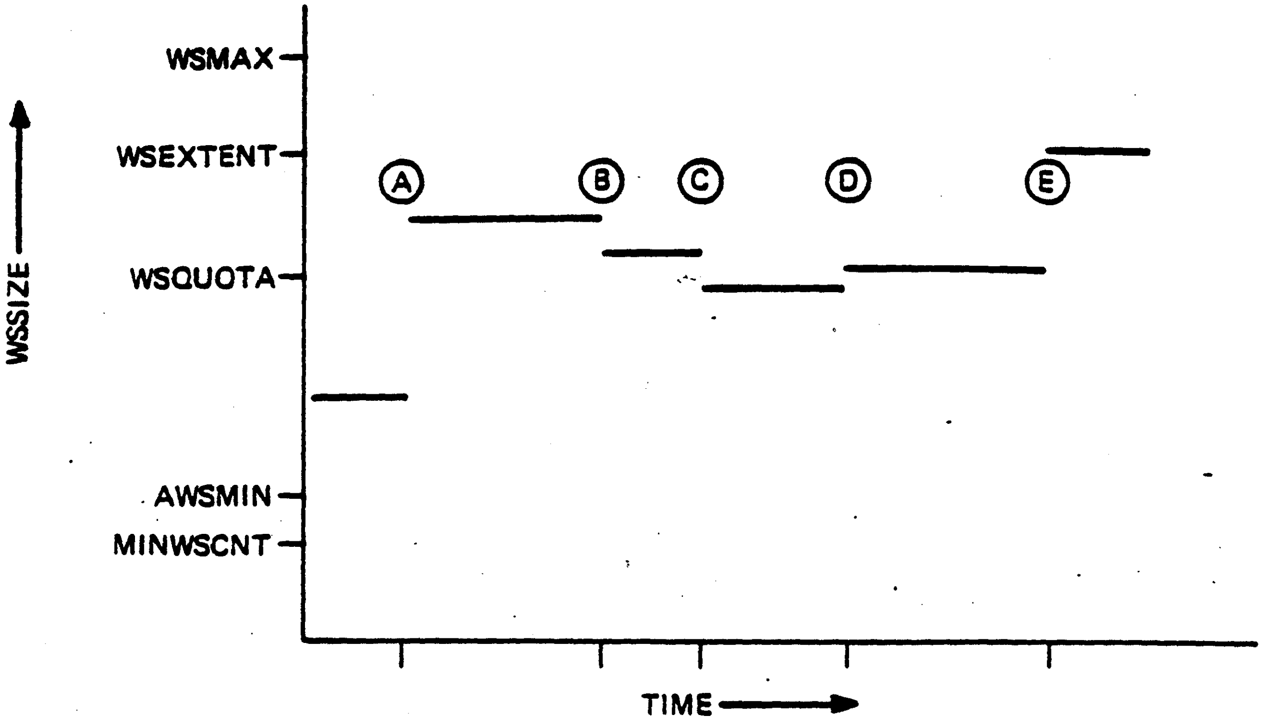
Figure Automatic Working Set Adjustment

Rules for Working Set Adjustment

1. If PFRATL < PFRate < PFRATH, no adjustment is necessary.
2. If PFRate > PFRATH then perhaps WSSIZE = WSSIZE + WSINC.
    - WSSIZE can grow to WSQUOTA anytime
    - WSSIZE can grow to WSEXTENT if free pages > BORROWLIM
3. If PFRate < PFRATL then perhaps WSSIZE = WSSIZE - WSDEC.
    - WSSIZE can shrink to AWSMIN (no smaller)

Example 2 Working Set Adjustment Algorithm

Example of Working Set Size Variation



TK-9012

Figure WSSIZE Variation Over Time

Table Reasons for Working Set Size Variations

Time	Reason for WSSIZE Change
a	Page faults > PFRATH Free page count > BORROWLIM
b	Page faults < PFRATL
c	Page faults < PFRATL
d	Page faults > PFRATH Free page count < BORROWLIM
e	Page faults > PFRATH Free page count > BORROWLIM

Forcing Processes to Quantum End

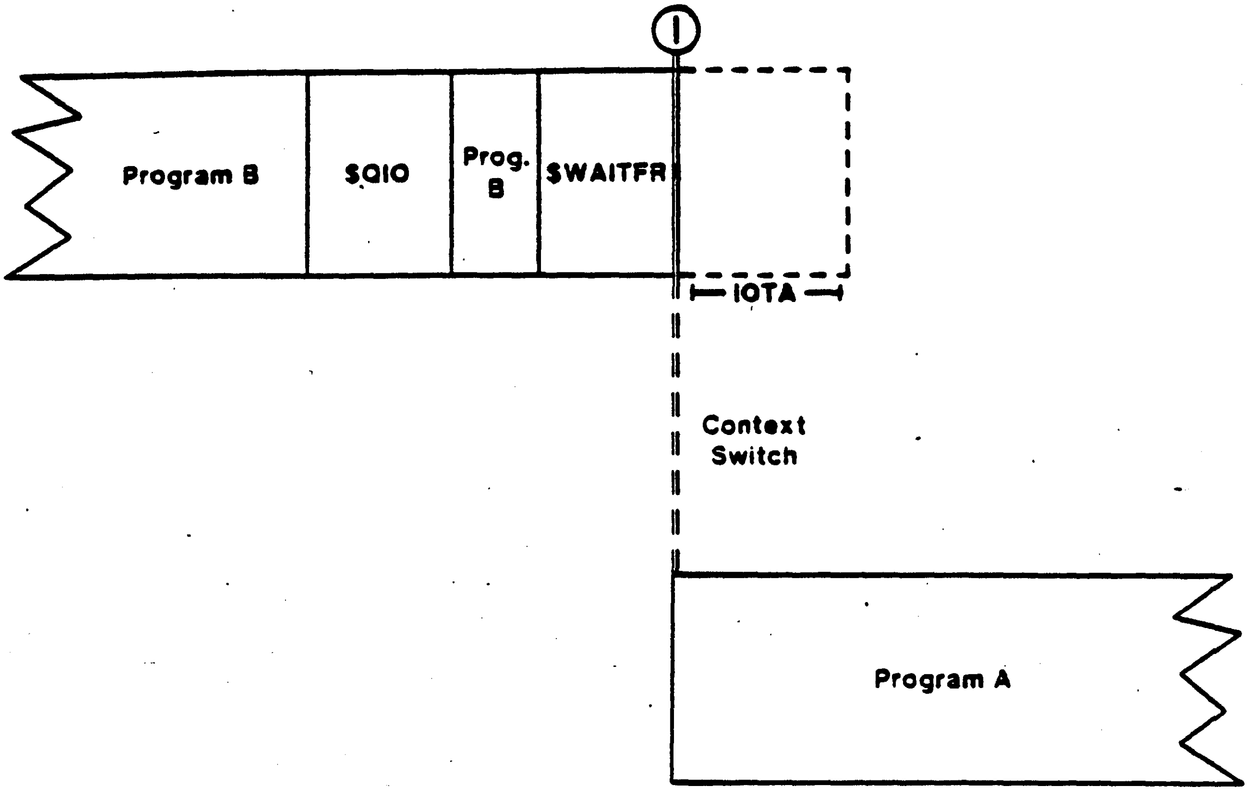


Figure Use of the IOTA System Parameter

- IOTA - special system parameter (in 10 ms units)
- Deduct IOTA units from time quantum when process enters wait state
- Used to force processes to quantum end
- Not charged to process CPU limit.

SOFTWARE PRIORITY LEVELS OF PROCESSES

Table Software Priority Levels of Processes on VMS

Process	**Base Priority**	**Purpose**
NULL	Ø	Consume idle CPU time
default user	4	User activities
SYMBIONT_n	4	Input/output symbiont
OPCOM	6	Operator communications
ODS-1 disk ACPs	8	ODS-1 disk file structure
Tape ACPS	8	Tape file structure
ERRFMT	7	Write error log buffers
JOB_CONTROL	8	Queue and accounting manager
NETACP	8	DECnet ACP
REMACP	8	Remote ACP
SWAPPER	16	System-wide memory manager

- Base priority of process determined by argument to $CREPRC system service
- Base priority of system processes
  - Most are established during system initialization
  - Base priority of ACPs is controlled by ACP_BASEPRIC system parameter
- Normal processes receive priority boosts

Table SYSGEN Parameters Relevant to Scheduling

Function	Parameter
Base priority for Ancillary Control Processes	ACP_BASEPRIO
Minimum number of working set pages	AWSMIN
Minimum amount of time that must elapse for significant sample of a process page fault rate	AWSTIME
Minimum number of pages required on free page list before working sets are allowed to grow beyond WSQUOTA (checked at quantum end)	BORROWLIM
Base default priority for processes	DEFPRI
Time alloted to each of a process's exit handlers after CPU limit expires	EXTRACPU
Amount of time to deduct from process quantum for each voluntary wait	IOTA (*)
Minimum number of fluid working set pages	MINWSCNT
Page fault rate above which VMS attempts to increase the process working set size	PFRATH
Page fault rate below which VMS attempts to decrease the process working set size	PFRATL
Maximum amount of CPU time a normal process can receive before control passes to a computable process of equal priority	QUANTUM
Number of pages for working set size decrease	WSDEC
Number of pages for working set size increase	WSINC
Maximum number of pages for any working set	WSMAX

(*) = special SYSGEN parameter

APPENDIX

```
$ SHOW MEMORY

              System Memory Resources on 20-JUL-1986 15:47:08.74

Physical Memory Usage (pages):     Total        Free      In Use    Modified
  Main Memory (4.00Mb)              8192        5852        2278          62

Slot Usage (slots):                Total        Free    Resident     Swapped
  Process Entry Slots                 19          12           7           0
  Balance Set Slots                   17          12           5           0

Fixed-Size Pool Areas (packets):   Total        Free      In Use        Size
  Small Packet (SRP) List            133          16         117          96
  I/O Request Packet (IRP) List       81           8          73         208
  Large Packet (LRP) List             14           5           9        1584

Dynamic Memory Usage (bytes):      Total        Free      In Use     Largest
  Nonpaged Dynamic Memory         222208       54880      167328       50624
  Paged Dynamic Memory            105472       36560       68912       35632

Paging File Usage (pages):                      Free      In Use       Total
  DISK$VMS:[SYS0.SYSEXE]SWAPFILE.SYS            4752        1248        6000
  DISK$VMS:[SYS0.SYSEXE]PAGEFILE.SYS            6662         338        7000

Of the physical pages in use, 1416 pages are permanently allocated to VMS.
```

Device Drivers.

LOGICAL NAME &
SYSTEM NAME TABLE

```
$ SHOW MEMORY/POOL/FULL

              System Memory Resources on 20-JUL-1986 16:13:10.81

Small Packet (SRP) Lookaside List              Packets       Bytes       Pages
    Current Total Size                             133       12768          25
    Initial Size (SRPCOUNT)     ) SYSGEN PARM       60        5760          12
    Maximum Size (SRPCOUNTV)                      3000      288000         563
    Free Space                                      17        1632
    Space in Use                                   116       11136
    Packet Size/Upper Bound (SRPSIZE)                           96
    Lower Bound on Allocation                                   32

I/O Request Packet (IRP) Lookaside List        Packets       Bytes       Pages
    Current Total Size                              83       17264          34
    Initial Size (IRPCOUNT)                         30        6240          13
    Maximum Size (IRPCOUNTV)                      2000      416000         813
    Free Space                                      10        2080
    Space in Use                                    73       15184
    Packet Size/Upper Bound (fixed)                            208
    Lower Bound on Allocation                                   97

Large Packet (LRP) Lookaside List              Packets       Bytes       Pages
    Current Total Size                              14       22176          44
    Initial Size (LRPCOUNT)                          8       12672          25
    Maximum Size (LRPCOUNTV)                        60       95040         186
    Free Space                                       5        7920
    Space in Use                                     9       14256
    Packet Size/Upper Bound (LRPSIZE + 80)                    1584
    Lower Bound on Allocation                                 1088

Nonpaged Dynamic Memory
    Current Size (bytes)           222208   Current Total Size (pages)     434
    Initial Size (NPAGEDYN)        222208   Initial Size (pages)           434
    Maximum Size (NPAGEVIR)        667648   Maximum Size (pages)          1304
    Free Space (bytes)              54880   Space in Use (bytes)        167328
    Size of Largest Block           50624   Size of Smallest Block          48
    Number of Free Blocks               7   Free Blocks LEQU 32 Bytes        0

Paged Dynamic Memory
    Current Size (PAGEDYN)         105472   Current Total Size (pages)     206
    Free Space (bytes)              36560   Space in Use (bytes)         68912
    Size of Largest Block           35632   Size of Smallest Block          48
    Number of Free Blocks               6   Free Blocks LEQU 32 Bytes        0
```

```
$ ANALYZE/SYSTEM

VAX/VMS System analyzer

SDA> SHOW SUMMARY

Current process summary
-----------------------
 Extended Indx Process name    Username    State Pri   PCB      PHD     Wkset
-- PID -- ---- --------------- ----------- ----- --- -------- -------- -----
 00000020 0000 NULL                         COM    0 800024A8 80002328     0
 00000021 0001 SWAPPER                      HIB   16 80002748 800025C8     0
 00000024 0004 JOB_CONTROL     SYSTEM       HIB    8 801093C0 8026C000   227
 00000026 0006 NETACP          DECNET       HIB   10 80112A50 8027BA00   187
 00000027 0007 EVL             DECNET       HIB    5 801130F0 8029AE00    43
 00000028 0008 REMACP          SYSTEM       HIB   13 8011A020 802AA800    30
 0000002B 000B SYSTEM          SYSTEM       CUR    5 8010E3D0 8028B400   665

SDA> SHOW SUMMARY/IMAGE

Current process summary
-----------------------
 Extended Indx Process name    Username    State Pri   PCB      PHD     Wkset
-- PID -- ---- --------------- ----------- ----- --- -------- -------- -----
 00000020 0000 NULL                         COM    0 800024A8 80002328     0
 00000021 0001 SWAPPER                      HIB   16 80002748 800025C8     0
 00000024 0004 JOB_CONTROL     SYSTEM       HIB    8 801093C0 8026C000   227
          DUA0:[SYS0.][SYSEXE]JOBCTL.EXE;2
 00000026 0006 NETACP          DECNET       HIB   10 80112A50 8027BA00   187
          DUA0:[SYS0.][SYSEXE]NETACP.EXE;6
 00000027 0007 EVL             DECNET       HIB    6 801130F0 8029AE00    43
          DUA0:[SYS0.][SYSEXE]EVL.EXE
 00000028 0008 REMACP          SYSTEM       HIB   13 8011A020 802AA800    30
          DUA0:[SYS0.][SYSEXE]REMACP.EXE;3
 0000002B 000B SYSTEM          SYSTEM       CUR    4 8010E3D0 8028B400   665
          DUA0:[SYS0.][SYSEXE]SDA.EXE;2
```

```
SDA> SHOW PROCESS/PCB

Process index: 000B  Name: SYSTEM  Extended PID: 0000002B
----------------------------------------------------------
Process status:  02040001   RES,PHDRES

PCB address                8010E3D0   JIB address                 801C2320
PHD address                8028B400   Swapfile disk address       01000361
Master internal PID        0001000B   Subprocess count                   0
Internal PID               0001000B   Creator internal PID        00000000
Extended PID               0000002B   Creator extended PID        00000000
State                         CUR     Termination mailbox             0000
Current priority                4     AST's enabled                   KESU
Base priority                   4     AST's active                    NONE
UIC                 [00001,000004]    AST's remaining                   21
Mutex count                     0     Buffered I/O count/limit        18/18
Waiting EF cluster              1     Direct I/O count/limit          18/18
Starting wait time         1B001B1A   BUFIO byte count/limit     20128/20128
Event flag wait mask       7FFFFFFF   # open files allowed left         15
Local EF cluster 0         E0000003   Timer entries allowed left        20
Local EF cluster 1         C0000000   Active page table count            0
Global cluster 2 pointer   00000000   Process WS page count            668
Global cluster 3 pointer   00000000   Global WS page count               0
```

RESOURCE WAIT
THIS ADDRESS WILL EXCEED
LOOK AT THIS IF YOU ARE WAITING FOR
ADDRESS OF A ~~STAK~~ SEMAPHORE (MUTEX) LIMIT
IT IS WAITING FOR.
IF NUMBER IS between 1 to 15 then it is a resource wait. The number will tell you what resource it is waiting for.

```
SDA> SHOW PROCESS/PHD

Process index: 000B  Name: SYSTEM  Extended PID: 0000002B
----------------------------------------------------------
Process header
--------------

First free P0 address       000A5800   Accumulated CPU time           00000E5A
Free PTEs between P0/P1        10699   CPU since last quantum             FFF9
First free P1 address       7FF5EE00   Subprocess quota                     10
Free page file pages            8896   AST limit                            24
Page fault cluster size           16   Process header index               0002
Page table cluster size            2   Backup address vector          000009BF
Flags                           0002   WSL index save area            00000980
Direct I/O count                 156   PTs having locked WSLs                5
Buffered I/O count               845   PTs having valid WSLs                18
Limit on CPU time           00000000   Active page tables                   21
Maximum page file count        10000   Maximum active PTs                   21
Total page faults               3628   Guaranteed fluid WS pages            20
File limit                        20   Extra dynamic WS entries            282
Timer queue limit                 20   Locked WSLE counts array           28F0
Paging file index           03000000   Valid WSLE counts array            2958
```

```
SDA> SET PROCESS/INDEX=28
SDA> SHOW PROCESS/PCB

Process index: 0008  Name: REMACP  Extended PID: 00000028
----------------------------------------------------------
Process status:  00148001   RES,NOACNT,PHDRES,LOGIN

PCB address                8011A020   JIB address                801C2730
PHD address                802AA800   Swapfile disk address      010004E1
Master internal PID        00010008   Subprocess count                  0
Internal PID               00010008   Creator internal PID       00000000
Extended PID               00000028   Creator extended PID       00000000
State                         HIB     Termination mailbox            0000
Current priority                 13   AST's enabled                  KESU
Base priority                     8   AST's active                   NONE
UIC                  [00001,000003]   AST's remaining                  99
Mutex count                       0   Buffered I/O count/limit   65534/65535
Waiting EF cluster                0   Direct I/O count/limit         18/18
Starting wait time         17001717   BUFIO byte count/limit     62783/62783
Event flag wait mask       7FFFFFFF   # open files allowed left      69
Local EF cluster 0         E0000000   Timer entries allowed left        8
Local EF cluster 1         00000000   Active page table count           0
Global cluster 2 pointer 00000000     Process WS page count            30
Global cluster 3 pointer 00000000     Global WS page count              0

SDA> SHOW PROCESS/PHD

Process index: 0008  Name: REMACP  Extended PID: 00000028
----------------------------------------------------------
Process header
--------------

First free P0 address      00008200    Accumulated CPU time        0000000F
Free PTEs between P0/P1       12455    CPU since last quantum          0005
First free P1 address      7FF9D000    Subprocess quota                   8
Free page file pages           1594    AST limit                        100
Page fault cluster size          16    Process header index            0004
Page table cluster size           2    Backup address vector       000009BF
Flags                          0006    WSL index save area         00000980
Direct I/O count                  2    PTs having locked WSLs             6
Buffered I/O count                5    PTs having valid WSLs              6
Limit on CPU time          00000000    Active page tables                 8
Maximum page file count        2048    Maximum active PTs                 8
Total page faults                64    Guaranteed fluid WS pages         20
File limit                       70    Extra dynamic WS entries         149
Timer queue limit                 8    Locked WSLE counts array        28F0
Paging file index          03000000    Valid WSLE counts array         2958
```

```
SDA> SHOW POOL/SUMMARY

Summary of IRP lookaside list

   53  FCB        =    11024  (68%)
   10  IRP        =     2080  (13%)
    1  NET        =      208  (1%)
    2  CXB        =      416  (2%)
    5  JIB        =     1040  (6%)
    5  RSB        =     1040  (6%)
    1  INIT       =      208  (1%)

Total space used = 16016 out of 17264 total bytes, 1248 bytes left

Total space utilization = 92%


Summary of LRP lookaside list

    8  CXB        =    12672  (89%)
    1  CDB        =     1584  (11%)

Total space used = 14256 out of 22176 total bytes, 7920 bytes left

Total space utilization = 64%


Summary of SRP lookaside list

    2  AQB        =      192  (1%)
    8  CRB        =      768  (6%)
    7  DDB        =      672  (5%)
    8  IDB        =      768  (6%)
    4  TQE        =      384  (3%)
   50  WCB        =     4800  (41%)
    5  BUFIO      =      480  (4%)
    1  NET        =       96  (0%)
    1  PTR        =       96  (0%)
   17  LKB        =     1632  (14%)
   12  RSB        =     1152  (9%)
    3  SCS        =      288  (2%)
    3  INIT       =      288  (2%)

Total space used = 11616 out of 12768 total bytes, 1152 bytes left

Total space utilization = 90%
```

```
Summary of non-paged pool contents

   37  UNKNOWN    =     19600  (11%)
    2  ADP        =      1792  (1%)
    1  AQB        =        32  (0%)
    1  LOG        =        32  (0%)
    5  PCB        =      1440  (0%)
   24  UCB        =     11584  (6%)
    2  VCB        =       480  (0%)
    1  TYPAHD     =       368  (0%)
    2  NET        =     13152  (7%)
    9  DPT        =     93488  (55%)
    3  RBM        =      3728  (2%)
    1  VCA        =       864  (0%)
    2  CDB        =       736  (0%)
    1  LKID       =       400  (0%)
    1  RSHT       =       272  (0%)
    7  SCS        =      5904  (3%)
    1  LOADCODE   =      2752  (1%)
    3  INIT       =      9840  (5%)
    1  CLASSDRV   =       512  (0%)
    1  UIS        =       352  (0%)

Total space used = 167328 out of 222208 total bytes, 54880 bytes left

Total space utilization = 75%


Summary of paged pool contents

    5  UNKNOWN    =     23232  (33%)
    1  PQB        =      2256  (3%)
   11  GSD        =       528  (0%)
   19  KFE        =      1232  (1%)
    1  MTL        =        32  (0%)
    1  NDB        =      2544  (3%)
   14  KFRH       =      4160  (6%)
    1  TWP        =     12336  (17%)
    1  RSHT       =       528  (0%)
    1  XWB        =     16384  (23%)
   40  LNM        =      3104  (4%)
    1  KFPB       =        16  (0%)
    1  CIA        =      2560  (3%)

Total space used = 68912 out of 105472 total bytes, 36560 bytes left

Total space utilization = 65%
```

DATA STRUCTURE TYPE DEFINITIONS

ACB	AST CONTROL BLOCK	
ACL	ACCESS CONTROL LIST QUEUE ENTRY	
ADP	UNIBUS ADAPTER CONTROL BLOCK	
AQB	ACP QUEUE BLOCK	
BRDCST	BROADCAST MESSAGE BLOCK	
BUFIO	BUFFERED I/O BLOCK	
CDB	X25 LES CHANNEL DATA BLOCK	
CDRP	CLASS DRIVER REQUEST PACKET	
CEB	COMMON EVENT BLOCK	
CHIP	Internal CHKPRO block	
CI	CI PORT SPECIFIC	(*** See Subtypes Below ***)
CIA	Compound Intrusion Analysis block	
CIDG	DATAGRAM BUFFER FOR CI PORT	
CIMSG	MESSAGE BUFFER FOR CI PORT	
CLASSDRV	CLASS DRIVER MAJOR STRUCTURE	(*** See Subtypes Below ***)
CLU	CLUSTER MAJOR STRUCTURE	(*** See Subtypes Below ***)
CRB	CHANNEL REQUEST BLOCK	
CXB	COMPLEX CHAINED BUFFER	
DCCB	Data Cache Control Block	
DDB	DEVICE DESCRIPTOR BLOCK	
DPT	DRIVER PROLOGUE TABLE	
DSRV	Disk Server structure type	(*** See Subtypes Below ***)
ERP	ERRORLOG PACKET	
EXTGSD	EXTENDED GLOBAL SECTION DESCRIPTOR	
FCB	FILE CONTROL BLOCK	
FLK	Fork Lock Request Block	
FRK	FORK BLOCK	
GSD	GLOBAL SECTION DESCRIPTOR BLOCK	
IDB	INTERRUPT DISPATCH BLOCK	
INIT	STRUCTURES SET UP BY INIT	(*** See Subtypes Below ***)
IRPE	I/O REQUEST PACKET EXTENSION	
IRP	I/O REQUEST PACKET	
JIB	JOB INFORMATION BLOCK	
JPB	JOB PARAMETER BLOCK	
KFD	Known File Device Directory block	
KFE	KNOWN FILE ENTRY	
KFPB	Known File list Pointer Block	
KFRH	KNOWN FILE IMAGE HEADER	
LKB	LOCK BLOCK	
LKID	LOCK ID TABLE	
LNM	LOGICAL NAME BLOCK	
LOADCODE	LOADABLE CODE	(*** See Subtypes Below ***)
LOG	LOGICAL NAME BLOCK	
LPD	X25 LES PROCESS DESCRIPTOR	
MBX	MAILBOX CONTROL BLOCK	
MP	ASMP related structure	(*** See Subtypes Below ***)
MTL	MOUNTED VOLUME LIST ENTRY	
MVL	MAGNETIC TAPE VOLUME LIST	
NDB	NETWORK NODE DESCRIPTOR BLOCK	
NET	NETWORK MESSAGE BLOCK	
ORB	Objects Rights Block	
PBH	PERFORMANCE BUFFER HEADER	
PCB	PROCESS CONTROL BLOCK	
PDB	PERFORMANCE DATA BLOCK	
PFB	Page Fault Monitor Buffer	
PFL	PAGE FILE CONTROL BLOCK	
PGD	PAGED DYNAMIC MEMORY	(*** See Subtypes Below ***)

PIB	PERFORMANCE INFORMATION BLOCK
PMB	Page Fault Monitor Control Block
PQB	PROCESS QUOTA BLOCK
PTR	POINTER CONTROL BLOCK
QVAST	QVSS AST block
RBM	Realtime SPT bit map
RIGHTSLIST	RIGHTS LIST
RSB	RESOURCE BLOCK
RSHT	RESOURCE HASH TABLE
RVT	RELATIVE VOLUME TABLE
SCS	SYSTEM COMMUNICATION SERVICES(*** See Subtypes Below ***)
SHB	SHARED MEMORY CONTROL BLOCK
SHMCEB	SHARED MEMORY MASTER COMMON EVENT BLOCK
SHMGSD	SHARED MEMORY GLOBAL SECTION DESCRIPTOR
SLAVCEB	SLAVE COMMON EVENT BLOCK
SSB	LOGICAL LINK SUBCHANNEL STATUS BLOCK
TQE	TIMER QUEUE ENTRY
TWP	Terminal driver write packet
TYPAHD	TERMINAL TYPEAHEAD BUFFER
UCB	UNIT CONTROL BLOCK
UIS	UIS Structure (*** See Subtypes Below ***)
VCA	Disk volume cache block
VCB	VOLUME CONTROL BLOCK
WCB	WINDOW CONTROL BLOCK
WQE	DECNET WORK QUEUE BLOCK
XWB	DECNET LOGICAL LINK CONTEXT BLOCK

SUBTYPE CODES

CODES THAT ARE SUBTYPABLE REFER TO A GENERIC FUNCTION AND WITHIN THAT FUNCTION THERE MAY BE MANY DIFFERENT SUB-TYPES OF BLOCKS.

THE SUB-TYPE IS IN THE 12TH BYTE.

SCS

SCS_CDL	CONNECT DISPATCH LIST
SCS_CDT	CONNECT DISPATCH TABLE
SCS_DIR	DIRECTORY BLOCK
SCS_PB	PATH BLOCK
SCS_PDT	PORT DESCRIPTOR TABLE
SCS_RDT	REQUEST DESCRIPTOR TABLE
SCS_SB	SYSTEM BLOCK
SCS_SPPB	SCA POLLER PROCESS BLOCK
SCS_SPNB	SCA POLLER NAME BLOCK

CI CI PORT SPECIFIC

CI_BDT	BUFFER DESCRIPTOR TABLE
CI_FQDT	FREE QUE DESCRIPTOR TABLE

LOADCODE	LOADABLE CODE
NON_PAGED	NON PAGED CODE
PAGED	PAGED CODE
LC_MP	MULTIPROCESSOR CODE
LC_SCS	SCS CODE
LC_CLS	CLUSTER CODE
LC_CHREML	CHAR/DECIMAL INS EMUL
LC_FPEMUL	FLOAT PNT EMULATOR
LC_MSCP	MSCP SERVER
LC_SYSL	SYSLOA

INIT	STRUCTURES SET UP BY INIT
PCBVEC	PROCESS CONTROL BLOCK VECTOR
PHVEC	PROCESS HEADER VECTOR
SWPMAP	SWAPPER MAP
MPWMAP	MODIFIED PAGE WRITER MAP
PRCMAP	PROCESS BITMAP
BOOTCB	BOOT CONTROL BLOCK
CONF	CONFIGURATION ARRAYS
CST	CLUSTER SYSTEM TABLE

CLASSDRV	CLASS DRIVER MAJOR STRUCTURE TYPE CODE
CD_CDDB	CLASS DRIVER DATA BLOCK
CD_BBRPG	BAD BLOCK REPLACEMENT PAGE
CD_SHDW_WRK	SHADOW SET WORK BUFFER

CLU	CLUSTER MAJOR STRUCTURE TYPE CODE
CLU_CSB	CONNECTION STATUS BLOCK
CLU_CLUVEC	CLUSTER SYSTEM VECTOR
CLU_CLUB	CLUSTER BLOCK
CLU_BTX	CLUSTER BLOCK TRANSFER EXTENSION
CLU_CLUDCB	CLUSTER DISK QUORUM CONTROL BLOCK
CLU_CLUOPT	CLUSTER OPTIMAL RECONFIGURATION CONTEXT BLOCK
CLU_LCKDIR	LOCK MANAGER DISTRIBUTED DIRECTORY VECTOR

PGD	PAGED DYNAMIC MEMORY
PGD_F11BC	F11BXQP BUFFER CACHE.

UIS	UIS Structure
UIS_ARD	Allocation region

DSRV   Disk Server structure type

DSRV_DSRV	Disk server structure
DSRV_HQB	Host Queue Block
DSRV_HRB	Host Request Block
DSRV_IOBUF	Server local I/O Buffer
DSRV_UQB	Unit Queue Block.

MP   ASMP related structure

MP_CONSARRY	Logical Console Array
MP_CONSBFCTL	Logical Console Buffer

Extracting the Structure Code Values

```
     $ LIBRARY/MACRO/EXTRACT=$DYNDEF/OUTPUT=DYNDEF.MAR SYS$LIBRARY:LIB.MLB
```

```
$DYNDEF
---------------
$EQU     DYN$C_ADP        1
$EQU     DYN$C_ACB        2
$EQU     DYN$C_AQB        3
$EQU     DYN$C_CEB        4
$EQU     DYN$C_CRB        5
$EQU     DYN$C_DDB        6
$EQU     DYN$C_FCB        7
$EQU     DYN$C_FRK        8
$EQU     DYN$C_IDB        9
$EQU     DYN$C_IRP        10
$EQU     DYN$C_LOG        11
  .        .    .         .
  .        .    .         .
  .        .    .         .
$EQU     DYN$C_RIGHTSLIST         66
  .        .    .         .
  .        .    .         .
  .        .    .         .
$EQU     DYN$C_LOADCODE   98
$EQU     DYN$C_NON_PAGED  1
$EQU     DYN$C_PAGED      2
$EQU     DYN$C_LC_MP      3
$EQU     DYN$C_LC_SCS     4
$EQU     DYN$C_LC_CLS     5
$EQU     DYN$C_LC_CHREML  6
$EQU     DYN$C_LC_FPEMUL  7
$EQU     DYN$C_LC_MSCP    8
$EQU     DYN$C_LC_SYSL    9
  .        .    .         .
  .        .    .         .
  .        .    .         .
         $DEFEND DYN,$GBL,DEF
         .ENDM
```