TABLE OF CONTENTS

COS/IOS Overview	Section 1
System Programs	Section 2
Program Switching and Control	Section 3
System Resources	Section 4
Station Protocol	Section 5
Inter-Task Communication	Section 6
Inter-IOP Communication	Section 7
COS Dump Analysis	Section 8
IOS Dump Analysis	Section 9
Job Flow Overview	Section 10
Dump Exercises	Section 11
Appendix A - Quick References	
Appendix B - COS/IOS Generation	
Quiz and Exercise Answers	

SOFTWARE FOR CUSTOMER ENGINEERS III - INTERNALS

Duration: 5 Days

Maximum Class Size: 10 Students

Prerequisites: Cray Employee
Advanced Troubleshooting Dump Analysis
Six months site experience (COS Preferred)

Course Description: A systems level course teaching you the terminology and skills the analysts use in determining the reason for an operating system hang. The course is focused on an analysis of four memory dumps to determine the cause of the software hang. COS/IOS system components are examined together for an operating system library overview. Inter-Program communication is studied by examining history trace entries.

Course Content:

1. COS/IOS Overview
2. System Programs
3. Program Switching and Control
4. System Resources
5. Station Protocol
6. Inter-Task Communication
7. Inter-IOP Communication
8. COS Dump Analysis
9. IOS Dump Analysis
10. Dump Exercises

Course Objectives:

1. Locate a needed system program and determine it's function.
2. Draw Central, Local and Buffer Memory Maps.
3. Analyze COS and IOS Memory Dumps for cause of hang.
4. Locate Hang Messages in dump and Hang Macro in the Listing.
5. Analyze COS and IOS History Trace Buffer entries.

Motivation:

1. To communicate better with customers, operators, and analyst.
2. The Engineer becomes more self sufficient.
3. Improves your understanding of system operation.
4. To help isolate problems that fail online only.
5. Increased reliability by improving isolation of two time hits.
6. More time for analyst to spend on software problems.
7. Future Cray products require stronger software background.

SOFTWARE FOR CUSTOMER ENGINEERS III

Monday	Tuesday	Wednesday	Thursday	Friday
COS/IOS Overview Functions Components EXEC KERNEL Exercise 1	Program Switching XP's SMOD Any Packet Disk Activity Link Exercise 3	Inter-Task Communication Exercise 6	COS Dump Analysis Hang Message Hang Macro History Trace Program Regs	IOS Dump Analysis Hang Message Hang Macro History Trace Program Regs Exercise 9

System Programs COS 15 Tasks IOS 500 Overlays Exercise 2	System Resources Disk BMR SSD CM LM BM Exercise 4 Station Protocol LCP Message Code SCB Exercise 5	Inter-IOP Communication Exercise 7	Exercise 8	Learning Logs Evaluations

COURSE MATERIALS

Software for Customer Engineers III	Workbook
COS Internal Reference	SM-0040
COS Table Descriptions	SM-0045
IOS Internal Reference	SM-0046
IOS Table Descriptions	SM-0007
FE Station Protocol Reference	SM-0042

2 IOS Dumps

2 COS Dumps

EXEC Listing

KERNEL Listing

Necessary Task and Overlay Listings

- EXP
- HSPTEST

READING ASSIGNMENT

Monday Night:

SM-0040	pages 1-1 to 1-7 1-14 to 1-28	COS Basics
SM-0046	pages 1-1 to 1-7 2-1 to 2-12 3-1 to 3-8	IOS Introduction Kernel Disk Subsystem
SM-0042	pages 1-1 to 4-2	Station Protocol

Tuesday Night:

SM-0040	pages 2-1 to 2-10 2-70 to 2-96 3-1 to 3-8	EXEC EXEC Debug Aids Tasks
SM-0046	pages 2-42 to 2-47	IOS Communication
SM-0046	Chapter 10	IOS Debug Aids

Wednesday Night:

SM-0040	Chapter 2-17 to 2-41 Chapter 8-1 to 8-40	EXEC Requests System Action Requests
SM-0046	Chapter 2-11 to 2-38	Kernel Service Requests

Thursday Night:

SM-0046	Chapter 9 Appendix A	IOS Macros Dump Analysis

EVALUATION METHOD

EVALUATION OF YOUR PROGRESS IN GAINING EXPERTISE IN THESE SKILLS IS ACCOMPLISHED BY ASSIGNING A COMPETENCY LEVEL TO EACH SKILL.

Level

0 No knowledge and no experience.

1 Has some knowledge and limited experience with this skill, but not sufficient to contribute in a work environment.

2 Can perform some parts of this skill satisfactorily but requires instruction and supervision to perform the entire skill.

3 Can perform some parts of this skill satisfactorily but requires periodic supervision and/or assistance.

4 Can perform this skill satisfactorily without assistance and/or supervision.

5 Can perform this skill with proficiency in speed and quality without supervision or assistance.

6 Can perform this skill with initiative and adaptability to special situations without supervision or assistance.

7 Can perform this skill and can lead others in performing it.

Successfully completing this course should give you a competency level of three (3) for most skills. Experience on the job will continue to increase your competency level.

Software for Customer Engineers III

Date: _____

Participant's Name:

_____

Instructor's Name:

_____

Region/Country:

_____

LEARNING LOG

CESW III									
Skills At the end of the course the learner is able to:									
Locate a specified System Program or routine and determine its function.									
Draw Central Memory, Local and Buffer Memory Maps.									
Analyze IOS and COS Memory Dumps.									
Locate Hang Messages and Hang Macro.									
Analyze History Trace Buffer Entries.									
Levels	0	1	2	3	* 4	5	6	7	No Basis For Judgement

Sessions attended/held ____/____

Exercises completed/assigned ____/____

Labs attended/held ____/____

This learning log is intended as an aid to the learner in establishing goals and plotting progress. It is not intended as an indicator of job performance and therefore should not be used in determining future job actions.

*Maximum level discernible by the instructor in an instructional environment.

INSTRUCTOR'S FEEDBACK

Sessions attended/held ____/____
Exercises completed/assigned ____/____
Labs attended/held ____/____

MET THE PREREQUISITES OF THE COURSE

not at all — yes — was over qualified

Specifics:

SELF APPRAISAL

too high — is correct — too low

3 levels 2 levels 1 level 1 level 2 levels 3 levels

Specifics:

WAS ACTIVE AND ATTENTIVE IN CLASS

not at all — to a normal degree — exceptionally so

Specifics:

MADE GOOD USE OF LAB TIME

not at all — to a normal degree — exceptionally so

Specifics:

MADE GOOD USE OF TERMINAL TIME

not at all — to a normal degree — exceptionally so

Specifics:

KEPT UP WITH THE REST OF THE CLASS

fell behind the class — yes — was ahead of the class

Specifics:

SHOWS A POSITIVE ATTITUDE ABOUT WORKING AT CRAY

not at all — to a normal degree — exceptionally so

Specifics:

Comments:

These are subjective appraisals based on the instructors brief and limited observations of the learners behavior during the class.

Software for Customer Engineers III

Date: _____

Participant's Name:

_____

Instructor's Name:

_____

Region/Country:

_____

LEARNING LOG

CESW III									
Skills At the end of the course the learner is able to:									
Locate a specified System Program or routine and determine its function.									
Draw Central Memory, Local and Buffer Memory Maps.									
Analyze IOS and COS Memory Dumps.									
Locate Hang Messages and Hang Macro.									
Analyze History Trace Buffer Entries.									
Levels	0	1	2	3	* 4	5	6	7	No Basis For Judgement

Sessions attended/held ____/____

Exercises completed/assigned ____/____

Labs attended/held ____/____

This learning log is intended as an aid to the learner in establishing goals and plotting progress. It is not intended as an indicator of job performance and therefore should not be used in determining future job actions.

*Maximum level discernible by the instructor in an instructional environment.

Software for Customer Engineers (1)

Date: _____

Participant's Name: _____

Instructor's Name: _____

Region/Country: _____

(CONTINUED)

CESW III											
Skills At the end of the course the learner is able to:											
Locate a specified System Program or routine and determine its function.											
Draw Central Memory, Local and Buffer Memory Maps.											
Analyze IOS and COS Memory Dumps.											
Locate Hung Messages and Hung Macro.											
Analyze History Trace Buffer Entries.											
Levels	0	1	2	3	4	5	6	7	No Basis for Judgement		

* Sensitive attended/held _____

Electronic computer/assigned _____

+ Less attended/held _____

This learning log is intended as an aid to the learner in assessing work and planning progress. It is not intended as an indicator of job performance and therefore should not be used in determining future "action".

"Maximum level discernible by the instructor in an instructional environment.

COS/IOS Overview

1

LEARNING OBJECTIVES

With the aid of all reference material, upon completion of the COS/IOS Overview module, the learner should be able to:

1. Explain what COS Functions are.

2. Explain what IOS Functions are.

3. List the Software Components and their functions.

COS FUNCTIONS

An operating system is a collection of programs which monitor and control the flow of work presented to the computer in the form of jobs.

The Cray Operating System (COS) is a multiprogramming operating system designed to process jobs which are submitted from one or more front-end computers.

COS consists of memory and mass storage resident programs for:

Scheduling, Loading, Supervising of Jobs

Allocating System Resources

Initiating and Controlling I/O Operations

Handling System Errors

Logging System Activities

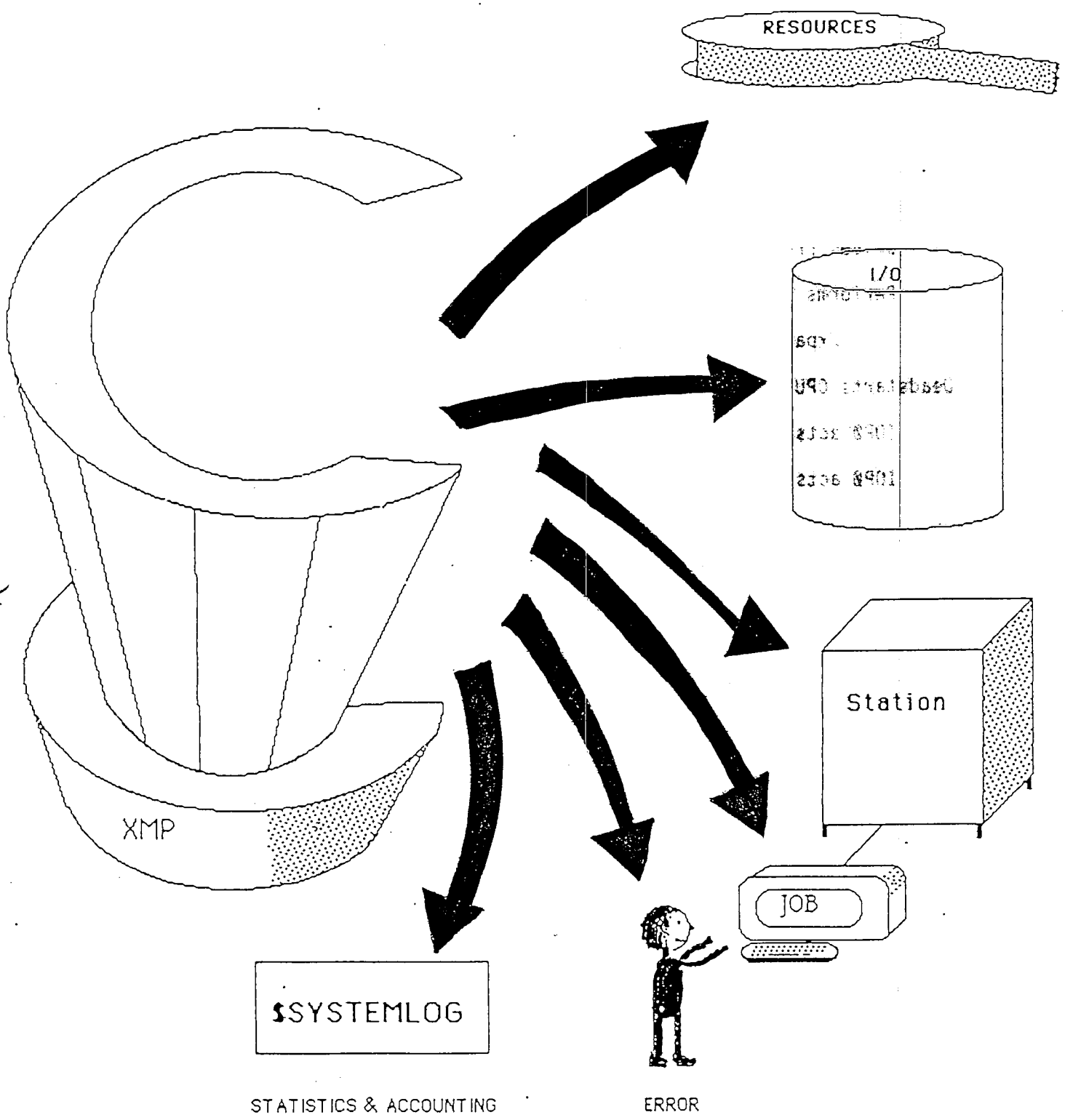
Accounting and Statistical Analysis

Types of Operating Systems:

Batch
- All job step processing is contained in the first file of the job.
- Many users submit their jobs and the computer executes these jobs without user interaction.
- Recognized by SUBMIT command.

Interactive
- Feature of Time Sharing.
- Every time the computer needs a command, it comes back to the programmer.
- Program has to wait for a response or action before it can continue.
- Recognized by an ! prompt.

COS FUNCTIONS



IOS FUNCTIONS

Directs hardware to efficiently perform its function

Acts as a slave to COS

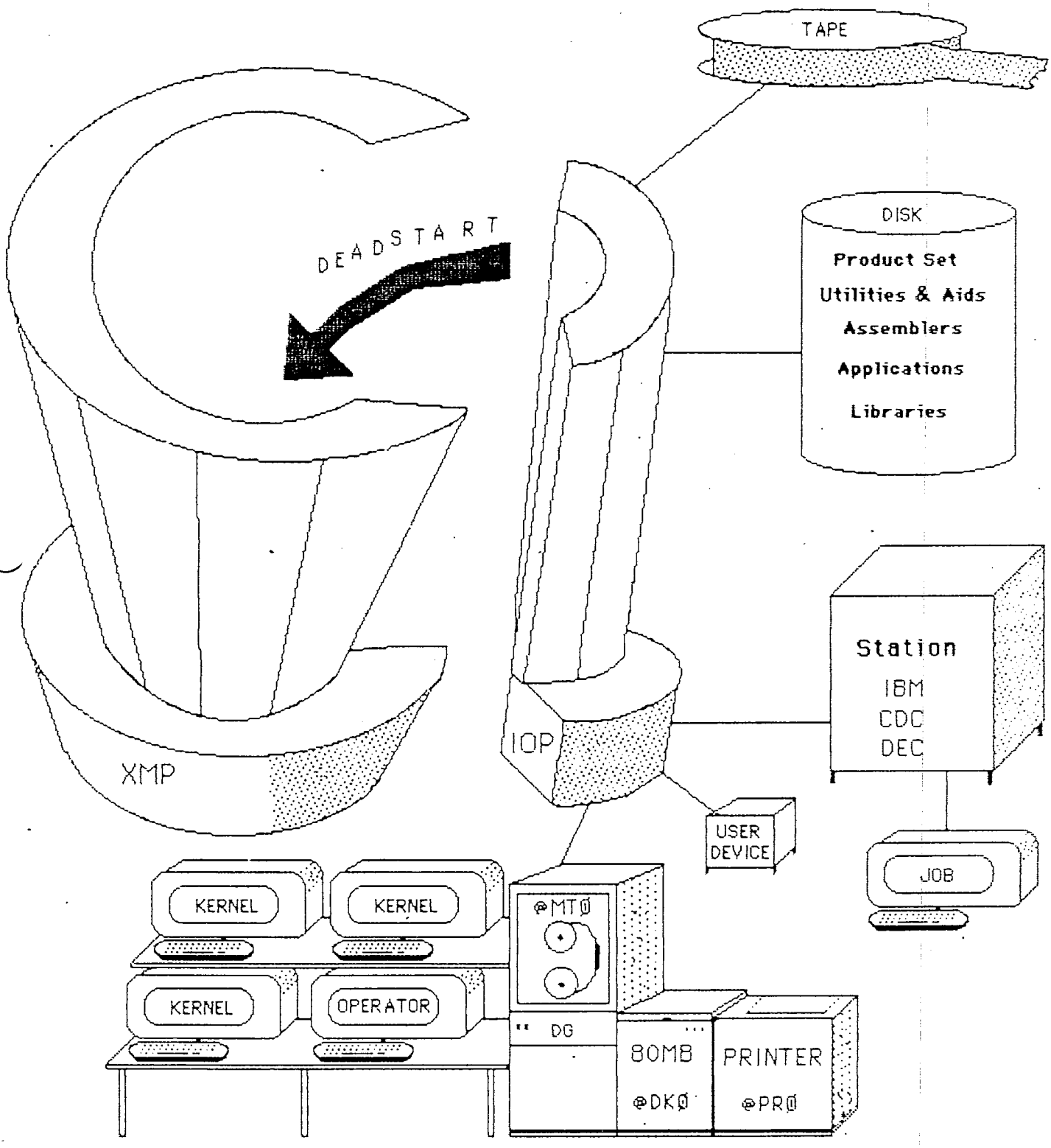
Performs IO between the CPU and peripherals attached to the IOP

- Drives the Disk Storage
- Drives Tape Storage System
- Drives Front End Channels
- Performs Master Operator Station functions
  - Expander Chassis and IOPØ

Deadstarts CPU

- IOPØ acts as the master processor
- IOPØ acts as maintenance control unit offline

IOS FUNCTIONS



COS SYSTEM COMPONENTS

Manages resources to provide a multiprogramming environment

Multiprogramming of user applications

Scheduling of applications by priority and job class

Manages disk and tape resources

Manages front-end communications

Manages file and program maintenance

Capable of modification at startup

Interrupt driven

COS is a collection of system programs.

Monitor (EXEC):

- Controls system
- Schedules next task or job for execution
- Performs I/O
- Accesses all memory
- Manages exchange package area
- Runs in non-interrupt mode

System Task Processor (STP):

- 15 tasks
- Processes all user requests
- Runs in interrupt mode

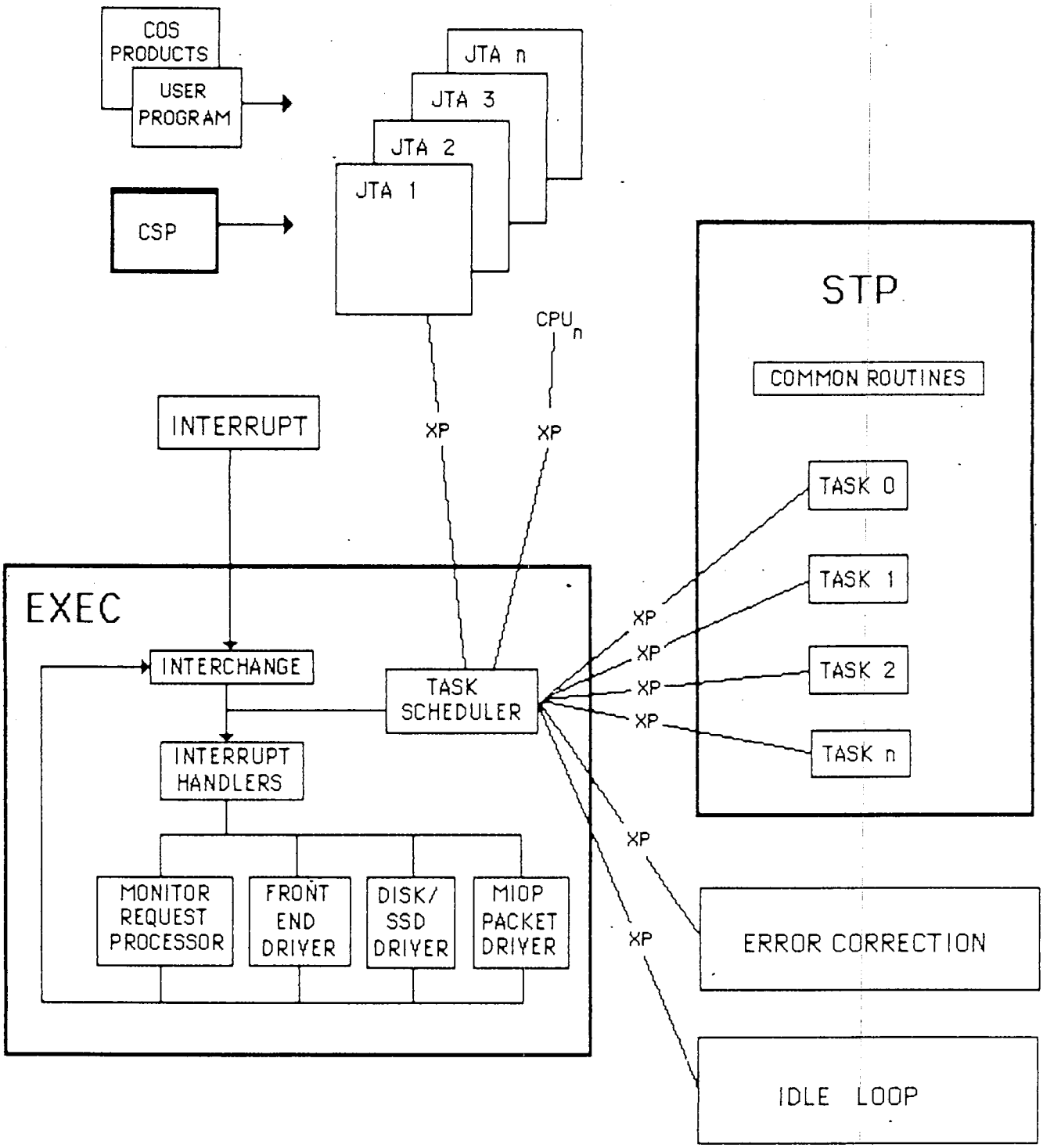
Control Statement Processor (CSP):

- Interprets all job control statements
- Calls systems tasks

Product Set

- Job Control Language
- Operational Aids and Utilities

COS COMPONENTS



EXEC

Runs in non-interrupt mode

BA=0

LA=I@MEM

Does privileged instructions

Has IO drivers (physical)

Handles interrupts

Exchange Processor

Entry point EN
Saves current RTC
Updates accrued CPU time
Interchange entry point is ENA
EXEC circles through Interchange until all flags are serviced and
no IO flags or interrupts are pending
Checks for time events
Calls IO handler if IO flag set
Calls appropriate interrupt handler
Entered from the scheduler before exchanging out of EXEC at EX

Interrupt Handlers

Clears flag
Processes channel error
Processes error exit breakpoints
IHT and CHT used for index address

Service Request Processor

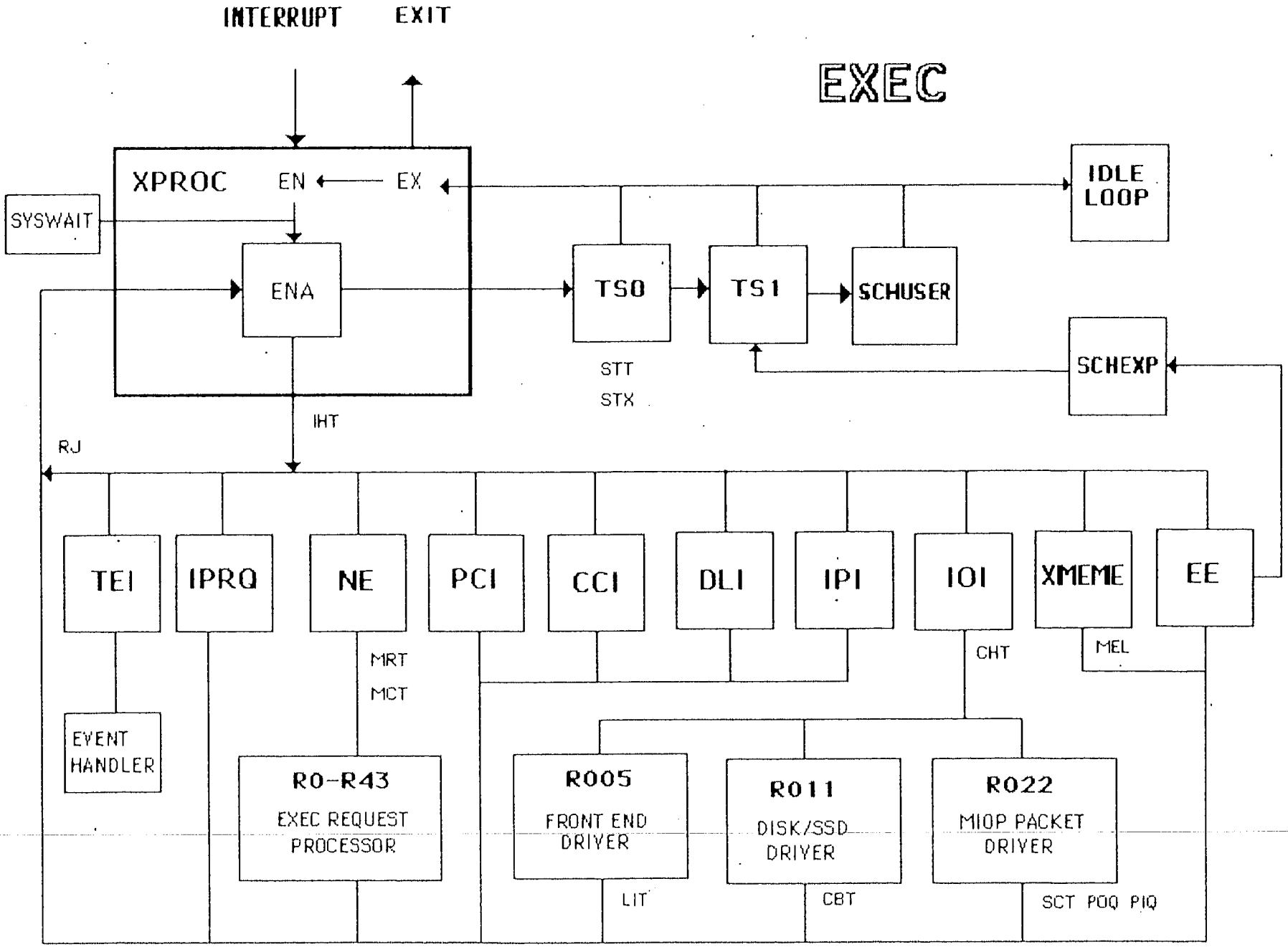
Privileged work for system monitor to do
Monitor calls - See SM-40, Page 2-17 for further detail.

Task Scheduler

Entered when all flags are processed
Entered on an Error Exit from a user
EXP will be exchanged to for processing the Error Exit
Schedules tasks by priority first
Schedules user after all tasks have been serviced
Schedules Idle Loop when nothing else is
Idle loop reads EXEC looking for memory errors
Error correction exchanged to on a memory error

Common Subroutines

Queue Control
Stop-Error Halt routine


INTERRUPT
EXIT
EXEC
XPROC
EN
EX
SYSWAIT
ENA
IDLE LOOP
TS0
TS1
SCHUSER
SCHEXP
STT
STX
IHT
RJ
TEI
IPRQ
NE
PCI
CCI
DLI
IPI
IOI
XMEME
EE
MRT
MCT
CHT
MEL
EVENT HANDLER
R0-R43
EXEC REQUEST PROCESSOR
R005
FRONT END DRIVER
R011
DISK/SSD DRIVER
R022
MIOP PACKET DRIVER
LIT
CBT
SCT POQ PIQ

EXEC REQUEST PROCESSOR

Performs essential services for tasks, in non-interruptable mode.

Process is as follows:

Task puts function number in S7
Task puts supplimental information in S6
Task does a normal exit (004)
Interchange sees the normal exit flag
NE clears the flag and uses S7 to index into Monitor Request Table
MRT contains the processor address

Requests exist to:

Create, ready or suspend a task
Enable Front-End Driver request
Enable Disk IO request
Connect and disconnect user to CPU
Post messages in the history trace table
Start and stop the operating system
Display or alter memory
Display or alter register or exchange packet
Set or clear system breakpoint
Report statistics

When an error is made by the processor an error code is in S6

EXEC REQUESTS

R000	null request
R001	create task
R002	ready task
R003	suspend task
R004	system hardware performance
R005	front-end transmission
R006	ready task after time delay
R007	reserved for installations
R010	change CPU status
R011	I/O request
R012	single bit error detection select
R000	unused
R014	ready task, suspend self
R015	initiate STP profile time slice
R016	request CPU
R017	disconnect CPU
R020	post data to circular trace buffer
R021	set memory size
R022	packet I/O request
R023	load and execute new copy of COS
R024	start system
R025	stop system
R026	task already created
R027	source ID mismatched
R030	bad function
R031	task parameter block changed
R032	invalid channel number
R033	clear system breakpoint
R034	CPU utilization report
R035	EXEC request report
R036	EXEC call count report
R040	channel interrupt count report
R041	switch to other CPU
R042	dump X-MP cluster to memory buffer

IOS SYSTEM COMPONENTS

KERNEL

Local memory resident

Executes in each I/O processor with minor modifications

Responsible for:
- Activity management
- Inter-activity communication
- Resource management
- Interrupt handling
- Inter-processor communication

OVERLAYS

Executable Programs or Subroutines
Reside in buffer memory
Read into local memory when needed
Makes up the bulk of the system
Re-entrant
Not all used by any one processor
An overlay table is maintained to provide information about the overlay
Operand Register %B contains the overlay's base address

The DISK input/output software moves data in streams between Central Memory in the mainframe and disks.

TAPE EXEC software processes requests from the Tape Queue Manager task in the mainframe. The Tape Exec performs functions related to tape I/O such as message routing, data formatting, data movement, and error recovery.

The INTERACTIVE STATION permits interaction with a job running in the mainframe. Interactive commands are entered at a console connected directly to the IOS.

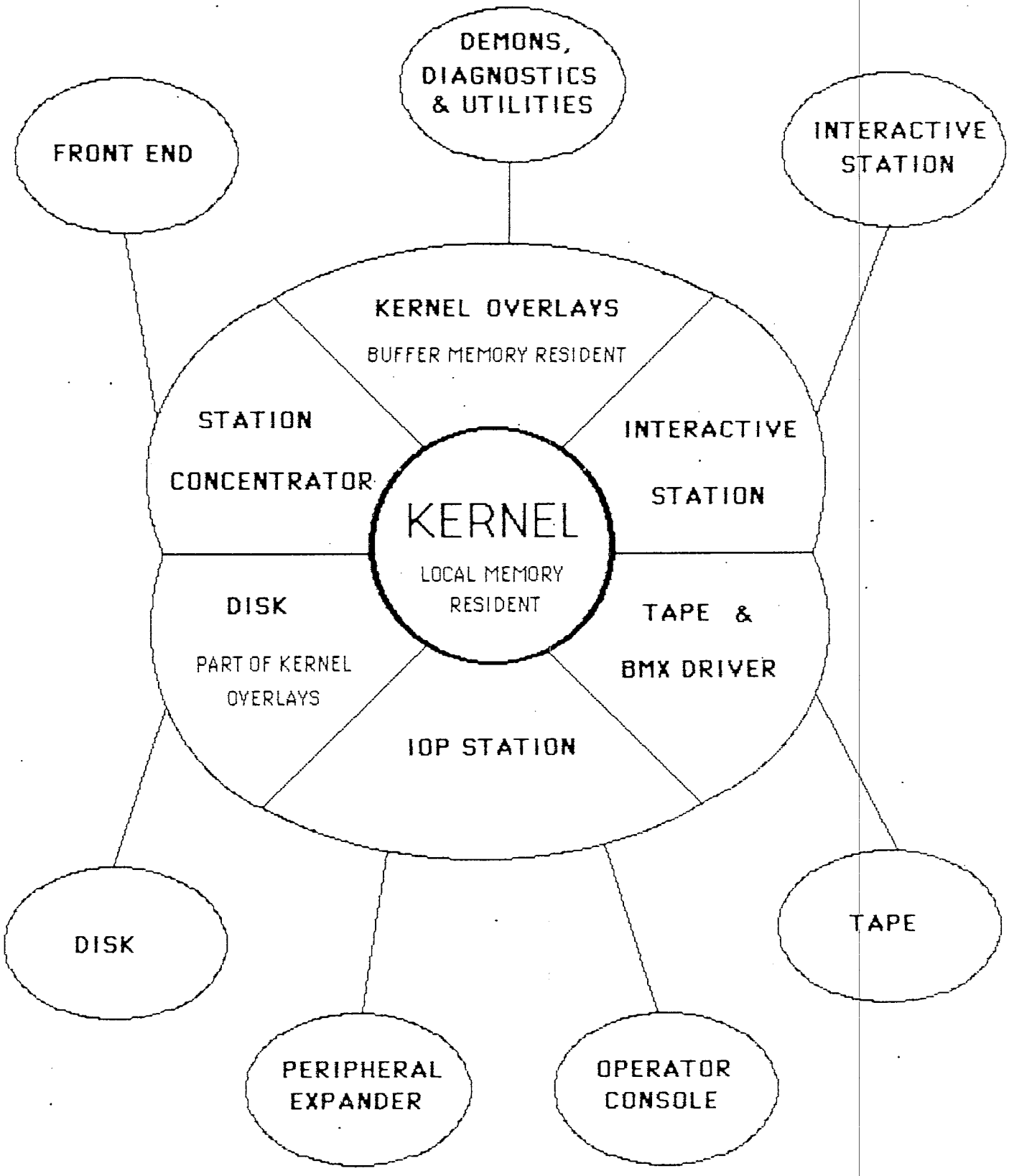
BLOCK MULTIPLEXER channel interface software drives the block multiplexer channel hardware. It contains device-independent command and interrupt code that executes requests from the Tape Exec.

STATION software runs in the Master I/O Processor (MIOP) and supports operator commands, station displays, and dataset staging. All dataset staging is performed without queuing; datasets are transferred directly to the mainframe.

CONCENTRATOR software accepts data from front-end computers into I/O Subsystem Buffer Memory, builds the data into a message, and sends it to the mainframe. The concentrator relieves the mainframe of the burden of handling an interrupt for each subsegment of messages transferred.

USER CHANNEL I/O software runs in the Master I/O Processor (MIOP) and supports access of I/O Subsystem channels by COS tasks. User Channels may be used for connecting new devices or mainframes to the I/O Subsystem.

IOS COMPONENTS



KERNEL

Local Memory Resident

Operand Registers 0-377

Unique to each Processor

Executes in non-interrupt mode

Interrupt Handler

Entered on a done flag set
No handler for high speed and buffer memory

Service Request Processor

Performs essential services for activities in non-interrupt mode
See SM-46, Page 2-11 to 2-38 for more detail

Activity Dispatcher

Transfers control from one activity to another
Swaps software stacks
Maintains overlay memory

Memory Control

Controls three local memory chains
Allocates buffer memory data, Software Stacks and DALs

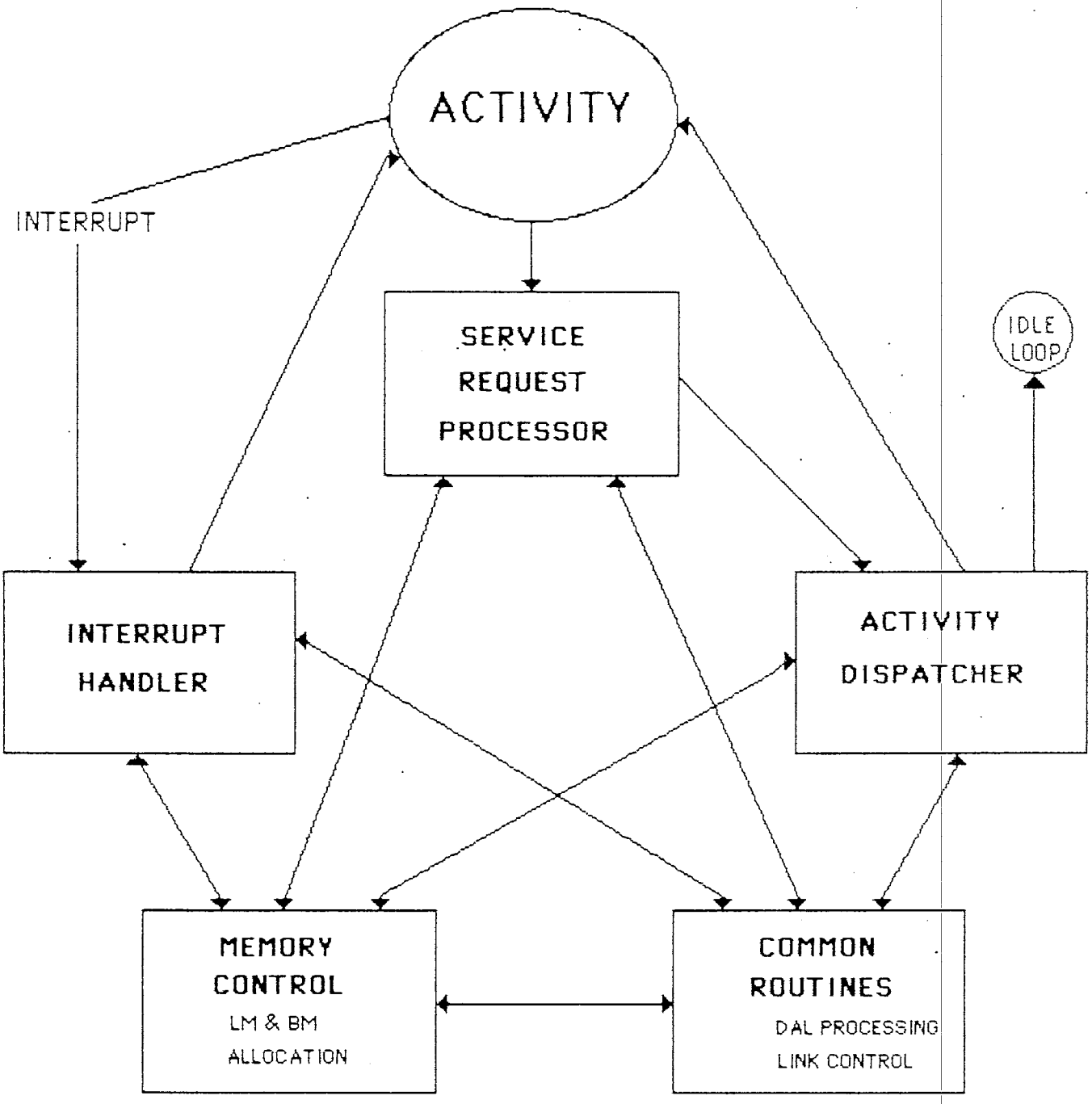
Common Subroutines

Setting up activity descriptor
Maintain queues
Maintain memory chains

Inter Activity Communication and Inter Processor Communication

Popcells, Disk Activity Links, A-A Messages

KERNEL


ACTIVITY
INTERRUPT
SERVICE
REQUEST
PROCESSOR
IDLE
LOOP
INTERRUPT
HANDLER
ACTIVITY
DISPATCHER
MEMORY
CONTROL
LM & BM
ALLOCATION
COMMON
ROUTINES
DAL PROCESSING
LINK CONTROL

SERVICE REQUEST PROCESS

Performs essential services for activities, in non-interruptible mode.

Activity calls a macro which passes parameters to another macro, which sets up parameters and does a return jump to service request process.

Service request process is as follows:

1. Lock out interrupts
2. Save A, B, E, and P in SMOD
3. Save specified operand registers in SMOD
   Save exit stack in SMOD if any registers saved
4. Get function code form RUNREG
5. Jump to address at FCTABLE + function code

Depending on function, control is passed to requester, kernel, or new overlay on completion.

Service Requests Exist for:

- Creating, rescheduling and terminating activities
- Passing control between overlays
- Locating an overlay in buffer memory
- Controlling push and timer queues
- Sending and receiving messages on CFT channels
- Sending responses to other IOPs
- Requesting another IOP to create or activate an activity
- Initiating front-end and block MUX I/O
- Sending messages to CPU and receiving a response
- Allocating and releasing local and buffer memory
- Moving data between buffer memory and central memory
- Flushing the overlay memory buffers

KERNEL REQUESTS

CODE	NAME	DESCRIPTION
1	PUSH	Put activity on a queue at priority.
2	POP	Remove activity from a queue and place it on CP queue at priority.
3	TERMINATE	Terminate an activity by releasing its AD and SMOD areas.
4	GIVEUP	Reschedule an active task by priority.
5	D4STIO	Initiates a Read or Write to a DD49.
6	D4SEEK	Initiates a Seek on a DD49.
7	PAUSE	Suspend an activity for tenths of a second.
11	TPUSH	Put activity on a queue and on a timer queue for tenths of a second.
12	GMDAL	Allocates MOS DAL.
13	RMDAL	Releases MOS DAL.
14	ASLEEP	Returns next popcell DAL. If none, push activity on popcell.
15	ALERT	Request another IOP to create an activity.
16	AWAKE	Request another IOP to activate an activity.
17	RESPOND	Send response to another IOP.
20	MSG	Send a message to a CRT.
21	MSGR	Send a message to a CRT and wait for response.
22	OUTPUT	Output a message to a CRT (station).
23	STATIO	Initiate I/O between a concentrator and a front-end.
25	RECEIVE	Input one character from a console.
26	GDAL	Allocates Local Memory DAL.
27	RDAL	Release Local Memory DAL.
30	GETMEM	Allocate local memory.
31	RELMEM	Release local memory.
32	BGET	Allocate a 512-word (4000 parcel) local buffer.
33	BRET	Release a 512-word local buffer.
34	SEND	Sends message to mainframe.
35	MGET	Allocate a 512-word MOS buffer.
36	MPUT	Release a 512-word MOS buffer.
37	OUTCALL	Calls an overlay in another IOP to execute once.
42	HSPR	Initiates a read on High Speed Channel.
43	HSPW	Initiates a write on High Speed Channel.
44	POLL	Send a message to the CPU.
45	TRANSFER	Move data between MOS and central memory.
46	MOSR	Read data from MOS to local memory.
47	MOSW	Write data from local to MOS memory.
50	CALL	Pass control to an overlay with return.
51	GOTO	Pass control to an overlay.
52	RETURN	Return control to an overlay.
53	FIND	Find MOS address and word length of an overlay.
54	FLUSH	Reinitialize overlay memory.
55	CREATE	Set up an independent activity and place it on a CPU queue.

KERNEL REQUESTS

CODE	NAME	DESCRIPTION
1	PUSH	Put activity on a queue at priority.
2	POP	Remove activity from a queue and place it on CP queue at priority.
3	TERMINATE	Terminate an activity by releasing its AD and SMOD areas.
4	DRIVER	Reschedule an active task by priority.
5	DMSTIO	Initiates a Read or Write to a DDMS.
6	DMSEEK	Initiates a Seek on a DDMS.
7	PAUSE	Suspend an activity for tenths of a second.
11	TPUSH	Put activity on a queue and on a timer queue for tenths of a second.
12	GMDAL	Allocates MOS DAL.
13	RMDAL	Releases MOS DAL.
14	ASLEEP	Returns next popcell DAL. If none, push activity on popcell.
15	ALERT	Request another IOP to create an activity.
16	AWAKE	Request another IOP to activate an activity.
17	RESPOND	Send response to another IOP.
20	MSG	Send a message to a CRT.
21	MSGW	Send a message to a CRT and wait for response.
22	OUTPUT	Output a message to a CRT (station).
23	STATIO	Initiate I/O between a concentrator and a front-end.
25	RECEIVE	Input one character from a console.
26	GDAL	Allocates Local Memory DAL.
27	RDAL	Release Local Memory DAL.
30	GETMEM	Allocate local memory.
31	RELMEM	Release local memory.
32	BGET	Allocate a 512-word (4000 parcel) local buffer.
33	BPET	Release a 512-word local buffer.
34	SEND	Sends message to mainframe.
35	MGET	Allocate a 512-word MOS buffer.
36	MPUT	Release a 512-word MOS buffer.
37	OUTCALL	Calls an overlay in another IOP to execute once.
42	HSPR	Initiates a read on High Speed Channel.
43	HSPW	Initiates a write on High Speed Channel.
44	POLL	Send a message to the CPU.
45	TRANSFER	Move data between MOS and central memory.
46	MOSR	Read data from MOS to local memory.
47	MOSW	Write data from local to MOS memory.
50	CALL	Pass control to an overlay with return.
51	GOTO	Pass control to an overlay.
52	RETURN	Return control to an overlay.
53	FIND	Find MOS address and word length of an overlay.
54	FLUSH	Reinitialize overlay memory.
55	CREATE	Set up an independent activity and place it on a CPU queue.

OVERVIEW

Match the following terms and their descriptions.

1. STATION	____ An interruptable COS program performing system functions
2. EXEC	____ 1 XMP CPU in monitor mode and executing EXEC at a time
3. KERNEL	____ Log of operating system and user activities
4. $SYSTEMLOG	____ Software that runs on the front end linking COS and the front end operating system
5. SINGLE THREADED	____ Priveledged work an overlay asks the IOS monitor to do
6. MULTITASKING	____ Priveledged work an STP Task asks the monitor to perform
7. TASK	____ IOS program running under Kernel control
8. KERNEL SERVICE REQUEST	____ Several CPU's working on the same job
9. EXECUTIVE REQUEST	____ Monitor for each IOP
10. OVERLAY	____ Monitor for the Mainframe COS

OVERVIEW

Match the following terms and their descriptions

1. STATION	___ An interruptable COS program performing system functions
2. EXEC	___ An XMP CPU in monitor mode and executing EXEC at a time
3. KERNEL	___ Log of operating system and user activities
4. SYSTEMLOG	___ Software that runs on the front end linking COS and the front end operating system
5. SINGLE THREADED	___ Privileged work an over ISP asks the IOS monitor to do
6. MULTITASKING	___ Privileged work an STP Task asks the monitor to perform
7. TASK	___ IOS program running under Kernel control
8. KERNEL SERVICE REQUEST	___ Identical CPU s working on the same job
9. EXECUTIVE REQUEST	___ Monitor for each JOP
10. OVERLAY	___ Monitor for the mainframe COS

System Programs

2

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon completion of this System Programs module, the learner should be able to:

1. Explain the process of generating COS and IOS.
2. List and explain the functions of 15 system tasks.
3. Define what an IOS overlay is.
4. List and explain the functions of 5 overlay subsystems.
5. Name a common overlay for each subsystem.
6. Locate a specified system program listing and explain its function.

SYSTEM GENERATION PROGRAM LIBRARIES

GENPL

Gen1	-	Generate Procedure Library
Gen2a	-	Generate JCL Binaries
Gen2b	-	Generate JCL Binaries
Gen3a	-	Generate COS Binaries
Gen3b	-	Generate IOS Binaries

COSPL

EXEC
TASKS
COSTXT

SNØØØ1 - Configuration Modifications for the Mainframe

IOPPL

KERNEL
OVERLAYS
$APTEXT

IOØØØ1 - Configuration Modifications for the I/O Subsystem

CAL BINARIES

There are some special macros used by CAL.

Macro definitions in program libraries must be assembled first.

$SYSTXT - User's CAL Macros

COSTXT - contains Global definitions for STP macros.

$APTEXT - contains Global definitions for IOS macros.

EXEC macros - defined at the beginning of EXEC listing.

Common Decks - Common Code used in Program Library.

- Defined by update directive *COMDECK
- Called from any regular or common deck.

```
JOB,JN=U1502A,T=800.
ACCOUNT,AC=265124,US=CRT,UPW=CRT.
*****************************************************************************************
*
*                        COS LISTINGS
*
ACCESS,PDN=SN0101,DN=CONFCOS,ID=V114BF1,OWN=SYSTEM.
ACCESS,DN=COSPL,ID=V114BF1,OWN=SYSTEM.
UPDATE,I=CONFCOS,P=COSPL,C=COSTXT,N=NEWCOS,Q=CT.
CAL,I=COSTXT,LIST,T=$COSTXT,S=0.L=0.
UPDATE,P=NEWCOS,N=0,I=0,C=LISTE,Q=EXEC.
CAL,I=LISTE,LIST,S=$COSTXT.
UPDATE,I=0,N=0,C=LISTT,Q=EXP:PDM.
CAL,I=LISTT,LIST,S=$COSTXT.
*****************************************************************************************
*
*                        IOS LISTINGS GENERATION
*
ACCESS,PDN=I00101,DN=CONFIOS,ID=V114BF1,OWN=SYSTEM.
ACCESS,DN=IOPPL,ID=V114BF1,OWN=SYSTEM.
UPDATE,I=CONFIOS,N=NEWIOS,C=APTXT,Q=AT.
APML,I=APTXT,T=$APTEXT,S=0.L=0.
UPDATE,I=0,N=0,C=LISTK,Q=K.
APML,I=LISTK,S=$APTEXT,LIST.
UPDATE,I=0,N=0,C=LISTO,Q=AMAP:ACOM:AMSG:BCOM:BEGIN:CALL:CDEM:CLI:CLOCK:CONCIO:^
CONSL:CRAY:CRAYIO:CRTDEM:DISK:DISPLAY:IAIOP:KEYBD.
APML,I=LISTO,LIST,S=$APTEXT.
```

TASK PROCESSOR CHARACTERISTICS

A system program to perform a system function.

Tasks have the following characteristics:

- Memory resident following EXEC
- Each task is a separate program loaded together
- Communicates with EXEC, each other and user jobs
- BA is end of EXEC 32000 (varies with release)
- LA is at the install parameter I@MEM
- Tasks operate in interrupt mode
- Each task has a priority (0-377)
- Each task has an ID (0-35)
- Task can be suspended by EXEC Requests
- A task may not suspend another
- Tasks are pre-emptable
- A task can create another task through an EXEC Request
- The Startup task creates the STP memory area

STP TASKS

Station Call Processor
ID 1 P 1

STaGer
ID 14 P 6

Disk Queue Manager
ID 5 P 2

Permanent Dataset Manager
ID 3 P 14

Job Class Manager
ID 12 P 12

Job ScHeduler
ID 11 P 13

user EXchange Processor
ID 2 P 11

Tape Queue Manager
ID 13 P 3

MEssage Processor
ID 7 P 10

log MesSaGe manager
ID 6 P 5

System Performance Monitor
ID 10 P 24

Flush Volatile Device
ID 15 P 15

Disk Error Correction
ID 4 P 20

startup STP - Z
ID 0 P 77

ISP Queue Manager
ID 16 P 4

EXEC

INTERRUPT HANDLERS

CHANNEL DRIVERS

EXEC REQUEST PROCESSOR

TASKS

Station Call Processor - SCP

- Monitors communication between CRAY and Front-End
- Passes staged datasets to stager
- Calls STG, DQM, JCM, and JSH

Stager - STG

- Manages dataset staging
- Moves segment buffer data to disk buffer data
- Used to be part of SCP

Disk Queue Manager - DQM

- Manages disk resources
- Performs IO request processing
- Allocation/Deallocation of disk space

Permanent Dataset Manager - PDM

- Performs permanent dataset functions
- Manages Dataset Catalog DSC

Job Class Manager - JCM

- Determines class of user job based on system resource needs
- Determines number of execution entries per job class

Job Scheduler - JSH

- Schedules user jobs by priority
- Performs memory management
- Controls rolling and recovery of job

Exchange Processor - EXP or UEP

- Processes all user job exits
- Receives all CSP user requests
- User makes requests to STP through EXP

Tape Queue Manager - TQM

- Manages allocation of tape drives
- Supplies tape related operator messages
- Manages tape IO request processing

Message Processor - MEP

- Relays error messages from EXEC to message task

STP ADDRESS POINTERS


```
0000000 000000000000000000002 000000000000000000000 000000000000000000000 17777777777777777777
0000004 000000000000000000000 000000000000000000000 000000000000000257244 00000000000000000313
0000010 000000000000000000000 000000000000000170752 000000000000000000001 00000000000000000003
0000014 000000000000000000003 000000000000000000010 000000000000000000000 04211400000000000012
0000020 000000000000000222262 000000000000000000000 000000000000000000000 000000000000000000000
0000024 000000000000000014152 000000000000000014516 000000000000000000000 000000000000000256122
0000030 000000000000000256140 000000000000000000000 000000000000000000000 04151724620061134304 64
0000034 030061136300631363406 5 030061164304611643246 5 002442357102074426 1120 000000000000000000001 01/03/85 01:11:55 "W
0000040 000000000000000000001 000000000000000000000 040525250000000065365 04150123046000002570 43
0000044 041516250000000026536 3 041523210000000016667 7 042101250000000012132 3 042103250000000016243 2 CNT J CSD DAT DCT
0000050 042521240000000014152 042122250000000016255 3 042130222000000022261 3 042521250000000025003 3 EQP JDRT kDXI % EQT P
0000054 042527250000000016665 6 043522250000000026427 3 044502250000000017067 7 044514250000000024715 7 EWT GRT h IBT ILT No
0000060 044523250000000024777 0 044520250000000025744 5 045123242411150022261 7 045130250000000017077 1 IST O IPT _%JSQBM % JXT
0000064 046111204000000032377 7 046103250000000022500 2 046111250000000022500 7 046130250000000022524 4 LIB LCT * LIT * LXT *
0000070 046523250000000022226 2 050104222000000022261 2 050104246000000016527 7 050110244000000025111 6 MST $ PDI % PDS PHR RN
0000074 050130250000000017075 2 050504250000000026000 6 051111250000000025734 5 051120230544000000505 PXT QDT RIT ^ RPLY E
0000100 051121246520000000046 5 051121250000000025312 0 051123250000000032354 3 051501206000000025612 2 RQST 5RQT VPRST cSAC \R
0000104 051504244000000015234 3 051504250000000006635 3 051521240000000025614 0 051524240000000016645 1 SDR SDT I SQP \`STP )
0000110 051530250000000016120 052104250000000026431 0 052121240000000014516 052130250000000021324 2 SXT PTDT h TQP NTXT
0000114 052504250000000024673 7 053120250000000024737 0 054101250000000765600 1 052122024150524600000 UDT M VPT N XAT \ TRACES
0000120 042121232520000025101 5 044524206520000026554 0 052121232520000041563 0 000000000000000000000 DQMT R ITCT k`TQMT
0000124 000000000000000000014 000000000000000000002 000000000000000000006 000000000000000000000
0000130 000000000000000000000 041124202420000000000 000000000000000267240 041106232405160000000 BTAD n BFMAN
0000134 000000000000000266640 041503236501310000000 0 000000000000000267340 041510202445162460000 0 m CCOPY n CHAINS
0000140 000000000000000267440 041510226515150000000 0 000000000000000267700 041514212405220000000 0 o CHKSM o CLEAR
0000144 000000000000000270000 041517234431112160000 0 000000000000000270100 041517234531242460000 0 p CONFIG p@CONVTS
0000150 000000000000000272440 041517240544000000000 0 000000000000000272640 041522202415132125100 0 u COPY u CRACKER
0000154 000000000000000272740 042103224541240000000 0 000000000000000273340 043116210451020000000 0 u DCJXT v FNDJB
0000160 000000000000000273500 043111260451302405100 0 000000000000000273440 043505250475272340000 0 w@FIXJXPR w GETOWN
0000164 000000000000000273540 043505250501012446400 0 000000000000000273600 043524232425150000000 0 w`GETPARM w GTMEM
0000170 000000000000000274000 043524250415020000000 0 000000000000000274100 043524250541240000000 0 x GTTCB x@GTTXT
0000174 000000000000000274200 044523240415172320000 0 000000000000000510600 044512250515252040000 0 x ISPCOM IJTSUB
0000200 000000000000000274300 045115212464000000000 0 000000000000000275000 045124202421162500000 0 x JMEM z JTADNT
0000204 000000000000000275140 046107232515070000000 0 000000000000000275240 046523216505252120000 0 z`LGMSG z MSGQUE
0000210 000000000000000275340 050101206455052505040 0 000000000000000275600 050127212471030000000 0 z PACKETQ ¦ PWENC
0000214 000000000000000276000 050515246434000000000 0 000000000000000276040 050525212525052460000 0 ¦ QMSG ¦ QUEUES
0000220 000000000000000276440 051105242511202305440 0 000000000000000276740 051114232425150000000 0 } REQRPLY } RLMEM
0000224 000000000000000277700 051114250541240000000 0 000000000000000300000 051504144501040000000 RLTXT SD2PD
0000230 000000000000000300100 051524240421012505140 0 000000000000000300340 051524240425222400000 @STPDATS STPERR
0000234 000000000000000300540 051524240465052320000 0 000000000000000300740 051524240521112324240 0 `STPMEM STPTIME
0000240 000000000000000301240 051524240525242246000 0 000000000000000301540 052504206475150000000 0 STPUTIL `UDCOM
0000244 000000000000000302740 053101210000000000000 0 000000000000000304540 052101246454401402004 0 VAD `TASK 0
0000250 051524240000000304600 055000000000000512440 0 055114236434000651000 052101246454401422004 0 STP Z ZLOG R TASK 1
0000254 051503240000000305100 052101246454401442004 0 052505240000000323540 041511236000000036010 0 SCP @TASK 2 UEP `CIO @
0000260 052111236000000036357 5 052101246454401462004 0 050104232000000046504 0 052101246454401502004 0 TIO ¦TASK 3 PDM J TASK 4
0000264 042105206000000046474 0 052101246454401522004 0 042121232000000045704 0 052101246454401542004 0 DEC ¦ TASK 5 DQM ^ TASK 6
0000270 046523216000000044010 0 052101246454401562004 0 046505240000000050014 0 052101246454401602004 0 MSG @@TASK 7 MEP `TASK 8
0000274 051520232000000050050 0 052101246454401622004 0 045123220000000044160 0 052101246454401423004 0 SPM @TASK 9 JSH C TASK 10
0000300 045103232000000045336 6 052101246454401423044 0 052121232000000036464 0 052101246454401423104 0 JCM V TASK 11 TQM TASK 12
0000304 051524216000000050110 0 052101246454401423144 0 043126210000000050274 0 052101246454401423204 0 STG @TASK 13 FVD TASK 14
0000310 044521232000000050364 0 025040246521201004251 6 042123100405240055546 7 051531140300661002644 0 IQM * STP ENDS AT 7SY006 -
0000314 052101246454401003104 0 044125234434402025204 0 031463146300601564104 0 024040100300601542006 0 TASK 2 HUNG AT 3330078 ( 006 0
0000320 030063144324621443050 2 020040100304601403004 0 030060140304661423046 1 020040100425222444051 6 03252218 1000 00016111 ERRAN
0000324 020115202415222362440 0 041060100364401002004 0 000000000000001554035 052101246454402224212 3 MACRO) B0 = TASK IDS
0000330 000000000000000000000 000000000000000000001 000000000000000000002 000000000000000000003
0000334 000000000000000000004 000000000000000000005 000000000000000000006 000000000000000000007
0000340 000000000000000000010 000000000000000000011 000000000000000000012 000000000000000000013
```

Log Manager - MSG

Writes messages from system and user to $SYSTEMLOG or user log
Analyzed by Extract or HERG

System Performance Monitor - SPM

Collects system performance statistics
Records performance statistics in $SYSTEMLOG
Sends information to Log Manager

Flush Volatile Device - FVD

Backs up volatile storage such as buffer memory and SSD

ISP Queue Monitor - IQM

Gives COS access to datasets or IO devices supported under MVS.
Eliminate the need to stage large front-end datasets to Cray disk.

Disk Error Correction - DEC

Used only on DCU 2/3 Disks
Attempts error correction
Called by DQM
NOT assembled if no DCU 2/3

Startup - Z & STP

Initiates tasks and system tables
Initiates system devices
Controls recovery of jobs and datasets
Relinquishes memory when complete

3 parts

Install - All mass storage is assumed vacant and COS is started as if for the first time.

Deadstart - Permanent datasets are recovered and COS is started as in after a normal shutdown.

Restart - Operator option after a system interruption to recover the IO and jobs.

Z	SCP	STG	DQM	PDM	JCM	JSH	DQM	DEC	EXP	MEP	MSG	SPM	IQM	FVD
AUT	-AUT	PDD	-DAT	DAT	-CSD	CSD	(DDL)	EQT	CNT	-AEM	AUT	CSD	ILT	EQT
CNT	-IBT	SDT	DCT	CSD	SDT	-JXT	-(DNT)		-DEX		(DSP)	DCT	VPT	DRT
DAT	-LCT	-SST	(DNT)	(DNT)		-MST	-(DSP)		(DNT)		JTA	+MCT	VCT	
(DNT)	-LIT		-DRT	DRT		-RJI	JXT		(DSP)		JXT	+ICT	TRB	
DRT	-LXT		(DSP)	-DSC		SDT	-LFT		-DUX		-LGJ	+STT	IST	
-DSC	PDD		-EQT	(DSP)		-TXT	-ODN		-FSH		PDD			
(DSP)	-SDT		GRT	DXT		JTA	(PDD)		GRT		SDT			
-DVL	-STT		JXT	EQT			SDT		JXT					
DXT			-RQT	JCB			-SWT		-LDT					
-EFT			JTA	JTA			TCB		-SM					
-EQT			PHR	JXT			-UPT		-VAX					
GRT			SQP	PDD			-JTA		-VUX					
JTA			TXT	-PDI			-TXT		JTA					
JXT				-PDS			IDD							
(ODT)				-QDT			IRT							
PDI				SDT			TRB							
QDT				-XAT										
-RJI														
SDT														
TCB														
TDT														
TXT														

PACKET DRIVER	FED DRIVER	DISK/SSD DRIVER
+SCT	+CHT	DCT
+PIQ	+CXT	EQT
+POQ	LIT	+CHT
+CHT	LXT	+CAT
+CAT	LCT	+CLT
+CLT	+CAT	+CIT
+CIT	+CLT	+CBT
CXT	+CIT	+PHR
		+SAC
		+SQP

USER		CSP
(BGN)	(LDT)	(DSP)
(BAT)	(OPT)	(JCB)
(DDL)	(JBI)	(LFT)
(DSP)	(DRPB)	(DNT)
(DNT)	(RCB)	
(JAC)	(NCB)	
(JCB)	(MHB)	
(LFT)	(IJPB)	
(ODN)		
(PDD)		
(JST)		

WHAT IS AN ACTIVITY?

Work for IOP to do:

- Drive operator console
- Pass a dataset

Interruptible program that runs on the IOP

Consists of one or more nested overlays

High priority activities are demons

OVERLAY CHARACTERISTICS

Executable program or subroutine that resides in buffer memory

Read into local memory when needed for activation

Overlay table contains an entry for each defined overlay name

An overlay may be called from the kernel console or another overlay

Overlays are read only and deallocated when no longer being used

Groups of overlay types make up a subsystem

Completely re-entrant

Kernel maintains an operand register (%B) containing the overlay base address

Overlay Format

Field	Parcel	Bits	Description
OV@NAM	0-3	0-15	Overlay name (up to eight ASCII characters)
OV@TYP	4	0	Type of overlay: 0 Executable 1 Data
OV@NUM	4	1-15	Overlay number
OV@PAR	5	0-15	Parameter information:
SM@NUM	-	0-6	Number of registers
SM@FST	-	7-15	First operand register
OV@ENT	6	-	Entry point (first executable statement)

```
DMEM,TYPE=IOP1,FWA=0,LWA=177777,FORMAT=PARCEL,R.                FDUMP 1.13      07/06/84     13:38:17
                                                                SYSDUMP         05/04/84     16:12:25

000024360  102006  020152  022154  024152    027155  071006  020152  022153
000024370  011177  024154  020151  012037    024001  030001  024152  020152
000024400  012003  024001  030001  004011    024152  020152  023154  133064
000024410  000026  001000  000000  000000    046504  024424  035434  000000
000024420  026654  024424  020003  016000    046504  026654  024414  000002
000024430  024414  027634  006264  026423    044123  050124  042523  000000
000024440  000071  000430  010000  024430    024431  024432  024433  024434
000024450  024435  024445  024441  024460    014000  005575  024410  010000
000024460  034410  020107  102002  070032    010001  024452  020452  013003
000024470  100003  102002  070023  010015    024464  020452  024465  014000
000024500  000071  024466  010464  054000    014000  034430  076011  020006
000024510  012013  024410  030410  026452    071026  020445  103033  010030
000024520  024464  010074  024465  010000    024466  010000  024467  010000
000024530  024470  010445  024471  010464    054000  010000  076011  103002
000024540  070010  010004  024464  010464    054000  014000  034430  076011
000024550  071033  020445  024416  010074    024415  020415  102006  010000
000024560  034416  027415  026416  071006    020445  012050  024446  020162
000024570  102002  070012  014000  002115    024444  077003  001341  077003
000024600  001633  075003  001273  020441    103023  010032  024464  010441
000024610  024465  010464  054000  010000    076011  103002  070010  010004
000024620  024464  010464  054000  014000    034430  076011  071023  020460
000024630  102002  070006  020441  024460    010000  024441  071033  077003
000024640  001633  014000  001730  024444    077003  001341  077003  001633
000024650  077003  001633  014000  001744    024444  077003  001341  077003
000024660  001633  014000  001757  024444    077003  001341  077003  001633
000024670  014000  001767  024444  077003    001341  077003  001633  014000
000024700  001774  024444  077003  001341    077003  001633  014000  002001
000024710  024444  077003  001341  077003    001633  077003  001633  014000
000024720  002006  024444  077003  001341    077003  001646  030446  004010
000024730  024423  020423  013061  102002    070010  010001  024430  010001
000024740  024431  010001  024432  070040    020423  013062  102002  070006
000024750  010001  024430  010001  024431    070027  020423  013063  102002
000024760  070004  010001  024430  070020    020423  013064  102002  070004
000024770  010001  024431  070011  020423    013065  102002  070004  075003
000025000  001273  070002  071143  014000    002015  024444  077003  001341
000025010  077003  001646  010000  024450    077003  001456  107013  030445
000025020  024453  020445  012001  024410    030410  024454  020453  013040
000025030  100003  102002  071027  077003    001456  107032  030445  024455
000025040  020445  012001  024410  030410    024456  020455  013040  100003
000025050  102002  071046  020453  023455    102002  070006  020454  023456
000025060  102002  070002  071057  020455    023453  024423  004020  024417
000025070  020456  023454  024424  101002    027423  004020  021417  106074
000025100  077003  001633  014000  002034    024444  077003  001341  077003
000025110  001646  030446  004010  024423    020423  013114  102002  070004
000025120  010001  024433  070006  020423    013123  103002  070002  071027
000025130  014000  002051  024444  077003    001341  077003  001646  030446
000025140  004010  024423  020423  013102    102002  070004  010001  024434
000025150  070006  020423  013127  103002    070002  071025  020432  102027
000025160  014000  002076  024444  077003    001341  077003  001646  030446
000025170  004010  024423  020423  013131    102002  070004  010001  024435
000025200  070006  020423  013116  103002    070002  071027  020453  024437
000025210  020454  024440  077003  001703    020455  023437  024423  004020
000025220  024417  020456  023440  024424    101002  027423  004020  021417
000025230  020423  103006  020424  017000    001000  101002  070005  014000
000025240  001000  024457  070003  020424    024457  020434  103002  070004
000025250  020457  024436  070003  010001    024436  020436  005002  024423
000025260  020457  005002  024424  020424    023423  024442  020457  023436
```

DISK SUBSYSTEM

Moves data between central memory and disk.

Performs disk error recovery.

COS must initiate I/O by making a disk request.

COS is responsible for device assignments and dataset allocations.

Part of the disk code is within the Kernel.

DISK INTERRUPT ANSWERING

Entered when a disk channel interrupts.

Initiates next I/O and schedules DISK demon or disk error recovery.

Executes in non-interruptible mode.

Allocates local memory buffers on reads.

Deallocates local memory buffers on write.

DISK DRIVING ROUTINES

Performs most of the physical I/O.

Used by disk interrupt answering, DISK demon, ACOM demon, and disk error recovery.

Executes in non-interruptible mode.

Routines available for:

- Selecting head and cylinder
- Setting up disk buffers
- Building and sending a DAL
- Initiating disk I/O

DISK DEMON ACTIVITY

Nucleus of the disk subsystem.

Evaluates requests pending on a disk channel's done queue.

Builds EDALS as needed.

In DIOP transfers data between local and buffer memory.

In BIOP transfers disk data over high-speed channel.

Creates disk error recovery activity if necessary.

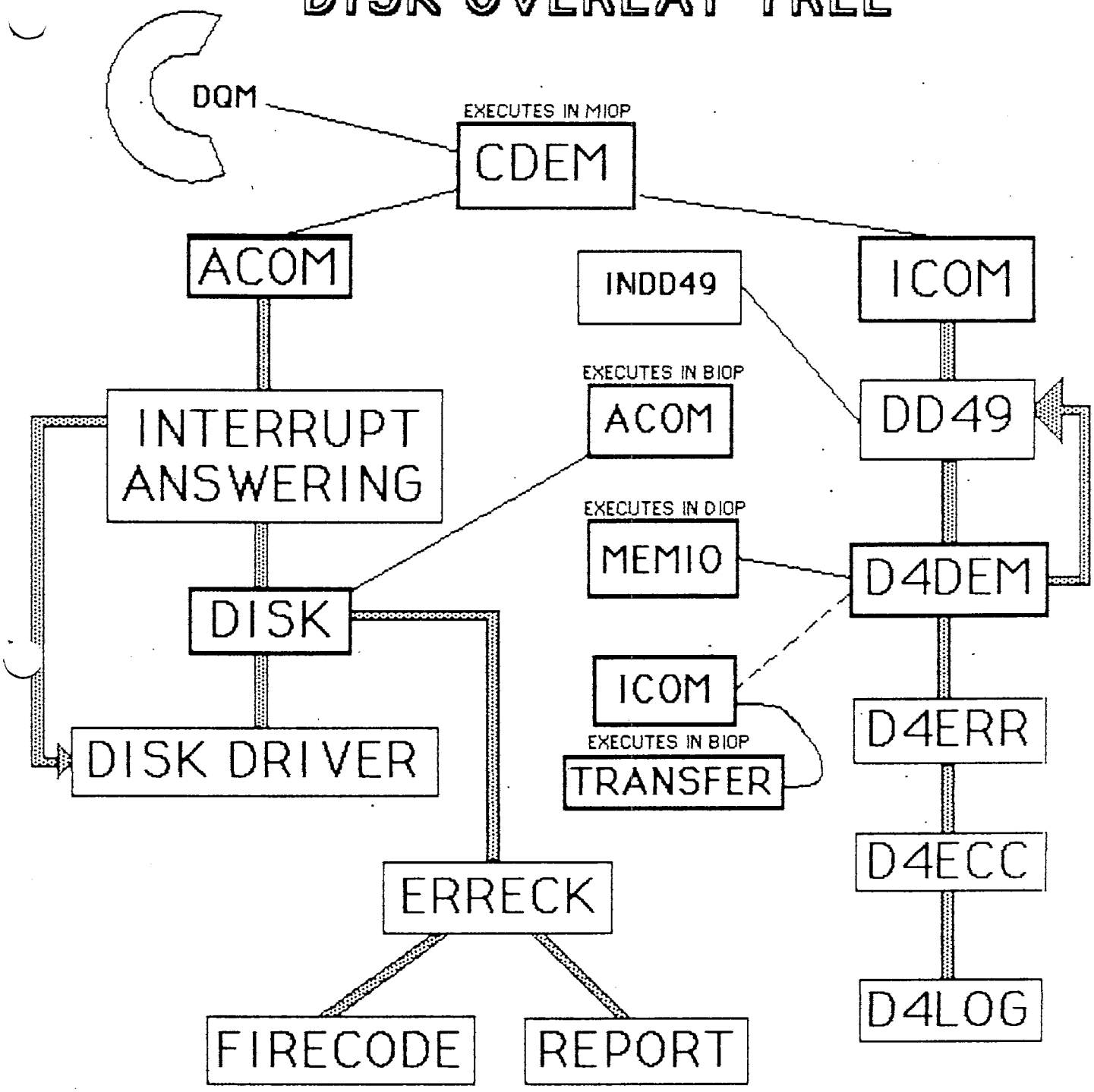
Usually activated by disk interrupt answering.

Executes often in non-interruptible mode.

Resides in buffer memory as disk overlay.

Sends EDAL with done status to MIOP.

DISK OVERLAY TREE



STATION SUBYSYSTEM

Provides a means for operator-COS communication.
    Controls operator consoles.
May be used as a batch job entry station.
    Jobs or datasets staged from tape.
May accept COS output and distribute it to mag tape or printer.
Allows on-line debugging of COS.

CHARACTERISTICS

Executes mostly in MIOP with some high speed transfers through BIOP.
    May have more than one station active at a time.
    Each station must have a dedicated console.
    They must share the expander peripherals.
    Two or more consoles may be supported by one station.
Appears to be just another front-end station to CPU.

MAIN COMPONENTS

STATION Overlay:
    Initializes a station when 'STATION' is typed in at the MIOP KERNEL console.
    Initiates one set of station console handling activities:
        KEYBD, CLI, and DISPLAY

KEYBD Overlay:
    Receives characters entered at the station console keyboard.
    Calls the CONSL overlay to echo the characters.
    Activates the CLI activity to process commands.

CLI Overlay:
    Interprets and executes the operator commands.
    Gets commands from a circular buffer filled by the KEYBD activity.
    Validates them, and calls appropriate overlay to process them.

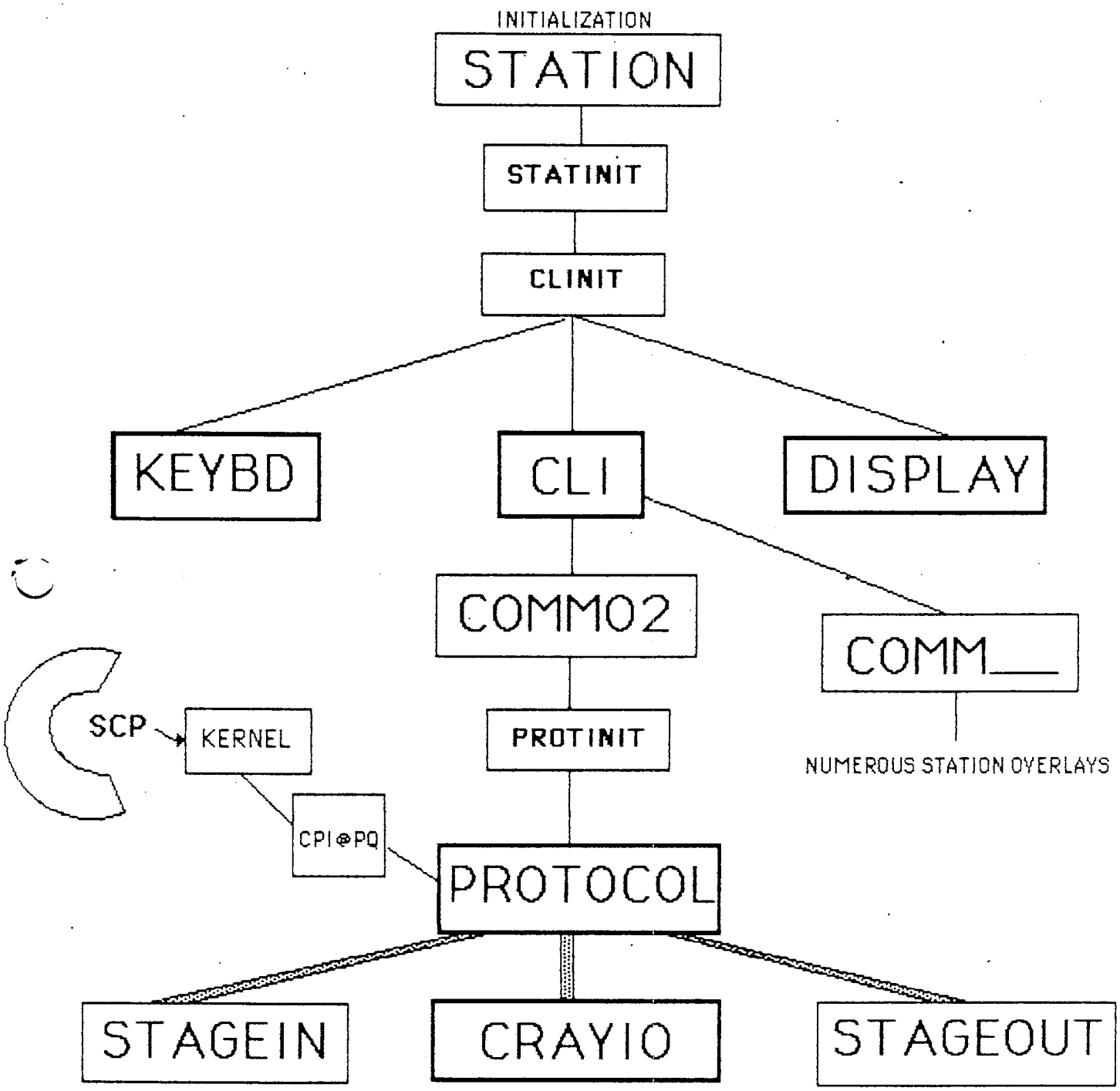
DISPLAY Overlay:
    Formats the operator display.
    Responds to requests from CLI and calls appropriate display overlay.

PROTOCOL Overlay:
    Manages station-COS communications for all active stations.
    Initiated by LOGON command.
    Terminated by LOGOFF or communication breakdown.
    Responsible for:
        Generating message sent to COS.
        Validating COS responses.
        Maintaining stream states.
        Creating activities to manage dataset transfers.
        Scheduling messages to COS.
        Distributing COS responses.

STAGEIN Overlay:
    Stages a dataset from the I/O subsystem to COS.
    Created by protocol activity.

STAGEOUT Overlay:
    Stages a dataset from COS to the I/O subsystem.
    Created by the protocol activity when CPU initiates staging on an output.

STATION OVERLAY TREE



CONCENTRATOR SUBSYSTEM

Allows apparent direct communication between the CPU and a front-end.

Looks like a CRAY channel pair to front end.

Thus no changes necessary to existing front-end stations.

May reduce the number of interrupts to the CPU per front-end message.

CHARACTERISTICS

Resides in buffer memory as overlays.

Executes mostly in MIOP with high speed transfers to CPU through BIOP.

One active concentrator for each front-end channel pair.

May have several logical IDs logged on to one concentrator (through same channel pair).

Each ID may have a different segment size.

Controlled via 'CONC' and 'ENDCONC' kernel console commands.

MAIN COMPONENTS

CONC Overlay:
Initializes concentrator resources.
Creates CONCIO activity.

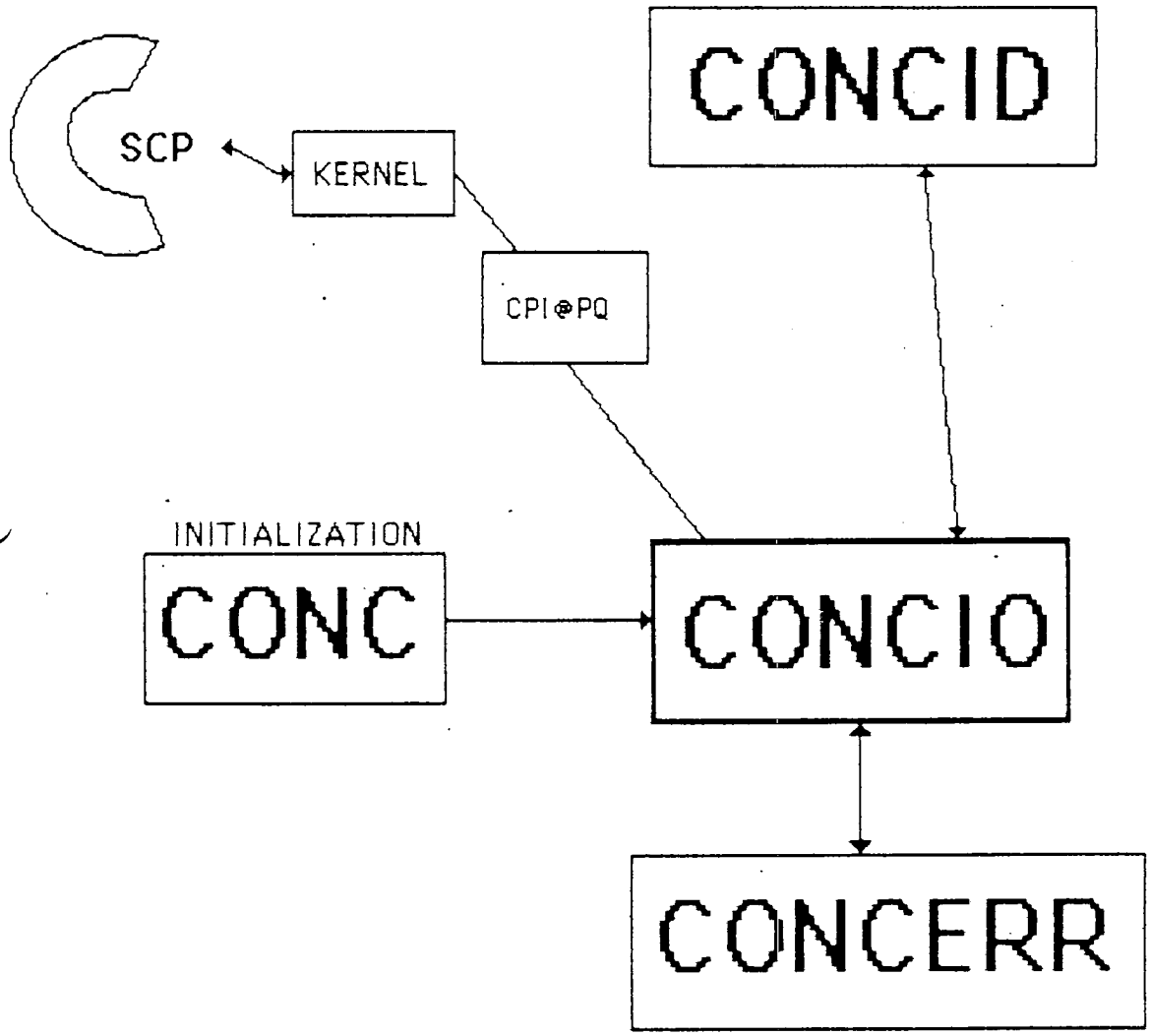
CONCIO Overlay:
Concentrator IO routines.
Accepts messages from front end and writes to buffer memory.
Sends message to front end reading from buffer memory.

CONCID Overlay:
Locate ID based table entries.
Termination processing.

CONCERR Overlay:
Handles concentrator hardware and software errors.

CRAYIO Overlay:
Gets central memory addresses for messages via channel extension table.
Does a STATIO request to BIOP to move LCP or Message to or from CM and BM.

STATION CONCENTRATOR OVERLAYS



NSC HYPERCHANNEL SUBSYSTEM

Links the station concentrator software to the JSC Hyperchannel Adapter.

CHARACTERISTICS

Resides and executes in MIOP.

Uses signal watch macros to talk to CONCIO.

MAIN COMPONENTS

NSC Overlay:
Initializes NSCIO Activity

NSCIO Overlay:
IO Control and Hyperchannel driver

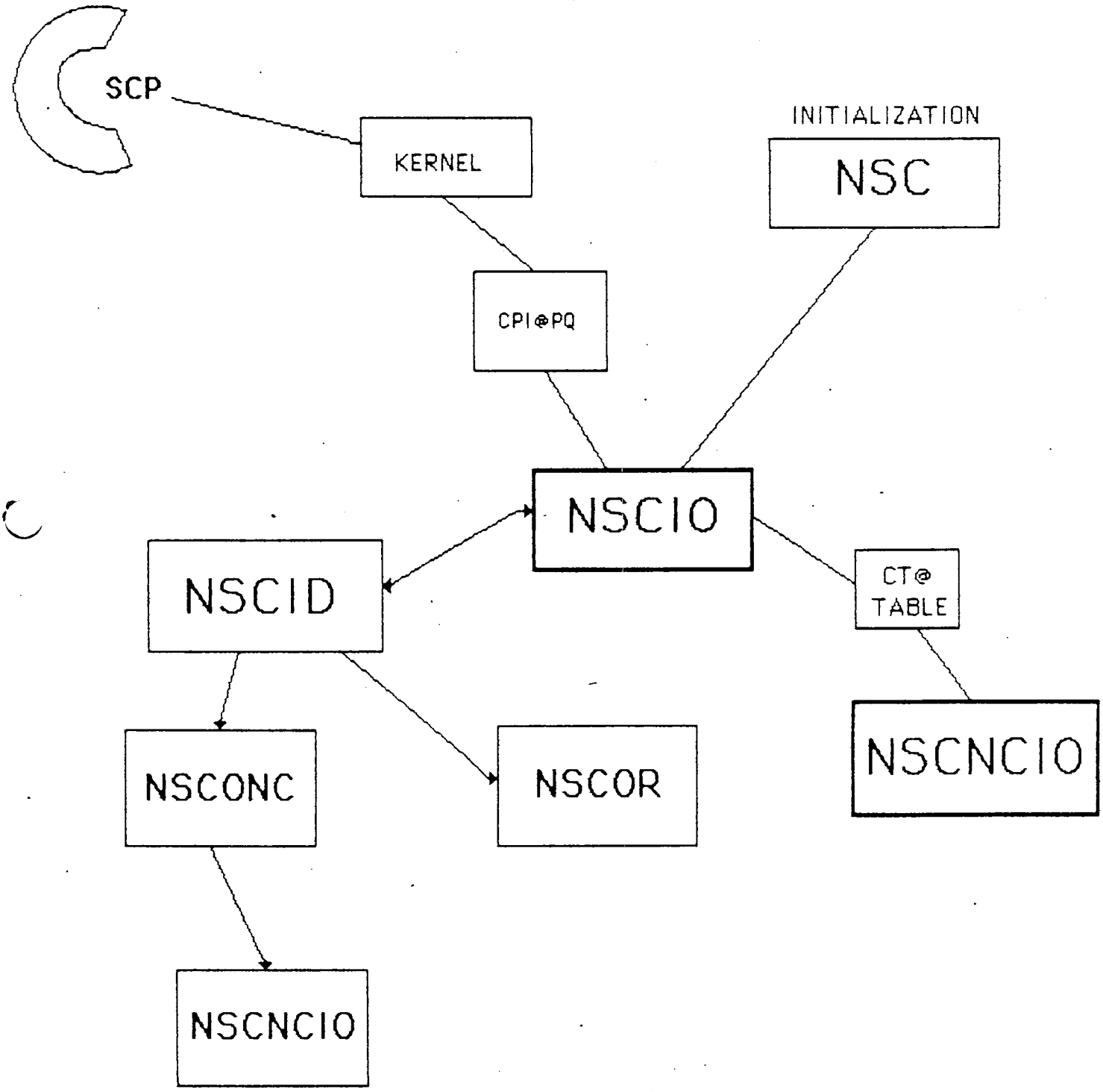
NSCID Overlay:
Logical IO processor for CXT

NSCOR Overlay:
Ordinal assignment

NSCONC Overlay:
NSC concentrator initialization

NSCNCIO Overlay:
NSC concentrator IO activity

NSC HYPERCHANNEL OVERLAY TREE



INTERACTIVE STATION SUBSYSTEM

Allows analyst to use COS interactive facility.

Access to TEDI and JCL statement control.

CHARACTERISTICS

Resides in buffer memory as overlays.

Executes mostly in MIOP with high-speed transfers to CPU through BIOP.

May support several consoles.

Consists of two parts:

- Interactive concentrator
- Interactive console

MAIN COMPONENTS

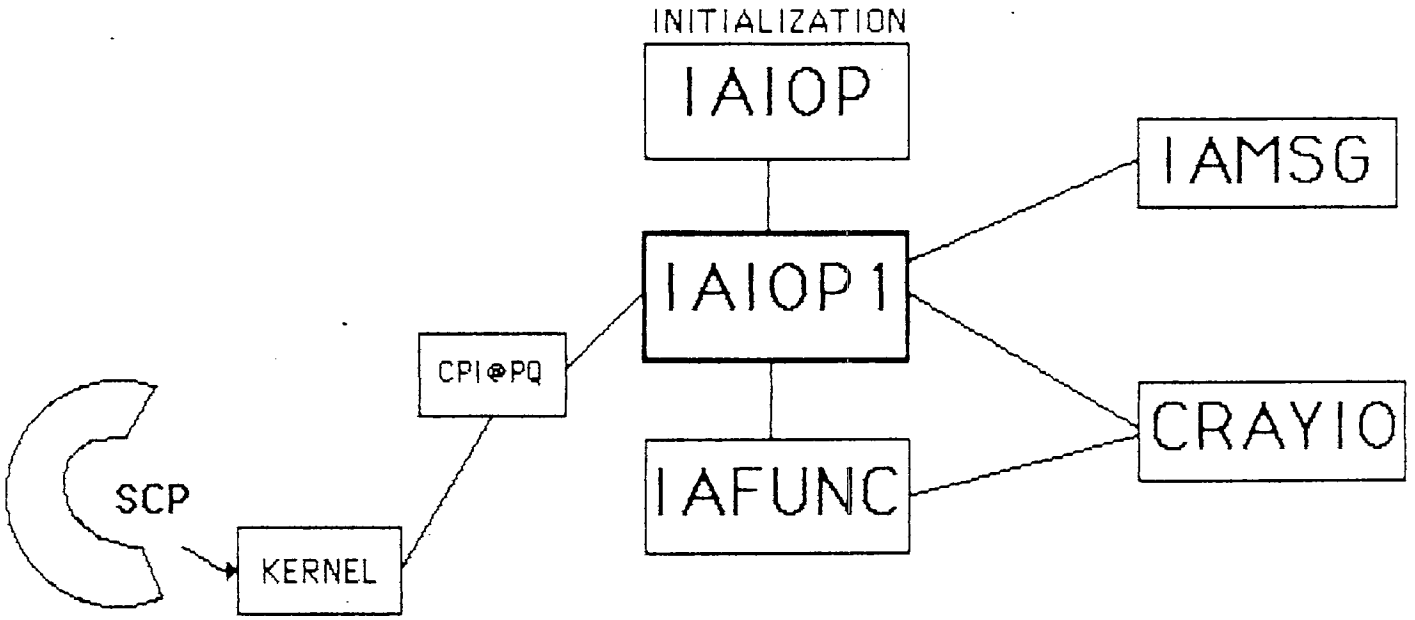
IAIOP Overlay:
Initializes IOP Station Concentrator and interprets commands.

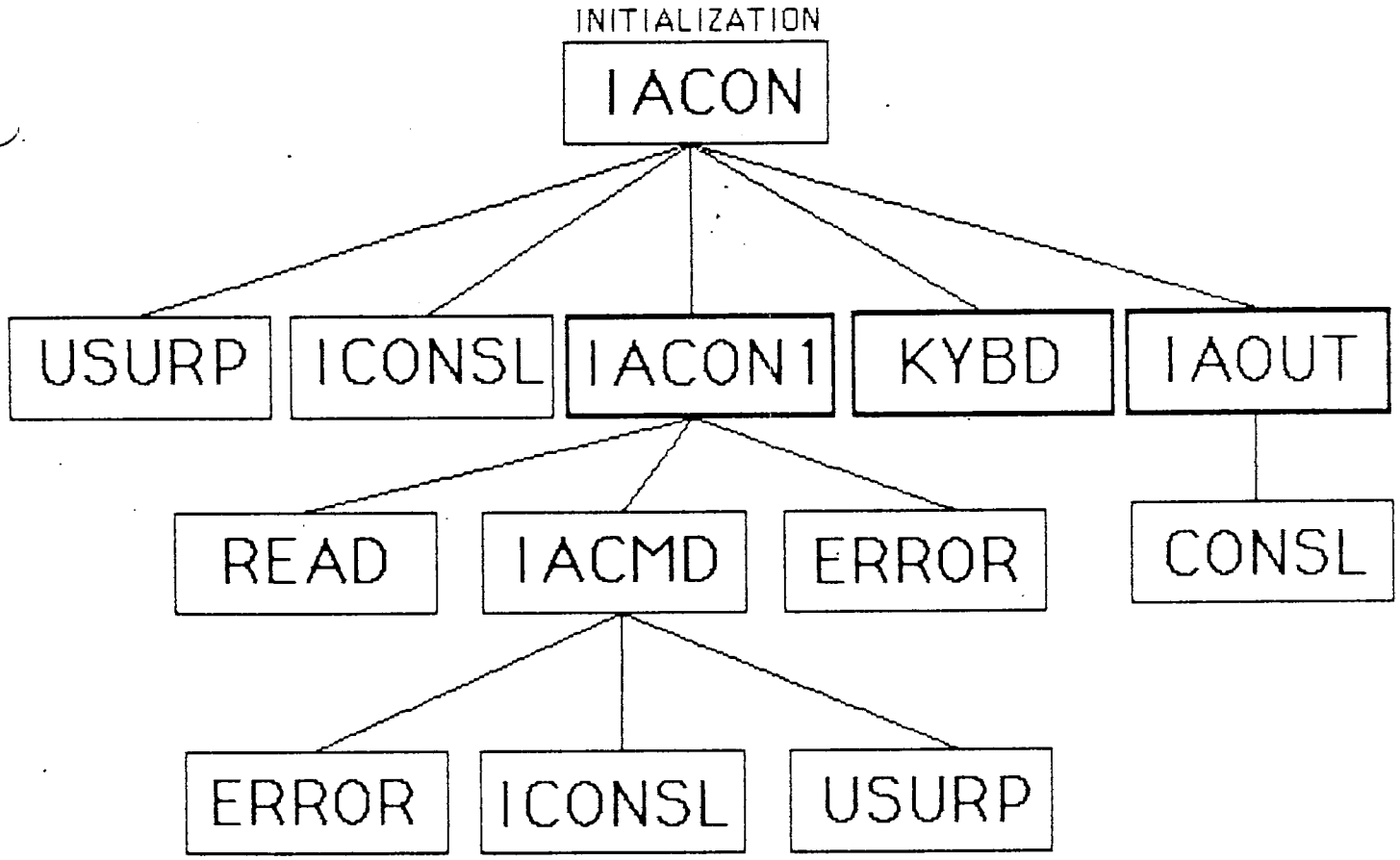
IACON Overlay:
Initializes interactive station console Ø.

ICONSL Overlay:
Acquires MOS Buffer and a local table for display screen data.

CRAYIO Overlay:
Does a STATIO request to BIOP to move LCP or Message to or from CM and BM.

INTERACTIVE OVERLAY TREE





TAPE SUBSYSTEM

Processes requests for tape I/O from TQM in CPU.
Moves data to and from central memory.
Performs error recovery and reporting.

CHARACTERISTICS

Consists of activities and demons in MIOP, BIOP and XIOP.
Processes requests asynchronously.
    Data movement to/from tape is independent of data movement to/from central memory.
Interfaces to block multiplexer driver to perform physical tape I/O.

MAIN COMPONENTS

BCOM Overlay:
    Interprocessor Tape Message Handler
    Directs DAL to appropriate activity

CONMAN Overlay:
    Processes configuration change requests

BYPASS Overlay:
    Controls transfer of data between XIOP and BIOP
    Sends DALs to BIOP
    One activity for all drives

BUFMAN Overlay:
    Controls allocation of Buffer Memory buffers for Tape Data

TEX Overlay:
    Main control overlay of tape driving
    Call TAPEIO and TAPEMOV to do I/O

BMXOPE Overlay:
    Processes OPEN and CLOSE requests

TAPEIO Overlay:
    Read/write control for TAPE EXEC
    Builds CPWs for driver and calls BMXSIO
    Calls TERROR on Tape error

TDEM Overlay:
    XIOP to/from buffer memory data handler

TDEMO Overlay:
    XIOP to buffer memory read reverse data handler

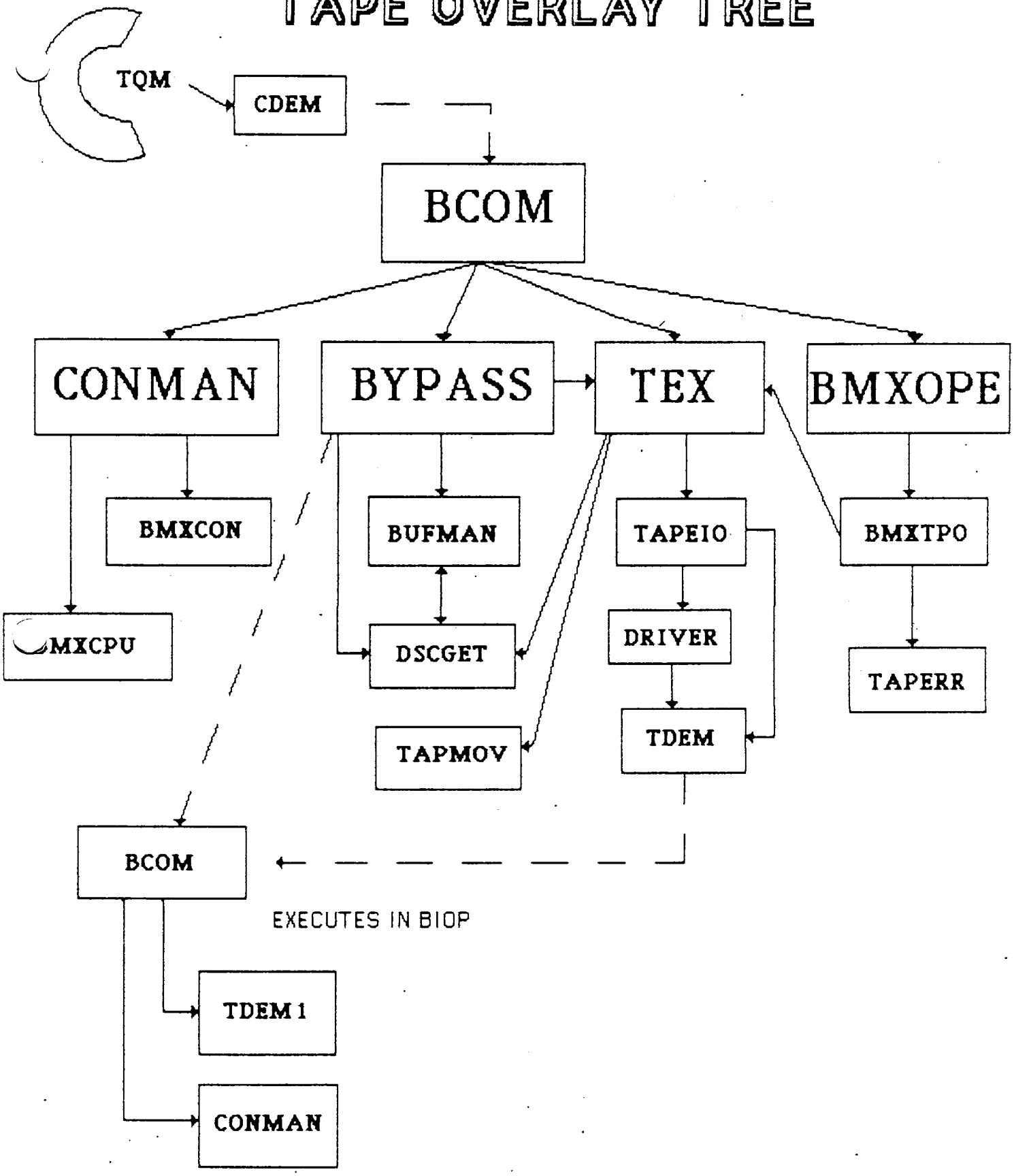
XDEM1 Overlay:
    BIOP to/from buffer memory data handler

TAPEMOV Overlay:
    Handles NONIO requests
    Build CPW for driver and calls BMXIO

BLOCK MULTIPLEXER DRIVER

Performs device independent command and interrupt sequences.
Performs channel selection.
Performs channel error reporting.

TAPE OVERLAY TREE



USER CHANNEL SUBSYSTEM

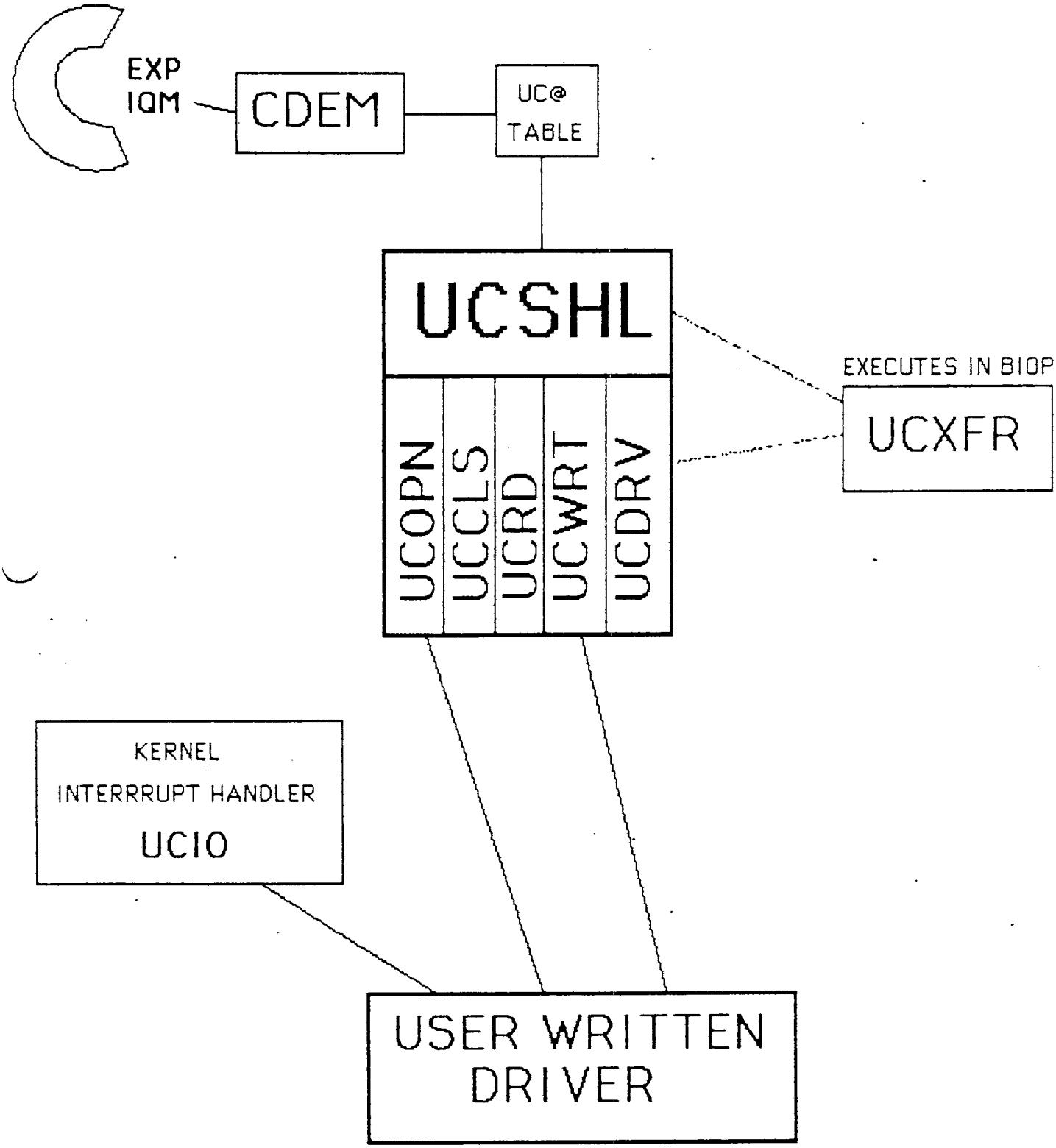
What is the user-channel shell?

- IOS overlays to process F-packets from the CPU. (UCSHL, CDEM, UCOPN, UCCLS, UCR,UCWRT)
- User-channel table (UCT).
- Kernel interrupt handler (UCIO).
- BIOP transfer overlay (UCXFR).
- Driver interface (SIGNAL/WATCH).
- Driver overlay.

What is the purpose of the shell?

- Allows COS jobs to invoke custom IOS driver overlays.
- Customer can write special-purpose channel drivers.
- Interface to driver overlay is simple, standardized.
- Shell takes care of request queuing, replies to CPU, table creation, memory allocation, creation of driver activity and data transfer.
- Driver only has to be concerned with the channel and device characteristics.
- CPU invokes driver by name, causes shell to link it to the requested channel.
- Accessible via F$DRIVER request from privileged COS jobs (see SM-0040, ch. 8).
- Accessible via common routine UDCOM for COS system tasks (STP). (See SM-0040, 4.10.)
- Independent shell and driver for input and output.
- Customer can implement new driver functions.

USER SHELL OVERLAY TREE



USER SHELL OVERLAY TREE

SYSTEM PROGRAMS

Match the following terms and their description

1. STATION CALL PROCESSOR	___ Inter IOP Message Handler for DD-19 and DD-29 disk activities
2. $APTEXT	___ Definitions for COS tables and constants
3. CONCIO	___ Consists of one overlay
4. IOP STATION	___ Work for IOP to perform
5. DEMON ACTIVITY	___ Processes all User exchanges using SØ as the function code
6. ACOM	___ Contains definitions of macros, constants, and tables for IOS
7. COSPL / IOPPL	___ Considered a front-end to COS
8. EXP	___ Communicates between the station and COS
9. COSTXT	___ Source listings for the operating system
10. ACTIVITY	___ Processes FEI channel activities in the IOP

SYSTEM PROGRAMS

Match the following terms and their description

1. STATION CALL PROCESSOR

2. SAPTEXT

3. CONCIO

4. IOP STATION

5. DEMON ACTIVITY

6. ACOM

7. COSPL / IOPPL

8. EXP

9. COSTXT

10. ACTIVITY

___ Inter IOP Message Handler for DD-19 and DD-29 disk activities

___ Definitions for COS tables and constants

___ Consists of one overlay

___ Work for IOP to perform

___ Processes all User exchanges using 58 as the function code

___ Contains definitions of macros, constants, and tables for IOS

___ Considered a front-end to COS

___ Communicates between the station and COS

___ Source routines for the operating system

___ Processes for channel activities in the IOP

Program Switching and Control

3

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon completion of the Program Swapping module, the learner should be able to:

1. Explain the mechanism with which COS switches programs.
2. Explain the mechanism with which IOS switches program.
3. Interpret the needed program executing environment.
4. Define an activity.
5. Interpret activity descriptors and storage modules.
6. Interpret Any Packets and DALS.

TASK EXCHANGING

EXEC

- Entered on an Interrupt
- Entered on every other exchange
- Processes reason for exchange
- Determines who to exchange back to

TASK

- Highest priority task exchanged back to first
- Determined from System Task Table
- All flags and channel interrupts are handled

USER

- Exchanged to if no tasks scheduled
- Users compete for CPU connection
- System Dataset Table holds queue
- Selected by class and priority

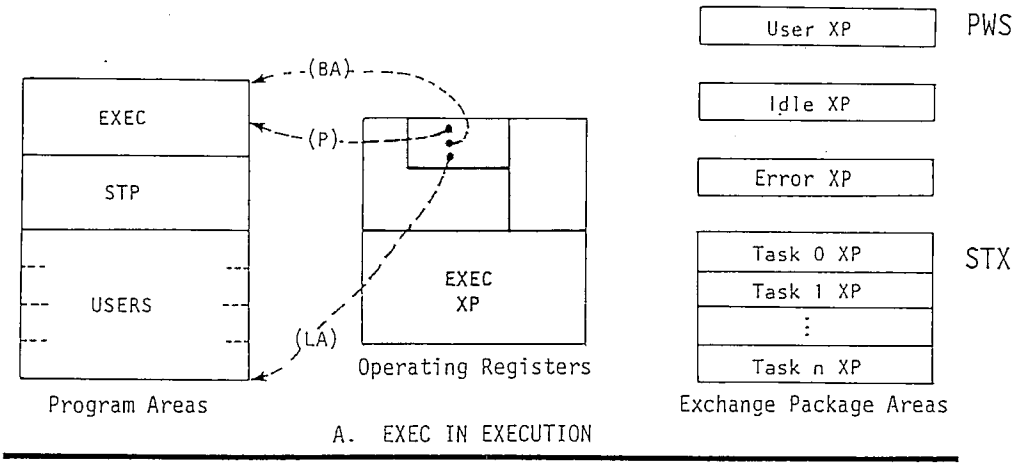
IDLE

- Exchanged to when nothing else is scheduled
- Reads memory looking for memory errors
- Reads Exec only

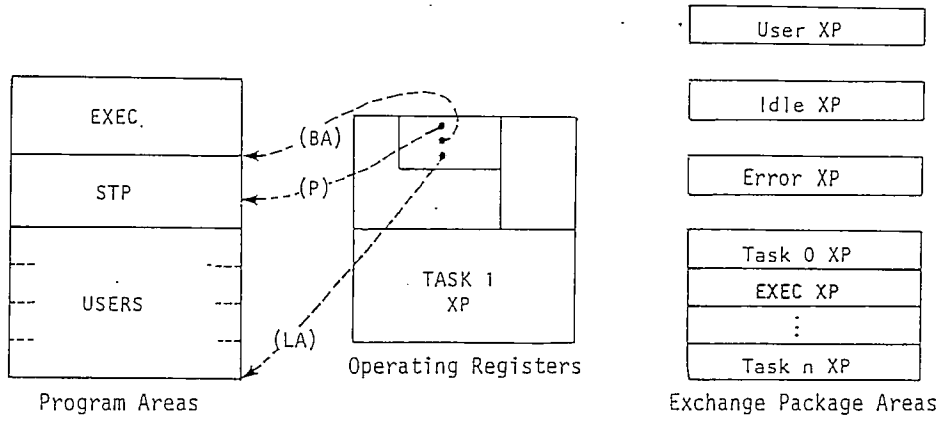
ERROR CORRECTION

- Logs error in MEL table
- Trys to correct error
- Logs the error in $SYSTEMLOG through STP

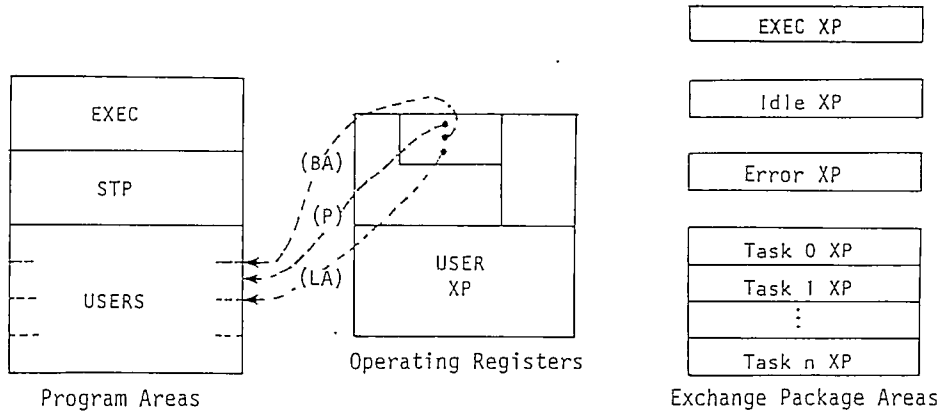
EXEC'S EXCHANGE PACKAGE



A. EXEC IN EXECUTION



B. TASK 1 IN EXECUTION



C. CURRENT USER IN EXECUTION

TASK XP AREA

Located in Exec Table Area

System Task Table contains Exchange Package Address

Has ASCII Label of Task name in STT

Exchange Packages in same order as task IDs

Exec table pointers contain STX address

SYSTEM TASK TABLE

Function - for scheduling and controlling STP tasks

STT Header

- STRTS Bit - request task scheduler flag
- Active task ID
- Active task exchange package address
- Active task parameter block address

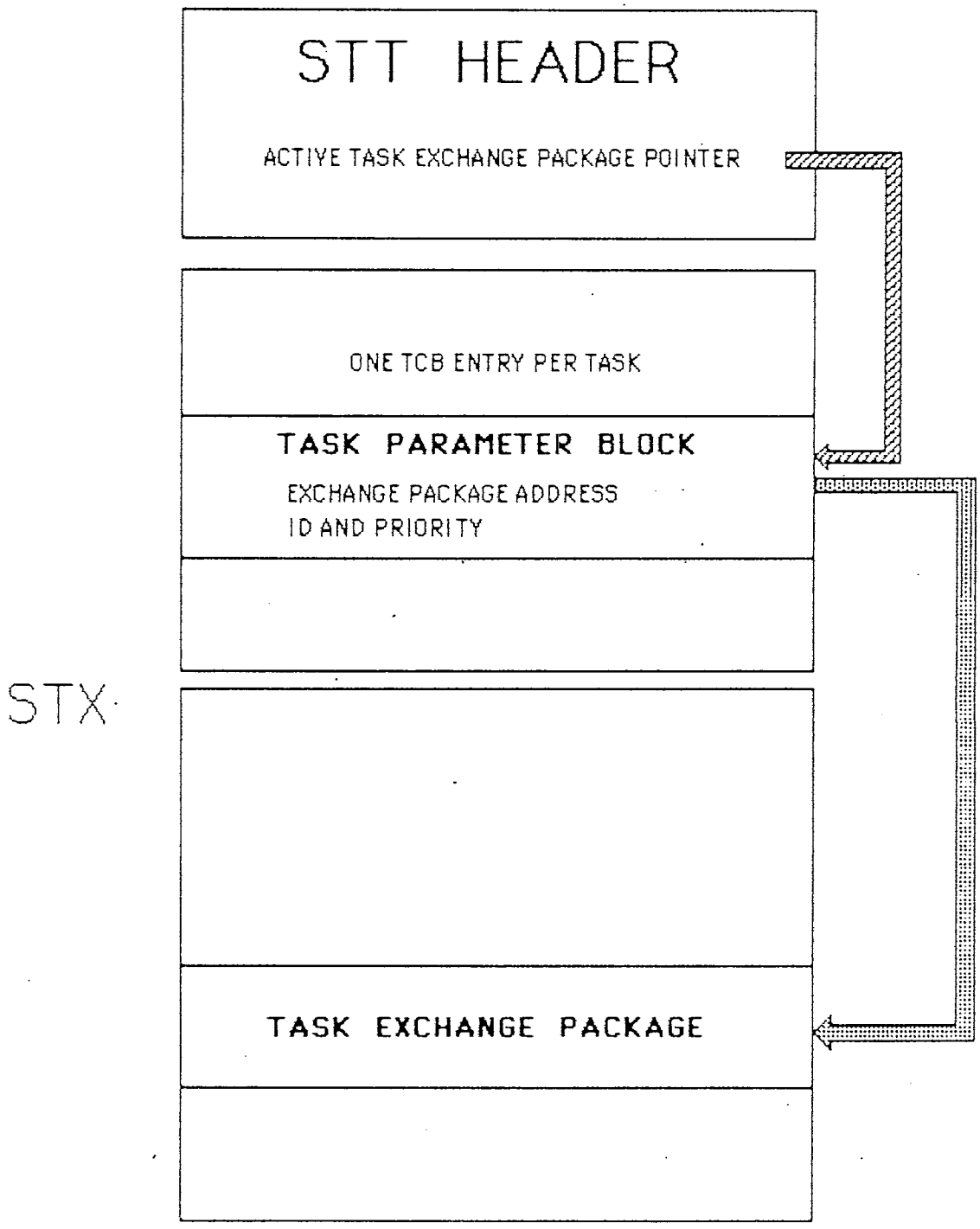
STT Part A - TPBs

- One entry for each task
- Ready bit
- Suspend bit
- Task Priority
- Task ID

STT Part B - STX

- One entry for each task
- Contains the exchange package for the task
- Located in the low memory XP area

SYSTEM TASK TABLE



ACTIVITY CHARACTERISTICS

Tasks executing in an IOP are called Activities.

An Activity is a routine (or routines) which perform a specific task.

Normally consists of nested Overlay calls.

Kernel maintains activities through the use of:

Activity Descriptors

Storage Modules (SMODs)

Popcells

DEMON ACTIVITIES

Activities may enable or disable interrupts.

A demon consists of one Overlay that performs high Priority Tasks.

Software Stacks (SMODS) are local memory resident.

Created on initialization by BEGIN overlay.

Activity Descriptors are never deallocated.

ACOM
AMSG
CDEM
DISK
BCOM
TDEM
CRTDEM
CLOCK

ACTIVITY DESCRIPTOR

Used by kernel to schedule and activate activities.

One for each activity.

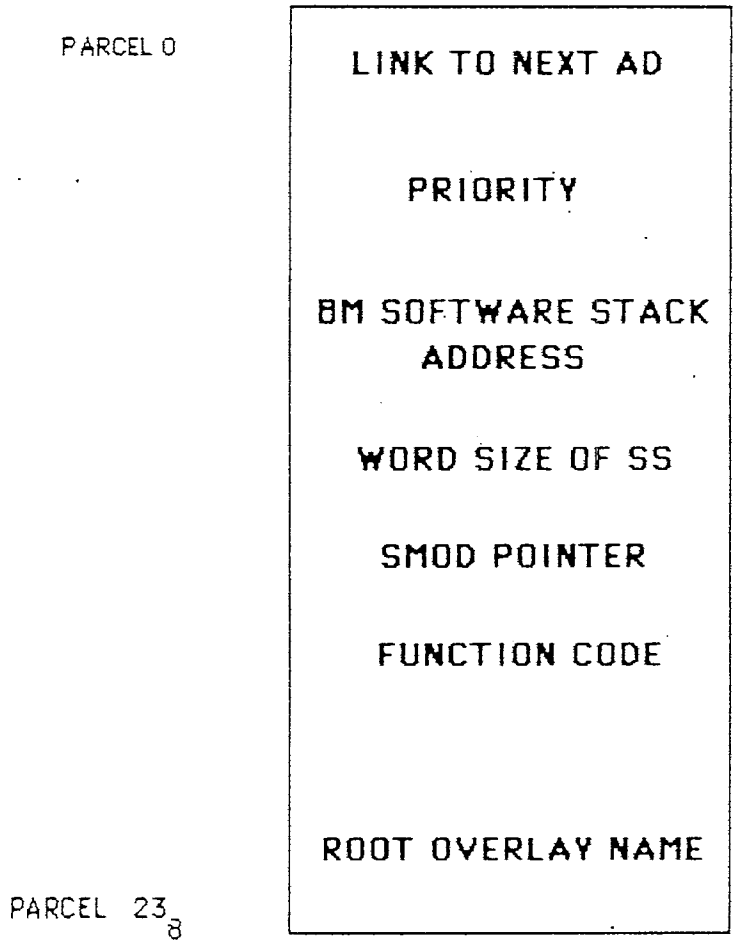
Built by the kernel as a part of CREATE service request.

Contains links, addresses and other information necessary to manage an activity.

Local memory resident.

Exists until an activity is terminated except with demons.

ACTIVITY DESCRIPTOR



IOS STORAGE MODULE (SMOD)

Used to save an overlay's executing environment.

One per overlay read into local memory.

Size varies depending on how many registers need to be saved.

Minimally contains:
- Links to activity descriptor and previous SMOD
- Overlay information
- A, B, C, E, and P register contents

May contain:
- Operand register contents essential to its overlay
- Program exit stack entries for its' overlay

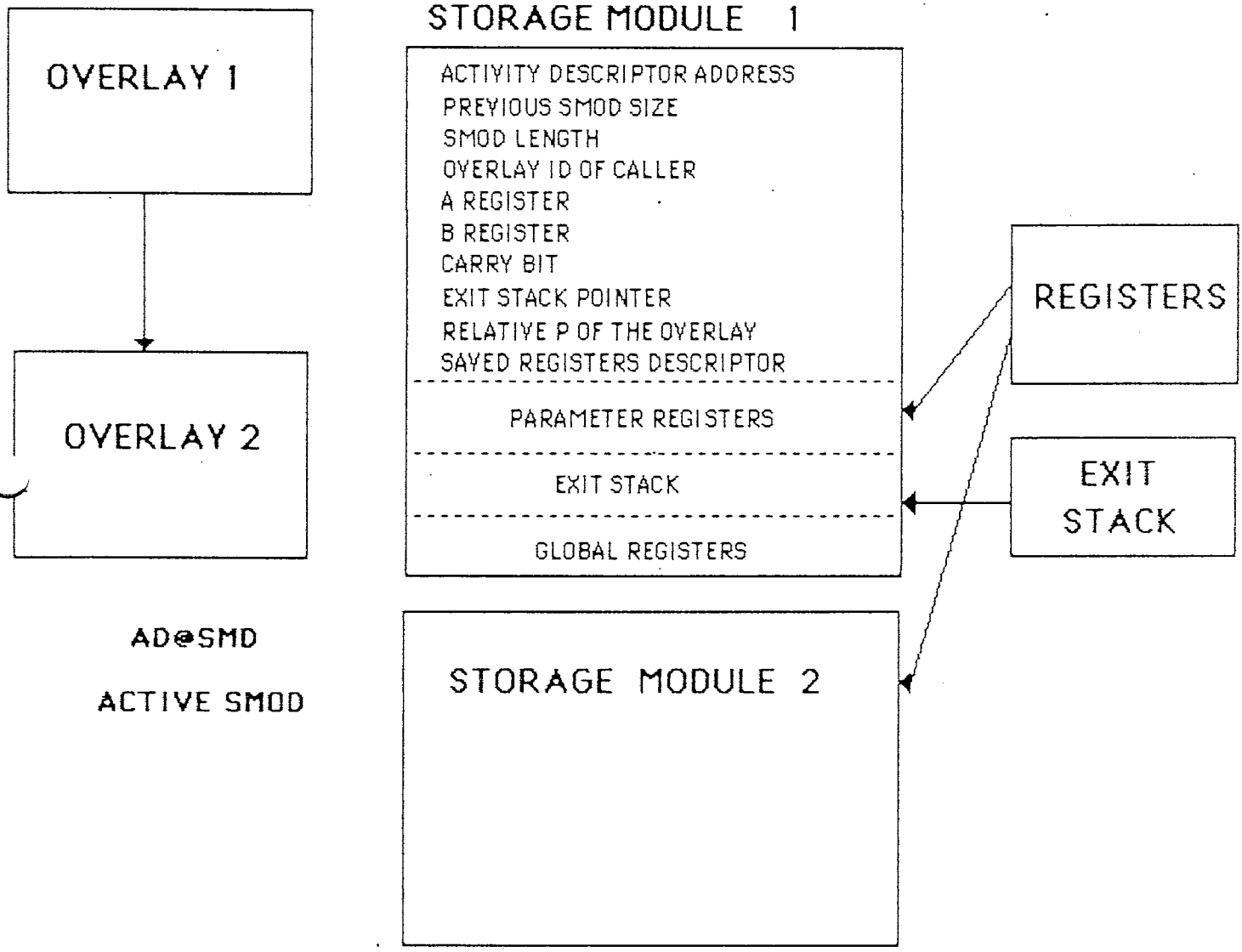
SMOD is partially updated when an overlay does a kernel service request.

If CALL results in loss of control, SMOD is completely updated.

Registers are re-loaded from SMOD when Service Request is completed or when overlay gets control back.

Initial SMOD set up through CREATE service request and is written to buffer memory as a software stack.

STORAGE MODULES



SOFTWARE STACK

There is a fixed stack in local memory where the SMODs for the current activity's overlays reside.

A SMOD is 'PUSHED' onto this stack when an overlay CALLS another overlay.

> A 'PUSH' consists of saving an overlay's registers and updating the SMOD pointer in the activity descriptor.

The caller's SMOD is 'popped' off this stack when the called overlay does a RETURN service request.

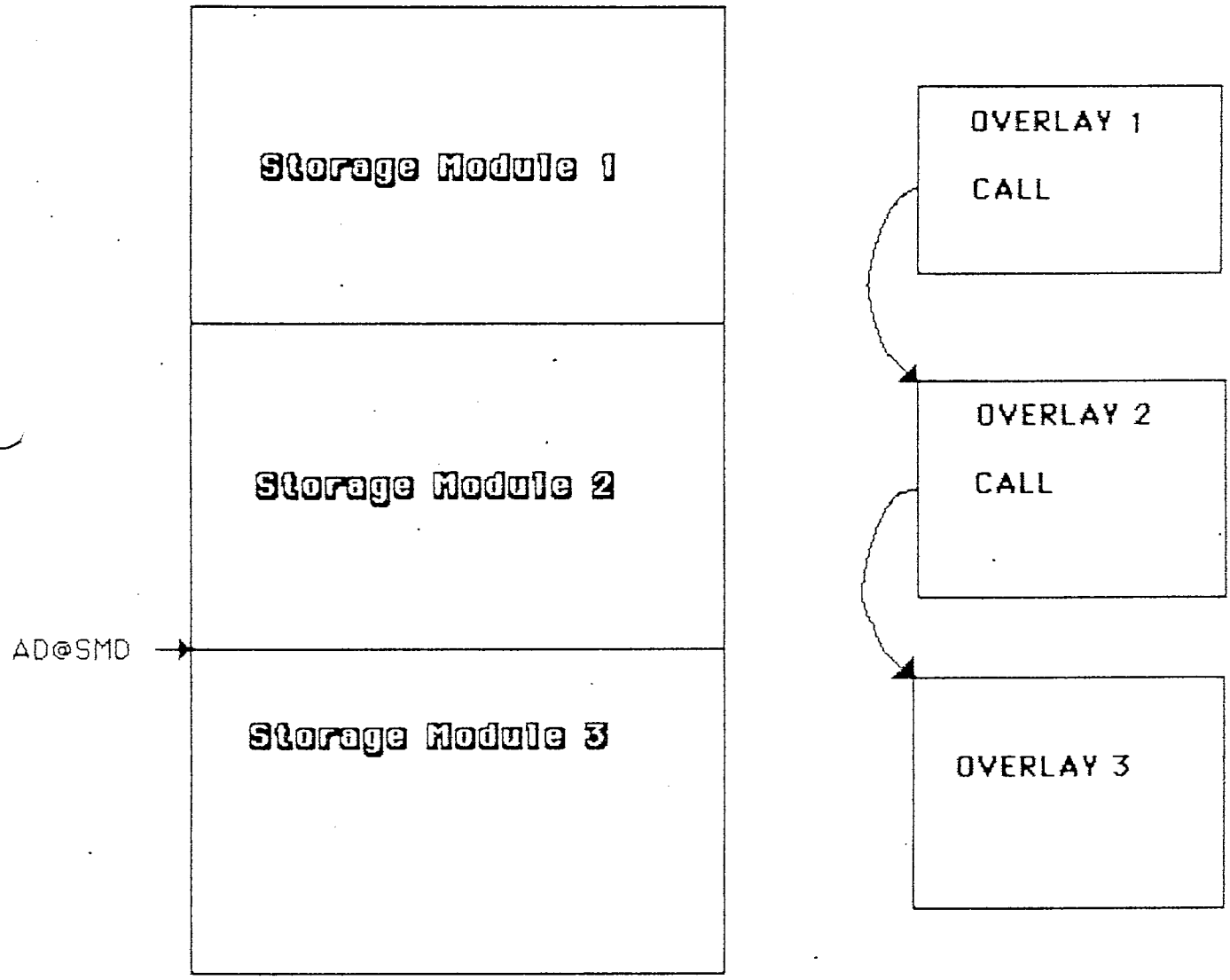
> A 'POP' consists of updating the SMOD pointer and restoring the calling overlay's registers.

This software stack is written out to buffer memory when an activity relinquishes control to the kernel and other activities are on the central processor queue.

> The local memory software stack is not free for use by another activity.

When an activity regains controls, its software stack will be read into the local software stack from the buffer memory.

SOFTWARE STACK



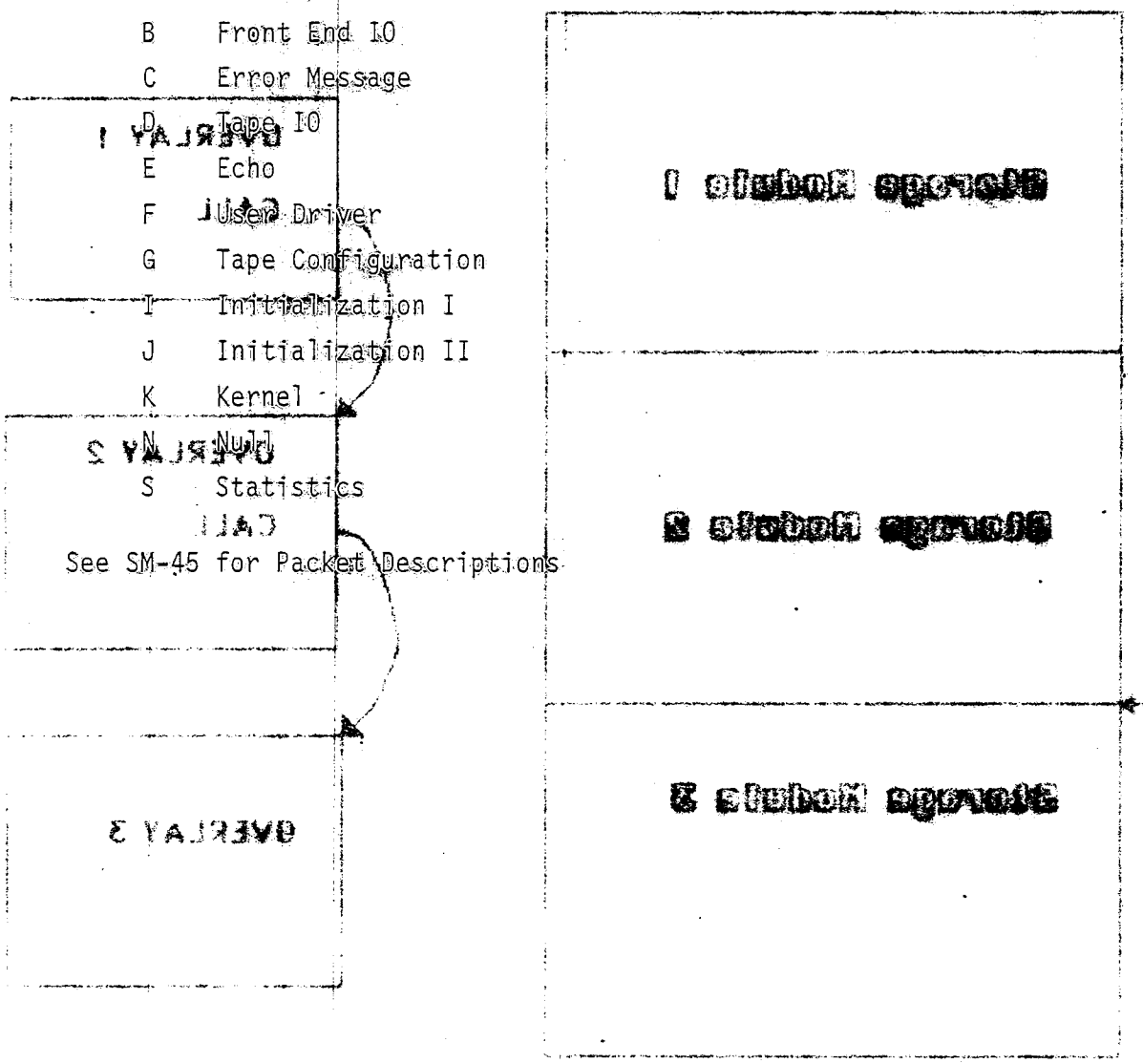
ANY PACKETS

Messages between COS and IOS

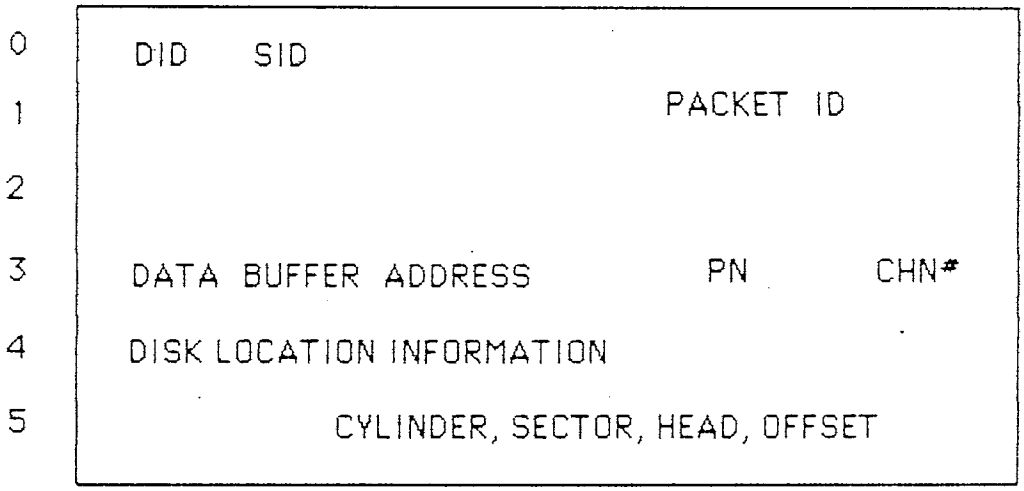
6 word block of information

C1	Cray Mainframe ID
EX	Exec IO
A	Disk I/O
B	Front End IO
C	Error Message
D	Tape IO
E	Echo
F	User Driver
G	Tape Configuration
I	Initialization I
J	Initialization II
K	Kernel
N	Null
S	Statistics

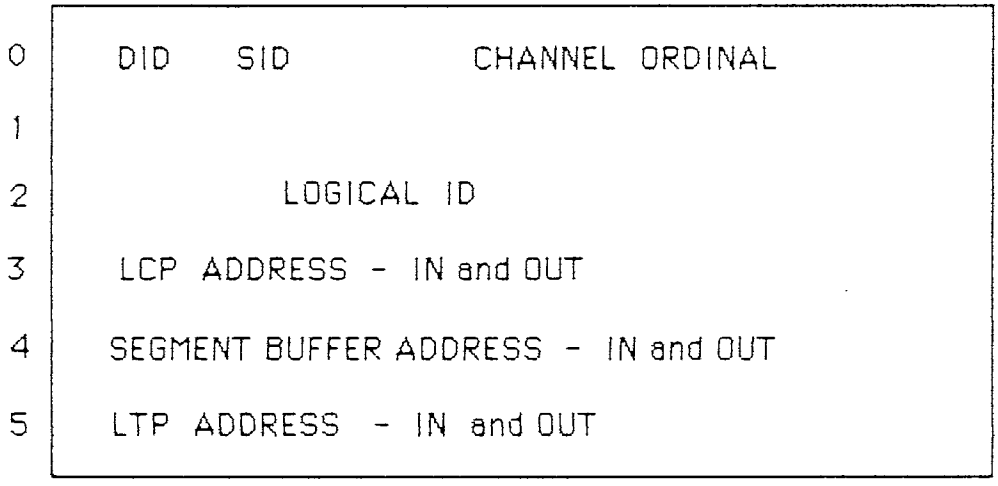
See SM-45 for Packet Descriptions



ANY PACKETS



A PACKET - DISK REQUEST



B PACKET - FRONT END REQUEST

DISK ACTIVITY LINK

Communication packet between IOPs

40_8 parcels long

Any packet contained within the DAL

Sending IOP initiates by a Kernel service request in CDEM

Passed through buffer memory

Pointed to by inter A-A message (see SM-46 pp. 2-43)

Receiving IOP reads in with ACOM, BCOM or ICOM

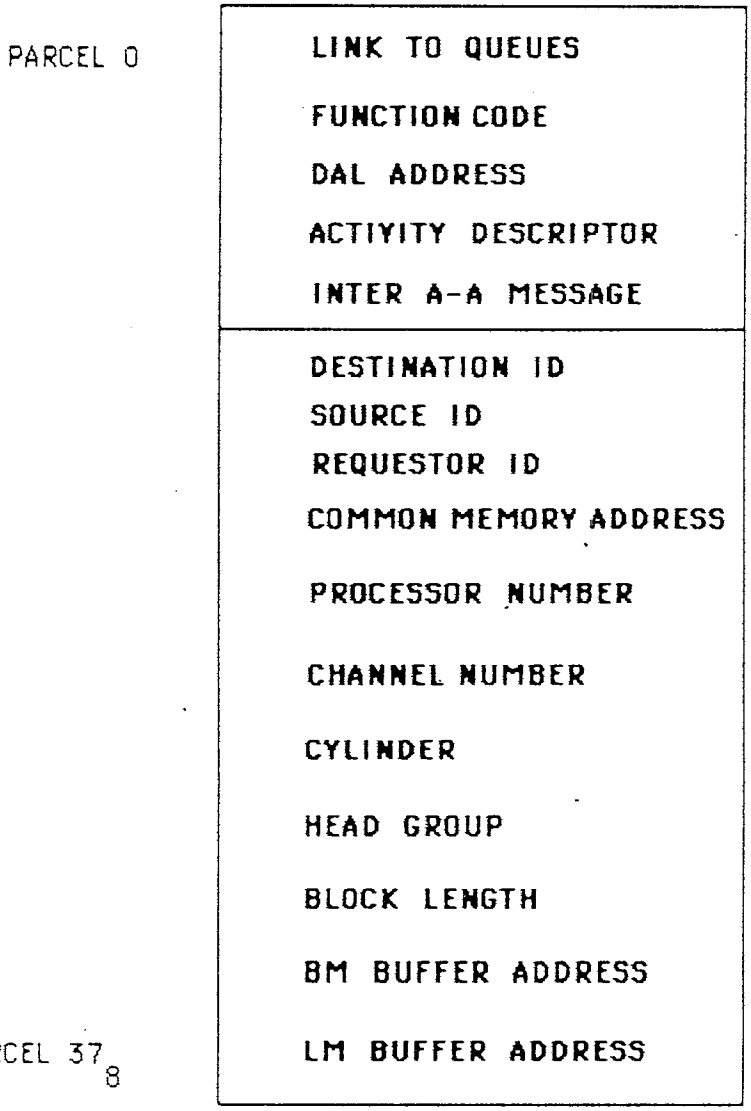
Master DAL

- Message packet created by a request from the mainframe
- 40_8 parcels in length
- Built by MIOP from the CPU B packet
- One for each I/O request
- Passed to disk subsystem in appropriate IOP
- Disk subsystem returns master DAL to MIOP when I/O request is completed.
- Disk subsystem uses master DAL as a template for building executable DALS.

Executable DAL

- Built from MDAL
- Controls the transfer of one sector between disk and central memory
- 40_8 parcels in length
- Built by disk subsystem from a master DAL
- One for each sector of disk requested
- Used by disk subsystem to keep track of where each sector of data is
- Usually passed by DIOP to BIOP for high speed transfer requests
  - Returned when transfer complete
- Also referred to as slave DAL or EDAL

DISK ACTIVITY LINK



DIGX ACTIVITY LINK

PARCEL 0

LINK TO QUEUES
FUNCTION CODE
DAL ADDRESS
ACTIVITY DESCRIPTOR
INTER A-A MESSAGE

DESTINATION ID
SOURCE ID
REQUESTOR ID
COMMON MEMORY ADDRESS
PROCESSOR NUMBER
CHANNEL NUMBER
CYLINDER
HEAD GROUP
BLOCK LENGTH
BM BUFFER ADDRESS
LM BUFFER ADDRESS

PARCEL 1

PROGRAM SWITCHING

Match the following terms and their descriptions

1. EXCHANGE PACKAGE	___ Defines an Activity in IOS
2. DISK ACTIVITY LINK	___ 6 words of information passed between COS and IOS
3. SYSTEM TASK TABLE	___ Activates ACOM, BCOM, ICOM giving Buffer Memory address of DAL being passed
4. ACTIVITY DESCRIPTOR	___ All SMOD's of an activity
5. STORAGE MODULE	___ Central Memory address for all COS Task exchange packages
6. ANY PACKET	___ Controls COS STP Tasks
7. SOFTWARE STACK	___ Central Memory location for user exchange packages
8. STX	___ Information passed through buffer memory to another IOP
9. PROCESSOR WORKING STORAGE	___ An Overlay program's executing enviroment
10. INTER A-A MESSAGE Interrupt Handler	___ A COS mainframe programs executing enviroment

PROGRAM SWITCHING

Match the following terms and their descriptions

1. EXCHANGE PACKAGE ____ Defines an Activity in IOS

2. DISK ACTIVITY LINK ____ 6 words of information passed between CCS and IOS

3. SYSTEM TASK TABLE ____ Activates ACOM, BCOM, ICOM giving Buffer Memory address of DAL being passed

4. ACTIVITY DESCRIPTOR ____ All SMCO's of an activity

5. STORAGE MODULE ____ Central Memory address for all CCS Task exchange packages

6. ANY PACKET ____ Controls CCS STP Tasks

7. SOFTWARE STACK ____ Central Memory location for user exchange packages

8. STX ____ Information passed through buffer memory to another IOP

9. PROCESSOR WORKING STORAGE ____ An overlay program's executing environment

10. INTER A-A MESSAGE Interrupt Handler ____ A CCS misinterpretive programs executing environment

System Resources

4

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon completion of this System Resources module, the learner should be able to:

1. Draw out a Cray computer system.
2. Explain the main function of each hardware component.
3. Label all mass storage device areas.
4. Map central memory, common memory, and local memory.

MIOP responsibilities are as follows:

- The MIOP is the first I/O Processor in the I/O Subsystem to be deadstarted. The MIOP initializes the contents of Buffer Memory and deadstarts the other processors in the configuration using Buffer Memory and accumulator channels to the other processors.
- The MIOP, in concert with the BIOP, deadstarts the mainframe.
- The MIOP handles all communication with the mainframe over the 6 Mbyte channel. This traffic includes disk and tape requests and station communications.
- The MIOP performs front-end and station software support.
- The MIOP handles input and output operations on the expander channel.
- The MIOP accepts information from the error channel and transmits it to the mainframe for inclusion in the system error log.
- The MIOP is the operator interface to the I/O Subsystem editor, which maintains deadstart and restart parameter files.
- The MIOP handles input and output operations on user channels for COS tasks.

BIOP responsibilities are as follows:

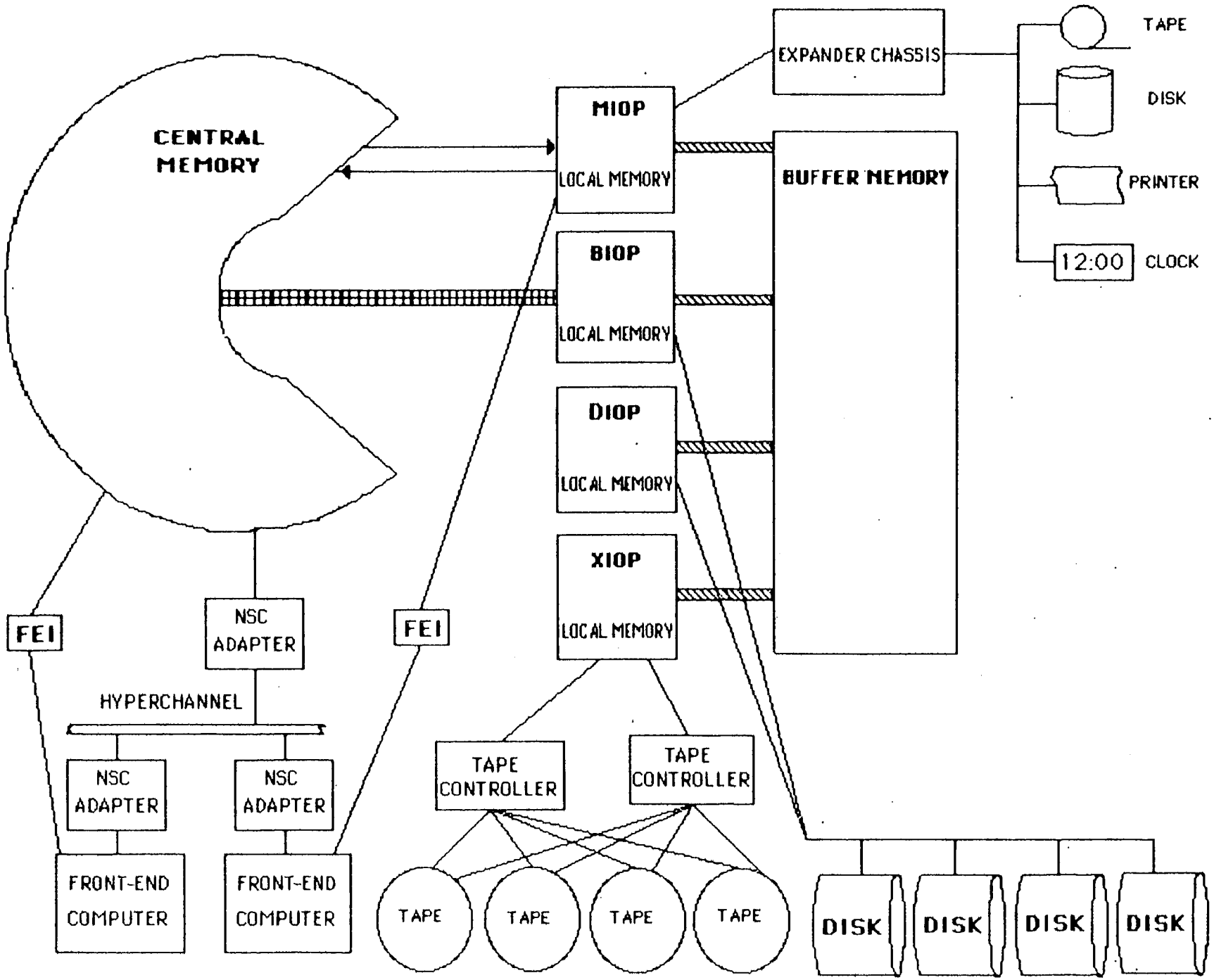
- The BIOP transfers data between Central Memory and Buffer Memory and vice versa across a 100 Mbyte channel.
- The BIOP performs disk I/O to and from disk units attached to its channels. (IOS software supports the DD-19, DD-29, and DD-49 Disk Storage Units.) It performs error recovery when errors are detected on data transfers.

DIOP responsibilities are as follows:

- The DIOP moves data from Buffer Memory to disk and vice versa at the request of packets from the mainframe via the MIOP. These packets also return status to the requester.
- When errors are detected in data transfers to or from disk, DIOP software attempts error recovery.
- If the optional second 100 Mbyte channel is present, the DIOP transfers data between Central Memory and Local Memory and vice versa.

XIOP responsibilities are as follows:

- The XIOP handles data from IBM-compatible tape drives and buffers the data to Buffer Memory at the request of packets from the mainframe.
- When errors are detected while transferring data to or from tape, the XIOP performs error recovery procedures.


CENTRAL MEMORY
MIOP
LOCAL MEMORY
BIOP
LOCAL MEMORY
DIOP
LOCAL MEMORY
XIOP
LOCAL MEMORY
BUFFER MEMORY
EXPANDER CHASSIS
TAPE
DISK
PRINTER
12:00
CLOCK
FEI
NSC ADAPTER
FEI
HYPERCHANNEL
NSC ADAPTER
NSC ADAPTER
FRONT-END COMPUTER
FRONT-END COMPUTER
TAPE CONTROLLER
TAPE CONTROLLER
TAPE
TAPE
TAPE
TAPE
DISK
DISK
DISK
DISK
HARDWARE

FEATURES	DD-19	DD-29	DD-49
Word Capacity / Drive	3.723×10^7	7.483×10^7	15.6×10^7
Word Capacity / Cylinder	92,160	92,160	175,000
Bit Capacity / Drive	2.424×10^9	4.789×10^9	20×10^9
Cylinders / Drive	411	823	891
Sectors / Track	18	18	44
Bits / Sector	32,768	32,768	32,768
Number of Heads	40	40	32
Latency	16.7ms	16.7ms	16.7ms
Access Time	15-80ms	15-80ms	2.5-32ms
Data Transfer Rate	35.4×10^6	35.4×10^6	200×10^6

Device Label Table - DVL

A device label table exists on the first usable track of a mass storage device and is created by STARTUP during an install. The device label table contains a bad track label for the device. The device label for the master device also contains pointers to the Dataset Catalog.

Engineering Flaw Table - EFT

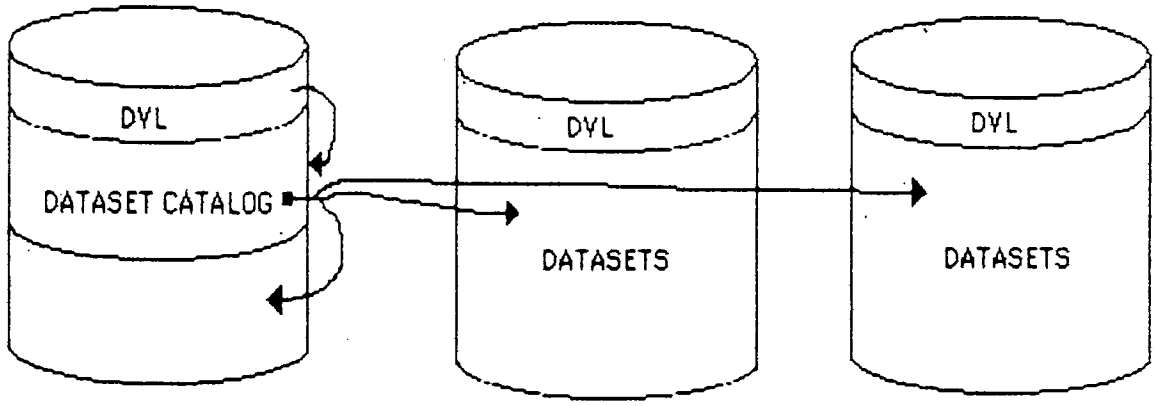
Engineering diagnostics use this disk-resident table to record flaws encountered during a surface analysis of a disk pack. STARTUP uses the table to make flawed tracks unavailable to COS.

The EFT is written to Sector 20 octal of the first track that can be successfully reread on the device (no more than 10 tracks are tried). No EFT is written if no track in the first 10 tracks can be written and reread successfully. Install reads back each DVL after writing it to verify the integrity of the DVL. If the Device Label cannot be read back perfectly, then the track is overwritten with a test pattern and a different track is tried.

Formatting

Before a unit can be introduced into the system, it must be formatted. Formatting is the process of writing cylinder, head, and sector addresses onto the disk drive. This process is performed off-line by field engineers, and unless addressing information has been inadvertantly destroyed, formatting is performed only once.

DISK STORAGE



MASTER DEVICE

DEVICE LABEL

DLB – DEVICE LABEL		MASTER DEVICE FLAG
LOGICAL DEVICE ID	49-A1-20 39A-A2-27	
MASTER DEVICE DATASET CATALOG (DSC)		

CRAY STATION. VERSION X.15, IOS L S R M 01/08/85 15.47.47

MASS STORAGE STATUS FRAME 0

FLAGS	DEVICE	SPACE FREE	SPACE PERM	ERRORS RECOV	ERRORS UNREC	LAST ERROR CYL	LAST ERROR HD	LAST ERROR SC
M P	49-A1-20	47%	53%	0	0	0	0	0
DP	49-A1-21	100%	0%	0	0	0	0	0
S	39-1-22A	74%	0%	2	0	3	4	0
S	39-1-22B	98%	0%	0	0	0	0	0
S	39-1-22C	99%	0%	8	0	12	3	0
S	39-1-23A	76%	0%	23	0	52	3	0
S	39-1-23B	99%	0%	7	0	14	3	0
S	39-1-23C	99%	0%	0	0	0	0	0
	49-A1-24	28%	67%	0	0	0	0	0
	49-A1-25	32%	84%	0	0	0	0	0
P	49-A1-26	100%	0%	0	0	0	0	0
	49-A1-27	38%	60%	5	0	633	3	45
	49-A1-30	28%	69%	0	0	0	0	0

>STO
>SNA

LOCAL MEMORY

The Kernel resides in local memory in each IOP.

The Kernel maintains four local memory chains.

Overlay Memory Chain:

- Located after Kernel
- Allocated in multiples of four parcels
- Implemented as a doubly linked list
- Used to hold executing overlays
- Operand register %MEMORY points to the beginning of the list
- Operand register %B points to the active overlay

Disk Activity Link Chain:

- Located after overlay memory
- Allocated in multiples of 40_8 parcels
- Implemented as a forward linked list
- Used for message space
- Memory location EDES points to the beginning of the list

Free Memory Chain:

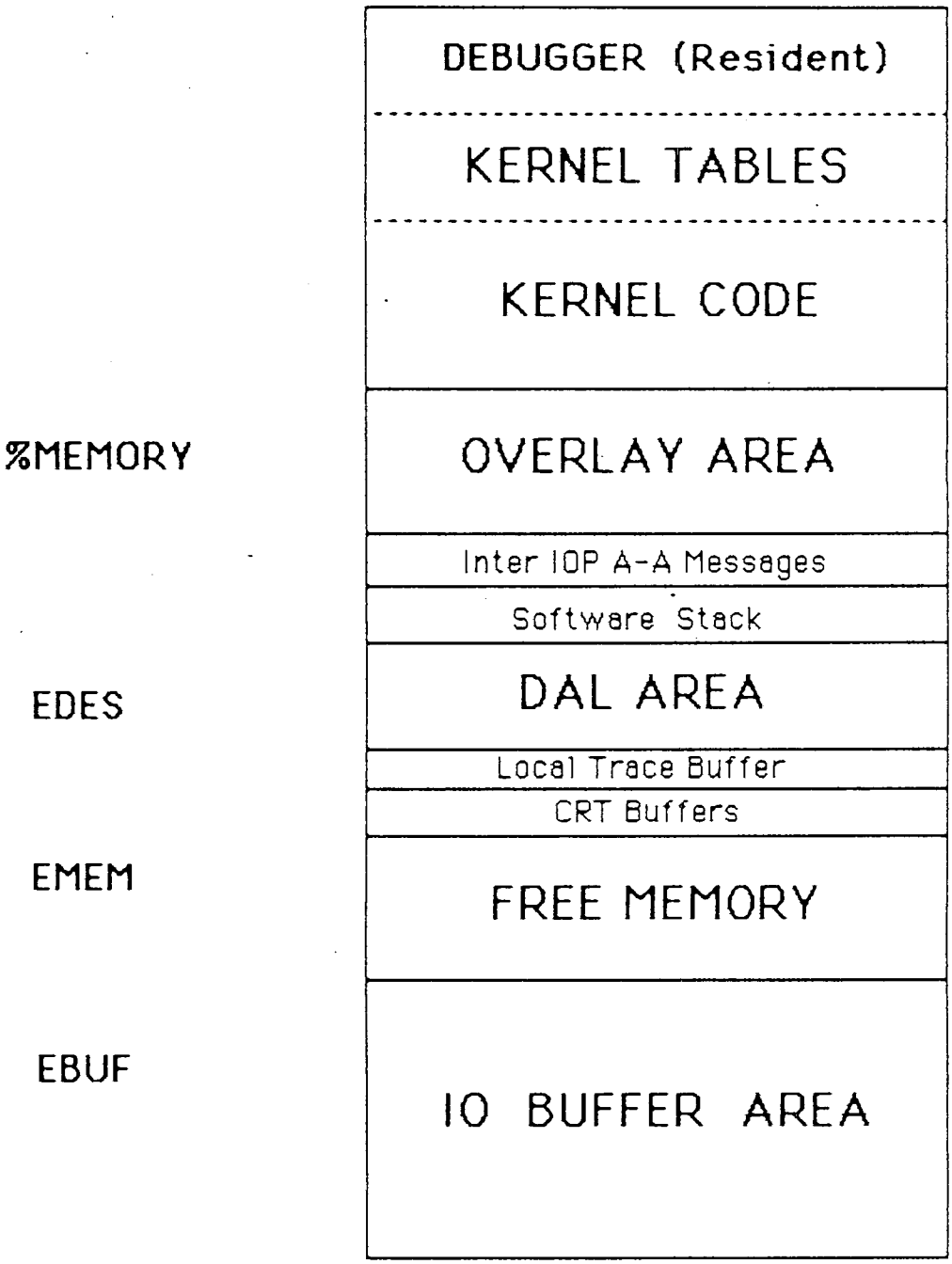
- Located after DALS
- Allocated in multiples of four parcels
- Implemented as a forward linked list
- Used for tables, activity descriptors and popcells
- Memory location EMEM points to the beginning of the list

Local I/O Buffer Chain:

- Located in upper memory
- Allocated in multiples of 4000_8 parcels
- Implemented as a forward linked list
- Used mainly for I/O buffers
- Memory location EBUF points to the beginning of the list

These pointers are built on initialization in the Kernel SIN routine or in the SYSS.

LOCAL MEMORY



BUFFER MEMORY

Deadstart Buffer

Image of the Kernel for initialization

System Directory

Pointers to Buffer Memory Address boundaries

Disk Activity Link Area

Interprocessor DAL area

Size of area set in AMAP for each IOP

Each message area is 40_8 parcels

AMAP

Configuration of IOP channels

Text Overlay

Overlays

IOS System Programs

Read only Programs

LISTO will give BM address of each overlay

Kernel Storage

Tables and Queues

Software Stack Area
An overlay's execution environment is saved here when it loses control of the CPU

Allocated in 400_8 blocks

I/O Buffer
IO dataset area

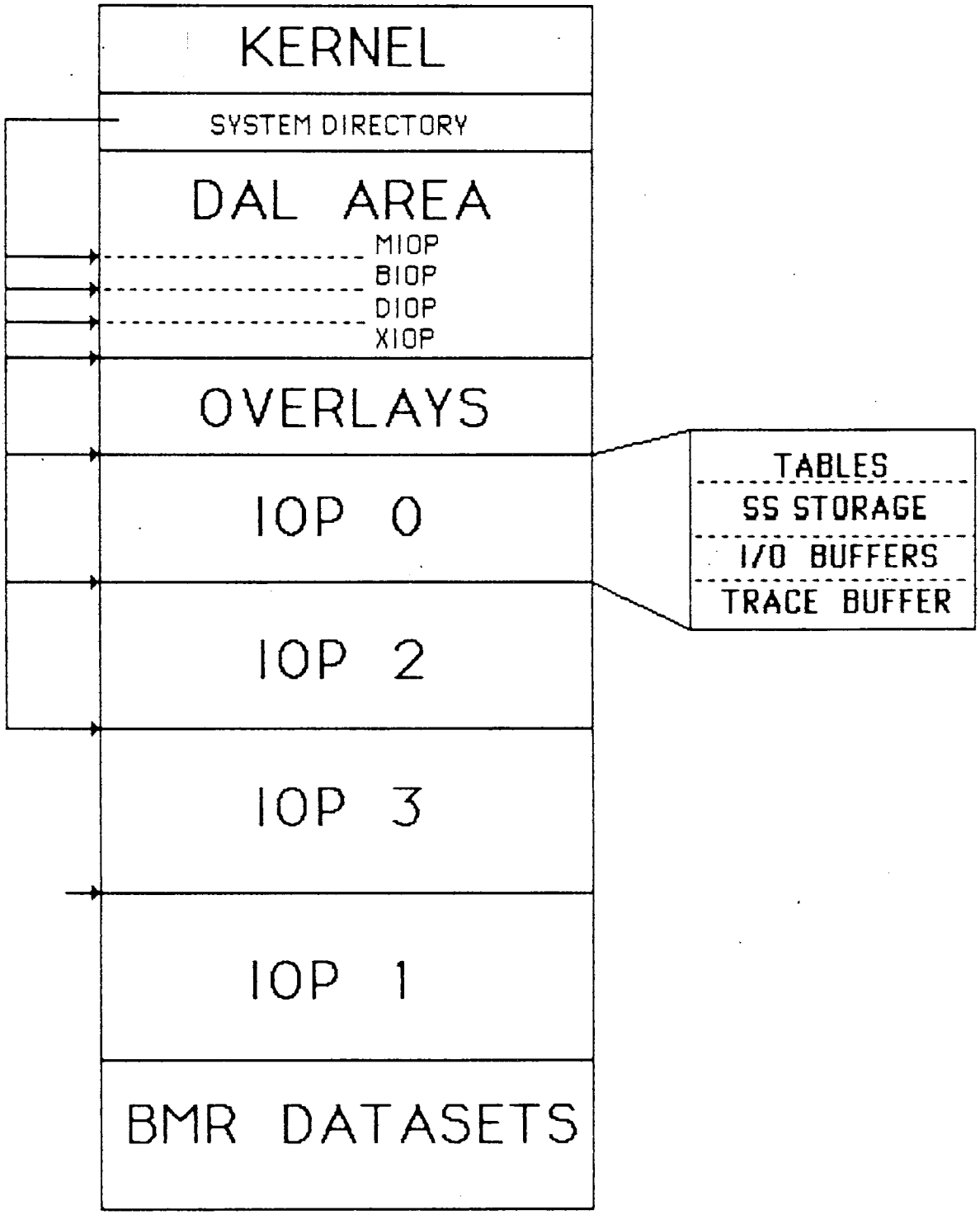
Trace Buffer
Circular history trace for debugging

Buffer Memory Resident Datasets

Mass storage area usable from JCL

Logical Device name is BMR-ØØ

BUFFER MEMORY



BUFFER MEMORY

SYSTEM DIRECTORY

Parcel	Bits	Description
0-1	0-15	MIOP message area
2-3	0-15	BIOP message area
4-5	0-15	IOP-2 message area
6-7	0-15	IOP-3 message area
10-13	0-15	Reserved
14-15	0-15	First overlay (AMAP) address
16-17	0-15	Unused
20-21	0-15	MIOP Kernel storage area
22-23	0-15	Size of MIOP storage area in 512-word blocks
24-25	0-15	BIOP Kernel storage area
26-27	0-15	Size of BIOP storage area in 512-word blocks
30-31	0-15	IOP-2 Kernel storage area
32-33	0-15	Size of IOP-2 storage area in 512-word blocks
34-35	0-15	IOP-3 Kernel storage area
36-37	0-15	Size of IOP-3 storage area in 512-word blocks

CENTRAL MEMORY

EXEC
- Exchange Package Area
- Table Area
- Monitor Instructions
- Interrupt Handlers
- Exec Request Processor
- Channel Drivers

System Task Processors
- 15 programs loaded after EXEC Handles Resource Needs and Scheduling
- Table Area within Task Area Loader Map Gives Relative Address

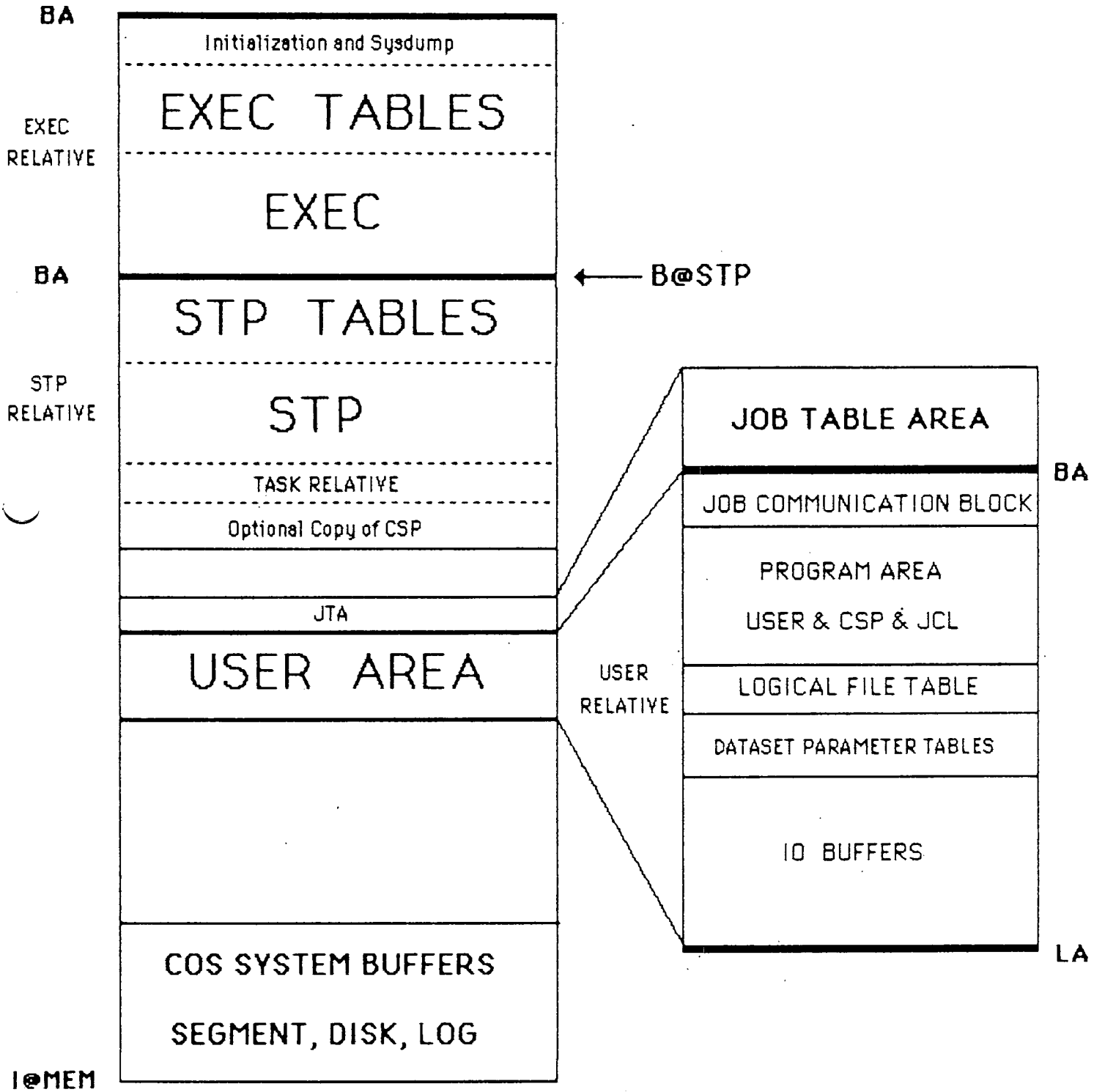
Job Table Area
- Job Related Information
- User Log
- User JCL
- Exchange Package
- B,T,V Registers
- Dataset Name Tables
- Not Accessible to User
- Can be Dumped

User Area
- Job Communication Block
- User Program/Data
  - Control Statement Processor
  - Product Set
  - User Program
- Logical File Tables
- Dataset Parameter Area
- I/O Buffers

System Log Buffer
- Used by MSG and MEP

System IO Buffers
- Disk.
- Segment

CENTRAL MEMORY



CENTRAL MEMORY
EXEC TABLES
EXEC
STP TABLES
STP
USER AREA
JOB TABLE AREA
USER RELATIVE
STP RELATIVE
EXEC RELATIVE
BA
LA

SYSTEM RESOURCES

Match these terms and their description:

1. MASTER DEVICE	____ Considered part of the IOP station
2. MIOP	____ Logically considered a fast disk storage device
3. BIOP	____ Main responsibility is handling the NSC hyperchannel and Front End channels
4. DIOP	____ Main Responsibility is to drive IBM type Block mux devices and tape units
5. XIOP	____ Contains the Dataset Catalogue
6. SSD	____ Contains a High Speed channel for use by MIOP for front-end activities accessed by STATIO
7. BUFFER MEMORY	____ Main responsibilty is to drive Disk Storage devices
8. EXPANDER CHASSIS	____ Memory used by each IOP for monitor and executing IOS activities
9. CENTRAL MEMORY	____ Working storage for IOP activities
10. LOCAL MEMORY	____ Memory where COS resides

SYSTEM RESOURCES

Match these terms and their description.

1. MASTER DEVICE	____ Considered part of the IOP station
2. MIOP	____ Logically considered a fast disk storage device
3. BIOP	____ Main responsibility is handling the MSC hyperchannel and front end channels
4. DIOP	____ Main Responsibility is to drive IBM type Block mux devices and tape units
5. XIOP	____ Contains the Dataset Catalogue
6. SSD	____ Contains a High Speed channel for use by MIOP for front-end activities accessed by SIATIO
7. BUFFER MEMORY	____ Main responsibility is to drive Disk Storage devices
8. EXPANDER CHASSIS	____ Memory used by each IOP for monitor and executing IOP activities
9. CENTRAL MEMORY	____ Working storage for IOP activities
10. LOCAL MEMORY	____ Memory where LOS resides

Station Protocol

5

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon the completion of this Station Protocol module, the learner should be able to:

1. List the difference between networking and stations.
2. Explain what a station is.
3. List some functions of a station.
4. Break down and interpret a station message link control package.
5. Explain the sequence a station goes through for a log-on.

WHAT IS A STATION?

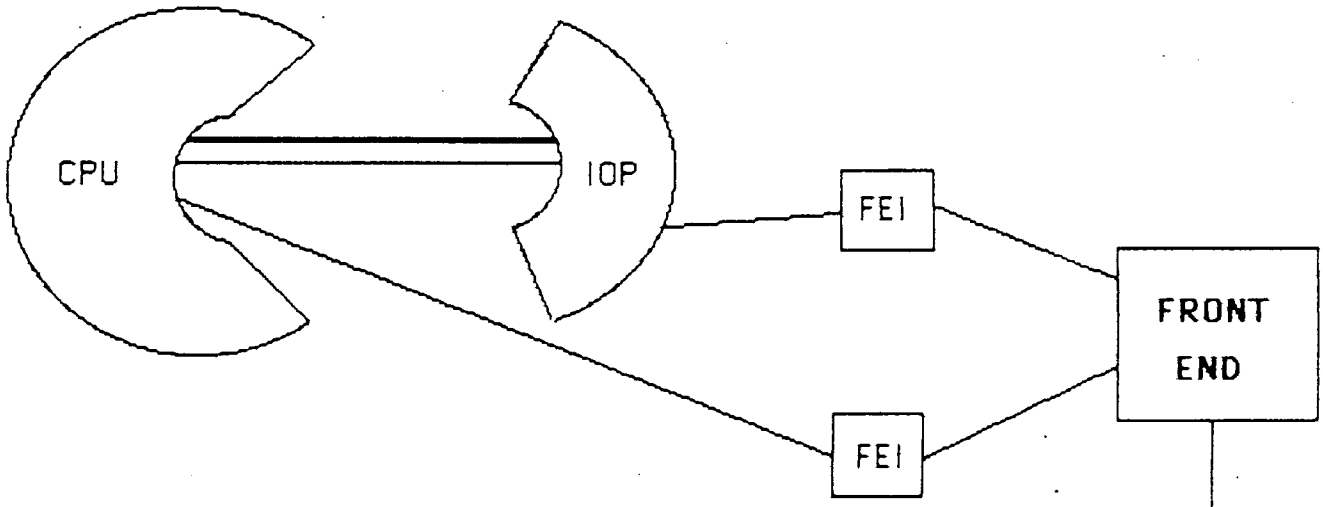
The hardware and software connecting a front-end computer to the Cray.

Functions:

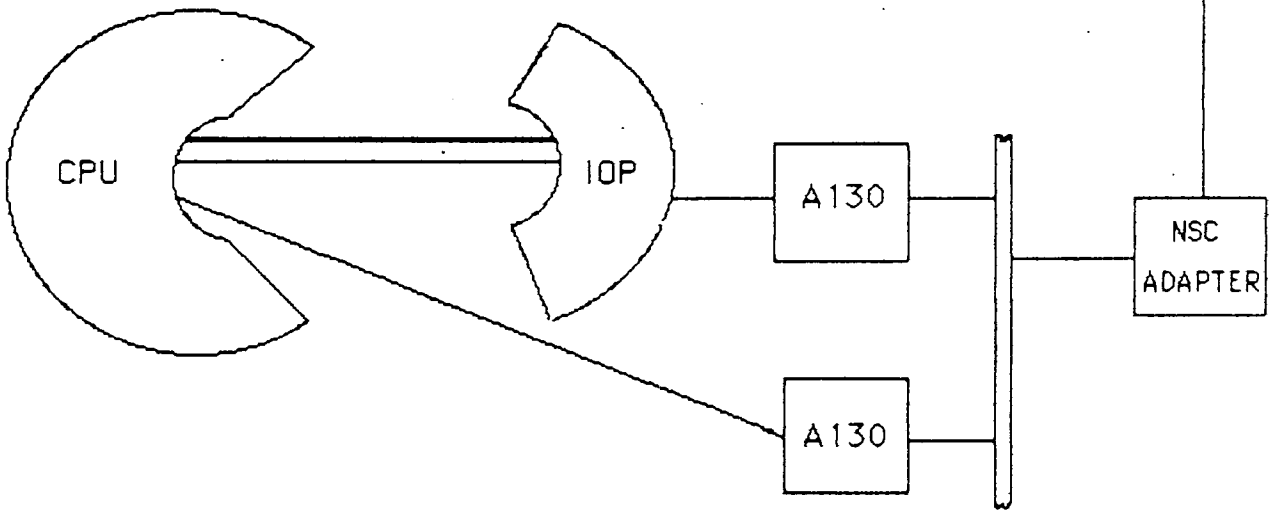
- Dataset Transfer
  - Input and Output datasets
  - Acquire, Fetch and Dispose datasets
- Operator Commands
  - Control
  - Status
- Interactive
- Maintenance Control Unit (MCU)
  - Deadstarting
  - Dumping

NETWORK VS. STATION

Network	Station
Generalized	Serves the CRAY
Hosts are peers	CRAY is the focus
Common protocol	CRAY-oriented protocol
Symetric functions	CRAY-oriented functions
Telecommunication	Channel to channel
Error recovery important	Error rate insignificant



FRONT END STATION



NSC HYPERCHANNEL

NSC HYPERchannel

High-speed digital communications.

Interconnect computers and/or computer peripheral subsystems at full channel rates over long distances.

Maximum rate 50 million bits per second.

Coaxial cable data trunks.

Equipment attached to trunk with adapters.

NSC HYPERchannel ADAPTERS

Function:

Operate coaxial data trunks (up to 4).

Transfer data to and from the coaxial data trunk.

Interface to different computers or peripherals (signal level, protocol).

Processor Adapter - Interfaces a processor to the HYPERchannel network.

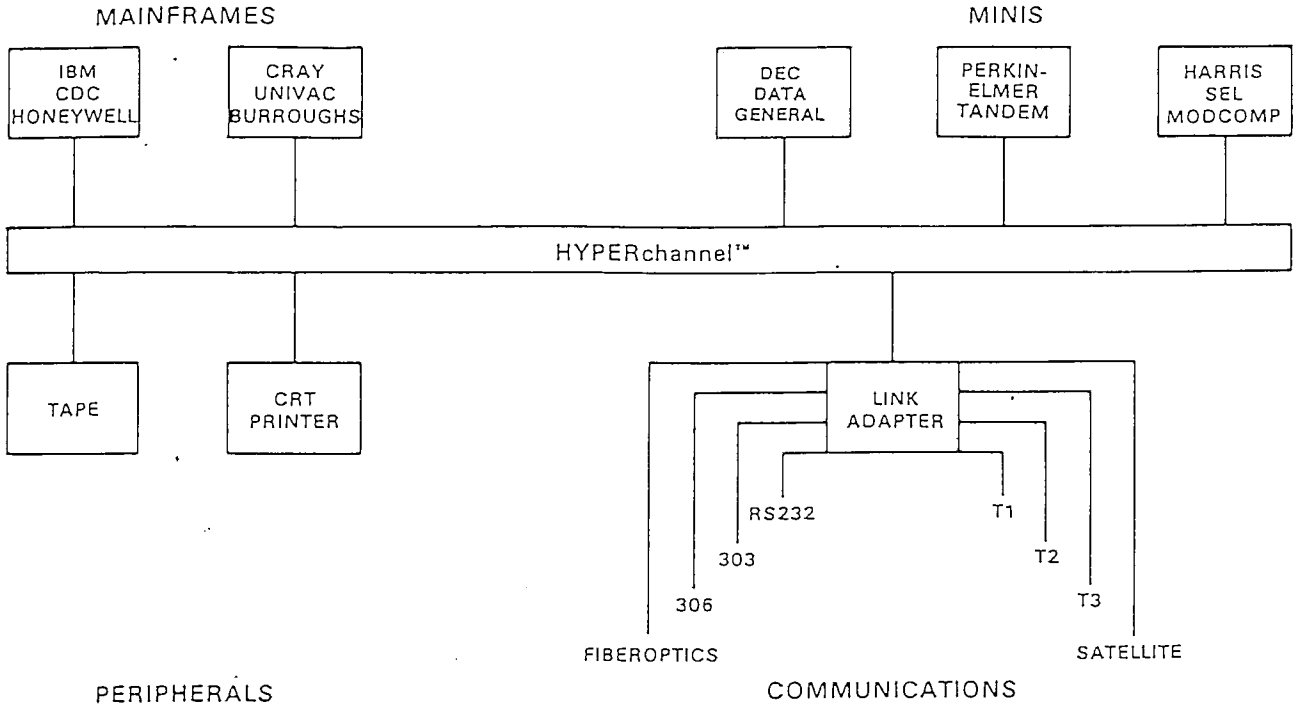
A110 - CDC 6000
A120 - CDC 7600
A130 - Cray Low Speed
A140 - Univac 1100
A220 - IBM 370
A410 - DEC
A420 - DG

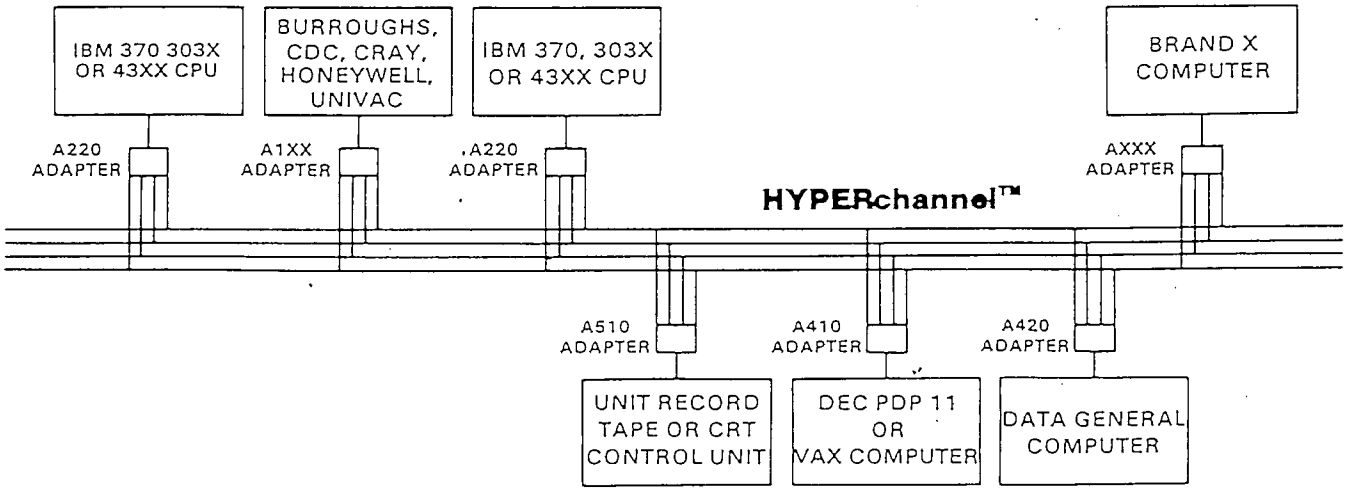
Device Adaptor - Attaches a peripheral subsystem to the HYPERchannel.

Link Adaptor - Allows two HYPERchannel Networks to communicate over Microwave or Bell T1 links (A710).

Satellite Link Adaptor - Allows communication between HYPERchannel networks over a satellite link (A720).

HYPERCHANNEL NETWORK





MESSAGE

A group of related transmissions in the same direction is called a message.

A message consists of:

LCP

0-N Subsegments

LTP (optional)

An entire message is sent in one direction. The computer receiving the message waits for the complete message before it sends a message of its own in the opposite direction.

SEGMENT

Datasets are transmitted in segments.

A segment is always preceeded by an LCP.

A segment may be made up of one or more subsegments.

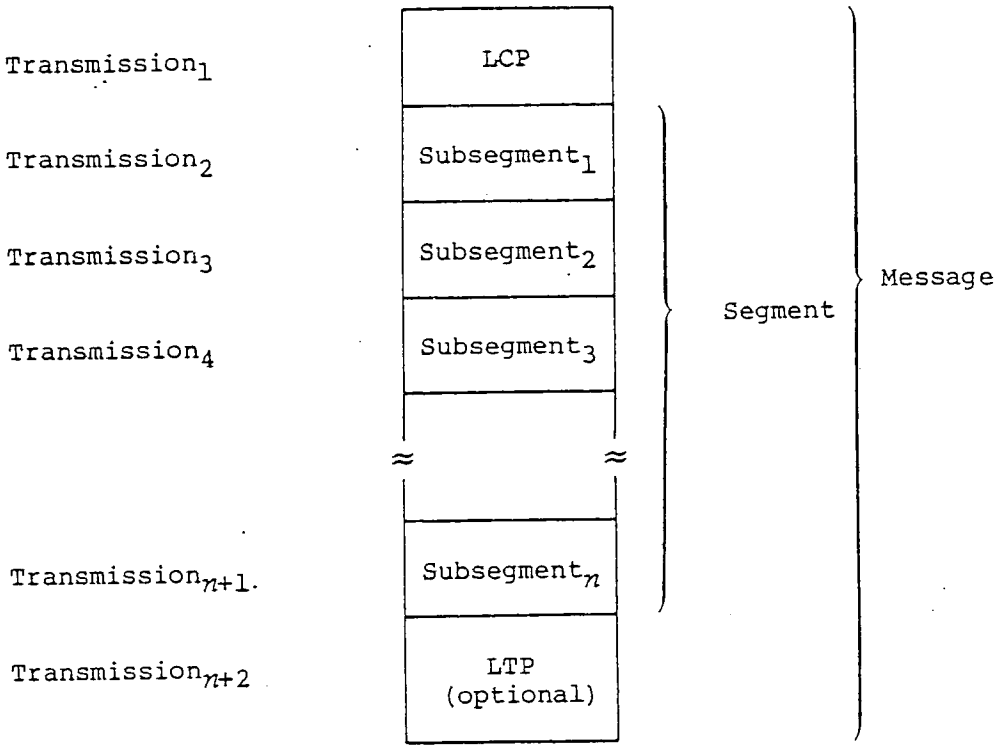
SUBSEGMENT

A block of data that is part of a segment.

It is the largest amount of data that is transmitted across the interface at the driver level.

Multiple of word size on CRAY and front-end machine.

A full subsegment is always transmitted.




```
CRAY STATION.  VERSION X.15, IOS.    L S R                    10/31/84   00:50:52

                LINK STATUS                                        FRAME 0

     QUEUE   ACTIVE   MAXIMUMS    SUBSEGMENT
ID   COUNT    I   O    I   O   AC     #     SIZE   CH,ORD
--   -----    -   -    -   -   --     -     -----  ------
AP       0    0   0    1   2    2     1       512   4,1
LL       0    0   0    8   8   16     1       512   4,11
UX       0    0   0    1   2    3     1       512   4,12
M4       0    0   0    4   4    5     1       512   4,3
DG       0    0   0    1   1    1     1       128   5
U3       0    0   0    8   8   16     1      1024   4,10
M6       0    0   0    2   2    4     1       512   6
IC       0    0   0    0   0    0     1       512   6
UB       0    0   0    0   0    0     1        32   4,14
DX       0    0   0    2   2    4     1       512   4,15
UA       0    0   0    0   0    0     1        32   4,13
II       0    0   0    0   0    0     1       512   4,8
73       0    1   0    2   2    4     1      3600   4,16

>-
>SNAP
```

LINK CONTROL PACKAGE (LCP)

Six 64-bit words.

LCP fields:

DID - destination mainframe identifier; 2 ASCII alphanumeric characters.

SID - source mainframe identifier; 2 ASCII alphanumeric characters.

NSSG - number of subsegments in a segment.

MN - message number.

MC - message code.

MSC - message subcode.

RQP - request pending flag.

STN - stream number.

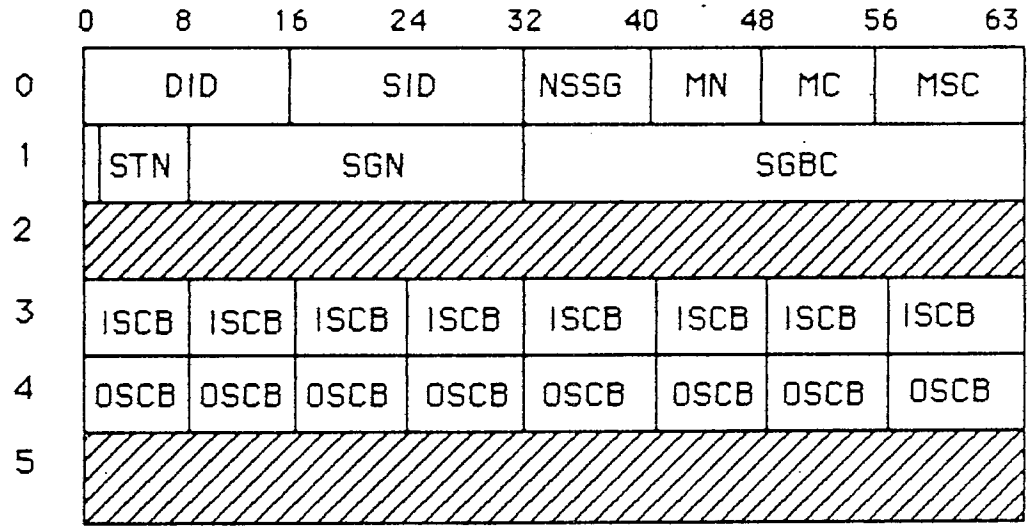
SGN - segment number.

SGBC - number of data bits in segment.

ISCBn - input stream control byte.

OSCBn - output stream control byte.

LINK CONTROL PACKET



Field	Word	Bits	Description
DID	0	0-15	Destination mainframe identifier; 2 ASCII alphanumeric characters.
SID	0	16-31	Source mainframe identifier; 2 ASCII alphanumeric characters.
NSSG	0	32-39	Number of subsegments in segment (0-n)
MN	0	40-47	Message number (0-255)
MC	0	48-55	Message code
MSC	0	56-63	Message subcode
RQP	1	0	Request Pending flag. Station messages are queued by COS, and are sent on receipt of any message not requiring a response, such as a control message. See preceding text.
STN	1	4-7	Stream number (1-8)
SGN	1	8-31	Segment number (0-total number of segments)
SGBC	1	32-63	Number of data bits in segment
$ISCB_m$	3	0-7, 8-15,etc.	Input stream m control byte† (stream ordinal m+8)
$OSCB_n$	4	0-7, 8-15,etc.	Output stream n control byte† (stream ordinal n+8)

STREAM

Software convention for multiplexing dataset transfer.

All messages related to a single dataset transfer including status.

Identified by a stream number.

Number of streams set via front-end parameters.

- Maximum of 16 streams.
- 8 maximum input streams.
- 8 maximum output streams.

STREAM CONTROL BYTES (SCB)

Types of stream control bytes:

- Input stream control bytes (ISCB).
- Output stream control bytes (OSCB).

Function of SCBs:

- Indicates status of a stream.
- Request to take action on the stream.
- Response to a request on a stream.

Included for all defined streams; ignored for undefined streams.

Checked for each message unless there is an error.

I/O relative to the LCP sender.

STREAM CONTROL BYTES

Octal Code	Mnemonic	Request/response	Sender	Receiver
00	IDL	Idle	X	X
01	RTS	Request to send	X	
02	PTR	Preparing to receive		X
03	SND	Sending	X	
04	RCV	Receiving		X
05	SUS	Suspend		X
06	END	End of dataset	X	
07	SVG	Saving dataset		X
10	SVD	Dataset saved		X
11	PPN	Postpone	X	X
12	CAN	Cancel	X	X
13	MCL	Master clear	X	X

STREAM CONTROL BYTE FLOW

Front-end is logged on.

Communications in an idle state.

Front-end sends RTS to the CRAY.

CRAY sends RCV to the front-end.

Front-end sends SND to the CRAY along with the job dataset.

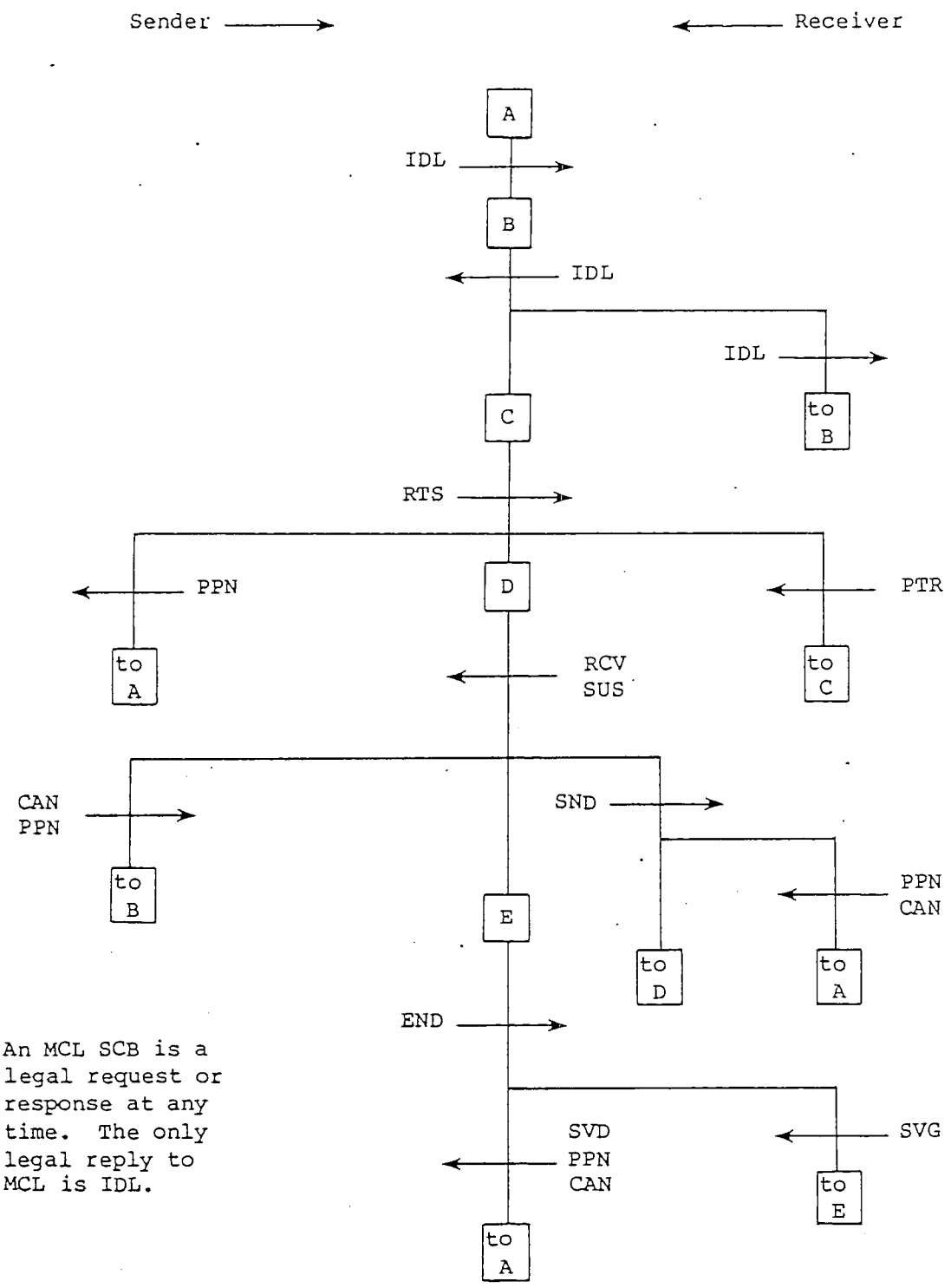
CRAY sends RCV to the front-end while decoding the message and saving the job dataset.

Front-end sends end to the CRAY until the CRAY has saved the dataset.

CRAY sends SVD to the front-end once the dataset has been saved.

The front-end and the CRAY then keep communications open by alternately sending and receiving IDL.

STREAM CONTROL BYTE FLOW



An MCL SCB is a legal request or response at any time. The only legal reply to MCL is IDL.

MESSAGE CODES

A message code specifies a condition on the communications link.

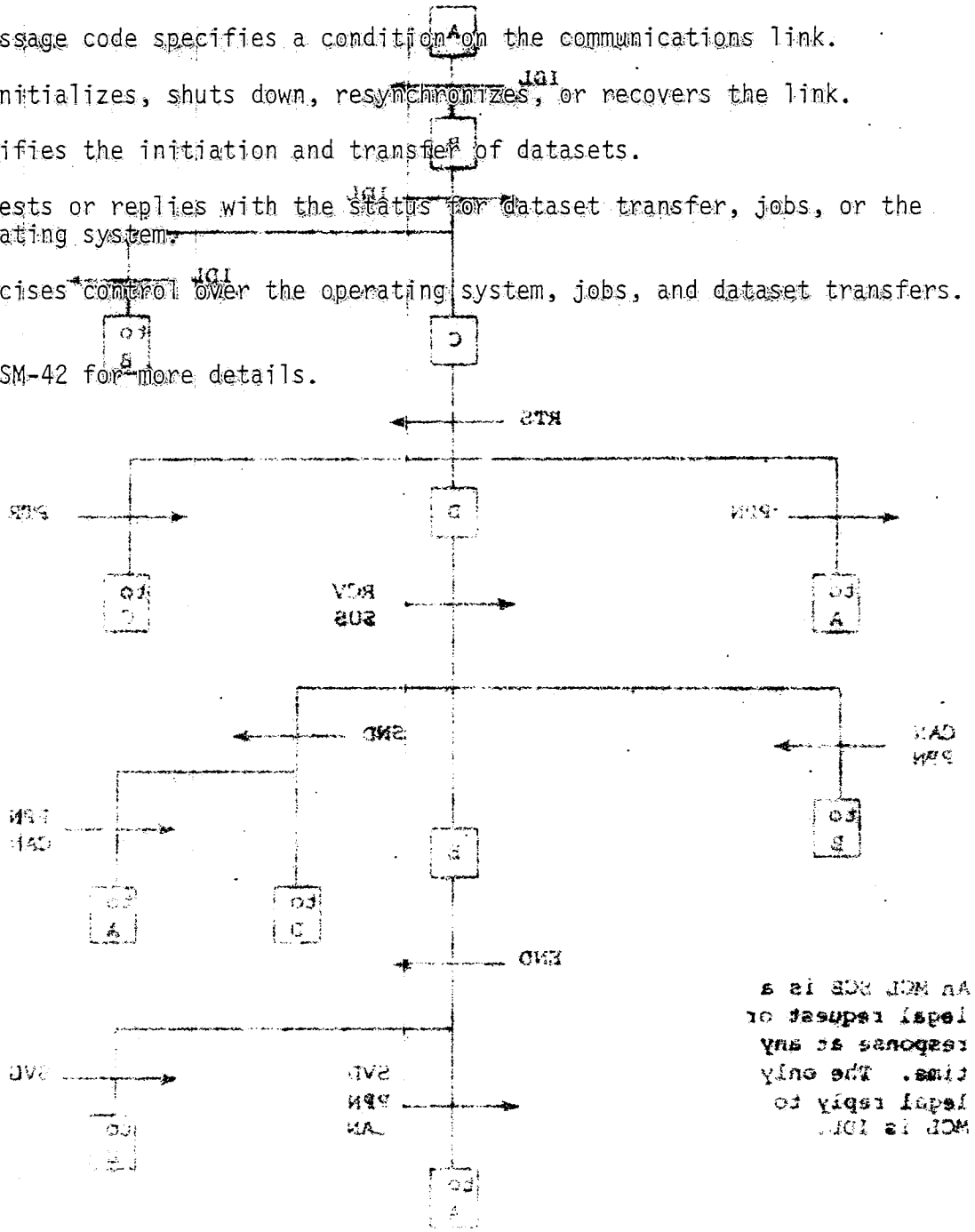
It initializes, shuts down, resynchronizes, or recovers the link.

Specifies the initiation and transfer of datasets.

Requests or replies with the status for dataset transfer, jobs, or the operating system.

Exercises control over the operating system, jobs, and dataset transfers.

See SM-42 for more details.



MESSAGE CODES

Code	Function	Sender		Segment	Stream Required
		Station	COS		
001	Logon	X		X	
003	Logoff	X			
004	Start		X	X	
005	Restart		X		
006	Dataset header	X	X	X	X
007	Dataset segment	X	X	X	X
011	Control	X	X		
012	Message error	X	X		
013	Dataset transfer request		X	X	
014	Dataset transfer reply	X		X	
015	Enter logfile request	X†		X	
016	Enter logfile reply		X	X	
021	Job status request	X†		X	
022	System status request	X†		X	
023	Dataset status request§	X†		X	
024	Link status request	X†		X	
025	Mass storage status request	X†		X	
026	Operator function request	X†		X	
027	Debug function request	X†		X	
031	Job status reply		X	X	
032	System status reply		X	X	
033	Dataset status reply§		X	X	
034	Link status reply		X	X	
035	Mass storage status reply		X	X	
036	Operator function reply		X	X	
037	Debug function reply		X	X	
040	Diagnostic echo request	X†		X	
041	Diagnostic echo reply		X	X	
042	Interactive request	X†		X	
043	Interactive reply		X	X	
044	Statclass request	X†			
045	Statclass reply		X		
046	Station message††		X		
047	Station reply	X†			
050	Tape configuration request	X†			
051	Tape configuration reply		X		
052	Tape job status request	X†			
053	Tape job status reply		X		
054	Configure request	X†			
055	Configure reply		X		
056	Dataset status request (ownership)§	X†	X	X	
057	Dataset status reply (ownership)§	X†	X	X	
070-077	Reserved for site use†††	X	X	X	X

Message sequence for logging on:

Logon -

Start -

Restart -

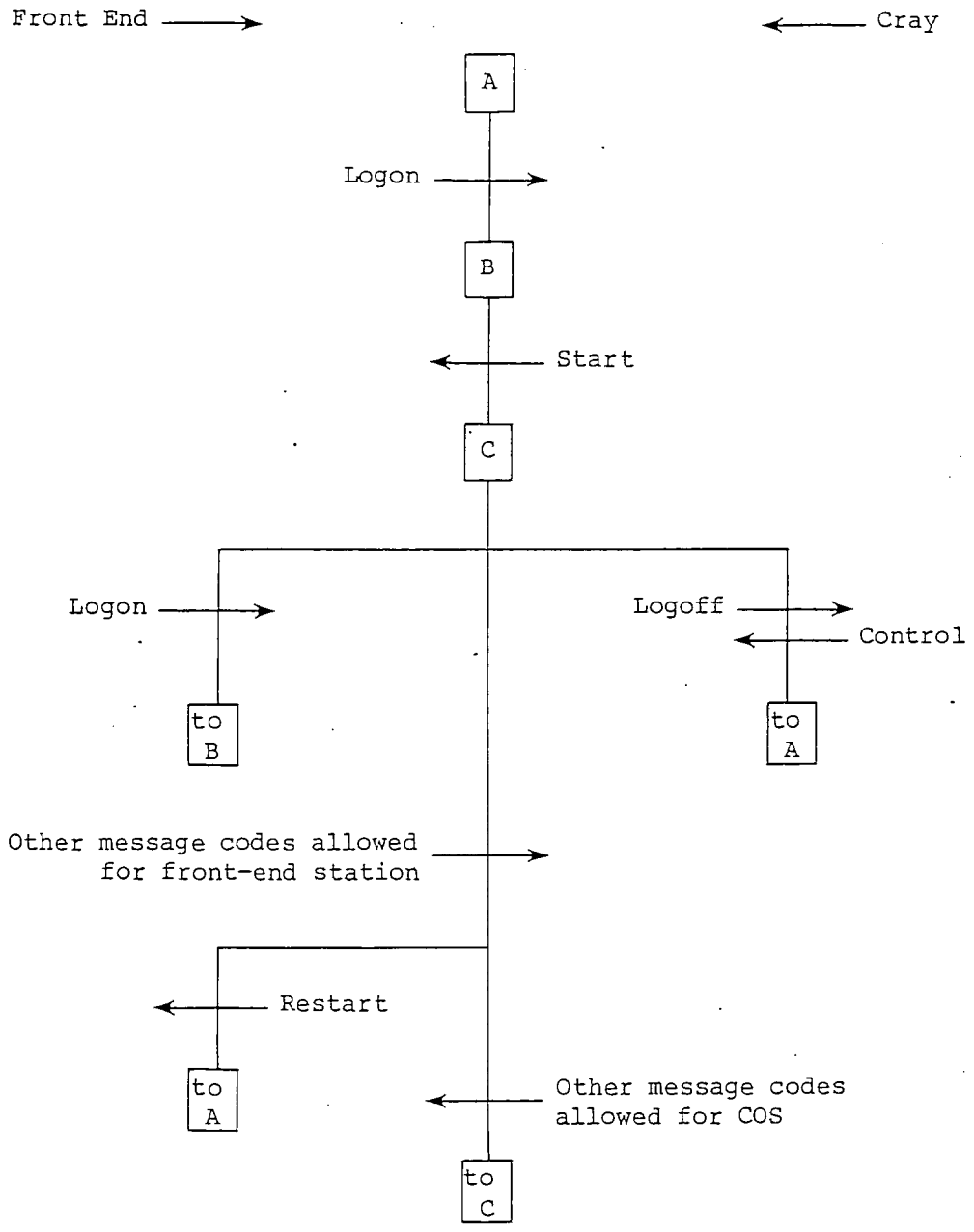
Control -

Logoff -

Control message used as Idle

Poll rate determines period of Idle message

MESSAGE CODE FLOW



MESSAGE CODE FLOW

STATION PROTOCOL

Match the following terms and their definitions:

1. LINK CONTROL PACKAGE	____ Does the word size,logic level and hardware protocol conversion between the Cray and front-end
2. MESSAGE CODE	____ Provides longer distance links between the Cray and a front end
3. DATASET HEADER	____ Links the Station communication to the COS tasks
4. STREAM CONTROL BYTE	____ Field in an LCP containing error messages
5. STATION	____ 6 word message header sent between the station and SCP
6. HYPERCHANNEL	____ Segment zero of a dataset transfer that precedes the dataset
7. FRONT-END INTERFACE	____ Data transferred accross the station Cray link with the LCP
8. SEGMENT	____ Defines what the segment attached to the LCP contains
9. MESSAGE SUB CODE	____ Defines the status of a dataset transfer from the request to the transfer completion
10. STATION CALL PROCESSOR	____ Software that runs on the front-end under the control of the front-end's operating system

STATION PROTOCOL

Match the following terms and their definitions:

1. LINK CONTROL PACKAGE	___ Does the word size, logic level and hardware protocol conversion between the Cray and front-end
2. MESSAGE CODE	___ Provides longer distance links between the Cray and a front end
3. DATASET HEADER	___ Links the Station communication to the COS tasks
4. STREAM CONTROL BYTE	___ Field in an LCP containing error messages
5. STATION	___ 5 word message header sent between the station and SCP
6. HYPERCHANNEL	___ Segment zero of a dataset transfer that precedes the dataset
7. FRONT-END INTERFACE	___ Data transferred across the station for Cray link with the LCP
8. SEGMENT	___ Defines what the segment attached to the LCP contains
9. MESSAGE SUB CODE	___ Defines the status of a dataset transfer from the request to the transfer completion
10. STATION CALL PROCESSOR	___ Software that runs on the front-end under the control of the front-end's operating system

Inter-Task Communication

6

OBJECTIVES

With the aid of all furnished reference materials, following the completion of this Intertask Communication module, the learner should be able to:

1. Describe how a task passes information to another task.
2. Describe the routines used for Synchronous Communication.
3. Describe the routines for Asynchronous Communication.

Impact in COS Rev. 1.14.

TASK TO TASK COMMUNICATION

Synchronous

User will not gain control of CPU until transfer is complete.

TSKREQ

Sends a request and suspends the task until reply is received.

Asynchronous

If a request is made, the user can be reconnected before the data transfer is finished.

PUTREQ

Sends a request to another task.

GETREQ

Gets a request from another task.

PUTREPLY

Sends a reply to a task.

GETREPLY

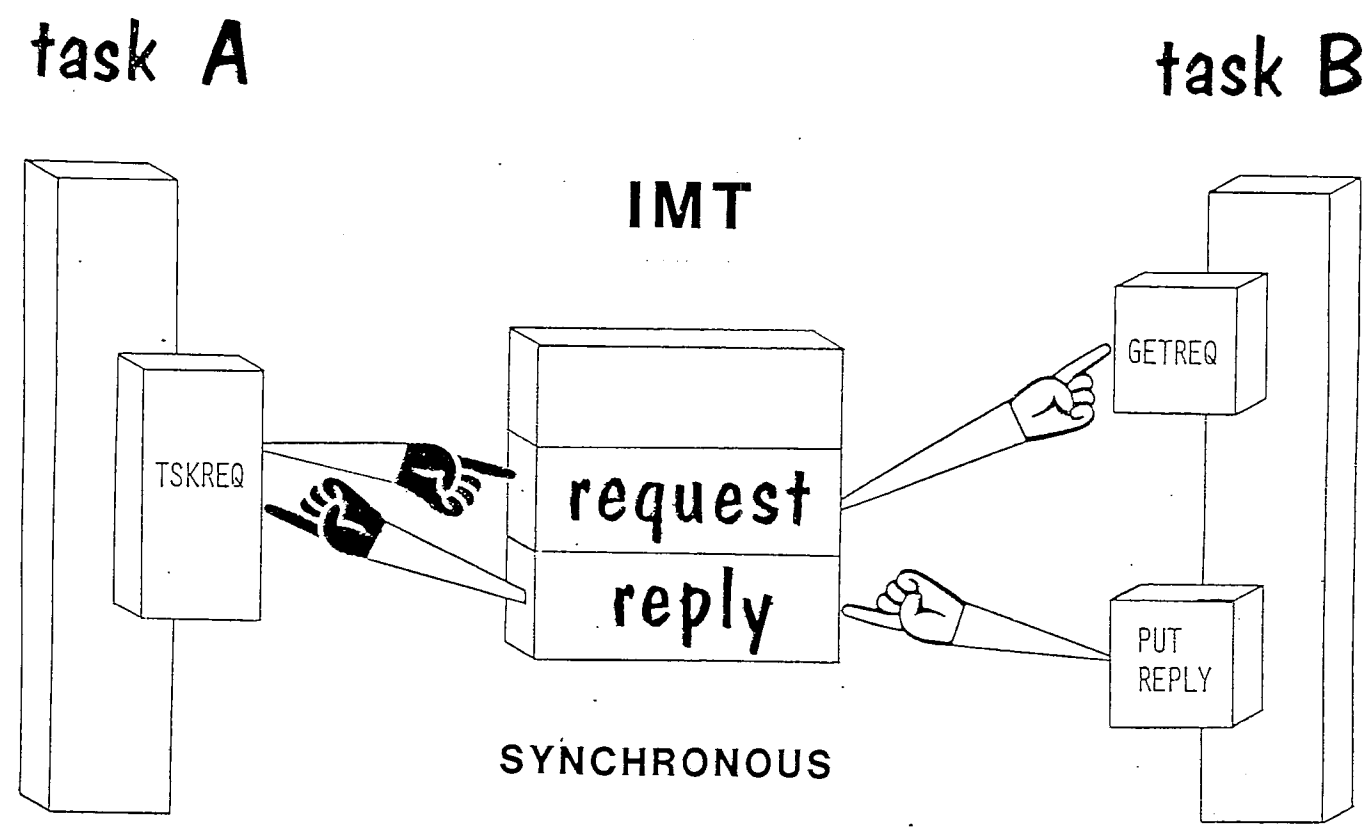
Gets a reply from a task.

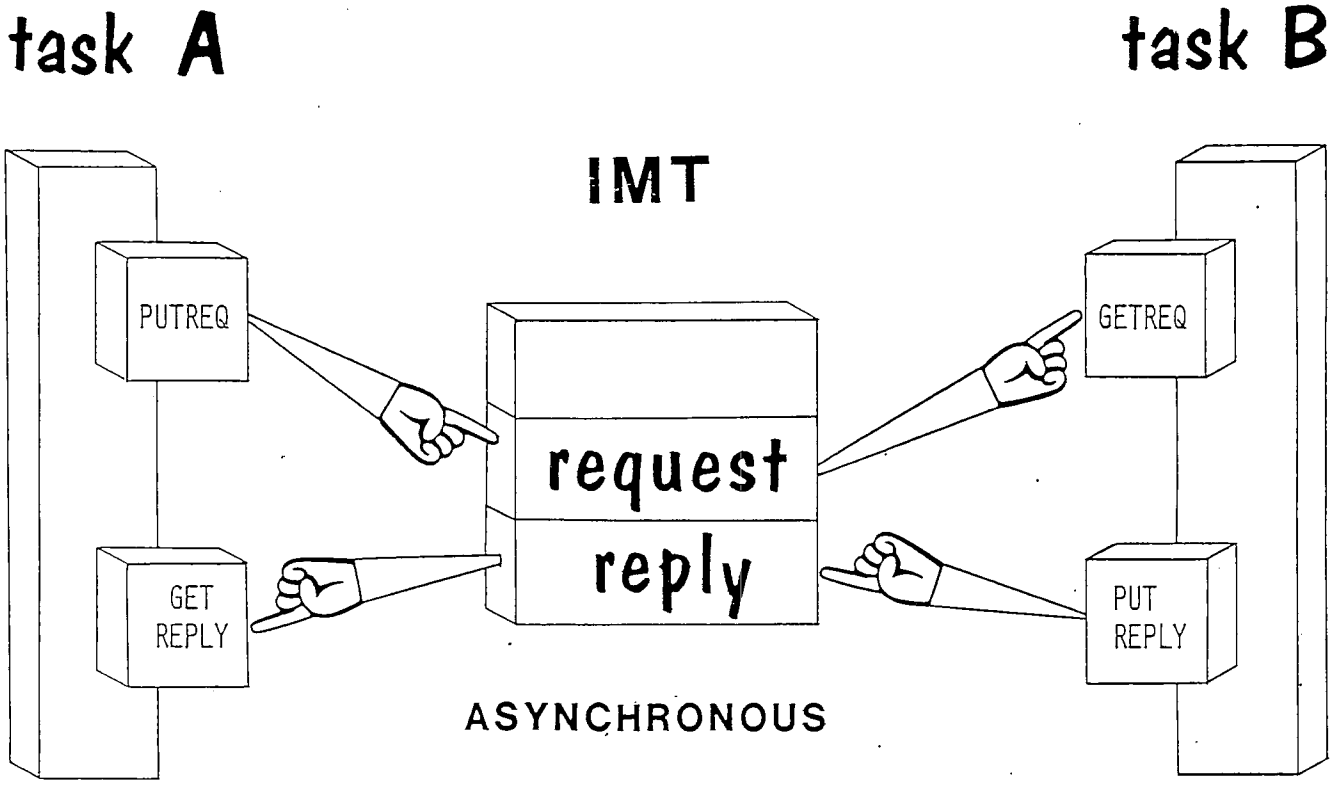
Task puts function code in S1 and S2.

Format is defined by the called task.

Task calls EXEC to activate the task.

Task scheduler in EXEC examines STT to determine highest priority task ready to execute.


task A
task B
IMT
TSKREQ
request
reply
GETREQ
PUT
REPLY
SYNCHRONOUS


task A
task B
IMT
PUTREQ
GET
REPLY
request
reply
GETREQ
PUT
REPLY
ASYNCHRONOUS

INTER-TASK CALLS - SCP CALLS

SCP/Any Task

- via RTSK:
    1. INIT n operator command (n=task id).

SCP/DQM

- via PUTREQ:

    Transfer:

    1. Write RCV datasets to mass storage.
    2. Read SND datasets from mass storage.

    *** SCP does not use CIO for DQM transfer requests ***

    Deallocate:

    1. Cancel SND or RCV datasets.
    2. Deallocate job input dataset.
        - job termination
        - operator KILL
    3. Idle active LXT entry.
        - release all RCV dataset space

    Allocate:

    1. Interactive job input dataset.
        - allocate a dummy dataset

SCP/JSH

- via RTSK:
    1. Notify JSH that a new job in on input Q.
    2. Change PRI of executing job.
    3. Rewrite CSD dataset when job class is turned ON/OFF.
    4. Alter number of JXTs available with LIMIT.

- via TSKREQ:
    1. Job debugging from operator console.
    2. Operator control (DROP, KILL, RESUME, etc.)
    3. Interactive job control.
        - attention
        - abort
    4. ACQUIRE, DISPOSE failures (abort job).

SCP/JCM

- via TSKREQ:

1. Assign job to a class.

   - job input dataset has arrived

2. Reassign job class; operator changed ...

   - front-end ID & TID (ENTER or ROUTE command)
   - priority, time limit (ENTER command)

3. Assign class to a job.

   - ENTER CLASS command

*** JCM is called ONLY for jobs on the input Q ***

SCP/MSG

- via TSKREQ:

1. Record operator type-ins (system and user logs).
2. Error message for DISPOSE disk read failure.
3. Log dataset transmission and reception messages.

SCP/PDM

- via PUTREQ:

1. See if dataset to ACQUIRE is on Cray.
2. Operator DATASET command processing.
3. Save spooled input datasets.
4. Delete spooled output datasets.

SCP/TQM

- via PUTREQ:

1. Process operator CONFIG command for XIOP tapes.

INTER-TASK CALLS - JCM CALLS

JCM/MSG

- via TSKREQ:

1. Job class assignment message to systemlog.
2. Fatal job class definition structure error message.

INTER-TASK CALLS - MEP CALLS

MEP/MSG

- via PUTREQ:

1. Forward EXEC messages to MSG.

INTER-TASK CALLS - JSH CALLS

JSH/DQM

- via PUTREQ:

Allocate:

1. Allocate job roll dataset.

Deallocate:

1. Release roll dataset space:

    - job terminates
    - larger DAT needed (job expands)

Transfer:

1. Write $ROLL.
2. Write CSD dataset.
3. Job roll in/out.

JSH/MSG

- via TSKREQ:

1. Write messages to user and system logs.

JSH/JCM

via TSKREQ:

1. Classify job which is to be rerun.
2. Re-order input Q when new job class structure invoked.

JSH/EXP

- via JTEP word:

1. Abort request (A$xxx) passed to JSH from another task.
2. Bad job class structure invoke (A$BINV).
3. Initialize job (JXEPC=0).
4. Time limit expired (A$TLE).
5. Too much memory requested (A$GSY).

INTER-TASK COMMUNICATION TABLES

STP contains three areas used for intertask communication.

The Intertask Communication Table (ICT)

The request queue table has one entry per task.

Each entry defines the address and size of the associated IMT.

All requests of a particular task are placed on the queue pointed to by it's ICT request entry.

The reply queue table has one entry per task

Each entry defines the address and size of the associated IMT.

All replies to a particular task are placed on a queue pointed to by its ICT reply entry.

The Intertask Queuing Table (IQT)

A bit map which defines available and active entries in the IMT.

Initialized to a given size at assembly time.

Number of bits at beginning of bit map is set to zero.

The rest of the bit map is set active.

The Intertask Message Table (IMT)

The IMT immediately follows the IQT and is the message's.

An IMT entry is three words:

Two words from S1 and S2 of the calling task.

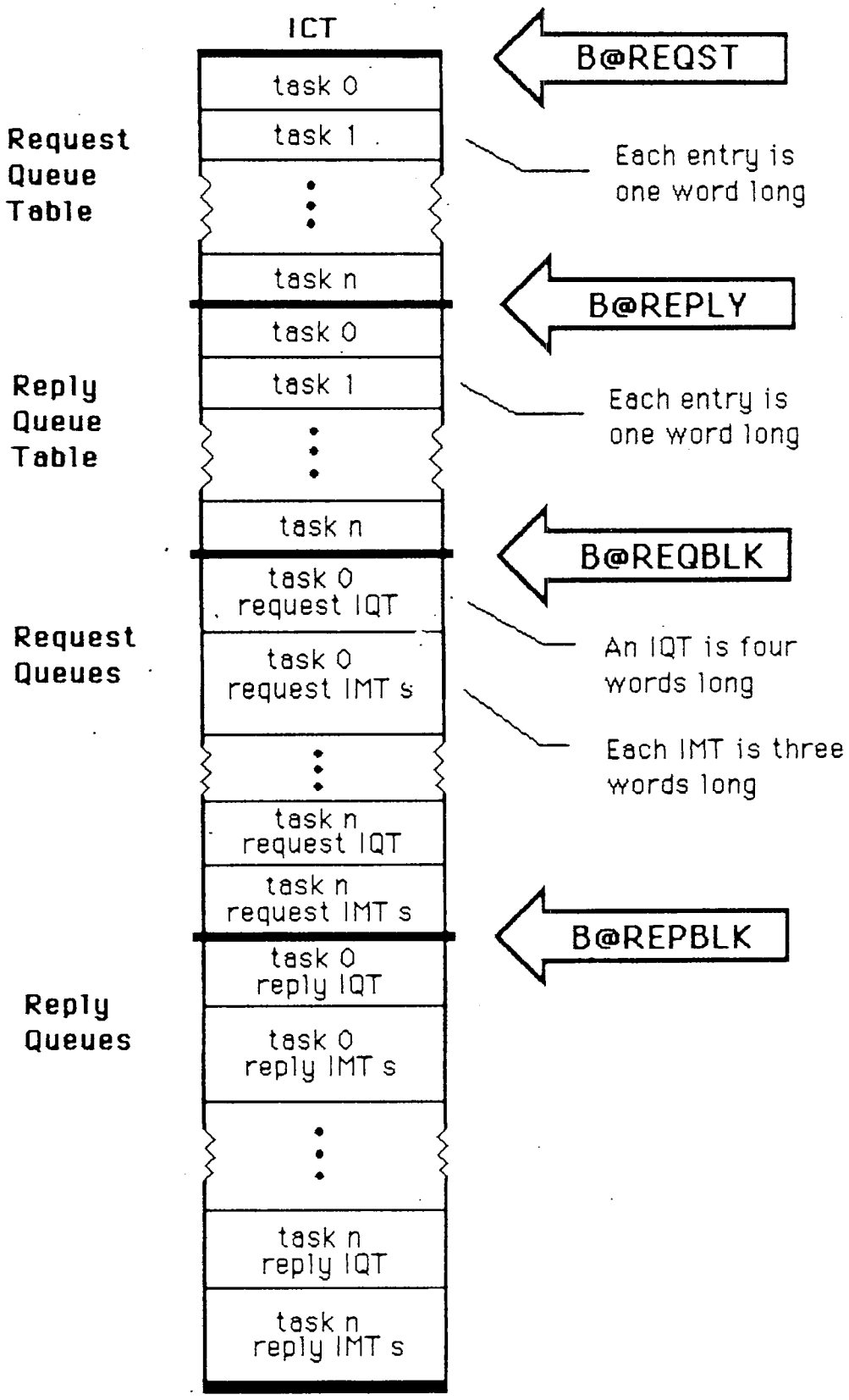
A word with the calling task ID.

One IMT entry for each possible request/reply in the IQT.

One task requests action of another by queuing a message on the request queue of the receiving task.

The second task replies by queueing a message on the reply queue of the first task.

Intertask Communication Table Details



PUTREQ:

Sets STPLCK
Allocates first available IMT in the called task's request queue
Puts the calling task's S1, S2 and task ID in the IMT
Determines whether to make an explicit or implicit EXEC request to ready the called task using the following logic:
IF Callee's priority caller's priority and Callee suspended threshold
CALL EXEC TASK-READY
ELSE make an explicit EXEC TASK-READY REQUEST
An explicit request is made by setting up S6 and S7 and doing EX
An implicit request is made setting the bit corresponding to the task ID in the System Task Activation word (STA)
On an exchange to EXEC (interrupt, etc.) - any task whose bit is set in the STA will be readied

GETREQ:

Sets STPLCK
Finds first IMT with the corresponding IQT bit set
Sets the IQT bit to zero
Gives the intertask message to the receiving task in S1 and S2
A2 contains the requesting task ID
Unlocks STPLCK

PUTREPLY:

Sets STPLCK
Allocates the first available IMT in the receiving task request Queue
Puts the replying task's S1 and S2 and task ID in the IMT
Determines whether to make an explicit or implicit EXEC request to ready the receiving task
Unlocks STPLCK

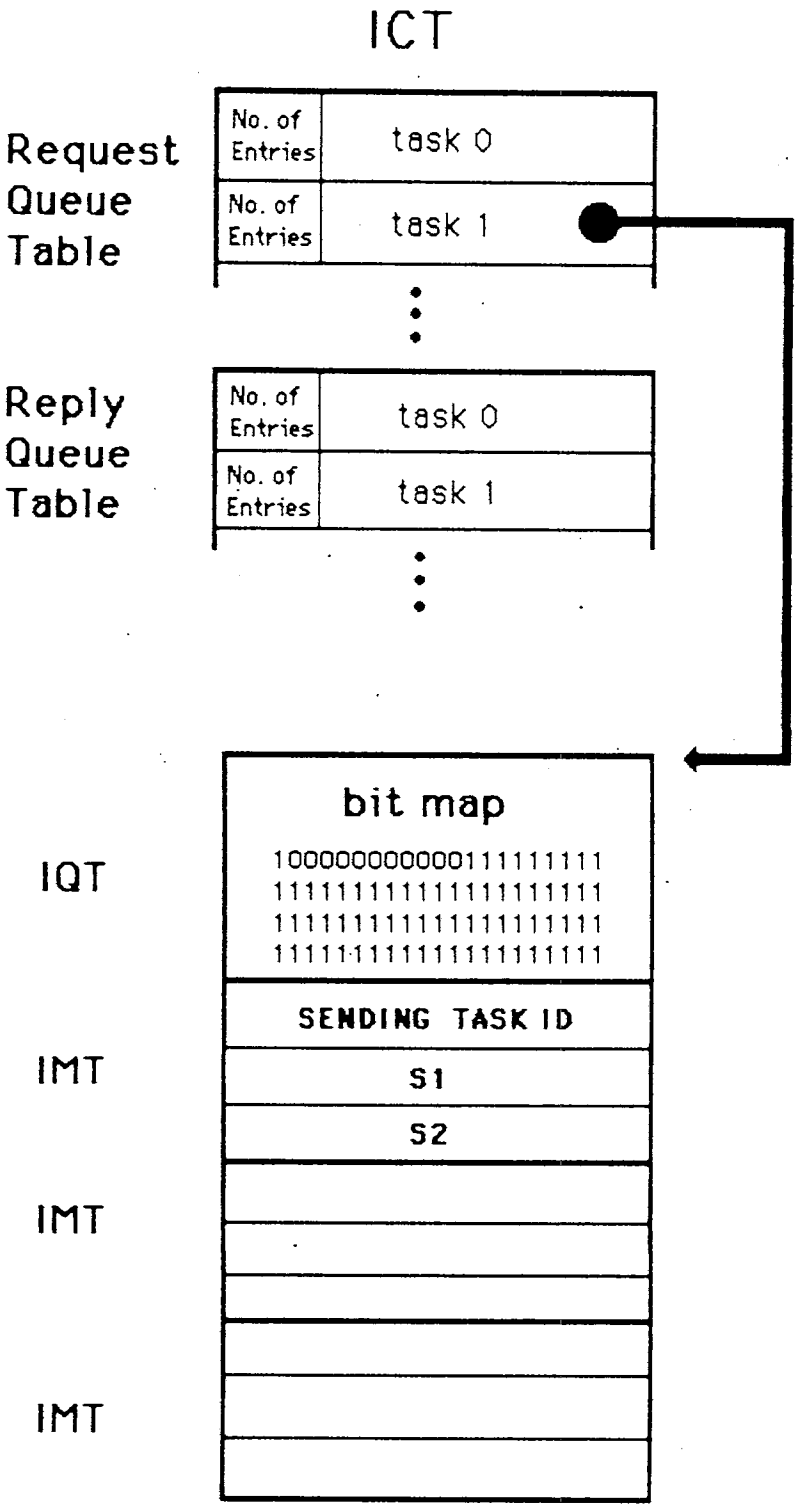
GETREPLY:

Sets STPLCK
Finds the first available IMT with a corresponding IQT bit set
Zeros the IQT bit
Gives the receiving task the intertask reply in S1 and S2
A2 contains the sending task ID
Unlocks STPLCK

TSKREQ:

Used for asynchronous intertask communication
First it does the equivalent of a PUTREQ
Then the task is suspended until a reply is received
Finally it does the equivalent of a GETREPLY
A called task still uses GETREQ and PUTREPLY to respond to TSKREQ

Intertask Communication Tables



INTER-TASK HISTORY TRACE TABLE

Contained in STPTAB as symbol ITCT

Each entry is eight words

- Routine name
- Elapsed clock periods since last entry
- Real time clock
- Task ID of current STP task (task making request or reply)
- S1 contents
- S2 contents
- Task ID of requested task (task receiving request or reply)

Next entry pointed to by header field

For S register breakdown, see SM-40

EXP	SM-40, page 8-3 to 8-35
JSH	SM-40, page 9-36 to 9-61
PDM	SM-40, page 10-2 to 10-13
MSG	SM-40, page 11-4 to 11-6
MEP	SM-40, page 12-2
DEC	SM-40, page 13-1
JCM	SM-40, page 15-2 to 15-5
TQM	SM-40, page 17-13 to 17-20
IQM	SM-40, page 16-26 to 16-27
FVD	SM-40, page 19-1
DQM	SM-40, Page 6-1

DQM TRACE

DQMT points to start of table

100 octal words long

TQM TRACE

TQMT points to start of table

21000 octal words long

Intertask Communications
History Trace Message Format

Routine Name	Elapsed Clock Periods Since Last Entry	Real Time Clock	Task ID of Current STP Task
S1	S2	Task ID of Requested/ Requesting Task	

Interbox Communications

History Trace Message Format

Routine Name	Elapsed Clock Periods Since Last Entry	Real Time Clock	Task ID of Current STP Task
S1	S2	Task ID of Requested/ Requesting Task	

INTERTASK COMMUNICATION

Match the following terms and their description:

1. IMT	____ Used to make an executive request
2. PUTREQ	____ Routine to initiate a syncronous request from another COS Task
3. TASKREQ	____ Used by the user to make system action requests of EXP
4. ITCT	____ Request/Reply packet between STP tasks
5. SØ & S1	____ Registers used for task to task requests/replies
6. S1 & S2	____ Trace table for all task to task requests/replies
7. S6 & S7	____ Asyncronous Request to another Task

INTERTASK COMMUNICATION

Match the following terms and their description:

1. IMT

2. PUTREQ

3. TASKREQ

4. ITCT

5. SQI & SI

6. SI & SQ

7. SQ & SZ

___ Used to make an executive request

___ Routine to initiate a synchronous request from another CUS Task

___ Used by the user to make system action requests of EXP

___ Request/Reply packet between STP tasks

___ Registers used for task to task requests/replies

___ Trace table for all task to task requests/replies

___ Asynchronous Request to another Task

Inter-IOP Communication

7

Inter-IOP Communication

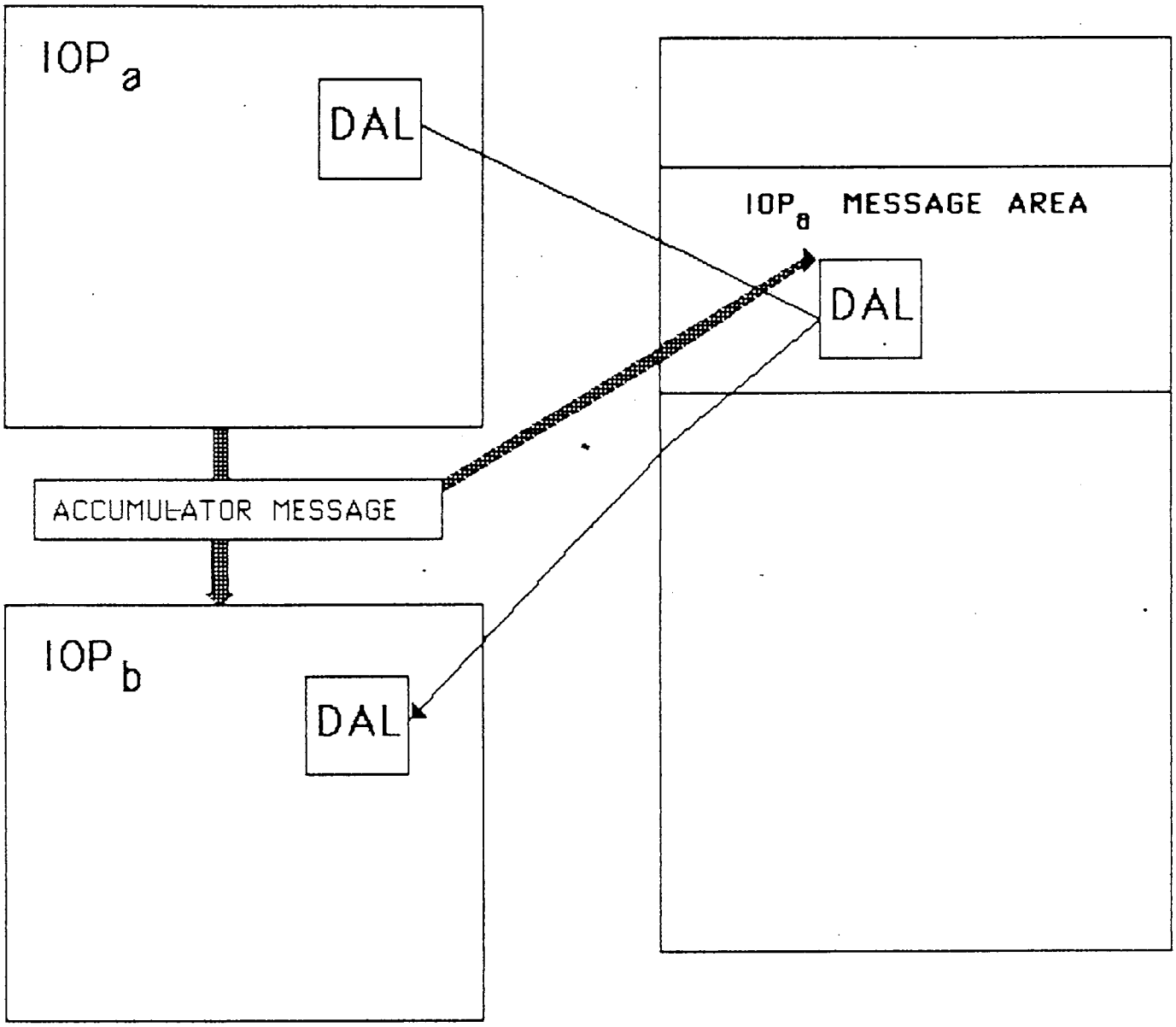
7

OBJECTIVES

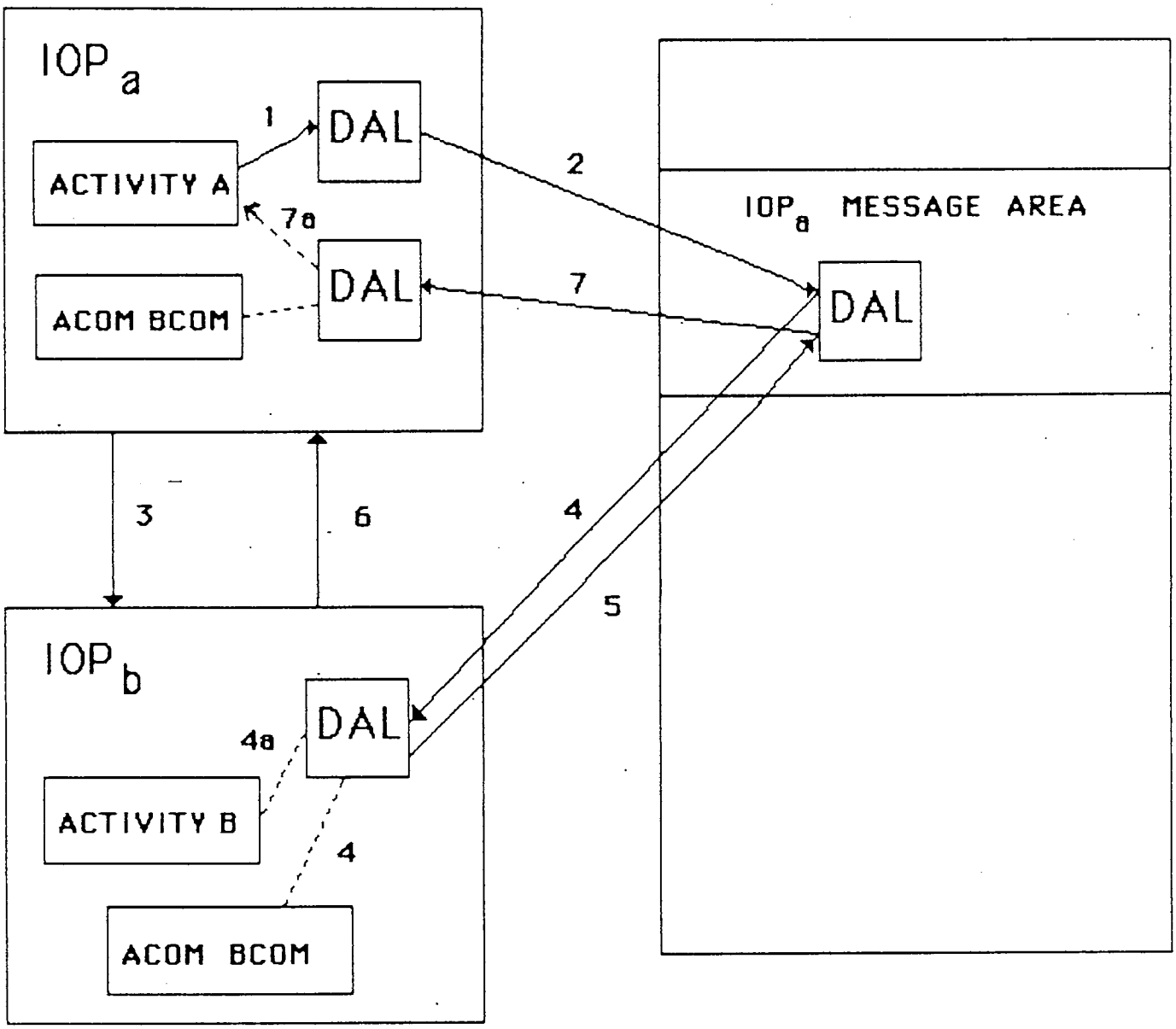
With the aid of all furnished reference materials, upon completion of this module the learner should be able to:

1. Breakdown a A-A message.
2. Locate the associated DAL.
3. Describe how an activity is created.
4. Explain the control of one activity over another.

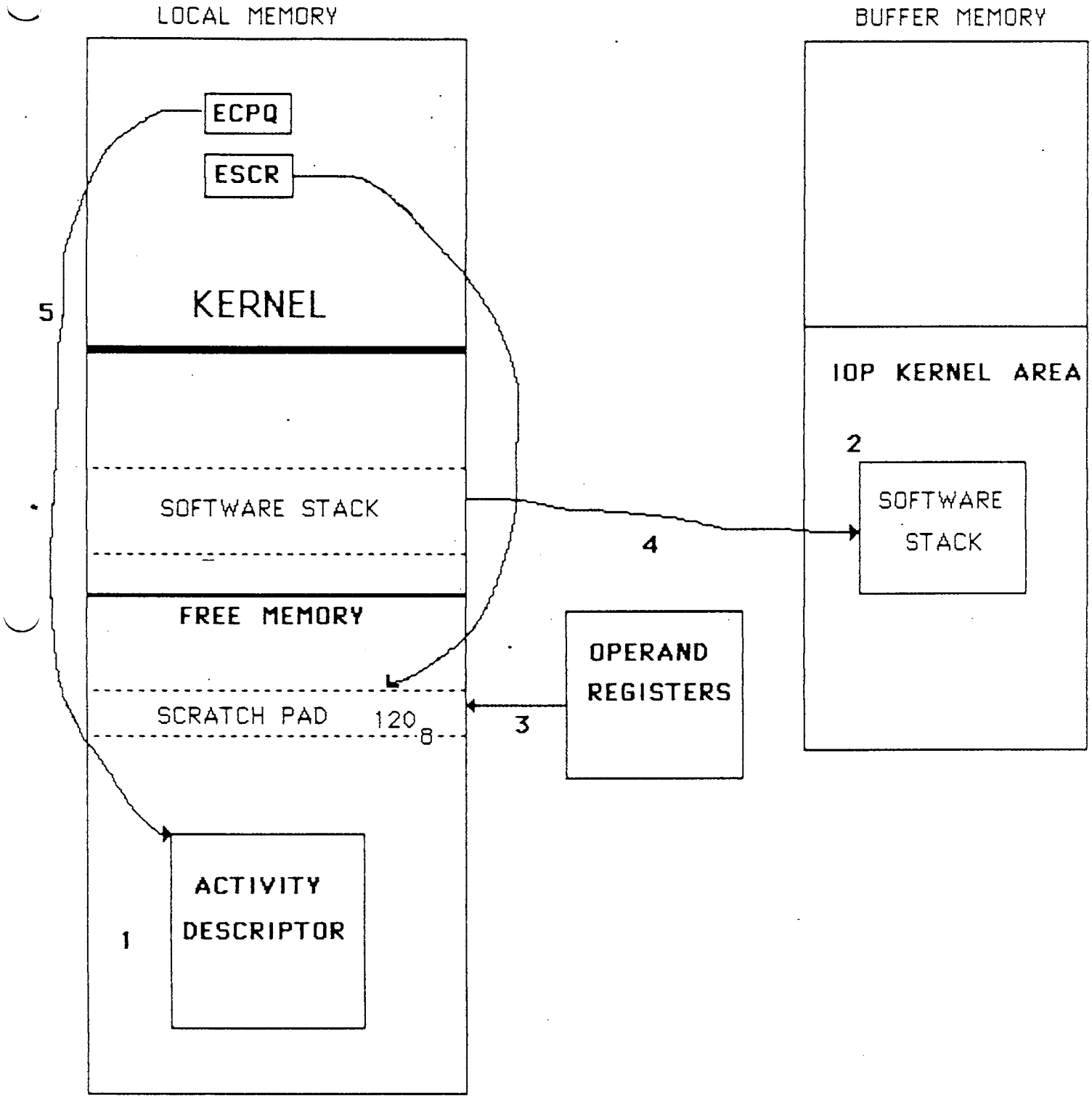
INTER-IOP COMMUNICATION



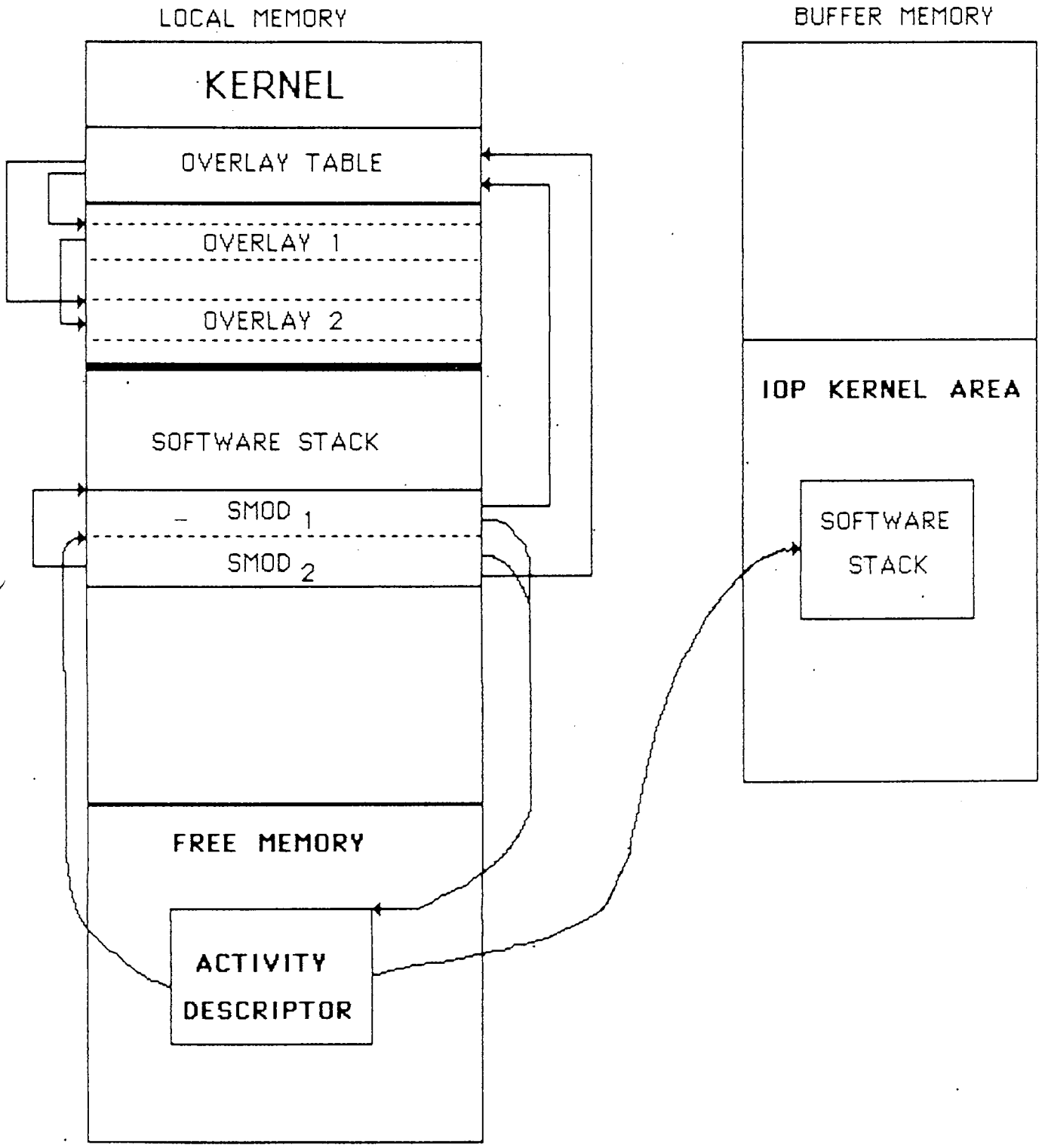
DISK ACTIVITY LINK FLOW



ACTIVITY CREATION



ACTIVITY TABLE LINKAGE



INTER-IOP COMMUNICATION

1. AMSG	____ Activates a suspended activity in another IOP
2. POPCELL	____ Processes Interprocessor DAL s that are requesting TAPE
3. DISK ACTIVITY LINK	____ Kernel service request to activate another overlay
4. AWAKE	____ 40 octal parcel message passed between IOP s through Buffer memory
5. CALL	____ Processes DAL requests from MIOP for DD-49 activities
6. ICOM	____ Processes the ALERT service request creating a POPCELL
7. BCOM	____ Processes DD-29 DAL s from MIOP, BIOP or DIOP
8. ACOM	____ Points to the Interprocessor DAL in Buffer Memory
9. INTER-IOP A-A MESSAGE Interrupt Handler	____ Kernel Service request used by MIOP to initiate a Buffer Memory to/from Central Memory for the Concentrator
10. STATIO	____ QUEUE and Activity descriptor information for the ALERT,AWAKE, ASLEEP, RESPOND mechanisim

COS Dump Analysis

8

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon completion of the COS Dump Analysis module, the learner should be able to:

1. Explain how the CRAY1SYSTEMDUMP is created.
2. Determine the EXEC Hang Address.
3. Determine the Hang Address Macro.
4. Locate the Hang Macro in the Listings.
5. Locate and analyze the History Trace Buffer.
6. Locate and analyze a COS Table.
7. Locate and analyze Memory Error Log.
8. Find the EXEC or Task execution environment.

CREATION OF A SYSTEM DUMP

Why is a dump taken:

CRAY HALT appears on Kernel Console

IOP HALT appears on the Kernel Console

Consoles fail to respond to operator commands

Cray system appears hung or dead

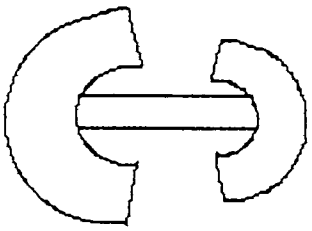
Cntl-D was pushed

To get a memory copy, SYSDUMP must be executed

SYSDUMP runs from MIOP and copies system memories to the master device

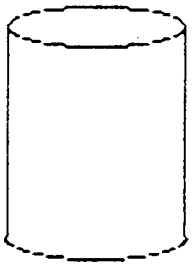
STARTUP processes buffer and creates permanent dataset

FORMATTED SYSTEM DUMPS



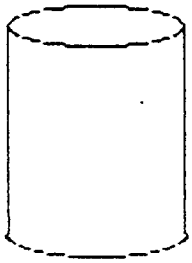
CRAY SYSTEM CONTENTS

CNTL-D initiates SYSDUMP



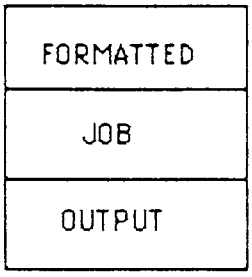
DISK SCRATCH BUFFER

STARTUP processes buffer



CRAY1SYSTEMDUMP

PDN=
OWN=
ED=
R=



FORMATTED DUMP
IN $OUT

JOB with FDUMP directives

FORMATTING OF A SYSTEM DUMP

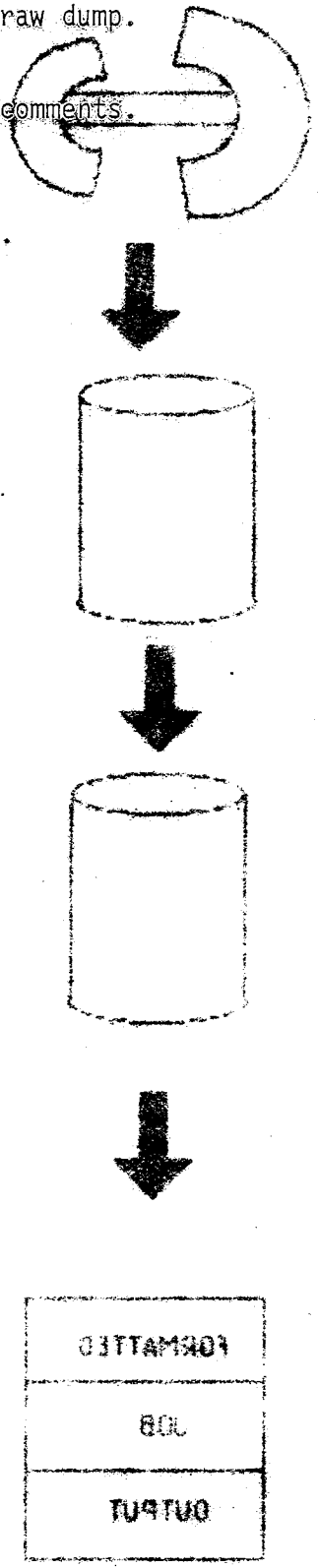
Requires a working Cray

If no Cray is available then the dump should have been a raw dump. (File 2 of tape or disk)

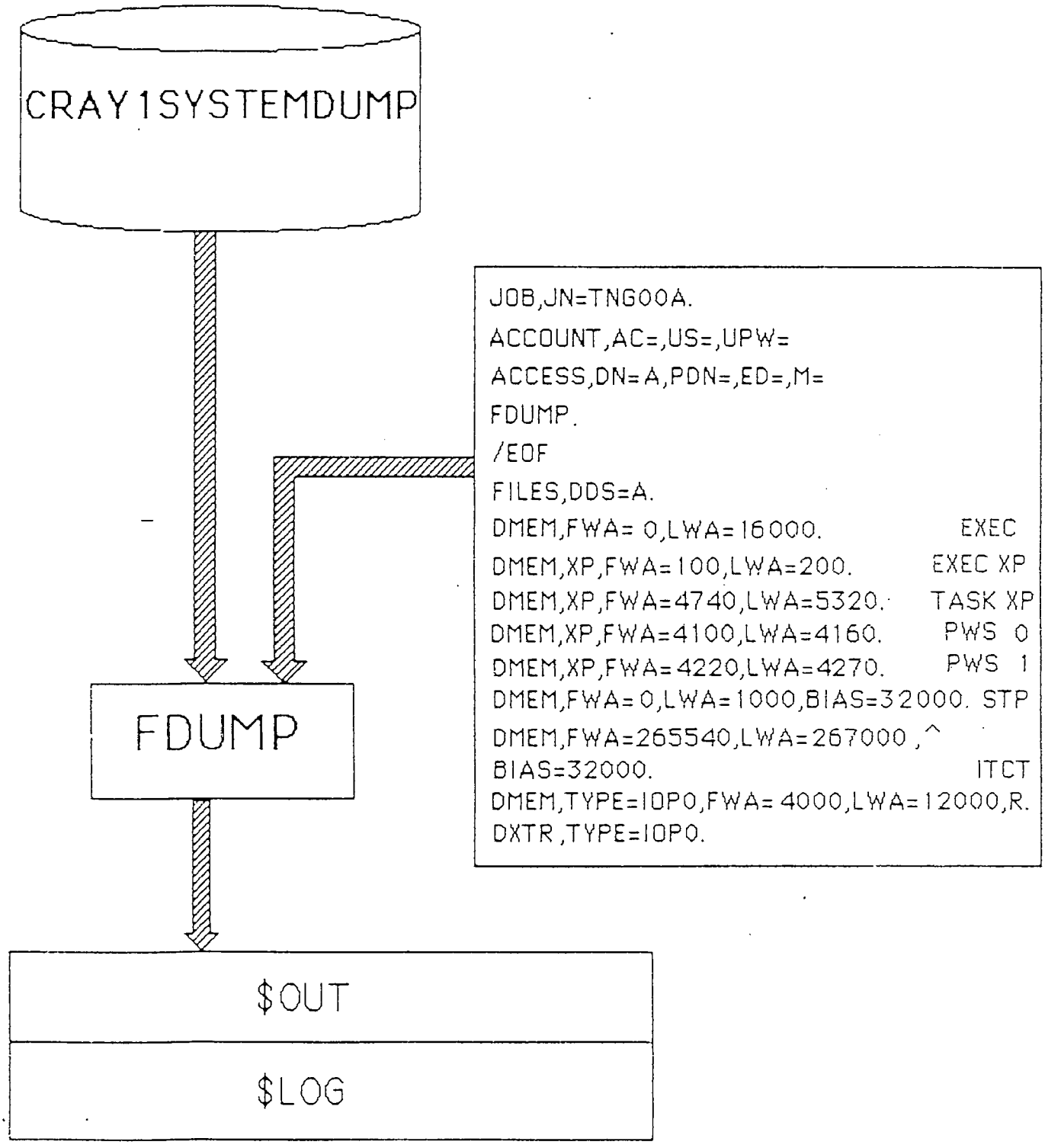
Directives tell the program FDUMP what to dump and allow comments.

Different directives for the mainframe and the IOP.

Initially dump Exec STOP Buffer and STP Hang Message area.



FDUMP


CRAY1SYSTEMDUMP
JOB,JN=TNG00A.
ACCOUNT,AC=,US=,UPW=
ACCESS,DN=A,PDN=,ED=,M=
FDUMP.
/EOF
FILES,DDS=A.
DMEM,FWA=0,LWA=16000. EXEC
DMEM,XP,FWA=100,LWA=200. EXEC XP
DMEM,XP,FWA=4740,LWA=5320. TASK XP
DMEM,XP,FWA=4100,LWA=4160. PWS 0
DMEM,XP,FWA=4220,LWA=4270. PWS 1
DMEM,FWA=0,LWA=1000,BIAS=32000. STP
DMEM,FWA=265540,LWA=267000,^
BIAS=32000. ITCT
DMEM,TYPE=IOP0,FWA=4000,LWA=12000,R.
DXTR,TYPE=IOP0.
FDUMP
$OUT
$LOG

REASONS FOR A HANG

The operating system detects an inconsistency.

Could be hardware or software detected

- XP flag or DN flag could have set
- A program is too large

Could be hardware or software caused

- A channel error
- A register overwritten

DUMP ANALYSIS PROCEDURE

Locate and read the Hang Messages

Locate the Hang Macro in the Listing

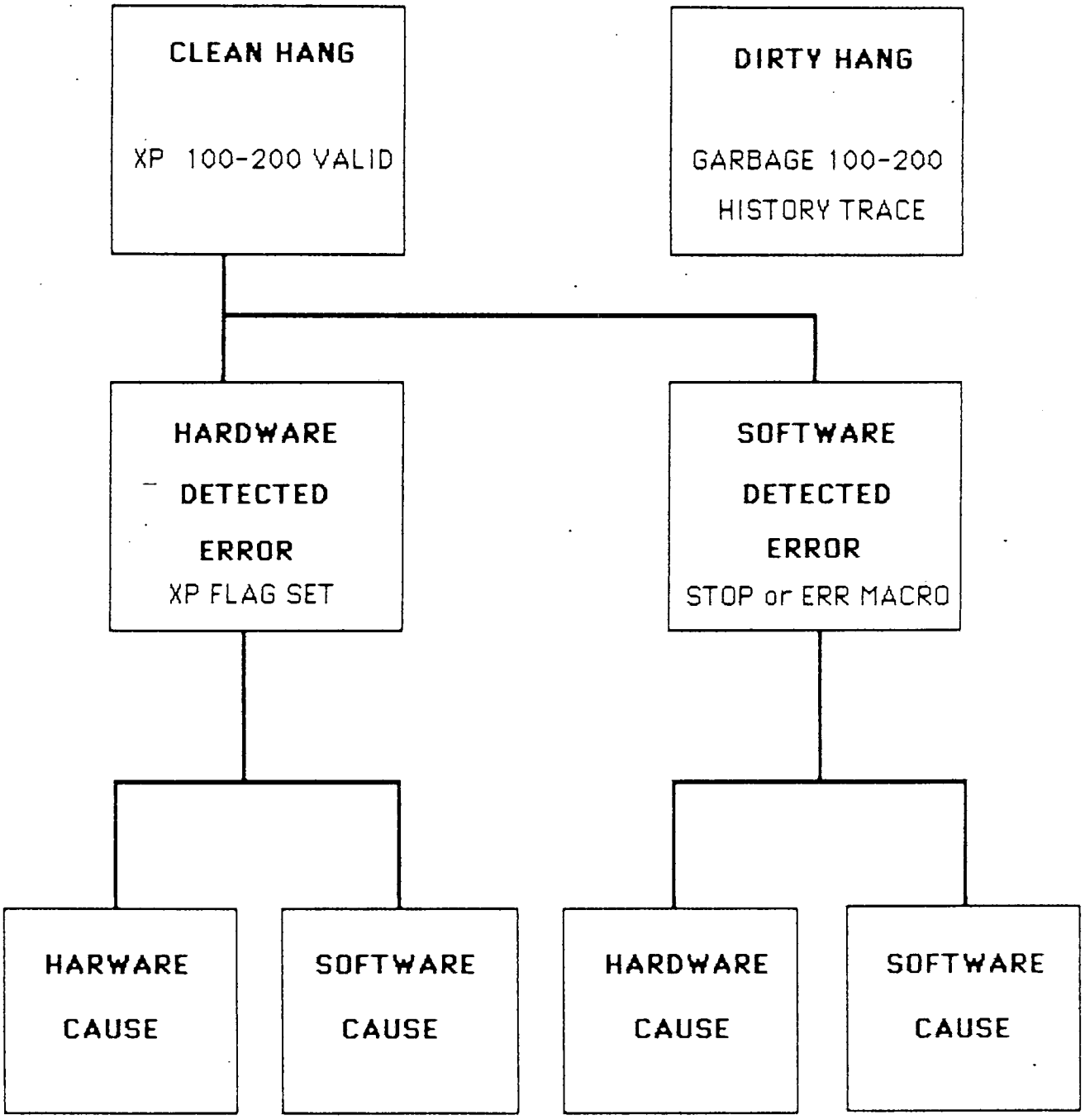
Analyze the Histrace Buffer

Decide on the program to begin with

Locate the necessary Program Executing Environment

Determine the reason for the error hang conditions

COS HANG


CLEAN HANG
XP 100-200 VALID
DIRTY HANG
GARBAGE 100-200
HISTORY TRACE
HARDWARE
DETECTED
ERROR
XP FLAG SET
SOFTWARE
DETECTED
ERROR
STOP or ERR MACRO
HARWARE
CAUSE
SOFTWARE
CAUSE
HARDWARE
CAUSE
SOFTWARE
CAUSE

This is a list intended to serve as a guide to analyzing CRAY1SYSTEMDUMP'S.

Record any operator interactions with COS or IOS at the time of the crash, record date and time in the crash log.

Examine KERNEL HALT MESSAGE to determine if COS halted IOS .

Take a dump according to CRAY RESEARCH System's analyst needs.

Run FDUMP to format important areas of the dump.

Dump the EXEC table area, STP table pointer area.

Make sure listing matches dump assembly date.

Examine exchange package area 100-200 of central memory.

Determine interrupted exchange package from A1 of hang XP.

If a channel error examine A2 and A3 of hang XP.

Determine if hang looks like a valid EXEC hang.

Examine the Stop Buffer around 1400 of central memory.

Check assembly date and revision against listings.

Determine hang message and number.

Read P and locate the $STOP macro in the listing.

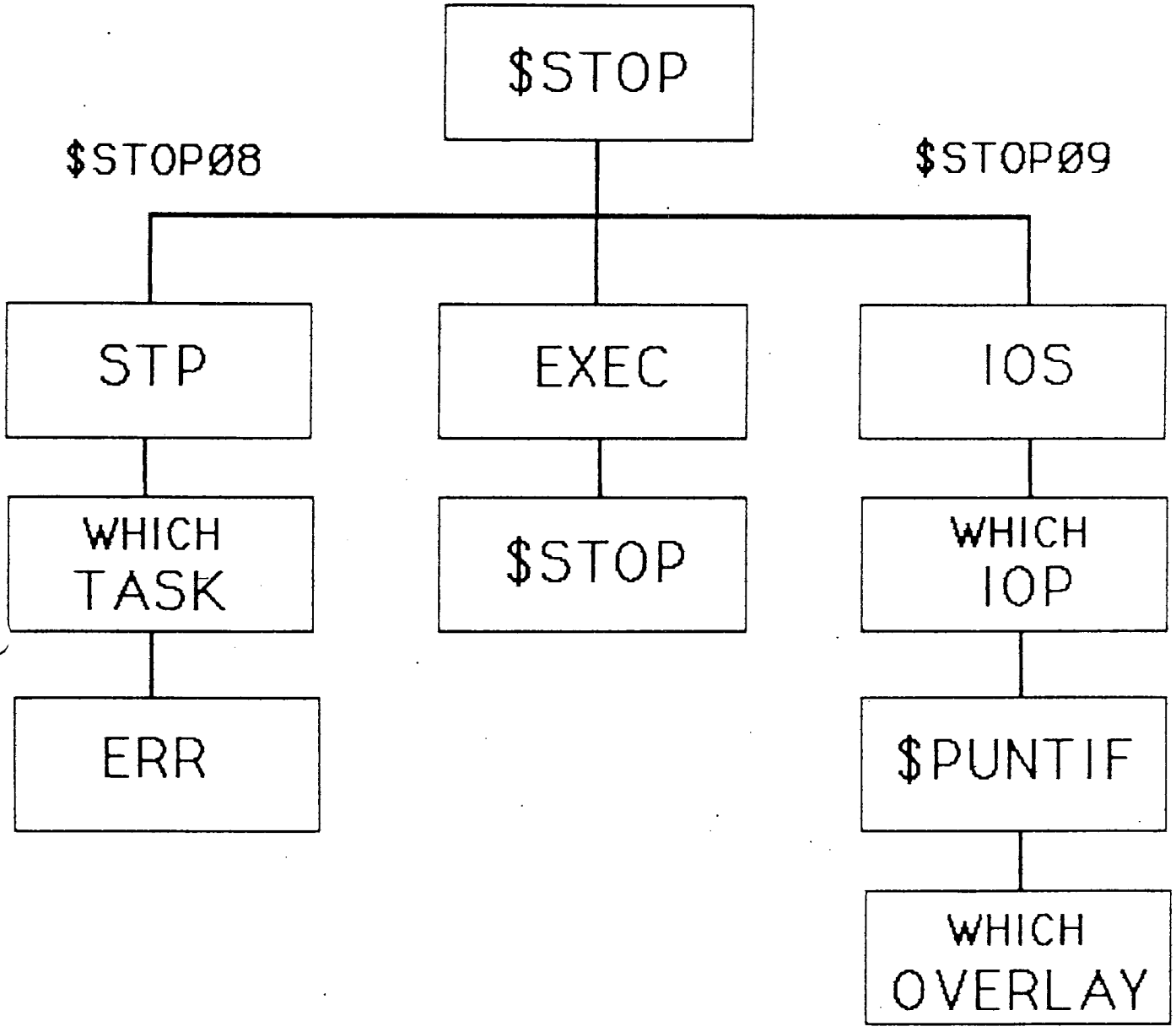
If it's a $STOP08 then the ERR hang macro was in STP code.

If it's a $STOP09 then the $PUNTIF hang macro was in IOS code.

If it's a $STOP00-03,STOP37-42 then it's likely hardware.

If it's any $STOP other than 08 or 09 then it's an EXEC initiated hang.

WHERE WAS THE HALT?



LOCATE THE HANG MESSAGE

LOCATE THE HANG MACRO IN LISTINGS

ANALYZE THE HISTORY TRACE

LOCATE NECESSARY POINTERS, REGISTERS, TABLES

```
*==============================*
!    S T O P   B U F F E R     !
*==============================*
EXEC STOPPED AT LABEL:  $STOP006
  W.P =          W.B0 =
--------------------------------
             message
------------END BUFFER ---------
```

The stop label is used in EXEC with the STOP macro. The STOP macro does not convert the values in P and B0 to ASCII characters, so their values appear in the dump. The value of P is in the word after the word containing W.P and the value of B0 is in the word after the word containing W.B0. These two values have been truncated to words.

The following convention is used for STOP labels and messages: the label has the form $STOP*ec*, where *ec* is a unique decimal number for each error condition. The stop message contains the routine name where the stop occurred and a short, descriptive error message.

(See SM-40, page 2-90 for more detail)

EXEC stop messages

Label	Code	Significance
$STOP000	EEF	Unknown error
$STOP001	EX	(A1) does not equal XP exchange address
$STP0002	APOIP	IOP channel error
$STOP003	APIIP	IOP channel error
$STOP004	EE	Program address range error
$STOP005	EEF	Floating-point error
$STOP006	EEF	Operand range error
$STOP007	EEF	Program range error
$STOP008	EEF	STP error exit (usually accompanied by SY006 message in STP memory)
$STOP009	EN	CPU halt requested by IOS
$STOP010	TECAN	Invalid event number
$STOP011	TS0	The STP Lock flag STPLK is set, but no task is marked as active.
$STOP012	R041	A PSWITCH request was received from a task other than the job scheduler (JSH).

STOP BUFFER

```
DMEM,FWA=1400,LWA=16000.              EXEC TABLES                    FDUMP 1.14          01/04/85     13:58:19     PAGE     2
                                                                     SYSDUMP             01/04/85     00:44:57

0001400 00000000000000000000 00000000000000000000 00000000000000000000 00000000000000000000
            LOCATIONS 0001404 THROUGH 0001417 CONTAIN 00000000000000000000
0001420 00010324052131240424 75 04152220254440260465 20 00010324050525202470 75 00000000000000000002  CPTYPE=CRAY XMP CPQUAN=
0001424 00000020646115202540 75 00000000000000000005 00000023441520252514 75 00000000000000000002   CLMAX=          NCPUS=
0001430 00000000047103230514 75 00000000000000000003 04151724620122212530 75 04151724620061134304 64    NCLS=        COS REV=COS 1.14
0001434 04253021241504202521 05 03006113630063136340 65 03006116430460164324 65 04150125000000000021 43 EXECDATE01/03/8501:10:55CAT    c
0001440 04150225000000000021 63 04151025000000000022 34 04151125000000000022 56 04151425000000000022 77 CBT     sCHT     CIT      CLT
0001444 04153025000000000023 17 04450325000000000026 15 04451025000000000026 31 04452024450400000026 45 CXT      ICT      IHT      IPRQ
0001450 04650325000000000026 50 04650523000000000027 14 04652225000000000027 46 05010325000000000030 12 MCT      MEL      MRT      PCT
0001454 05011124200000000031 45 05011724200000000033 30 05012225000000000036 73 05012724600000000040 26 PIQ     ePOQ     PRT      PWS
0001460 05111125000000000043 07 05111524600000000043 00 05150325000000000043 13 05152425000000000043 23 RIT      RMS      SCT      STT
0001464 05152424051114000055 65 05152426000000000047 40 05210225000000000053 20 05210525000000000053 32 STPRL  uSTX     TBT      TET
0001470 05212025000000000054 64 05410625000000000056 66 05412425000000000056 40 00000000000000000000 TPT    4XFT     vXTT
0001474 03647517236475175236475 03647517236475175236475 03647517236475175236475 03647517236475175236475 ================================
0001500 02044010020040100514 40 05204023620120100200 40 04104025220106100430 40 04244024420040100200 41 !     S T O P   B U F F E R    !
0001504 03647517236475172364 75 03647517236475172364 75 03647517236475172364 75 03647517236475172364 75 ================================
0001510 04253021241440246521 17 05012021242040202520 40 04610120442514100200 40 02212325047520140300 70 EXEC STOPPED AT LABEL   SSTOP008
0001514 05345624036400000000 00 00000000000000000201 30 05345620430060172000 00 00000000000000000200 13 W.P=           XW.BOO=
0001520 02645513226455132264 55 02645510040455244425 07 04452325042522246200 55 02645513226455132264 55 ---------- A-REGISTERS ---------
0001524 00000000000000777771 00 00000000000000000050 00 00000000000000000043 73 00000000000000000043 75 .       @
0001530 00000000000000003326 62 00000000000000000053 22 00000000000000000040 40 00000000000000000000
0001534 02645513226455132264 55 02645510051455244425 07 04452325042522246200 55 02645513226455132264 55 ---------- S-REGISTERS ----------
0001540 04000000000000000000 00 00000050100002012640 00 07467636377400000041 11 00000000003777760000 0 @            h y y   HI
0001544 00000000000000014033 12 00000000000000003006 62 00000000000000000040 00000000000000000007
0001550 02645513226455132264 55 02645513226455132264 55 02645513226455132264 55 02645513226455132264 55 --------------------------------
0001554 04250521402055100515 24 05004021251122236510 40 04253022252054100515 05 04244024652120100441 01 EEF - STP ERROR EXIT, SEE STP HA
0001560 04710710046505246515 01 04350510020040100200 40 02004010020040100200 40 02004010020040100200 40 NG MESSAGE
0001564 03647517236475175236475 02010523442040236430 40 04112521443105244200 75 03647517236475175236475 ======== END OF BUFFER =========
0001570 04151425251524212511 23 04151425251524212510 61 04300000000000000000 00 00000000000000000000 CLUSTERSCLUSTER1F
            LOCATIONS 0001574 THROUGH 0001607 CONTAIN 00000000000000000000
```

```
0000244 00000000000000003027 40 05310121000000000000 00 00000000000000003045 40 05210124645440140200 40          VAD          `TASK 0
0000250 05152424000000003046 00 05500000000000005124 40 05511423643400006510 00 05210124645440142200 40 STP     Z       ZLOG  R TASK 1
0000254 05150324000000003051 00 05210124645440144200 40 05250524000000003235 40 04151123600000003601 00 SCP    @TASK 2  UEP     `CIO    @
0000260 05211123600000003635 75 05210124645440146200 40 05010423200000004650 40 05210124645440150200 40 TIO    ]TASK 3  PDM     ] TASK 4
0000264 04210520600000004647 40 05210124645440152200 40 04212123200000004570 40 05210124645440154200 40 DEC    i TASK 5  DQM      `TASK 6
0000270 04652321600000004401 00 05210124645440156200 40 04650524000000005001 40 05210124645440160200 40 MSG    @@TASK 7  MEP     `TASK 8
0000274 05152023200000005005 00 05210124645440162200 40 04512322000000004416 00 05210124645440142300 40 SPM    @TASK 9  JSH    C TASK 10
0000300 04510323200000004533 66 05210124645440142304 40 05212123200000003646 40 05210124645440142310 40 JCM    V TASK 11 TQM     TASK 12
0000304 05152421600000005011 00 05210124645440142314 40 04312621000000005027 40 05210124645440142320 40 STG    @TASK 13 FVD     TASK 14
0000310 04452123200000005036 40 02504024652120100425 16 04212310040524005554 67 05153114030066100264 40 IQM     * STP ENDS AT  7SY006 -
0000314 05210124645440100310 40 04412523443440202520 40 03146314630060156410 40 02404010030060154200 60 TASK  2 HUNG AT 333007B (  006 0
0000320 03006314432462144305 02 02004010030460140300 40 03006014030466142304 61 02004010042522244405 16 0325221B   1000 00016111   ERRAN
0000324 02011520241522236244 00 04106010036440100200 40 00000000000000015540 35 05210124645440222421 23  MACRO) B0 =            TASK IDS
0000330 00000000000000000000 00000000000000000001 00000000000000000002 00000000000000000003
0000334 00000000000000000004 00000000000000000005 00000000000000000006 00000000000000000007
0000340 00000000000000000010 00000000000000000011 00000000000000000012 00000000000000000013
```

LOCATING THE MACRO

P or BØ in the STOP Buffer word address with no parcel

Match to Exec listing

If STP halted then locate STP hang message in STP table area

Will contain task ERR address and instructions surrounding macro to ease matching to the listing

Use STP map to determine task address and ID

Convert STP absolute address to the listing relative address

When a task detects a fatal condition, it will notify EXEC to stop the system through one of the following error macros:

Marco	Condition
ERRSZ	S0 = zero
ERRSN	S0 # zero
ERRSM	S0 negative
ERRSP	S0 positive
ERRAZ	A0 = 0
ERRAN	A0 # 0
ERRAM	A0 negative
ERRAP	A0 positive
ERRU	Unconditional

The error macros generate a conditional or unconditional call to an STP common routine (ERROR1) that will format the STP hang message (SY006), and execute an ERR exchange to EXEC. An ERR exchange from a system task will cause EXEC to stop the system with a $STOP008 code in the EXEC stop buffer.

Hang Macro In The Listings

```
EXEC --CRAY OPERATING SYSTEM EXECUTIVE--                    CRAY XMP  CAL 1.14(12/11/84)  12/13/84 14:50:14  Page 164
XPROC/EE                                                                                                        (164)

                                    *       STOP ON AN ERROR EXCHANGE FROM STP                                      E.2774
                                    *                                                                               E.2775
                          20110c    EEF     =       *                                                               E.2776
20110c  <macro>                             GETPW   A6,A7,S1                                                LC5921HA.229
20113a  1261 00000010                       S1      W@PWSAEF,A6                                             LC5921HA.230
     c  052141                              S0      S1<S@XPFPE      Position floating point error flag      M06586DA.1184
     d  <macro>                     $STOP005 STOP    SM,(EEF - FLOATING POINT ERROR )                            E.2779
20117a  052142                              S0      S1<S@XPORE      Position operand range error flag       M06586DA.1185
     b  <macro>                     $STOP006 STOP    SM,(EEF - OPERAND RANGE ERROR )                             E.2781
20123a  052143                              S0      S1<S@XPPRE      Position program range error flag       M06586DA.1186
     b  <macro>                     $STOP007 STOP    SM,(EEF - PROGRAM RANGE ERROR )                             E.2783
20127a  052146                              S0      S1<S@XPEE       Position error exit flag                M06586DA.1187
     b  <macro>                     $STOP008 STOP    SM,(EEF - STP ERROR EXIT, SEE STP HANG MESSAGE )            E.2787
20133a  <macro>                     $STOP000 STOP    UC,(EEF - UNKNOWN ERROR )                                   E.2788
```

```
                                    ************************************************************************M06586KA.2615
                                    **                                                                      M06586KA.2616
                                    * NAME   CRASH                                                          M06586KA.2617
                                    *                                                                        KC3860A.270
                                    * PURPOSE  TO ALLOW A USER TO HALT COS (TO BE USED FOR DEBUGGING        M06586KA.2618
                                    *          PURPOSES).                                                   M06586KA.2619
                                    *                                                                       M06586KA.2620
                                    * ENTRY  A1 == TCB ADDR                                                 M06586KA.2621
                                    *        A2 == JTA ADDR                                                 M06586KA.2622
                                    *                                                                       M06586KA.2623
                                    * EXIT   A1 == TCB ADDR                                                 M06586KA.2624
                                    *        A2 == JTA ADDR                                                 M06586KA.2625
                                    *                                                                       M06586KA.2626
                                    * METHOD IF (PARCEL D OF CRASHF NONZERO) HALT COS                       M06586KA.2627
                                    *        ELSE  ABORT THE USER WITH 'UNDEFINED USER CALL'                M06586KA.2628
                                    *                                                                       M06586KA.2629
                                    **                                                                      M06586KA.2630
                                    ************************************************************************M06586KA.2631
                                                                                                            M06586KA.2632
                          7247b+    CRASH   =       *                                                       M06586KA.2633
7247b  1000 00000000X                       A0      CRASHF,0                                                M06586KA.2634
     d  <macro>                             ERRAN                   IF (CRASHF NONZERO) HALT                M06586KA.2635
7251b  0401 00000032                        S1      ERIUC           ELSE  'INVALID USER CALL                M06586KA.2636
     d  006 00001461b+                      J       ABTJ1             AND ABORT THE USER                    M06586KA.2637
```

HISTORY TRACE

XFT - History Trace Function Table

Debug passes and makes entries into table

Function table gives ability to shut off different types of entries

XTT - History Trace Table

1024 word circular buffer maintained by Debug in Exec

Tasks create messages for Debug

Word 2 from Header points to next entry

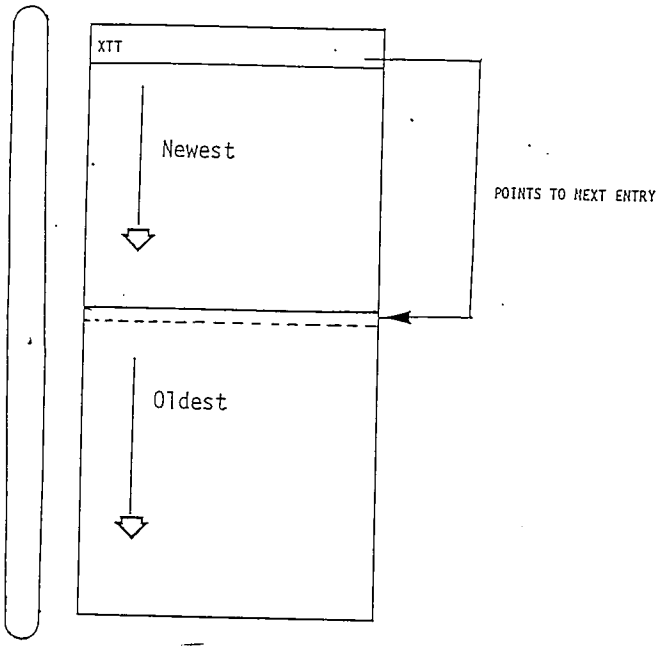
Address biased by XTT address

Entries made with the Post Macro

Each entry is four words

See SM-040, page 2-75 for each Entry

HISTORY TRACE




```
DMEM,FWA=1400,LWA=16000.          EXEC TABLES                    FDUMP 1.14          01/04/85     13:58:19     PAGE    10
                                                                 SYSDUMP             01/04/85     00:44:57

0005434 0000000000004300123457 0002155570671522134231 0000000000000000000000 0000000000000000000000      #  /     H
0005440 0000000000000000000000 0000000000000000000000 0000000000004300123457 0002155570670162143634          -a              # -/
0005444 0000000000000000077141 0002155570670242726745 0000000000000000077141 0002155570670270753012                          -a
0005450 0000000000410000130264 0002155570670062355204 0000000000422000130264 0002155570667564527221     @
  05454 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0005460 0000000000000000000000 0000000000000000000000 0000000000001760000000 0521202500000000000000                          TPT
0005464 0000000000000000000000 0000000000000000000000 0000000000000000004454 0000000000000000004622                           '
0005470 0000000000000000004705 0000000000000000004475 0000000000000000004643 0000000000000000004350                      =
0005474 0000000000000000004516 0000000000000000004371 0000000000000000004601 0000000000000000004560          N                                p
0005500 0000000000000000004412 0000000000000000004664 0000000000000000000000 0000000000000000000000
0005504 0000000000000000004433 0000000000000000000000 0000000000000000000000 0000000000000000000000
0005510 0000000000000000004537 0000000000000000000000 0000000000000000000000 0000000000000000000000                          _
            LOCATIONS 0005514 THROUGH 0005557 CONTAIN 000000000000000000000
0005560 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000004327
0005564 0515242405111400000000 0001000000000000000000 0541062500000000000000 0405142300000000000000 STPRL    @       XFT     ALL
0005570 0445172220000000000001 0525162120000000000001 0515162120000000000001 0425162120000000000001 IOI      UNE     SNE     ENE
0005574 0501032220000000000000 0541202060000000000000 0431052220000000000001 0515032200000000000000 PCI      XPC     FEI     SCH
0005600 0421112360000000000000 0445242320000000000001 0425052220000000000000 0431052360000000000001 DIO      ITM     EEI     FEO
0005604 0515052160000000000000 0515032222000000000001 0431052120000000000000 0515032360000000000001 SEG      SCI     FEE     SCO
0005610 0521232600000000000000 0451112500000000000000 0451222120000000000000 0451232060000000000000 TSX      JIT     JRE     JSC
0005614 0435052500000000000001 0461112040000000000000 0415202520000000000000 0451232200000000000001 GET      LIB     CPU     JSH
0005620 0515232100000000000000 0465052320000000000000 0465032520000000000000 0445202220000000000000 SSD      MEM     MCU     IPI
0005624 0421142220000000000000 0515312460000000000000 0471272500000000000000 0445202440000000000000 DLI      SYS     NWT     IPR
0005630 0445202020000000000000 0405232060000000000001 0465052060000000000000 0445212620000000000000 IPA      ASC     MEC     IQY
0005634 0364751723647510044111 0515242365113110052122 0405032122012420241114 0424401723647517236475 ===== HISTORY TRACE TABLE ======
0005640 0541242500000000000000 0002155570670014046052 0000000000000000000140 0000000000000000000000 XTT          OL*
0005644 0040240000277021605000 0035306600000000001340 0000000000000025047472 0425302400000000000000      G   X           TO:EXP
0005650 0030240000324002205000 0032401240000000003601 0425302402652325251520 0000000000000000000003          *     EXP-SUSP
0005654 0040240000277227605160 0000000000000000001236 0000000000000025047472 0451232200000000000000        p             TO:JSH
0005660 0030240000451414405160 0045162520000000006612 0000000000000000000000 0000000000000001232020      L2 p NU                      4
0005664 0040240000451415005160 0045162520000000001162 0000000000000025047472 0451232200000000000000      L4 p NU     r       TO:JSH
0005670 0030240000451015605160 0045165700000000023350 0000000000000000000000 0000000000000001332020      H7 p N     &
0005674 0040240000451016205160 0045165700000000001162 0000000000000025047472 0451232200000000000000      H9 p N      r      TO:JSH
0005700 0030240000277021205160 0045125520000000026456 0000000000000000000006 0000000000000000000014       E p J     -.
0005704 0040240000440103205100 0044010560000000001622 0000000000000025047472 0465232160000000000000          @              TO:MSG
0005710 0030240000440102605100 0044010560000000014754 0000000000000000000006 0000000000000000000003          @
0005714 0040240000277021605160 0045125520000000001441 0000000000000025047472 0451232200000000000000        G p J       !      TO:JSH
0005720 0030240000453036205160 0044745060000000003651 0000000000000000171125 0000000000000001224020      Xy p <             U       (
0005724 0040240000453036605160 0044745060000000001162 0000000000000025047472 0451232200000000000000      X{ p <      r      TO:JSH
0005730 0030240000446127605160 0044263160000000004441 1020000000000000213300 0000000000046347353016      1_ p  g     !              3
0005734 0040240000446130205160 0044263160000000003315 0000000000000025047472 0451232200000000000000      1a p  g             TO:JSH
0005740 0030240000453036205160 0044617260000000003337 0000000000000000213300 0000000000000001224020      Xy p 1                       (
0005744 0040240000453036605160 0044617260000000001162 0000000000000025047472 0451232200000000000000      X{ p 1      r      TO:JSH
0005750 0030240000442476605160 0044263160000000002376 0451232202652325251520 0000000000000000000003        p  g     JSH-SUSP
0005754 0040240000324002605000 0032401240000000001700 0000000000000025047472 0425302400000000000000          *             TO:EXP
0005760 0030240000324002205000 0032401240000000004033 0425302402652325251520 0000000000000000000003          *     EXP-SUSP
0005764 0040240000002000004100 0000000000000000001430 0000000000000025047472 0525232125100000213300          @             TO:USER
0005770 0020240000000200604100 0000000000000000001472 0000000000000000000100 0000000000000000000000          @      :      @
0005774 0040240000324002605000 0032401240000000001473 0000000000000025047472 0425302400000000000000          *     ;      TO:EXP
0006000 0420254004146106401761 0000000020000000014356 0360741322645513251524 0475201322645513237076 D  C1                <<-----STOP----->>
0006004 0040240000446130205160 0044263160000000003315 0000000000000025047472 0451232200000000000000      1a p  g             TO:JSH
0006010 0030240000453036205160 0044617260000000003337 0000000000000000213300 0000000000000001224020      Xy p 1                       (
0006014 0040240000453036605160 0044617260000000001162 0000000000000025047472 0451232200000000000000      X{ p 1      r      TO:JSH
0006020 0030240000442476605160 0044263160000000002376 0451232202652325251520 0000000000000000000003        p  g     JSH-SUSP
0006024 0040240000030145604100 0002537760000000001430 0000000000000025047472 0525232125100000213300          @             TO:USER
0006030 0020240000011505004100 0000740740000000021756 0000000000000000000014 0000000000000000011464      M  @ <      #                    4
0006034 0040240000324002605000 0032401240000000001473 0000000000000025047472 0425302400000000000000          *     ;      TO:EXP
```

MEMORY ERROR LOG

Interrupt Handler is XMEME

Could have occurred reading Exec Area in Idle

Corrects error if correctable

Communicates to Memory Error Processor (MEP)

Has an Exchange Package

A1 is Interrupted Exchange Package

Makes calls to:

Save Memory Error Info

Possibly suspend other CPU

Count Memory Errors

Analyze Syndrome Bits

Halt on Uncorrectable Hit

Build Message Table and Packet

Call MEP to Log the Error

Call to attempt Memory Correction

Returns to Exec Scheduler

Disables single bit count after the installation parameter I@MECCT exceeds 20 in the specified time by I@METO.

Address 15 in STP disables Bit Counting when non-zero.

A single bit hit should not hang the system.

A double bit hit in the user should not hang the system.

A double bit hit in STP will hang the system.

A double bit hit in EXEC is unpredictable.

Memory Error Log

```
--CRAY OPERATING SYSTEM EXECUTIVE--                    CRAY XMP  CAL 1.14(12/11/84)  12/13/84 14:50:14  Page 45
area                                                                                                      (45)

                                  ********************************************************************** LC5921VA.187
                                  **                                                                     LC5921VA.188
                                  *                                                                      LC5921VA.189
                                  *       X-MP cluster register dump area.                               LC5921VA.190
                                  **                                                                     LC5922GE.5
                                  *                                                                      LC5922GE.6
                                  *       Name:     Memory Error Log Table (MEL).                        LC5922GE.7
                                  *                                                                      LC5922GE.8
                                  *       Purpose:  Logs all single and double bit memory errors.        LC5922GE.9
                                  *                                                                      LC5922GE.10
                                  **                                                                     LC5922GE.11
                                  ********************************************************************** LC5922GE.12
                                                                                                         LC5922GE.13
 2714a  <macro>                           ISFWB   NW=4                                                   LC5922GE.14
                              0   B@MEL    BSS     0                                                      LC5922GE.15
        0465052300000000000000            CON     'MEL'L          Header                                 LC5922GE.16
 2715   0521172504051417200000            DATA    'TOTAL='L                                              LC5922GE.17
 2716                         1   MELTC    BSSZ    1                                                      LC5922GE.18
 2717   0515112344351421236400            DATA    'SINGLE='L                                             LC5922GE.19
 2720                         1   MELSC    BSSZ    1                                                      LC5922GE.20
 2721   0421172524111421236400            DATA    'DOUBLE='L                                             LC5922GE.21
 2722                         1   MELDC    BSSZ    1                                                      LC5922GE.22
 2723   0421052504250325036400            DATA    'DETECT='L                                             P10711AA.1
 2724                         1   MELDS    BSSZ    1               Non-zero if Single bit detection       P10711AA.2
                                  *                                 disabled                               P10711AA.3
 2725   0415172445110520652075            DATA    'CORRECT='L                                            P10711AA.4
                                  $CRAYX   IFE     C@CPTYPE,NE,@CRAYXMP                                   P10711AA.5
        <skipped>--               MELNC    CON     0               Non-zero if Single bit correction      P10711AA.6
                                  *                                 disabled                               P10711AA.7
                                  $CRAYX   ELSE                                                           P10711AA.8
 2726   00000000000000000001      MELNC    CON     1               Non-zero if Single bit correction      P10711AA.9
                                  *                                 disabled (Default = 1 for XMP)         P10711AA.10
                                  $CRAYX   ENDIF                                                          P10711AA.11
 2727   0415252445110523452075            DATA    'CURRENT='L                                            LC5922GE.23
 2730                         1   MELCB    BSSZ    1               Current Bank                           LC5922GE.24
 2731                         1   MELCC    BSSZ    1               Current Chip select                    LC5922GE.25
 2732                         1   MELCS    BSSZ    1               Current syndrome                       LC5922GE.26
 2733                         1   MELCE    BSSZ    1               Current error type                     LC5922GE.27
 2734                         1   MELCR    BSSZ    1               Current read mode                      LC5922GE.28
 2735                         1   MELCT    BSSZ    1               Current RTC                            LC5922GE.29
 2736   0461012465207500000000            DATA    'LAST='L                                               LC5922GE.30
 2737                         1   MELLB    BSSZ    1               Last Bank                              LC5922GE.31
 2740                         1   MELLC    BSSZ    1               Last Chip select                       LC5922GE.32
 2741                         1   MELLS    BSSZ    1               Last syndrome                          LC5922GE.33
 2742                         1   MELLE    BSSZ    1               Last error type                        LC5922GE.34
 2743                         1   MELLR    BSSZ    1               Last read mode                         LC5922GE.35
 2744                         1   MELLT    BSSZ    1               Last RTC                               LC5922GE.36
                             31   L@MEL    =       W.*-B@MEL       Required for FDUMP                     LC5922JA.12
```

POINTERS AND TABLES

Tables have ASCII labels.

Some tables exist in the task itself.

Useful to avoid dumping all of memory.

Used with COS debugger and the IOP operator station.

Used by analyst only.

Pointers to:

- EXEC Tables
- STP Tables
- STP Common Routines beginning Address
- STP Task IDs and Base Address

Helps match STP listings to memory.

STP hang message also in this area.

COS TABLE POINTERS

```
'EM, FWA=1400, LWA=16000.               EXEC TABLES                      FDUMP 1.14      01/04/85      13:58:19      PAGE    2
                                                                         SYSDUMP         01/04/85      00:44:57

0001400 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
             LOCATIONS 0001404 THROUGH 0001417 CONTAIN 0000000000000000000000
0001420 0001032405213124042475 0415222025444026046520 0001032405052520247075 0000000000000000000002  CPTYPE=CRAY XMP CPQUAN=
0001424 0000002064611520254075 0000000000000000000005 0000002344152025251475 0000000000000000000002   CLMAX=          NCPUS=
0001430 0000000004710323051475 0000000000000000000003 0415172462012221253075 0415172462006113430464    NCLS=        COS REV=COS 1.14
0001434 0425302124150420252105 0300611363006313634065 0300611643046016432465 0415012500000000002143 EXECDATE01/03/8501:10:55CAT     c
0001440 0415022500000000002163 0415102500000000002234 0415112500000000002256 0415142500000000002277 CBT     sCHT      CIT       CLT
0001444 0415302500000000002317 0445032500000000002615 0445102500000000002631 0445202445040000002645 CXT      ICT      IHT       IPRQ
0001450 0465032500000000002650 0465052300000000002714 0465222500000000002746 0501032500000000003012 MCT      MEL      MRT       PCT
0001454 0501112420000000003145 0501172420000000003330 0501222500000000003673 0501272460000000004026 PIQ     ePOQ      PRT       PWS
0001460 0511112500000000004307 0511152460000000004300 0515032500000000004313 0515242500000000004323 RIT      RMS      SCT       STT
0001464 0515242405111400005565 0515242600000000004740 0521022500000000005320 0521052500000000005332 STPRL   uSTX      TBT       TET
0001470 0521202500000000005464 0541062500000000005566 0541242500000000005640 0000000000000000000000 TPT     4XFT     vXTT
0001474 0364751723647517236475 0364751723647517236475 0364751723647517236475 0364751723647517236475 ================================
0001500 0204401002004010051440 0520402362012010020040 0410402522010610043040 0424402442004010020041 !      S T O P    B U F F E R    !
0001504 0364751723647517236475 0364751723647517236475 0364751723647517236475 0364751723647517236475 ================================
0001510 0425302124144024652117 0501202124204020252040 0461012044251410020040 0221232504752014030070 EXEC STOPPED AT LABEL   $STOP008
0001514 0534562403640000000000 0000000000000000020130 0534562043006017200000 0000000000000000020013 W.P=             XW.B00=
0001520 0264551322645513226455 0264551004045524442507 0445232504252224620055 0264551322645513226455 ---------- A-REGISTERS ---------
0001524 0000000000000077777100 0000000000000000005000 0000000000000000004373 0000000000000000004375          @
0001530 0000000000000000332662 0000000000000000005322 0000000000000000004040 0000000000000000000000
0001534 0264551322645513226455 0264551005145524442507 0445232504252224620055 0264551322645513226455 ---------- S-REGISTERS ---------
0001540 0400000000000000000000 0000005010000201264000 0746763637740000044111 0000000000037777600000 @                h y y   HI
0001544 0000000000000001403312 0000000000000000300662 0000000000000000000040 0000000000000000000007
0001550 0264551322645513226455 0264551322645513226455 0264551322645513226455 0264551322645513226455 --------------------------------
0001554 0425052142005510051524 0500402125112223651040 0425302225205410051505 0424402465212010044101 EEF - STP ERROR EXIT, SEE STP HA
0001560 0471071004650524651501 0435051002004010020040 0200401002004010020040 0200401002004010020040 NG MESSAGE
0001564 0364751723647517236475 0201052344204023643040 0411252144310524420075 0364751723647517236475 ======== END OF BUFFER ========
0001570 0415142525152421251123 0415142525152421251061 0430000000000000000000 0000000000000000000000 CLUSTERSCLUSTER1F
```

```
DMEM, BIAS=32000, FWA=0, LWA=1000.      STP TABLES                       FDUMP 1.14      01/04/85      13:58:19      PAGE    36
                                                                         SYSDUMP         01/04/85      00:44:57

0000000 0000000000000000000002 0000000000000000000000 0000000000000000000000 1777777777777777777777
0000004 0000000000000000000000 0000000000000000000000 0000000000000000257244 0000000000000000000313
0000010 0000000000000000000000 0000000000000000170752 0000000000000000000001 0000000000000000000003
0000014 0000000000000000000003 0000000000000000000010 0000000000000000000000 0421140000000000000012                      DL
0000020 0000000000000000222262 0000000000000000000000 0000000000000000000000 0000000000000000000000         $
0000024 0000000000000000014152 0000000000000000014516 0000000000000000000000 0000000000000000256122          j        N          \R
0000030 0000000000000000256140 0000000000000000000000 0000000000000000000000 0415172462006113430464         \`                 COS 1.14
0000034 0300611363006313634065 0300611643046116432465 0024423571020744261120 0000000000000000000001 01/03/8501:11:55 "w    bP
0000040 0000000000000000000001 0000000000000000000000 0405252500000000065365 0415012304600000257043                  AUT     j CALL   ~#
0000044 0415162500000000265363 0415232100000000166677 0421012500000000121323 0421032500000000162432 CNT     j CSD      DAT       DCT
0000050 0425212400000000014152 0421222500000000162553 0421302220000000222613 0425212500000000250033 EQP      jDRT     kDXI    % EQT      P
0000054 0425272500000000166656 0435222500000000264273 0445022500000000170677 0445142500000000247157 EWT       GRT     h IBT      ILT     No
0000060 0445232500000000247770 0445202500000000257445 0451232424111500222617 0451302500000000170771 IST     o IPT    _%JSQBM  % JXT
0000064 0461112040000000323777 0461032500000000225002 0461112500000000225007 0461302500000000225244 LIB       LCT    * LIT    * LXT      *
0000070 0465232500000000222262 0501042220000000222612 0501042460000000165277 0501102440000000251116 MST     $ PDI    % PDS      PHR     RN
0000074 0501302500000000170752 0505042500000000260006 0511112500000000257345 0511202305440000000505 PXT       QDT    ` RIT    ~ RPLY     E
0000100 0511212465200000000465 0511212500000000253120 0511232500000000323543 0515012060000000256122 RQST     5RQT     VPRST     cSAC    \R
0000104 0515042440000000152343 0515042500000000066353 0515212400000000256140 0515242400000000166451 SDR       SDT     I SQP    \`STP     )
0000110 0515302500000000016120 0521042500000000264310 0521212400000000014516 0521302500000000213242 SXT      PTDT     h TQP     NTXT
0000114 0525042500000000246737 0531202500000000247370 0541012500000007656001 0521222024150524600000 UDT     M VPT    N XAT    \ TRACES
0000120 0421212325200000251015 0445242065200000265540 0521212325200000415630 0000000000000000000000 DQMT    R ITCT   k`TQMT
0000124 0000000000000000000014 0000000000000000000002 0000000000000000000006 0000000000000000000000
0000130 0000000000000000000000 0411242024200000000000 0000000000000000267240 0411062324051600000000           BTAD             n BFMAN
0000134 0000000000000000266640 0415032365013100000000 0000000000000000267340 0415102024451624600000         m CCOPY            n CHAINS
0000140 0000000000000000267440 0415102265151500000000 0000000000000000267700 0415142124052200000000         o CHKSM            o CLEAR
0000144 0000000000000000270000 0415172344311121600000 0000000000000000270100 0415172345312424600000         p CONFIG           p@CONVTS
0000150 0000000000000000272440 0415172405440000000000 0000000000000000272640 0415222024151321251000         u COPY             u CRACKER
0000154 0000000000000000272740 0421032245412400000000 0000000000000000273340 0431162104510200000000         u DCJXT            v FNDJB
0000160 0000000000000000273500 0431112604513024051000 0000000000000000273440 0435052504752723400000         w@FIXJXPR          w GETOWN
0000164 0000000000000000273540 0435052505010124446400 0000000000000000273600 0435242324251500000000         w`GETPARM          w GTMEM
0000170 0000000000000000274000 0435242504150200000000 0000000000000000274100 0435242505412400000000         x GTTCB            x@GTTXT
0000174 0000000000000000274200 0445232404151723200000 0000000000000000510600 0445122505152520400000         x ISPCOM             IJTSUB
0000200 0000000000000000274300 0451152124640000000000 0000000000000000275000 0451242024211625000000         x JMEM             z JTADNT
0000204 0000000000000000275140 0461072325150700000000 0000000000000000275240 0465232165052521200000         z`LGMSG            z MSGQUE
0000210 0000000000000000275340 0501012064550525050400 0000000000000000275600 0501272124710300000000         z PACKETQ          { PWENC
0000214 0000000000000000276000 0505152464340000000000 0000000000000000276040 0505252125250524600000         ¦ QMSG             ¦ QUEUES
0000220 0000000000000000276440 0511052425112023054400 0000000000000000276740 0511142324251500000000         } REQRPLY          } RLMEM
0000224 0000000000000000277700 0511142505412400000000 0000000000000000300000 0515041445010400000000           RLTXT              SD2PD
0000230 0000000000000000300100 0515242404210125051400 0000000000000000300340 0515242404252224400000          @STPDATS            STPERR
0000234 0000000000000000300540 0515242404650523200000 0000000000000000300740 0515242405211123242400          `STPMEM             STPTIME
0000240 0000000000000000301240 0515242405252422246000 0000000000000000301540 0525042064751500000000           STPUTIL           `UDCOM
0000244 0000000000000000302740 0531012100000000000000 0000000000000000304540 0521012464544014020040           VAD               `TASK 0
0000250 0515242400000000304600 0550000000000000512440 0551142364340000651000 0521012464544014220040 STP       Z        ZLOG    R TASK 1
0000254 0515032400000000305100 0521012464544014420040 0525052400000000323540 0415112360000000360100 SCP      @TASK 2   UEP     `CIO      @
0000260 0521112360000000363575 0521012464544014620040 0501042320000000465040 0521012464544015020040 TIO      }TASK 3   PDM     j TASK 4
0000264 0421052060000000464740 0521012464544015220040 0421212320000000457040 0521012464544015420040 DEC     i TASK 5   DQM     ~ TASK 6
0000270 0465232160000000440100 0521012464544015620040 0465052400000000500140 0521012464544016020040 MSG     @@TASK 7   MEP     `TASK 8
0000274 0515202320000000500500 0521012464544016220040 0451232200000000441600 0521012464544014230040 SPM      @TASK 9   JSH     C TASK 10
0000300 0451032320000000453366 0521012464544014230440 0521212320000000364640 0521012464544014231040 JCM     V TASK 11  TQM       TASK 12
0000304 0515242160000000501100 0521012464544014231440 0431262100000000502740 0521012464544014232040 STG      @TASK 13  FVD       TASK 14
0000310 0445212320000000503640 0250402465212010042516 0421231004052400555467 0515311403006610026440 IQM       * STP ENDS AT   7SY006 -
0000314 0521012464544010031040 0441252344344020252040 0314631463006015641040 0240401003006015420060 TASK  2 HUNG AT 333007B (   006 0
0000320 0300631443246214430502 0200401003046014030040 0300601403046614230461 0200401004252224440516 0325221B   1000 00016111    ERRAN
0000324 0201152024152223624400 0410601003644010020040 0000000000000001554035 0521012464544022242123  MACRO) B0 =               TASK IDS
0000330 0000000000000000000000 0000000000000000000001 0000000000000000000002 0000000000000000000003
0000334 0000000000000000000004 0000000000000000000005 0000000000000000000006 0000000000000000000007
0000340 0000000000000000000010 0000000000000000000011 0000000000000000000012 0000000000000000000013
```

$STOP008

A. Determine the halting task from address SY006 in STP.

B. Locate the particular hang macro by finding the address in the task.

C. Use the following suggestions to approach halts in individual tasks.

1. Station Call Processor

Tables to be concerned with are LXT, SST, LIT and rarely SDT.

Locate in SCP XP A2 which is usually LXT entry pointer.

Determine from LXT the ID and last message codes and SCBs exchanged with front end.

SST contains SCP continuation address, pointer to expected SCB table, and some flags. In particular, I/O outstanding flag. Also, SST points to SDT entry of dataset currently being processed.

Use EXEC trace to determine the last few SCB exchanges.

2. Disk Queue Manager

Tables are: EQT, DCT, RQT, DNT, DAT

Two typical DQM Hangs:

° 0 AI encountered in DAT on R/W or deallocation.

This difficult problem with allocation usually follows change to DQM, PDM or EXP.

Must locate failing DAT by looking in RQT entry for DNT address which will be used to locate DAT.

Often multiple dumps are required to locate this.

EQLST field may help.

° Looping in DQM

Usually related to configuration of EQT.

Use EXEC trace to note only DQM running and locate P address in DQM.

Problem is probably in device initiator part of DQM.

Use EQT and DCT.

3. Permanent Dataset Manager

Tables are PDD, DSC, DNT, DSC, and DSP.

PDM save area in SPT at address BGNCON
FCODE - Function Code
SVBØ - Return Address
PDDBSE - Address of PDD

Trace not usually useful.

Typical problem:

Read past EOD on DSC due to bad input from another task.

4. Job Class Manager

Tables are SDT and CSD.

5. Job Scheduler

Tables are JXT, MST and occasionally SDT.

Use trace to see what job state changes and memory moves have occurred.

Validate job size in JXT with MST.

JSH storage and constants at address B@STP.

Typical problem:

Operand range error.
Look at memory moves.

6. User Exchange Processor

Tables are JTA and JXT

Use A1 in EXP and XP to locate JTA address

JTA must be dumped

EXP temporary storage is in JTA at W@JTAI, W@JTS7 and at W@JTESTK.

First word of stack is stack pointer and first word of each entry is usually B0.
Stack used a lot by JCL routines.

S0 of user's XP (word 30) in JTA is user's function code.

Use trace to see what job-related events were occurring.

7. Exec Message Processor (MEP)

   Table is METTA in STP

8. Log Manager

   Tables are LOGJXT, $SYSTEMLOG, PDD, SYSTEMLOG DNT, $SYSLOG DSP.

   Messages are queued up in pool 1.

   Storage in STP at address LGBMEM.
   CCSYS and CCUSR valuable

9. System Performance Monitor

   Tables are IC, STT, MCT, DCT, CSD.

   Storage in STP at SPM INTVL.

10. Disk Error Correction

    No Hangs in it

11. Tape Queue Manager

12. Overlay Manager

13. Flush Volatile Device - FVD

14. Start Up

    Tables - many

    Usually looking at code can indicate what was wrong

    Break pointing is helpful

    Save areas are resident in STARTUP - sometimes STARTUP must be dumped.

Z	SCP	STG	DQM	PDM	JCM	JSH	DQM	DEC	EXP	MEP	MSG	SPM	IQM	FVD
AUT	-AUT	PDD	-DAT	DAT	-CSD	CSD	(DDL)	EQT	CNT	-AEM	AUT	CSD	ILT	EQT
CNT	-IBT	SDT	DCT	CSD	SDT	-JXT	-(DNT)		-DEX		(DSP)	DCT	VPT	DRT
DAT	-LCT	-SST	(DNT)	(DNT)		-MST	-(DSP)		(DNT)		JTA	+MCT	VCT	
(DNT)	-LIT		-DRT	DRT		-RJI	JXT		(DSP)		JXT	+ICT	TRB	
DRT	-LXT		(DSP)	-DSC		SDT	-LFT		-DUX		-LGJ	+STT	IST	
-DSC	PDD		-EQT	(DSP)		-TXT	-ODN		-FSH		PDD			
(DSP)	-SDT		GRT	DXT		JTA	(PDD)		GRT		SDT			
-DVL	-STT		JXT	EQT			SDT		JXT					
DXT			-RQT	JCB			-SWT		-LDT					
-EFT			JTA	JTA			TCB		-SM					
-EQT			PHR	JXT			-UPT		-VAX					
GRT			SQP	PDD			-JTA		-VUX					
JTA			TXT	-PDI			-TXT		JTA					
JXT				-PDS			IDD							
(ODT)				-QDT			IRT							
PDI				SDT			TRB							
QDT				-XAT										
-RJI														
SDT														
TCB														
TDT														
TXT														

PACKET DRIVER	FED DRIVER	DISK/SSD DRIVER
+SCT	+CHT	DCT
+PIQ	+CXT	EQT
+POQ	LIT	+CHT
+CHT	LXT	+CAT
+CAT	LCT	+CLT
+CLT	+CAT	+CIT
+CIT	+CLT	+CBT
CXT	+CIT	+PHR
		+SAC
		+SQP

USER		CSP
(BGN)	(LDT)	(DSP)
(BAT)	(OPT)	(JCB)
(DDL)	(JBI)	(LFT)
(DSP)	(DRPB)	(DNT)
(DNT)	(RCB)	
(JAC)	(NCB)	
(JCB)	(MHB)	
(LFT)	(IJPB)	
(ODN)		
(PDD)		
(JST)		

Z	SCP	S16	DDM	PDM	JCM	JSH	DOM	DEC	EXP	MEP	MSG	SPM	IOM	FVD
AUT	-AUT	PDD	-DAT	DAT	CSD	CSD	(DDL)	EOT	CNT	-AEM	AUT	CSD	ILT	EOT
CNT	-IBT	SDT	DCT	CSD	SDT	-JXT	-(DNT)		-DEX		(DSP)	DCT	VPT	DRT
DAT	-LCT	-SST	(DNT)	(DNT)		-MST	-(DSP)		(DNT)		JTA	+MCT	VCT	
(DNT)	-LIT		-DRT	DRT		-RJI	JXT		(DSP)		JXT	+ICT	TRB	
DRT	-LXT		(DSP)	-DSC		SDT	-LFT		-DUX		-LGL	+STT	IST	
-DSC	PDD		-EOT	(DSP)		-TXT	-NDD		-FSH		PDD			
(DSP)	-SDT		GRT	DXT		JTA	(PDD)		GRT		SDT			
-DVL	-STT		JXT	EOT			SDT		JXT					
DXT			-PDI	JCB			-SWT		-LDT					
-EFT			JTA	JTA			TCB		-SM					
-EOT			PHR	JXT			-UPT		-VAX					
GRT			CDP	PDD			-JTA		-VUX					
JTA			TXT	-PDI			-TXT		JTA					
JXT				-PDS			DDI							
(LDT)				-ODT			TRI							
PDI				SDT			TRB							
ODT				-XAT										
-RJI														
SUT														
TCB														
TDT														
TXT														

PACKET DRIVER	FED DRIVER	DISK/SSD DRIVER	USER		CSP
+SCT	+CHT	DCT	(BGN)	(LDT)	(DSP)
+PIO	+CXT	EOT	(BAT)	(OPT)	(JCB)
+PDD	LIT	+CHT	(DDL)	(JBI)	(LTT)
+CHT	LXT	+CAT	(DSP)	(DRPB)	(DNT)
+CAT	LCT	+CLT	(DNT)	(RCB)	
+CLT	+CAT	+CIT	(JAC)	(NCB)	
+CIT	+CLT	+CBT	(JCB)	(MHB)	
CXT	+CIT	+PHR	(LFT)	(IJPB)	
		+SAC	(NDD)		
		+SDP	(PDD)		
			(JST)		

COS DUMP ANALYSIS

1. STOP BUFFER	___ System utility to format a dump created by SYSDUMP
2. $STOP MACRO	___ Allows analyst to turn on and off any type of history trace entry
3. ERRSN MACRO	___ History trace for inter-task communication
4. EXCHANGE PACKAGES 100-200	___ Location in STP TABLES where the STP hang message is put on an STP hang condition
5. XFT	___ Creates a copy of the Cray system memory on the master device scratch buffer for dumps
6. XTT	___ Macro executed by a task when a hang condition exists
7. ITCT	___ Location to look first in a dump to determine who halted first
8. FDUMP	___ Macro executed by EXEC to hang the operating system
9. SYSDUMP	___ Table containing History trace events of the system
10. SYØØ6	___ Executing CPU Exchange packages on a hang

CDS DUMP ANALYSIS

1. STOP BUFFER	____ System utility to format a dump created by SYSDUMP
2. $STOP MACRO	____ Allows analyst to turn on and off any type of history trace entry
3. ERRSN MACRO	____ History trace for inter-task communication
4. EXCHANGE PACKAGES 100-200	____ Location in STP TABLES where the STP hang message is put on an STP hang condition
5. XRT	____ Creates a copy of the Cray system memory on the master device scratch buffer for dumps
6. XTT	____ Macro executed by a task when a hang condition exists
7. ITCT	____ Location to look first in a dump to determine who halted first
8. PDUMP	____ Macro executed by EXEC to hang the operating system
9. SYSDUMP	____ Table containing History trace events of the system
10. SYSOS	____ Executing CPU Exchange packages on a hang

IOS Dump Analysis

9

LEARNING OBJECTIVES

With the aid of all furnished reference materials, upon completion of the IOS Dump Analysis module, the learner should be able to:

1. Determine if IOS halted and which IOP halted first.
2. Locate the $PUNTIF Hang Message.
3. Determine the $PUNTIF Macro.
4. Locate the $PUNTIF Macro in the listings.
5. Analyze the History Trace entries.
6. Analyze the Operand Registers.
7. Locate and analyze the ERRBUFF Tables.
8. Find the exeuction Environment leading to the hang.

IOS HANG

Determine the hung IOP from the hang message.

Run FDUMP to print out the appropriate IOP memory, registers and history trace.

Examine the Operand Registers

Determine the overlay base address from operand register 3.

Determine the overlay number currently running from operand register 63.

Determine executing overlay from LISTO or the address pointed to by operand register 3.

Examine the Exit Stack and busy-done flags for the channels.

FDUMP will provide a formatted print-out.

EXSAVE will also contain this information.

Determine the E Pointer value and the E Pointer-1 contents.

If this is less then op reg 3 then the hang is in the KERNEL.

If this is more than op reg 3 then the hang is in an OVERLAY and must have %B subtracted from it for a relative address.

Locate the $PUNTIF macro in the listing.

Examine information in the history trace buffer.

The time is ms.

The overlay that logged an event.

Kernel events are logged.

EACT+1 points to the active activity descriptor chain for all activities in the IOP.

ESMD points to active SMOD.

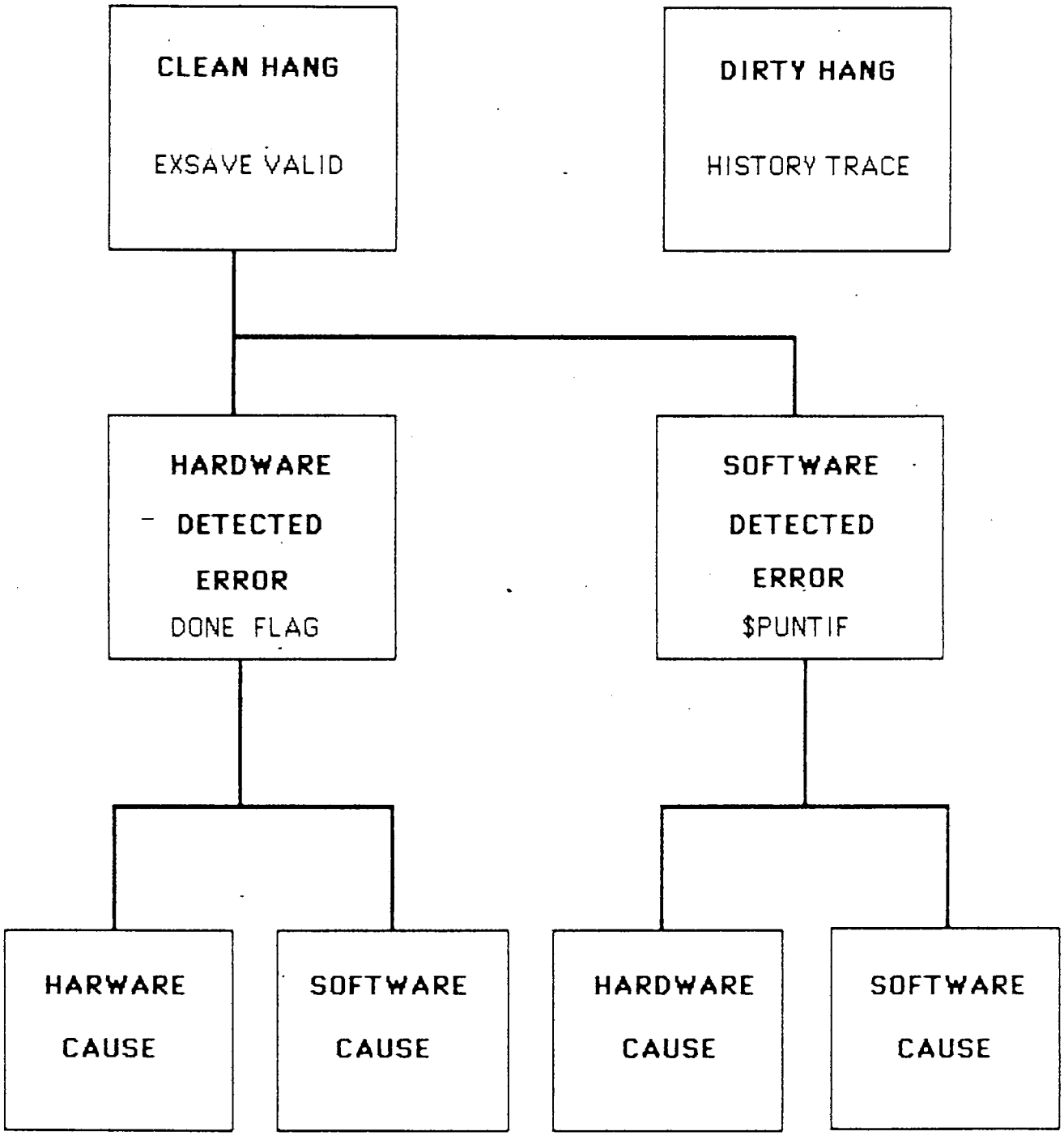
Activity descriptor address.

Size of SMOD entry.

Overlay table address.

A,B,C,E,Exit Stack.

IOS HANG


CLEAN HANG
EXSAVE VALID
DIRTY HANG
HISTORY TRACE
HARDWARE
DETECTED
ERROR
DONE FLAG
SOFTWARE
DETECTED
ERROR
$PUNTIF
HARWARE
CAUSE
SOFTWARE
CAUSE
HARDWARE
CAUSE
SOFTWARE
CAUSE

KERNEL ERROR HALT PROCESS

Entered at 'STOPIT' when software detects an error ($PUNTIF MACRO is executed).

Disables interrupts

Saves A, C, B, E registers, exit stack, and all channel DN and BZ flags at 'EXSAVE' in kernel.

Saves operand registers at 'OPREGS' in kernel.

Sends error halt message to kernel console

Halts other IOP s (at 'DONE4")

Passes control to SYSDUMP.

$PUNTIF FORMAT

LOCATION	RESULT	OPERAND
	PUNTIF	cond1, andor, cond2, CODE=
	cond1	Valid AMPL conditional clause
	andor	'and' or 'or'
	code	Symbol of the form PT$ _____ which defines the reason for halt

KERNEL HALT (PUNT) CODES

CODE	MEANING
000	No error code specified on $PUNTIF macro
001	Local memory error (always HARDWARE)
002	Buffer memory error on deadstart (always HARDWARE)
003	Buffer memory error (always HARDWARE)
004	High-speed channel error (always HARDWARE)
005	Invalid message received from CPU
006	Invalid parameter in disk request from CPU
007	Program was executing at location 0
010	Local memory location 0 was overwritten
011	Undefined message received on IOP communication channel
012	Overlay does not exist
013	Station stack overflow or underflow
014	Local memory buffer not available
015	Buffer memory disk buffer not available
016	Invalid local buffer release call
017	Buffer memory incorrectly configured
020	IOP message channels incorrectly configured
021	SMOD is too large for area on buffer memory
022	Invalid local memory address
023	Illegal interrupt program sequence code
024	Stop request received from CPU
025	Low-speed channel error (always HARDWARE)
026	Block number validation trap
040	Block Mux interrupt processor error
041	Bad CRW address in device table
042	Block mux start I/O error
044	Block Mux configuration error
050	DD49 disk software error

HALT MESSAGE

,TYPE=IOP0,FWA=4000,LWA=12000,R. IOP0 LOCAL MEMORY FDUMP 1.14 01/04/85 18:20:31 PAGE 7
SYSDUMP 01/03/85 23:55:42

```
 005000  000000  000000  000000  000000    064560  000000  000000  000000
 005010  000000  000000  000000  000000    064570  000000  000000  000000
 005020  000000  000000  000000  000000    000000  000000  000000  000000
 005030  000000  000043  000000  013740    073410  100313  000141  000000
 005040  000000  000000  000000  000000    000000  000000  000000  000000
 005050  000000  000000  000000  000000    000037  106777  000000  000000
 005060  000000  000000  001000  000000    000001  060000  000000  000000
 005070  000047  072100  072170  072260    072350  000000  000000  000000
 005100  000000  000000  000000  000000    000000  000000  000000  000000
 005110  000000  000000  000000  000000    000000  000000  073244  000000
 005120  000000  000000  000000  000000    000000  000000  040000  072464
 005130  072524  072724  000000  072664    072564  072624  000000  000000
 005140  072764  000000  000000  000000    000000  000000  000000  000000
 005150  000000  000000  000000  000000    000000  000000  000000  076450
 005160  000000  045340  066240  070300    000057  164000  160000  000010
 005170  000010  000037  000010  000241    064750  000011  000001  021000
 005200  000313  000006  000400  063330    000003  000000  010610  000024
 005210  000015  000060  064764  000007    000002  105000  076620  073040
 005220  000001  000001  000003  000125    000027  000065  000005  000000
 005230  000000  000000  000000  017240    000000  000000  017306  017367
 005240  000000  017415  017717  017415    017717  017415  017717  000000
 005250  000000  030022  027575  025565    026163  000000  000000  026677
 005260  026677  000000  000000  026677    026677  000000  000000  027121
 005270  027121  000000  000000  000000    000000  010610  000000  063350
 005300  000000  014610  000006  063700    000000  020610  000010  064230
 005310  000000  024610  000012  064650    000000  010610  000000  000000
 005320  000000  014610  000000  000000    000000  020610  000000  000000
 005330  000000  024610  000000  000000    000000  010610  000000  000000
 005340  000000  014610  000006  000000    000000  020610  000010  064610
 005350  000000  024610  000012  005361    005431  005361  005361  000000
         LOCATIONS 000005360 THROUGH 000005427 CONTAIN 000000
 005430  000000  000000  000000  000000    000000  063600  064130  064460
 005440  000000  000000  000000  000000    000000  000000  000000  000000
 005450  000000  000000  000001  000002    000003  000000  000006  000010
 005460  000012  050554  000000  000000    000000  064770  000000  000000
 005470  000000  000000  000000  000120    000000  000000  000000  000000
         LOCATIONS 000005500 THROUGH 000005547 CONTAIN 000000
 005550  000000  071100  000002  065000    000560  002600  001000  020000
 005560  000000  177777  177777  177777    177777  177777  177777  177777
 005570  177777  177777  177777  177777    177777  000002  064000  000002
 005600  000040  002400  000000  000000    000000  000000  000000  000000
 005610  000000  000000  000000  000000    000000  000000  000000  000000
 005620  000000  000000  000000  000000    000002  062000  000000  005015
 005630  020111  047520  026461  020110    040514  052040  030060  032040
 005640  005015  000000  005015  041522    040531  020110  040514  052040
 005650  005015  000000  000000  000000    044517  050040  051524  047520
 005660  020103  052522  051105  047124    020117  053105  051114  040531
 005670  035040  020040  020040  020040    020040  020040  005015  000000
 005700  020120  052516  052040  053501    051440  044516  020124  044105
 005710  020113  042522  047105  046040    005015  000000  000000  000000
 005720  000000  000000  000000  000000    000000  000000  000000  000000
 005730  004441  017130  000000  000000    002301  017352  000000  000000
 005740  001641  017466  000000  000000    001201  017560  000000  000000
 005750  000661  017630  000000  000000    006521  017663  000000  000000
 005760  002040  030610  000000  000000    010420  030712  000430  032560
 005770  011000  031333  000430  045740    001300  031773  001430  000000
 006000  006760  032047  000430  000000    006560  032406  001430  000000
```

LOCATING THE MACRO

Found through Exit Stack contents.

Exit stack can be found in two places.

R on FDUMP directives gives all registers

EXSAVE area

E Pointer-1 give $PUNTIF Address.

E Pointer in Punt of KERNEL.

Operand registers give overlay number.

```
DMEM,TYPE=IOP1,FWA=4000,LWA=12000,R.          IOP1 LOCAL MEMORY          FDUMP 1.14      01/04/85    18:20:31    PAGE    16
                                               IOP-1 INTERNAL REGISTERS SYSDUMP           01/03/85    23:55:42

P=016313
A=000000    C=1
B=014
EXIT STACK:   E=05
  00  017157    003545    003613    003710
  04  012141    016313    016201    016201
  10  000000    000000    000000    000000
  14  000000    000000    000000    000000
```

```
DMEM,TYPE=IOP1,FWA=4000,LWA=12000,R.          IOP1 LOCAL MEMORY          FDUMP 1.14      01/04/85    18:20:31    PAGE    14
                                               IOP-1 OPERAND REGISTERS  SYSDUMP          01/03/85    23:55:42

000000000    000000  012224  005155  025530    000000  000000  045310  045334
000000010    000045  000666  000000  000000    000000  000000  000000  000001
000000020    000006  000001  057301  060000    000006  000001  000000  000000
000000030    000000  005353  005357  042567    000000  000000  000000  000000
000000040    000000  177777  043774  000000    000000  005156  005157  000020
000000050    045224  000024  051770  005157    060000  000023  000000  000000
000000060    017240  100313  005220  000010    016276  000000  000000  000000
000000070    000000  000000  000001  000000    001174  000000  000456  000000
000000100    000001  025536  000000  007655    000005  000033  000000  000001
000000110    000000  000045  075674  000107    104423  000000  000106  175761
000000120    042560  005730  000000  002201    000043  000000  000000  013652
000000130    000000  026670  000000  013110    000000  025250  025064  005770
000000140    000004  000000  000040  005563    027730  000001  000000  000000
000000150    000000  011345  000005  000001    000004  000000  000000  025520
000000160    002210  025074  000014  060000    000000  000000  046154  001104
000000170    000000  000003  000004  000000    000000  000000  037054  000000
000000200    000000  000000  000001  000000    000000  000001  000001  041554
000000210    000000  046154  000000  000001    041613  005433  041254  005431
000000220    000001  057307  000001  024006    000006  000000  000002  000000
000000230    013730  041254  000010  000006    000001  057301  060000  000006
000000240    000010  000037  107517  062500    000260  000000  000000  000000
000000250    000000  000000  046314  050116    050340  050637  042520  177777
000000260    000005  001000  000000  000000    000000  000000  000000  000000
```

HANG MACRO

```
I/O PROCESSOR KERNEL, VERSION 1.14, SN12, MENDOTA HEIGHTS          IOP  APML X.15(12/21/84) 12/21/84 15:17:29  Page 48
COMMON MEMORY CONTROL                                              Block: TWO                              (48)

                                        * TIMEOUT                                                                K.1726
                                                                                                                 K.1727
12111    <macro>                        $IF (A = MIO$TOUT)                                                       K.1728
                                                                                        .>>>>> 1                  $IF      1
12115   002000  000000                   I = 0
12117   010002  024025                   RICF = MIO$TMO          .SET ERROR TYPE                                  K.1730
12121   070005                           P = CEREC               .GOTO ERROR ROUTINE                              K.1731
12122    <macro>                        $ENDIF                                                                   K.1732
                                                                                        .<<<<< 1                  $ENDIF   1
12122    <macro>                       $ENDIF                                                                    K.1733
                                                                                        .<<<<< 0                  $ENDIF   1
12122    <macro>                      $ENDTIL                                                                    K.1734
12123   010001  024025                RICF = MIO$DTE             .DEFAULT TO DATA ERROR                          K.1735
12125   001000                       EXIT                                                                       K.1736
                                                                                                                 K.1737
                                ************************************************************************     K.1738
                                *                     MEMORY I/O ERROR RECOVERY                        *     K.1739
                                *                                                                      *     K.1740
                                *   B    - CHANNEL TO RECOVER                                          *     K.1741
                                *   RICL - CURRENT RETRY COUNT                                         *     K.1742
                                *   RICF - ERROR CODE                                                  *     K.1743
                                *                                                                      *     K.1744
                                ************************************************************************     K.1745
                                                                                                                 K.1746
12126                           CEREC   *                                                                        K.1747
12126    <macro>                        $PUNTIF (B = RI%HISP) OR (B = RI%HISP + 1),CODE=PT$HISP                  K.1748
                                                                                        .>>>>> 0                  $IF      2
                                                                                        .<<<<< 0                  $ENDIF   2
12142   160000                         IOB:0                   .CLEAR CHANNEL                                   K.1749
                                                                                                                 K.1750
                                       * IF IN MIOP - CHECK ERROR LOG CHANNEL                                    K.1751
                                       *              IF SET, LET IT RECORD ERROR                               K.1752
                                                                                                                 K.1753
12143    <macro>                       $IF (RI%MYID = MIOP) AND (ERA = ON)                                       K.1754
                                                                                        .>>>>> 0                  $IF      1
12151   077000 /030025                  R = IREPORTO             .RECORD ERROR                                   K.1755
12153    <macro>                       $ENDIF                                                                    K.1756
                                                                                        .<<<<< 0                  $ENDIF   1
                                                                                                                 K.1757
                                       * UPDATE RETRY COUNT - SENSE LIMIT                                        K.1758
                                                                                                                 K.1759
12153   026026                         RICL = RICL + 1                                                           K.1760
                                                                                                                 K.1761
                                       * ERROR FATAL IF RETRY LIMIT EXCEEDED                                     K.1762
                                                                                                                 K.1763
12154    <macro>                       $IF (A > MIO$MAX)                                                         K.1764
                                                                                        .>>>>> 0                  $IF      1
12163    <macro>                         $PUNTIF (B = [MOS]),CODE=PT$MOS  .MOS ERROR                             K.1765
12167    <macro>                         $PUNTIF CODE=PT$HISP        .HIGH SPEED CHANNEL ERROR                   K.1766
12171    <macro>                       $ENDIF                                                                    K.1767
                                                                                        .<<<<< 0                  $ENDIF   1
                                                                                                                 K.1768
                                       * RESET PARAMETERS FROM SAVED SET                                         K.1769
                                       * AND RETRY THE I/O                                                       K.1770
                                                                                                                 K.1771
12171    <macro>                       $IF (B = [MOS])             .IF BUFFER MEMORY                             K.1772
                                                                                        .>>>>> 0                  $IF      1
```

HISTORY TRACE

Provides a means for tracing, to a certain degree, the path of execution through the code.

Stores pertinent data relating to selected events in a local memory buffer.

Local trace buffer is dumped to a circular buffer in buffer memory.

May be used in debugging and fine tuning the system.

See SM-46 page 11-4 for more detail.

Parameters depend on type of entry printed in format by FDUMP.

HISTORY TRACE

DXTR TYPE=IOPO

EVENT	RTC-L	OVL-#	PAR-1	PAR-2	PAR-3	PAR-4	PAR-5
I-HAND	007655	CONSL	000006	177004	050554	000057	000010
TASK	007655	CONSL	056004	001067	000426	076560	000010
MEM-IO	007655	CONSL	000002	000001	037063	100164	000021
K-CALL	007655	CONSL	MOSR	001067	065133	076560	000456
TASK	007655	CONSL	056004	000006	000426	076560	000010
K-FNCT	007655	BABEL	CALL	065133	000426	010060	006430
K-CALL	007655	BABEL	CALL	001223	065065	076560	000462
TASK	007655	BABEL	057634	001417	000372	076560	000010
K-CALL	007655	BTD	RETURN	000227	065133	076560	000450
TASK	007655	BTD	057350	000006	000016	076560	000010
OVL-LD	007655	BTD	000016	057350	000021	000000	025066
MEM-IO	007655	BABEL	000001	000000	032735	057350	000053
K-FNCT	007655	BABEL	CALL	065133	000016	006020	007430
K-CALL	007655	BABEL	CALL	001417	065065	076560	000462
INTRPT	007655	BABEL	000021	061057	057634	000000	131214
INTRPT	007655	BABEL	000016	061057	057634	000000	131214
INTRPT	007655	BABEL	000007	061057	057634	000000	131214
TASK	007655	BABEL	057634	001223	000372	076560	000010
MEM-IO	007655	KERNEL	000001	000002	111200	064770	000033
MEM-IO	007655	KERNEL	000004	000002	107200	064770	000024
A-TO-A	007655	CRAYIO	000007	100313	000043	000000	000000
MEM-IO	007655	CRAYIO	000003	000000	013740	005030	000010
K-FNCT	007655	CRAYIO	FE-IO	000001	060000	001000	000000
K-CALL	007655	CRAYIO	FE-IO	000707	065031	073410	000464
INTRPT	007655	CRAYIO	000007	035553	034674	000000	131214
TASK	007655	CRAYIO	034674	000657	000430	073410	000010
A-TO-A	007655	CRAYIO	000007	100312	000043	000000	000000
MEM-IO	007655	CRAYIO	000003	000000	013730	005030	000010
K-FNCT	007655	CRAYIO	FE-IO	000001	024006	000006	000001
K-CALL	007655	CRAYIO	FE-IO	000657	065031	073410	000464
TASK	007655	CRAYIO	034674	000436	000430	073410	000010
K-CALL	007655	CRAYIO	RELDAL	000436	065031	073410	000464
TASK	007655	CRAYIO	034674	000366	000430	073410	000010
MEM-IO	007655	KERNEL	000001	000002	107200	064770	000024
MEM-IO	007655	KERNEL	000004	000002	111200	064770	000033
K-CALL	007655	CONSL	RETURN	000677	065133	076560	000456
TASK	007654	CONSL	056004	001067	000426	076560	000010
MEM-IO	007654	CONSL	000002	000001	037042	100164	000021
K-CALL	007654	CONSL	MOSR	001067	065133	076560	000456
TASK	007654	CONSL	056004	000006	000426	076560	000010
K-FNCT	007654	BABEL	CALL	065133	000426	010060	006430
K-CALL	007654	BABEL	CALL	001223	065065	076560	000462
INTRPT	007654	BABEL	000020	057642	057634	000000	131214
I-HAND	007654	BABEL	000020	000000	070300	070340	000102
TASK	007654	BABEL	057634	000006	000372	076560	000010
OVL-LD	007654	BABEL	000372	057634	000020	000000	025065
MEM-IO	007654	DISPO1	000001	000000	145317	057634	000347
K-FNCT	007654	DISPO1	CALL	065065	000372	007700	005430
K-CALL	007654	DISPO1	CALL	000444	065021	076560	000461
TASK	007654	DISPO1	041130	003624	000437	076560	000010
K-CALL	007654	MULTIP	RETURN	000063	065066	076560	000440

DXTR TYPE=IOP1

EVENT	RTC-L	OVL-#	PAR-1	PAR-2	PAR-3	PAR-4	PAR-5
MEM-IO	007655	AMSG	000006	000001	057301	060000	000006
K-CALL	007655	AMSG	TRANSF	001174	045334	045310	000457
INTRPT	007655	AMSG	000006	025536	025530	000045	075674
I-HAND	007655	AMSG	000006	100313	047125	000077	000023
TASK	007655	AMSG	025530	000006	000010	045310	000023
ACOM	007655	ACOM	000043	041254	100312	000000	000000
MEM-IO	007655	ACOM	000002	000000	013730	041254	000010
TASK	007655	ACOM	031064	000006	000007	045250	000023
INTRPT	007655	KERNEL	000006	013107	000000	000045	075674
I-HAND	007655	KERNEL	000006	100312	047124	000100	000023
INTRPT	007654	KERNEL	000007	013114	000000	000045	065043
A-TO-A	007654	AMSG	000007	100311	000044	000000	000400
MEM-IO	007654	AMSG	000003	000000	013720	042014	000010
TASK	007654	AMSG	025530	001174	000010	045310	000023
MEM-IO	007654	AMSG	000010	000037	107517	062500	000260
MEM-IO	007654	AMSG	000002	000001	060520	062500	000260
MEM-IO	007654	AMSG	000010	000037	106777	060000	000520
MEM-IO	007654	AMSG	000002	000001	060000	060000	000520
K-CALL	007654	AMSG	TRANSF	001174	045334	045310	000457
TASK	007654	AMSG	025530	000006	000010	045310	000023
ACOM	007654	ACOM	000043	042014	100311	000000	000000
INTRPT	007654	ACOM	000007	012036	031064	000045	064637
MEM-IO	007654	ACOM	000002	000000	013720	042014	000010
TASK	007654	ACOM	031064	000006	000007	045250	000023
A-TO-A	007654	AMSG	000007	100310	000044	000000	000400
MEM-IO	007654	AMSG	000003	000000	013710	037554	000010
TASK	007654	AMSG	025530	001174	000010	045310	000023
MEM-IO	007654	AMSG	000010	000001	057271	060000	000006
MEM-IO	007654	AMSG	000002	000001	024000	060000	000006
K-CALL	007654	AMSG	TRANSF	001174	045334	045310	000457
INTRPT	007654	AMSG	000006	025536	025530	000045	064637
I-HAND	007654	AMSG	000006	100311	047123	000077	000023
TASK	007654	AMSG	025530	000006	000010	045310	000023
ACOM	007654	ACOM	000043	037554	100310	000000	000000
MEM-IO	007654	ACOM	000002	000000	013710	037554	000010
TASK	007654	ACOM	031064	000006	000007	045250	000023
INTRPT	007654	KERNEL	000006	013114	000000	000045	064637
I-HAND	007654	KERNEL	000006	100310	047122	000100	000023
INTRPT	007476	KERNEL	000007	013110	000000	000035	107710
A-TO-A	007476	AMSG	000007	100307	000044	000000	000000
MEM-IO	007476	AMSG	000003	000000	013700	037314	000010
TASK	007476	AMSG	025530	001174	000010	045310	000023
MEM-IO	007476	AMSG	000004	000001	057520	062500	000260
MEM-IO	007476	AMSG	000006	000037	107517	062500	000260
MEM-IO	007476	AMSG	000004	000001	057000	060000	000520
INTRPT	007476	AMSG	000007	012044	025530	000035	107710
MEM-IO	007476	AMSG	000006	000037	106777	060000	000520
K-CALL	007476	AMSG	TRANSF	001174	045334	045310	000457
TASK	007476	AMSG	025530	000006	000010	045310	000023
ACOM	007476	ACOM	000043	037314	100307	000000	000000
MEM-IO	007476	ACOM	000002	000000	013700	037314	000010

ERRBUFF

Error logger entry buffer

512 word buffer for error logger

Before IOP #27

After IOP #27

See error logger HM-29, page 7-60 for breakdown

A single bit hit in Buffer Memory is reported.

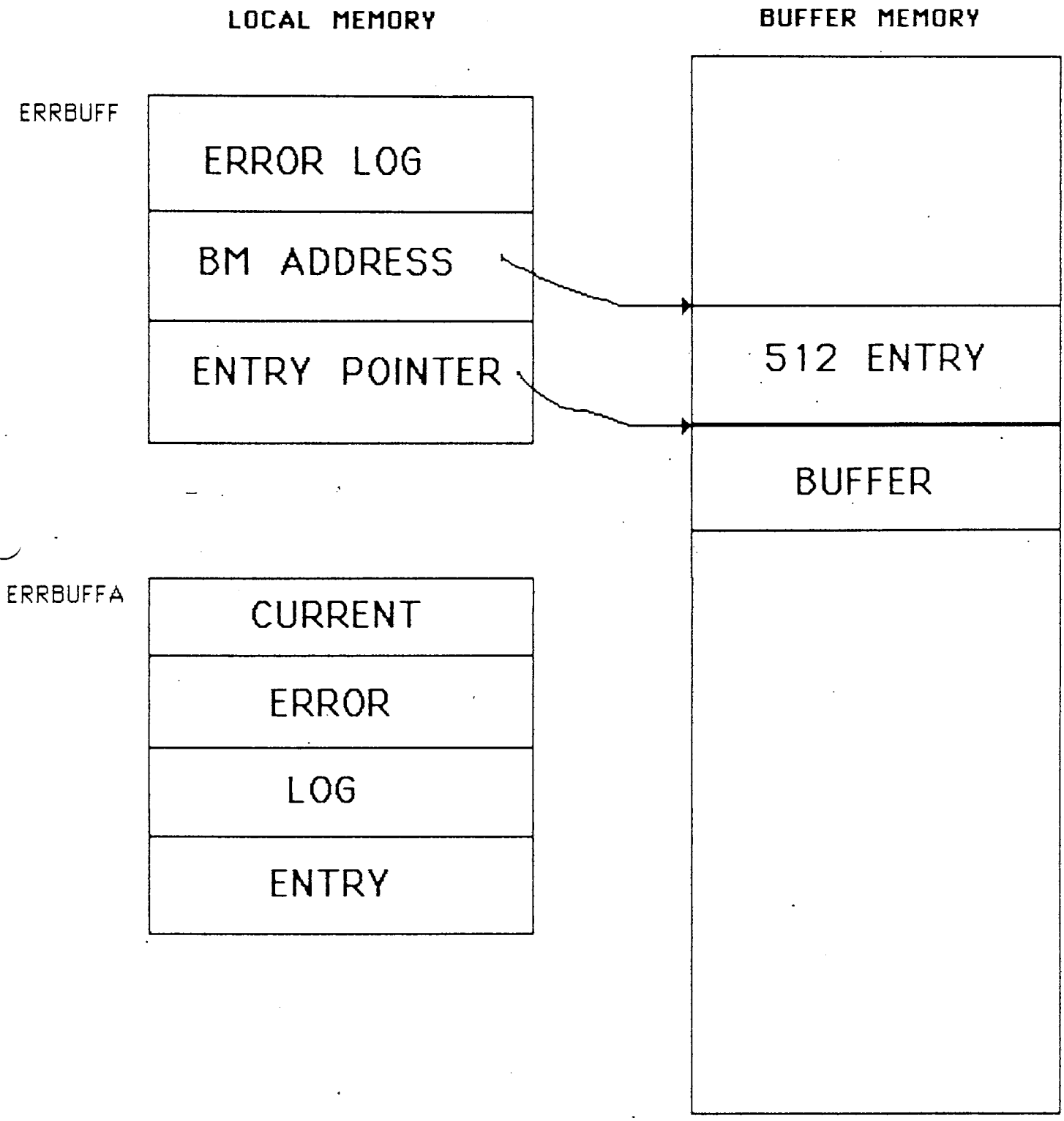
A double bit hit in Buffer Memory will hang the system.

A single bit hit in Local Memory (2AW-majority rule) is undetectable or reportable.

A parity error will hang the system.

A high speed channel error will hang the system.

ERRBUFF


LOCAL MEMORY
BUFFER MEMORY
ERRBUFF
ERROR LOG
BM ADDRESS
ENTRY POINTER
512 ENTRY
BUFFER
ERRBUFFA
CURRENT
ERROR
LOG
ENTRY

POINTERS AND TABLES

Operand registers contain pointers

Local memory has a table area

Tables are not labeled, to find a table you need matching listings

Match up with ASCII label tables

Demon Queue Pointers

Disk Control Block Printers

CRT Buffers

EACT - Activity Descriptor Chain Pointers

Buffer Memory Allocation

Disk Buffer Pointes DAL Packet Pointers

EITB - Interrupt Handler Table

Interprocessor MSG Queue

Local Memory Errors

Buffer Memory Errors

Halt Message

Overlay Table

OPERAND REGISTER POINTERS


```
                        ************************************************************************      K.43
                        *                                                                      *      K.44
                        *                                                                      *      K.45
                        *       Kernal operand registers                                       *      K.46
                        *                                                                      *      K.47
                        ************************************************************************      K.48
                                                                                                      K.49
                                BASEREG   AA                                                          K.50
                                SCRATCH   AB,AC

                                                                                                      K.52
0   <macro>                 0   REGISTER  AA              .Base register                              $REG   2
                        0       EQUALS    $REGORG                                                     K.53
0   <macro>                     REGISTER  (AB,AC)         .APML scratch registers                     $REG   2
                        1   AB  EQUALS    $REGORG                                                     $REG   2
                        2   AC  EQUALS    $REGORG                                                     K.54
0   <macro>                     REGISTER  %B              .Overlay base address                       $REG   2
                        3   %B  EQUALS    $REGORG                                                     K.55
0   <macro>                     REGISTER  BB                                                          $REG   2
                        4   BB  EQUALS    $REGORG                                                     K.56
0   <macro>                     REGISTER  %PSMOD          .Previous SMOD address                      $REG   2
                        5   %PSMOD EQUALS $REGORG                                                     K.57
0   <macro>                     REGISTER  %ACTIVE         .Activity descriptor address                $REG   2
                        6   %ACTIVE EQUALS $REGORG                                                    K.58
0   <macro>                     REGISTER  %SMOD           .Current SMOD address                       $REG   2
                        7   %SMOD  EQUALS $REGORG                                                     K.59
0   <macro>                     REGISTER  %FUNC           .Kernel service request type                $REG   2
                        10  %FUNC  EQUALS $REGORG                                                     K.60
0   <macro>                     REGISTER  %EX             .Kernel call entrance                       $REG   2
                        11  %EX    EQUALS $REGORG                                                     K.61
0   <macro>                     REGISTER  (BD,BE,BF,BG,BH) .CRT I/O                                   $REG   2
                        12  BD     EQUALS $REGORG                                                     $REG   2
                        13  BE     EQUALS $REGORG                                                     $REG   2
                        14  BF     EQUALS $REGORG                                                     $REG   2
                        15  BG     EQUALS $REGORG                                                     $REG   2
                        16  BH     EQUALS $REGORG                                                     K.62
0   <macro>                     REGISTER  %INIT           .Initialization complete flag               $REG   2
                        17  %INIT  EQUALS $REGORG                                                     K.63
0   <macro>                     REGISTER  (CA,CB,CC,CD,CE,CF) .MOS and Lisp I/O                       $REG   2
                        20  CA     EQUALS $REGORG                                                     $REG   2
                        21  CB     EQUALS $REGORG                                                     $REG   2
                        22  CC     EQUALS $REGORG                                                     $REG   2
                        23  CD     EQUALS $REGORG                                                     $REG   2
                        24  CE     EQUALS $REGORG                                                     $REG   2
                        25  CF     EQUALS $REGORG                                                     K.64
0   <macro>                     REGISTER  CL                                                          $REG   2
                        26  CL     EQUALS $REGORG                                                     K.65
0   <macro>                     REGISTER  %DEBBA          . TEMP **************************       $REG   2
                        27  %DEBBA EQUALS $REGORG                                                     K.66
0   <macro>                     REGISTER  %DEBUG          .Debugger base address                      $REG   2
                        30  %DEBUG EQUALS $REGORG                                                     K.67
0   <macro>                     REGISTER  (ED,EE,EF,EG)   .Kernel registers                           $REG   2
                        31  ED     EQUALS $REGORG                                                     $REG   2
                        32  EE     EQUALS $REGORG                                                     $REG   2
                        33  EF     EQUALS $REGORG                                                     $REG   2
                        34  EG     EQUALS $REGORG                                                     $REG   2
0   <macro>                     REGISTER  (EH,EI,EJ,EK)                                               K.68
                        35  EH     EQUALS $REGORG                                                     $REG   2
```

KERNEL TABLE POINTERS

```
I/O PROCESSOR KERNEL, VERSION 1.14, SN12, MENDOTA HEIGHTS        IOP  APML X.15(12/21/84) 12/21/84 15:17:29  Page 25
tables and data areas                                                                                      (25)

                                ************************************************************************      K.598
                                *                                                                      *      K.599
                                *                                                                      *      K.600
                                *       Preallocated tables and data areas                             *      K.601
                                *                                                                      *      K.602
                                ************************************************************************

                                                                                                               K.604
                                        BLOCK   ONE                                                            K.605
1140w  025124202411142125145 2  CON     '*TABLES*'                                                             K.606
       <edit>                   CON     '"SDATE"'                                                               *EDITED  0
1141w  030462136310611363406 4  CON     '12/21/84'                                                              *EDITED  0
                                                                                                               K.607
       <edit>                   CON     '"STIME"'                                                               *EDITED  0
1142w  030465164304671643107 1  CON     '15:17:29'                                                              *EDITED  0
                                ,

                                ************************************************************************      K.609
                                *                                                                      *      K.610
                                *                                                                      *      K.611
                                *       Execution control flags                                        *      K.612
                                *                                                                      *      K.613
                                ************************************************************************      K.614
                                                                                                               K.615
4614   000000                   BNFLG   FDATA   I@BNFLG              .0 = no block number check

                                *       DEBLOC is the location to which the Debugger is copied to              K.618
                                *       allow debugging during IOP initialization.  The Debugger               K.619
                                *       relocatable code occupies a maximum of 51 words.                       K.620
                                                                                                               K.621
                                                                                                               K.622
                        174004  DEBLOC  EQUALS  O'174004

                                *       Disk domon queue.  The queue entries are linked via cell               K.625
                                *       DD@DEM in the DCB.                                                     K.626
                                                                                                               K.627
4615   000000                   DISKQ   0                              .Pointer to first entry                K.628
4616   000000                   DISKQT  0                              .Pointer to last entry                 K.629

                                        ****************************************                               K.631
                                        *                                      *                               K.632
                                        *  DD49 spiral conversion table        *                               K.633
                                        *                                      *                               K.634
                                        ****************************************                               K.635
                                                                                                               K.636
                                                                                                               K.637
4617   000000                   D4CNVRT 0                                                                      K.638
4620   000012                           D'10                                                                   K.639
4621   000025                           D'21                                                                   K.640
4622   000037                           D'31

                                        ********************************                                       K.642
```

MEMORY CHAIN HEADERS

SYMBOL	ADDRESS	CONTENTS
EDES		Pointer to First Free DAL Entry
EDES + 1		Pointer to Last Free DAL Entry
EDNM		Number of Free DAL Entries Available
EMEM		Pointer to First Free Memory Entry
EMEMSIZ		Maximum Free Memory Available
EBUF		Pointer to First Free I/O Buffer
EBUH		Pointer to Last Free I/O Buffer
EBNM		Number of I/O Buffers Available
EBUFSIZ		Maximum Number of I/O Buffers Available

FINDING THE EXECUTING ENVIRONMENT

This section is a list intended to serve as a guide to analyzing I/O Subsystem dumps.

Record any operator interactions with the Kernel for later reference. Record time and date dump was taken.

Examine punt code in message. Get meaning of coded dumps from LISTP key-in.

Take dump according to Cray Research systems analyst's specifications.

Dump Kernel tables, DALS, non-Kernel tables and free memory, Disk Control Blocks (DCBs).

Make sure listing matches dump. Assembly date in listing should match date in Deadstart Done message. If they do not match, the approximate difference in the Kernel can be determined from the dump by finding IAA in the listing and comparing it to the first entry in the exit stack.

Determine overlay base address from operand register 3 (%B). If zero, the halt occurred within the Kernel resident area.

Determine overlay number currently running from operand register 63 (%OVLNUM).

Determine overlay from LISTO listing or by examining address in operand register 3. (First eight characters in an overlay is its name in ASCII.)

Get exit stack (E) pointer from location EXSAVE+3. EXSAVE is a label in the Kernel immediately after the Kernel tables. Its format is:

0 - Contents of A at halt
1 - Carry (C)
2 - B register
3 - Exit Stack pointer (E)
4-23 - Exit Stack
24-163 - DN + BZ (done and busy status) of all channels at halt

Get P from EXSAVE+4+E-1. P is the address of a return jump or point in code when an interrupt occurred.

If P is in an overlay, determine relative P by subtracting base address (%B). (P is in an overlay if P greater than %B.)

Once the appropriate $PUNTIF macro is found, use trace dump to determine sequence of events leading up to halt (section 10 or this manual describes trace entries).

EXSAVE LAYOUT AFTER A HALT

*EXSAVE *
A
C
B
E
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0 BZ
0 DN
•
•
•
47 BZ
47 DN

Examine information in history traces:

The time of the trace entry in millisecond increments is saved, which gives useful impression of length of time between events.

The overlay in control at the time of the trace is always saved. If the Kernel is in control (usually in the idle loop), the overlay number is 177777.

On task activation traces ('TASK' - type 3) the address of the Activity Descriptor is saved. This information is useful to find which of the activities in the system is doing what and to analyze activity conflicts and possible timing problems.

Service requests to the Kernel are always traced and are usually helpful to understand sequence of an activity's functions. In FDUMPs they are printed as K-FNCT (in raw dumps they are type 5).

If you suspect what is causing a halt (for example, A RELMEM) but cannot find it in trace (it may have happened too far in the past), try to recreate halt but with only one or two specific trace events enabled.

Look for a pattern in the trace dump that would indicate that the system is looping through a group of overlays repeatedly. Try to isolate the reason for the loop.

If the Kernel does not halt but seems to hang up and you are unable to access the system via the Kernel CRT, there may be an infinite loop in an overlay.

Find the overlay in register 3 (%B).

Examine the operand registers to determine the extent and reason for the loop.

The done (DN) and busy (BZ) states of all channels are saved in the dump immediately after the exit stack. Although they are normally printed by FDUMP, you must have some more information when reading a raw dump:

If the channel is DN, the first parcel of the two for this channel is set to 1.

If the channel is BZ, the second parcel is set to 1.

Accumulator channels from the IOPs for communication usually have the input half busy (BZ).

The command channel for input from the Cray mainframe is normally awaiting input and is busy (BZ).

The CRT input channels are usually waiting for characters to be entered from the keyboard and thus are busy (BZ).

Important Operand Registers

Symbol	Value	Description
AA	0	Base address of Kernel (always zero)
AB,AC	1,2	APML Scratch Registers
%B	3	Base address of overlay (zero if none)
%ACTIVE	6	Current Activity Descriptor (177777 if none)
%SMOD	7	Address of SMOD being run (177777 if none)
%OVLNUM	63	Pointer to current overlay number (177777 if none)
%PUNT	64	Address of Punt Routine
RD,RE	100,101	E and (E) at last interrupt
%CLKU,%CLKL	102,103	Kernel Real-time Clock
%ICHAN	106	Channel of last interrupt (zero if all are processed)
%HISP	165	Channel number of high speed channel (zero if none present)

Examine dedicated Kernel registers for irregularities.

Examine Activity Descriptors (AD) to see if information in them is meaningful. The last AD activated will have its address in %ACTIVE (operand register 6) if it is active. All Activity Descriptors in the IOP are linked together through parcel 1 (KA$ALNK) of the Activity Descriptors. The chain begins at EACT+1.

Examine SMOD area in Local Memory to find registers at time of last service request to Kernel. Address of SMOD area is found in location ESMD in the Kernel tables. There is one SMOD area for each IOP, used only when an activity is active. Other SMODs are saved in Buffer Memory at an address found in the AD at location AD@MSU (parcels 3 and 4).

Things found in SMOD that are good clues to problems are:

Activity Descriptor addresses in local memory

Size of SMOD entry

Overlay table address of overlay making last call

A,B,C,E

Exit stack

Operand registers saved on last CALL

Global registers if SMOD just read into memory

For disk-related problems, examine DCBs to see if any irregularities are evident. Note the following items.

Pointers to DCBs are found in Kernel tables DCCB through DCCB+17. (The last DCB to have any processing done is often in register DA register 200.)

Channels that are being used have a nonzero value in parcel 0 (DD@FLG) and addresses of executable DALs are in parcel 2 (DD@EDL).

The DCB contains a count of recovered and unrecovered errors on the channel and saved information about the last unrecovered error (such as cylinder, head, section).

PROGRAM EXECUTION ENVIROMENT

PROCESSOR QUEUE - ECPQ

OVERLAY NUMBER %OVLNUM -

OVERLAY NAME -

OVERLAY BASE ADDRESS %B -

ACTIVE AD %ACTIVE -

ACTIVE SMOD %SMOD -

CURRENT AD SS ESMD -

LOCATION 0 OF SS -

LOCATION AD+SMD@P -

SMOD+SM@CAL@P -

LAST HISTORY TRACE EVENT -

PROGRAM EXECUTION ENVIRONMENT

PROCESSOR QUEUE - ECPQ

OVERLAY NUMBER %OVLNUM -

OVERLAY NAME -

OVERLAY BASE ADDRESS %B -

ACTIVE AD %ACTIVE -

ACTIVE SMOD %SMOD -

CURRENT AD SS ESMD -

LOCATION 0 OF SS -

LOCATION AD+SMD●P -

SMOD+SM●CAL●P -

LAST HISTORY TRACE EVENT -

IOS DUMP ANALYSIS

Match the Following terms and their description:

1. HALT MESSAGE	____ Operand register containing the address the executing overlay is loaded at
2. $PUNTIF	____ Points to the Hang macro address to help locate it in the listing
3. %B	____ Operand register containing the overlay number that was executing
4. EXIT STACK	____ Area in the Kernel tables where the IOP s A, B, C, E, Exit stack and Busy Done flags are saved on a $PUNTIF
5. CONTENTS OF E-1 IN STACK	____ Operand register containing the address the active SMOD is at
6. %OVLNUM	____ Return jump history stack after an interrupt or an exit
7. %ACTIVE	____ Macro executed by the IOS to halt on an error condition
8. EXSAVE	____ Operand register containing the address of the active activity descriptor
9. BUSY and DONE flag set	____ Tells you which IOP halted and the punt code attached to the $PUNTIF macro
10. %SMOD	____ High Speed Channel Error

IOS DUMP ANALYSIS

Match the following terms and their description.

1. HALT MESSAGE
2. $PUNTIF
3. RB
4. EXIT STACK
5. CONTENTS OF E-1 IN STACK
6. $OVLNUM
7. $ACTIVE
8. EXSAVE
9. BUSY and DONE Flag set
10. $SMOD

___ Operand register containing the address the executing overlay is loaded at

___ Points to the Hang macro address to help locate it in the listing

___ Operand register containing the overlay number that was executing

___ Area in the Kernel tables where the IOP's A, B, C, E, Exit stack and Busy Done flags are saved on a $PUNTIF

___ Operand register containing the address the active SMOD is at

___ Return Jump history stack after an interrupt or an exit

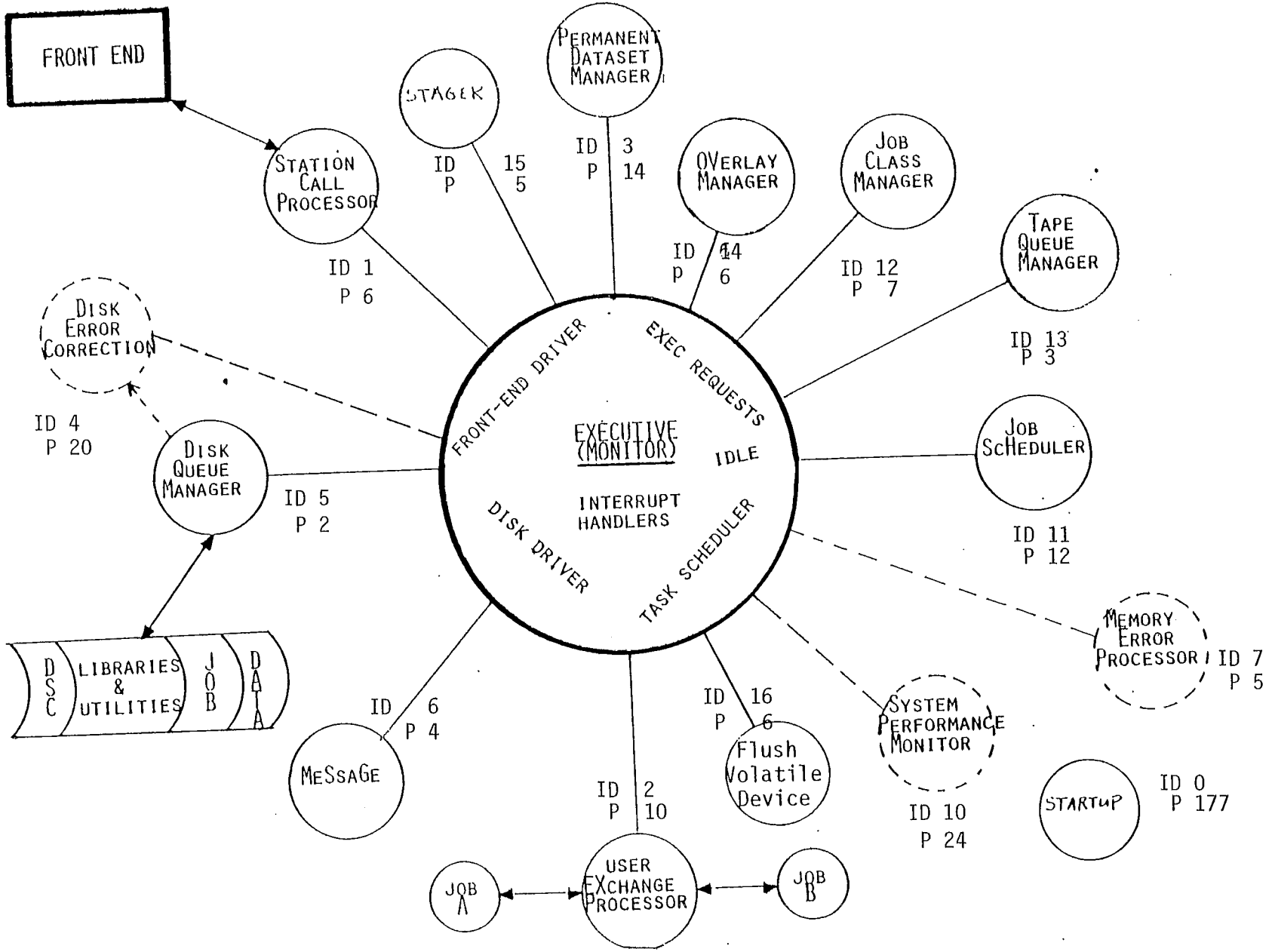
___ Macro executed by the IOS to halt on an error condition

___ Operand register containing the address of the active activity descriptor

___ Tells you which IOP halted and the punt code attached to the $PUNTIF macro

___ High Speed Channel Error

Appendix A - Quick References


FRONT END
STATION CALL PROCESSOR
ID 1
P 6
STAGER
ID
P
15
5
PERMANENT DATASET MANAGER
ID 3
P 14
OVERLAY MANAGER
ID 14
P 6
JOB CLASS MANAGER
ID 12
P 7
TAPE QUEUE MANAGER
ID 13
P 3
JOB SCHEDULER
ID 11
P 12
MEMORY ERROR PROCESSOR
ID 7
P 5
SYSTEM PERFORMANCE MONITOR
ID 10
P 24
STARTUP
ID 0
P 177
Flush Volatile Device
ID 16
P 6
USER EXCHANGE PROCESSOR
ID 2
P 10
JOB A
JOB B
MESSAGE
ID 6
P 4
DISK QUEUE MANAGER
ID 5
P 2
DISK ERROR CORRECTION
ID 4
P 20
DSC
LIBRARIES & UTILITIES
JOB
DATA
FRONT-END DRIVER
EXEC REQUESTS
EXECUTIVE (MONITOR)
IDLE
INTERRUPT HANDLERS
DISK DRIVER
TASK SCHEDULER

OVERLAY DESCRIPTIONS C

This appendix describes the overlays used by I/O Subsystem software. The list is alphabetical according to overlay name. Each overlay has one or more of the following type codes.

Type	Description
D	Data overlays. These overlays contain data and cannot be executed.
K	Kernel overlays
F	File system overlays associated with the I/O Subsystem Editor. See Appendix F of the I/O Subsystem (IOS) Operator's Guide, CRI publication SG-0051, for a description of the capabilities of the Editor.
C	Concentrator overlays
S	Station overlays
I	Interactive station overlays
B	Block multiplexer overlays. These overlays drive the block multiplexer channels.
T	Tape overlays
R	System dump and restart overlays
Ns	NSC (Network Systems Corporation) channel concentrator overlays

KERNEL HALT (PUNT) CODES

CODE	MEANING
ØØØ	No error code specified on $PUNTIF macro
ØØ1	Local memory error (always HARDWARE)
ØØ2	Buffer memory error on deadstart (always HARDWARE)
ØØ3	Buffer memory error (always HARDWARE)
ØØ4	High-speed channel error (always HARDWARE)
ØØ5	Invalid message received from CPU
ØØ6	Invalid parameter in disk request from CPU
ØØ7	Program was executing at location Ø
Ø1Ø	Local memory location Ø was overwritten
Ø11	Undefined message received on IOP communication channel
Ø12	Overlay does not exist
Ø13	Station stack overflow or underflow
Ø14	Local memory buffer not available
Ø15	Buffer memory disk buffer not available
Ø16	Invalid local buffer release call
Ø17	Buffer memory incorrectly configured
Ø2Ø	IOP message channels incorrectly configured
Ø21	SMOD is too large for area on buffer memory
Ø22	Invalid local memory address
Ø23	Illegal interrupt program sequence code
Ø24	Stop request received from CPU
Ø25	Low-speed channel error (always HARDWARE)
Ø26	Block number validation trap
Ø4Ø	Block Mux interrupt processor error
Ø41	Bad CRW address in device table
Ø42	Block mux start I/O error
Ø44	Block Mux configuration error
Ø5Ø	DD49 disk software error
Ø51	The debugger was not loaded
Ø52	Bad buffer memory allocation request
Ø53	Bad local memory address on hi speed I/O call
Ø54	Invalid I/O length specified
Ø55	No hi speed channel configured for request
Ø56	Illegal activity activation requested
Ø57	Buffer memory DAL queue exhausted
Ø6Ø	Illegal Demon call
Ø61	Undefined kernel service request
Ø62	Illegal kernel service request
Ø63	Bad kernel service request parameter
Ø64	Requested device not configured
Ø65	Illegal IOP requested on kernel service request
Ø66	Requested queue full
Ø67	Illegal I/O address specified
Ø7Ø	SMOD error
Ø71	Local memory space exhausted
Ø72	Illegal overlay load requested
Ø73	Corrupted local memory chain
Ø74	Bad local memory release
Ø75	Bad I/O parameter
Ø76	Unexpected interrupt received
Ø77	Disk error
1ØØ	AMAP not available for system initialization
1Ø1	Illegal overlay number read during initialization
1Ø2	IOP initialization error
1Ø3	MOS configuration error
1Ø4	Overlay too large for loading
1Ø5	Premature tape EOF encountered
1Ø6	Exit stack fault
1Ø7	Illegal device type configured

No.	Ovl Name	Size	BM-Upr	BM-Lwr
ØØ	SDMPA	111Ø	Ø1	Ø21373
Ø3	SDMPD	Ø24Ø	Ø1	Ø22Ø23
Ø6	AMAP	Ø41Ø	ØØ	Ø2Ø67Ø
11	BLOCK	1534	ØØ	Ø21756
14	BTD	Ø254	ØØ	Ø22305
17	BXSET	Ø34Ø	ØØ	Ø22524
22	CHNTST	Ø774	ØØ	Ø23747
25	CONFIG	1334	ØØ	Ø24573
3Ø	CPTEST	237Ø	ØØ	Ø25142
33	DISKIO	1Ø24	ØØ	Ø2656Ø
36	DKIOEX	Ø43Ø	ØØ	Ø3ØØ12
41	DKSETØ	Ø26Ø	ØØ	Ø3Ø22Ø
44	DOMP	2554	ØØ	Ø3Ø663
47	ERR	16Ø4	ØØ	Ø31656
52	F8ØME	21Ø4	ØØ	Ø36ØØ4
55	GETEXT	Ø164	ØØ	Ø3737Ø
6Ø	HSPTES	2214	ØØ	Ø4Ø615
63	INDD39	212Ø	ØØ	Ø41663
66	LISTP	27ØØ	ØØ	Ø43114
71	MFINIT	233Ø	ØØ	Ø43674
74	MULTIPLY	64	ØØ	Ø456Ø4
77	OTB	Ø15Ø	ØØ	Ø46Ø71
Ø1Ø2	PATCH	Ø64Ø	ØØ	Ø514Ø
Ø1Ø5	SCRUB	Ø17Ø	ØØ	Ø52212
Ø11Ø	STARTØ	Ø534	ØØ	Ø53Ø14
Ø113	START3	Ø66Ø	ØØ	Ø53415
Ø116	STOP	Ø26Ø	ØØ	Ø54Ø22
Ø121	TCOM	Ø23Ø	ØØ	Ø55262
Ø124	TRACE	133Ø	ØØ	Ø5612Ø
Ø127	UBTAPE	15ØØ	ØØ	Ø56535
Ø132	WATCH	Ø234	ØØ	Ø57274
Ø135	XDISK	Ø754	ØØ	Ø57755
Ø14Ø	XDK	1Ø2Ø	ØØ	Ø61557
Ø143	XPR	Ø52Ø	ØØ	Ø62466
Ø146	XPRNTB	Ø65Ø	ØØ	Ø6325Ø5
Ø151	XTAPEB	132Ø	ØØ	Ø63712
Ø154	ACOM	21ØØ	ØØ	Ø64511
Ø157	D3ERR	255Ø	ØØ	Ø653Ø3
Ø162	D3MSG	Ø46Ø	ØØ	Ø66753
Ø165	D3SLR	1144	ØØ	Ø67514
Ø17Ø	D4ERR	254Ø	ØØ	Ø7Ø7Ø4
Ø173	D4MSG	Ø514	ØØ	Ø72424
Ø176	D4RLR	Ø654	ØØ	Ø73Ø36
Ø2Ø1	D4STAT	Ø234	ØØ	Ø73766
Ø2Ø4	ERRECK	317Ø	ØØ	Ø75516
Ø2Ø7	MEMIO	2Ø64	ØØ	Ø77257
Ø212	XDKERR	2Ø7Ø	ØØ	Ø6Ø711
Ø215	BMXCON	324Ø	ØØ	1ØØ553
Ø22Ø	BMXOPE	Ø51Ø	ØØ	1Ø3Ø4Ø
Ø223	BCOM	173Ø	ØØ	1Ø4Ø26
Ø226	CONMAN	112Ø	ØØ	1Ø5333
Ø231	TAPEIO	4ØØØ	ØØ	1Ø6176
Ø234	TAPFUN	Ø4Ø4	ØØ	1Ø7534
Ø237	TDEMØ	Ø5Ø4	ØØ	111Ø5Ø
Ø242	TEX	146Ø	ØØ	112ØØØ
Ø245	TRCLN	Ø23Ø	ØØ	113ØØ3
Ø25Ø	TREQC	Ø214	ØØ	113536
Ø253	TRORN	Ø554	ØØ	113746
Ø256	TRREDO	Ø754	ØØ	114433
Ø261	TRWRT	Ø7Ø4	ØØ	115356
Ø264	SDMP1	1614	ØØ	117167

No.	Ovl Name	Size	BM-Upr	BM-Lwr
Ø1	SDMPB	Ø46Ø	Ø1	Ø21615
Ø4	SDMPE	Ø154	Ø1	Ø22Ø73
Ø7	AMSG	174Ø	ØØ	Ø2Ø772
12	BMOL	461Ø	ØØ	Ø33652
15	BTH	Ø374	ØØ	Ø2236Ø
2Ø	CALL	2Ø1Ø	ØØ	Ø22614
23	CLKSNC	1Ø1Ø	ØØ	Ø24146
26	CPOL	2474	ØØ	Ø35265
31	CRAY	133Ø	ØØ	Ø2564Ø
34	DIVIDE	Ø444	ØØ	Ø26765
37	DKLOOK	Ø1ØØ	ØØ	Ø3Ø12Ø
42	DM3	Ø23Ø	ØØ	Ø3Ø274
45	DTB	Ø24Ø	ØØ	Ø31416
5Ø	ERRDMP	1514	ØØ	Ø32217
53	FDMPDR	Ø52Ø	ØØ	Ø37165
56	HDRPAG	1Ø74	ØØ	Ø37425
61	INFORM	Ø2ØØ	ØØ	Ø4126Ø
64	INDD49	2164	ØØ	Ø423Ø7
67	LPT	136Ø	ØØ	Ø36671
72	MOSTES	1524	ØØ	Ø44362
75	NOBEAT	Ø4ØØ	ØØ	Ø45621
Ø1ØØ	OUTCALL	Ø124	ØØ	Ø46123
Ø1Ø3	PLOTIT	Ø6Ø4	ØØ	Ø5157Ø
Ø1Ø6	SETIME	1474	ØØ	Ø5225Ø
Ø111	START1	Ø314	ØØ	Ø53143
Ø114	START4	Ø33Ø	ØØ	Ø53571
Ø117	SYSS	236Ø	ØØ	Ø54Ø76
Ø122	TDUMP	1514	ØØ	Ø5533Ø
Ø125	TRACK	Ø164	ØØ	Ø564Ø6
Ø13Ø	UNBLK	Ø73Ø	ØØ	Ø57Ø55
Ø133	XCLOCK	172Ø	ØØ	Ø57343
Ø136	XDISKA	1454	ØØ	Ø6Ø15Ø
Ø141	XMT	122Ø	ØØ	Ø61763
Ø144	XPRINT	Ø774	ØØ	Ø62612
Ø147	XTAPE	Ø75Ø	ØØ	Ø63357
Ø152	XTAPEC	Ø61Ø	ØØ	Ø64176
Ø155	BADDAT	Ø26Ø	ØØ	Ø65131
Ø16Ø	D3IOR	2454	ØØ	Ø66Ø35
Ø163	D3RLR	Ø7ØØ	ØØ	Ø67Ø67
Ø166	D4DEM	337Ø	ØØ	Ø67745
Ø171	D4IOR	275Ø	ØØ	Ø71434
Ø174	D4OVR	Ø61Ø	ØØ	Ø72547
Ø177	D4SKR	1424	ØØ	Ø73211
Ø2Ø2	DD49	342Ø	ØØ	Ø74Ø35
Ø2Ø5	FIRECODE	Ø414	ØØ	Ø76354
Ø21Ø	REPORT	Ø7ØØ	ØØ	Ø77674
Ø213	XDKFMT	114Ø	ØØ	Ø61327
Ø216	BMXCPU	2674	ØØ	1Ø1423
Ø221	BMXSIO	155Ø	ØØ	1Ø3162
Ø224	BUFMAN	214Ø	ØØ	1Ø4414
Ø227	DSCGET	117Ø	ØØ	1Ø5557
Ø232	TAPEND	152Ø	ØØ	1Ø7176
Ø235	TAPMOV	1174	ØØ	1Ø7635
Ø24Ø	TDEM1	246Ø	ØØ	111171
Ø243	TRBOC	Ø45Ø	ØØ	112514
Ø246	TRDCK	137Ø	ØØ	113Ø51
Ø251	TRIDB	Ø31Ø	ØØ	113573
Ø254	TRRDB	132Ø	ØØ	114101
Ø257	TRSET	1Ø24	ØØ	114626
Ø262	RSTRT	2154	ØØ	115537
Ø265	SDMP2	175Ø	ØØ	117532

No.	Ovl Name	Size	BM-Upr	BM-Lwr
Ø2	SDMPC	Ø35Ø	Ø1	Ø21731
Ø5	MFCHK	1524	Ø1	Ø22126
1Ø	BEGIN	176Ø	ØØ	Ø21362
13	BMOLP	1244	ØØ	Ø35Ø14
16	BTO	Ø224	ØØ	Ø22457
21	CDEM	2544	ØØ	Ø23216
24	CLOCK	1114	ØØ	Ø2435Ø
27	CPSPIN	Ø3ØØ	ØØ	Ø25Ø62
32	CRTDEM	215Ø	ØØ	Ø26126
35	DKDMP	346Ø	ØØ	Ø27Ø76
4Ø	DKSET	Ø3ØØ	ØØ	Ø3Ø14Ø
43	DOM	15Ø4	ØØ	Ø3Ø342
46	ECHOCP	Ø74Ø	ØØ	Ø31466
51	F8ØM	444Ø	ØØ	Ø32542
54	FIND	Ø274	ØØ	Ø37311
57	HPDATA	3644	ØØ	Ø37644
62	INDD29	1614	ØØ	Ø4132Ø
65	LISTO	Ø64Ø	ØØ	Ø42744
7Ø	MAGR	122Ø	ØØ	Ø36425
73	MSGHND	3364	ØØ	Ø447Ø7
76	OBIT	Ø64Ø	ØØ	Ø45721
Ø1Ø1	OVLNUM	Ø1524Ø	ØØ	Ø4615Ø
Ø1Ø4	PRTAPE	13Ø4	ØØ	Ø51731
Ø1Ø7	START	1124	ØØ	Ø52567
Ø112	START2	Ø734	ØØ	Ø53226
Ø115	STATS	Ø614	ØØ	Ø53657
Ø12Ø	SYSTEXT	234Ø	ØØ	Ø54572
Ø123	TIME	1224	ØØ	Ø55653
Ø126	TSTASH	Ø35Ø	ØØ	Ø56443
Ø131	USURP	Ø144	ØØ	Ø57243
Ø134	XCLS	Ø13Ø	ØØ	Ø57727
Ø137	XDISKB	113Ø	ØØ	Ø6Ø463
Ø142	XOPN	1174	ØØ	Ø62227
Ø145	XPRNTA	Ø76Ø	ØØ	Ø63Ø11
Ø15Ø	XTAPEA	Ø6Ø4	ØØ	Ø63551
Ø153	XTAPED	Ø644	ØØ	Ø6434Ø
Ø156	D3ECC	Ø37Ø	ØØ	Ø652Ø5
Ø161	D3LOG	1Ø14	ØØ	Ø6655Ø
Ø164	D3SKR	1224	ØØ	Ø67247
Ø167	D4ECC	Ø2Ø4	ØØ	Ø7Ø643
Ø172	D4LOG	Ø77Ø	ØØ	Ø72226
Ø175	D4RES	Ø524	ØØ	Ø72711
Ø2ØØ	D4SLR	124Ø	ØØ	Ø73516
Ø2Ø3	DISK	2664	ØØ	Ø74741
Ø2Ø6	ICOM	3ØØØ	ØØ	Ø76457
Ø211	TRANSFR	1454	ØØ	1ØØ54
Ø214	BMXAIO	Ø72Ø	ØØ	1ØØ367
Ø217	BMXDEM	317Ø	ØØ	1Ø22Ø2
Ø222	BMXTPO	145Ø	ØØ	1Ø3514
Ø225	BYPASS	1334	ØØ	1Ø5Ø44
Ø23Ø	TAPDIS	Ø7Ø4	ØØ	1Ø6Ø15
Ø233	TAPERR	5Ø	ØØ	1Ø7522
Ø236	TDEM	366Ø	ØØ	11ØØ74
Ø241	TERROR	1354	ØØ	111705
Ø244	TRCER	Ø664	ØØ	112626
Ø247	TRDCKØ	Ø7Ø4	ØØ	113347
Ø252	TRINR	Ø344	ØØ	113655
Ø255	TRRDF	Ø23Ø	ØØ	114365
Ø26Ø	TRTELL	1514	ØØ	115033
Ø263	SDMPØ	3764	ØØ	116172
Ø266	SDMP3	Ø22Ø	ØØ	12Ø124

0267	SDMP3A	0550	00	120170
0270	SDMP4	0350	00	120322
0271	SDMP5	0210	00	120414
0272	SDMP6	0234	00	120456
0273	SDMP7	0250	00	120525
0274	SDMP8	0610	00	120577
0275	SDMP9	0454	00	120741
0276	SDMP10	0144	00	121054
0277	CLEAR	0520	00	121105
0300	CLEAR0	0154	00	121231
0301	COPY	1040	00	121264
0302	COPY0	1074	00	121474
0303	COPY1	0670	00	121713
0304	COPY2	0744	00	122071
0305	COPY3	0604	00	122262
0306	COPY4	0474	00	122423
0307	COPY5	0550	00	122542
0310	DDUMP	0610	00	122674
0311	DDUMP0	0374	00	123036
0312	DEF	0650	00	123135
0313	DELET0	0414	00	123307
0314	DELETE	0650	00	123412
0315	DLOAD	0560	00	123564
0316	DLOAD0	1130	00	123720
0317	DSTAT	0514	00	124146
0320	DSTAT0	0504	00	124271
0321	EDDELE	0104	00	124412
0322	EDINST	0544	00	124433
0323	EDIT	0644	00	124564
0324	EDIT0	1460	00	124735
0325	EDIT1	1174	00	125251
0326	EDPRNT	0534	00	125510
0327	EDREPL	0134	00	125637
0330	EDTYPE	0234	00	125666
0331	FDUMP	0770	00	125735
0332	FDUMP0	1040	00	126133
0333	FLAW	0630	00	126343
0334	FLOAD	0714	00	126620
0335	FLOAD0	1300	00	127003
0336	FSTAT	0650	00	127263
0337	FSTAT0	0550	00	127435
0340	INIT	0630	00	127567
0341	INIT0	0774	00	127735
0342	LINGET	0350	00	130134
0343	LINPUT	0664	00	130226
0344	PROC	0564	00	130403
0345	PROC0	0554	00	130540
0346	RENAME	0600	00	130673
0347	RENAM0	0430	00	131033
0350	XFMACC	0244	00	131141
0351	XFMCLS	0240	00	131212
0352	XFMCRE	0710	00	131262
0353	XFMDEL	0174	00	131444
0354	XFMDIR	0350	00	131503
0355	XFMDST	0634	00	131575
0356	XFMFIL	0324	00	131744
0357	XFMFST	0144	00	132031
0360	XFMFLW	0244	00	132062
0361	XFMFND	0404	00	132133
0362	XFMGET	0504	00	132234
0363	XFMIO	1020	00	132355
0364	XFMNIT	1054	00	132561
0365	XFMPUT	0414	00	132774
0366	XFMSTR	0200	00	133077
0367	FORMAT	0434	00	126511
0370	CONC	1374	00	133137
0371	CONCERR	1134	00	133436
0372	CONCID	1214	00	133665
0373	CONCIO	3144	00	134130
0374	ENDCONC	0500	00	134761
0375	ENTRID	0130	00	135101
0376	REMVID	0100	00	135127
0377	ACQTRM	60	00	135147
0400	ACQUIRE	0404	00	135163
0401	AMPEX	0600	00	135264
0402	BABEL	1634	00	135424
0403	BARDAT	0420	00	135773
0404	BMAGET	0404	00	136077
0405	BMGET	0374	00	136200
0406	BXDIS	0544	00	136277
0407	CFINIT	1020	00	136430
0410	CFREAD	0560	00	136634
0411	CLI	1720	00	136770
0412	CLINIT	1414	00	137354
0413	CLUSTR	1000	00	137657
0414	CMGET	0220	00	140057
0415	CNCDIS	2020	00	140123
0416	COMBO	0134	00	140527
0417	COMM01	3744	00	140556
0420	COMM02	1430	00	141547
0421	COMM03	0710	00	142055
0422	COMM04	1074	00	142237
0423	COMM05	1210	00	142456
0424	COMM06	0604	00	142720
0425	COMM07	0710	00	143061
0426	COMM08	1164	00	143243
0427	COMM09	1604	00	143500
0430	COMM10	1464	00	144041
0431	COMM11	1044	00	144356
0432	COMM12	0760	00	144567
0433	COMM13	1764	00	144763
0434	COMM14	1310	00	145360
0435	COMM15	1104	00	145642
0436	COMM16	0730	00	146063
0437	CONSL	1250	00	146251
0440	CPUGET	0350	00	146523
0441	CRAYIO	1100	00	146615
0442	DBGET	0370	00	147035
0443	DECOD2	0660	00	147133
0444	DECODE	1630	00	147307
0445	DELMSG	1200	00	147655
0446	DESCRIBE	0204	00	150115
0447	DEVDAT	0560	00	150156
0450	DISP01	3714	00	150312
0451	DISP02	1220	00	151275
0452	DISPLAY	1300	00	151541
0453	DKDIS	0420	00	152021
0454	DKSTAT	2560	00	152125
0455	ERRDIS	1424	00	152661
0456	ERROR	0234	00	153166
0457	F80	5754	00	153235
0460	F132	5754	00	154630
0461	FMGET	0430	00	156223
0462	GRAPH	1230	00	156331
0463	GRCDIS	2074	00	156577
0464	GRPDIS	1220	00	157216
0465	GRUDIS	2050	00	157462
0466	HELP	0474	00	160074
0467	HELP2	1370	00	160213
0470	HELPD	020544	00	160511
0471	HSPGET	0400	00	164642
0472	ICONSL	0330	00	164742
0473	IDEBUG	1220	00	165030
0474	IDLGET	0450	00	165274
0475	IDRCT	1660	00	165406
0476	IFRMT	0530	00	165762
0477	KEYBD	0514	00	166110
0500	LCP	1144	00	166233
0501	LINK	1254	00	166464
0502	LOGON	1400	00	166737
0503	MESSAGE	0254	00	167237
0504	MSTAT	1544	00	167312
0505	MSTDIS	1540	00	167643
0506	NEWDIS	0260	00	170173
0507	NSCNET	1550	00	170247
0510	NSCSFP	2220	00	170601
0511	OFRMT	0770	00	171245
0512	ONLINE	0440	00	171443
0513	POST	0514	00	171553
0514	PROTINIT	1254	00	171676
0515	PROTOCOL	1510	00	172151
0516	QUEUE	0340	00	172473
0517	READ	1414	00	172563
0520	SNAP	0474	00	173066
0521	SOROC	0440	00	173205
0522	SSDGRA	1070	00	173315
0523	STADIS	2140	00	173533
0524	STAGEIN	1414	00	174163
0525	STAGEOUT	1504	00	174466
0526	STATCL	1374	00	175007
0527	STATINIT	0540	00	175306
0530	STATION	64	00	175436
0531	STIO	0350	00	175453
0532	STMSG	2044	00	175545
0533	STPLOT	0604	00	176156
0534	STREAMS	1114	00	176317
0535	STRSTAT	1460	00	176542
0536	STSGET	0350	00	177056
0537	STTAPI	0654	00	177150
0540	STTAPO	0570	01	177323
0541	STUBPR	1760	00	177461
0542	STXDKI	1044	01	000055
0543	STXDKO	0720	01	000266
0544	SUMDAY	2740	01	000452
0545	SUMHOW	0664	01	001242
0546	SUMTIM	0674	01	001417
0547	SYNTAX	2654	01	001576
0550	SYSTAT	1754	01	002351
0551	TAPEC	1650	01	002744
0552	TEC455	0300	01	003316
0553	TJOB	1674	01	003376
0554	TKSTAT	2110	01	003755
0555	TURNPG	0774	01	004377
0556	UPDATE	0430	01	004576
0557	XFRMT	1500	01	004704
0560	XJSTAT	2410	01	005224
0561	XMPXP	2260	01	005726
0562	XMP4XP	2754	01	006402
0563	IACMD	1750	01	007175
0564	IACON	0760	01	007567
0565	IACON1	0404	01	007763
0566	IAFUNC	1444	01	010[illegible]

Z	SCP	STG	DQM	PDM	JCM	JSH	DQM	DEC	EXP	MEP	MSG	SPM	IQM	FVD
AUT	-AUT	PDD	-DAT	DAT	-CSD	CSD	(DDL)	EQT	CNT	-AEM	AUT	CSD	ILT	EQT
CNT	-IBT	SDT	DCT	CSD	SDT	-JXT	-(DNT)		-DEX		(DSP)	DCT	VPT	DRT
DAT	-LCT	-SST	(DNT)	(DNT)		-MST	-(DSP)		(DNT)		JTA	+MCT	VCT	
(DNT)	-LIT		-DRT	DRT		-RJI	JXT		(DSP)		JXT	+ICT	TRB	
DRT	-LXT		(DSP)	-DSC		SDT	-LFT		-DUX		-LGJ	+STT	IST	
-DSC	PDD		-EQT	(DSP)		-TXT	-ODN		-FSH		PDD			
(DSP)	-SDT		GRT	DXT		JTA	(PDD)		GRT		SDT			
-DVL	-STT		JXT	EQT			SDT		JXT					
DXT			-RQT	JCB			-SWT		-LDT					
-EFT			JTA	JTA			TCB		-SM					
-EQT			PHR	JXT			-UPT		-VAX					
GRT			SQP	PDD			-JTA		-VUX					
JTA			TXT	-PDI			-TXT		JTA					
JXT				-PDS			IDD							
(ODT)				-QDT			IRT							
PDI				SDT			TRB							
QDT				-XAT										
-RJI														
SDT														
TCB														
TDT														
TXT														

PACKET DRIVER	FED DRIVER	DISK/SSD DRIVER
+SCT	+CHT	DCT
+PIQ	+CXT	EQT
+POQ	LIT	+CHT
+CHT	LXT	+CAT
+CAT	LCT	+CLT
+CLT	+CAT	+CIT
+CIT	+CLT	+CBT
CXT	+CIT	+PHR
		+SAC
		+SQP

USER		CSP
(BGN)	(LDT)	(DSP)
(BAT)	(OPT)	(JCB)
(DDL)	(JBI)	(LFT)
(DSP)	(DRPB)	(DNT)
(DNT)	(RCB)	
(JAC)	(NCB)	
(JCB)	(MHB)	
(LFT)	(IJPB)	
(ODN)		
(PDD)		
(JST)		

The EXEC table area contains all EXEC tables, alphabetically ordered. Most table layouts are described in the COS Table Descriptions Internal Reference Manual, publication SM-0045. The other tables are internally documented. The tables are:

CAT Channel Address Table

CBT Channel Buffer Table containing one entry of working storage for each disk driver channel.

CHT Channel Table containing a 1-word entry for each side (input and output) of a physical channel. An entry contains a pointer to the Channel Processor Table for the channel-assigned task ID and the address of the channel processor assigned to the side of the channel. Input sides are assigned even numbers; output sides odd numbers.

CLT Channel Limit Table

CXT Channel Extension Table

ICT Interrupt Count Table

IHT Interrupt Handler Table

MCT Monitor Count Table

MEL- Memory Error Log Table

MRT Monitor Request Table

PCT Packet Counter Table

PIQ Packet Input Queue

POQ Packet Ouput Queue

PRT Packet Routing Table

PWS Processor working storage

RMS Read Margin Select Table

SCT Subsystem Control Table

STT System Task Table consisting of three parts: a header, a task parameter word area, and an Exchange Package area

STX System Task Exchange Package Table

TBT Task Breakpoint Table

TET Time Event Table

XFT History Function Table

XTT History Trace Table

This area contains tables accessible to all STP tasks (not necessarily in the order noted).

AUT Active User Table containing an entry for each logged on interactive user

CMOD Communications modules in 6-word groups that form a pool from which they are allocated as needed. Two words are used as control; two are used as input registers; and two are used as output registers. A task receives all of its requests and makes all of its replies through a CMOD.

CNT Configuration Table containing information on the availability and type of each device known to the system. The CNT is used only for tape devices.

CPT Class Parameter Table used by JCM. It contains all job statement parameters used to determine the job class.

CSD Class Structure Definition Table containing the job class structure. For each class defined in the structure, there is a class map; these appear in CSD in descending order. A header precedes the class maps. Variable length characteristic expressions for each class follow the maps.

DAT Dataset Allocation Table. A DAT exists for each dataset being used by the system and defines where the dataset logically resides on mass storage, that is, on which logical devices and what portion of a device.

DCT Device Channel Table serving as a link between a physical or logical disk channel and the EQT. It is an interface to the EXEC disk driver. The DCT holds channel system performance data.

DRT Device Reservation Table. A DRT exists for each logical disk device known to the system. A DRT contains a bit map showing available and reserved tracks on the device.

DXI Permanent Dataset Catalog Extension Information Table containing information used by the Permanent Dataset Manager (PDM) such as the size of the Dataset Catalog Extension Table (DXT)

ECT Error Code Table for controlling abort and reprieve processing done by EXP. It contains a 1-word entry for each system error code and is defined using the ERDEF macro.

EPQ EXEC Packet Queue containing APT entries destined for EXEC. Defined by COMQP (see COMQP in the COS Table Descriptions Internal Reference Manual, publication SM-0045).

EQT Equipment Table containing an entry for each disk device known to the system

GRT Generic Resource Table containing an entry for each generic resource in the system.

IBT Interactive Buffer Table for managing the Interactive Buffer Pool

ILT ISP Link Table. The ILT contains an entry for each input or output channel using ISP protocol. The ILT is used by STP task IQM and its associated common routine ISPCOM.

IPT Interjob Communication Path Table

IST ISPMAIN Status Table. The IST contains an entry for each ISP system that COS allows to log on. It is used by STP tasks IQM and EXP.

JSQBM Job Sequence Number Bit Map. The JSQBM indicates which job sequence numbers are being used.

JXT Job Execution Table. The JXT controls all active jobs in the system and can contain as many as 256 entries. Entry 0 (the first entry) is used to represent the system itself.

LCT Link Configuration Table containing an entry for each CPU channel used for front-end communications

LIT Link Interface Table. SCP assigns an LIT entry at startup to each CPU channel used for front-end communications. This table is used primarily for channel control.

LXT Link Interface Extension Table. EXEC assigns an LXT entry for a front-end station at log-on time and releases the entry at log off. This table is used primarily for EXEC-STP communication of information on a front-end station.

MST Memory Segment Table containing an entry for each segment of memory allocated by the Job Scheduler (JSH) as well as an entry for each free segment. The number of entries in the MST is set to twice the number of JXT entries plus four words. Each MST entry is one word in length.

PDI Permanent Dataset Information Table containing information used by the Permanent Dataset Manager (PDM), such as the number of overflow and hash pages.

PDS Permanent Dataset Table consisting of a 1-word header followed by a 1-word entry for each active permanent dataset. The entry indicates how a dataset is accessed and if multiple access exists. If so, the entry tells how many users are accessing the dataset.

PHR Physical Request Table. DQM uses the PHR to pass data transfer requests to the SSD driver.

PXT Processor Execution Table contains status information for each physical processor, including which user task is currently connected.

QDT Queued Dataset Table describing the multitype attributes for a disposed dataset. The table is managed by the Permanent Dataset Manager (PDM) and Exchange Processor (EXP) tasks. The number of entries in the QDT must equal the SDT entry count.

RIT Registered ID Table for Interjob Communication.

RJI Rolled Job Index Table containing for each defined JXT, an entry describing the job assigned to the JXT entry, allowing the recovery of jobs from mass storage.

RQT Request Table used to queue transfer requests for disk management. DQM uses the RQT to manage both logical and physical disk requests. RQT entries are queued to an EQT entry.

SAC SSD Active Channel Table. The SSD driver uses the SAC to control SSD channel activity.

SBU System Billing Unit Table containing the values obtained when system billing units are calculated for system resources.

SDR System Directory containing a Dataset Name Table for each of the datasets comprising the system library. The SDR is initialized after a system startup.

SDT System Dataset Table containing an entry for each dataset spooled to or from a front-end system. An SDT entry can have appendages allocated out of an STP memory pool to contain TEXT field and station slot information.

SQP SSD Queued Packet Table. Used to submit SSD data transfer requests in the form of A-packets to the SSD driver. The queue is circular, with the driver removing requests at OUT, and DQM adding requests at IN.

SST Stager Stream Table. Eight input stream and eight output stream SSTs are contained within each LXT.

STPD STP Dump Directory containing pointers to task origins, buffers, and so on. An entry gives a mnemonic in ASCII plus the relative STP address for the area.

SXT System Execution Table containing pointers to execution profile information for system tasks.

TDT Tape Device Table. The Tape Queue Manager task uses the Tape Device Table to control online tape devices. The TDT contains an entry for each tape device in the system.

TPQ Task Packet Queue containing pointers to each task's QPT entry, which contains APT entries destined for the STP tasks.

TXT Task Execution Table contains all information to control all user tasks within the system.

UCT User Call Table containing a count of the number of times each type of user call is made. This table is used by the System Performance Monitor (SPM).

UDT User Driver Channel Table. The UDT contains an entry for each input or output channel eligible for use by user jobs, subsystem jobs, or the STP task IQM.

VPT ISP Virtual Circuit Pointer Table. The VPT contains one entry for each available ISP connection. Each ISP dataset uses one entry.

Details of the STP tables are given in the COS Table Descriptions Internal Reference Manual, publication SM-0045.

Tables that may reside in the user field include the following:

BAT Binary Audit Table. This table contains an entry for each permanent dataset that meets requirements specified on the AUDIT control statement.

DDL Dataset Definition List. A DDL in the user field accompanies each request to create a DNT.

DSP Dataset Parameter Area. A DSP in the user field contains the status of a particular dataset and the location of the I/O buffer for the dataset.

JAC Job Accounting Table. This table defines an area for data to be returned to the user by an accounting request.

JCB Job Communication Block, residing at the very beginning of the user field and containing information used by both COS and library routines. Copies of the more important pointers are kept in the job's JTA to assist in JCB validation and re-creation.

LFT Logical File Table. This table in the user field contains an entry for each dataset name and alias referenced by FORTRAN users. Each entry points to the DSP for a dataset.

ODN Open Dataset Name Table. A request to open a dataset for a job contains a pointer to the ODN table in the user field.

PDD Permanent Dataset Definition Table. A PDD in CSP is used for many permanent dataset requests.

See the COS Table Descriptions Internal Reference Manual, publication SM-0045, for detailed descriptions of these tables. This table is available as a listable tape.

STP COMMON ROUTINES

MODULE	ENTRY POINTS	PURPOSE
ERROR	ERROR0, ERROR1	hang the system
REQRPLY	TSKREQ, PUTREQ, GETREQ, PUTREPLY, REPLIES, GETREPLY	task-to-task communications
STPMEM	MEMAL, MEMDE, PMEMDE, SSLDE	memory pool management
CHAINS	CHAIN, CHAINF, UNCHAIN, JCHAIN, JCHAINF, JUNCHAIN	chain management
STPTIME	RQST2, RT2JD, JD2RT	date/time calculations
QUEUES	DQSD2, EQSD2	SDT queue management
QMSG	NXTMSG, FREEMSG, ENQMSG	interactive station message management
MSGQUE	MSGQUE	SCP/operator message processing

STP COMMON ROUTINES

MODULE	ENTRY POINTS	PURPOSE
STPUTIL	BTO, $OTB, $DTB, SFN, $NOCV	utility routines
STPDATS	GETDAT, RELDAT	DAT management
JMEM	JMEMAL, JMEMDE	JTA memory pool management
JTADNT	GETDNT, GETLFT, RELDNT	JTA DNT management
FIXJXPR	FIXJXO, FIXPRI	job pri. calculations
CRACKER	IND	JOB control stmt. cracker
GETPARM	GETPARM	parameter cracker
CONFIG	CONFIG	configuration changes

Kernel Common Routines

NAME	DESCRIPTION
ERGC	BUILDS A NEW ACTIVITY DESCRIPTOR AND SMOD. PLACES AD ON CP QUEUE.
EREB	FINDS SPACE FOR AD AND CLEARS IT. PLACES AD ON CHAIN OF AD.
EREC	GETS MOS FOR SMOD AND SETS UP PARAMETERS IN AD
ERED	INITIALIZES SMOD IN SCRATCH AREA
STOREGS	STORES SPECIFIED OPERAND REGISTERS IN SMOD
LODREGS	LOADS SPECIFIED OPERAND REGISTERS FROM SMOD
EDQU	REMOVES ACTIVITY FROM SPECIFIED QUEUE
ENQU	PUTS ACTIVITY ON SPECIFIED QUEUE
EQCP	PUTS ACTIVITY ON CP QUEUE AT PRIORITY
EQUE	PUTS ACTIVITY ON SPECIFIED QUEUE AT PRIORITY
EPOQ	PUTS ENTRY ON MESSAGE QUEUE
ETOQ	REMOVE ENTRY FROM MESSAGE QUEUE
QTIME	PUT AN ACTIVITY ON TIMER QUEUE. ACTIVITY MUST ALSO BE ON ANOTHER QUEUE.
DQTIME	REMOVE AN ACTIVITY FROM TIMER QUEUE
DQFIND	LOCATE AND REMOVE AN ENTRY FROM A QUEUE
EMSGIOP	SEND MESSAGE TO ANOTHER IOP VIA ACCUMULATOR CHANNEL

$SYSTEXT MACROS

PDUMP
SETSTKAT
FORCE
NEWENTRY
INSMAC
FTIO
BASERR
CPUTYPE
SVREGS
LDREGS
OPENA
@VALREG
@GENREW
@GENTP
$RCW -
$TSTRG
$BIO
PRM1SYM
PRM2SYM
RPM2MIO
RECUR
SNAPSHOT
%REGADDR
%QDOUBLE
%FORTIO
%ARGADDR
FEDLP

(SYERP ERROR CODES)

DEFLOCK
LOCKON
LOCKOFF

COSTXT MACROS

DRT
EQT
$GTTOKEN
$DOICH
CONFIG
%GETNUM
FLAW
STRTFLW
ENDFLW
SETIF
POST
VTRACE
TSKREQ
RJ
LOCK
UNLOCK
ERRU
ERRAZ
ERRAN
ERRAP
ERRAM
ERRSZ
ERRSN
ERRSP
ERRSM
CAPTION
ENDFIELD
%GETSYM
FIELD@
$LINE
GN
GD
$SREG
$LREG

LOADCL
SAVECL
MSGQ
SMDEF
WAIT$SET

SAPTEXT MACROS

EXEC SPECIFIC MACROS

CLEARIP
COPYXP
X$SIO
GETPW
GETSRO
I$FWB
SETCL
SETIP
STOP
FALLTHRU

$APTEXT MACROS

Exit Stack Macros
- EGET
- EPUT
- EINCR
- EDECR
- EXSGET
- EXSPUT

Execution Control Macros
- $IF
- $UNTIL
- $GOTO
- $PUNTIF

Data Definition Macros
- ADDRESS
- FIELD
- ISFIELD
- TABLE

Data Access Macros
- GET
- LOAD
- PUT
- STORE
- RGET
- RPUT
- RSTORE
- FLDADD
- FLDSUB

Overlay and Register Definition Macros
- OVERLAY
- REGDEFS
- REGISTER

QUIZ 3

ANSWERS

QUIZ 3

ANSWERS

OVERVIEW

Match the following terms and their descriptions.

1. STATION	_7_ An interruptable COS program performing system functions
2. EXEC	_5_ 1 XMP CPU in monitor mode and executing EXEC at a time
3. KERNEL	_4_ Log of operating system and user activities
4. $SYSTEMLOG	_1_ Software that runs on the front end linking COS and the front end operating system
5. SINGLE THREADED	_8_ Priveledged work an overlay asks the IOS monitor to do
6. MULTITASKING	_9_ Priveledged work an STP Task asks the monitor to perform
7. TASK	_10_ IOS program running under Kernel control
8. KERNEL SERVICE REQUEST	_6_ Several CPU's working on the same job
9. EXECUTIVE REQUEST	_3_ Monitor for each IOP
10. OVERLAY	_2_ Monitor for the Mainframe COS

SYSTEM PROGRAMS

Match the following terms and their description

1. STATION CALL PROCESSOR	_6_ Inter IOP Message Handler for DD-19 and DD-29 disk activities
2. $APTEXT	_9_ Definitions for COS tables and constants
3. CONCIO	_5_ Consists of one overlay
4. IOP STATION	_10_ Work for IOP to perform
5. DEMON ACTIVITY	_8_ Processes all User exchanges using SØ as the function code
6. ACOM	_2_ Contains definitions of macros, constants, and tables for IOS
7. COSPL / IOPPL	_4_ Considered a front end to COS
8. EXP	_1_ Communicates between the FE station and COS
9. COSTXT	_7_ Source listings for the operating system
10. ACTIVITY	_3_ Processes FEI channel activities in the IOP

PROGRAM SWITCHING

Match the following terms and their descriptions

1. EXCHANGE PACKAGE	_4_ Defines an Activity in IOS
2. DISK ACTIVITY LINK	_6_ 6 words of information passed between COS and IOS
3. SYSTEM TASK TABLE	_10_ Activates ACOM, BCOM, ICOM giving Buffer Memory address of DAL being passed
4. ACTIVITY DESCRIPTOR	_7_ All SMOD's of an activity
5. STORAGE MODULE	_8_ Central Memory address for all COS Task exchange packages
6. ANY PACKET	_3_ Controls COS STP Tasks
7. SOFTWARE STACK	_9_ Central Memory location for user exchange package
8. STX	_2_ Information passed through buffer memory to another IOP
9. PROCESSOR WORKING STORAGE	_5_ An Overlay program's executing enviroment
10. INTER A-A MESSAGE Interrupt Handler	_1_ A COS mainframe programs executing enviroment

SYSTEM RESOURCES

Match these terms and their description:

1. MASTER DEVICE	_8_ Considered part of the IOP station
2. MIOP	_6_ Logically considered a fast disk storage device
3. BIOP	_2_ Main responsibility is handling the NSC hyperchannel and Front End channels
4. DIOP	_5_ Main Responsibility is to drive IBM type Block mux devices and tape units
5. XIOP	_1_ Contains the Dataset Catalogue
6. SSD	_3_ Contains a High Speed channel for use by MIOP for front end activities STATIO request
7. BUFFER MEMORY	_4_ Main responsibilty is to drive Disk Storage devices
8. EXPANDER CHASSIS	_10_ Memory used by each IOP for monitor and executing IOS activities
9. CENTRAL MEMORY	_7_ Working storage for IOP activities
10. LOCAL MEMORY	_9_ Memory where COS resides

STATION PROTOCOL

Match the following terms and their definitions:

1. LINK CONTROL PACKAGE	_7_ Does the word size,logic level and hardware protocol conversion between the Cray and front end
2. MESSAGE CODE	_6_ Provides longer distance links between the Cray and a front end
3. DATASET HEADER	_10_ Links the Station communication to the COS tasks
4. STREAM CONTROL BYTE	_9_ Field in an LCP containing error messages
5. STATION	_1_ 6 word message header sent between the station and SCP
6. HYPERCHANNEL	_3_ Segment zero of a dataset transfer that precedes the dataset
7. FRONT END INTERFACE	_8_ Data transferred accross the station Cray link with the LCP
8. SEGMENT	_2_ Defines what the segment attached to the LCP contains
9. MESSAGE SUB CODE	_4_ Defines the status of a dataset transfer from the request to the transfer completion
10. STATION CALL PROCESSOR	_5_ Software that runs on the front end under the control of the front end's operating system

INTERTASK COMMUNICATION

Match the following terms and their description:

1. IMT	_7_ Used to make an executive request
2. PUTREQ	_2_ Routine to initiate a syncronous request from another COS Task
3. TASKREQ	_5_ Used by the user to make system action requests of EXP
4. ITCT	_1_ Request/Reply packet between STP tasks
5. SØ & S1	_6_ Registers used for task to task requests/replies
6. S1 & S2	_4_ Trace table for all task to task requests/replies
7. S6 & S7	_3_ Asyncronous Request to another Task

INTER-IOP COMMUNICATION

1. AMSG	_4_ Activates a suspended activity in another IOP
2. POPCELL	_7_ Processes Interprocessor DAL's that are requesting TAPE
3. DISK ACTIVITY LINK	_5_ Kernel service request to activate another overlay
4. AWAKE	_3_ 40 octal parcel message passed between IOP's through Buffer memory
5. CALL	_6_ Processes DAL requests from MIOP for DD-49 activities
6. ICOM	_1_ Processes the ALERT service request creating a POPCELL
7. BCOM	_8_ Processes DD-29 DAL's from MIOP, BIOP or DIOP
8. ACOM	_9_ Points to the Interprocessor DAL in Buffer Memory
9. INTER-IOP A-A MESSAGE	_10_ Kernel Service request used by MIOP to initiate a Buffer Memory to/from Central Memory for the Concentrator
10. STATIO	_2_ QUEUE and Activity descriptor information for the ALERT,AWAKE, ASLEEP, RESPOND mechanisim

COS DUMP ANALYSIS

1. STOP BUFFER	_8_ System utility to format a dump created by SYSDUMP
2. $STOP MACRO	_5_ Allows analyst to turn on and off any type of history trace entry
3. ERRSN MACRO	_7_ History trace for inter-task communication
4. EXCHANGE PACKAGES 100-200	_10_ Location in STP TABLES where the STP hang message is put on an STP hang condition
5. XFT	_9_ Creates a copy of the Cray system memory on the master device scratch buffer for dumps
6. XTT	_3_ Macro executed by a task when a hang condition exists
7. ITCT	_1_ Location to look first in a dump to determine who halted first
8. FDUMP	_2_ Macro executed by EXEC to hang the operating system
9. SYSDUMP	_6_ Table containing History trace events of the system
10. SY00 06	_4_ Executing CPU Exchange packages on a hang

IOS DUMP ANALYSIS

Match the Following terms and their description:

1. HALT MESSAGE	_3_ Operand register containing the address the executing overlay is loaded at
2. $PUNTIF	_5_ Points to the Hang macro address to help locate it in the listing
3. %B	_6_ Operand register containing the overlay number that was executing
4. EXIT STACK	_8_ Area in the Kernel tables where the IOP's A, B, C, E, Exit stack and Busy Done flags are saved on a $PUNTIF
5. CONTENTS OF E-1 IN STACK	_10_ Operand register containing the address the active SMOD is at
6. %OVLNUM	_4_ Return jump history stack after an interrupt or an exit
7. %ACTIVE	_2_ Macro executed by the IOS to halt on an error condition
8. EXSAVE	_7_ Operand register containing the address of the active activity descriptor
9. BUSY and DONE flag set	_1_ Tells you which IOP halted and the punt code attached to the $PUNTIF macro
10. %SMOD	_9_ High Speed Channel Error

IOS DUMP ANALYSIS

Match the following terms and their description

1. HALT MESSAGE	_3_ Operand register containing the address the executing overlay is loaded at
2. $PUNTIF	_5_ Points to the Hang-macro address to help locate it in the listing
3. RB	_6_ Operand register containing the overlay number that was executing
4. EXIT STACK	_8_ Area in the Kernel tables where the IOP's A, B, C, E, Exit stack and Busy Done flags are saved on a $PUNTIF
5. CONTENTS OF E-1 IN STACK	_10_ Operand register containing the address the active SMCD is at
6. $OVLNUM	_4_ Return jump history stack after an interrupt or an exit
7. $ACTIVE	_2_ Macro executed by the IOS to halt on an error condition
8. EXSAVE	_12_ Operand register containing the address of the active activity descriptor
9. BUSY and DONE flag set	_1_ Tells you which IOP halted and the punt code attached to the $PUNTIF macro
10. $SMOD	_9_ High Speed Channel Error

EXERCISE 3
ANSWERS

EXERCISE 3
ANSWERS

Exercise 1 EXEC & KERNEL Monitor Flow

Skill: To have some familiarity with the COS/IOS monitors

Tasks:

COS

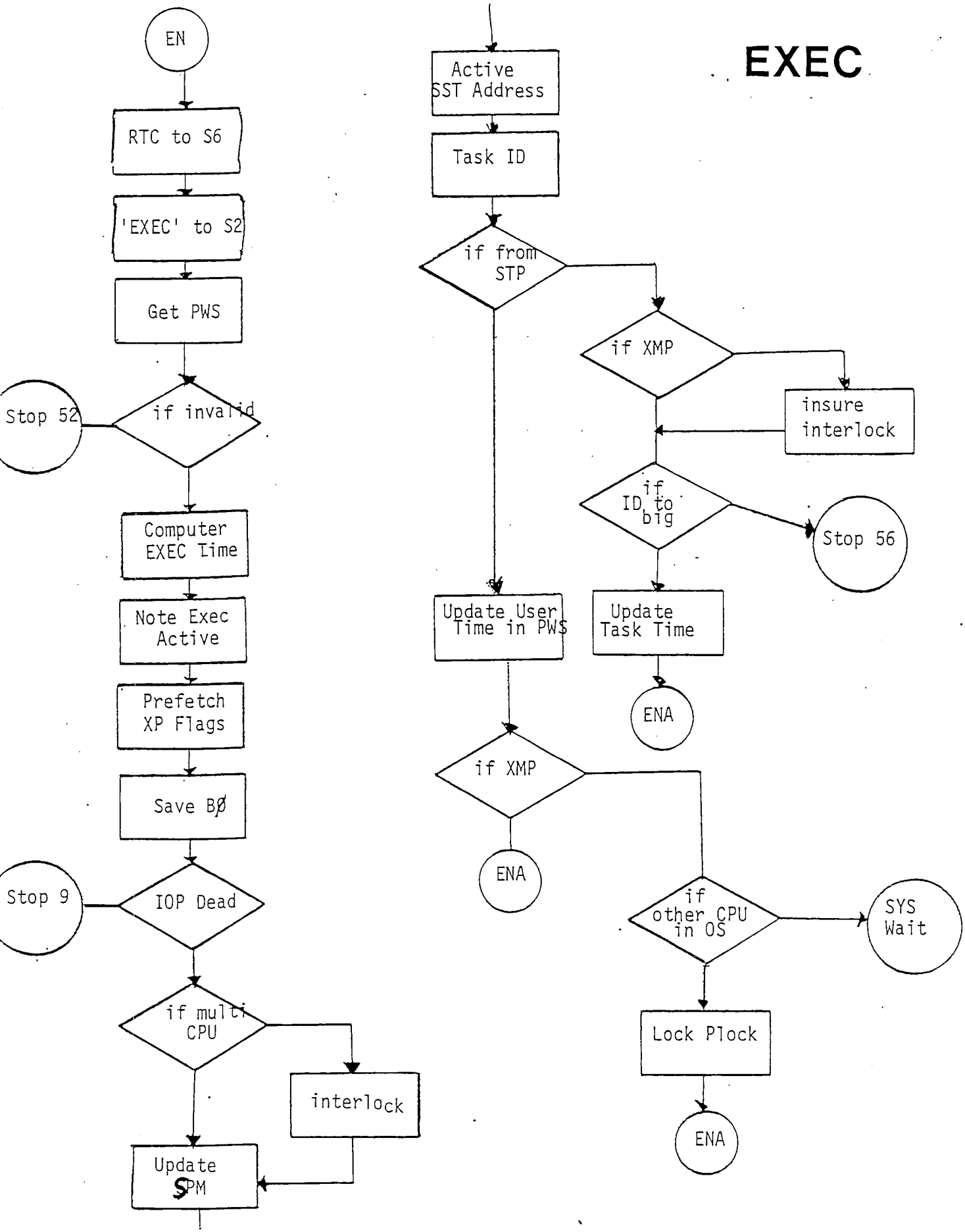
a. Flowchart what occurs in EXEC on an interrupt starting at EN (four pages of code only)

IOS

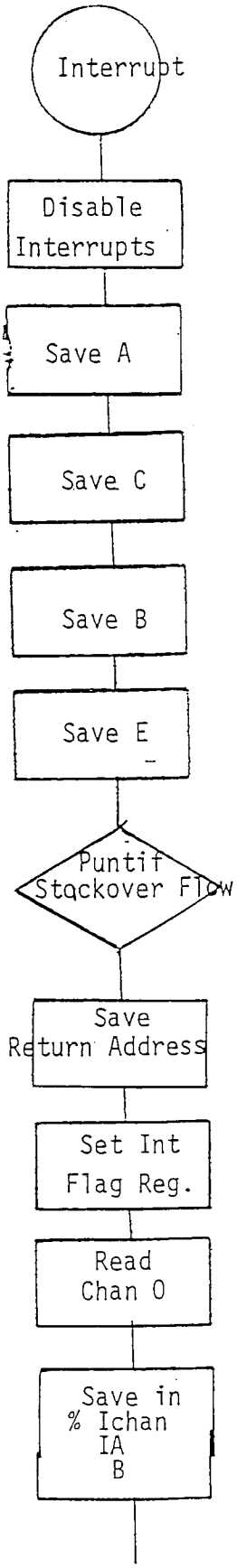
a. Flowchart what occurs in the KERNEL starting at address INTERRUPT (two pages of code only)

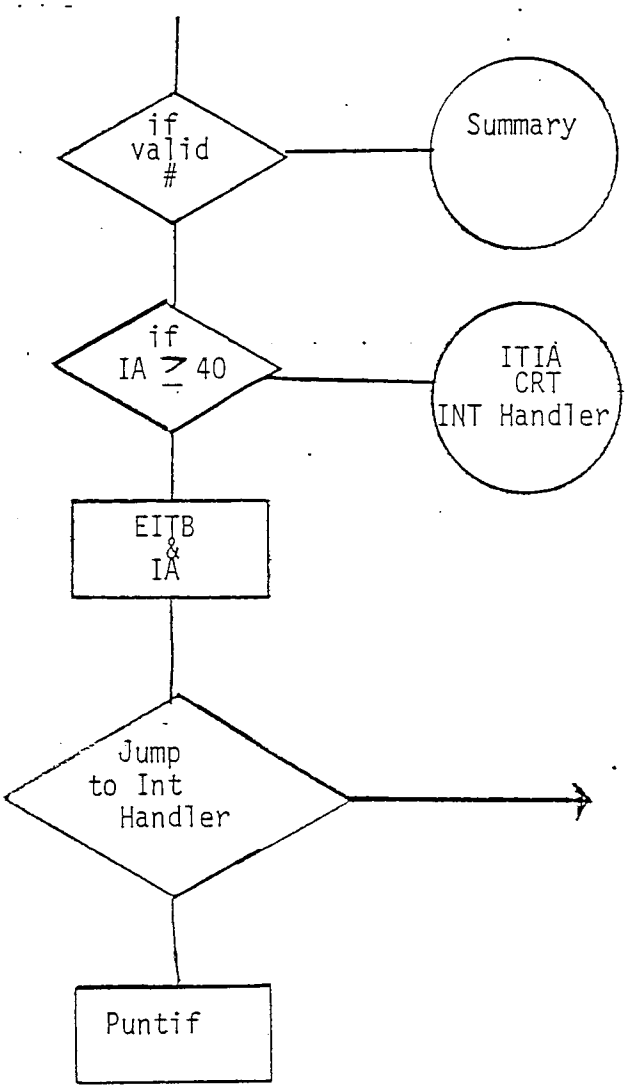
Intended Lesson Results: To be able to go into the monitors and know the flow through the monitor on an interrupt and how to find your way around the monitor listings

EXEC


EN
RTC to S6
'EXEC' to S2
Get PWS
Stop 52
if invalid
Computer EXEC Time
Note Exec Active
Prefetch XP Flags
Save BØ
Stop 9
IOP Dead
if multi CPU
interlock
Update SPM
Active SST Address
Task ID
if from STP
if XMP
insure interlock
if ID to big
Stop 56
Update User Time in PWS
Update Task Time
ENA
if XMP
ENA
if other CPU in OS
SYS Wait
Lock Plock
ENA

KERNEL


Interrupt
Disable Interrupts
Save A
Save C
Save B
Save E
Puntif Stackover Flow
Save Return Address
Set Int Flag Reg.
Read Chan 0
Save in % Ichan IA B


if valid #
Summary
if IA ≥ 40
ITIA CRT INT Handler
EITB & IA
Jump to Int Handler
Puntif

Exercise 2 TASKS AND OVERLAYS

Skill: To have some familiarity with the system programs

COS

a. List the Functions performed by PDM
   1. Save User Input, Output Dataset
   2. Access User, Spooled Dataset
   3. Delete User, Spooled Dataset
   4. DSC Page Request
   5. DXT Page Request
   6. Load User, Input, Output Dataset
   7. Update Active PDS
   8. Permanent Dataset Request
   9. Dump Request
   10. Dequeue System Dataset Table (SDT) entry
   11. Queue SDT entry to Available, Input, Output Queue
   12. Adjust, Modify User Dataset
   13. Rewrite Job Input Dataset SDT entry
   14. Pseudo access for RRJ Rolled Job Recovery in Z
   15 Access for PDSDUMP
   16. Permit Alternate User Dataset Access

b. What table is the PDM function request code passed in
   1. Permanent Dataset Definition PDD

IOS

a. List the Functions performed by ACOM
   1. Initiate Disk Request
   2. Release Disk Activity Link DAL in Originating IOP
   3. Transfer Data Central to Buffer Memory over HISP
   4. Transfer Data Buffer to Central Memory over HISP
   5. Send status to Mainframe task
   6. Central memory to Buffer memory done
   7. Buffer memory to Central memory done

b. When is ACOM used and by who
   1. Used when an A-A message 110000 to 140000 is recieved on an A-A interrupt
   2. Used by the Disk Subsystem for interprocessor DAL's and BIOP BM to CM to BM transfers
   3. Integral part of Disk Subsystem

Intended Lesson Results: To be able to locate a system program listing and determine what the program does and who uses it when.

Exercise 3 **Tasks and Activities**

Tasks: COS Dump 2

a. List all the Tasks in the STT in sequential order
b. Give their priority
c. Give their task ID number
d. Give the task exchange package address

TASK	PRIORITY	ID	XP ADDRESS
1. STP	777	0	4740
2. SCP	1	1	4760
3. EXP	11	2	5000
4. PDM	14	3	5020
5. DEC	20	4	5040
6. DQM	2	5	5060
7. MSG	5	6	5100
8. MEP	10	7	5120
9. SPM	24	10	5140
10. JSH	13	11	5160
11. JCM	12	12	5200
12. TQM	3	13	5220
13. STG	6	14	5240
14. FVD	15	15	5260
15. IQM	4	16	5300

Tasks: IOS Dump 2

a. List all the Activities in the EACT+1 chain in sequential order
b. Give each activities priority
c. Give each activities storage module address

EACT+1 IS LM ADDRESS 5116 AND CONTAINS 73530

AD ADDRESS	NAME	PRI	SMOD ADDRESS
73530	IAOUT	10	64770
73404	KEYBD	10	
103164	IAIOP1	10	
102204	NSCNCIO	10	
101670	NSCNCIO	10	
101440	READ	6	
101414	KEYBD	3	
73274	TSTASH	7	
72440	PROTOCOL	5	
73554	CLI	6	
73320	NSCIO	1	
73360	CONCIO	10	
73204	CONCIO	10	
73024	SCRUB	17	
72764	ICOM	Ø	73010
72724	BCOM	Ø	72750
72664	CDEM	Ø	72710
72624	CRTDEM	7	72650
72564	CLOCK	7	72610
72524	AMSG	Ø	72550
72464	ACOM	Ø	72510

Intended Lesson Results: To be able to identify the tasks and activites in the system and at what address their executing enviroments are

Exercise 4 System Memory Maps

Skill: To be able to find the address that programs are executing at

COS Dmp 2

a. At what address do the EXEC tables end at 15650

b. At what address does STP begin at 32000

c. At what STP relative addresses are the tasks loaded at

TASK	STP RELATIVE ADDRESS
STP	274446
SCP	274713
EXP	313316
PDM	454012
DEC	453741
DQM	446170
MSG	427424
MEP	467066
SPM	467373
JSH	431067
JCM	442536
TQM	354202
STG	467761
FVD	471611
IQM	472511

d. At what address does STP end 535472

IOS Dmp 2

a. At what address do the Kernel tables ends at 12360

b. At what address does the Overlay memory chain begin at 31300

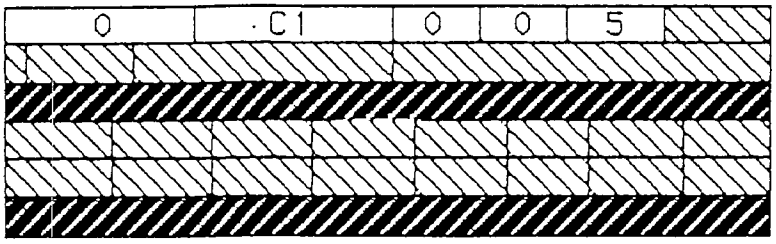
c. At what address does the DAL chain begin at (5162) is 67540

d. At what address does free memory chain begin at (5157) is 73174

e. At what address does IO buffer chain begin at (5165) is 154000

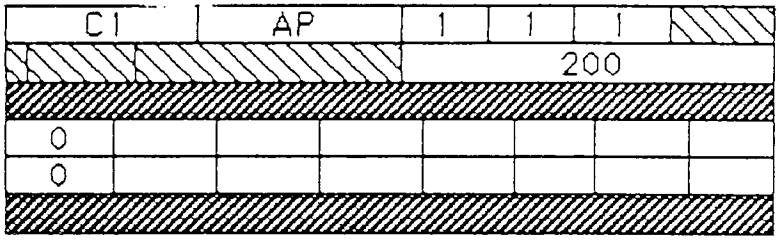
Intended Lesson Results: To be able to determine where to find addresses pointers for relative programs and have some perspective on how Cray system memories are utilized.

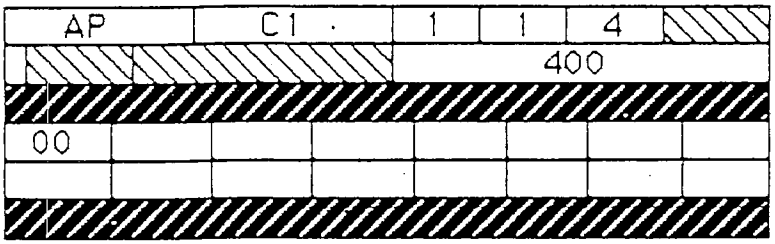
LCP SEQUENCE



RESTART ➡ FE

CRAY ⬅ LOGON



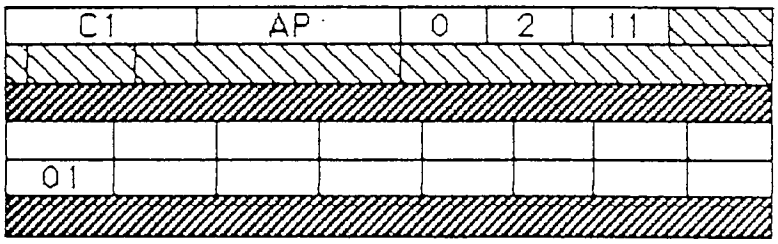


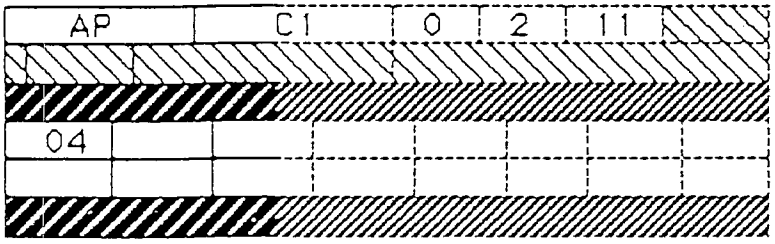
START ➡ FE

IDLE

CRAY ⬅ CONTROL

RTS



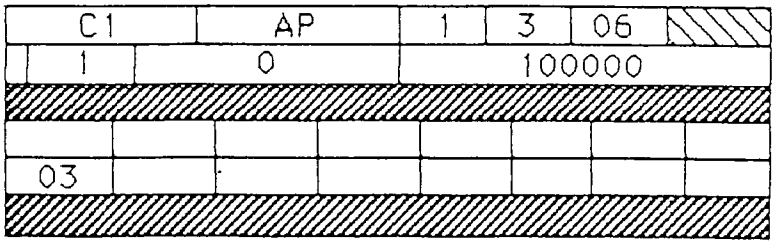


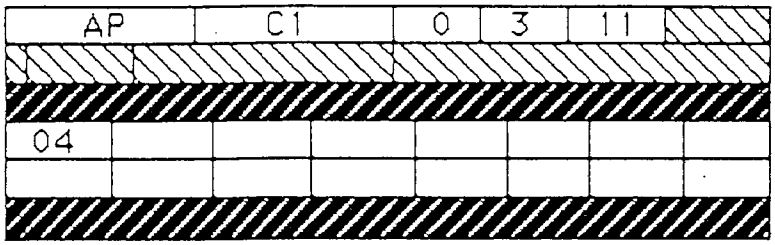
CONTROL ➡ FE

RCV

CRAY ⬅ DATASET HEADER

SND



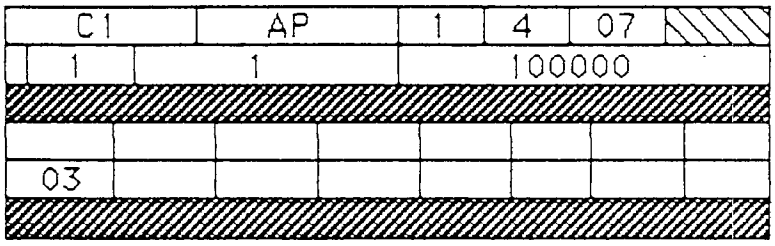


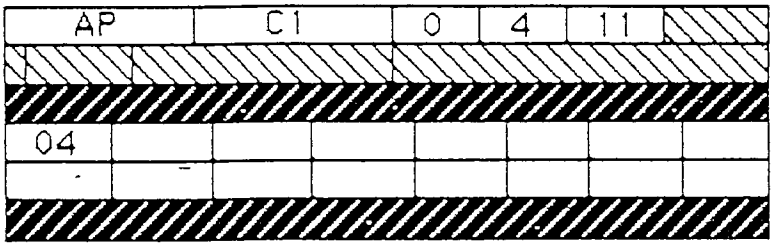
CONTROL ➡ FE

RCV

CRAY ⬅ DATASET SEGMENT

SND



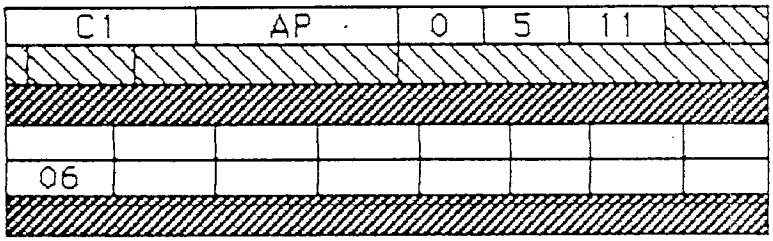


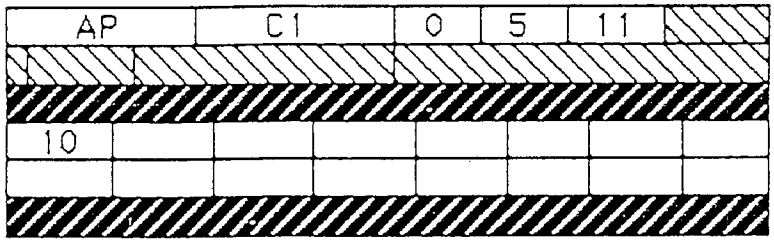
CONTROL ➡ FE

RCV

CRAY ⬅ CONTROL

END



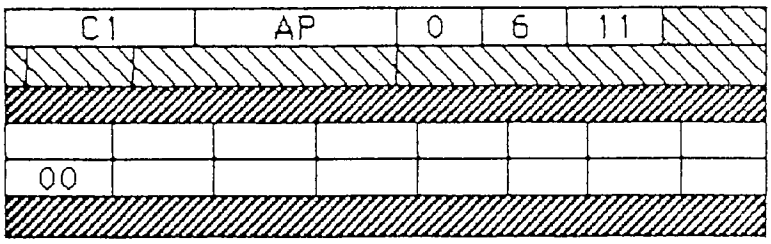


CONTROL ➡ FE

SVD

CRAY ⬅ CONTROL

IDLE



Exercise 6 COS Dump Analysis

Skill: To analyze a COS dump for Failure

COS DUMP 1

a. Locate the Stop buffer

address 1474 STOP08 STP hang

b. Locate the STP hang message if an STP hang

address 32310 task 2 Hang address 333007B

-323540B

c. Locate the STOP macro

Listing Relative 6247B

d. Locate the ERR macro if an STP hang

ERRAN macro in EXP at address 6247B

e. Locate the active Task

EXP

f. Locate the active task exchange package

address 4240

g. Analyze the EXEC history trace Events

CRASHF flag was set non-zero with SCP debugger by analyst
Then a job was submitted with a F$CRASH System Action Request
This is useful for debugging

Intended Lesson Results: To be able to find out which program halted COS and study the events that lead up to the halt and why it halted.

Cos Dump 1

DXTR TYPE=CPU

FUNCT	P-ADDRESS	SAVED	BOO	XA	PN	SM00-SM13	INTERVAL	DATA		
EXEC ASC	0411461060	000000008	1761	0	0	01010110000000	00000000000000	0360741322645513251524	0475201322645513237076	<<----STOP---->>
EXEC ENE	003240020	00324012C	5000	0	0	01010000000000	00000000014356	0000000000000025047472	0425302400000000000000	TO:EXP
EXEC UNE	000002000	00000000A	4100	0	0	01010000000000	00000000001473	0000000000000000000100	0000000000000000000000	@
EXEC ENE	00000200A	00000000A	4100	0	0	01010000000000	00000000001472	0000000000000025047472	0525232125100000213300	TO:USER
EXEC SNE	003240028	00324012C	5000	0	0	01010000000000	00000000001430	0425302402652325251520	0000000000000000000003	EXP-SUSP
EXEC ENE	003240020	00324012C	5000	0	0	01010000000000	00000000004033	0000000000000025047472	0425302400000000000000	TO:EXP
EXEC SNE	00442476D	00442631D	5160	0	0	01010000000000	00000000002376	0000000000000025047472	0451232202652325251520	0000000000000000000003 JSH-SUSP
EXEC ENE	004530360	00446172D	5160	0	0	01010000000000	00000000001700	0451232202652325251520	0451232200000000000000	TO:JSH
EXEC SNE	004530368	00446172D	5160	0	0	01010000000000	00000000001162	0000000000000000213300	0000000000000001224020	
EXEC ENE	00446130B	00442631D	5160	0	0	01010000000000	00000000003337	0000000000000025047472	0451232200000000000000	TO:JSH

ITEM COUNT EXCEEDS 10 ITEM DXTR LIMIT; DIRECTIVE TERMINATED

**** FILES,DDS=DUMP2,SYM1=EXECSYM.
**** AUTO.

13

COS Dump 1

```
DMEM,FWA=0,LWA=16000.                EXEC TABLES                 FDUMP 1.14        01/08/85    18:07:03     PAGE    11
                                                                 SYSDUMP           01/04/85    00:44:57

0005670 00302400004510156051600 00451657000000000023350 00000000000000000000000 00000000000000001332020  H7 p N   &
0005674 00402400004510162051600 00451657000000000001162 00000000000000025047472 04512322000000000000000  H9 p N    r      TO:JSH
0005700 00302400002770212051600 00451255200000000026456 00000000000000000000006 00000000000000000000014   E p J   -.
0005704 00402400004401032051000 00440105600000000001622 00000000000000025047472 04652321600000000000000     @                TO:MSG
0005710 00302400004401026051000 00440105600000000014754 00000000000000000000006 00000000000000000000003     @
0005714 00402400002770216051600 00451255200000000001441 00000000000000025047472 04512322000000000000000   G p J    !      TO:JSH .
0005720 00302400004530362051600 00447450600000000003651 00000000000000000171125 00000000000000001224020  Xy p <               U       (
0005724 00402400004530366051600 00447450600000000001162 00000000000000025047472 04512322000000000000000  X| p <    r      TO:JSH
0005730 00302400004461276051600 00442631600000000004441 10200000000000000213300 00000000000463473530016  1_ p   g  !                 3
0005734 00402400004461302051600 00442631600000000003315 00000000000000025047472 04512322000000000000000  1a p   g         TO:JSH
0005740 00302400004530362051600 00446172600000000003337 00000000000000000213300 00000000000000001224020  Xy p 1                      (
0005744 00402400004530366051600 00446172600000000001162 00000000000000025047472 04512322000000000000000  X| p 1     r      TO:JSH
0005750 00302400004424766051600 00442631600000000002376 04512322026523252515200 00000000000000000000003     p   g     JSH-SUSP
0005754 00402400003240026050000 00324012400000000001700 00000000000000025047472 04253024000000000000000          *           TO:EXP
0005760 00302400003240022050000 00324012400000000004033 04253024026523252515200 00000000000000000000003          *     EXP-SUSP
0005764 00402400000002000041000 00000000000000000001430 00000000000000025047472 05252321251000000213300    @                 TO:USER
0005770 00202400000002006041000 00000000000000000001472 00000000000000000000100 00000000000000000000000    @          :         @
0005774 00402400003240026050000 00324012400000000001473 00000000000000025047472 04253024000000000000000          *    :      TO:EXP
0006000 04202540004146106401761 00000000200000000014356 03607413226455132515240 04752013226455132370760 D CI          <<----STOP---->>
0006004 00402400004461302051600 00442631600000000003315 00000000000000025047472 04512322000000000000000  1a p   g         TO:JSH
0006010 00302400004530362051600 00446172600000000003337 00000000000000000213300 00000000000000001224020  Xy p 1                      (
0006014 00402400004530366051600 00446172600000000001162 00000000000000025047472 04512322000000000000000  X| p 1     r      TO:JSH
0006020 00302400004424766051600 00442631600000000002376 04512322026523252515200 00000000000000000000003     p   g     JSH-SUSP
0006024 00402400003014560041000 00025377600000000001430 00000000000000025047472 05252321251000000213300    @                 TO:USER
0006030 00202400000115050041000 00007407400000000021756 00000000000000000000014 00000000000000000011464   M @ <     #                    4
0006034 00402400003240026050000 00324012400000000001473 00000000000000025047472 04253024000000000000000          *    ;      TO:EXP
0006040 00302400003240022050000 00324012400000000007053 04253024026523252515200 00000000000000000000003          *    +EXP-SUSP
0006044 00402400000115050041000 00007407400000000001430 00000000000000025047472 05252321251000000213300   M @ <              TO:USER
0006050 00202400000115200041000 00011506200000000001721 00000000000000000000055 10000000000000000011462  M@ @ M                -          2
0006054 00402400003240026050000 00324012400000000001473 00000000000000025047472 04253024000000000000000          *    ;      TO:EXP
0006060 00302400003240022050000 00324012400000000030273 04253024026523252515200 00000000000000000000003          *   0 EXP-SUSP.
0006064 00402400000115200041000 00011506200000000001430 00000000000000025047472 05252321251000000213300  M@ @ M               TO:USER
0006070 00202400000115350041000 00011506200000000001513 00000000000000000000010 10000000000000000011462  Mt @ M     K                    2
0006074 00402400003240026050000 00324012400000000001473 00000000000000025047472 04253024000000000000000          *    ;      TO:EXP
0006100 00302400002770212050000 00353066000000000063717 00000000000000000000011 00000000000000000000014   E   X    g
0006104 00402400004424772051600 00442631600000000001620 00000000000000025047472 04512322000000000000000     p   g         TO:JSH
0006110 00302400004431264051600 00443106400000000002734 00000000000000000213313 00000000000000006532020   Z p
0006114 00402400004431270051600 00443106400000000001162 00000000000000025047472 04512322000000000000000   \ p       r      TO:JSH
0006120 00302400004455232051600 00445221600000000004726 10000000000000000213300 00000000000000000000017  -M p *G
0006124 00402400004455236051600 00445221600000000002561 00000000000000025047472 04512322000000000000000  -0 p *G    q      TO:JSH
0006130 00302400004530362051600 00443352200000000003475 00000000000000000171125 00000000000000001224020  Xy p       =        U       (
0006134 00402400004530366051600 00443352200000000001162 00000000000000025047472 04512322000000000000000  X| p       r      TO:JSH
0006140 00302400002772272051600 00000000000000000002373 00000000000000000000002 00000000000000000000002   ] p
0006144 00402400002770216050000 00353066000000000001342 00000000000000025047472 04253024000000000000000   G   X             TO:EXP
0006150 00302400003240022050000 00324012400000000003601 04253024026523252515200 00000000000000000000003          *     EXP-SUSP
0006154 00402400002772276051600 00000000000000000001236 00000000000000025047472 04512322000000000000000     p              TO:JSH
0006160 00302400004514144051600 00451604200000000006025 00000000000000000054456 00000000000000001232020  L2 p N              Y.        4
0006164 00402400004514150051600 00451604200000000001162 00000000000000025047472 04512322000000000000000  L4 p N     r      TO:JSH
0006170 00302400004510156051600 00451614200000000003535 00000000000000000054456 00000000000000001332020  H7 p N1    |        Y.
0006174 00402400004510162051600 00451614200000000001162 00000000000000025047472 04512322000000000000000  H9 p N1    r      TO:JSH
0006200 00302400004510156051600 00451657000000000006215 00000000000000000054456 00000000000000001332020  H7 p N              Y.
0006204 00402400004510162051600 00451657000000000001162 00000000000000025047472 04512322000000000000000  H9 p N     r      TO:JSH
0006210 00302400004530362051600 00447450600000000015307 00000000000000000171125 00000000000000001224020  Xy p <               U       (
0006214 00402400004530366051600 00447450600000000001162 00000000000000025047472 04512322000000000000000  X| p <     r      TO:JSH
0006220 00302400004461276051600 00442631600000000004441 10200000000000000213300 00000000000472515340016  1_ p   g   !                :
0006224 00402400004461302051600 00442631600000000003315 00000000000000025047472 04512322000000000000000  1a p   g          TO:JSH
0006230 00302400004530362051600 00446172600000000003337 00000000000000000213300 00000000000000001224020  Xy p 1                      (
```

40

```
*****************************************************************HC
**                                                               HC
* NAME    CRASH                                                 HC
*                                                               K
* PURPOSE  TO ALLOW A USER TO HALT COS (TO BE USED FOR DEBUGGING HC
*          PURPOSES).                                           HC
*                                                               HC
* ENTRY  A1 == TCB ADDR                                         HC
*        A2 == JTA ADDR                                         HC
*                                                               HC
* EXIT   A1 == TCB ADDR                                         HC
*        A2 == JTA ADDR                                         HC
*                                                               HC
* METHOD IF (PARCEL 0 OF CRASHF NONZERO) HALT COS               HC
*        ELSE  ABORT THE USER WITH "UNDEFINED USER CALL"        HC
*                                                               HC
**                                                              HC
*****************************************************************HC
                                                                HC
7247b4  CRASH   =       *                                       HC
7247b4 1000 00000000X   A0      CRASHF,0                        HC
     d  <macro>         ERRAN                   IF (CRASHF NONZERO) HALT
7251b4 0401 00000032    S1      ERIUC           ELSE "INVALID USER CALL
     d 006-00001461b4   J       ABTJ1             AND ABORT THE USER
```



```
*****************************************************************
**
* NAME    CSB
*
* PURPOSE - PERFORM THE VARIOUS FUNCTIONS WHICH ARE RELATED TO
*           CONDITIONAL CONTROL STATEMENT BLOCKS.
```

```
PROJECT: U15¤2                ¤E¤BER: CRASH                 DATE: 85/¤2/13
LIBRARY: SWCE                 LEVEL:  ¤1.21                 TI¤E: 18:43
TYPE:    CNTL                 USERID: U15¤2                 PAGE: ¤1 OF ¤1
START
 COL  ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8

   1  JOB,JN=U15¤2A.
   1  ACCOUNT,AC=2651¤¤,US=SYSTE¤,UPW=CRAY.
   1  ********************************************************************************
   1  *
   1  *     THIS JOB CREATES THE DATASET FOE CESW I EXERCISE 1
   1  *
   1  ********************************************************************************
   1  CAL.
   1  LDR.
   1  /EOF
  11            IDENT     CRASH
  11            START     HERE
   1  HERE       =         *
  11            S¤        F$CRASH
  11            EX
  11            ENDP
  11            END
```

```
          ...         CYCTIME   $CYCLES   INTERVAL=1,UNITS=SEC,TYPE=RECIP                              LC5922WD.3
          0377405006325610650743O  CYCTIME   CON       D'9499999990.0E-18  1/105263158                 REC.
```



```
12054   0405022305112400000000  ABORTF   CON      'ABORT'L      JOB-ABORT FLAG:  SET PARCEL 0 NONZERO   STPTAB.586
                                                                  CRASH SYSTEM WHEN ANY JOB ABORTS       STPTAB.58
12055   0415222025151000000000  CRASHF   CON      'CRASH'L      COS-HALT FLAG:  SET PARCEL 0 NONZERO    H06586CP.59
                                *                                  TO ENABLE F$CRASH CALL               H06586CP.60
12056   0251242406415172535307          CON      '*TSCONV='                                             LC5923RB.2
12057   0000000000000000047516  TSCONV   CON      'ON'R         NZ if *TSCONV enabled                   LC5923RB.3
12060   0000000000000000000000  JXTMAX   CON      D'0           MAX ACTIVE JXT ENTRIES ALLOWED.         STPTAB.588
12061   0000000000000000000010  TQUICK   CON      D'8           TIME OF A QUICK JOB.                    STPTAB.589
12062   0000000000000000000020  TLBIAS   CON      D'16          BIAS FOR INITIAL PRIORITY CURVE.        STPTAB.590
                                *                                                                       STPTAB.591
                                *                                                                       N08113ED.7
                                *         SYSTEM EXECUTION TABLE (SXT)                                  N08113ED.8
                                *         THE SYSTEM EXECUTION TABLE IS USED TO CONTROL WHICH TASKS HAVE N08113ED.9
                                *         EXECUTION PROFILING ENABLED FOR THEM AND WHERE THE APPROPRIATE N08113ED.10
                                *         INFORMATION IS.   THERE IS ONE ENTRY PER TASK. IN TASK ID ORDERN08113ED.11
                                *                                                                       N08113ED.12
12063   0515302500000000000000           CON      'SXT'L                                                N08113ED.13
12064                       17  B@SXT    BSSZ     MAXTN                                                 N08113ED.14
                                *                                                                       N08113ED.15
                                *         MEMORY POOL TABLES                                            STPTAB.592
                                *                                                                       STPTAB.593
                         6      SZ@MPH   =        SZ@MPH        GET SYMBOL INTO SYMBOL TABLE            STPTAB.594
12103   0000000000000000000005  POOLTBL  CON      MAXPOOL       MAXIMUM VALID POOL NUMBER               N09044BB.45
12104   0000000000372000012111           VWD      D'16/0,D'24/MP1S,D'24/POOL1                           STPTAB.596
12105   0000000000100000016031           VWD      D'16/0,D'24/MP2S,D'24/POOL2                           STPTAB.597
12106   0000000000000000017031           VWD      D'16/0,D'24/MP3S,D'24/POOL3                           KC4532UA.2
12107   0000000000000300017033           VWD      D'16/0,D'24/MP4S,D'24/POOL4 ISP POOL                  N09044BB.46
12110   0000000000026400017036           VWD      D'16/0,D'24/MP5S,D'24/POOL5                           N09402AC.5
12111   0000000101010100003720  POOL1    VWD      D'16/0,D'24/O'01010101,D'24/MP1S                      STPTAB.598
12112                     3716           BSSZ     MP1S-MPHT,                                            STPTAB.599
16030   0000000101010100003720           VWD      D'16/0,D'24/O'01010101,D'24/MP1S                      STPTAB.600
16031   0000000202020200001000  POOL2    VWD      D'16/0,D'24/O'02020202,D'24/MP2S                      STPTAB.601
16032                      776           BSSZ     MP2S-MPHT                                             STPTAB.602
17030   0000000202020200001000           VWD      D'16/0,D'24/O'02020202,D'24/MP2S                      STPTAB.603
17031   0000000303030300000000  POOL3    VWD      D'16/0,D'24/O'3030303,D'24/MP3S                       KC4532UA.3
                                         IFE      I@MP3SZ,NE,0,1  DEFINE A POOL IF TAPE DRIVES DEFINED  KC4532UA.4
        <skipped>                        BSSZ       MP3S-MPHT                                           KC4532UA.5
17032   0000000303030300000000           VWD      D'16/0,D'24/O'3030303,D'24/MP3S                       KC4532UA.6
17033   0000000404040400000003  POOL4    VWD      D'16/0,D'24/O'04040404,D'24/MP4S                      N09044BB.47
17034                        1           BSSZ     MP4S-MPHT                                             N09044BB.48
17035   0000000404040400000003           VWD      D'16/0,D'24/O'04040404,D'24/MP4S                      N09044BB.49
17036   0000000505050500000264  POOL5    VWD      D'16/0,D'24/O'05050505,D'24/MP5S                      N09402AC.6
17037                      262           BSSZ     MP5S-MPHT                                             N09402AC.7
17321   0000000505050500000264           VWD      D'16/0,D'24/O'05050505,D'24/MP5S                      N09402AC.8
17322   0000000000000000000000  IDWRD    CON      0             MEMORY POOL ID'S                        STPTAB.604
17323   0000000000000001010101           CON      O'01010101                                            STPTAB.605
17324   0000000000000002020202           CON      O'02020202                                            STPTAB.606
17325   0000000000000003030303           CON      O'03030303                                            KC4532UA.7
```

Exercise 7 IOS Dump Analysis

Skill: To Analyze an IOS dump for failure

IOS DUMP 1

a. Locate the hang message in the punted IOP

address 5630 IOP1 HALT 004

b. Locate the PUNTIF macro

address 12141 in the Kernel High Speed Channel Driver

c. Analyze the history trace table events

d. Locate the active overlay number, name and activity descriptor

#10 AMSG activity descriptor is at 45310

e. Locate the active SMOD (ESMD)

address 45334

f. Locate the relative P register of the active programs SMOD

p= 1174 relative to %B

g. Determine what operand registers have useful overlay parameter registers

%HISP contains the channel number

High speed channel transfer from AMSG and the Busy & Done flag were set on completion and recovery was tryed

Intended Lesson Results: To be able to find out which program halted IOS

```
1113

1113    <macro>
1120    <macro>
1125    <macro>
1132    <macro>
1137    <macro>
1144    <macro>

1152    <macro>


1174    <macro>
1201    <macro>
1210   002000  000000
1212   020430  024000
1214    <macro>

1216   002000  000000
1220   020430  024000
1222    <macro>
1224   075000 /000006


1226
1226    <macro>
1233    <macro>

1235    <macro>
1237    <macro>

1237   071713
```

```
*****************************************************************      AMSG.339
*                                                               *      AMSG.340
*       AM$XFER   Transfer data between the Cray and buffer memory.     AMSG.341
*                                                               *      AMSG.342
*****************************************************************      AMSG.343
                                                                       AMSG.344
XFER    *                                                              AMSG.345
                                                                       AMSG.346
*       Initialize registers                                           AMSG.347
                                                                       AMSG.348
        GET     R!MG,DA@LNO,R!DAL   .transfer length                   AMSG.349
        GET     R!MI,DA@CM0,R!DAL   .Cray address                      AMSG.350
        GET     R!MJ,DA@CM1,R!DAL                                      AMSG.351
        GET     R!MK,DA@BM0,R!DAL   .buffer memory address             AMSG.352
        GET     R!ML,DA@BM1,R!DAL                                      AMSG.353
        GET     R!MM,DA@FC,R!DAL    .function code                     AMSG.354
                                                                       AMSG.355
        TRANSFER  R!MM,R!MI,R!MJ,R!MK,R!ML,R!MG                        AMSG.356
                                                                       AMSG.357
*       Send a reply                                                   AMSG.358
                                                                       AMSG.359
        STORE  AM$XFRDN,DA@IFC,R!DAL                                   AMSG.360
        STORE  0,DA@RC,R!DAL                                           AMSG.361
        I = 0                       .disable interrupts                AMSG.362
        R!DS = R!DAL                                                   AMSG.363
        RKERN  DMGA                 .send message to the MIOP          AMSG.364
                                                                       AMSG.365
        I = 0                       .disable interrupts                AMSG.366
        R!DY = R!DAL                                                   AMSG.367
        RKERN  !IONN                .release the local DAL             AMSG.368
        P = MIMM                                                       AMSG.369


*****************************************************************      AMSG.371
*                                                               *      AMSG.372
*       AM$XFRDN  Transfer between the Cray and buffer memory has      AMSG.373
*                 completed.  Reactivate the waiting activity.  *      AMSG.374
*                                                               *      AMSG.375
*****************************************************************      AMSG.376
                                                                       AMSG.377
                                                                       AMSG.378
XFRDN   *                                                              AMSG.379
        GET     R!EH,DA@ACT,R!DAL   .Get issuing activity address      AMSG.380
        $IF (A # 0)                                 .>>>>> 0       $IF     1
                                                                       AMSG.381
          RKERN    EQCP             .Reschedule the activity          AMSG.382
        $ENDIF                                      .<<<<< 0       $ENDIF  1
                                                                       AMSG.383
        P = MFREED                  .Dispose of the DALs
```

```
                                                                                         $ENDIF      1
                                                                         .<<<<< 1       K.1677
                                                   .TRY AGAIN                            K.1678
                          P = CEREC                                                      $ENDIF      1
12032   070074          $ENDIF                                           .<<<<< 0       K.1679
12033    <macro>                                                                         K.1680
                         EXIT                                                            K.1681
12033   001000                                                                           K.1682
                   ************************ MEMORY I/O SUBROUTINES ********************** K.1683
                   *                                                                   * K.1684
                   ********************************************************************* K.1685
                   *                                                                   * K.1686
                   *             WAIT FOR MEMORY I/O TO COMPLETE                       * K.1687
                   *                                                                   * K.1688
                   *   B - CHANNEL NUMBER TO TEST                                        K.1689
                   *                                                                     K.1690
                   *********************************************************************  K.1691
                                                                                         K.1692
                   CHNWTDN  *                        .INITIALIZE TIMER                   K.1693
12034                   R!MIOTICK = 0                                                    K.1694
12034   010000  024071  R!MIOTOCK = A                .UNTIL I/O COMPLETE                 K.1695
12036   024102          $UNTIL (IOB = DN)            .COUNT LOOPS                        K.1696
12037    <macro>          R!MIOTICK = R!MIOTICK + 1  .IF MILLISECOND EXPIRES              $IF        1
12041   026071            $IF (A = MIO$MSEC)                           .>>>>> 0          K.1697
12042    <macro>                                                                         K.1698
                                                                                         K.1699
                      * IT IS POSSIBLE FOR THE I/O CHANNELS                              K.1700
                      * BETWEEN THE IOS AND THE CPU TO BE LOCKED OUT                     K.1701
                      * INDEFINITELY BY HIGHER PRIORITY CONSTANT                         K.1702
                      * MEMORY REFERENCES SUCH AS VECTOR STORES AND                      K.1703
                      * LOADS. THIS CODE GENERATES A REQUEST TO THE CPU                  K.1704
                      * TO INTERRUPT CURRENT MEMORY REFERENCES TO ALLOW                  K.1705
                      * I/O TO COMPLETE.                                                 K.1706
                                                                                          $IF        1
                      $IF (B = R!%HISP) OR (B = R!%HISP + 1) .HIGH-SPEED CHANNEL
                                                               .>>>>> 1                  K.1707
12046    <macro>                                                                          $IF        1
                        $IF (R!MIOTOCK = 0)     .FIRST MS ONLY   .>>>>> 2                K.1708
12060    <macro>                                                                         K.1709
                          I = 0                   .SAVE B                                K.1710
12063   002000  000000    R!CF = B                .DESTINATION IOP                       K.1711
12065   050000  024025    R!EI = MIOP                                                    K.1712
12067   010000  024036    RSTORE MQ$DLAY,DA@QUE,R!EH .MESSAGE                            K.1713
12071    <macro>          R = EMSGIOP             .SEND MESSAGE                          K.1714
12076   077000 /003316    B = R!CF                .RESTORE B                             K.1715
12100   020025  054000                                                                   K.1716
                          * LOG OCCURENCE                                                K.1717

                          (HSPLOCK) = (HSPLOCK) + 1                                      K.1718
12102   014000 /005446  024001                                                            $ENDIF     1
        036001          $ENDIF                                    .<<<<< 2               K.1719
12106    <macro>                                                                          $ENDIF     1
                      $ENDIF                                      .<<<<< 1               K.1720
12106    <macro>                                                                         K.1721
                                                                                         K.1722
                      * UPDATE COUNTERS                                                  K.1723
                                                                                         K.1724
12106   010000  024071  R!MIOTICK = 0             .CLEAR LOOP COUNTER                    K.1725
1211(   026102          R!MIOTOCK = R!MIOTOCK + 1 .ADVANCE TIMEOUT COUNTER
```

96

IOS DMP1

DXTR TYPE=IOP1

FDUMP 1.14 01/31/85 15:54:27 PAGE 54
SYSDUMP 01/03/85 23:55:42

EVENT	RTC-L	OVL-#	PAR-1	PAR-2	PAR-3	PAR-4	PAR-5
MEM-IO	007655	AMSG	000006	000001	057301	060000	000006
K-CALL	007655	AMSG	TRANSF	001174	045334	045310	000457
INTRPT	007655	AMSG	000006	025536	025530	000045	075674
I-HAND	007655	AMSG	000006	100313	047125	000077	000023
TASK	007655	AMSG	025530	000006	000010	045310	000023
ACOM	007655	ACOM	000043	041254	100312	000000	000000
MEM-IO	007655	ACOM	000002	000000	013730	041254	000010
TASK	007655	ACOM	031064	000006	000007	045250	000023
INTRPT	007655	KERNEL	000006	013107	000000	000045	075674
I-HAND	007655	KERNEL	000006	100312	047124	000100	000023
INTRPT	007654	KERNEL	000007	013114	000000	000045	065043
A-TO-A	007654	AMSG	000007	100311	000044	000000	000400
MEM-IO	007654	AMSG	000003	000000	013720	042014	000010
TASK	007654	AMSG	025530	001174	000010	045310	000023
MEM-IO	007654	AMSG	000010	000037	107517	062500	000260
MEM-IO	007654	AMSG	000002	000001	060520	062500	000260
MEM-IO	007654	AMSG	000010	000037	106777	060000	000520
MEM-IO	007654	AMSG	000002	000001	060000	060000	000520
K-CALL	007654	AMSG	TRANSF	001174	045334	045310	000457
TASK	007654	AMSG	025530	000006	000010	045310	000023
ACOM	007654	ACOM	000043	042014	100311	000000	000000
INTRPT	007654	ACOM	000007	012036	031064	000045	064637
MEM-IO	007654	ACOM	000002	000000	013720	042014	000010
TASK	007654	ACOM	031064	000006	000007	045250	000023
A-TO-A	007654	AMSG	000007	100310	000044	000000	000400
MEM-IO	007654	AMSG	000003	000000	013710	037554	000010
TASK	007654	AMSG	025530	001174	000010	045310	000023
MEM-IO	007654	AMSG	000010	000001	057271	060000	000006
MEM-IO	007654	AMSG	000002	000001	024000	060000	000006
K-CALL	007654	AMSG	TRANSF	001174	045334	045310	000457
INTRPT	007654	AMSG	000006	025536	025530	000045	064637
I-HAND	007654	AMSG	000006	100311	047123	000077	000023
TASK	007654	AMSG	025530	000006	000010	045310	000023
ACOM	007654	ACOM	000043	037554	100310	000000	000000
MEM-IO	007654	ACOM	000002	000000	013710	037554	000010
TASK	007654	ACOM	031064	000006	000007	045250	000023
INTRPT	007654	KERNEL	000006	013114	000000	000045	064637
I-HAND	007654	KERNEL	000006	100310	047122	000100	000023
INTRPT	007476	KERNEL	000007	013110	000000	000035	107710
A-TO-A	007476	AMSG	000007	100307	000044	000000	000000
MEM-IO	007476	AMSG	000003	000000	013700	037314	000010
TASK	007476	AMSG	025530	001174	000010	045310	000023
MEM-IO	007476	AMSG	000004	000001	057520	062500	000260
MEM-IO	007476	AMSG	000006	000037	107517	062500	000260
MEM-IO	007476	AMSG	000004	000001	057000	060000	000520
INTRPT	007476	AMSG	000007	012044	025530	000035	107710
MEM-IO	007476	AMSG	000006	000037	106777	060000	000520
K-CALL	007476	AMSG	TRANSF	001174	045334	045310	000457
TASK	007476	AMSG	025530	000006	000010	045310	000023
ACOM	007476	ACOM	000043	037314	100307	000000	000000
MEM-IO	007476	ACOM	000002	000000	013700	037314	000010

Exercise 6 COS Dump Analysis

Skill: To analyze a COS dump for Failure

COS DUMP 2

a. Locate the Stop buffer

address 1474 STOP06 Operand Range Error in STP

b. Locate the STP hang message if an STP hang

none

c. Locate the STOP macro

address 17557B

d. Locate the ERR macro if an STP hang

None

e. Locate the active Task

JSH The Job Scheduler at address 217 relative in COPY

f. Locate the active task exchange package

address 5160

g. Analyze the EXEC history trace events

An Operand Range Error occured while JSH was using an STP common routine COPY to move a job using V0 and V1 block transfers, instruction 176 and 177.

Intended Lesson Results: To be able to find out which program halted COS and study the events that lead up to the halt and why it halted.

```
                        ********************************************************************************  N09340BA.1
                        **                                                                                N09340BA.2
                        *    NAME:      MOVEMEM                                                           N09340BA.3
                        *                                                                                 N09340BA.4
                        *    PURPOSE:   To move a block of words in memory                                N09340BA.5
                        *                                                                                 N09340BA.6
                        *    ENTRY:     A3 - Address of destination area                                  N09340BA.7
                        *               A5 - Address of source area                                       N09340BA.8
                        *               A7 - Number of words to move                                      N09340BA.9
                        *                                                                                 N09340BA.10
                        *    EXIT:      (A7) words are moved from (A5) to (A3)                            N09340BA.11
                        *                                                                                 N09340BA.12
                        *    DESTROYS:  (A0),(S0-S3)                                                      N09340BA.13
                        *                                                                                 N09340BA.14
                        *    METHOD:    If A3=A5, then return, else call common routine COPY.             N09340BA.15
                        **                                                                                N09340BA.16
                        ********************************************************************************  N09340BA.17
                                                                                                          N09340BA.18
                  7333d+ MOVEMEM  =          *                                                            N09340BA.19
  7333d  <macro>                   $JUMP      MEM10,A3,E0,A5                                               N09340BA.20
                                                                                                          N09340BA.21
  7334c  024000                    A0         B00                                                          N09340BA.22
      d  1100 00007350+            MEMB0,0    A0                                                           N09340BA.23
  7335b  1306 00007351+            MEMS6,0    S6                                                           N09340BA.24
      d  1307 00007352+            MEMS7,0    S7                                                           N09340BA.25
                        *                                                                                 N09340BA.26
                        *       Make a POST call.                                                         N09340BA.27
                        *                                                                                 N09340BA.28
  7336b  071105                    S1         A5                                                           N09340BA.29
      c  071203                    S2         A3                                                           N09340BA.30
      d  071707                    S7         A7                                                           N09340BA.31
  7337a  054130                    S1         S1<D'24                                                      N09340BA.32
      b  054230                    S2         S2<D'24                                                      N09340BA.33
      c  051117                    S1         S1!S7                                                        N09340BA.34
      d  0407 00046526             S7         'MV'R                                                        N09340BA.35
  7340b  054760                    S7         S7<D'48                                                      N09340BA.36
      c  051117                    S1         S1!S7                                                        N09340BA.37
      d  <macro>                   POST       32,S1,S2                                                     N09340BA.38
                        *                                                                                 N09340BA.39
                        *       Set up registers for COPY call.                                           N09340BA.40
                        *                                                                                 N09340BA.41
  7342a  071104                    S1         A4                                                           N09340BA.42
      b  071206                    S2         A6                                                           N09340BA.43
      c  071307                    S3         A7                                                           N09340BA.44
                                                                                                          N09340BA.45
      d  030405                    A4         A5          Source address                                   N09340BA.46
  7343a  030503                    A5         A3          Destination address                              N09340BA.47
      b  030607                    A6         A7          Word count                                       N09340BA.48
      c  0207 00000000X            A7         JSHSTK      Stack pointer                                    N09340BA.49
  7344a  007 00000000aX            R          COPY                                                         N09340BA.50
                        *                                                                                 N09340BA.51
                        *       Restore registers.                                                        N09340BA.52
                        *                                                                                 N09340BA.53
      c  030504                    A5         A4                                                           N09340BA.54
      d  023410                    A4         S1                                                           N09340BA.55
  7345a  023620                    A6         S2                                                           N09340BA.56
      b  023730                    A7         S3                                                           N09340BA.57
```



```
  7345c  1206 00007351+            S6         MEMS6,0                                                      N09340BA.58
  7346d  1207 00007352+            S7         MEMS7,0                                                      N09340BA.59
      c  1000 00007350+            A0         MEMB0,0                                                      N09340BA.60
  7347a  025000                    B00        A0                                                           N09340BA.61
                                                                                                          N09340BA.62
                                                                                                          N09340BA.63
```

```
                                                                                COPY.8
                ********************************************************************** COPY.9
                *                                                                    * COPY.10
                *       COPY    COPY A BLOCK OF MEMORY TO ANOTHER LOCATION           * COPY.11
                *                                                                    * COPY.12
                *       ENTRY   A4 = FROM ADDRESS                                    * COPY.13
                *               A5 = TO ADDRESS                                      * COPY.14
                *               A6 = COUNT OF WORDS TO MOVE                          * COPY.15
                *               A7 = STACK POINTER                                   * COPY.16
                *                                                                    * COPY.17
                *       EXIT    BLOCK IS COPIED                                      * COPY.18
                *                                                                    * COPY.19
                *       DESTROYS  (A0),(S0,S6,S7)                                    * COPY.20
                *                                                                    * COPY.21
                ********************************************************************** COPY.22
                                                                                COPY.23
                        ENTRY   COPY                                            COPY.24
                        ENTRY   B@COPY                                          COPY.25
                        EXT     ERROR1                                          COPY.26
                        EXT     SAVEVEC                                         COPY.27
                                                                                COPY.28
 0  0250522064752026225052  B@COPY  CON     "*@COPY@*"                          COPY.29
                                                                                COPY.30
                    21  REGMAX  =       D"17                                    COPY.31
                                                                                COPY.32
 1a <macro>             COPY    $SUB    AREG=1:6                                COPY.33
                10c+    COPY    =       *       $SUB macro changes A0, A7, S6, S7   $SUB
26c 030006                      A0      A6                                      COPY.34
  d <macro>                     ERRAM           bad word count                  COPY.35
30b <macro>                     $JUMP   $RETURN,A0,Z    request of 0 words      COPY.36
  d <macro>                     $JUMP   $RETURN,A4,EQ,A5  same address          COPY.37
                                                                                COPY.38
31c 022121                      A1      REGMAX                                  COPY.39
  d 022301                      A3      1                                       COPY.40
32a <macro>                     $IF     A4,LT,A5        (source add)<(dest. add)=>  COPY.41
  d 031303                        A3      -A3             bottom up copy.       COPY.42
33a 030446                        A4      A4+A6           Adjust block pointers.  COPY.43
  b 030556                        A5      A5+A6                                 COPY.44
  c 031440                        A4      A4-1                                  COPY.45
  d 031550                        A5      A5-1                                  COPY.46
34a <macro>                     $ENDIF                                          COPY.47
                                                                                COPY.48
  a <macro>                     $JUMP   MOVVEC,A6,GT,A1  check to see if it is faster to do a  COPY.49
                                                         vector move or scalar move  COPY.50
                                                                                COPY.51
                34d+    MOVREG  =       *               scalar move             COPY.52
  d <macro>                     $LOOP   A6,N                                    COPY.53
35c 1240 00000000                 S0      0,A4                                  COPY.54
36a 031660                        A6      A6-1                                  COPY.55
  b 030443                        A4      A4+A3                                 COPY.56
  c 1350 00000000                 0,A5    S0                                    COPY.57
37a 030553                        A5      A5+A3                                 COPY.58
  b <macro>                     $ENDLOOP                                        COPY.59
  d 006 00000002a+              J       $RETURN                                 COPY.60
                                                                                COPY.61
                40b+    MOVVEC  =       *               vector move             COPY.62
40b 0202 00000100               A2      64                                      COPY.63
```

```
 40d  002002                VL      A2          set VL for vector save              COPY
 41a  1001 00000000         A1      CTID,0                                          COPY
   c  1011 00000000X        A1      SAVEVEC,A1                                      COPY
 42a  030001                A0      A1          address of vector save area         COPY
   b  <macro>               ERRAZ               ERROR if no allocated space         COPY
                                                                                    COPY
 43d  177000                ,A0,1   V0          save V0                             COPY
 44a  030012                A0      A1+A2                                           COPY
   b  177010                ,A0,1   V1          save V1                             COPY
                                                                                    COPY
   c  022240                A2      32                                              COPY
   d  002002                VL      A2                                              COPY
 45a  032223                A2      A2*A3                                           COPY
   b  0201 00000100         A1      64                                              COPY
                                                                                    COPY
   d  <macro>               $LOOP   A6,GE,A1                                        COPY
 46c  030004                 A0      A4                                             COPY
   d  030442                 A4      A4+A2                                          COPY
 47a  176003                 V0      ,A0,A3                                         COPY
   b  030004                 A0      A4                                             COPY
   c  030442                 A4      A4+A2                                          COPY
   d  176103                 V1      ,A0,A3                                         COPY
                                                                                    COPY
 50a  030005                 A0      A5                                             COPY
   b  030552                 A5      A5+A2                                          COPY
   c  177003                 ,A0,A3  V0                                             COPY
   d  030005                 A0      A5                                             COPY
 51a  030552                 A5      A5+A2                                          COPY
   b  177013                 ,A0,A3  V1                                             COPY
                                                                                    COPY
   c  031661                 A6      A6-A1                                          COPY
   d  <macro>               $ENDLOOP                                                COPY
                                                                                    COPY
 52b  030006                A0      A6                                              COPY
   c  <macro>               $IF     AN                                              COPY
 53a  002006                 VL      A6         move the residue (mod 64)           COPY
   b  030004                 A0      A4                                             COPY
   c  176003                 V0      ,A0,A3                                         COPY
   d  030005                 A0      A5                                             COPY
 54a  177003                 ,A0,A3  V0                                             COPY
   b  <macro>               $ENDIF                                                  COPY
                                                                                    COPY
   b  0201 00000100.        A1.     64          set up for restoring of the vectors COPY
   d  002001                VL      A1                                              COPY
                                                                                    COPY
 55a  1002 00000000         A2      CTID,0                                          COPY
   c  1022 00000000X        A2      SAVEVEC,A2                                      COPY
                                                                                    COPY
 56a  030002                A0      A2                                              COPY
   b  176000                V0      ,A0,1       restore V0                          COPY
   c  030021                A0      A2+A1                                           COPY
   d  176100                V1      ,A0,1       restore V1                          COPY
                                                                                    COPY
 67a  006 00000002a+        J       $RETURN                                         COPY
                            END                                                     COPY
```



```
  2a+ $RETURN               2:26 D    2:27 R    2:30    2:31    2:54    3:54
  0+  B@COPY                2:18 E    2:22 D
 10c+ COPY                  2:17 E    2:27 D
  0   CTID      COSTXT      3: 2 R    3:46 R
  0X  ERROR1                2:19 D    2:27      2:29    3: 6
 34d+ MOVREG                2:46 D
```

51

IOS DMP 2

EVENT	RTC-L	OVL-#	PAR-1	PAR-2	PAR-3	PAR-4	PAR-5
I-HAND	034701	KERNEL	000020	100012	066600	066630	000001
K-FNCT	034561	DISPLA	PUSH	073274	073274	100112	000000
K-CALL	034561	DISPLA	TPUSH	000070	064770	073274	000046
TASK	034561	DISPLA	042514	001206	000441	073274	000010
K-CALL	034561	DISPO1	RETURN	000067	065021	073274	000461
TASK	034561	DISPO1	050500	000444	000437	073274	000010
K-CALL	034561	BABEL	RETURN	001173	065065	073274	000462
TASK	034561	BABEL	046634	001130	000372	073274	000010
K-FNCT	034561	BABEL	RELMEM	002004	103474	000000	000000
K-CALL	034561	BABEL	RELMEM	001130	065065	073274	000462
TASK	034561	BABEL	046634	001120	000372	073274	000010
MEM-IO	034561	BABEL	000004	000001	030130	103500	000200
K-CALL	034561	BABEL	MOSW	001120	065065	073274	000462
TASK	034561	BABEL	046634	001223	000372	073274	000010
K-CALL	034561	CONSL	RETURN	000677	065133	073274	000456
TASK	034560	CONSL	055210	001067	000426	073274	000010
MEM-IO	034560	CONSL	000002	000001	036503	077520	000021
K-CALL	034560	CONSL	MOSR	001067	065133	073274	000456
TASK	034560	CONSL	055210	000006	000426	073274	000010
K-FNCT	034560	BABEL	CALL	065133	000426	010060	006430
K-CALL	034560	BABEL	CALL	001223	065065	073274	000462
TASK	034560	BABEL	046634	001223	000372	073274	000010
K-CALL	034560	CONSL	RETURN	000677	065133	073274	000456
TASK	034560	CONSL	055210	001067	000426	073274	000010
MEM-IO	034560	CONSL	000002	000001	036462	077520	000021
K-CALL	034560	CONSL	MOSR	001067	065133	073274	000456
TASK	034560	CONSL	055210	000006	000426	073274	000010
K-FNCT	034560	BABEL	CALL	065133	000426	010060	006430
K-CALL	034560	BABEL	CALL	001223	065065	073274	000462
TASK	034560	BABEL	046634	001223	000372	073274	000010
K-CALL	034560	CONSL	RETURN	000677	065133	073274	000456
TASK	034560	CONSL	055210	001067	000426	073274	000010
MEM-IO	034560	CONSL	000002	000001	036441	077520	000021
K-CALL	034560	CONSL	MOSR	001067	065133	073274	000456
TASK	034560	CONSL	055210	000006	000426	073274	000010
K-FNCT	034560	BABEL	CALL	065133	000426	010060	006430
K-CALL	034560	BABEL	CALL	001223	065065	073274	000462
TASK	034560	BABEL	046634	001223	000372	073274	000010
K-CALL	034560	CONSL	RETURN	000677	065133	073274	000456
TASK	034560	CONSL	055210	001067	000426	073274	000010
MEM-IO	034560	CONSL	000002	000001	036420	077520	000021
K-CALL	034560	CONSL	MOSR	001067	065133	073274	000456
TASK	034560	CONSL	055210	000006	000426	073274	000010
K-FNCT	034560	BABEL	CALL	065133	000426	010060	006430
K-CALL	034560	BABEL	CALL	001223	065065	073274	000462
TASK	034560	BABEL	046634	001223	000372	073274	000010
K-CALL	034560	CONSL	RETURN	000677	065133	073274	000456
TASK	034560	CONSL	055210	001067	000426	073274	000010
MEM-IO	034560	CONSL	000002	000001	036377	077520	000021
K-CALL	034560	CONSL	MOSR	001067	065133	073274	000456
TASK	034560	CONSL	055210	000006	000426	073274	000010

95

```
                        ;*************************************************************************          K.6567
                        ;*                    IOS-CPU INPUT INTERRUPT HANDLER                    *          K.6568
                        ;*************************************************************************          K.6569
                                                                                                           K.6570
                                ;                                                                          K.6571
                                ; CLEAR INTERRUPT - CHECK STATUS                                           K.6572
                                ;                                                                          K.6573
                                                                                                           K.6574
25565                   ICRI    ;                                                                          K.6575
25565   160000                  IOB:0                   .CLEAR CHANNEL                                     K.6576
25566   171000                  IOB:11                  .READ STATUS                                       K.6577
25567   024056                  R!!A = A                .SAVE                                              K.6578
25570    <macro>                PUT R!!A,CPI@ST,R!%LSPI .ENTER IN TABLE                                    K.6579
25575   170000                  IOB:10                  .CURRENT ADDRESS                                   K.6580
25576   024060                  R!!C = A                .SAVE                                              K.6581
25577    <macro>                PUT R!!C,CPI@CA,R!%LSPI .ENTER IN TABLE                                    K.6582
25604    <macro>                GET R!DS,CPI@DA,R!%LSPI .ACTIVE DAL ADDRESS                                K.6583
25611   020060  023210  013010 R!!D = R!!C - R!DS - DA@@LH .LENGTH OF TRANSFER                             K.6584
        024061
25615    <macro>                GET R!!B,DA@DIL,R!DS    .DESTINATION ID                                    K.6585
25623    <macro>                ADDRESS R!DZ,CPI@PQ,R!%LSPI .ADDRESS OF POLL QUEUE                         K.6586
                                                                                                           K.6587
                                ;*************************                                                  K.6588
                                ;*        TRACE        *                                                  K.6589
                                ;*                     *                                                  K.6590
                                ;* 1. CHANNEL          *                                                  K.6591
                                ;* 2. STATUS           *                                                  K.6592
                                ;* 3. START ADDRESS    *                                                  K.6593
                                ;* 4. END ADDRESS      *                                                  K.6594
                                ;* 5. DESTINATION ID   *                                                  K.6595
                                ;*************************                                                  K.6596
                                                                                                           K.6597
25626    <macro>                KRECORD TF$CHN,(B,R!!A,R!DS,R!!C,R!!B)                                     K.6598
                                                                                                           K.6599
                                ; CHECK FOR BAD LENGTH OR ERROR STATUS                                     K.6600
                                                                                                           K.6601
25645    <macro>                $IF (R!!D # DA@@LE) OR (0 # R!!A & 17)  ;ERROR?                            K.6602
                                                                                >>>>> 0                    $IF     1
25657    <macro>                  $PUNTIF CODE=PT$LOSP                                                    K.6603
25661    <macro>                $ENDIF                                                                    K.6604
                                                                                <<<<< 0                    $ENDIF  1
                                                                                                           K.6605
                                ;                                                                          K.6606
                                ; PROCESS PACKET BY TYPE                                                   K.6607
                                ;                                                                          K.6608
                                                                                                           K.6609
                                ; DISK                                                                     K.6610
                                                                                                           K.6611
25661    <macro>                $IF (R!!B = RQ$DISK)                                                      K.6612
                                                                                >>>>> 0                    $IF     1
25665    <macro>                  FLDADD 1,CPI@DP,R!%LSPI ;INCREMENT POPULATION COUNT                     K.6613
25671   072132                  R = CDEM                ;QUEUE FOR CDEM                                    K.6614
                                                                                                           K.6615
                                ; TAPE                                                                     K.6616
                                                                                                           K.6617
25672    <macro>                $ELSEIF (R!!B = RQ$BMX0)                                                  K.6618
25677   072124                  R = CDEM                .QUEUE FOR CDEM                                    K.6619
```

CIA : 11 - READ READY WAITING/ERROR FLAGS

A CIA : 11 function request reads the content of the interface status register into the accumulator. The status bit assignment is listed in table 7-10.

Table 7-10. Ready waiting/error flags

Bit	Meaning
2^0	Channel Parity Error flag for bits 2^0-2^3
2^1	Channel Parity Error flag for bits 2^4-2^7
2^2	Channel Parity Error flag for bits 2^8-2^{11}
2^3	Channel Parity Error flag for bits 2^{12}-2^{15}
2^{15}	Ready waiting flag

CHANNEL FOR OUTPUT TO CRAY MAINFRAME CHANNEL

The IOP can have one or more channels for sending data to a Cray mainframe input channel. Data is transferred in block mode directly from Local Memory. The functions are listed below.

COA : 0† Clear channel.

COA : 1† Enter Local Memory address; start transfer.

COA : 2 Enter parcel count.

COA : 3 Clear error flag.

COA : 4 Set/clear external control signals.

COA : 6†† Clear Interrupt Enable flag.

COA : 7†† Set Interrupt Enable flag.

COA : 10 Read Local Memory address.

COA : 11 Read error flags.

† Allow 1 CP before checking Busy and Done flags.
†† Allow 1 CP before checking the interrupt channel number (IOR : 10).

JOB FLOW

10

MODULE OBJECTIVES

With the aid of all furnished reference materials, upon completion of the Job Flow module, the learner should be able to:

1. Trace a Job Flow through the Systems Programs.
2. Explain when the job is passed to the next program.
3. Explain the function of each program.

STATION JOB OUTPUT

Prepare the COS job.

Submit the file containing the COS JOB to the station.

The station queues the job for transfer to the Cray.

The station asks the Cray if it can send the file.

The Cray responds saying yes you can.

The station sends the file.

The Cray places the file on the input queue.

JOB DATASET

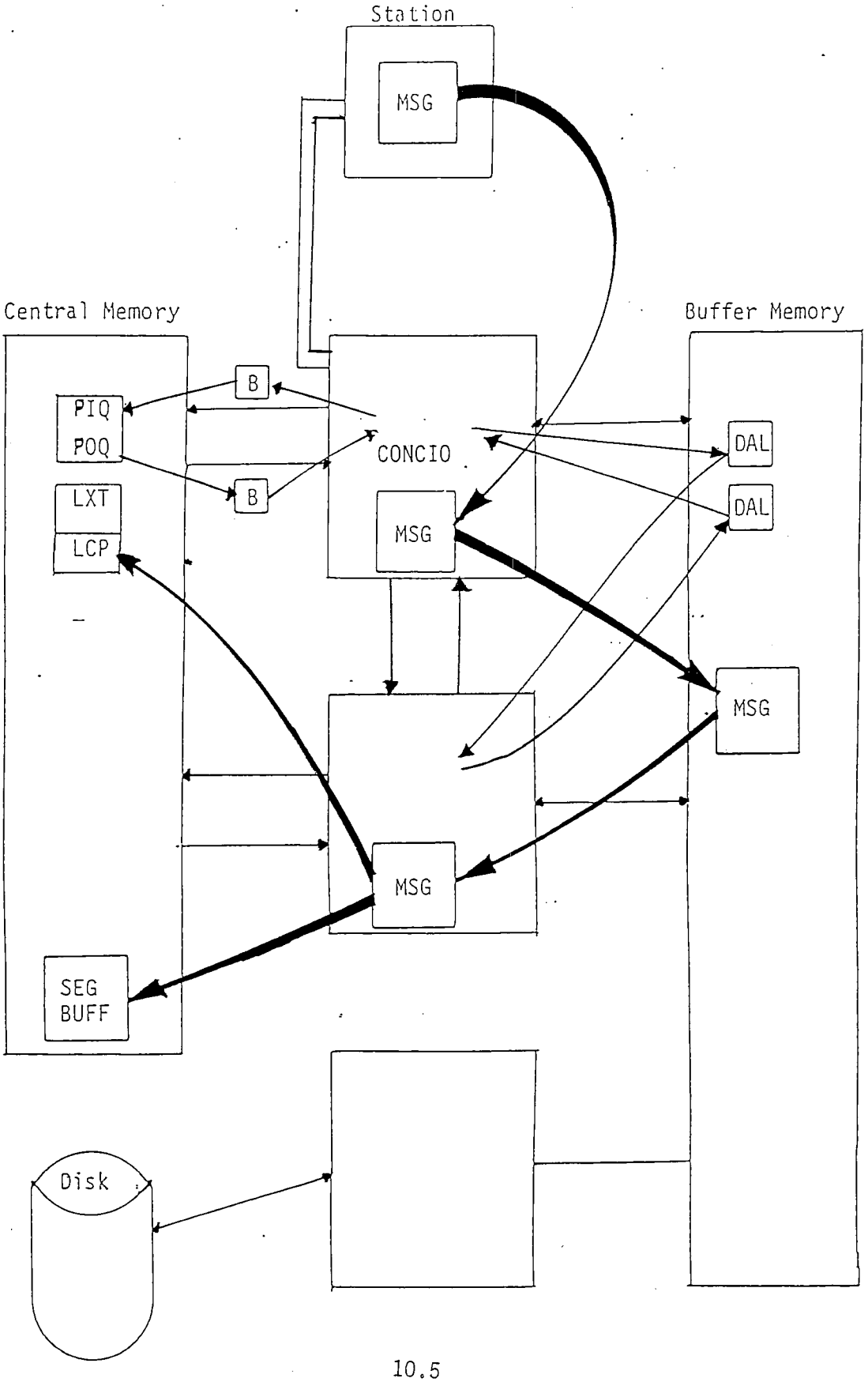
```
        SECTOR     1

000001  0000000000000000000012 0451172042611223436520 0405232464151625027040 0200401002004010020040        JOB,JN=PASSCNT.
000005  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
000009  0200401002004010020040 0200401002004010020040 0200401002004010020040 1000000000000000000012
000013  0405032064752523452054 0405031723106615230462 0320542525147525047107 0261252405347525047107 ACCOUNT,AC=265124,US=TNG,UPW=TNG
000017  0270401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040 .                 CAL.
000021  0200401002004010020040 0200401002004010020040 1000000000000000000012 0415012302704010020040
000025  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
        *****
000033  0200401002004010020040 1000000000000000000012 0461042442704010020040 0200401002004010020040            LDR.
000037  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
        *****                                                                                                        IDENT
000045  1000000000000000000000 1600000000000000000012 0200401002004010020040 0200402224210523452040    LOOP
000049  0200401002011423647520 0200401002004010020040 0200401002004010020040 0200401002004010020040
000053  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040              START      HERE
000057  1000000000000000000012 0200401002004010020040 0200402465210124452040 0200401002011021251105
000061  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
000065  0200401002004010020040 0200401002004010020040 0200401002004010020040 1000000000000000000012
000069  0250401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040 *
000073  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040                     ONE
000077  0200401002004010020040 0200401002004010020040 1000000000000000000012 0475162122004010020040   CON       1
000081  0200402064751610020040 0200401002006110020040 0200401002004010020040 0200401002004010020040
000085  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040             HERE      S1
000089  0200401002004010020040 1000000000000000000012 0441052444244010020040 0200402463044010020040    ONE,
000093  0200401002011723442454 0200401002004010020040 0200401002004010020040 0200401002004010020040
000097  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040        LOOP     S2        S2+S
000101  1000000000000000000012 0461172365004010020040 0200402463104010020040 0200401002012314425523
000105  0304401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040 1
000109  0200401002004010020040 0200401002004010020040 0200401002004010020040 1000000000000000000012         J        LOOP
000113  0200401002004010020040 0200402242004010020040 0200401002011423647520 0200401002004010020040
000117  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
000121  0200401002004010020040 0200401002004010020040 1000000000000000000012 0200401002004010020040   ENDP
000125  0200402124710424020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
000129  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040                    END
000133  0200401002004010020040 1000000000000000000012 0200401002004010020040 0200402124710410020040
000137  0200401002004010020040 0200401002004010020040 0200401002004010020040 0200401002004010020040
        *****
000145  1000000000000000000000 1600000000000000000000 1700000000000000000000 0000000000000000000000
000149  0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
        *****
000509  0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
```

OVERVIEW OF FRONT-END MESSAGE FLOW

1. Front end sends message consisting of an LCP and possibly segments and an LTP to MIOP's local memory (CONCIO).
2. MIOP writes the message to MIOP buffer memory area (CONCIO).
3. MIOP builds B packet and sends to COS SCP.
4. SCP determines what the segment buffer and the LXT address is.
5. EXEC builds a B packet with the address information, sending it back to MIOP.
6. MIOP gets central memory address from B packet and builds a DAL and writes it to buffer memory.
7. MIOP sends BIOP an accumulator message with the buffer memory address of the DAL.
8. BIOP reads the message into local memory.
9. BIOP writes the message to central memory over the high speed channel using the DAL address information.
10. When BIOP is done with the transfer, a DAL will be sent to MIOP via buffer memory.
11. MIOP builds a B packet and sends it back to SCP telling it the message is in central memory.
12. SCP processes message LCP and builds a response LCP.
13. MIOP receives response LCP address from the B packet.
14. MIOP sends address information to BIOP in a DAL through buffer memory.
15. BIOP reads COS response message from central memory to local memory.
16. BIOP writes response message to buffer memory.
17. MIOP reads response message to local memory (CONCIO).
18. MIOP sends response message to the front end (CONCIO).

STATION CONCENTRATOR



STATION CONCENTRATOR TABLES

Concentrator Table (CT@)

The Concentrator Table is a LM-resident table containing addresses and information used by a concentrator. This table is used only by software executing in the MIOP.

Concentrator ID Table (CE@)

Contains the B Packet for each logical ID logged on to COS. Contains central memory and buffer memory addresses for each ID.

Front-End Channel Information Table (FEI@)

The FEI tables are created at system initialization time. One FEI table is created for each physical channel pair configured as a front-end interface.

Front-End Channel Ordinal Reservation Table (COR@)

The COR tables are created at system initialization time. There is one COR table for each of the maximum number of front-ends concurrently logged on through all NSC Adapter channels.

NSCIO Table Definition (NIO@)

The NSCIO Table is the control table for I/O between the IOS and an A130 NSC Adapter. The table is created by the NSC routine when an NSC channel is initialized.

Input Status Buffer (NSB@) NIO@IB

This table describes the input status buffer used to read status from the NSC A130 Adapter.

Output Function Buffer (NFB@) NIO@OB

This table describes the output function buffer, used to send functions to the NSC A130 Adapter.

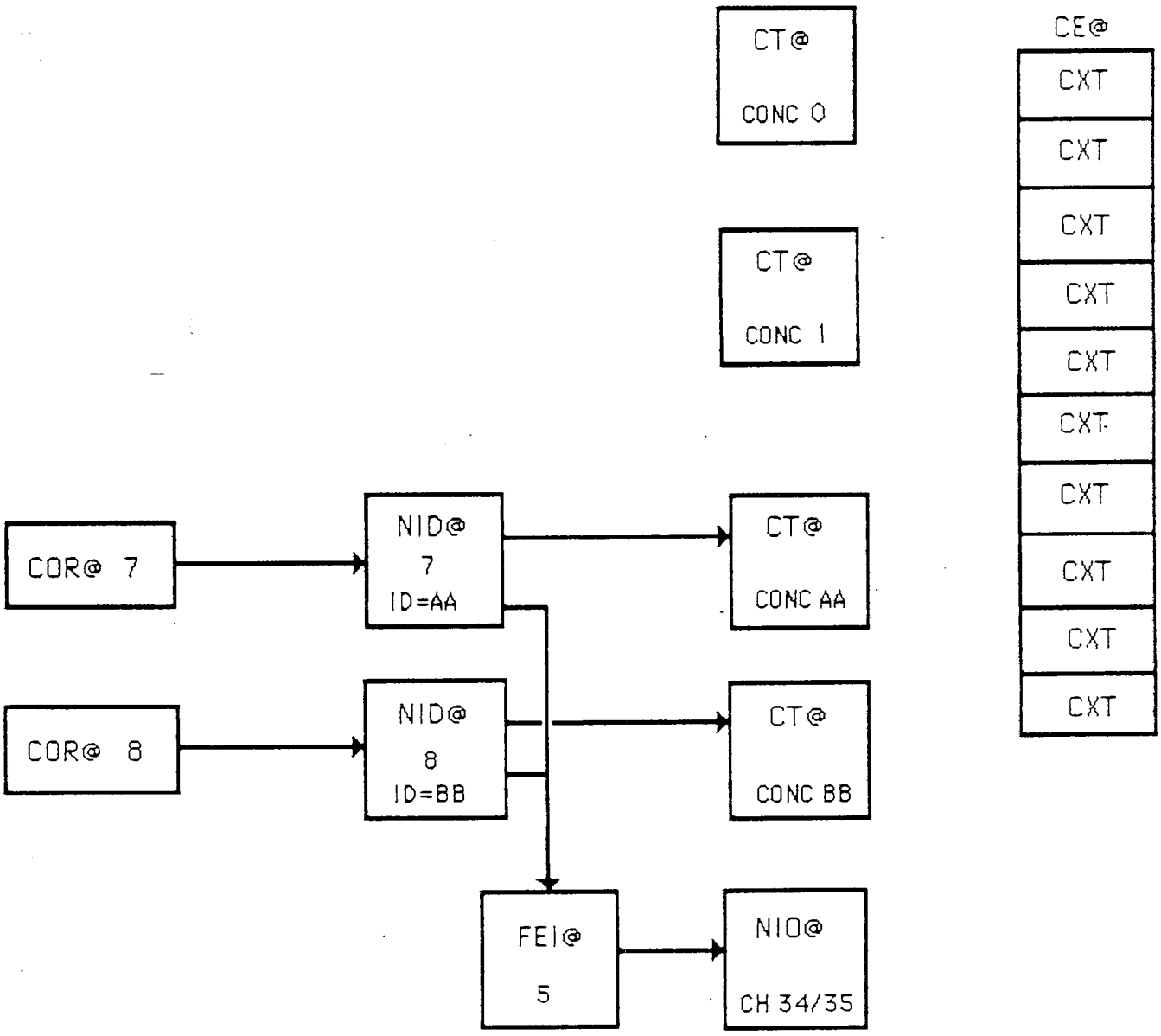
Message Proper (MP@) NIO@MP

This table describes the Message Proper, which is used for communication between trunks on the NSC A130 Adapter.

NSC Front-End ID Table (NID@)

The NSCID tables are created by the NSCIO routine when an NSC channel is initialized. There is one table created for each of the maximum number of front-ends concurrently logged on through all NSC adapter channels.

CONCENTRATOR TABLES



STATION CALL PROCESSOR

Respond to front-end requests

- Streaming
- IO
- Status

Provides for operator guidance

Receiving datasets and jobs

Transmit output dataset

Independent of front-end type

Allocates IO Buffers

- Segment buffer - filled on read from FE channel
- When full call DQM

Recovery from link error and front-end failures

Readied by Exec when IO pair completes on a FE assigned channel

Enters dataset in the SDT

SYSTEM DATASET TABLE

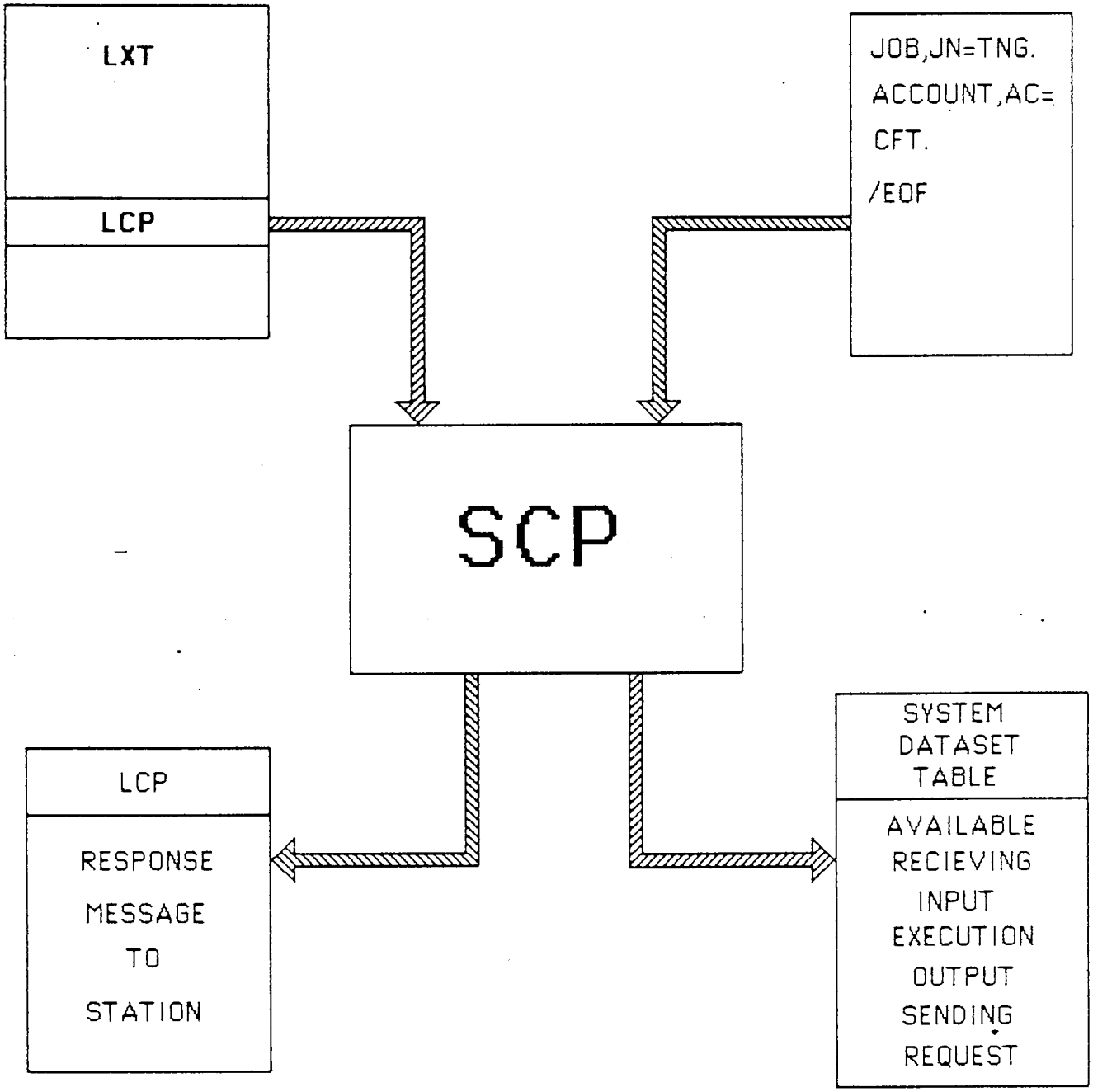
SDT contains information on all datasets that are transferring

- An entry on a queue represents one dataset

Seven Queues

- Available Queue - contains available memory for all SDT
- Input Queue - jobs waiting to be initiated
- Execute Queue - jobs already initiated
- Output Queue - datasets waiting to be sent to front-end
- Sending Queue - datasets in the process of being sent
- Receiving Queue - datasets in the process of being received
- Request Queue - for dataset acquire requests

STATION CALL PROCESSOR



FED DRIVER

Three types of drivers:

IFC Channel Coupler Request Processor
IOP Request Processor
NSC Request Processor

CHT - Channel Table

Contains the task parameter block address

Contains the LIT entry address

Handler address update by the FED after each interrupt

CXT - Channel Extension Table

There is one entry for each I/O Subsystem Ordinal

Passes the message to packet driver

LCT - Link Configuration Table

There is one LCT entry per channel configured for front-end I/O.

The ordinal defines whether the channel is on the CPU or I/O subsystem the type defines the channel's characteristics

The channel may be configured as 'On.' which will simulate a CHANNEL ON operator command during COS startup.

LIT - Link Interface Table

The Link Interface Table is used by both the Station Call Processor and EXEC and contains SCP-EXEC communication areas, working storage, and channel buffers.

An LIT entry is assigned by SCP at deadstart to each channel which is to be used by SCP for link interface communications.

LXT - Link Extension Table

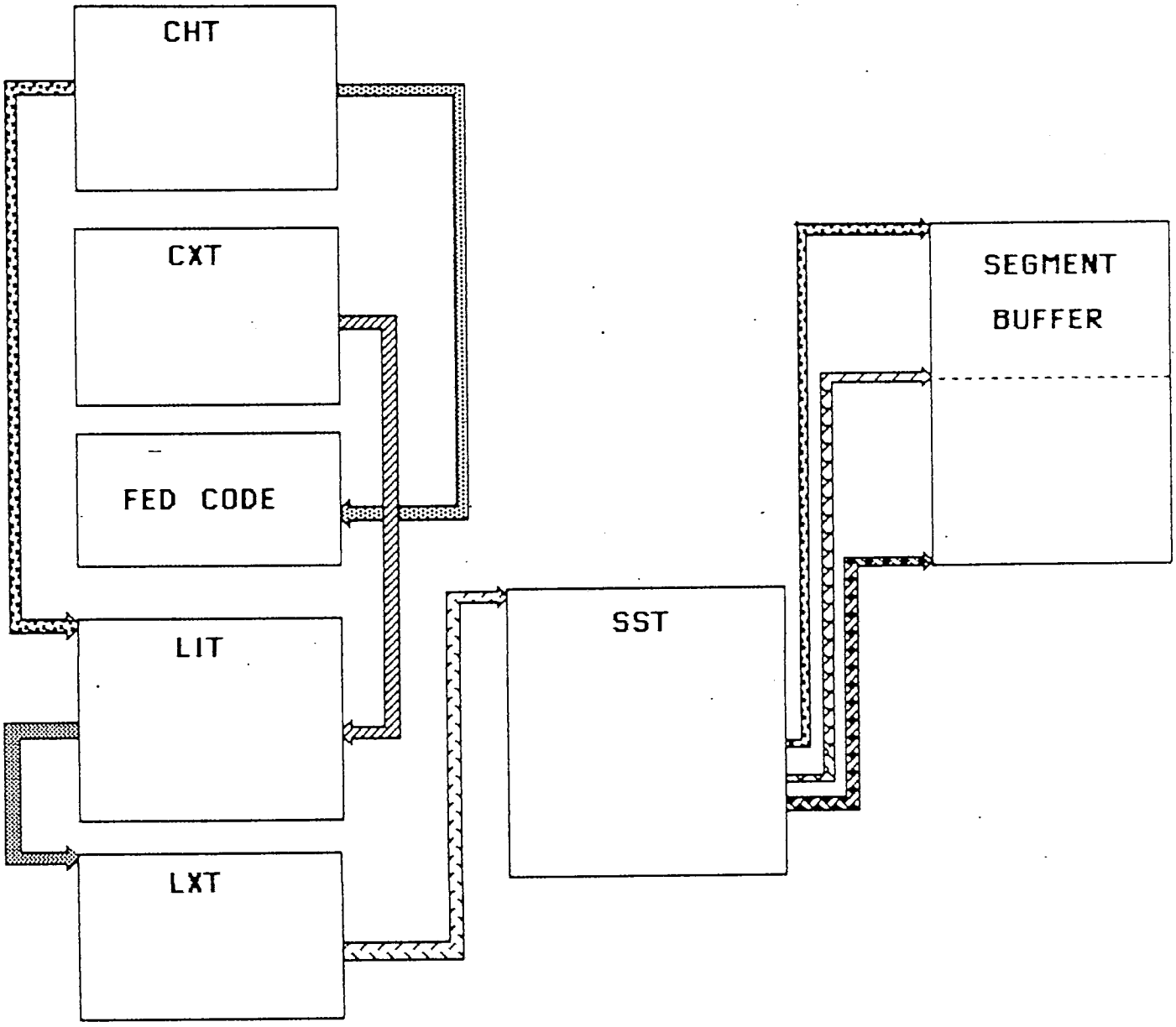
The Link Extension Table (LXT) is used for communication between EXEC and STP. It contains an entry for each configured logical front-end ID; each entry is allocated by FED on receipt of a log on message, and released after an output operation if the OFF bit is set.

Receipt of a front-end message is signaled by FED with INT (interrupt) bit. FED does not modify an entry after setting INT until the next output request is received for that entry.

SST - Stager Stream Table

Manages Segment and Disk Buffers.

EXEC's FRONT-END TABLES



STAGER

Called by SCP

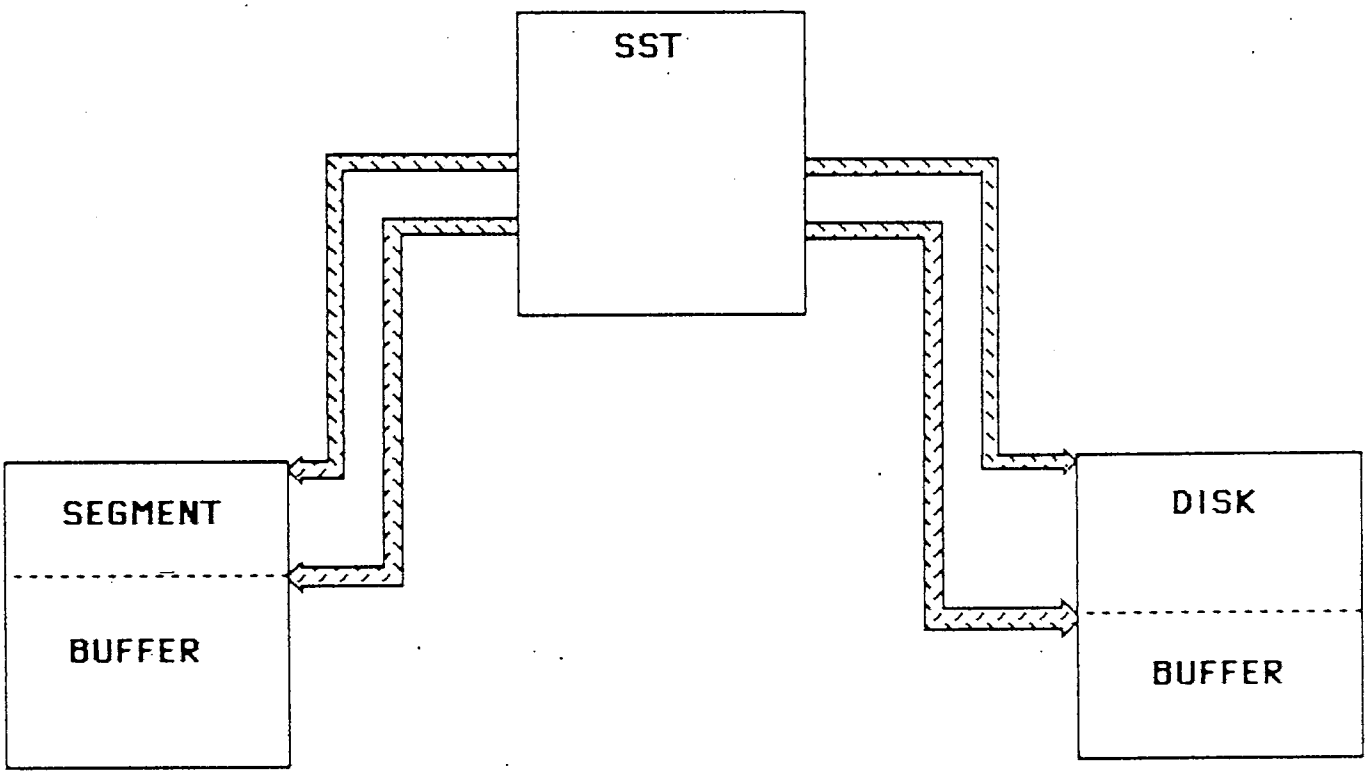
Manages Segment and Disk Buffer

Moves Data between Segment Buffer and Disk Buffer

The Staging Stream Tables are contained in the LXT. It holds information concerning the state of streams for an ID.

SST used to be called the Link Stream Table (LST)

STAGER STREAM TABLE



DISK QUEUE MANAGER - DQM

Manages disk storage request queue

Performs IO request processing

- Preallocate disk space
- Queue IO requests
- Deallocate disk space

Update Device Reservation Table (DRT) for each disk

Readied by another task when needed

Allocation/Deallocation of disk space

- Pre-allocate
- Dynamic Allocation

Disk space is allocated by track

- If specified by request, the Logical Device Name from the DNT is used
- Otherwise DQM rotates among the controllers and disks as specified by the order of EQT

DATASET NAME TABLE

One entry for each dataset

Built when a dataset is made local to a job

Contained in the job table area

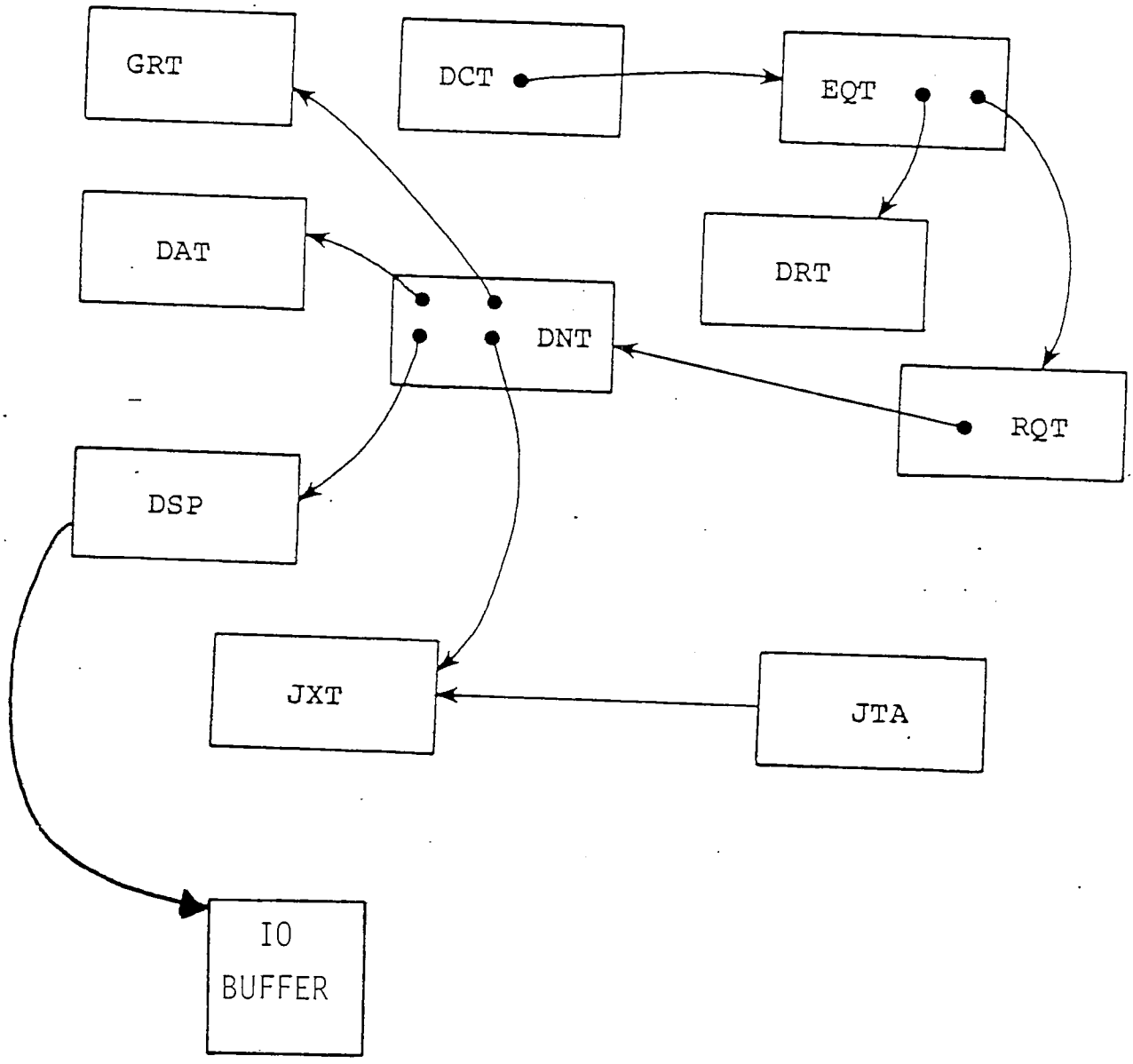
Maintained and used by DQM

Contains the status of the dataset

Points to the datasets' other tables

A system verb has a DNT in the SDR

DATASET NAME TABLE LINKS



DISK DEVICE TABLES

DCT - Device Control Table

Points to current active EQT entry for each channel

DRT - Device Reservation table

One DRT for each device-Disk,SSD,BMR

Contains a bit map indicating which tracks are allocated

Bit positions in the DRT correspond to the allocation index logical track address

Each bit in the bit map represents one allocation unit (AU), such as one track on a disk.

A set bit implies that the AU is in use.

EQT - Equipment Table

One entry per disk unit

Contains status and error information

Contains current request queue

Has the control word for request chain

Points to DRT and IO request queue

RQT - Request Queue Table

Entries are double linked

Points to the DNT for the dataset

Queue header is in EQT entry

One queue chain for each disk

Contains physical IO requests

User requests placed on end of queue

System requests placed second on chain

CNT - Configuration Table

The CNT informs the operating system of the status of on-line tape and disk devices.

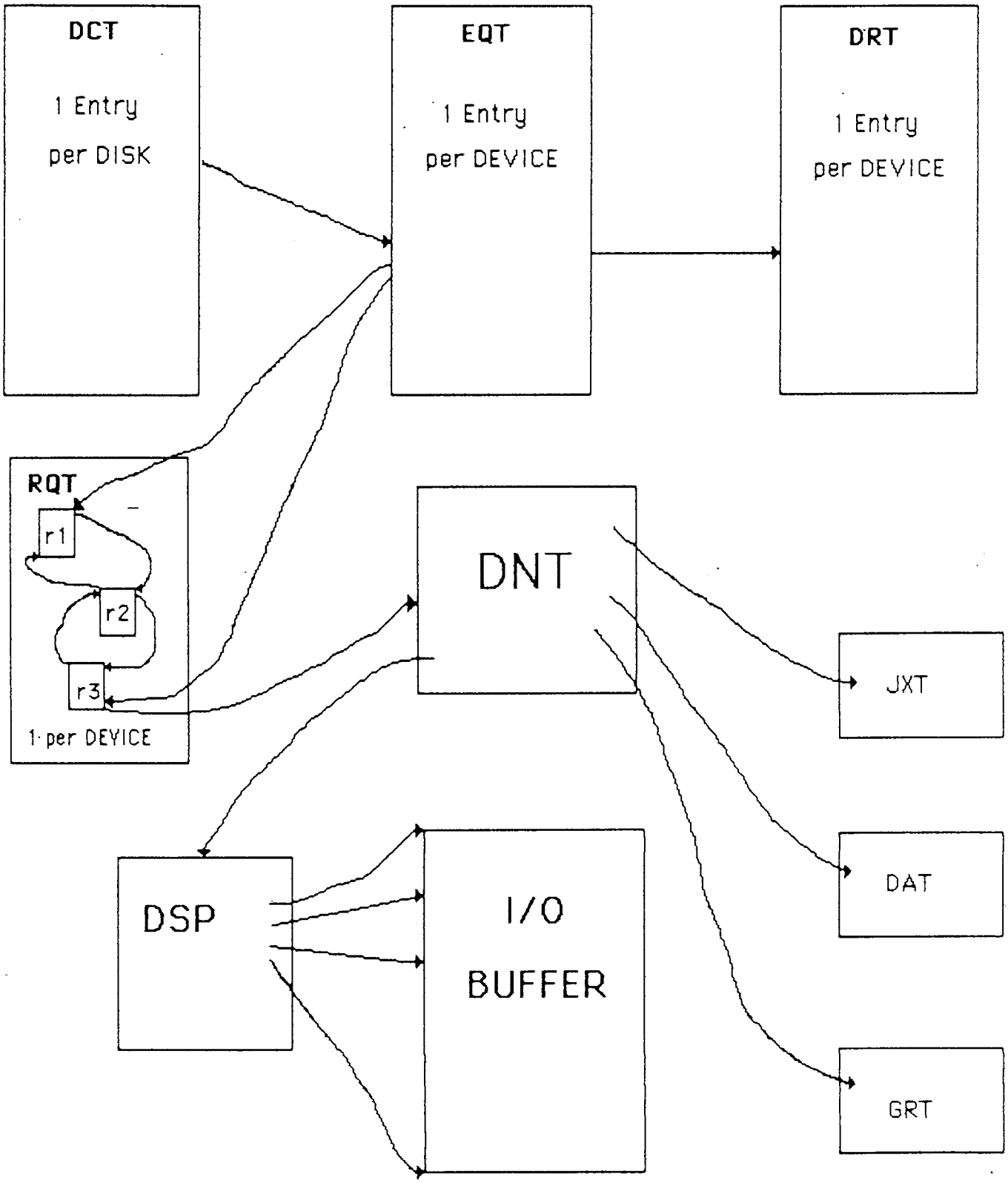
The CNT can be changed during startup by the parameter file or by operator commands.

Each entry, disk or tape occupies 12 words:

A tape entry consists of a 4-word configuration table and from 0 to 4 tape subentries.

A disk entry consists of a 3-word configuration table and 9 words that contain no useful information.

DQM TABLES LINKAGE



OVERVIEW OF A DISK READ THROUGH DIOP (WITHOUT HSP CHANNEL)

1. DQM builds an A packet with the central memory address and disk positional information and the EXEC packet driver sends the packet to IOS.

2. MIOP builds a Master DAL from the packet and writes it to the MIOP message area in buffer memory (CDEM).

3. MIOP sends DIOP an accumulator message with the buffer memory address of the MDAL (CDEM).

4. DIOP reads in the MDAL from buffer memory (ACOM).

5. DIOP builds EDALs for each sector and allocates the local memory and buffer memory disk buffers (DISK).

6. DIOP reads in the sectors and passes the data to the DIOP data buffers in buffer memory (DISK).

7. DIOP changes the EDAL parcel 1 to 4 and writes it to DIOP message area (DISK).

8. DIOP sends BIOP an accumulator message with the buffer memory address of the EDAL (DISK).

9. BIOP reads in the EDAL(ACOM).

10. BIOP reads the DIOP data buffer from buffer memory and sends it over the high-speed channel to the mainframe central memory disk buffer (ACOM).

11. BIOP changes the EDAL parcel 1 to 7 and writes it to DIOP message area in buffer memory (ACOM).

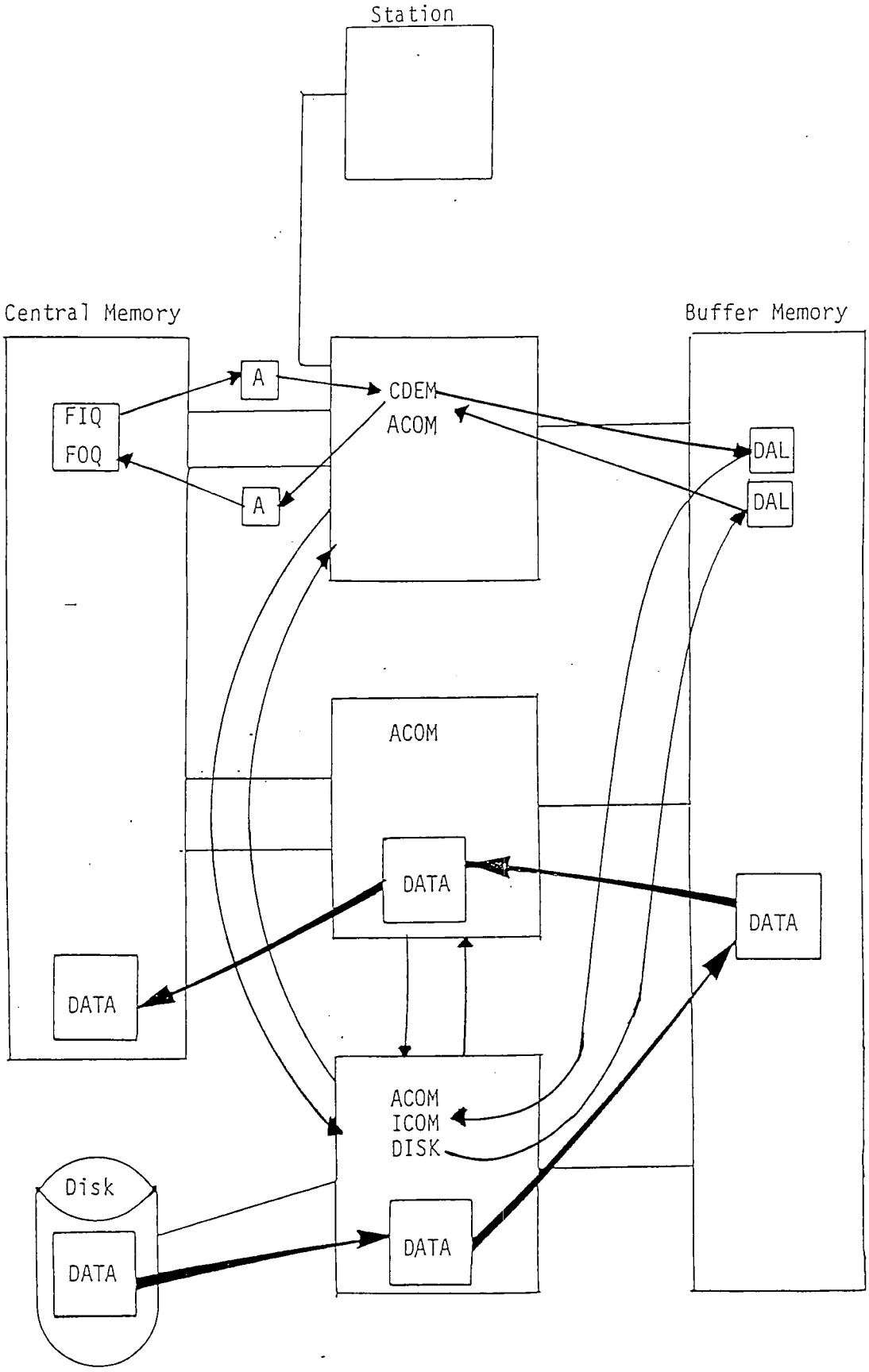
12. DIOP reads in the last EDAL and changes parcel 1 to 5 and sends MIOP the MDAL (DISK).

13. MIOP reads in the MDAL and builds an A packet with parcel 16, byte 1 a Ø (ACOM).

14. MIOP sends the A packet back to EXEC who notifies DQM.

15. DQM reads the A packet for status.

DISK IO


Station
Central Memory
Buffer Memory
A
FIQ
FOQ
A
CDEM
ACOM
DAL
DAL
ACOM
DATA
DATA
DATA
ACOM
ICOM
DISK
Disk
DATA
DATA

DISK SUBSYSTEM TABLES

Buffer Memory Disk Buffer Allocate Table (BBDISK BB@)

This table controls allocation of disk buffers in Buffer Memory. Each processor has a reserved area in Buffer Memory for its buffers.

Disk Control Block Table (DCCB)

The Channel Control Block contains a pointer to a disk control block for each defined disk channel. The channel numbers are in octal. This table is used only by Kernel software executing in the BIOP or the DIOP.

Disk Control Block (DD@/DB@)

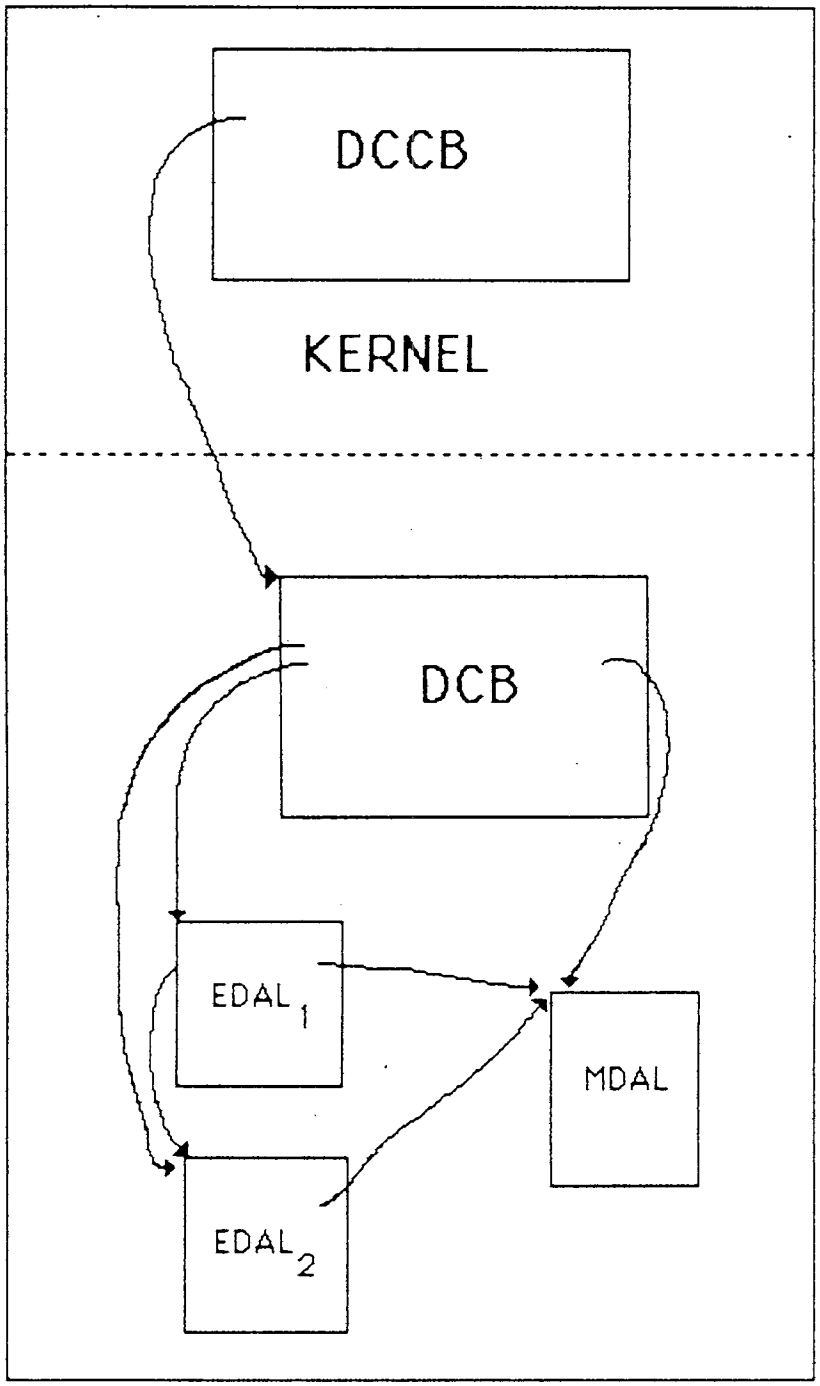
The Disk Control Block (DCB) contains 64_8 parcels of control information for disk operations. Each disk channel that is defined has a Disk Control Block. This table is used only by Kernel software executing in the BIOP or DIOP. The Channel Control Block (DCCB) contains a pointer to each of these tables.

The read-ahead information for a disk unit follows the Disk Control Block (DCB). Six parcels are allocated for each sector to be read. Parcel 12 of the DCB points to the first read-ahead area, and parcel 10 contains a count of the number of areas allocated.

Disk Demon Queue (DISKQ)

The disk demon queue contains the Disk Control Blocks (DCBs) for the disk units requesting services from the disk demon. The queue thread runs through the cell DD@DEM in the DCB. This queue is used only by Kernel software executing in the BIOP or DIOP.

DCB TO DAL LINKAGE



JOB CLASS MANAGER - JCM

Assigns class before entry into the SDT input queue

Determines class of user job based on system resource needs

Determines number of execution entries per job class

Replaces JCL priority with class rank priority

Assigns number of available entries for each class on startup

The job class structure is defined by writing job class structure definition statements and running the JCSDEF utility

See SM-44 for JCSDEF utility

CSD - Class Structure Definition

The job class structure definition is contained in the CSD.

The CSD header, which contains general information about the structure, precedes the class maps.

One CSD class map exists for each class defined in the structure.

Class maps appear in descending rank order.

The variable length characteristic expressions follow the class maps, and each class contains a pointer to its expression.

The CSD class expressions are variable length.

The length of the CSD must be a multiple of 512 words.

STATCLASS

```
CRAY STATION.  VERSION 1.13, IOS.    L S R M                  05/08/84   23:56:14

                                CRAY JOB CLASS STATUS                      FRAME  1

                                     CSDN =  DAY-X

          JOBS IN SYSTEM          14        ACTIVE JOBS            13
          MAX JXTS (LIMIT)         0        AVAIL POOL JXTS    -   67
          DEFINED CLASSES         24        CLASSES WTG JXTS       1

     CLASS        ACTIVE      WAITING      RESERVED      MAXIMUM      STATUS
     FORDLG            0            0             1            2         OFF
     PHILSM            0            0             2            2         OFF
     PHILLG            0            0             1            1         OFF
     TNGSM             0            0             2            6         ON
     TNGMED            0            0             1            3         ON
     TNGLG             0            0             1            2         ON
     OLDIAGS           0            0             3            5         ON
     TEST              0            0             4            4         ON

>+
>SNAP
```

DSC - Dataset Catalog

The DSC resides on disk and is divided into 512-word pages, each page consisting of a block control word, a 7-word header, and eight 63-word entries.

There are two types of pages, hash pages and overflow pages.

The PDN is hashed to determine the hash page number to be searched for a matching or available DSC entry.

If that hash page is full and the function is SAVE or MODIFY, the entry is placed in the sequential overflow page area.

DXT - Dataset Catalog Extension Table

The DXT serves as a repository for information that does not fit conveniently into the DSC.

The DXT is a system dataset similar to the DSC itself. It is located or created during Deadstart; or it can be created during an Install.

It is a permanent dataset with the name $DSC-EXTENSION and edition number 4095.

PDS - Permanent Dataset Table

The PDS is STP resident and contains an entry for each active (accessed) permanent dataset.

A PDS entry indicates how a dataset is accessed and, if multiple access exists, how many users are accessing the dataset.

QDT - Queued Dataset Table

The Queued Dataset Table is an STP-resident table that describes the multitype attributes for a dataset that has been disposed.

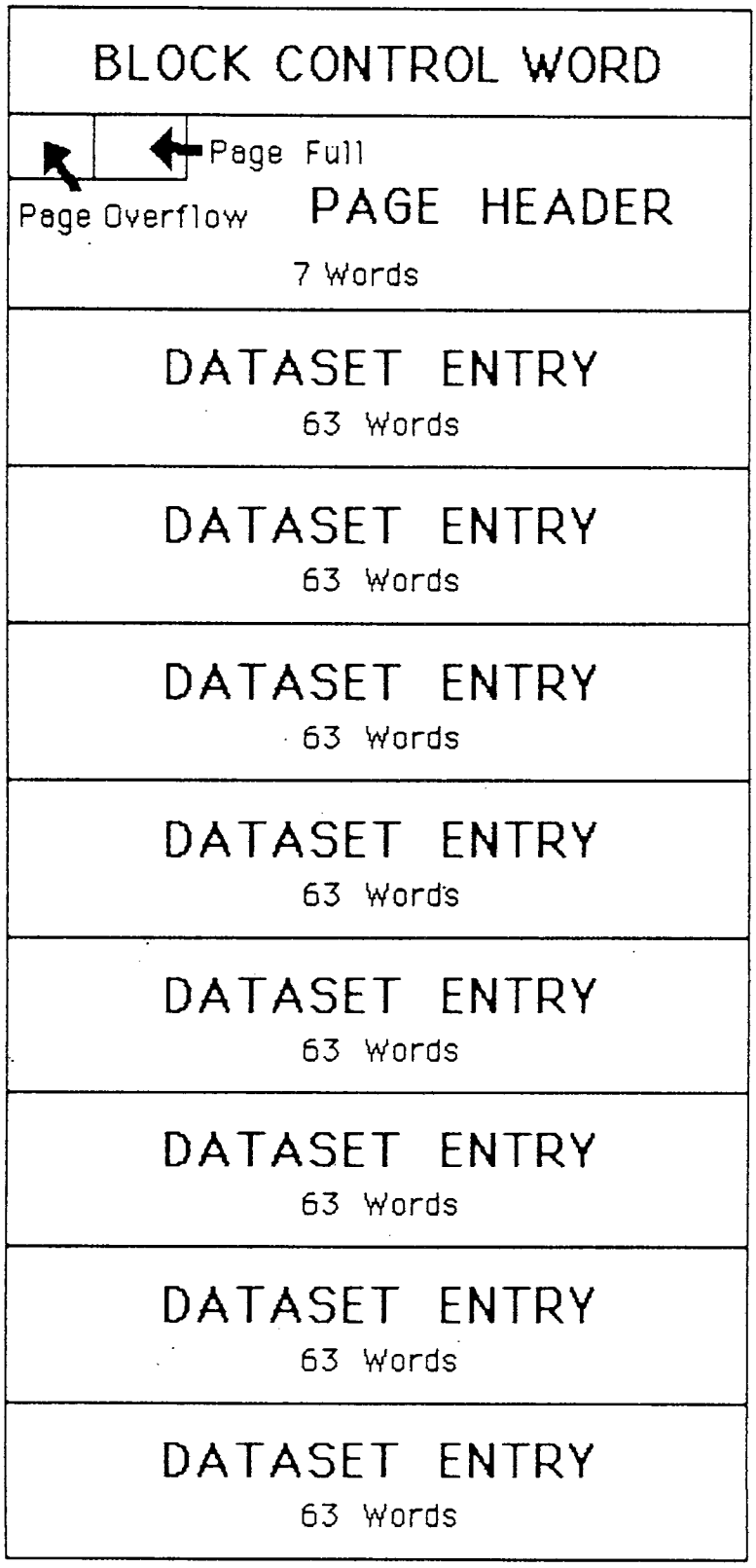
This table is managed by PDM and EXP.

The number of entries in the QDT equals the SDT entry count.

PDD - Permanent Dataset Definition

A PDD is a parameter list that accompanies a Permanent Dataset Management request.

DATASET CATALOG



1 DSC PAGE
EQUALS
512 WORDS
OR
1 DISC SECTOR

JOB SCHEDULER - JSH

Schedules user jobs by priority

Performs memory management

Controls rolling and recovery of job

Saves user registers when rolled

Selects job from SDT

Initiates jobs

Controls jobs in execution, time limit - memory requests

Writes Class Structure Definition CSD

Enters job in JXT by

- Class
- Priority
- Time of submission
- Job priorities may be 0-15
- JXT can contain 63 entries

Zeros out the Job Table Area (JTA) and Job Communication Block (JCB)

Initializes the JCB pointers

Creates the DNT entries for $CS, $LOG, $IN, $OUT in the JTA

Initializes the Roll Job Index (RJI) table entry for possible job recovery.

STATUS

```
CRAY STATION.  VERSION 1.13, IOS.    L S R M                  06/07/84   12:43:35

                                CRAY SYSTEM STATUS

                                  CSDN = TDAY-X

QUEUES E                                                                  FRAME  2

                                                          TIME     FIELD
 JSQ   DC  DATASET   CLASS     STATUS      PRI      USED  LIMIT   LENGTH   ID   TID
------ --  --------  --------  ----------  -------  ----  -----   ------   --  --------
 1995  IN  G$IOLIB  SMALL     EXECUTE     7.0        54   100       94   V3  U1199
 1978  IN  G82DEB   MEDIUM    WAIT-I/O    7.0        17   200      173   DX  00VSPIAA
 1851  IN  IOSSIM   MEDIUM    ROLL-IN     7.0       198   800      273   V3  U1112
 2021  IN  TEST     SMALL     WAIT-SYS    7.0         0    20       37   V3  U0378
 2013  IN  SEG      SMALL     WAIT-I/O    7.0         5    45      220   DX  0UKSAHAA
 2023  IN  SUBSYS   TEST      WAIT-I/O    7.0         0    99       45   V3  UTS
 1897  IN  GENDD43  MEDIUM    WAIT-CPU    7.0       129   700       87   V3  U1314
 1933  IN  PP8      MEDIUM    ROLLED      7.0        25   600      530   V3  U1454
                                 END OF DATA

>
>SNAP
```

JOB ROLLOUT

The Job Scheduler (JSH) task determines a need by a job for user memory and creates a request to have another job in memory rolled-out.

JSH searches the Job Execution Table (JXT) for a candidate for job roll-out.

JSH after locating a candidate may have to compact memory to obtain a large enough free segment of memory.

JSH sets the proper job states of the waiting job and the job about to be rolled-out.

The Disk Queue Manager (DQM) task allocates disk space if needed, and has the memory resident job area written to disk.

The Job Scheduler (JSH) task updates the Memory Segment Table (MST).

The Job Scheduler (JSH) task lifts the suspension on the job waiting for memory liberation.

JOB ROLLING

Memory priority rises while a job is waiting for memory

Memory priority falls when a job is in memory and executing

High priority jobs may pre-empt memory from lower priority jobs which writes the lower priority job to disk (Rolled-Out)

Rolled job priority is rising so that eventually it will be written back to memory (Rolled-in)

Priority Algorithm is tunable

Rolled jobs may be continued or rerun after a restart

Dynamic memory allocation for user

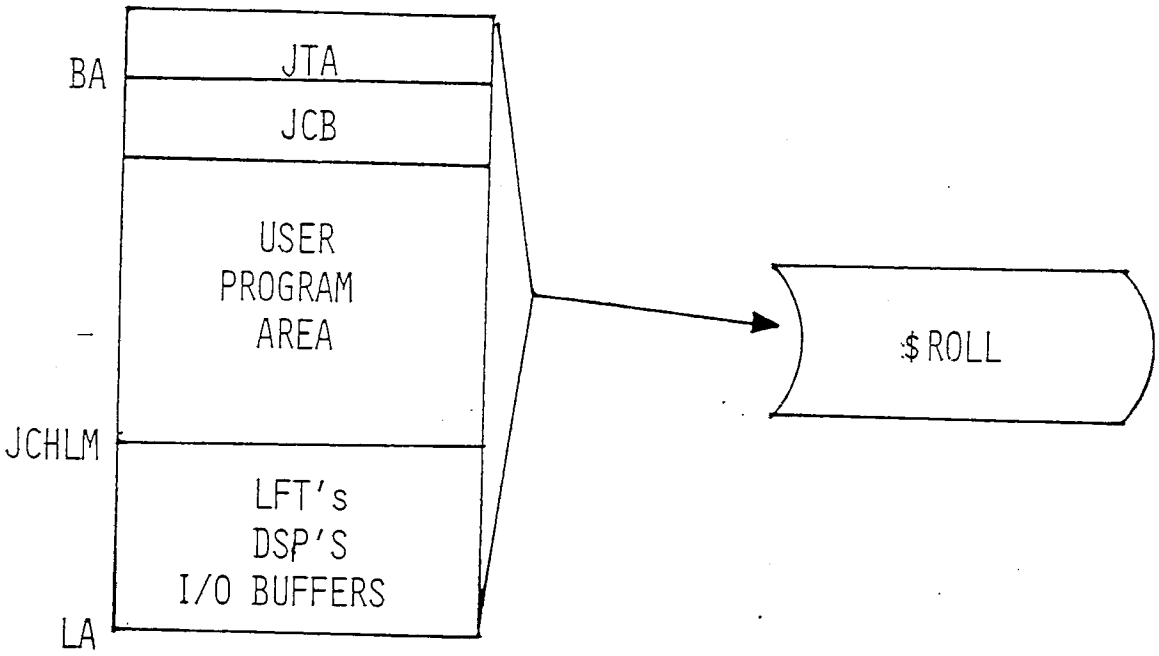
First fit method - allocates memory by block beginning at lower memory

Moves jobs if required, beginning at high end of memory

Encourages large blocks of free memory at the high end of memory

JOB RECOVERY

The entire user area (JTA-LA) is rolled to the disk.



There must be an image of a job on $ROLL in order for the startup routine to recover a job.

JSH TABLES

SDT - System Dataset Table

An SDT entry is created in System Task Processor (STP) resident memory for each dataset that is spooled to or from a front-end system, or submitted for execution by an executing job.

For datasets that are submitted as jobs to the CRAY, the first control statement (the JOB statement) must be cracked to obtain job scheduling information.

JXT - Job Execution Table

The JXT contains current status of job, its location in memory or if rolled and the current priorities of the job.

RJI - Rolled Job Index Table

The RJI table contains entries for each defined JXT entry describing the job assigned to the JXT entry and controlling the recovery of jobs from mass storage entires.

Entry zero issued mainly to validate the roll index dataset.

Other entires indicate which JXT entries have active jobs, and hold information used to locate roll images.

MST - Memory Segment Table

The MST in STP memory contains a 1-word entry for each segment of memory that has been allocated by the Job Scheduler plus additional entries that describe free segments.

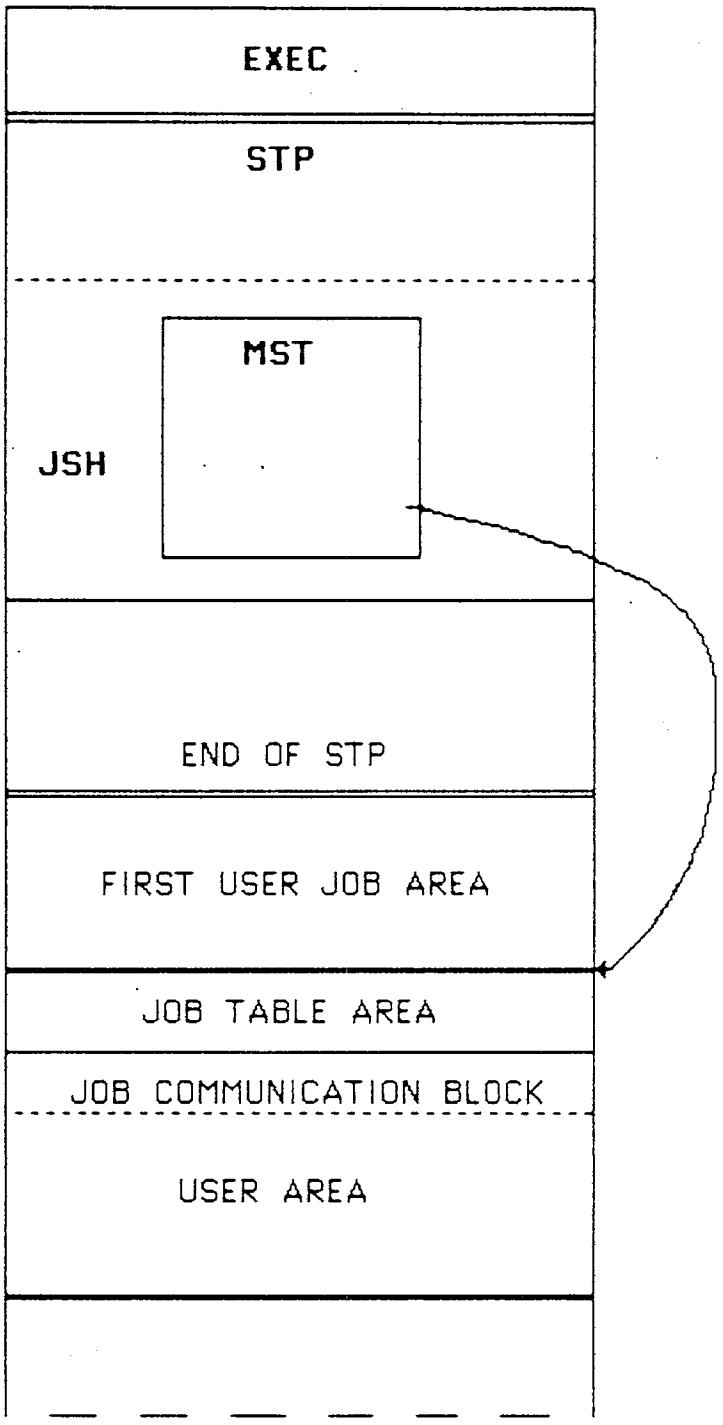
MST entries are stored in ascending order according to the beginning address of the segment (MSADDR).

Any free space between two allocated segments is consolidated and is represented by a single entry stored in the MST between entries for the two allocated segments.

The last entry in the table is always followed by a zero word.

To provide for the case where every allocated segment is surrounded by free segments, the MST must have twice as many words in it as the maximum number of allocated segments, plus two more.

MEMORY SEGMENT TABLE



USER EXCHANGER PROCESSOR (EXP)

Processes all user normal action requests

Processes all user error exits

Processes JSH requests to initiate or abort a user job

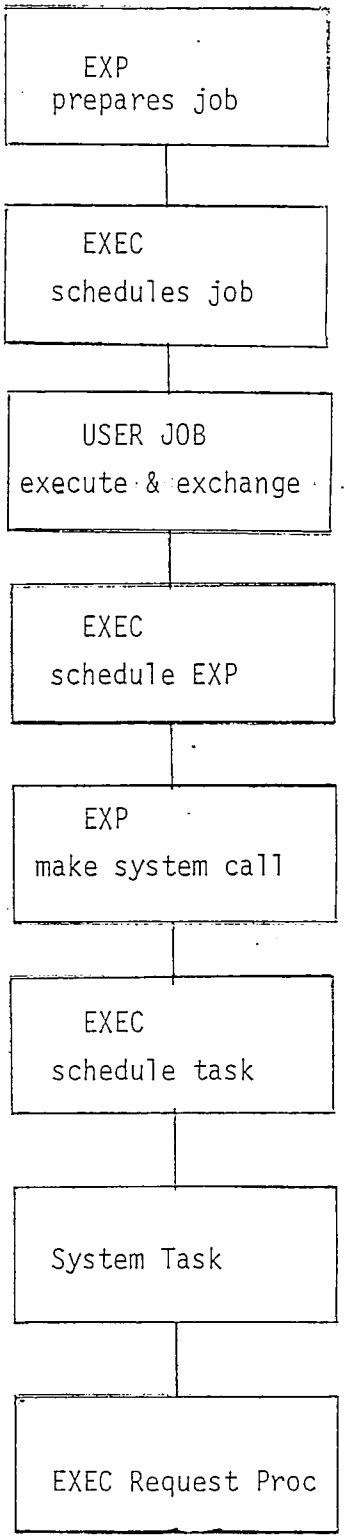
All requests made through the exchange processor request word (JTEP) in a user JTA. JTEP is word 67_8

Copies CSP into user field

Request octal call goes to SØ and then the user executes an Exit

See SM-40, page 8.3 for System Requests

User Exchange Processor



USER AND CSP

Not a task but a routine copied into user areas.

A system program that executes in the user field.

When a job is submitted, JSH calls EXP to copy CSP in User Area BA+200

Makes system requests through a normal exchange.

EXP then processes the exchange and makes the requests to EXEC.

CSP initiates the Job and cracks the JCL statements, processes errors and terminates the job

Uses DSP,LFT and JCB

Makes $IN, $OUT, $CS and $LOG

Under all the same restrictions as the user

Passes messages to $LOG

A copy of CSP resides on disk or after STP

Initiate the job:

1. Copy system bulletin to logfile if it exists and is required by the installation.

2. Enter system header into logfile.

3. Process JOB statement, ending job if any errors encountered. Skip this step if job is interactive.

4. If CSP received recovery station:

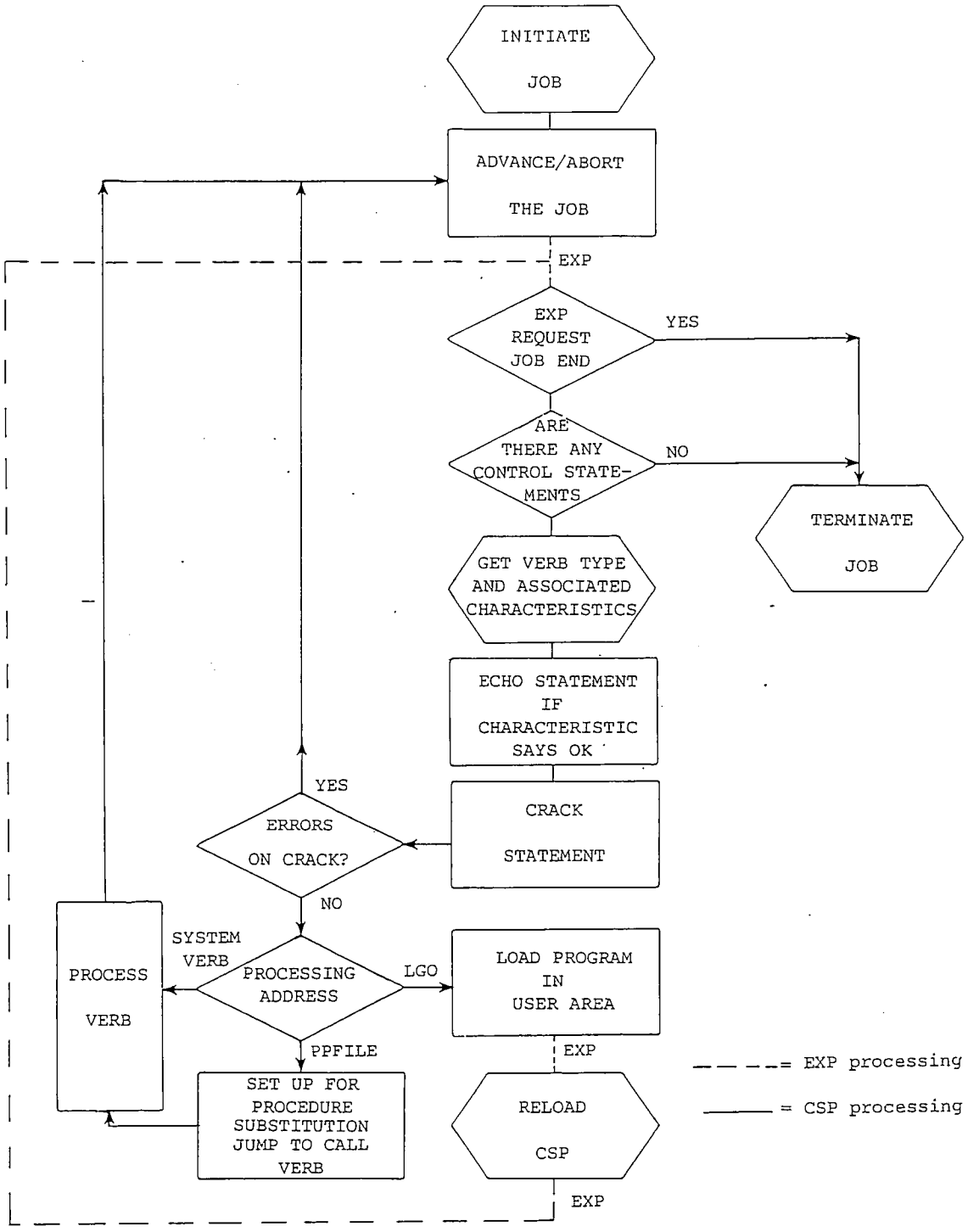
   a. Issue corresponding message to logfile.
   b. Terminate job if status indicates to do so.

5. Validate the structure of $CS if the job is not interactive; terminate the job if any errors found.

6. Assign the datasets $IN and $OUT.

7. Jump to job advancement.

CSP FLOW


INITIATE JOB
ADVANCE/ABORT THE JOB
EXP
EXP REQUEST JOB END
YES
ARE THERE ANY CONTROL STATE-MENTS
NO
TERMINATE JOB
GET VERB TYPE AND ASSOCIATED CHARACTERISTICS
ECHO STATEMENT IF CHARACTERISTIC SAYS OK
CRACK STATEMENT
YES
ERRORS ON CRACK?
NO
SYSTEM VERB
PROCESSING ADDRESS
LGO
LOAD PROGRAM IN USER AREA
PROCESS VERB
PPFILE
SET UP FOR PROCEDURE SUBSTITUTION JUMP TO CALL VERB
EXP
RELOAD CSP
EXP
= EXP processing
= CSP processing

USER AREAS

The user area of the program is a contiguous area of memory. The system appends a Job Table Area (JTA) to the user's BA-LA. Users may access any address between their BA-LA but cannot access out of the area thus making the JTA area unaccessible to a user job, except through system calls.

The JTA contains job related information such as accounting data, a user's $LOG buffer, XP area, TCBs etc; just about anything you would want to know about a job but the coding itself. The JTA is a dynamic area of memory and may expand and contract.

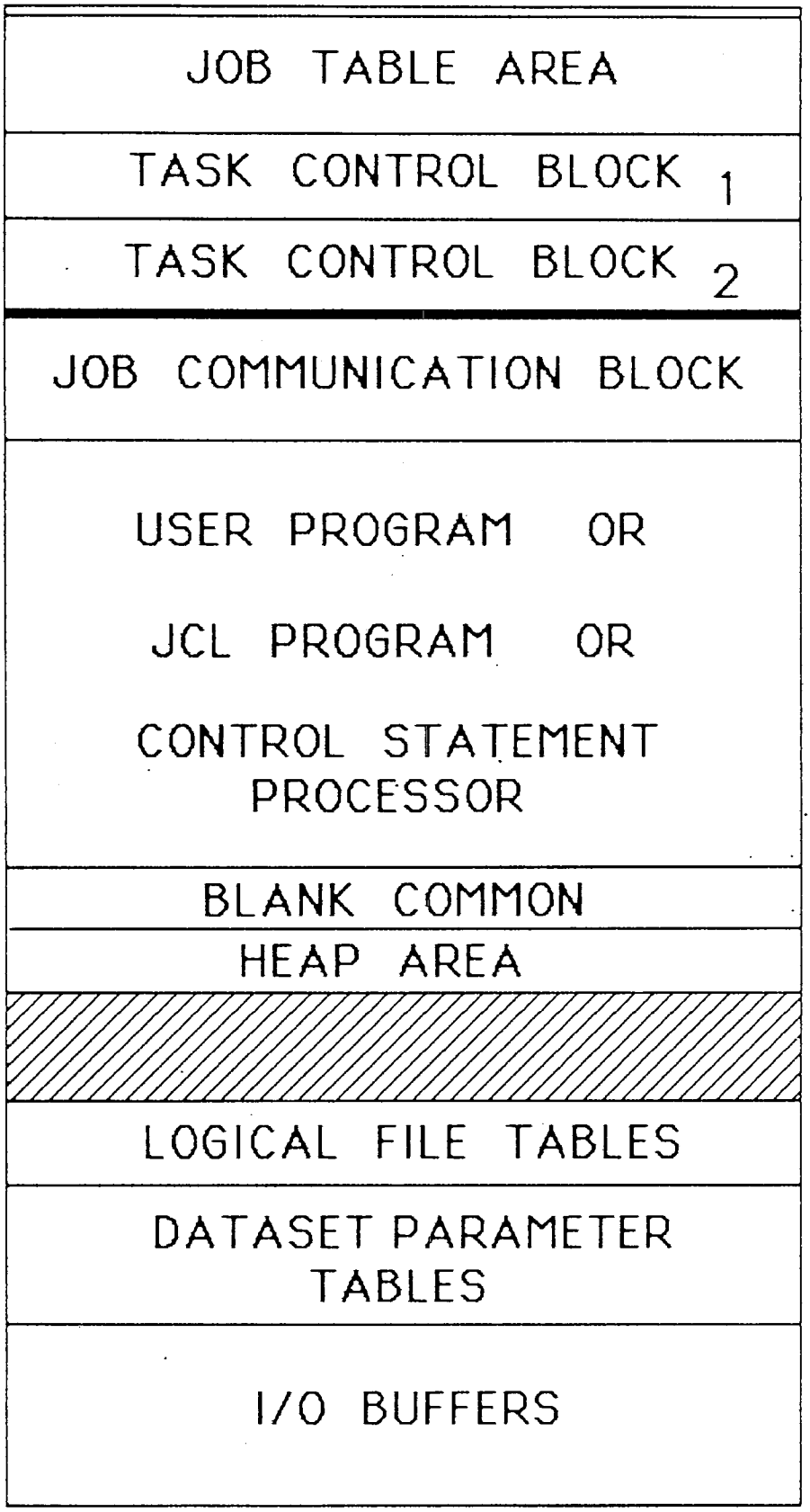
Next in the user area is the job communication block of 128 decimal words. This area is the operating system/job communication area. It contains I/O table pointers, the latest control statement image, COS revision level, etc; the Control Statement Processor (CSP) uses this area for the interpretation of control statements.

Next is the user code/data area. CSP, utilities, compilers and other programs are loaded into this area. Then comes the Heap area that is used when multitasking. The length of the user area is determined by the contents of JCB word HLM. JCHLM is the last address of user code. If a user expands or contracts memory it is at this word in memory.

The area immediately following the user program area is reserved for I/O tables used by the system.

The remainder of the user area (to LA) is used for user I/O dataset buffers. Each buffer is at least 512 words (Block) or multiples thereof. The number of buffers is adjustable by the user.

USER AREA


JOB TABLE AREA
TASK CONTROL BLOCK 1
TASK CONTROL BLOCK 2
USER BA
JOB COMMUNICATION BLOCK
USER PROGRAM OR
JCL PROGRAM OR
CONTROL STATEMENT PROCESSOR
BLANK COMMON
HEAP AREA
JCHLM
LOGICAL FILE TABLES
DATASET PARAMETER TABLES
I/O BUFFERS
USER LA

ACCESS A PERMANENT DATASET

A job makes a request for permission to use a permanent dataset.

The Exchange Package Processor (EXP) task examines the request for valid parameters and issues a request to the Permanent Dataset Manager (PDM) task for access and the Job Scheduler (JSH) task for job suspension.

The Permanent Dataset Manager (PDM) task validates access to the selected dataset for the user job.

PDM manages the necessary system tables so that the user has the desired permissions granted.

PDM creates a Permanent Dataset Table (PDS) entry for the active dataset.

PDM creates a Dataset Name Table (DNT) entry for the user and insures the Dataset Allocation Table (DAT) for the dataset is made resident in the requesting job area thus making the dataset local for that job.

The Job Scheduler (JSH) task sets the jobs status bit to SUSPK thus disallowing the job from roll-out and memory movement until the job's Dataset Allocation Table entry is read into the user area from the Disk Dataset Catalog (DSC).

JSH resets the job's status to the wait state following completion of generation of the local dataset by the Permanent Dataset Manager (PDM) task.

DATASET ALLOCATION

DAT - Dataset Allocation Table

A Dataset Allocation Table defines the mass storage logical location of a dataset by specifying the logical devices and the portions that are used in each device.

There is one DAT for each active dataset in the system.

If the dataset is permanent, the DAT is entered in the catalog and can be used by more than one user.

The DAT is composed of as many 16-word pages as necessary to represent the mass storage occupied by the dataset.

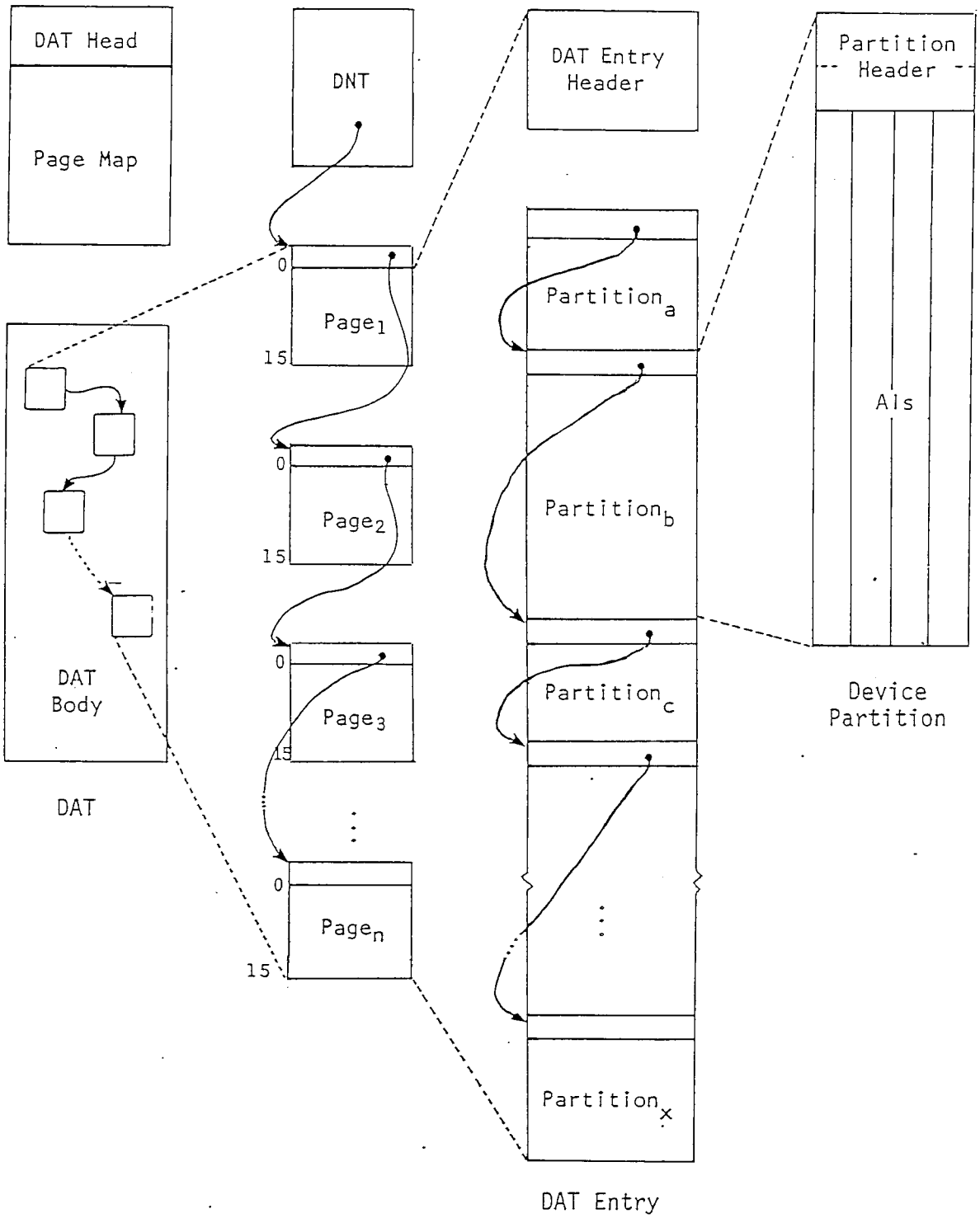
The first page includes a page header, an entry header, at least one partition header, and the first 11 words of the table, which is divided into partitions.

A partition represents a portion of the dataset on a single logical device.

Each page after the first includes a page header and can include partition headers to begin new partitions.

The table consists of a header and a bit map.

DATASET ALLOCATION TABLE



A job issues a write request through a system macro causing physical I/O.

The Exchange Package Processor (EXP) task examines the request for valid parameters and forwards the request for I/O.

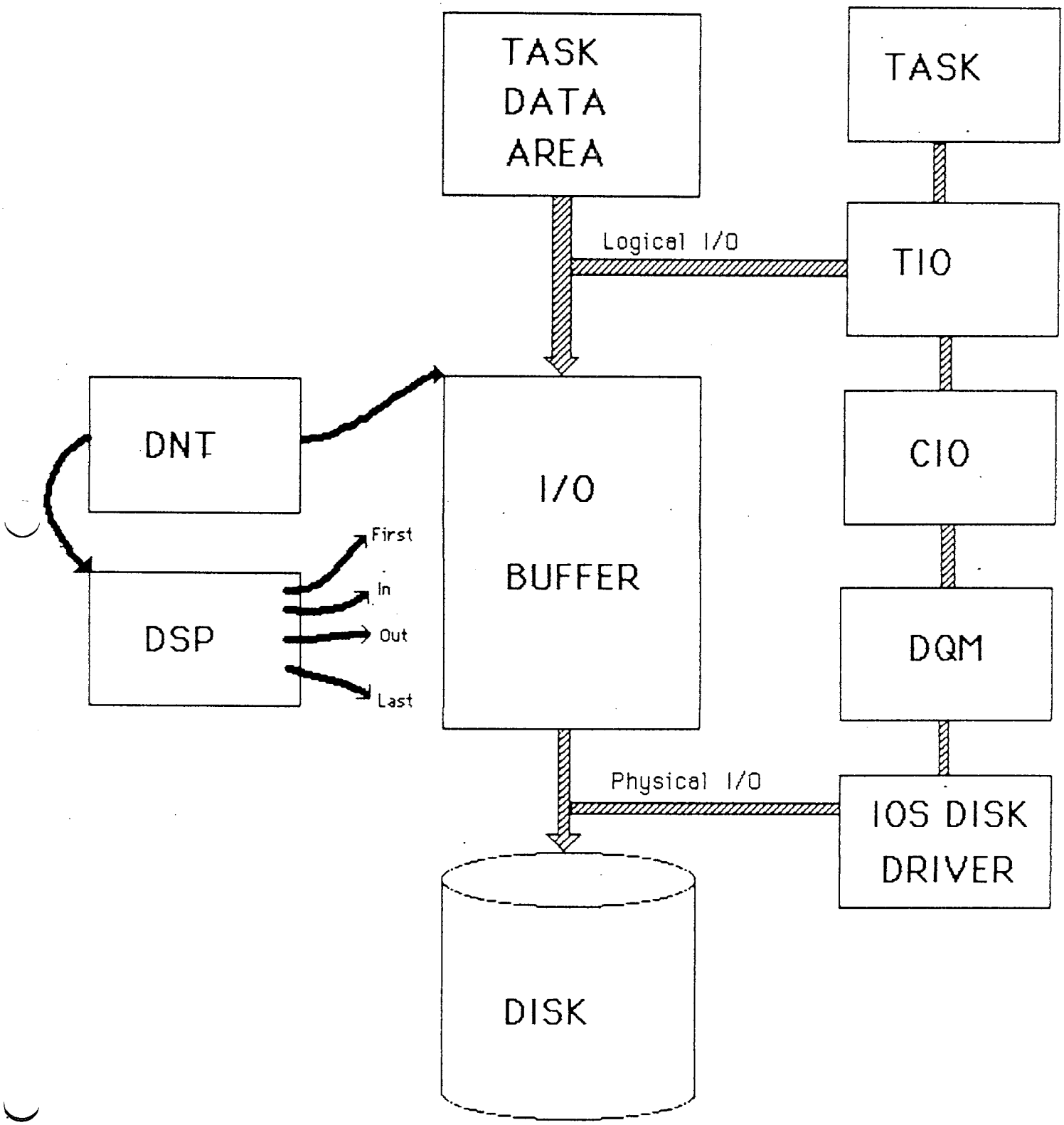
The Task I/O (TIO) routines transfer data, inserting RCWs where needed, from the job data area to the job disk buffers on a write.

TIO determines the disk buffers are greater than half full and calls Circular I/O (CIO).

Circular I/O (CIO) issues a write request of the Disk Queue Manager (DQM) task and has the job suspended by the Job Scheduler (JSH).

The Disk Queue Manager (DQM) task allocates disk on a write if needed.

I/O OVERVIEW


TASK DATA AREA
TASK
Logical I/O
TIO
DNT
I/O BUFFER
CIO
First
In
Out
Last
DSP
DQM
Physical I/O
IOS DISK DRIVER
DISK

TASK I/O ROUTINES

Task I/O (TIO) is a set of reentrant common routines in STP logically considered part of any system task that calls it.

TIO interprets only COS blocked format and therefore, only operates on blocked datasets.

It allows a systems programmer to do logical I/O at the system task level without being concerned about physical I/O.

The following COS system tasks call TIO:

Exchange Processor (EXP)

Startup

Log Manager (MSG)

The logical I/O may be performed on either a dataset related to the system or a user task related dataset.

TIO does not allocate or deallocate any of the control structures or buffers for the request, but assumes all control structures and buffers are set up correctly before the request by the system task.

CIRCULAR I/O ROUTINES (CIO)

Physical I/O on a dataset uses a circular buffering technique initiated by a set of STP common routines known as CIO.

CIO routines are directly callable from system tasks.

The following system tasks directly call CIO within COS:

Exchange Processor (EXP)

Log Manager (MSG)

Permanent Dataset Manager (PDM)

CIO calls either the:

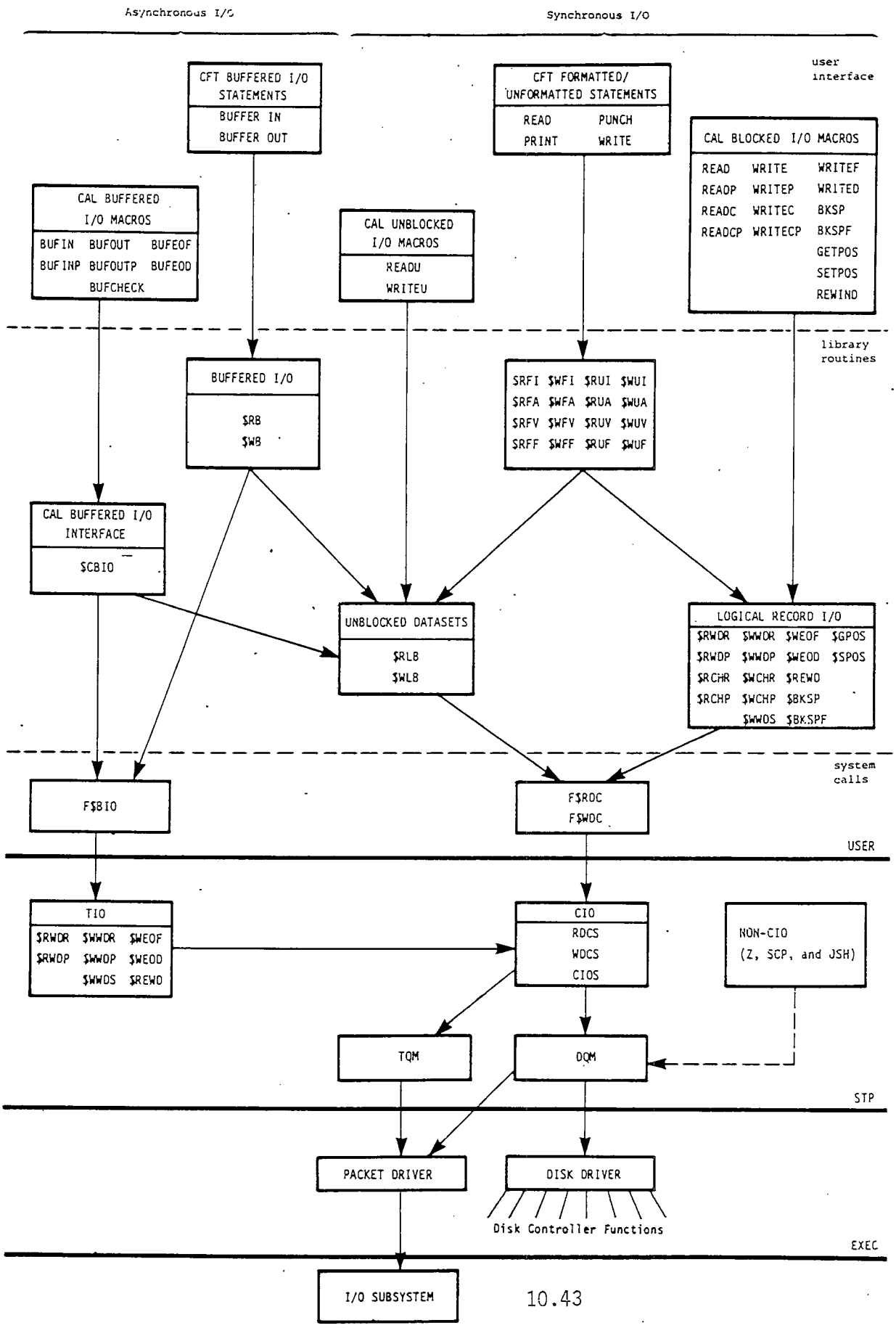
Disk Queue Manager (DQM) or the

Tape Queue Manager (TQM)

to perform physical sector transfers. These calls occur through intertask communication (PUTREQ) from CIO.

These calls are issued by user programs or tasks when data is to be transferred between the I/O buffer defined by the DSP and mass storage.

SYSTEM I/O CALLS



JOB TERMINATION

A job terminates with an exit control statement or EOF interpreted by the Control Statement Processor (CSP).

CSP posts termination messages to the logfiles via the Message (MSG) task.

The Exchange Package Processor (EXP) task copies the jobs $LOG and $OUT.

EXP changes $LOG to scratch and releases scratch datasets as well as $IN, $CS.

The Permanent Dataset Manager (PDM) creates a Dataset Catalog Entry (DSC) for each job output dataset thus making the dataset permanent.

EXP has created entries for the job's output datasets in the SDT.

The Disk Queue Manager (DQM) task allocates disk for $OUT if needed, and has the buffer written to disk via the disk driver.

The Job Scheduler (JSH) task terminates the job by releasing the executing queue entry in the Job Execution Table (JXT).

JSH releases the job's memory and releases the job's dataset and updates the Memory Segment Table (MST).

The Station Call Processor (SCP) task manages system memory buffers and transmits the job's output datasets to the front-end via the front-end driver.

Upon receipt of each output dataset by the front-end, the Station Call Processor (SCP) requests the Permanent Dataset (PDM) to delete the Dataset Catalog (DSC) entry for the dataset and DQM releases the dataset's disk space.

$LOG

```
13:40:14    0.0000   CSP    ------------------------------------------------------------------------------
13:40:14    0.0000   CSP
13:40:14    0.0000   CSP     ***  Use the    NEWS     control statement for general CRAY news. ***
13:40:14    0.0000   CSP
13:40:14    0.0000   CSP    09/27/84 - An archive of datasets not accessed in 14 days was run at
13:40:14    0.0000   CSP               0630, 09/27, on SN201.
13:40:14    0.0000   CSP
13:40:14    0.0001   CSP               **********                                          *********
13:40:14    0.0001   CSP    09/24/84 - ********** DO NOT USE SCATTER/GATHER ON SN201 *********
13:40:14    0.0001   CSP               **********                                          *********
13:40:14    0.0001   CSP
13:40:14    0.0001   CSP             - BATCH on SN201 will end at 2000 on Monday, October 8 and
13:40:14    0.0001   CSP               Tuesday, October 9.
13:40:14    0.0001   CSP
13:40:14    0.0001   CSP
13:40:14    0.0001   CSP    ------------------------------------------------------------------------------
13:40:14    0.0002   CSP            CRAY X-MP SERIAL-201/40    CRI - MENDOTA HEIGHTS, MINN. 10/03/84
13:40:14    0.0002   CSP
13:40:14    0.0002   CSP            CRAY OS - EDITION 183 OF XMs            COS X.14  ASSEMBLY DATE 09/26/84
13:40:14    0.0002   CSP
13:40:14    0.0002   CSP
13:40:14    0.0002   CSP    JOB,JN=U1502A.
13:40:14    0.0009   CSP    ACCOUNT,AC=,US=,UPW=.
13:40:15    0.0059   CSP    CAL.
13:40:15    0.0061   USER   CA001 - [CAL] CAL VERSION X.14 (09/28/84) - CRAY XMP
13:40:15    0.0145   USER   CA034 - [CAL] OPDEF LONGALD  REDEFINED IN BINARY TEXT $SYSTXT
13:40:15    0.2904   USER   CA002 - [CAL] ASSEMBLY TIME:     0.2845 CPU SECONDS
13:40:15    0.2905   USER   CA003 - [CAL] MEMORY WORDS: 84153 + I/O BUFFERS:  6348
13:40:15    0.2910   CSP    LDR.
13:40:16    0.2924   USER   LD000 - BEGIN EXECUTION
13:40:24    8.3742   ABORT  AB023 - JOB TIME LIMIT EXCEEDED
13:40:24    8.3742   ABORT  AB000 - JOB STEP ABORTED.    P = 00000201c
13:40:24    8.3742   ABORT  AB000 - BASE    03044000 LIMIT  03072000 CPU NUMBER   03
13:40:24    8.3742   CSP    END OF JOB
13:40:24    8.3743   CSP
13:40:24    8.3743   CSP
13:40:25    8.3743   USER       JOB NAME -                          U1502A
13:40:25    8.3744   USER       USER NUMBER -                       TNG
13:40:25    8.3744   USER       TIME EXECUTING IN CPU -             0000:00:08.3743
13:40:25    8.3744   USER       TIME WAITING TO EXECUTE -           0000:00:00.4575
13:40:25    8.3744   USER       TIME WAITING FOR I/O -              0000:00:01.3601
13:40:25    8.3744   USER       TIME WAITING IN INPUT QUEUE -       0000:00:00.0152
13:40:25    8.3744   USER       MEMORY * CPU TIME (MWDS*SEC) -              0.14559
13:40:25    8.3745   USER       MEMORY * I/O WAIT TIME (MWDS*SEC) -         0.07759
13:40:25    8.3745   USER       MINIMUM JOB SIZE (WORDS) -          14848
13:40:25    8.3745   USER       MAXIMUM JOB SIZE (WORDS) -          92672
13:40:25    8.3745   USER       MINIMUM FL (WORDS) -                11264
13:40:25    8.3745   USER       MAXIMUM FL (WORDS) -                89088
13:40:25    8.3745   USER       MINIMUM JTA (WORDS) -                3072
13:40:25    8.3745   USER       MAXIMUM JTA (WORDS) -                3584
13:40:25    8.3745   USER       DISK SECTORS MOVED -                  355
13:40:25    8.3745   USER       USER I/O REQUESTS -                    62
13:40:25    8.3745   USER       OPEN CALLS -                           25
13:40:25    8.3745   USER       CLOSE CALLS -                          24
13:40:25    8.3746   USER       MEMORY RESIDENT DATASETS -              0
13:40:25    8.3746   USER       TEMPORARY DATASET SECTORS USED -        1
13:40:25    8.3746   USER       PERMANENT DATASET SECTORS ACCESSED -   96
13:40:25    8.3746   USER       PERMANENT DATASET SECTORS SAVED -       0
13:40:25    8.3746   USER       SECTORS RECEIVED FROM FRONT END -       0
13:40:25    8.3746   USER       SECTORS QUEUED TO FRONT END -           0
```

$LOG

```
13:40:14 0.0000 CSP     ------------------------------------------------------------------------
13:40:14 0.0000 CSP
13:40:14 0.0000 CSP     *** Use the NEWS control statement for general CRAY news. ***
13:40:14 0.0000 CSP
13:40:14 0.0000 CSP     09/27/84 - An archive of datasets not accessed in 18 days was run at
13:40:14 0.0000 CSP                0630, 09/27, on SN201.
13:40:14 0.0000 CSP
13:40:14 0.0001 CSP                ********** **********
13:40:14 0.0001 CSP     09/24/84 - ********** DO NOT USE SCAT|SR/CAINER ON SN201 **********
13:40:14 0.0001 CSP                ********** **********
13:40:14 0.0001 CSP
13:40:14 0.0001 CSP              - BATCH on SN201 will end at 2000 on Monday, October 8 and
13:40:14 0.0001 CSP                Tuesday, October 9.
13:40:14 0.0001 CSP
13:40:14 0.0001 CSP
13:40:14 0.0001 CSP     ------------------------------------------------------------------------
13:40:14 0.0002 CSP     CRAY X-MP SERIAL-201/40     CRI - MENDOTA HEIGHTS, MINN. 10/04/84
13:40:14 0.0002 CSP
13:40:14 0.0002 CSP     CRAY OS - EDITION 183 OF XMS      COS X.14  ASSEMBLY DATE 09/24/84
13:40:14 0.0002 CSP
13:40:14 0.0002 CSP
13:40:14 0.0002 CSP     JOB,JN=U1502A.
13:40:14 0.0003 CSP     ACCOUNT,AC=,US=,UPW=.
13:40:14 0.0059 CSP     CAL.
13:40:15 0.0061 USER    CA001 - [CAL] CAL VERSION X.14 (09/24/84) - CRAY X-MP
13:40:15 0.0149 USER    CA014 - [CAL] OPDEF LDNCALC REDEFINED IN SYMBOL TEXT $SYSTXT
13:40:15 0.3908 USER    CA002 - [CAL] ASSEMBLY TIME:    0.2889 CPU SECONDS
13:40:15 0.2908 USER    CA003 - [CAL] MEMORY WORDS: 84131 + I/O BUFFERS: 6160
13:40:15 0.2916 CSP     LDR.
13:40:16 0.2924 USER    LD000 - BEGIN EXECUTION
13:40:24 8.3742 ABORT   AB023 - JOB TIME LIMIT EXCEEDED
13:40:24 8.3742 ABORT   AB000 - JOB STEP ABORTED.   P = 00000201c
13:40:24 8.3742 ABORT   AB000 - EXIT     00004000 LIMIT 00012000 CPU NUMBER   03
13:40:24 8.3742 CSP     END OF JOB.
13:40:24 8.3743 CSP
13:40:24 8.3743 CSP
13:40:25 8.3743 USER    JOB NAME -                                  U1502A
13:40:25 8.3744 USER    USER NUMBER -                               ENG
13:40:25 8.3744 USER    TIME EXECUTING IN CPU -                     0000:00:08.3743
13:40:25 8.3744 USER    TIME WAITING TO EXECUTE -                   0000:00:02.4575
13:40:25 8.3744 USER    TIME WAITING FOR I/O -                      0000:00:01.3601
13:40:25 8.3744 USER    TIME WAITING IN INPUT QUEUE -               0000:00:00.0150
13:40:25 8.3744 USER    MEMORY * CPU TIME (MWDS*SEC) -              0.14535
13:40:25 8.3745 USER    MEMORY * I/O WAIT TIME (MWDS*SEC) -         0.07752
13:40:25 8.3745 USER    MINIMUM JOB SIZE (WORDS) -                  10944
13:40:25 8.3745 USER    MAXIMUM JOB SIZE (WORDS) -                  99612
13:40:25 8.3745 USER    MINIMUM FL (WORDS) -                        11264
13:40:25 8.3745 USER    MAXIMUM FL (WORDS) -                        89088
13:40:25 8.3745 USER    MINIMUM JTA (WORDS) -                       3072
13:40:25 8.3745 USER    MAXIMUM JTA (WORDS) -                       3584
13:40:25 8.3745 USER    DISK SECTORS MOVED -                        704
13:40:25 8.3745 USER    USER I/O REQUESTS -                         52
13:40:25 8.3745 USER    OPEN CALLS -                                25
13:40:25 8.3745 USER    CLOSE CALLS -                               24
13:40:25 8.3746 USER    MEMORY RESIDENT DATASETS -                  0
13:40:25 8.3746 USER    TEMPORARY DATASET SECTORS USED -            1
13:40:25 8.3746 USER    PERMANENT DATASET SECTORS ACCESSED -        96
13:40:25 8.3745 USER    PERMANENT DATASET SECTORS SAVED -           0
13:40:25 8.3746 USER    SECTORS RECEIVED FROM FRONT END -           0
13:40:25 8.3746 USER    SECTORS QUEUED TO FRONT END -               0
```

Dump Exercises

11

Exercise 1 EXEC & KERNEL Monitor Flow

Skill: To have some familiarity with the COS/IOS monitors

Tasks:

COS

a. Flowchart what occurs in EXEC on an interrupt starting at EN (four pages of code only)

IOS

a. Flowchart what occurs in the KERNEL starting at address INTERRUPT (two pages of code only)

Resources:

SM-0040	
SM-0046	
EXEC listing	pages 116 to 119
KERNEL listing	pages 112 to 113

Related Reading:

SM-0040	pages 2-1 to 2-6
SM-0046	pages 2-1 to 2-15

Intended Lesson Results: To be able to go into the monitors and know the flow through the monitor on an interrupt and how to find your way around the monitor listings

Exercise 1 EXEC & KERNEL Monitor Flow

Skill: To have some familiarity with the COS/IOS monitors

Tasks:

COS

a. Flowchart what occurs in EXEC on an interrupt starting at EN
(four pages of code only)

IOS

a. Flowchart what occurs in the KERNEL starting at address
INTERRUPT (two pages of code only)

Resources:

SM-0040
SM-0046
EXEC listing pages 1-16 to 1-19
KERNEL listing pages 1-12 to 1-13

Related Reading:

SM-0040 pages 2-1 to 2-6

SM-0046 pages 2-1 to 2-13

Intended Lesson Results: To be able to go into the monitors and know the flow through the monitor on an interrupt and how to find your way around the monitor listings

Exercise 2 TASKS AND OVERLAYS

Skill: To have some familiarity with the system programs

Tasks:

COS

a. List the Functions performed by PDM
b. What table is the PDM function request code passed in?

IOS

a. List the Functions performed by ACOM
b. When is ACOM used and by whom ?

Resources:

SM-0040
SM-0046
PDM listing
ACOM listing
KERNEL listing
$APTEXT (optional)
Classroom set of COS and IOS listings

Related Reading:

SM-0040 pages 10-1 to 10-3

SM-0046 pages 3-1 to 3-8

Intended Lesson Results: To be able to locate a system program listing and determine what the program does and who uses it when.

Exercise 2 TASKS AND OVERLAYS

Skill: To have some familiarity with the system programs

Tasks:

COS

a. List the functions performed by PDM
b. What task is the PDM function request code passed in?

IOS

a. List the functions performed by ACOM
b. When is ACOM used and by whom ?

Resources:

SM-0040
SM-0045
PDM listing
ACOM listing
KERNEL listing
SAPTEXT (optional)
Classroom set of COS and IOS listings

Related Reading:

SM-0040 pages 10-1 to 10-3

SM-0045 pages 3-1 to 3-8

Intended Lesson Results: To be able to locate a system program listing and determine what the program does and who uses it when.

Exercise 3 Tasks and Activities

Tasks: COS

a. List all the Tasks in the STT in sequential order
b. Give their priority
c. Give their task ID number
d. Give the task exchange package address

Tasks: IOS

a. List all the Activities in the EACT+1 chain in sequential order
b. Give each activities priority
c. Give each activities storage module address

Resources:

COS DUMP
IOS DUMP
SM-045
SM-0007
$ULTLTXT (optional)
$APTEXT (optional)

Related Reading:

SM-0040	pages	2-1 to 2-14
SM-0046	pages	2-6 to 2-8, 2-11, A1 to A4
SM-0045	pages	470 to 472
SM-0007	page	2-1
$APTEXT	page	89 (optional)
$UTLTXT	page	323 (optional)

Intended Lesson Results: To be able to identify the tasks and activites in the system and at what address their executing enviroments are.

Exercise 3 Tasks and Activities

Tasks: COS

a. List all the Tasks in the TCT in sequential order
b. Give their priority
c. Give their task ID number
d. Give the task exchange package address

Tasks: IOS

a. List all the Activities in the EACT+1 chain in sequential order
b. Give each activities priority
c. Give each activities stack base module address

Resources:

COS DUMP
IOS DUMP
SM-0045
SM-0007
SULTLTXT (optional)
SAPTEXT (optional)

Related Reading:

SM-0040	pages	2-1 to 2-14
SM-0046	pages	2-5 to 2-8, 2-11, A1 to A4
SM-0045	pages	4?0 to 4?2
SM-0007	page	2-1
SAPTEXT	page	88 (optional)
SULTLTXT	page	327 (optional)

Intended Lesson Results: To be able to identify the tasks and activities in the system and at what address their executing environments are.

Exercise 4 System Memory Maps

Skill: To be able to find the address that programs are executing.

Tasks:

COS

a. At what address do the EXEC tables end?
b. At what address does STP begin?
c. At what STP relative addresses are the tasks loaded?
d. At what address does STP end?

IOS

a. At what address do the Kernel tables end?
b. At what address does the Overlay memory chain begin?
c. At what address does the DAL chain begin?
d. At what address does the free memory chain begin?
e. At what address does the IO buffer chain begin?

Resources:

EXEC Listing
KERNEL Listing
COS Dump
IOS Dump
STPTABLES

Related Reading:

SM-0040	pages	1-6 to 1-17
SM-0046	pages	2-1 to 2-3
STPTABLES	pages	20 to 25

Intended Lesson Results: To be able to determine where to find address pointers for relative programs and have some perspective on how Cray system memories are utilized

Exercise 4 System Memory Maps

Skill: To be able to find the address that programs are executing.

Tasks:

COS

a. At what address do the EXEC tables end?
b. At what address does STP begin?
c. At what STP relative addresses are the tasks loaded?
d. At what address does STP end?

IOS

a. At what address do the Kernel tables end?
b. At what address does the Overlay memory chain begin?
c. At what address does the DAL drain begin?
d. At what address does the free memory chain begin?
e. At what address does the IO buffer chain begin?

Resources:

EXEC Listing
KERNEL Listing
COS Dump
IOS Dump
STPTABLES

Related Reading:

SM-0040	pages	1-6 to 1-17
SM-0046	pages	2-1 to 2-3
STPTABLES	pages	27 to 35

Intended Lesson Results: To be able to determine where to find address pointers for relative programs and have some perspective on how Cray system memories are utilized

EXERCISE 5 LCP and SCB Sequence Flow

Skill: To be able to breakdown a SCP-Station LCP event.

Tasks:

a. Fill in the needed LCP fields for the following events

1. RESTART the station after COS has gone down

2. LOGON the station to COS

3. Send a 512-word Job to the Cray input Queue

4. Go to IDLE SCB's

Resources:

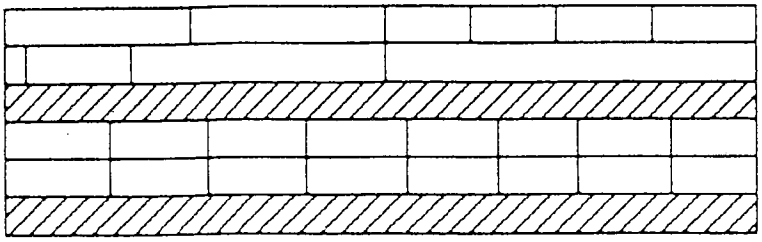
SM-0042
SM-0040
SM-0045
$APTEXT

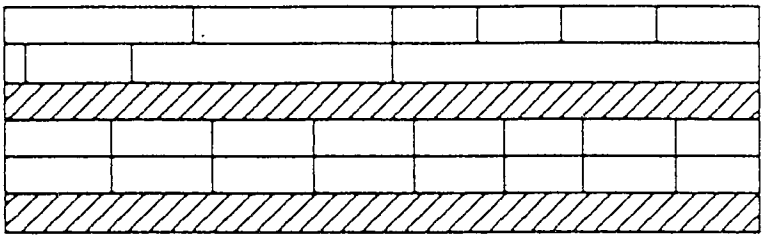
Related Reading:

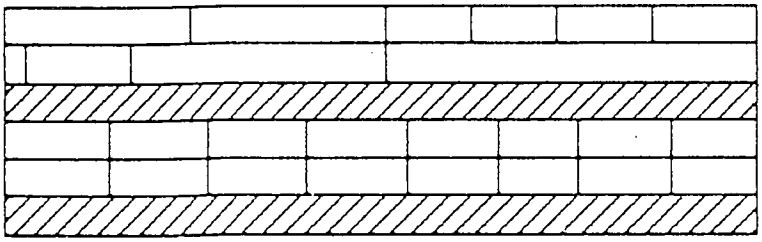
SM-0042	pages	3-1 to 3-4, 4-10 to 4-15
SM-0040	pages	7-1 to 7-8
SM-0046	pages	6-20 to 6-22 (optional)
$APTEXT	pages	140 to 157 (optioanl)

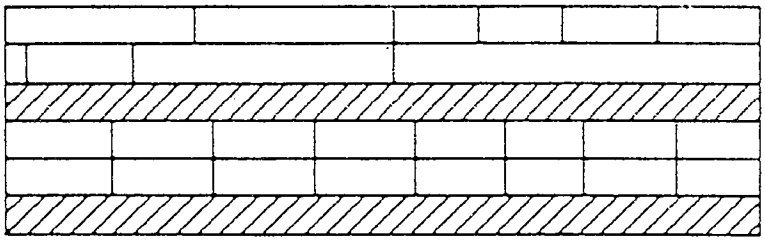
Intended Lesson Results: To familarize yourself with the SM-42 Protocol definitions and understand the software protocol necessary to transfer a job dataset over the station link.

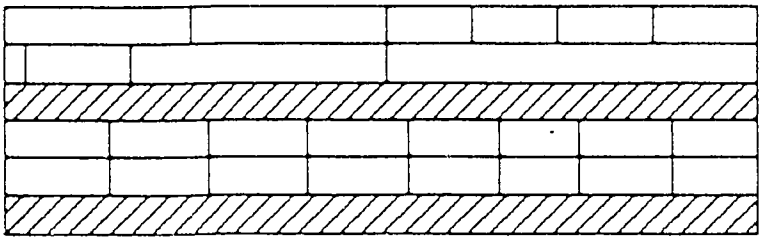
PS. LCP s are recorded in the History Trace Buffer

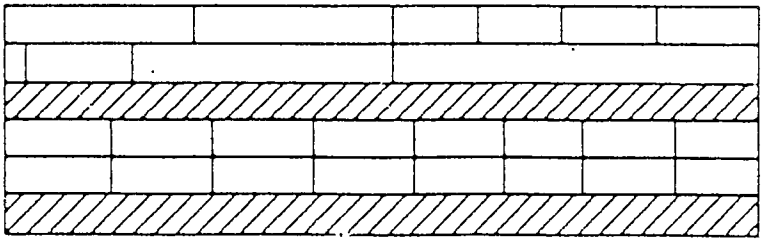


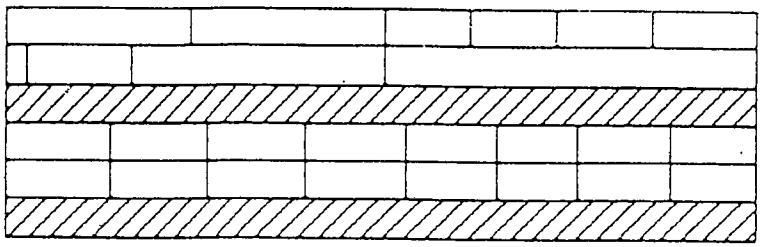


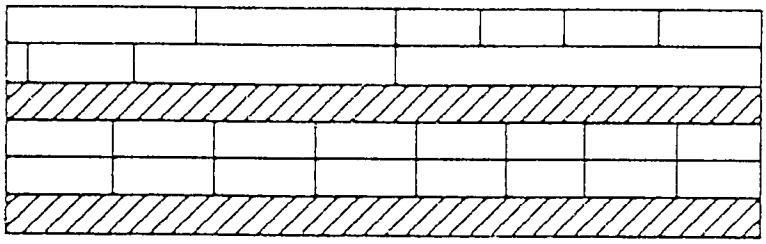


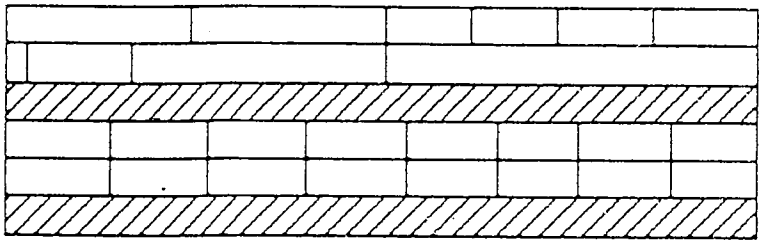


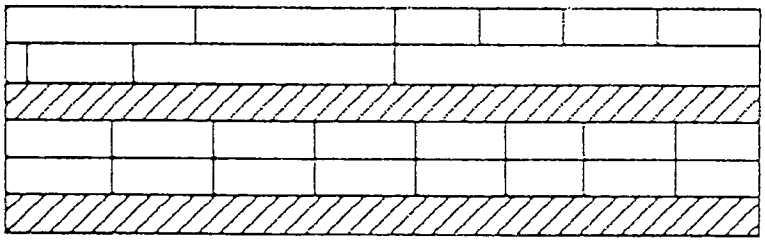


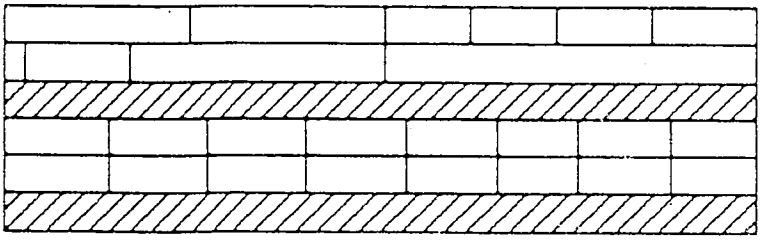


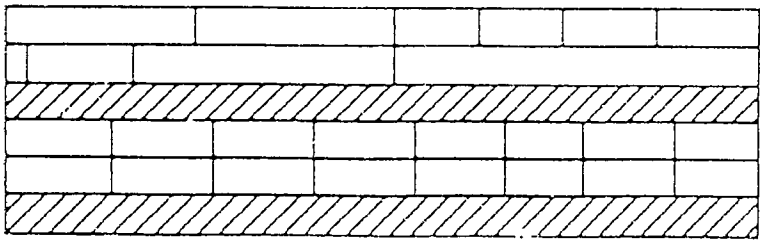


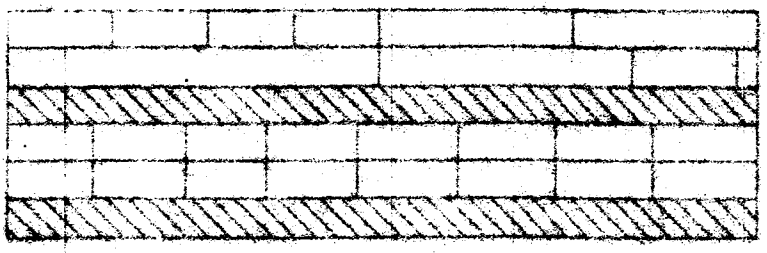


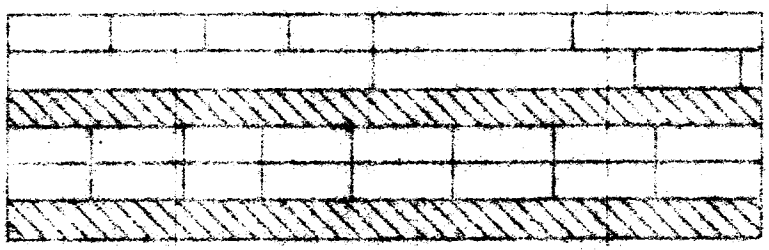


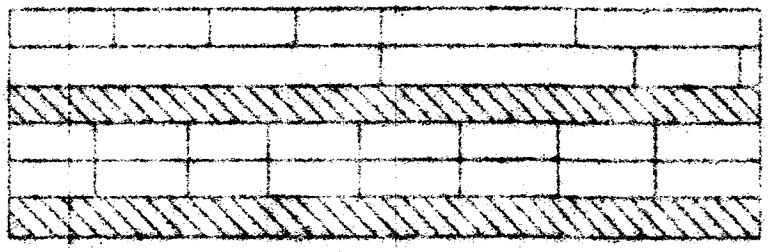


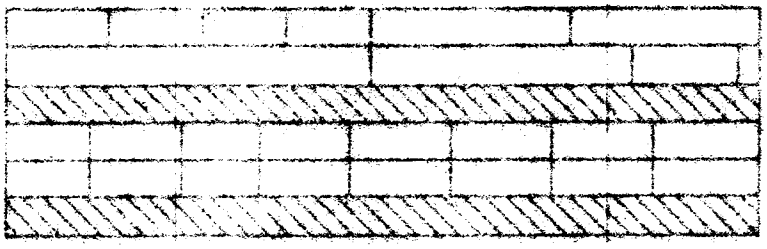


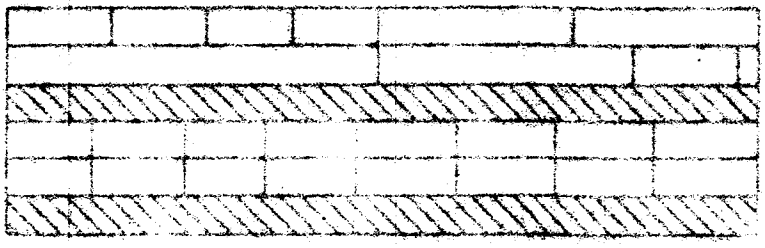


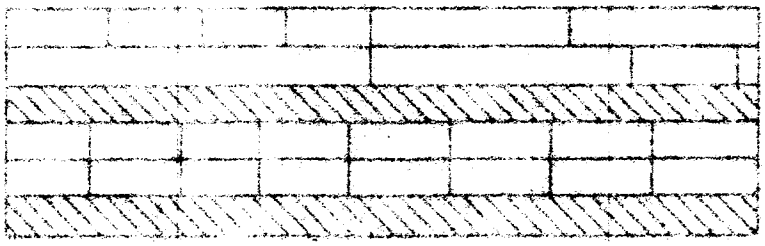












Exercise 6 COS Dump Analysis

Skill: To analyze a COS dump for Failure

Tasks:

a. Locate the Stop buffer

b. Locate the STP hang message if an STP hang

c. Locate the STOP macro

d. Locate the ERR macro if an STP hang

e. Locate the active Task

f. Locate the active task exchange package

g. Analyze the EXEC history trace events

h. Analyze the ITCT intertask communication history trace events

Resources:

SM-0040
SM-0045
COS Listings
COS Dump

Related Reading:

SM-0040 pages 2-74 to 2-95

Intended Lesson Results: To be able to find out which program halted COS and study the events that lead up to the halt and why it halted.

Exercise 6 — CDS Dump Analysis

Skill: To analyze a CDS dump for failure

Tasks:

a. Locate the Stcp buffer

b. Locate the STP hang message if an STP hang

c. Locate the STOP macro

d. Locate the ERR macro if an STP hang

e. Locate the active Task

f. Locate the active task exchange package

g. Analyze the EXEC history trace events

h. Analyze the ITCT intertask communication history trace events

Resources:

SM-0040
SM-0045
CDS Listings
CDS Dump

Related Reading:

SM-0040 pages 2-74 to 2-95

Intended Lesson Results: To be able to find out which program halted CDS and study the events that lead up to the halt and why it halted.

Exercise 7 IOS Dump Analysis

Skill: To Analyze an IOS dump for failure

Tasks:

a. Locate the hang message in the punted IOP

b. Locate the PUNTIF macro

c. Analyze the history trace table events

d. Locate the active overlay number, name and activity descriptor

e. Locate the active SMOD (ESMD)

f. Locate the relative P register of the active programs SMOD

g. Determine what operand registers have useful overlay parameter registers

Resources:

IOS Dump
IOS Listings
SM-0046
SM-0007

Related Reading:

SM-0046 pages 11-4 to 11-17
Appendix A

Intended Lesson Results: To be able to find out which program halted IOS and the events leading to the halt and why it halted.

Exercise 7 IOS Dump Analysis

Skill: To Analyze an IOS dump for failure

Tasks:

a. Locate the hang message in the punted IOP

b. Locate the PUNT.F macro

c. Analyze the history trace table events

d. Locate the active overlay number, name and activity descriptor

e. Locate the active SMOD (ESMOD)

f. Locate the relative P register of the active programs SMOD

g. Determine what operand registers have useful overlay parameter registers

Resources:

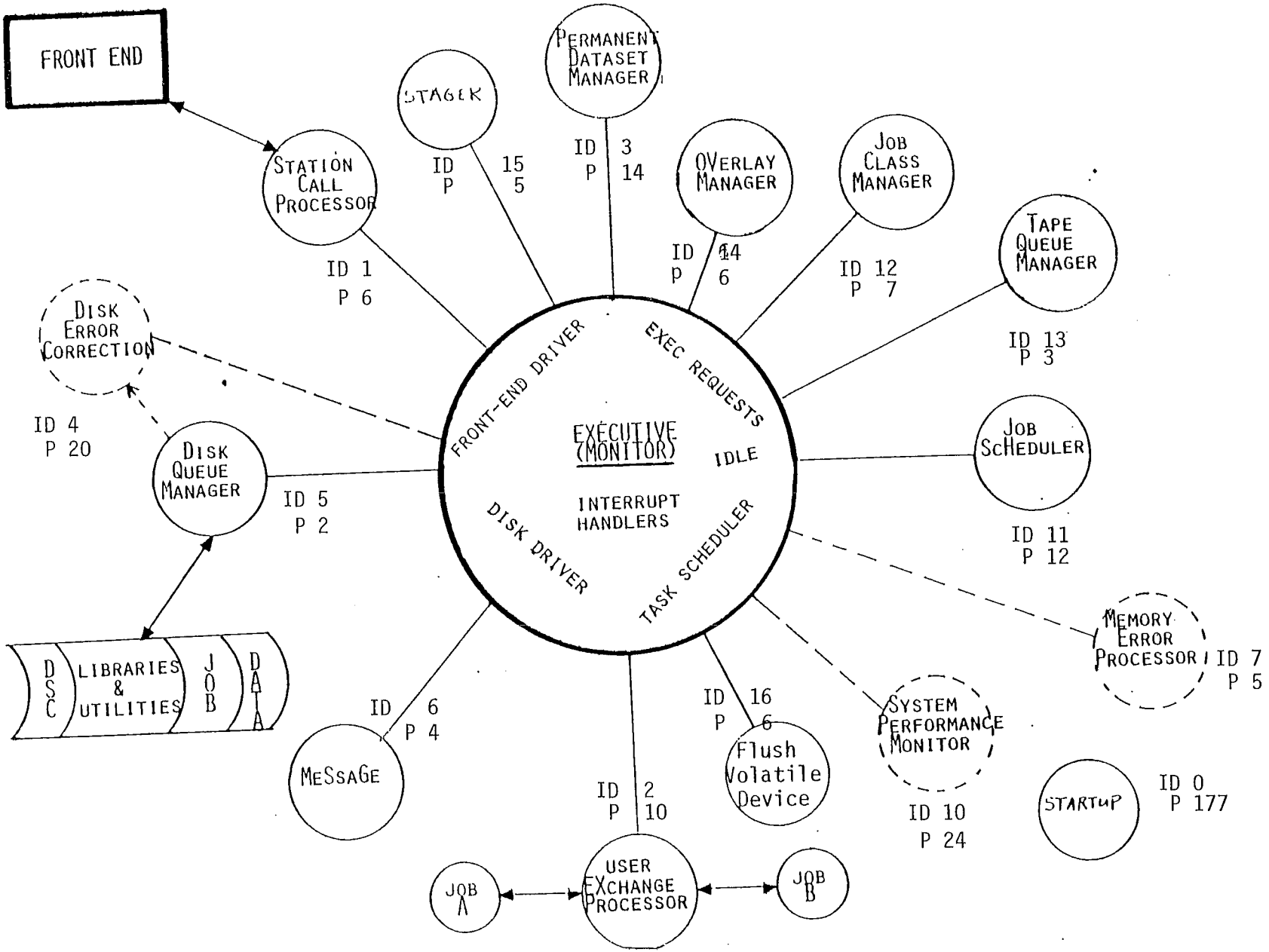
IOS Dump
IOS Listings
SM-0046
SM-0047

Related Reading:

SM-0046 pages 11-4 to 11-17
Appendix A

Intended Lesson Results: To be able to find out which program halted IOS and the events leading to the halt and why it halted.

Appendix A - Quick References


FRONT END
STATION CALL PROCESSOR
ID 1
P 6
STAGER
ID 15
P 5
PERMANENT DATASET MANAGER
ID 3
P 14
OVERLAY MANAGER
ID 14
P 6
JOB CLASS MANAGER
ID 12
P 7
TAPE QUEUE MANAGER
ID 13
P 3
DISK ERROR CORRECTION
ID 4
P 20
DISK QUEUE MANAGER
ID 5
P 2
FRONT-END DRIVER
EXEC REQUESTS
EXECUTIVE (MONITOR)
IDLE
INTERRUPT HANDLERS
DISK DRIVER
TASK SCHEDULER
JOB SCHEDULER
ID 11
P 12
MEMORY ERROR PROCESSOR
ID 7
P 5
DSC
LIBRARIES & UTILITIES
JOB
DATA
MESSAGE
ID 6
P 4
ID 16
P 6
Flush Volatile Device
SYSTEM PERFORMANCE MONITOR
ID 10
P 24
STARTUP
ID 0
P 177
ID 2
P 10
USER EXCHANGE PROCESSOR
JOB A
JOB B

OVERLAY DESCRIPTIONS C

This appendix describes the overlays used by I/O Subsystem software. The list is alphabetical according to overlay name. Each overlay has one or more of the following type codes.

Type	Description
D	Data overlays. These overlays contain data and cannot be executed.
K	Kernel overlays
F	File system overlays associated with the I/O Subsystem Editor. See Appendix F of the I/O Subsystem (IOS) Operator's Guide, CRI publication SG-0051, for a description of the capabilities of the Editor.
C	Concentrator overlays
S	Station overlays
I	Interactive station overlays
B	Block multiplexer overlays. These overlays drive the block multiplexer channels.
T	Tape overlays
R	System dump and restart overlays
Ns	NSC (Network Systems Corporation) channel concentrator overlays

KERNEL HALT (PUNT) CODES

CODE	MEANING
ØØØ	No error code specified on $PUNTIF macro
ØØ1	Local memory error (always HARDWARE)
ØØ2	Buffer memory error on deadstart (always HARDWARE)
ØØ3	Buffer memory error (always HARDWARE)
ØØ4	High-speed channel error (always HARDWARE)
ØØ5	Invalid message received from CPU
ØØ6	Invalid parameter in disk request from CPU
ØØ7	Program was executing at location Ø
Ø1Ø	Local memory location Ø was overwritten
Ø11	Undefined message received on IOP communication channel
Ø12	Overlay does not exist
Ø13	Station stack overflow or underflow
Ø14	Local memory buffer not available
Ø15	Buffer memory disk buffer not available
Ø16	Invalid local buffer release call
Ø17	Buffer memory incorrectly configured
Ø2Ø	IOP message channels incorrectly configured
Ø21	SMOD is too large for area on buffer memory
Ø22	Invalid local memory address
Ø23	Illegal interrupt program sequence code
Ø24	Stop request received from CPU
Ø25	Low-speed channel error (always HARDWARE)
Ø26	Block number validation trap
Ø4Ø	Block Mux interrupt processor error
Ø41	Bad CRW address in device table
Ø42	Block mux start I/O error
Ø44	Block Mux configuration error
Ø5Ø	DD49 disk software error
Ø51	The debugger was not loaded
Ø52	Bad buffer memory allocation request
Ø53	Bad local memory address on hi speed I/O call
Ø54	Invalid I/O length specified
Ø55	No hi speed channel configured for request
Ø56	Illegal activity activation requested
Ø57	Buffer memory DAL queue exhausted
Ø6Ø	Illegal Demon call
Ø61	Undefined kernel service request
Ø62	Illegal kernel service request
Ø63	Bad kernel service request parameter
Ø64	Requested device not configured
Ø65	Illegal IOP requested on kernel service request
Ø66	Requested queue full
Ø67	Illegal I/O address specified
Ø7Ø	SMOD error
Ø71	Local memory space exhausted
Ø72	Illegal overlay load requested
Ø73	Corrupted local memory chain
Ø74	Bad local memory release
Ø75	Bad I/O parameter
Ø76	Unexpected interrupt received
Ø77	Disk error
1ØØ	AMAP not available for system initialization
1Ø1	Illegal overlay number read during initialization
1Ø2	IOP initialization error
1Ø3	MOS configuration error
1Ø4	Overlay too large for loading
1Ø5	Premature tape EOF encountered
1Ø6	Exit stack fault
1Ø7	Illegal device type configured

No.	Ovl Name	Size	BM-Upr	BM-Lwr
ØØ	SDMPA	111Ø	Ø1	Ø21373
Ø3	SDMPD	Ø24Ø	Ø1	Ø22Ø23
Ø6	AMAP	Ø41Ø	ØØ	Ø2Ø67Ø
11	BLOCK	1534	ØØ	Ø21756
14	BTD	Ø254	ØØ	Ø223Ø5
17	BXSET	Ø34Ø	ØØ	Ø22524
22	CHNTST	Ø774	ØØ	Ø23747
25	CONFIG	1334	ØØ	Ø24573
3Ø	CPTEST	237Ø	ØØ	Ø25142
33	DISKIO	1Ø24	ØØ	Ø26564Ø
36	DKIOEX	Ø43Ø	ØØ	Ø3ØØ12
41	DKSETØ	Ø26Ø	ØØ	Ø3Ø22Ø
44	DOMP	2554	ØØ	Ø3Ø663
47	ERR	16Ø4	ØØ	Ø31656
52	F8ØME	21Ø4	ØØ	Ø36ØØ4
55	GETEXT	Ø164	ØØ	Ø3737Ø
6Ø	HSPTES	2214	ØØ	Ø4Ø615
63	INDD39	212Ø	ØØ	Ø41663
66	LISTP	27ØØ	ØØ	Ø43114
71	MFINIT	233Ø	ØØ	Ø43674
74	MULTIPLY	64	ØØ	Ø4564Ø4
77	OTB	Ø15Ø	ØØ	Ø46Ø71
Ø1Ø2	PATCH	Ø64Ø	ØØ	Ø5142Ø
Ø1Ø5	SCRUB	Ø17Ø	ØØ	Ø52212
Ø11Ø	STARTØ	Ø534	ØØ	Ø53Ø14
Ø113	START3	Ø66Ø	ØØ	Ø53415
Ø116	STOP	Ø26Ø	ØØ	Ø54Ø22
Ø121	TCOM	Ø23Ø	ØØ	Ø55262
Ø124	TRACE	133Ø	ØØ	Ø5612Ø
Ø127	UBTAPE	15ØØ	ØØ	Ø56535
Ø132	WATCH	Ø234	ØØ	Ø57274
Ø135	XDISK	Ø754	ØØ	Ø57755
Ø14Ø	XDK	1Ø2Ø	ØØ	Ø61557
Ø143	XPR	Ø52Ø	ØØ	Ø62466
Ø146	XPRNTB	Ø65Ø	ØØ	Ø632Ø5
Ø151	XTAPEB	132Ø	ØØ	Ø63712
Ø154	ACOM	21ØØ	ØØ	Ø64511
Ø157	D3ERR	255Ø	ØØ	Ø653Ø3
Ø162	D3MSG	Ø46Ø	ØØ	Ø66753
Ø165	D3SLR	1144	ØØ	Ø67514
Ø17Ø	D4ERR	254Ø	ØØ	Ø7Ø7Ø4
Ø173	D4MSG	Ø514	ØØ	Ø72424
Ø176	D4RLR	Ø654	ØØ	Ø73Ø36
Ø2Ø1	D4STAT	Ø234	ØØ	Ø73766
Ø2Ø4	ERRECK	317Ø	ØØ	Ø75516
Ø2Ø7	MEMIO	2Ø64	ØØ	Ø77257
Ø212	XDKERR	2Ø7Ø	ØØ	Ø6Ø711
Ø215	BMXCON	324Ø	ØØ	1ØØ553
Ø22Ø	BMXOPE	Ø51Ø	ØØ	1Ø3Ø4Ø
Ø223	BCOM	173Ø	ØØ	1Ø4Ø26
Ø226	CONMAN	112Ø	ØØ	1Ø5333
Ø231	TAPEIO	4ØØØ	ØØ	1Ø6176
Ø234	TAPFUN	Ø4Ø4	ØØ	1Ø7534
Ø237	TDEMØ	Ø5Ø4	ØØ	111Ø5Ø
Ø242	TEX	146Ø	ØØ	112ØØØ
Ø245	TRCLN	Ø23Ø	ØØ	113ØØ3
Ø25Ø	TREQC	Ø214	ØØ	1135Ø3Ø
Ø253	TRORN	Ø554	ØØ	113746
Ø256	TRREDO	Ø754	ØØ	114433
Ø261	TRWRT	Ø7Ø4	ØØ	115356
Ø264	SDMP1	1614	ØØ	117167
Ø1	SDMPB	Ø46Ø	Ø1	Ø21615
Ø4	SDMPE	Ø154	Ø1	Ø22Ø73
Ø7	AMSG	174Ø	ØØ	Ø2Ø772
12	BMOL	461Ø	ØØ	Ø33652
15	BTH	Ø374	ØØ	Ø2236Ø
2Ø	CALL	2Ø1Ø	ØØ	Ø22614
23	CLKSNC	1Ø1Ø	ØØ	Ø24146
26	CPOL	2474	ØØ	Ø35265
31	CRAY	133Ø	ØØ	Ø2564Ø
34	DIVIDE	Ø444	ØØ	Ø26765
37	DKLOOK	Ø1ØØ	ØØ	Ø3Ø12Ø
42	DM3	Ø23Ø	ØØ	Ø3Ø274
45	DTB	Ø24Ø	ØØ	Ø31416
5Ø	ERRDMP	1514	ØØ	Ø32217
53	FDMPDR	Ø52Ø	ØØ	Ø37165
56	HDRPAG	1Ø74	ØØ	Ø37425
61	INFORM	Ø2ØØ	ØØ	Ø4126Ø
64	INDD49	2164	ØØ	Ø423Ø7
67	LPT	136Ø	ØØ	Ø36671
72	MOSTES	1524	ØØ	Ø44362
75	NOBEAT	Ø4ØØ	ØØ	Ø45621
Ø1ØØ	OUTCALL	Ø124	ØØ	Ø46123
Ø1Ø3	PLOTIT	Ø6Ø4	ØØ	Ø5157Ø
Ø1Ø6	SETIME	1474	ØØ	Ø5225Ø
Ø111	START1	Ø314	ØØ	Ø53143
Ø114	START4	Ø33Ø	ØØ	Ø53571
Ø117	SYSS	236Ø	ØØ	Ø54Ø76
Ø122	TDUMP	1514	ØØ	Ø5533Ø
Ø125	TRACK	Ø164	ØØ	Ø56Ø6
Ø13Ø	UNBLK	Ø73Ø	ØØ	Ø57Ø55
Ø133	XCLOCK	172Ø	ØØ	Ø57343
Ø136	XDISKA	1454	ØØ	Ø6Ø15Ø
Ø141	XMT	122Ø	ØØ	Ø61763
Ø144	XPRINT	Ø774	ØØ	Ø62612
Ø147	XTAPE	Ø75Ø	ØØ	Ø63357
Ø152	XTAPEC	Ø61Ø	ØØ	Ø64176
Ø155	BADDAT	Ø26Ø	ØØ	Ø65131
Ø16Ø	D3IOR	2454	ØØ	Ø66Ø35
Ø163	D3RLR	Ø7ØØ	ØØ	Ø67Ø67
Ø166	D4DEM	337Ø	ØØ	Ø67745
Ø171	D4IOR	275Ø	ØØ	Ø71434
Ø174	D4OVR	Ø61Ø	ØØ	Ø72547
Ø177	D4SKR	1424	ØØ	Ø73211
Ø2Ø2	DD49	342Ø	ØØ	Ø74Ø35
Ø2Ø5	FIRECODE	Ø414	ØØ	Ø76354
Ø21Ø	REPORT	Ø7ØØ	ØØ	Ø77674
Ø213	XDKFMT	114Ø	ØØ	Ø61327
Ø216	BMXCPU	2674	ØØ	1Ø1423
Ø221	BMXSIO	155Ø	ØØ	1Ø3162
Ø224	BUFMAN	214Ø	ØØ	1Ø4414
Ø227	DSCGET	117Ø	ØØ	1Ø5557
Ø232	TAPEND	152Ø	ØØ	1Ø7176
Ø235	TAPMOV	1174	ØØ	1Ø7635
Ø24Ø	TDEM1	246Ø	ØØ	111171
Ø243	TRBOC	Ø45Ø	ØØ	112514
Ø246	TRDCK	137Ø	ØØ	113Ø51
Ø251	TRIDB	Ø31Ø	ØØ	113573
Ø254	TRRDB	132Ø	ØØ	114101
Ø257	TRSET	1Ø24	ØØ	114626
Ø262	RSTRT	2154	ØØ	115537
Ø265	SDMP2	175Ø	ØØ	117532
Ø2	SDMPC	Ø35Ø	Ø1	Ø21731
Ø5	MFCHK	1524	Ø1	Ø22126
1Ø	BEGIN	176Ø	ØØ	Ø21362
13	BMOLP	1244	ØØ	Ø35Ø14
16	BTO	Ø224	ØØ	Ø22457
21	CDEM	2544	ØØ	Ø23216
24	CLOCK	1114	ØØ	Ø2435Ø
27	CPSPIN	Ø3ØØ	ØØ	Ø25Ø62
32	CRTDEM	215Ø	ØØ	Ø26126
35	DKDMP	346Ø	ØØ	Ø27Ø76
4Ø	DKSET	Ø3ØØ	ØØ	Ø3Ø14Ø
43	DOM	15Ø4	ØØ	Ø3Ø342
46	ECHOCP	Ø74Ø	ØØ	Ø31466
51	F8ØM	444Ø	ØØ	Ø32542
54	FIND	Ø274	ØØ	Ø37311
57	HPDATA	3644	ØØ	Ø37644
62	INDD29	1614	ØØ	Ø4132Ø
65	LISTO	Ø64Ø	ØØ	Ø42744
7Ø	MAGR	122Ø	ØØ	Ø36425
73	MSGHND	3364	ØØ	Ø447Ø7
76	OBIT	Ø64Ø	ØØ	Ø45721
Ø1Ø1	OVLNUM	Ø1524Ø	ØØ	Ø4615Ø
Ø1Ø4	PRTAPE	13Ø4	ØØ	Ø51731
Ø1Ø7	START	1124	ØØ	Ø52567
Ø112	START2	Ø734	ØØ	Ø53226
Ø115	STATS	Ø614	ØØ	Ø53657
Ø12Ø	SYSTEXT	234Ø	ØØ	Ø54572
Ø123	TIME	1224	ØØ	Ø55653
Ø126	TSTASH	Ø35Ø	ØØ	Ø56443
Ø131	USURP	Ø144	ØØ	Ø57243
Ø134	XCLS	Ø13Ø	ØØ	Ø57727
Ø137	XDISKB	113Ø	ØØ	Ø6Ø463
Ø142	XOPN	1174	ØØ	Ø62227
Ø145	XPRNTA	Ø76Ø	ØØ	Ø63Ø11
Ø15Ø	XTAPEA	Ø6Ø4	ØØ	Ø63551
Ø153	XTAPED	Ø644	ØØ	Ø6434Ø
Ø156	D3ECC	Ø37Ø	ØØ	Ø652Ø5
Ø161	D3LOG	1Ø14	ØØ	Ø6655Ø
Ø164	D3SKR	1224	ØØ	Ø67247
Ø167	D4ECC	Ø2Ø4	ØØ	Ø7Ø643
Ø172	D4LOG	Ø77Ø	ØØ	Ø72226
Ø175	D4RES	Ø524	ØØ	Ø72711
Ø2ØØ	D4SLR	124Ø	ØØ	Ø73516
Ø2Ø3	DISK	2664	ØØ	Ø74741
Ø2Ø6	ICOM	3ØØØ	ØØ	Ø76457
Ø211	TRANSFR	1454	ØØ	1ØØØ54
Ø214	BMXAIO	Ø72Ø	ØØ	1ØØ367
Ø217	BMXDEM	317Ø	ØØ	1Ø22Ø2
Ø222	BMXTPO	145Ø	ØØ	1Ø3514
Ø225	BYPASS	1334	ØØ	1Ø5Ø44
Ø23Ø	TAPDIS	Ø7Ø4	ØØ	1Ø6Ø15
Ø233	TAPERR	5Ø	ØØ	1Ø7522
Ø236	TDEM	366Ø	ØØ	11ØØ74
Ø241	TERROR	1354	ØØ	111705
Ø244	TRCER	Ø664	ØØ	112626
Ø247	TRDCKØ	Ø7Ø4	ØØ	113347
Ø252	TRINR	Ø344	ØØ	113655
Ø255	TRRDF	Ø23Ø	ØØ	114365
Ø26Ø	TRTELL	1514	ØØ	115Ø33
Ø263	SDMPØ	3764	ØØ	116172
Ø266	SDMP3	Ø22Ø	ØØ	12Ø124

Ø267	SDMP3A	Ø55Ø	ØØ	12Ø17Ø
Ø272	SDMP6	Ø234	ØØ	12Ø456
Ø275	SDMP9	Ø454	ØØ	12Ø741
Ø3ØØ	CLEARØ	Ø154	ØØ	121231
Ø3Ø3	COPY1	Ø67Ø	ØØ	121713
Ø3Ø6	COPY4	Ø474	ØØ	122423
Ø311	DDUMPØ	Ø374	ØØ	123Ø36
Ø314	DELETE	Ø65Ø	ØØ	123412
Ø317	DSTAT	Ø514	ØØ	124146
Ø322	EDINST	Ø544	ØØ	124433
Ø325	EDIT1	1174	ØØ	125251
Ø33Ø	EDTYPE	Ø234	ØØ	125666
Ø333	FLAW	Ø63Ø	ØØ	126343
Ø336	FSTAT	Ø65Ø	ØØ	127263
Ø341	INITØ	Ø774	ØØ	127735
Ø344	PROC	Ø564	ØØ	13Ø4Ø3
Ø347	RENAMØ	Ø43Ø	ØØ	131Ø33
Ø352	XFMCRE	Ø71Ø	ØØ	131262
Ø355	XFMDST	Ø634	ØØ	131575
Ø36Ø	XFMFLW	Ø244	ØØ	132Ø62
Ø363	XFMIO	1Ø2Ø	ØØ	132355
Ø366	XFMSTR	Ø2ØØ	ØØ	133Ø77
Ø371	CONCERR	1134	ØØ	133436
Ø374	ENDCONC	Ø5ØØ	ØØ	134761
Ø377	ACQTRM	6Ø	ØØ	135147
Ø4Ø2	BABEL	1634	ØØ	135424
Ø4Ø5	BMGET	Ø374	ØØ	1362ØØ
Ø41Ø	CFREAD	Ø56Ø	ØØ	136634
Ø413	CLUSTR	1ØØØ	ØØ	137657
Ø416	COMBO	Ø134	ØØ	14Ø527
Ø421	COMMØ3	Ø71Ø	ØØ	142Ø55
Ø424	COMMØ6	Ø6Ø4	ØØ	14272Ø
Ø427	COMMØ9	16Ø4	ØØ	1435ØØ
Ø432	COMM12	Ø76Ø	ØØ	144567
Ø435	COMM15	11Ø4	ØØ	145642
Ø44Ø	CPUGET	Ø35Ø	ØØ	146523
Ø443	DECOD2	Ø66Ø	ØØ	147133
Ø446	DESCRIBE	Ø2Ø4	ØØ	15Ø115
Ø451	DISPØ2	122Ø	ØØ	151275
Ø454	DKSTAT	256Ø	ØØ	152125
Ø457	F8Ø	5754	ØØ	153235
Ø462	GRAPH	123Ø	ØØ	156331
Ø465	GRUDIS	2Ø5Ø	ØØ	157462
Ø47Ø	HELPD	Ø2Ø544	ØØ	16Ø511
Ø473	IDEBUG	122Ø	ØØ	165Ø3Ø
Ø476	IFRMT	Ø53Ø	ØØ	165762
Ø5Ø1	LINK	1254	ØØ	166464
Ø5Ø4	MSTAT	1544	ØØ	167312
Ø5Ø7	NSCNET	155Ø	ØØ	17Ø247
Ø512	ONLINE	Ø44Ø	ØØ	171443
Ø515	PROTOCOL	151Ø	ØØ	172151
Ø52Ø	SNAP	Ø474	ØØ	173Ø66
Ø523	STADIS	214Ø	ØØ	173533
Ø526	STATCL	1374	ØØ	175ØØ7
Ø531	STIO	Ø35Ø	ØØ	175453
Ø534	STREAMS	1114	ØØ	176317
Ø537	STTAPI	Ø654	ØØ	177152
Ø542	STXDKI	1Ø44	Ø1	ØØØØ55
Ø545	SUMHOW	Ø664	Ø1	ØØ1242
Ø55Ø	SYSTAT	1754	Ø1	ØØ2351
Ø553	TJOB	1674	Ø1	ØØ3376
Ø556	UPDATE	Ø43Ø	Ø1	ØØ4576
Ø561	XMPXP	226Ø	Ø1	ØØ5726
Ø564	IACON	Ø76Ø	Ø1	ØØ7567
Ø27Ø	SDMP4	Ø35Ø	ØØ	12Ø322
Ø273	SDMP7	Ø25Ø	ØØ	12Ø525
Ø276	SDMP1Ø	Ø144	ØØ	121Ø54
Ø3Ø1	COPY	1Ø4Ø	ØØ	121264
Ø3Ø4	COPY2	Ø744	ØØ	122Ø71
Ø3Ø7	COPY5	Ø55Ø	ØØ	122542
Ø312	DEF	Ø65Ø	ØØ	123135
Ø315	DLOAD	Ø56Ø	ØØ	123564
Ø32Ø	DSTATØ	Ø5Ø4	ØØ	124271
Ø323	EDIT	Ø644	ØØ	124564
Ø326	EDPRNT	Ø534	ØØ	12551Ø
Ø331	FDUMP	Ø77Ø	ØØ	125735
Ø334	FLOAD	Ø714	ØØ	12662Ø
Ø337	FSTATØ	Ø55Ø	ØØ	127435
Ø342	LINGET	Ø35Ø	ØØ	13Ø134
Ø345	PROCØ	Ø554	ØØ	13Ø54Ø
Ø35Ø	XFMACC	Ø244	ØØ	131141
Ø353	XFMDEL	Ø174	ØØ	131444
Ø356	XFMFIL	Ø324	ØØ	131744
Ø361	XFMFND	Ø4Ø4	ØØ	132133
Ø364	XFMNIT	1Ø54	ØØ	132561
Ø367	FORMAT	Ø434	ØØ	126511
Ø372	CONCID	1214	ØØ	133665
Ø375	ENTRID	Ø13Ø	ØØ	135Ø1
Ø4ØØ	ACQUIRE	Ø4Ø4	ØØ	135163
Ø4Ø3	BARDAT	Ø42Ø	ØØ	135773
Ø4Ø6	BXDIS	Ø544	ØØ	136277
Ø411	CLI	172Ø	ØØ	13677Ø
Ø414	CMGET	Ø22Ø	ØØ	14ØØ57
Ø417	COMMØ1	3744	ØØ	14Ø556
Ø422	COMMØ4	1Ø74	ØØ	142237
Ø425	COMMØ7	Ø71Ø	ØØ	143Ø61
Ø43Ø	COMM1Ø	1464	ØØ	144Ø41
Ø433	COMM13	1764	ØØ	144763
Ø436	COMM16	Ø73Ø	ØØ	146Ø63
Ø441	CRAYIO	11ØØ	ØØ	146615
Ø444	DECODE	163Ø	ØØ	1473Ø7
Ø447	DEVDAT	Ø56Ø	ØØ	15Ø156
Ø452	DISPLAY	13ØØ	ØØ	151541
Ø455	ERRDIS	1424	ØØ	152661
Ø46Ø	F132	5754	ØØ	15463Ø
Ø463	GRCDIS	2Ø74	ØØ	156577
Ø466	HELP	Ø474	ØØ	16Ø074
Ø471	HSPGET	Ø4ØØ	ØØ	164642
Ø474	IDLGET	Ø45Ø	ØØ	165274
Ø477	KEYBD	Ø514	ØØ	16611Ø
Ø5Ø2	LOGON	14ØØ	ØØ	166737
Ø5Ø5	MSTDIS	154Ø	ØØ	167643
Ø51Ø	NSCSFP	222Ø	ØØ	17Ø6Ø1
Ø513	POST	Ø514	ØØ	171553
Ø516	QUEUE	Ø34Ø	ØØ	172473
Ø521	SOROC	Ø44Ø	ØØ	1732Ø5
Ø524	STAGEIN	1414	ØØ	174163
Ø527	STATINIT	Ø54Ø	ØØ	1753Ø6
Ø532	STMSG	2Ø44	ØØ	175545
Ø535	STRSTAT	146Ø	ØØ	176542
Ø54Ø	STTAPO	Ø57Ø	ØØ	177323
Ø543	STXDKO	Ø72Ø	Ø1	ØØØ266
Ø546	SUMTIM	Ø674	Ø1	ØØ1417
Ø551	TAPEC	165Ø	Ø1	ØØ2744
Ø554	TKSTAT	211Ø	Ø1	ØØ3755
Ø557	XFRMT	15ØØ	Ø1	ØØ47Ø4
Ø562	XMP4XP	2754	Ø1	ØØ64Ø2
Ø565	IACON1	Ø4Ø4	Ø1	ØØ7763
Ø271	SDMP5	Ø21Ø	ØØ	12Ø414
Ø274	SDMP8	Ø61Ø	ØØ	12Ø577
Ø277	CLEAR	Ø52Ø	ØØ	121105
Ø3Ø2	COPYØ	1Ø74	ØØ	121474
Ø3Ø5	COPY3	Ø6Ø4	ØØ	122262
Ø31Ø	DDUMP	Ø61Ø	ØØ	122674
Ø313	DELETØ	Ø414	ØØ	123Ø7
Ø316	DLOADØ	113Ø	ØØ	12372Ø
Ø321	EDDELE	Ø1Ø4	ØØ	124412
Ø324	EDITØ	146Ø	ØØ	124735
Ø327	EDREPL	Ø134	ØØ	125637
Ø332	FDUMPØ	1Ø4Ø	ØØ	126133
Ø335	FLOADØ	13ØØ	ØØ	127ØØ3
Ø34Ø	INIT	Ø63Ø	ØØ	127567
Ø343	LINPUT	Ø664	ØØ	13Ø226
Ø346	RENAME	Ø6ØØ	ØØ	13Ø673
Ø351	XFMCLS	Ø24Ø	ØØ	131212
Ø354	XFMDIR	Ø35Ø	ØØ	131503
Ø357	XFMFST	Ø144	ØØ	132Ø31
Ø362	XFMGET	Ø5Ø4	ØØ	132234
Ø365	XFMPUT	Ø414	ØØ	132774
Ø37Ø	CONC	1374	ØØ	133137
Ø373	CONCIO	3144	ØØ	13413Ø
Ø376	REMVID	Ø1ØØ	ØØ	135127
Ø4Ø1	AMPEX	Ø6ØØ	ØØ	135264
Ø4Ø4	BMAGET	Ø4Ø4	ØØ	136Ø77
Ø4Ø7	CFINIT	1Ø2Ø	ØØ	136430
Ø412	CLINIT	1414	ØØ	137354
Ø415	CNCDIS	2Ø2Ø	ØØ	14Ø123
Ø42Ø	COMMØ2	143Ø	ØØ	141547
Ø423	COMMØ5	121Ø	ØØ	142456
Ø426	COMMØ8	1164	ØØ	143243
Ø431	COMM11	1Ø44	ØØ	144356
Ø434	COMM14	131Ø	ØØ	145360
Ø437	CONSL	125Ø	ØØ	146251
Ø442	DBGET	Ø37Ø	ØØ	147Ø35
Ø445	DELMSG	12ØØ	ØØ	147655
Ø45Ø	DISPØ1	3714	ØØ	15Ø312
Ø453	DKDIS	Ø42Ø	ØØ	152Ø21
Ø456	ERROR	Ø234	ØØ	153166
Ø461	FMGET	Ø43Ø	ØØ	156223
Ø464	GRPDIS	122Ø	ØØ	157216
Ø467	HELP2	137Ø	ØØ	16Ø213
Ø472	ICONSL	Ø33Ø	ØØ	164742
Ø475	IDRCT	166Ø	ØØ	165406
Ø5ØØ	LCP	1144	ØØ	166233
Ø5Ø3	MESSAGE	Ø254	ØØ	167237
Ø5Ø6	NEWDIS	Ø26Ø	ØØ	17Ø173
Ø511	OFRMT	Ø77Ø	ØØ	171245
Ø514	PROTINIT	1254	ØØ	171676
Ø517	READ	1414	ØØ	172563
Ø522	SSDGRA	1Ø7Ø	ØØ	173315
Ø525	STAGEOUT	15Ø4	ØØ	174466
Ø53Ø	STATION	64	ØØ	175436
Ø533	STPLOT	Ø6Ø4	ØØ	176156
Ø536	STSGET	Ø35Ø	ØØ	177056
Ø541	STUBPR	176Ø	ØØ	177461
Ø544	SUMDAY	274Ø	Ø1	ØØØ452
Ø547	SYNTAX	2654	Ø1	ØØ1576
Ø552	TEC455	Ø3ØØ	Ø1	ØØ3316
Ø555	TURNPG	Ø774	Ø1	ØØ4377
Ø56Ø	XJSTAT	241Ø	Ø1	ØØ5224
Ø563	IACMD	175Ø	Ø1	ØØ7175
Ø566	IAFUNC	1444	Ø1	Ø1ØØ

Z	SCP	STG	DQM	PDM	JCM	JSH	DQM	DEC	EXP	MEP	MSG	SPM	IQM	FVD
AUT	-AUT	PDD	-DAT	DAT	-CSD	CSD	(DDL)	EQT	CNT	-AEM	AUT	CSD	ILT	EQT
CNT	-IBT	SDT	DCT	CSD	SDT	-JXT	-(DNT)		-DEX		(DSP)	DCT	VPT	DRT
DAT	-LCT	-SST	(DNT)	(DNT)		-MST	-(DSP)		(DNT)		JTA	+MCT	VCT	
(DNT)	-LIT		-DRT	DRT		-RJI	JXT		(DSP)		JXT	+ICT	TRB	
DRT	-LXT		(DSP)	-DSC		SDT	-LFT		-DUX		-LGJ	+STT	IST	
-DSC	PDD		-EQT	(DSP)		-TXT	-ODN		-FSH		PDD			
(DSP)	-SDT		GRT	DXT		JTA	(PDD)		GRT		SDT			
-DVL	-STT		JXT	EQT			SDT		JXT					
DXT			-RQT	JCB			-SWT		-LDT					
-EFT			JTA	JTA			TCB		-SM					
-EQT			PHR	JXT			-UPT		-VAX					
GRT			SQP	PDD			-JTA		-VUX					
JTA			TXT	-PDI			-TXT		JTA					
JXT				-PDS			IDD							
(ODT)				-QDT			IRT							
PDI				SDT			TRB							
QDT				-XAT										
-RJI														
SDT														
TCB														
TDT														
TXT														

PACKET DRIVER	FED DRIVER	DISK/SSD DRIVER
+SCT	+CHT	DCT
+PIQ	+CXT	EQT
+POQ	LIT	+CHT
+CHT	LXT	+CAT
+CAT	LCT	+CLT
+CLT	+CAT	+CIT
+CIT	+CLT	+CBT
CXT	+CIT	+PHR
		+SAC
		+SQP

USER		CSP
(BGN)	(LDT)	(DSP)
(BAT)	(OPT)	(JCB)
(DDL)	(JBI)	(LFT)
(DSP)	(DRPB)	(DNT)
(DNT)	(RCB)	
(JAC)	(NCB)	
(JCB)	(MHB)	
(LFT)	(IJPB)	
(ODN)		
(PDD)		
(JST)		