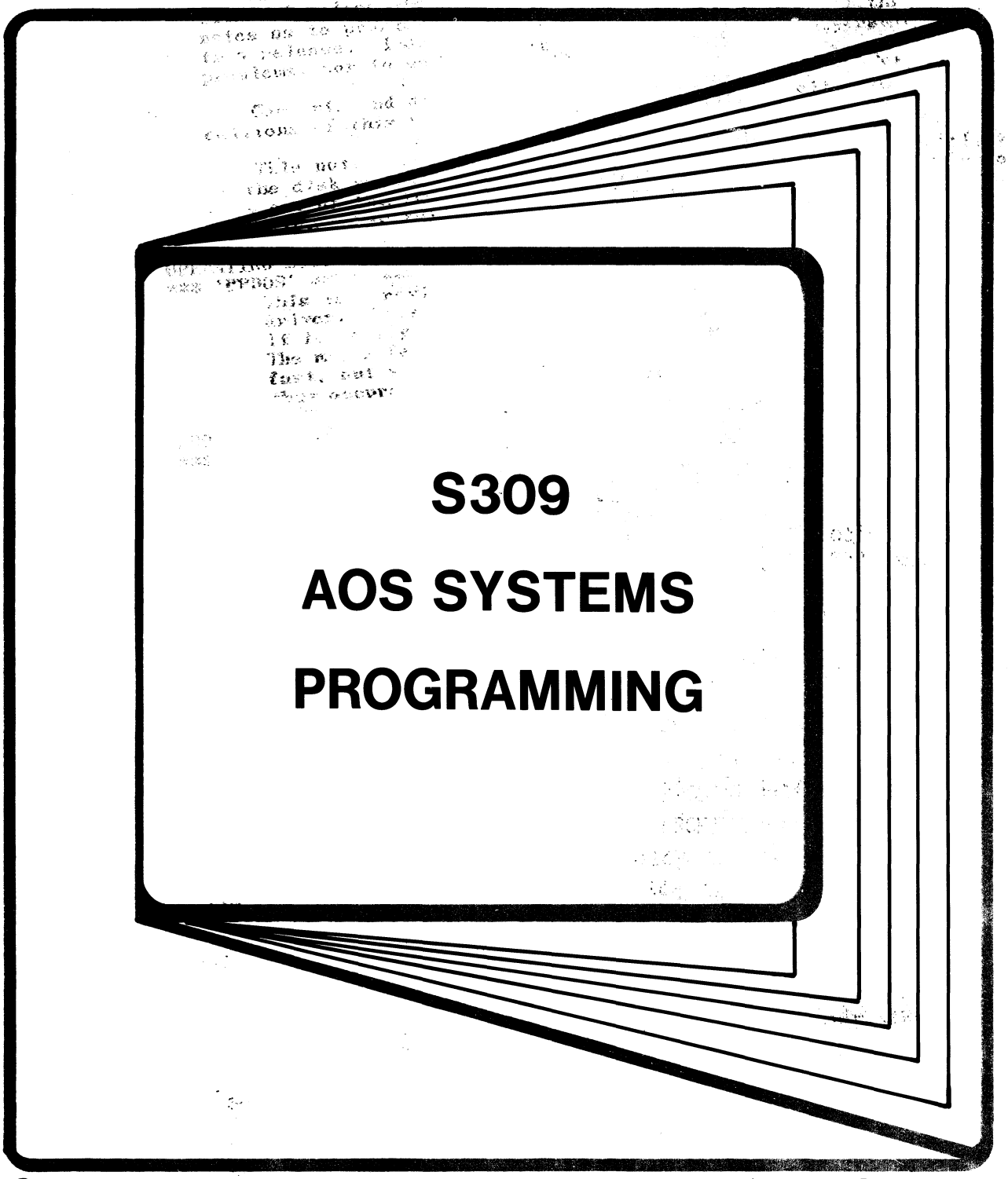
DataGeneral



S309

AOS SYSTEMS PROGRAMMING

Student Handbook

Educational Services

019-000079-00

NOTICE

Data General Corporation (DGC) has prepared this manual for use by DGC personnel and/or customers as a guide to the proper installation, operation, and maintenance of DGC equipment and software. The drawings and specifications contained herein are the property of DGC and shall neither be reproduced in whole or in part without DGC prior written approval nor be implied to grant any license to make, use, or sell equipment manufactured in accordance herewith.

DGC reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including but not limited to typographical or arithmetic errors, company policy and pricing information.

The following are trademarks of Data General Corporation, Westboro, Massachusetts:

U. S. REGISTERED TRADEMARKS	TRADEMARKS
ECLIPSE	DASHER
NOVA	microNOVA
INFOS	DG/L

S309

AOS SYSTEMS PROGRAMMING

COURSE DESCRIPTION

S309 AOS SYSTEMS PROGRAMMER

PREREQUISITES: S209 and the appropriate language course.

GOAL: To instruct and exercise the students in the features of AOS available to user programs through the use of system calls. Emphasis will be placed on design and implementation in the lab using Eclipse Assembler and supported features of FORTRAN IV /and V.

SUMMARY OF MAJOR TOPICS:

Record and Block I/O

IPC

Directory structure and Generic files

Multiprocess environments

Multi Tasking

Shared and Unshared Memory

Overlays shared routines and resource management

Accounting structures

Interfacing User Drivers

Extending the Error and Help files

Binary Synchronous communications. ALM, MCA

Utilities

DURATION: 5 Days

This page intentionally left blank.

COURSE OUTLINE

Course Title: S309 AOS SYSTEMS PROGRAMMER

Section		Modules	
I	INTRODUCTION	A.	MANUALS AND HANDOUTS
		B.	SYSTEM CALL FORMAT
II	I/O CREATION AND ACCESS	A.	DIRECTORIES
		B.	GENERIC AND LINK FILES
		C.	RECORD I/O
		D.	DIRECT BLOCK I/O
		E.	IPC FACILITY
III	SCHEDULING	A.	PROCESSES
		B.	TASKING
		C.	GHOST
IV	MEMORY MANAGEMENT	A.	MAP HARDWARE
		B.	UNSHARED MEMORY
		C.	SHARED MEMORY
		D.	PROGRAM STRUCTURE
		E.	PROCESS TRADEOFFS
V	INTERFACING	A.	SYSTEM DATA
		B.	USER DRIVERS
		C.	MASTER CONSOLE SECURITY
		D.	CONSOLE CONTROL CHARACTERS
VI	COMMUNICATIONS	A.	ASYNCHRONOUS
		B.	BINARY SYNCHRONOUS

MODULE TOPICS

Section I — INTRODUCTION

Module Title: A: MANUALS AND HANDOUTS

1. Manual Summary
2. Structure and use of the student handout

Module Title: B: SYSTEM CALL FORMATS

1. URT.LB
2. System command definitions SYSID.SR
3. System calls and returns
4. PARU.SR

Section II — I/O CREATION AND ACCESS

Module Title: A:DIRECTORIES

1. Disk directory hierarchy
2. Directory creation and deletion
3. Control Point Directories
4. Hash frame size
5. Directory access
6. Initializing and releasing logical Disks

Module Title: B: GENERIC AND LINK FILES

1. Peripheral Directory
2. Generic filename types
3. Setting generic filenames
4. Creating and deleting links
5. Resolving Links

Module Title: C: RECORD I/O

1. Filenames
2. Creating and opening files
3. Assigning channel numbers
4. Disk files, index levels, element sizes
5. Magnetic tape files, blocks, mount option
6. Byte devices:
   - Console characteristics and control characters
   - Ownership
   - ASCII/Binary modes
7. Record types
8. File Access:
   - Positioning
   - Read/Write

Section II I/O CREATION AND ACCESS (Continued)

Module Title: D: DIRECT BLOCK I/O

1. Filenames and Device names
2. Magnetic tape structures
   blocks, filenumbers, positioning,
   tape volumes
3. Logical Disk files
   blocks, offsets
4. DG Disk Structure
5. Data Channel Printer
   VFU control
6. Block access
   channel numbers, positioning,
   read/write

Module Title: E: IPC

1. Sending messages to consoles
2. Reads/Writes of an IPC file
3. Primitive IPC ports & headers
4. Transmitting & receiving messages
5. Creating IPC-type files

Section III SCHEDULING

Module Title: A: PROCESSING

1. Process Creation & Termination
2. Process Control Block Structure
3. Process Control of a subordinate process
4. Synchronization & Communication between
   processes using IPC files and common disk files
5. Message handling between running processes

Module Title: B: GHOST

1. Ghost interaction with system and user context.
2. Scheduling considerations involving the Ghost.
3. Ghost initialization.
4. The three functions the Ghost performs.

Module Title: C: TASKING

1. Task creation and termination.
2. Task Control Block structure.

Module Title: C: TASKING (continued)

3. Overhead in scheduling a task.
4. Synchronization and communication between tasks.
5. Task control in the user program.

Section IV MEMORY MANAGEMENT

Module Title: A: MAP HARDWARE

1. A description of the MAP features
2. How the MAP works
3. Where the MAP is located
4. The sequence of events involving the MAP
5. The MAP protection capability
6. MAP TRAPS

Module Title: B: UNSHARED MEMORY

1. Unshared Overlays
2. Unshared in-line code
3. Stack Usage
4. System Tables

Module Title: C: SHARED MEMORY

1. Least Recently Used Algorithmn
2. Shared Data
3. Shared In-Line Code
4. Shared Overlays
5. Shared Routines

Module Title: D: PROGRAM STRUCTURE

1. Memory allocation
2. Resource Calls
3. Shared & Unshared memory use

Module Title: E: PROCESS TRADEOFFS

1. Choosing Process Type and Priority for efficient use of memory
2. Bias Factor

Section V INTERFACE

Module Title: A: DRIVERS

1. Interrupt Vector Table and Device Control Table.
2. System control of an interrupt service routine.
3. Introducing and removing an interrupt service routine.
4. Task scheduling upon exit from an interrupt service routine.
5. Communicating with the multi task environment and the multi process environment.
6. Direct access to devices at the task level.
7. Adding extra devices at AOSGEN time.
8. Patching PMGR for foreign consoles.

Module Title: B: CONTROL CHARACTERS

1. Control C Characters sequence.
2. Disabling/Enabling control characters.
3. Defining an interrupt processing task.

Module Title: C: SYSTEM DATA

1. Write user help files.
2. Update and access ERMES.
3. EXEC control.
4. Access a process run time statistics.
5. Process histograms.
6. Access SYSLOG data.
7. Spooling files.
8. Floating point unit.

Module Title: D: SECURING THE MASTER CONSOLE

1. CLI.PR
2. CLI System Calls
3. Disabling Control C Interrupts.
4. UP Macro.

Section VI COMMUNICATIONS

Module Title: A: ASYNCHRONOUS

1. ALM line characteristics, consolenames.
2. MCA Protocols.
3. MCA Record and Block Data Transfers.

Module Title: B: BINARY SYNCHRONOUS

1. Bi-synch protocol.
2. Enabling and disabling a BSC Line.
3. Point-to-point communications.
4. Receiving and sending data over a BSC Line.
5. BSC Line error statistics.
6. Multipoint communications.

SECTION GOAL

Course Title: S309 SYSTEM PROGRAMMER

Section I INTRODUCTION

To familiarize the student with the general administrative procedures and conventions of this course.

MODULE OBJECTIVES

Section I INTRODUCTION

Module Title: A: MANUALS AND HANDOUTS

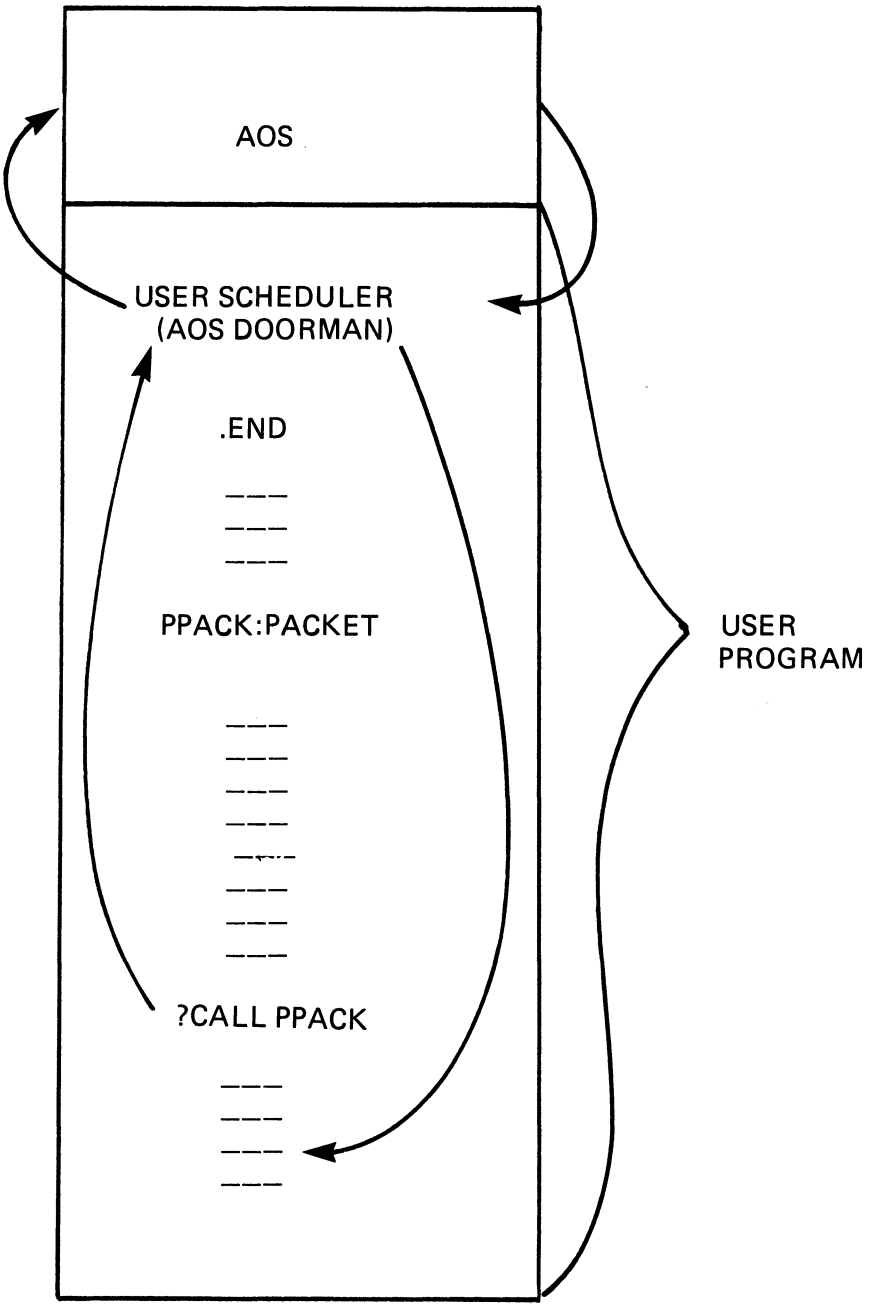
Upon successful completion of this module the student will be able to identify the different manuals and handouts used throughout this course.

Module Title: B: SYSTEM CALL FORMAT

Upon successful completion of this module the student will be able to:

1. Identify a system call
2. Recognize a parameter packet; by using PARU.SR construct packet-parameters.
3. Identify exceptional system call returns and error codes.

AOS/USER INTERFACE



SYSTEM CALLS

Transfer of execution
into AOS

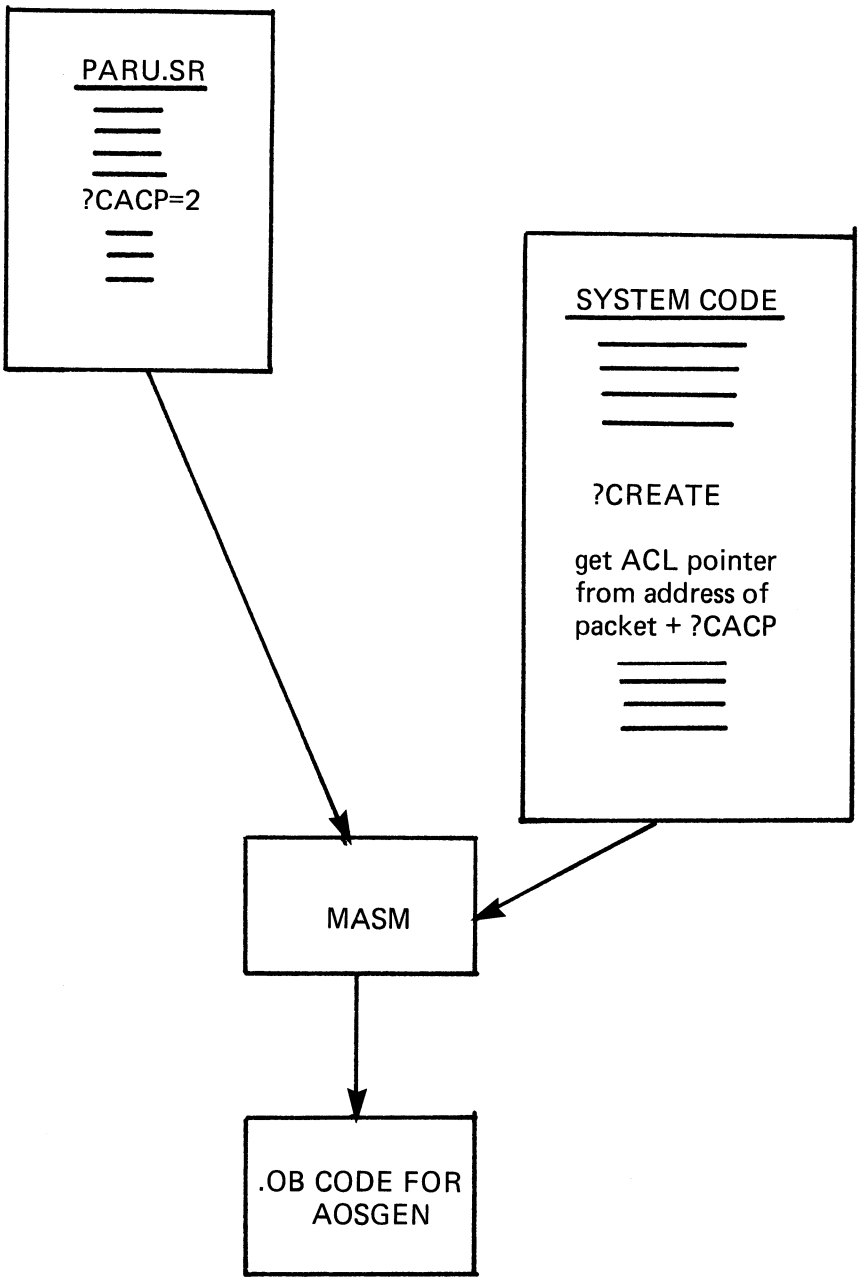
Hardest part for you
is setting up
parameter packet

Bind automatically
includes scheduler
from URT.LB

ASSEMBLER: *DIRECT JUMP TO SCHEDULER
*DIRECT ACCESS TO ADDRESSES FOR PACKET

COMPILERS: *JUMP TO RUNTIME LIBRARY ROUTINE,
ROUTINE JUMPS TO SCHEDULER
*YOU PASS NAMES OF STORAGE LOCATIONS,
ROUTINE FILLS IN PACKET

USER PARAMETER FILE



PARU.SR

Ultimate source for
Parameter Packets

Used by AOS Code to
Get Arguments
and
Return Results

You Use it as a Guide
When Building Packets
*Overall Structure
*Individual Values

Your Assembly Language
Code Can Manipulate
Packets at Runtime

SECTION GOAL

Course Title: S309 SYSTEMS PROGRAMMER

Section II I/O CREATION AND ACCESS

To enable the programmer to create and access particular file types supported by AOS.

MODULE OBJECTIVES

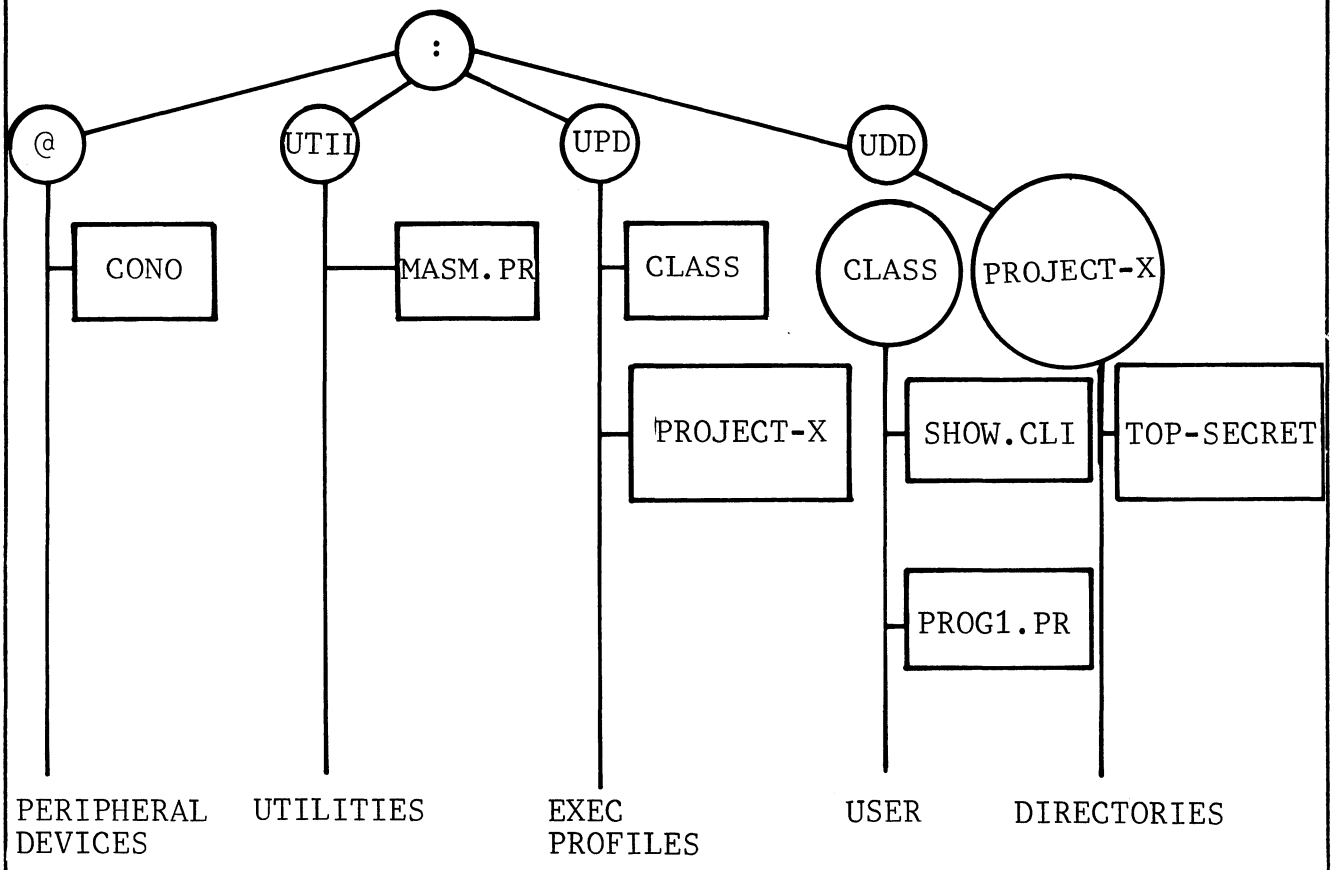
Section II I/O CREATION AND ACCESS

Module Title: A: DIRECTORIES

Upon successful completion of this module the student will be able to:

1. Describe how to create and utilize the disk directory structure.
2. Identify the system calls required to protect directories and user files.

DIRECTORY HEIRACHY



DIRECTORY CREATION

?CREATE

DIRECTORY PARAMETER PACKAGE INCLUDES:

PARAMETER	DEFAULT
CPD TYPE	(MAX SIZE REQUIRED)
ACL	(YOUR NAME OWARE)
HASH FRAME SIZE	(7)
NUMBER OF INDEX LEVELS	(3)
TIME BLOCK	(CURRENT SYSTEM TIME)

?DELETE

ACCESS AND PROTECTION

ACL PROTECTS AGAINST UNWANTED USERS

DEFAULT – PROCESS USER NAME
– ALL ACCESS PRIVILEGES

USER DEFINED – AT FILE CREATION
– SYSTEM CALL ?SACL

DIRECTORIES CAN BE MADE PERMANENT
PERMANENT DIRECTORIES CANNOT BE DELETED
?SATR

GETTING A REPORT ON CURRENT ACL

?GACL – FILE'S ENTIRE LIST

?GTACP – ACCESS CONTROL PRIVILEGES FOR YOUR USERNAME ALONE

FILE ACCESS

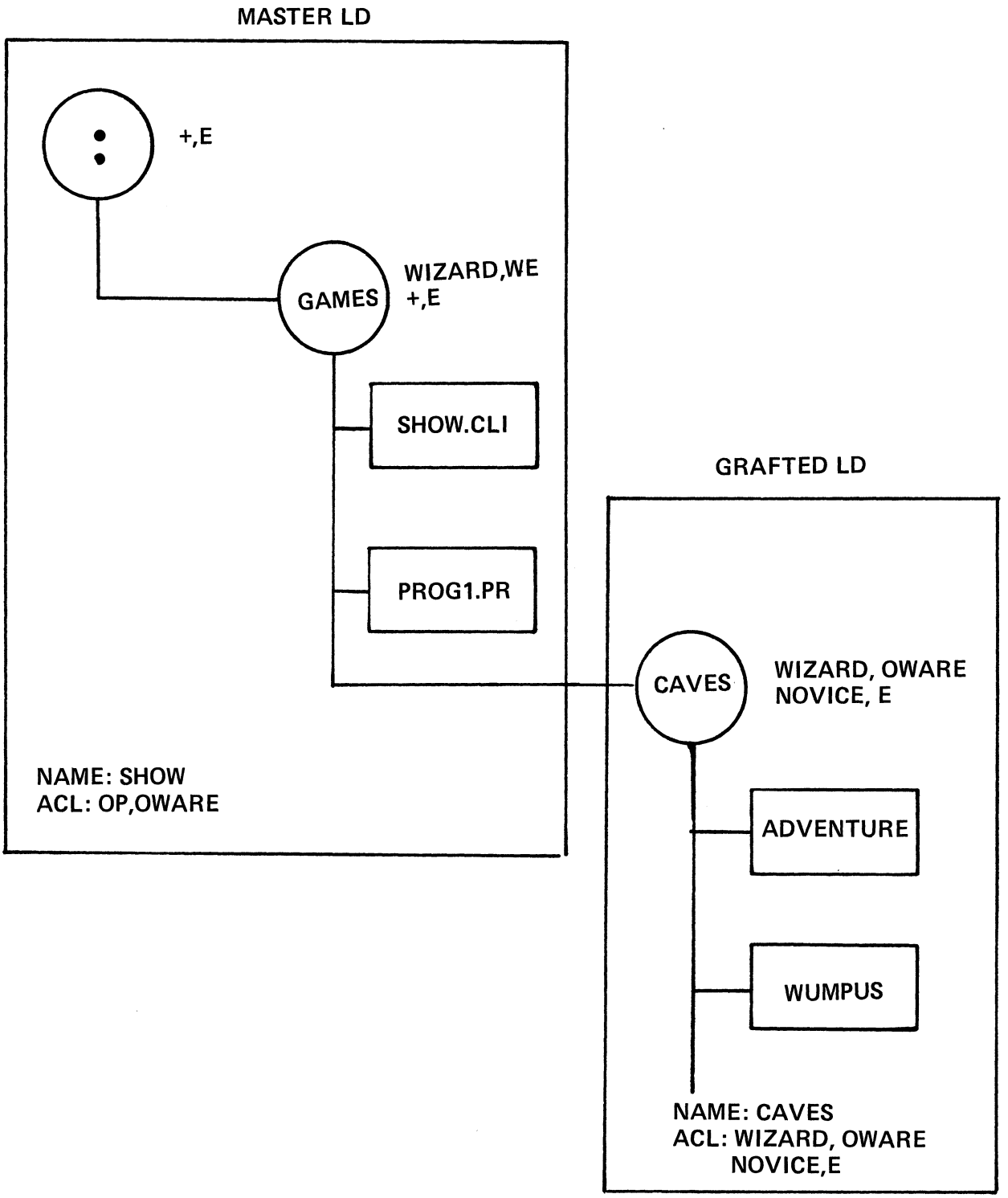
DIRECT PATHNAME ?GNAME

SEARCHLIST ?GLIST, ?SLIST

CHANGE THE WORKING DIRECTORY ?DIR

NEXT FILENAME ?GNFN

LOGICAL DISK GRAFTS



GRAFTING LOGICAL DISKS

ENTIRE LOGICAL DISK APPEARS AS A DIRECTORY
UNDER CURRENT WORKING DIRECTORY

TO GRAFT YOU NEED:

OWNER ACCESS TO NEW LD

WRITE OR APPEND ACCESS TO
THE WORKING DIRECTORY

INITIALIZE BY PHYSICAL DEVICE NAME

?INIT TO DPF5

RELEASE BY LOGICAL DISK NAME

?RELEASE TO :GAMES:CAVES

DISKS GIVEN LOGICAL NAMES AT FORMATTING TIME

MASTER LD = LOGICAL NAME IS IGNORED & REPLACED BY:

MODULE SYSTEM CALL SUMMARY

SECTION II I/O CREATION AND ACCESS

MODULE TITLE: A: DIRECTORIES

?CREATE	CREATE A FILE AS A DIRECTORY TYPE
?DELETE	DELETE A DIRECTORY
?GACL, ?SACL	OBTAIN AND SET THE ACCESS CONTROL LIST
?GLIST, ?SLIST	OBTAIN AND SET THE SEARCHLIST
?GTACP	OBTAIN THE ACL PRIVILEGES
?FSTAT	OBTAIN STATUS INFORMATION
?SATR	SET THE PERMANENT ATTRIBUTE
?DIR	CHANGE THE WORKING DIRECTORY
?GNFN	OBTAIN THE NEXT FILE IN A DIRECTORY
?INIT	GRAFT A LOGICAL DISK ONTO THE DIRECTORY STRUCTURE
?RELEASE	REMOVE A LOGICAL DISK FROM THE DIRECTORY STRUCTURE
?GNAME	GET COMPLETE PATHNAME

MODULE OBJECTIVES

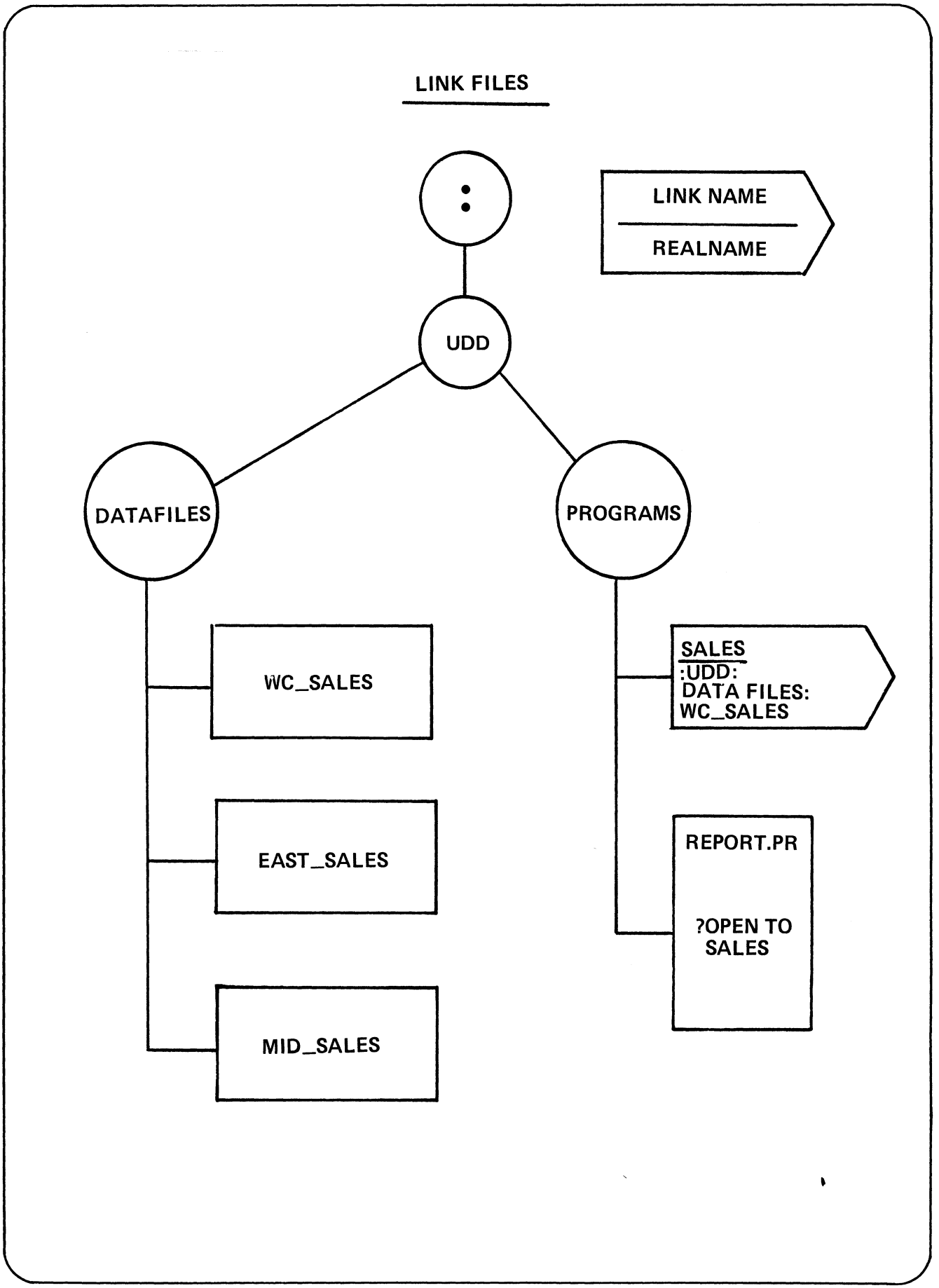
Section II I/O CREATION AND ACCESS

Module Title: B: GENERIC AND LINK FILES

Upon successful completion of this module the student will be able to:

1. Design programs to use generic files

2. List the options that support file independence.



LINK ENTRIES

LINK ENTRY IS A FILENAME

CONTENTS OF THE LINK ENTRY IS A PATHNAME

LINK NAME IS REPLACED BY CONTENTS FOR ALL SYS CALL EXCEPT ?DELETE

LINK ENTRIES ARE SPECIFIED AT CREATION TIME (?CREATE) AND CAN BE DELETED

CONTENTS OF THE LINK ENTRY CAN BE REQUESTED ?GLINK

GENERIC FILENAMES

FILENAMES PRECEDED BY @ PATHNAME PREFIX

FOUR TYPES:

INPUT	@INPUT, @DATA
OUTPUT	@OUTPUT, @LIST
DEVICE	@CONSOLE
EMPTY	@NULL

SET BY FATHER PROCESS

MUST BLOCK IF PASSING OWN GENERICS

USING GENERIC FILENAMES

ENCOURAGES FILE/DEVICE INDEPENDENCE

TREAT LIKE ANY OTHER FILE

ASSIGN A CHANNEL NUMBER

READ AND WRITE

PROCESS CANNOT CHANGE ITS OWN GENERIC FILES

MODULE SYSTEM CALL SUMMARY

SECTION II I/O CREATION AND ACCESS

MODULE TITLE: B: GENERIC AND LINK FILES

?PROC	USE THE FLAGS TO DEFINE THE GENERIC FILES AT PROCESS CREATION.
?CREATE	USE THE LINK FLAGS TO DEFINE THE TYPE OF FILE CREATED.
?DELETE	DELETE A FILE.
?GLINK	RESOLVE A LINK FILE.

MODULE OBJECTIVES

Section II I/O CREATION AND ACCESS

Module Title: C: RECORD I/O

Upon successful completion of this module the student will be able to:

1. List the system calls and describe the major areas of the parameters packets used to access record-structured files and devices.
2. Describe the structures and applications of the different record types.
3. For PER-type devices define the characteristics and the modes.
4. For PER-type devices choose the appropriate type of ownership for particular programming applications.
5. Describe how data is stored on magnetic tape and how to access that data.
6. Design disk files that are stored most efficiently and accessed most quickly.

FILENAMES

THE SYSTEM ENCOURAGES FILE INDEPENDENCE – NO RESERVED NAMES

CON5 IN YOUR DIRECTORY USED TO HOLD OUTPUT WHEN CONSOLE BUSY

DEVICE NAME TYPES

SINGLE FILE (SPOOLABLE PER.)

MULTIPLE FILE (DISK, MCA, MAG TAPE)

DISK FILE TYPES

USER DATA

LINK

DIRECTORY

PROGRAM

-
-
-
-

FILENAME ACCESS

SAME PROCEDURE TO ACCESS ANY FILE TYPE

- ASSOCIATE A CHANNEL TO A FILENAME (?OPEN)
- READ AND WRITE RECORDS FROM/TO CHANNEL (?READ & ?WRITE)
- FREE THE CHANNEL (?CLOSE) [AUTOMATIC ON PROCESS TERMINATION]

**NOTE: PARAMETER PACKET STRUCTURE FOR ABOVE SYSTEM CALLS ARE IDENTICAL

RECORD FORMATS

RECORD = UNIT OF DATA TRANSFERED
BETWEEN PROGRAM AND AOS

RECORD IS A LOGICAL UNIT
(i.e., NO. OF BYTES)

BLOCK IS A PHYSICAL UNIT
(i.e., SET BY HARDWARE)

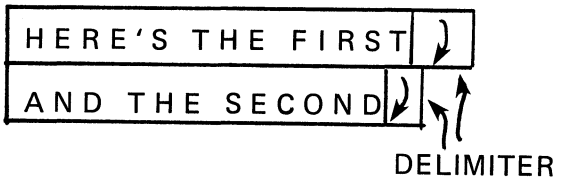
RECORD SIZE INDEPENDENT OF
DEVICE'S PHYSICAL BLOCKINGS

FORMAT SELECTED AT
CREATE, OPEN, OR READ/WRITE

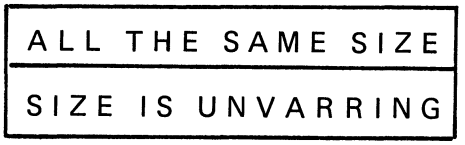
LENGTH OR LIMIT SET AT
OPEN OR READ/WRITE

RECORD FORMATS

DATA SENSITIVE

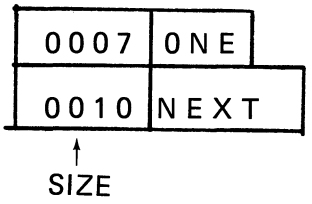


FIXED



DYNAMIC

VARIABLE



FIXED

- USER RESPONSIBLE FOR RECORD SIZE

- ALLOW FOR EASIER JUMPING AROUND IN FILE

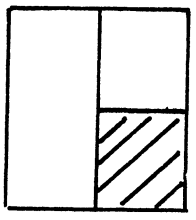
- RECORD SIZE DOES NOT HAVE TO BE A MULTIPLE OF ELEMENT SIZE

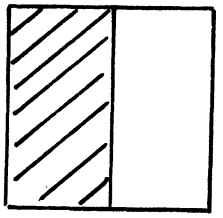
HOWEVER

- TO INCREASE ACCESS SPEED RECORD SIZE CAN BE MADE A MULTIPLE OF ELEMENT SIZE

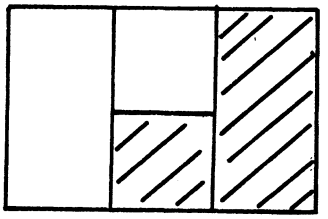
- EXAMPLES RECORD = 1.5 BLOCKS
  GET SECOND RECORD

ELEMENT = 2





ELEMENT = 3



DYNAMIC

- SIZE OF RECORD CAN CHANGE FOR EACH TRANSFER
- CAN'T POSITION BY RECORD
- NO PHYSICAL DELIMITERS BETWEEN RECORDS
- EXAMPLE

  SYSTEM LOG FILE

VARIABLE

- SIZE IS AUTOMATICALLY RECORDED WITH EACH WRITE
- SIZE BLOCK EXPECTED TO HEAD EACH RECORD READ
- HEADER IS TRANSPARENT TO USER
- HEADER IS 4 DIGIT ASCII; COUNT INCLUDES THESE FOUR
- ⅋ BYTES TRANSFERRED RETURNED TO USER BY SYSTEM
- ADVANTAGES
  - HAS FLEXIBILITY OF DYNAMIC RECORD WITHOUT USER PAYING COST OF KEEPING TRACK OF INPUT RECORD LENGTH.
- DISADVANTAGES
  - CAN ONLY READ RECORDS WITH HEADERS

DATA SENSITIVE

- SIZE IS DETERMINED BY OCCURRENCE OF DELIMITER

- DEFAULT DELIMITERS:

FORM FEED	〈014〉
NEW LINE	〈012〉
CARRIAGE RETURN	〈015〉
NULL	〈000〉

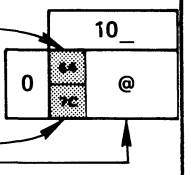
- DELIMITERS CAN BE SET BY USER AT ?OPEN

- DELIMITER IS PART OF RECORD THAT USER SEES

- # OF BYTES TRANSFERRED RETURNED TO USER BY SYSTEM

ASCII CHARACTER CODES
WITH DISPLAY TERMINAL, MODELS 6052/6053 CONTROL FUNCTIONS

LEGEND:



↥ *means* CONTROL

octal	00_		CONTROL FUNCTION	01_		CONTROL FUNCTION	02_		CONTROL FUNCTION	03_		CONTROL FUNCTION	04_		05_	
0	0 00	NUL	NULL	8 08	BS *(BACK-SPACE)*	HOME	16 10	DLE ↥P	WRITE CURSOR HEADER	24 18	CAN ↥X	CURSOR RIGHT	32 20	SPACE	40 28	(
1	1 01	SOH ↥A	PRINT FORM	9 09	HT *(TAB)*	HORIZONTAL TAB	17 11	DC1 ↥Q	PRINT PAGE	25 19	EM ↥Y	CURSOR LEFT	33 5A	!	41 29	)
2	2 02	STX ↥B	START OF TEXT	10 0A	NL *(NEW LINE)*	NEW LINE	18 12	DC2 ↥R	ROLL ON	26 3F	SUB ↥Z	CURSOR DOWN	34 7F	" *(QUOTE)*	42 5C	*
3	3 03	ETX ↥C	ENABLE BLINK	11 0B	VT *(VERT. TAB)*	ERASE LINE	19 13	DC3 ↥S	ROLL OFF	27 27	ESC *(ESCAPE)*	ESCAPE	35 7B	#	43 4E	+
4	4 37	EOT ↥D	INHIBIT BLINK	12 0C	FF *(FORM FEED)*	ERASE PAGE	20 3C	DC4 ↥T	UNDERSCORE ON	28 1C	FS ↥\	DIM ON	36 5B	$	44 6B	, *(COMMA)*
5	5 2D	ENQ ↥E	CURSOR ADDRESS READ	13 0D	RT *(RETURN)*	RETURN	21 3D	NAK ↥U	UNDERSCORE OFF	29 1D	CS ↥]	DIM OFF	37 6C	%	45 60	—
6	6 2E	ACK ↥F	PRINT DONE	14 0E	SO ↥N	BLINK ON	22 32	SYN ↥V	SYNCHRONOUS IDLE	30 1E	RS ↥↑	FUNCTION KEY HEADER	38 5B	&	46 4B	. *(PERIOD)*
7	7 2F	BEL ↥G	BELL	15 0F	SI ↥O	BLINK OFF	23 26	ETB ↥W	CURSOR UP	31 1F	US ↥←	READ CURSOR HEADER	39 7D	' *(APOS)*	47 61	/

octal	06_		07_		10_		11_		12_		13_		14_		15_		16_		17_	
0	48 F0	0	56 F8	8	64 7C	@	72 C8	H	80 D7	P	88 E7	X	96 79	` *(GRAVE)*	104 88	h	112 97	p	120 A7	x
1	49 F1	1	57 F9	9	65 C1	A	73 C9	I	81 D8	Q	89 E8	Y	97 81	a	105 89	i	113 98	q	121 A8	y
2	50 F2	2	58 7A	:	66 C2	B	74 D1	J	82 D9	R	90 E9	Z	98 82	b	106 91	j	114 99	r	122 A9	z
3	51 F3	3	59 5E	;	67 C3	C	75 D2	K	83 E2	S	91 8D	[	99 83	c	107 92	k	115 A2	s	123 C0	{
4	52 F4	4	60 4C	<	68 C4	D	76 D3	L	84 E3	T	92 E0	\	100 84	d	108 93	l	116 A3	t	124 4F	|
5	53 F5	5	61 7E	=	69 C5	E	77 D4	M	85 E4	U	93 9D	]	101 85	e	109 94	m	117 A4	u	125 D0	}
6	54 F6	6	62 6E	>	70 C6	F	78 D5	N	86 E5	V	94 5F	↑ or ^	102 86	f	110 95	n	118 A5	v	126 A1	~ *(TILDE)*
7	55 F7	7	63 6F	?	71 C7	G	79 D6	O	87 E6	W	95 6D	← or _	103 87	g	111 96	o	119 A6	w̄	127 07	DEL *(RUBOUT)*

CHARACTER CODE IN OCTAL AT TOP AND LEFT OF CHARTS.

DELIMITER TABLE

ALL PARITIES OF PUNCTUATION MARKS

WORD 3

	0	6			0			7			1			7		
		!	"					'	(	)			,	–	.	/
10	32	3	4	5	6	7	8	9	40	1	2	3	4	5	6	7
8	40	1	2	3	4	5	6	7	50	1	2	3	4	5	6	7

TABLE:	DECIMAL	OCTAL
0	0-15	0-17
0	16-31	20-37
60717	32-47	40-57
61	48-63	60-77
0	64-79	100-117
34	80-95	120-137
0	96-111	140-157
24	112-127	160-177
0	128-143	200-217
0	144-159	220-237
60717	160-175	240-257
61	176-191	260-277
0	192-207	300-317
34	208-223	320-337
0	224-239	340-357
0	240-255	360-377

FILE POSITIONING

BEGINNING OR END (FLAG IN ?OPEN)

NEXT RECORD OR BYTE BY DEFAULT (REC = 0)
CURRENT POSITION REPORTED BY ?GPOS

REPOSITIONING (?READ, ?WRITE, ?SPOS)

BY RECORD FOR FIXED (?IRN IS RECORD NUMBER)

BY BYTE FOR OTHERS (?IRN IS BYTE COUNT)

FROM BEGINNING = "ABSOLUTE" (?IPST = 1)

FROM CURRENT POSITION = "RELATIVE" (?IPST = 0)

POSITIONING PACKET VALUES

?IPST	?IRNH/L	RESULT
0	0	NEXT RECORD
0	N	N RECORDS (FIXED) OR N BYTES (OTHERS) FROM CURRENT POSITION
I	0	BEGINNING OF FILE
I	-I	END OF FILE
I	N	N RECORDS (FIXED) OR N BYTES (OTHERS) FROM START OF FILE

BYTE DEVICE CHARACTERISTICS

DEFAULT SET BY AOSGEN

DEFAULT CAN BE RESET BY PID 2

OBTAIN OR SET THE CURRENT CHARACTERISTICS (?GCHR ?SCHR)

DESCRIBE SOFTWARE INTERPRETATION OF DATA
RATHER THAN AFFECTING HARDWARE FEATURES
(i.e., YOU CAN'T CHANGE BAUD RATE)

MODES:

TEXT CHARACTERS SUBJECT TO INTERPRETATION
BASED ON EACH DEVICES CHARACTERISTICS

BINARY NO CHARACTER INTERPRETATION DONE

BYTE DEVICES CONTROL CHARACTERS

TEXT MODE ONLY

ALL CHARACTERS PASSED LITERALLY TO PROGRAM EXCEPT

CTRL C – START OF CONTROL SEQUENCE

CTRL D – END OF FILE

CTRL O – DISCARD OUTPUT

CTRL P – LET NEXT CHARACTER THRU

CTRL Q – RESTART OUTPUT

CTRL S – FREEZE OUTPUT

CTRL T – RESERVED

CTRL U – ERASE CURRENT LINE

CTRL V – RESERVED

BYTE DEVICES – CONTROL SEQUENCES

TEXT MODE ONLY

CONTROL C FOLLOWED BY ANY CHARACTER

CONTROL C NEVER PASSED TO PROGRAM

CHARACTER IS PASSED TO PROGRAM ONLY IF

1) NOT A CONTROL CHARACTER

2) CONTROL CHARACTER BETWEEN CONTROL I – CONTROL M (HOR. TAB, NL, VERT. TAB, FF, CR)

IF CHARACTER IS A CONTROL CHARACTER OUTSIDE I – M RANGE IT IS NEVER DELIVERED TO PROGRAM

CONTROL A, B, C, E INITIATE SYSTEM ACTIVITY

OWNERSHIP OF BYTE DEVICES

OWNED BY ONLY ONE DEVICE AT A TIME

GENERIC — PARENT CHOOSES

OWNS WHILE PROCESS IS ACTIVE

EXAMPLE: CLI

OPEN — PROCESS CHOOSES FOR ITSELF

OWNS UNTIL CLOSED

EXAMPLE: EXEC

ASSIGN — PROCESS CHOOSES FOR ITSELF (?ASSIGN)

OWNS UNTIL FREED (?DEASSIGN)

MODULE SYSTEM SUMMARY

SECTION II I/O CREATION AND ACCESS

MODULE TITLE: C: RECORD I/O

?CREATE	CREATE A FILE AND DEFINE RECORD TYPE.
?OPEN	ASSOCIATE A CHANNEL NUMBER WITH A FILE.
?CLOSE	DISASSOCIATE A CHANNEL NUMBER FROM A FILE.
?ASSIGN	ASSIGN A DEVICE TO A PROCESS.
?DEASSIGN	DEASSIGN A DEVICE FROM A PROCESS.
?GCHR, ?SCHR	READ OR SET THE CHARACTERISTICS OF A BYTE DEVICE
?READ/?WRITE	TRANSFER DATA IN UNITS OF A RECORD.
?GPOS, ?SPOS	GET/SET THE POSITION OF A FILE EITHER AS A CHARACTER OFFSET OR AS A RECORD OFFSET (FIXED).

This page intentionally left blank.

MODULE OBJECTIVES

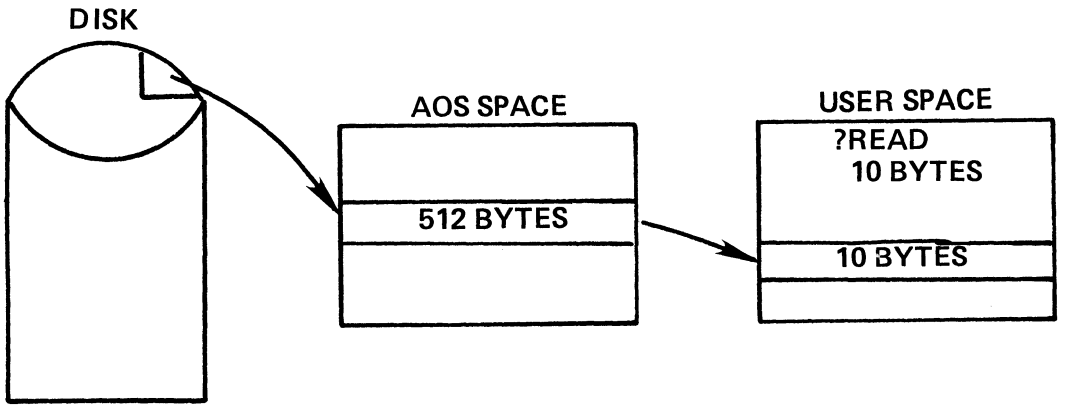
Section II I/O CREATION AND ACCESS

Module Title: D: DIRECT BLOCK I/O

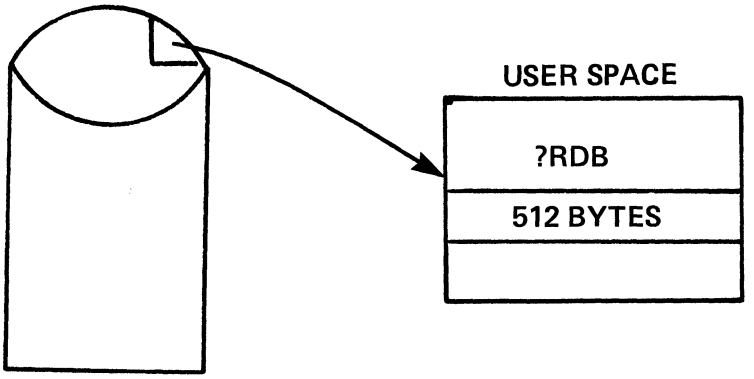
Upon successful completion of this module the student will be able to:

1. Define a magnetic tape structure and access data on the tape.
2. Access on AOS-structured disk file in multiples of 512-byte blocks.
3. Access an arbitrarily-structured disk device in multiples of 512-byte blocks.
4. Identify the technique for writing to a data channel line printer in blocks and passing on VFU control information.

I/O
RECORD TRANSFERS



I/O
BLOCK TRANSFER



BLOCK I/O

FILENAMES AND DEVICENAMES

- DISKS & DISKETTES
- MAGNETIC TAPE FILES, MAGTAPE UNITS
- MCA FILENAME, MCA UNIT
- DATA CHANNEL LINE PRINTER

TRANSFERRING BLOCKS

- DISK – 512 BYTES
- MAG TAPE/MCA – SET ON TRANSFER

BUFFERING DONE BY USER

SAME GENERAL PROCEDURE FOR
CREATION AND ACCESS AS FOR RECORD I/O

- ?GOPEN
- ?WRB
- ?RDB
- ?GCLOSE

DISK FILES

STRUCTURE

- 512 BYTE DISK BLOCKS
- FILE ELEMENTS
- INDEX LEVELS

DESIGN AND ACCESS

- LARGE FILE ELEMENTS -
  - CONTIGUOUS AREAS
  - QUICK ACCESS
- SMALL FILE ELEMENTS -
  - MORE DISK ACCESSES
  - INDEX STRUCTURE
  - EFFICIENT USE OF DISK SPACE

DISKS

FILE VS. DEVICE I/O

FILE I/O – ASSUMES VALID AOS

DIRECTORY & FILE STRUCTURE

?INIT DEVICE

?GOPEN FILE

?WRB/?RDB DATA BLOCKS WITHIN FILE

DEVICE I/O – ANY FORMAT

?GOPEN DEVICE

?WRB/?RDB ANY BLOCK ON DEVICE

MAGNETIC TAPE ACCESS

OPENING

TO A SPECIFIC FILE NUMBER

TO THE START OF THE DEVICE

POSITIONING

BLOCK COUNT ZERO

GIVE BLOCKNUMBER WHEN READING OR WRITING

POSITIONING TO FILENUMBER 0 = REWIND

TRANSFERRING DATA

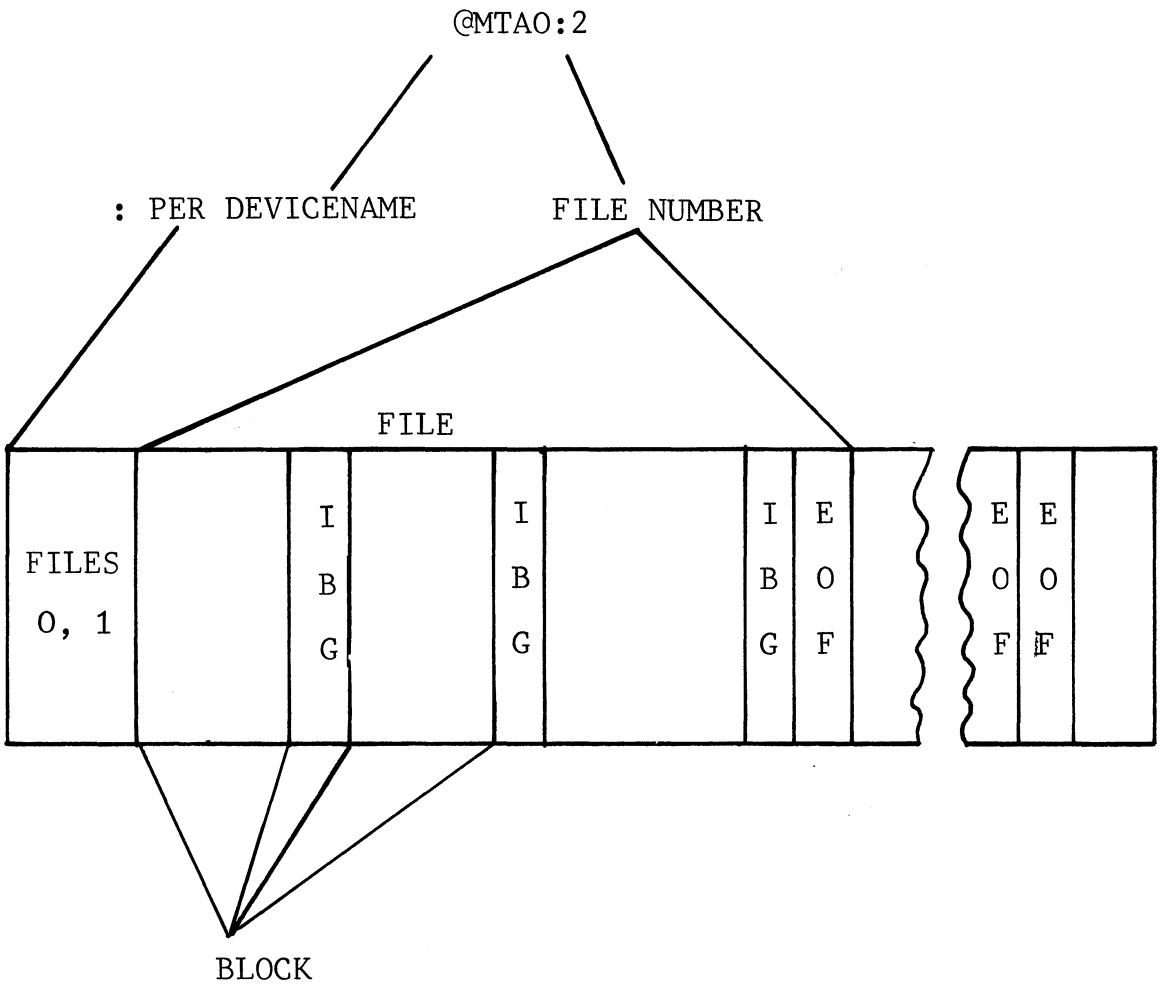
RECORD:

BLOCKSIZE SET AT OPEN

BLOCK:

BLOCKSIZE SELECTED IN EACH TRANSFER

MAGNETIC TAPE FILES



MAGNETIC TAPE STRUCTURE

MODULE SYSTEM CALL SUMMARY

SECTION II I/O CREATION AND ACCESS

MODULE TITLE: D: DIRECT BLOCK I/O

?GOPEN	OPENS A FILE AND ASSOCIATES A CHANNEL NUMBER FOR BLOCK I/O
?GCLOSE	CLOSES AND DISASSOCIATES A FILE USED FOR BLOCK I/O.
?RDB, ?WRB	TRANSFERS DATA IN MULTIPLE, OF 512–BYTE BLOCKS

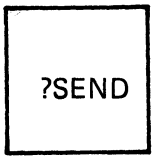
MODULE OBJECTIVES

Section II I/O CREATION AND ACCESS

Module Title: E: IPC FACILITY

Upon successful completion of this module the student will be able to:

1. Use ?SEND to transmit a message from a program to a console.
2. Define the procedures for creating and opening an IPC file as a communications device between different processes.
3. Define the procedures needed to communicate between two processes using the primitive IPC facility.



PROCESS ⟶ @CONSOLE DEVICE OF ANY PROCESS

255 BYTES MAXIMUM

IDENTIFIED AS BEING FROM ?SEND BY

FROM PID XXX: MESSAGE ON THE RECEIVING CONSOLE

SUPRESS MESSAGE DISPLAY THRU CHARACTERISTICS

FILE IPC

COMMON IPC-TYPE FILE SHARED BETWEEN PROCESSES

PROCEDURE

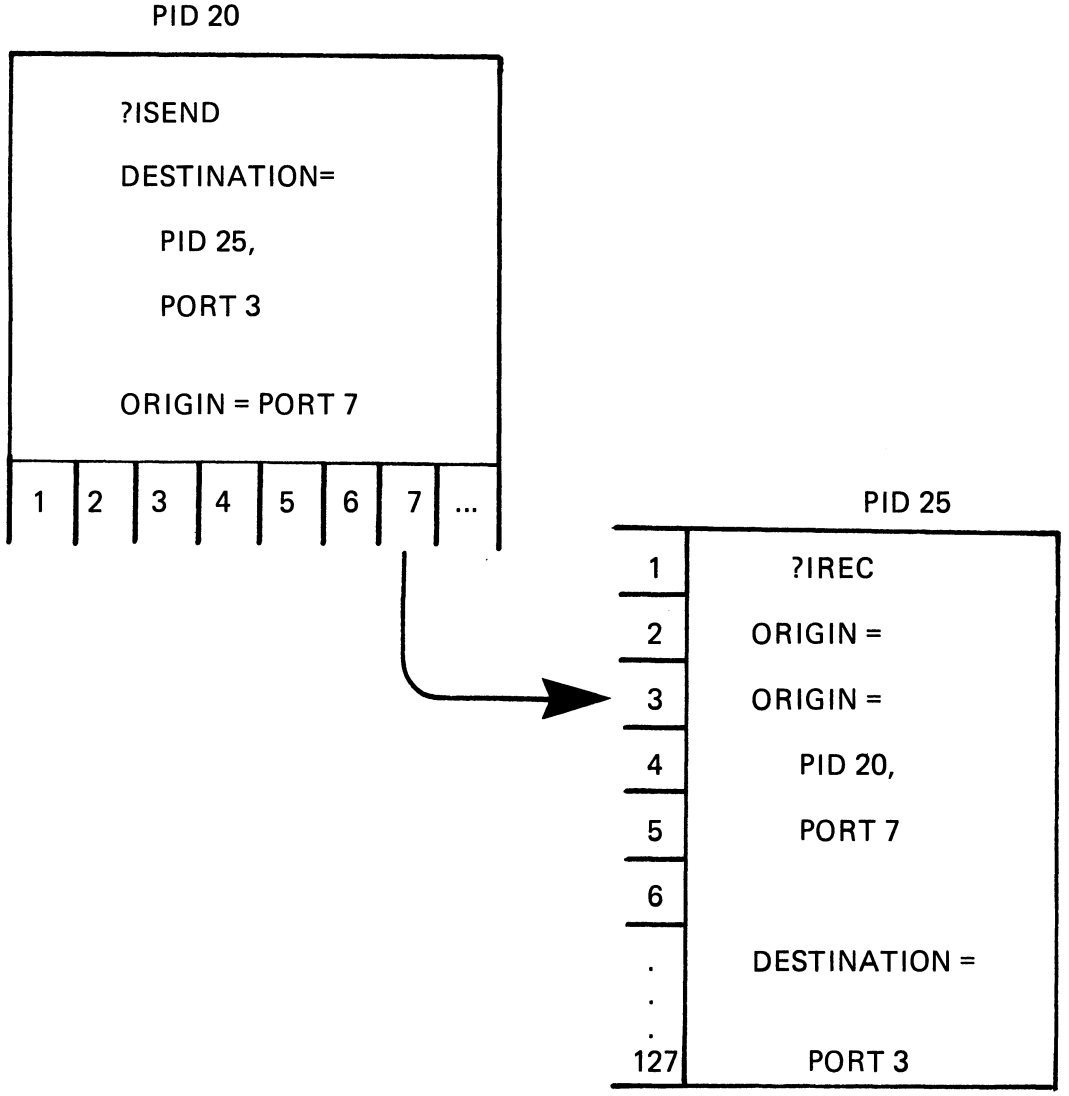
RECEIVER CREATES

SYNCHRONIZE OPENS

READ/WRITE AS IF IT WERE
A DEVICE

CLOSE (SYNCHRONIZED)

PRIMITIVE IPC



HEADERS

SYSTEM FLAGS –

LOOPING – SENDING A MSG TO ONESELF

SPOOLING – HOLDING MSG TILL ?IREC IS DONE

USER FLAGS – 16–BIT MESSAGE

PORTS DEFINE THE COMMUNICATIONS LINK

MESSAGE LENGTH – CAN BE ZERO

MESSAGE ADDRESS

THE TRANSMITTER'S HEADER ALWAYS OVERWRITES THE RECEIVER'S HEADER EXCEPT FOR BUFFER ADDRESS

RECEIVER HEADER CAN SPECIFY ZERO ORIGIN PORT TO ACCEPT MESSAGE FROM ANYONE

RECEIVER CAN SPECIFY 0 DESTINATION PORT TO ACCEPT MESSAGE SENT TO ANY OF ITS PORTS

?TPORT

FILE CREATION NOT NECESSARY

RETURNS A GLOBAL PORT # TO THE SAME PROCESS

THAT OWNS THE LOCAL PORT #

ONLY PROBLEM IS NOTIFYING OTHER PROCESSES
OF GLOBAL PORT #

SHARED MEMORY
COMMON DISK FILE

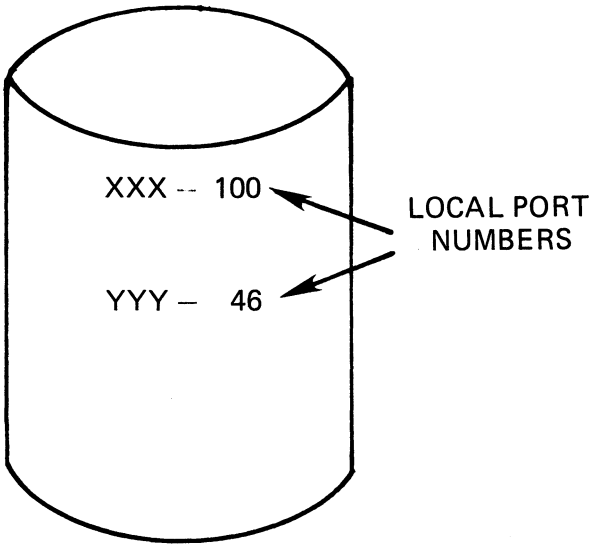
LOOKING UP A GLOBAL PORT NUMBER

TARGET PROCESS

?CREATE IPC FILE XXX

?ILKUP ON YYY

?IREC



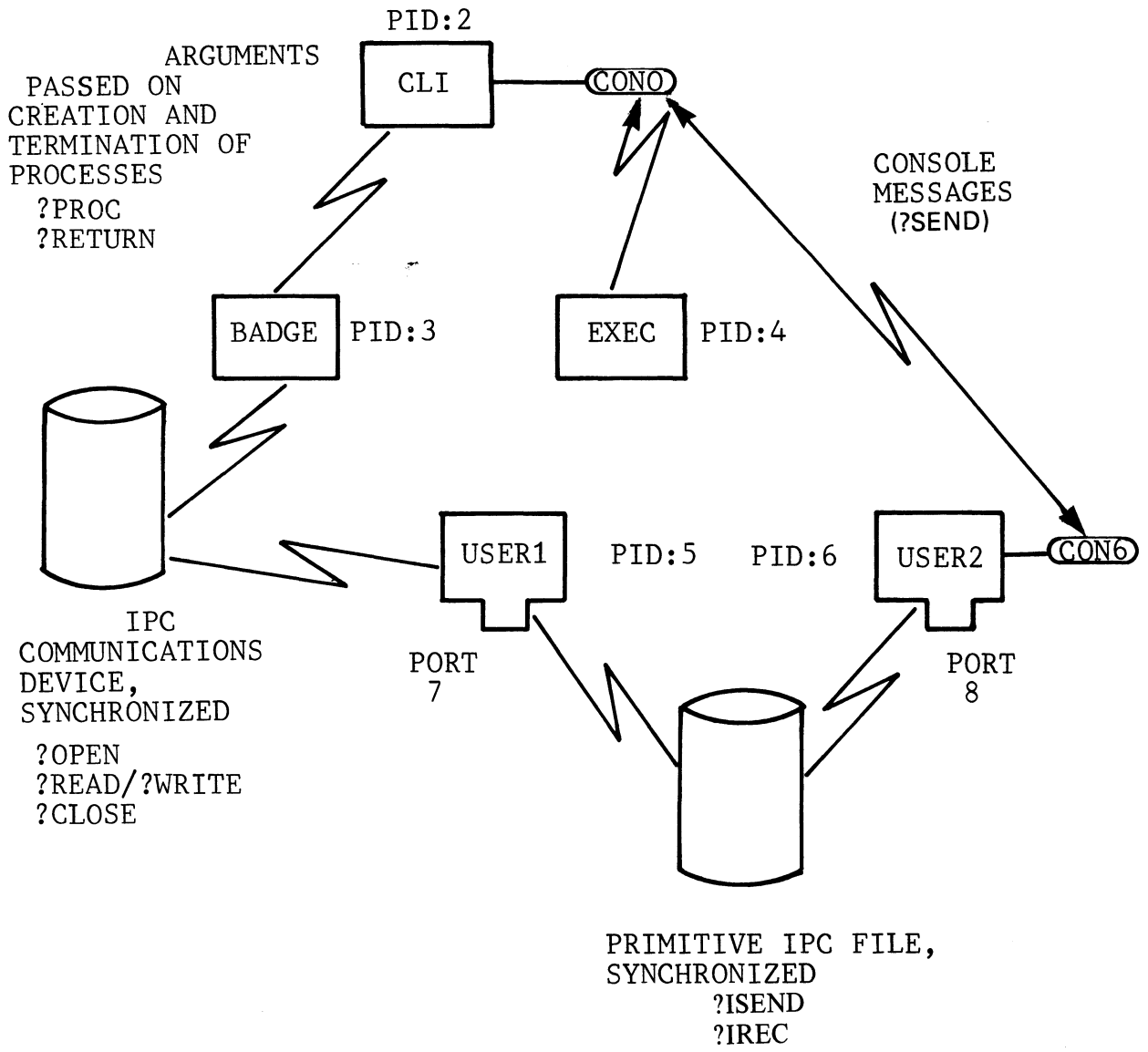
SENDING PROCESS

?CREATE IPC FILE YYY

?ILKUP ON XXX

?ISEND

IPC CONCEPTS



MODULE SYSTEM CALL SUMMARY

SECTION II I/O CREATION AND ACCESS

MODULE TITLE: E: IPC

?SEND	SEND A MESSAGE TO A CONSOLE
?CREATE	CREATE AN IPC–TYPE FILE
?OPEN	USE THE ?IIPC FLAG TO OPEN A FILE FOR DEVICE COMMUNICATIONS
?READ/WRITE	ACCESS A FILE BEING USED AS A COMMUNICATIONS DEVICE
?IREC, ?ISEND	RECEIVE OR SEND A MESSAGE USING THE PRIMITIVE IPC FACILITY
?GPORT	FIND THE OWNER OF A PORT
?TPORT	TRANSLATE A LOCAL PORT NUMBER TO A GLOBAL PORT NUMBER
?ILKUP	DETERMINES THE PORT NUMBER FOR AN IPC–TYPE FILENAME
?GCPN	GETS THE PORT NUMBER OF A PROCESS CONSOLE

This page intentionally left blank.

MODULE OBJECTIVES

Section III SCHEDULING

Module Title: A: PROCESSING

Upon successful completion of this module the student will be able to:

1. Create and terminate a process.
2. Describe the major areas of the process control block.
3. By using system calls describe how a process can control a subordinate process.
4. Design and implement a program that synchronizes and communicates between more than one process, and defend your choice of the available options.
5. Describe the overhead involved in scheduling processes, particularly with respect to the use of the MAP unit.

PROCESS SCHEDULING

USED BY AOS TO DETERMINE CPU ALLOCATION FOR PROCESSES

HIGHEST PRIORITY ELIGIBLE PROCESS RUNS

PRIORITY ENQUEUE FACTOR (PNQF)

TYPE	PRIORITY
RESIDENT AND PREEMPTIBLE	ASSIGNED (1-255)
SWAPPABLE	DERIVED (100000 + PRI + (4*BR))

BEHAVIOR

PROCESSES OF EQUAL PRIORITY SCHEDULED ON A ROUND ROBIN

PROCESS CREATION

ALL APPLICATION PROGRAMS MUST RESIDE IN THE BOUNDARIES OF A PROCESS (?PROC)

MAXIMUM OF 64 WORKING PROCESSES

PROCESS PARAMETER PACKET INCLUDES

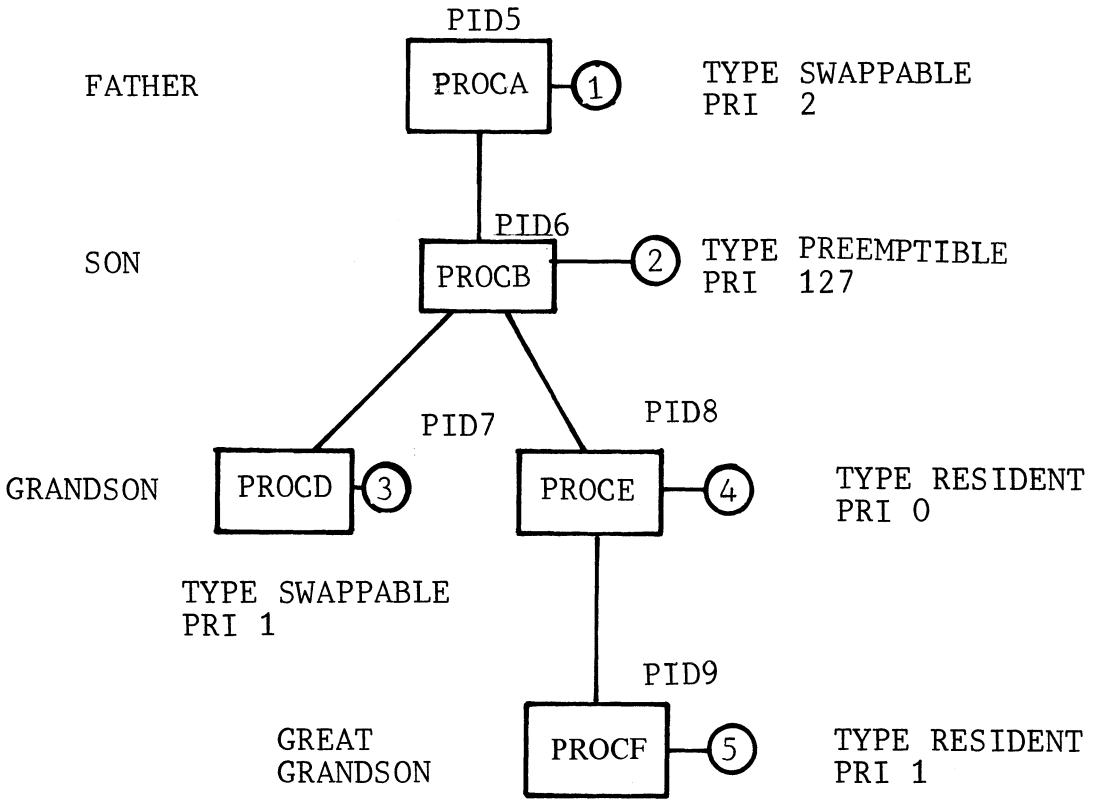
- PRIVILEGES/LIMITATIONS
- PROGRAM PATHNAME
- MEMORY SIZE
- PRIORITY
- TYPE
- MAXIMUM NUMBER OF CONCURRENT SYSTEM CALLS
- MAXIMUM NUMBER OF SON PROCESSES THAT CAN BE CREATED
- GENERIC FILES
- DEFAULT VALUES

PROCESS TERMINATION

A PROCESS CAN TERMINATE ITSELF (?TERM)
OR ANY SUBORDINATE PROCESS

1. PROCA – CAN TERMINATE ANY PID IN THE TREE
2. PROCB – CAN TERMINATE ANY PID EXCEPT PID5
3. PROCD – CAN ONLY TERMINATE SELF, PID7
4. PROCE – CAN TERMINATE PID8 OR PID9
5. PROCF – CAN ONLY TERMINATE SELF, PID9

SUPER PROCESS HAS ABILITYTO TERMINATE ANY PROCESS
IN HIERARCHY



PROCESS TABLE ACCESS

NECESSARY WHENEVER A SYSTEM CALL NEEDS A PROCESS NAME OR PID

"PROCESS NAME" FOR USERNAME JONES

FULL: JONES: PROCA

SIMPLE: PROCA – SYSTEM SUPPLIES JONES

PROCESS CONTROL PARAMETER PACKETS INCLUDE THE ABILITY TO:

- RETURN ANY FATHER PROCESSES PID (?DADID) BY GIVING SON'S PID
- RETURN A PROCESS USERNAME (?GUNM)
- RETURN A PROCESS NAME AND PID (?PNAME)
- RETURN A PATHNAME OF A PROGRAM (?GPRNM)

MORE EFFICIENT ACCESS BY USING PID

PROCESS CONTROL BLOCK

AOS MAINTAINS DATABASES ON
QUEUES FOR SCHEDULING

DATABASE TYPES:

- PROCESS TABLES
- CONTROL BLOCKS

.ELQUE:ELIGIBLE QUEUE = CB's AND PTBL's

ORDER:

- CORE MANAGER CB ALWAYS FIRST ON THE QUEUE (CMTSK)
- CONTROL BLOCKS, ORDERED FIFO
- PTBL's, ORDERED BY PNQF
- RTPTB, CIRCLE LINKED TO START OF ELQUE

.IESWP: INELIGIBLE SWAPPABLE PROCESS

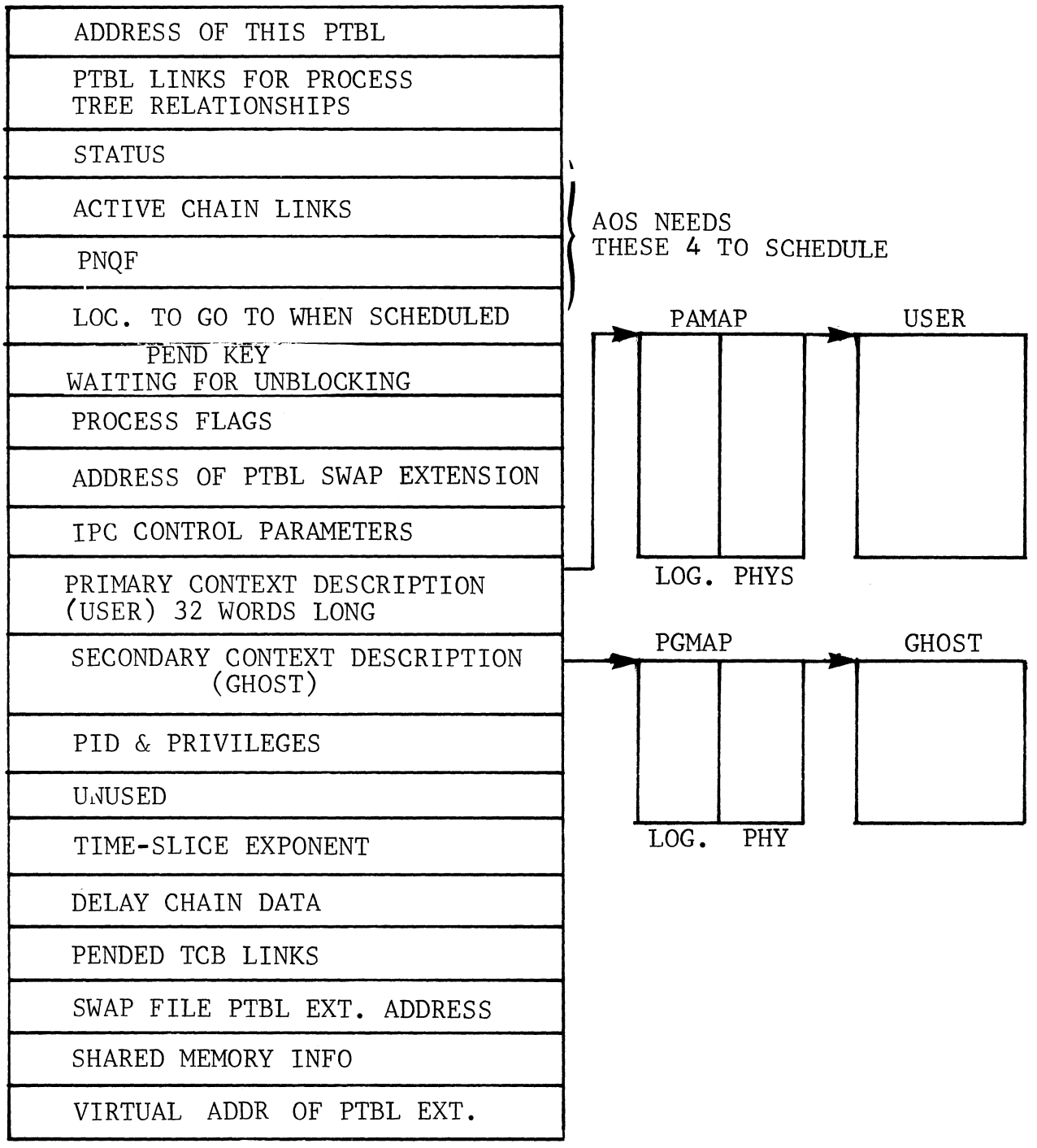
.IERES: INELIGIBLE PRE-EMPTIBLE PROCESSES

.BLKQ: BLOCKED PROCESSES

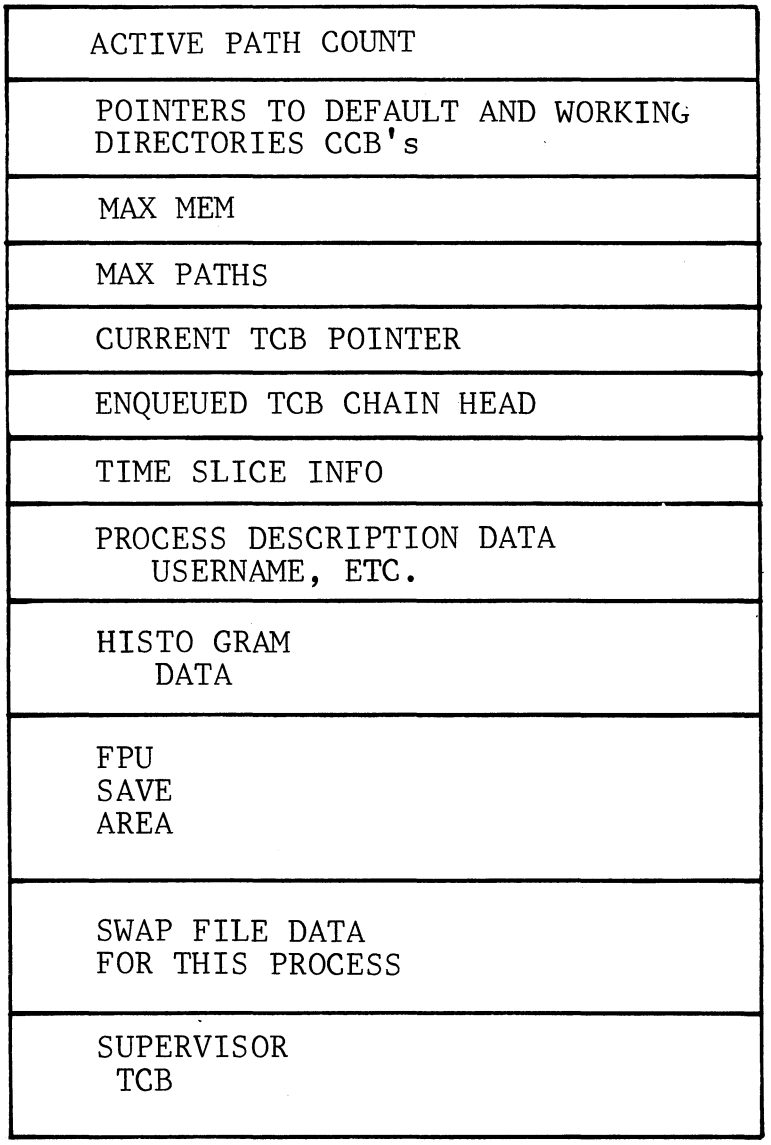
.CMQUE: PROCESSES TO BE SWAPPED IN OR OUT

SCHEMATIC OF PROCESS TABLE RELATIONSHIPS

RESIDENT PORTION



SWAPPABLE PORTION OF PTBL (SWAPPED WITH A PROCESS TO DISK)



IN AOS HYPERSPACE

This page intentionally left blank

PROCESS CONTROL

GENERAL CHARACTERISTICS

- BLOCKING/UNBLOCKING PROCESSES (?BLKPR/?UBLPR)
- PROCESSES CAN CHANGE TYPE OR PRIORITY (?CTYPE) (?PRIPR)
  - SWAPPABLE PREEMPTIBLE RESIDENT
  - PRIORITY CHANGES ALLOW SHIFTS IN SCHEDULING
- PROGRAM CHAINING (?CHAIN)
  - SAVES ON RESOURCE ALLOCATION
  - USED ALL SONS
- TERMINATE A PROCESS AND CREATE (?BRKFL) A BREAK FILE
- (?SUPROC)

  SUPERPROCESS MODE

  OVERRIDE PROCESS HIERARCHY

SYNCHRONIZATION AND COMMUNICATION

- COMMON DISK FILES AS A COMMUNICATION DEVICE
  - NO SYNCHRONIZATION INVOLVED
  - FOLLOWS THE STANDARD I/O CYCLE
  - PRIVILEGES NOT NECESSARY
- FILE IPC AS A COMMUNICATION DEVICE
  - FULL DUPLEX COMMUNICATION LINK
  - SYNCHRONIZATION AND PORT MATCHING HANDLED BY AOS
  - FOLLOWS THE STANDARD I/O CYCLE
  - ?PVIP PRIVILEGE NOT NEEDED

INTERPROCESS COMMUNICATION AND SYNCHRONIZATION

- IPC PRIMITIVES
  - SOPHISTICATED PROCESS SYNCHRONIZATION
  - FREE FORMAT MESSAGES
  - MUST HAVE ?PVIP PRIVILEGE
- SYSTEM FLAGS DIRECT SYSTEM ACTION
  - MESSAGE SPOOLING
  - ARBITRARY ORDERING OF SEND OR RECEIVE

SENDING INITIAL IPC MESSAGES

FROM A USER PROCESS:

- ANY LENGTH MESSAGE
- ANY FORMAT
- PASSED THRU THE ?PROC AND ?CHAIN

INITIAL CLI MESSAGE INFORMATION

- EDITED VERSION OF ORIGINAL CLI COMMAND
- COMMAND TREE
- CLI FILE INFORMATION
- SET BY ARGUMENTS TO PROCESS OR XEQ COMMAND

?GTMES PARAMETER PACKET INCLUDES

- REQUEST TYPE
- ARGUMENT OR PREDEFINED VARIABLE NUMBER
- BYTE POINTER TO SWITCH
- BYTE POINTER TO MESSAGE BUFFER

RECEIVING INITIAL IPC MESSAGE (?GTMES)

FROM A USER PROCESS:

ONLY OPTION IS ENTIRE MESSAGE

FROM CLI:

- ENTIRE MESSAGE (CRYPTIC)
- LET ?GTMES ANALYZE CLI FORMAT
  - JUST COMMAND LINE
  - ARGUMENT COUNT
  - EXTRACT AN ARGUMENT
  - TEST FOR A SWITCH
  - REPORT ALL SWITCHES

SENDING A FINAL IPC MESSAGE

?RETURN

- CAN SEND
  - ERROR CODE
  - MESSAGE
  - FLAGS
    - CLI FORMAT & INTERPRETATION
    - FREE FORMAT

?TERM

- ONLY WHEN IT TERMINATES ITSELF
- CAN SEND MESSAGE (CAN INCLUDE ERROR CODE)
- FLAGS IN USER FLAG SECTION OF HEADER

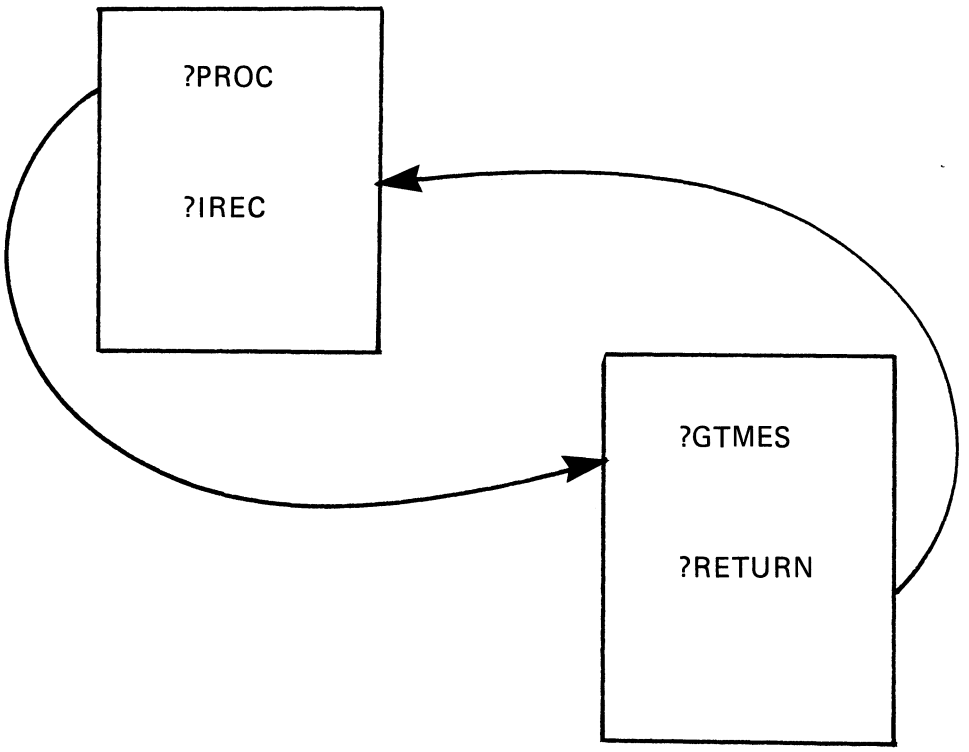
AOS GENERATED MESSAGES

- TRAPS
- CONSOLE INTERRUPTS
- ?TERM FROM ANOTHER PROCESS
- CERTAIN SERIOUS ERRORS

RECEIVING A FINAL IPC MESSAGE

?IREC
ON GLOBAL PORT
?SPTM

USER FLAGS SHOW
FORMAT OF MESSAGE



MODULE SYSTEM CALL SUMMARY

Section III SCHEDULING

Module Title: A: PROCESSING

?PROC	-	Create a Process
?TERM	-	Terminate a process
?DADID	-	Get a father PID
?GUNM	-	Get a process username
?PNAME	-	Get a full process name
?GPRNM	-	Get the full pathname of an initial program associated with a PID
?BLKPR	-	Block a process
?UBLPR	-	Unblock a process
?CTYPE	-	Change a process type
?PRIPR	-	Change a process priority
?BRKFL	-	Terminate a process and create a break file
?SUSER	-	Enter or leave superuser mode
?OPEN	-	Open a disk file
?READ	-	Read a file
?WRITE	-	Write to a file
?ISEND	-	SEND on IPC message
?IREC	-	Receive on IPC message
?RETURN	-	Return to a father process
?GTMES	-	Get the CLI message
?CHAIN	-	Run a new program under current process

MODULE OBJECTIVES

Section III SCHEDULING

Module Title: B: TASKING

Upon successful completion of this module the student will be able to:

1. Design and implement a re-entrant multitasking program.
2. Distinguish the differences between AOS's scheduling of a process and a task.
3. Describe the major areas of the Task Control Block
4. Describe how the TCB's are used by the task scheduler.

TASK CONCEPTS

TASK

- LOGICALLY COMPLETE, ASYNCHRONOUS LOCUS OF CONTROL THROUGH A PROGRAM
- BASIC PROGRAM UNIT CAPABLE OF USING SYSTEM RESOURCES

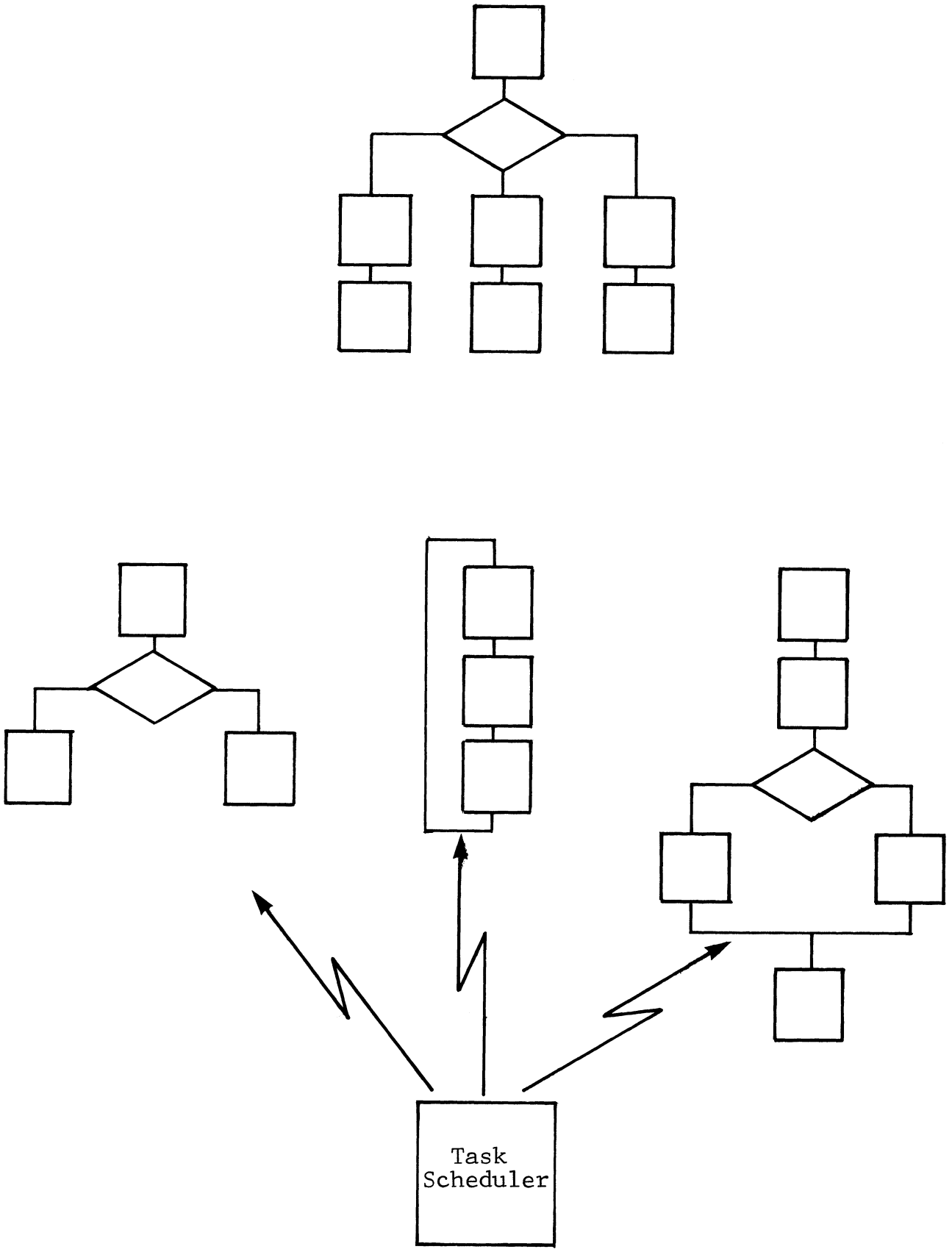
PROGRAM

- CURRENT EXECUTABLE CONTENTS OF A PROCESS'S ADDRESS SPACE
- CONTAINS ONE OR MORE TASKS

MUTI-TASKING

- WHEN PROGRAMS CONTAIN MORE THAN ONE TASK

SINGLE VS. MULTI TASK ENVIRONMENT



TASKS

A TASK

- HAS A FIXED ID NUMBER, UNIQUE WITHIN A PROGRAM
- CAN EXIST IN ONE OF FOUR STATES:

  EXECUTING

  READY

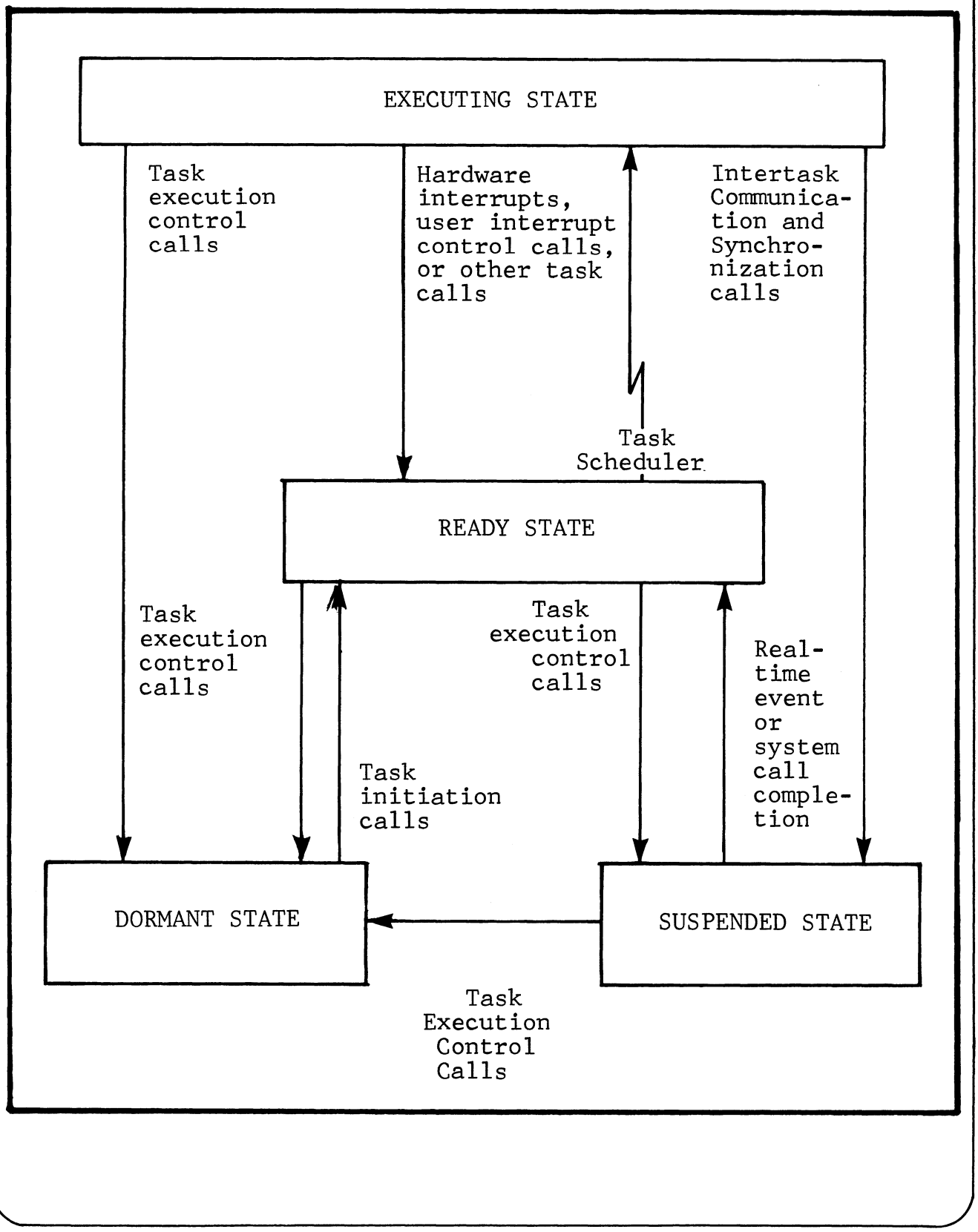
  SUSPENDED

  DORMANT

- HAS A PRIORITY FROM 0 (HIGH) TO 255 (LOW), WHICH MAY BE CHANGED DURING EXECUTION

A PROGRAM MAY CONTAIN UP TO 32 TASKS

TASK STATE TRANSITION



TASK CONTROL

TASK CALLS ARE AVAILABLE TO

- CREATE/TERMINATE/READY/SUSPEND A TASK OR TASKS
- SEND/RECEIVE MESSAGES BETWEEN TASKS
- CONTROL TASK TIMING

THE TASK MONITOR

- MAINTAINS INFORMATION ABOUT EACH TASK IN A TCB
- GIVES CONTROL TO THE HIGHEST PRIORITY READY TASK

TASK INITIATION CALLS (?TASK)

- USED TO INITIALIZE ONE OR MORE TASKS
- CALL CAN RESIDE IN ANY USER CONTEXT
- ALL TASKS EXCEPT DEFAULT TASK MUST BE INITIALIZED

?UTSK ROUTINE

- GIVEN CONTROL BEFORE TASK
- DUMMY ROUTINE IN URT.LB
- CAN BE DESIGNED AND USED BY USER PROGRAM

EXTENDED PACKET

- DELAYS TASK INITIATION
- CAN REPEAT INITIATIONS PERIODICALLY
- ONE TCB USED BY "QUEUE MANAGER" TASK FROM URT.LB
- MANAGER MUST BE INITIATED (?IQTS) BEFORE EXTENDED ?TASK

TDP

TASK DEFINITION PACKET

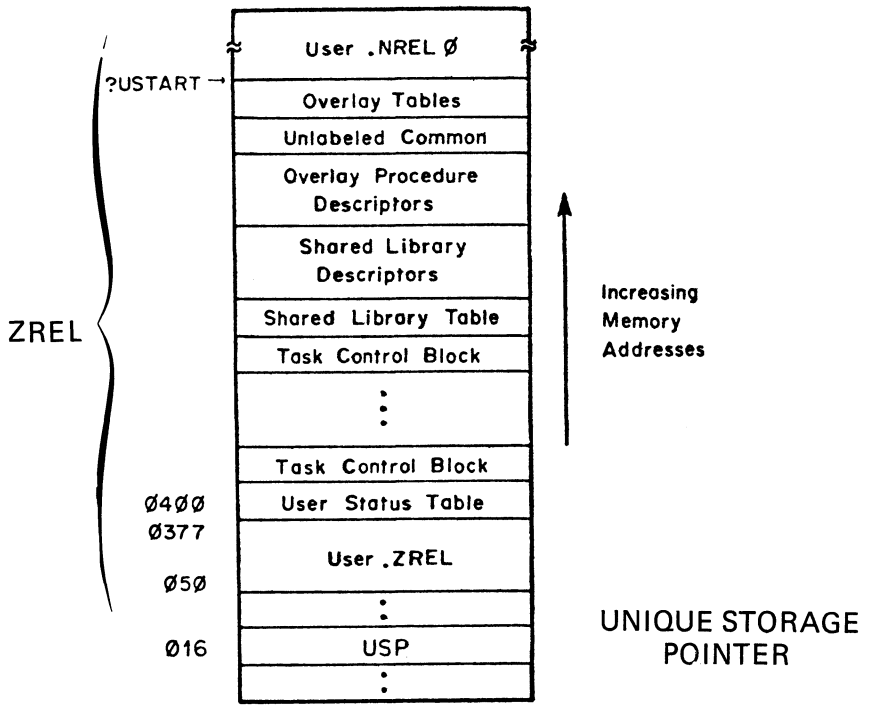
. STANDARD PACKET DEFINED:

- SINGLE OR MULTIPLE TASK ASSIGNMENTS
- START ADDRESS
- MESSAGES
- PRIORITY AND ID
- STACK BASE AND SIZE

. EXTENDED PACKET DEFINES ALL OF ABOVE PLUS:

- TIME OF DAY FOR TASK CREATION
- NUMBER OF TIMES FOR CREATION TO OCCUR
- ELAPSED TIME BETWEEN CREATION
- FOR TASK QUEING A TASK QUEUE MANAGER MUST BE DEFINED
- TASKS CAN BE DEQUEUED IF THEY ARE NOT ACTIVE

USER PAGE ZERO



USER STATUS TABLE

? Offset Mnemonic	Contents
USTEZ	Number of memory words to be saved for each task (consult the discussion of TCB offset ?TELN).
USTES	Starting address of task area to be saved. (also see TCB offset ?TELN).
USTSS	Reserved for system use.
USTSE	Reserved for system use.
USTS1	Reserved for system use.
USTS2	Reserved for system use.
USTDA	Reserved for system use.
USTFL	Reserved for system use.
USTSL	Address of start of Shared Library Directory
USTIT	Starting address of ?INTWT task's TCB.
USTRV	Revision level of program.
USTTC	Number of tasks allocated for this program.
USTCT	Address of currently active task's TCB.
USTAC	Starting address of active TCB queue.
USTFC	Starting address of inactive TCB queue. Dormant tasks
USTBL	Number of 2048-byte unshared memory paged used by the program.
USTOD	Address of first entry in primitive Overlay Node Table.
USTST	Number of the first shared memory page.
USTSZ	Number of 2048-byte shared memory pages used by the program.

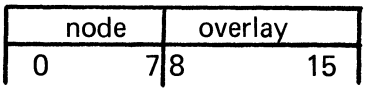
TASK CONTROL BLOCK

E30–PARU

Offset Mnemonic	Contents
?TLNK	Address of the next TCB in this queue.
?TSTAT	Task event flags; setting any of these places the task into the suspended state:
?TSPN, ?TSIG	Task has issued a system call whose execution is taking place in system space.
?TSSG	Task is waiting for an overlay area or a shared routine; or the task has issued an ?XMTW call and the message has not been received; or the task is awaiting a message (?REC).
?TSSP	Task has been suspended by one of the following calls: ?IDSUS, ?PRSUS, or ?SUS.
?TSRC	Task has issued ?TRCON and is awaiting a message from the console.
?TSIW	System action is occurring in preparation for the task's execution.
?TSGS, ?TSAB	(Flags used by operating system).
?TSUF	(Flag used by high level languages.)
?TSP	Stack pointer.
?TFP	Frame pointer.
?TSL	Stack limit.
?TSO	Address of stack fault handler.
?TAC0	Contents of AC0.

Offset Mnemonic	Contents
?TELN	Starting address of this task's memory save area (or 0 for no save area). If there is a save area, it must be the same size as the save area specified in UST offsets USTEZ and USTES.
?TFPS	Starting address of an 18-word area used to save this task's floating point unit state. System call ?IFPU stores this address in ?TFPS.
?TCUD	Description of this task's current general resource (if any). Four descriptors are possible:
BITS IN ?TSTAT	1) bit 0-15=0, no general resource is in use. 2) bit 0=0, bits 1-15 contain an address which equals or exceeds ?USTART. This is the address of a root procedure entry. 3) bit 0=0, bits 1-15 contain an address less than ?USTART. This address points to a two-word overlay procedure descriptor:

?OVEDS



?OVEOF entry's offset into the overlay.

4) bit 0=1, bit 1=0, bits 2-15 contain the address of a library entry descriptor defined as follows:

?SLEDS	(used by the system).
?SLEOF	entry offset into the shared routine.
?TIDPR	left byte: task I.D. right byte: task priority.
?TSLK	(Used by the system.)

TASK CONTROL BLOCK (Cont'd)

?TAC1	Contents of AC1.
?TAC2	Contents of AC2.
?TAC3	Contents of AC3.
?TPC	Bit 0: state of Carry; Bits 1-15: contents of PC.
?TUSP	Contents of location 16, the Unique Storage Position (USP). Manual stack printer (old)
?TKAD	Address of kill processing routine specified in ?KILAD call by this task.
?TGEX	(Used by the system.)

TASK TERMINATION

* KILLING A TASK=EMPTY A TCB

  - SUICIDE (?KILL)
    - NO ERRORS
    - NO RETURN

* IN CASE OF ATTACK BY NEXT TWO CALLS, DEFEND FIRST WITH ?KILAD: SETS ADDRESS OF ROUTINE TO REPLACE ONE KILL ATTEMPT

  - HOMICIDE (?IDKIL)
    - KILL BY ID
    - TASK COULD KILL SELF WITH OWN ID

  - GENOCIDE (?PRKIL)
    - KILL BY PRIORITY
    - TASK COULD KILL SELF WITH OWN PRIORITY

CHANGING PRIORITIES AND STATES

- TASK PRIORITY CHANGE

  CALLING TASK CAN:

  1. CHANGE ITSELF (?PRI)
  2. CHANGE TASK PRIORITY OF A (?IDPRI) GIVEN I.D.

- TASK SUSPENSION

  CALLING TASK CAN:

  1. SUSPEND ITSELF (?SUS)
  2. SUSPEND BY I.D. (?IDSUS)
  3. SUSPEND BY PRIORITY (?PRSUS)

- READY A TASK

  CALLING TASK CAN:

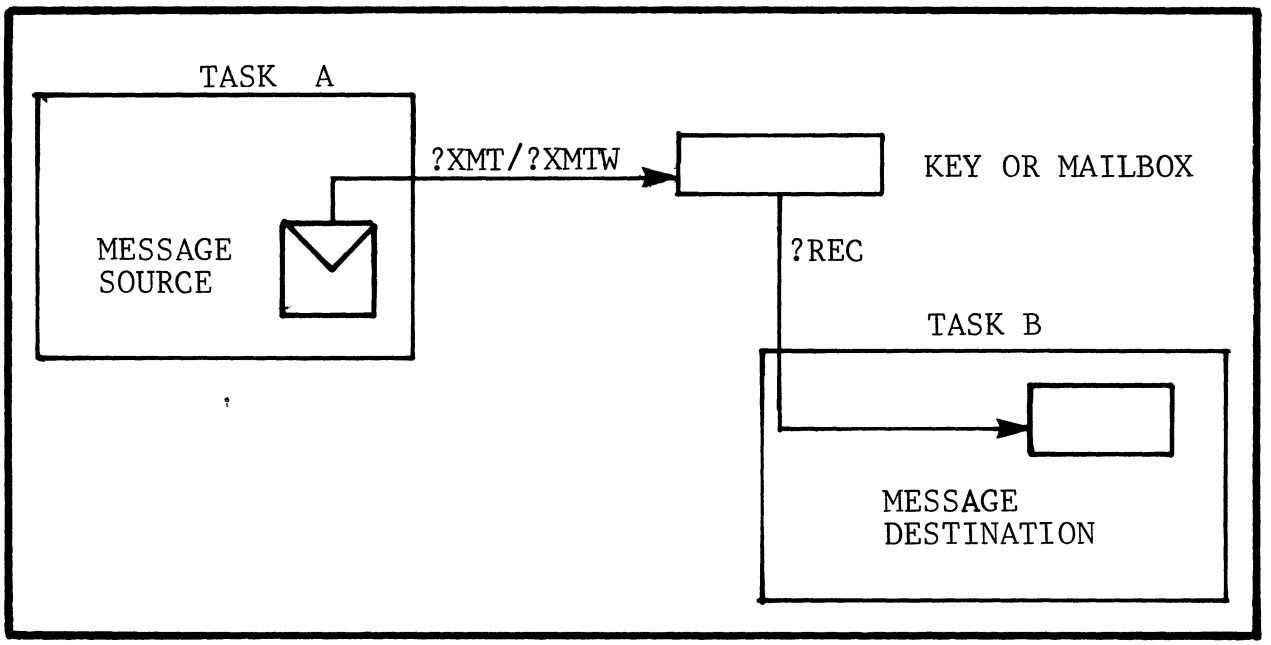
  1. READY BY I.D. (?IDRDY)
  2. READY BY PRIORITY (?PRRDY)

INTER-TASK COMMUNICATION

- TASKS CAN TRANSMIT AND RECEIVE ONE WORD MESSAGES

  MESSAGES CAN BE:

  - BROADCASTED
  - USED FOR SYNCHRONIZATION
  - USED TO PROTECT CRITICAL DATA REGIONS



MODULE SYSTEM CALL SUMMARY

Section III SCHEDULING

Module Title: B: TASKING

?TASK	-	Activate one or more tasks
?KILL	-	Kill calling task (self termination)
?KILAD	-	Define a kill processing routine
?PRI	-	Change Priority of calling task
?IDSTAT	-	Get the status of a task
?IQTSK	-	Create a queued task manager
?OUKIL	-	Exit on overlay and kill task
?*TOD	-	Get/Set time of day
?*DAY	-	Get/Set date
?TRCON	-	Read a task message
?XMT	-	Transmit a message
?REC	-	Receive a message
?XMTW	-	Transmit a message and suspend until a receive
?PRSUS	-	Suspend all tasks of a given priority
?PRRDY	-	Ready all tasks of a given priority
?PRKIL	-	Kill all tasks of a given priority
?DELAY	-	Delay a task
?INTWT	-	Define a console interrupt task
?GHRZ	-	Get the frequency of the real time clock
?IDGOTO	-	Redirect a task
?IDKIL	-	Kill a task (by ID, etc.)
?IDPRI	-	Change the priority of a task by ID.
?DRSCH	-	Disable scheduling
?ERSCH	-	Enable scheduling
?DQTSK	-	Deque one or more tasks
?LFED	-	LFE mode disable
?LFEE	-	LFE mode enable
?LFES	-	LFE mode status

MODULE OBJECTIVES

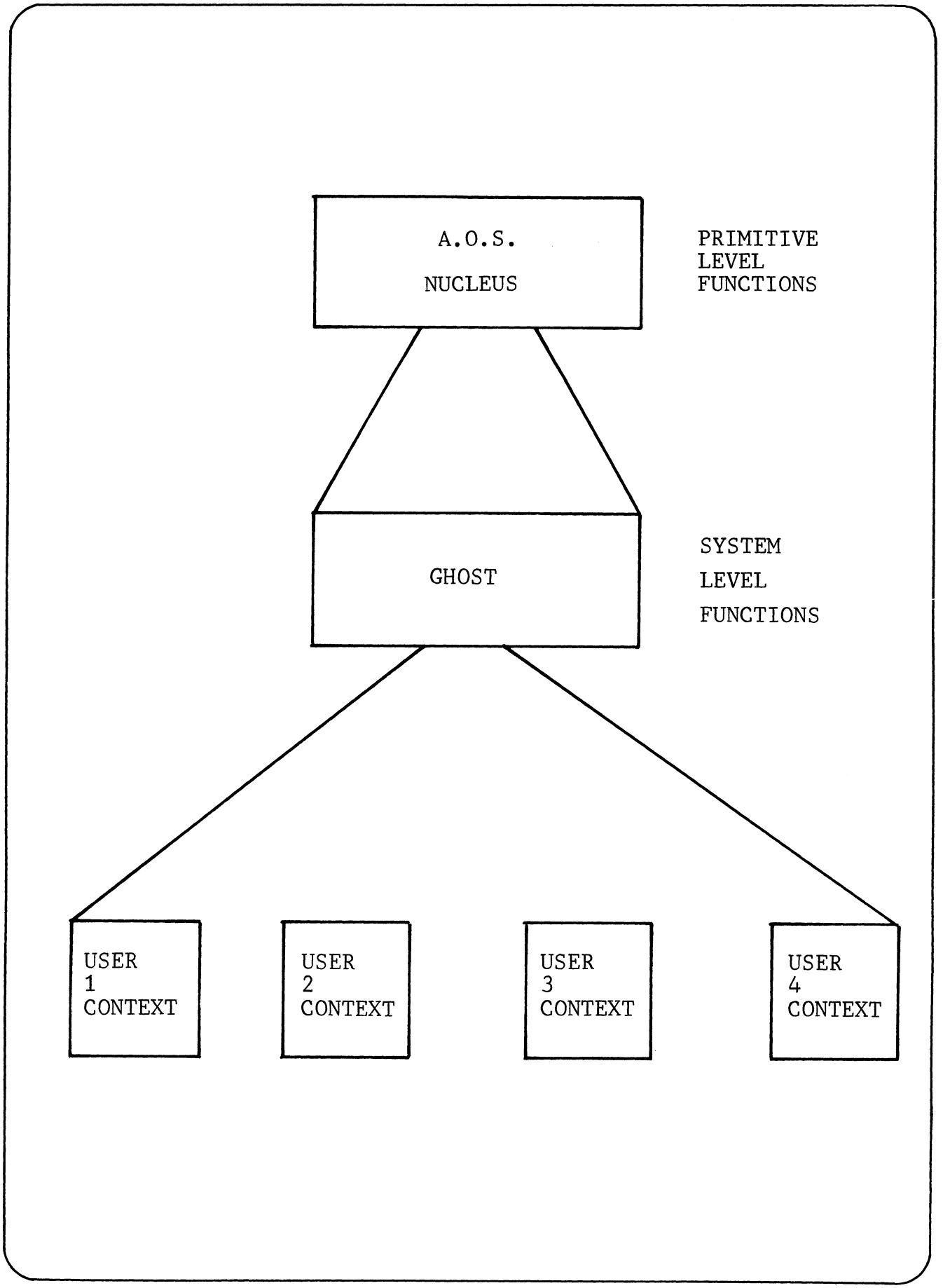
Section III SCHEDULING

Module Title: C: GHOST

Upon successful completion of this module the student will be able to:

1. Describe the interaction between the Ghost and AOS, and the Ghost and User Processes.

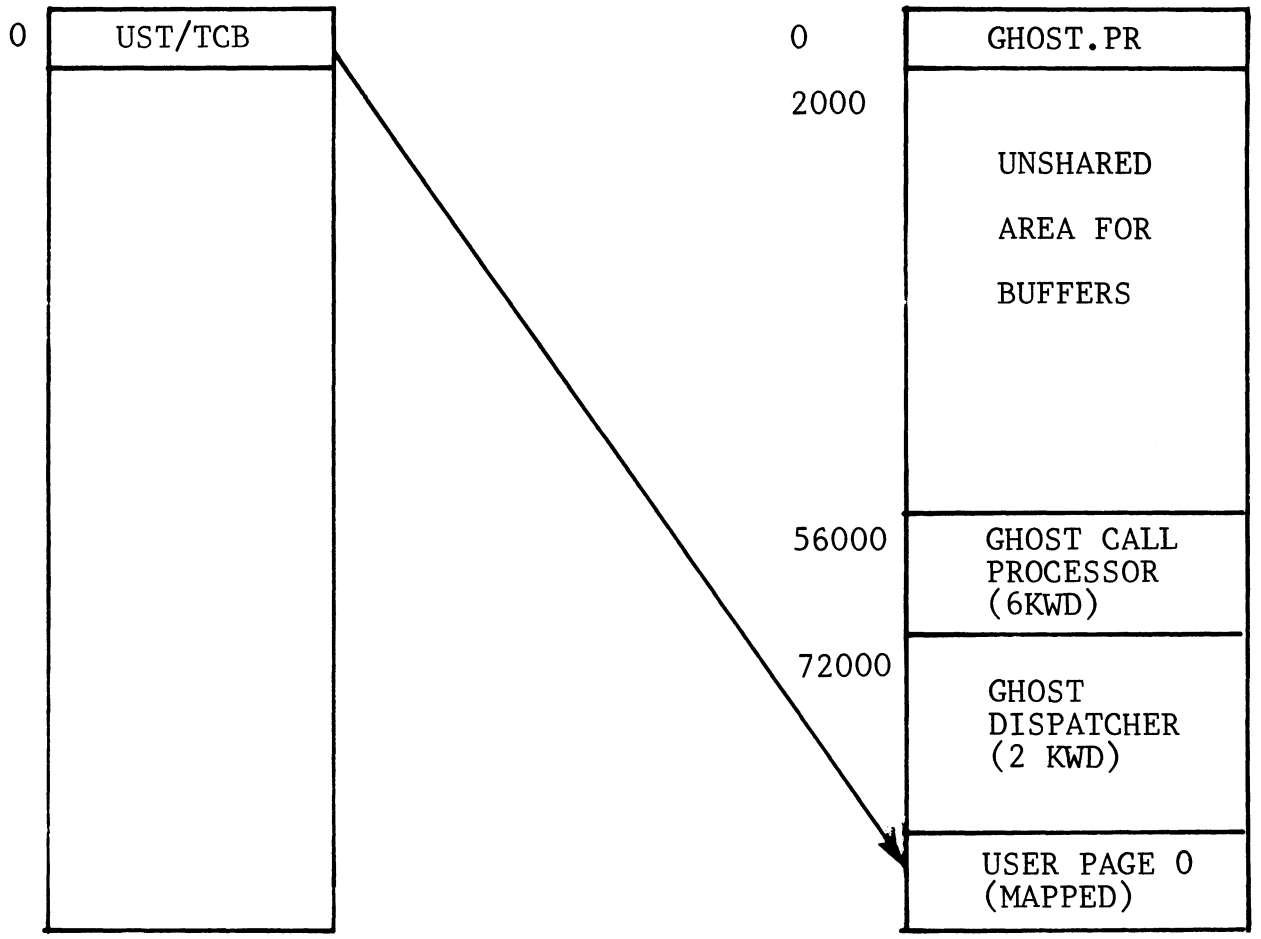
2. Describe the mechanisms the ghost uses for scheduling system calls.



GHOST

- USER PROCESS ADDRESS SPACES IN PTBL
  - USER CONTEXT
  - GHOST CONTEXT
- GHOST INITIALIZATION DEPENDS ON TYPE OF PROGRAM
- GHOST PERFORMS FOLLOWING FUNCTIONS
  1. Resolves Generic File Name References
  2. Deblocks Records To/From Blocks of Block Devices
  3. Processes Certain System Calls
- DEBUG IS A PART OF THE GHOST
- GHOST CODE IS SHARED
  - USES AT LEAST 1KWD UNSHARED PER USER PROCESS.
- GHOST CPU TIME CHARGED TO USER PROCESS.

LOGICAL/PHYSICAL ORGANIZATION



* GHOST.PR LOADED AT GHOST INITIALIZATION TIME
* GHOST.OL CONTAINS ALL GHOST PROCESSING MODULES
* - Resident - shared overlays (not like user overlays)
  - Loaded by GHOST.PR (its only function)
* GHOST PROCESSES SYSTEM CALLS FOR USER BY MAPPING USER PAGE 0 (UST/TCB's) INTO ITS PAGE 31 (76000_8)AND SETS B0 IN TCB TO INDICATE GHOST IS PROCESSING IT. GHOST HAS NO TCB's OF ITS OWN.
* GHOST IS CORE RESIDENT, AND DOES NOT EXIST IN THE USER PROCESS HIERARCHY

MODULE SYSTEM CALL SUMMARY

Section III SCHEDULING

Module Title: C : GHOST

?GOPEN	-	Open a file for block I/O
?GCLOSE	-	Closes a file opened by ?GOPEN
?MBTG	-	Move bytes to ghost from user
?MBFG	-	Move bytes from ghost to user

SECTION GOAL

Course Title: S309 SYSTEM PROGRAMMER

Section IV MEMORY MANAGEMENT

To understand the options available for effi c ient memory utilization.

MODULE OBJECTIVES

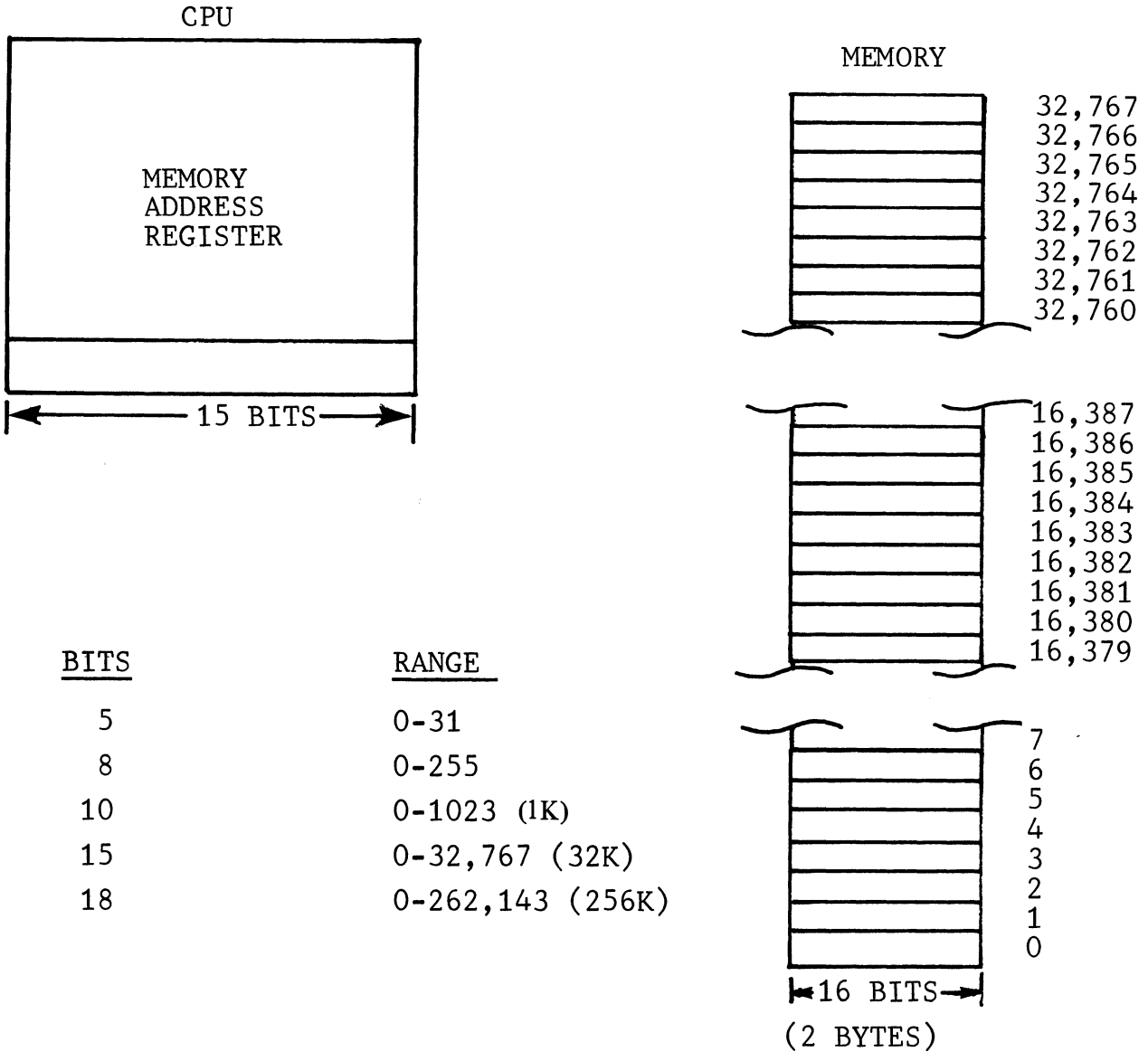
Section IV MEMORY MANAGEMENT

Module Title: A: MAP HARDWARE

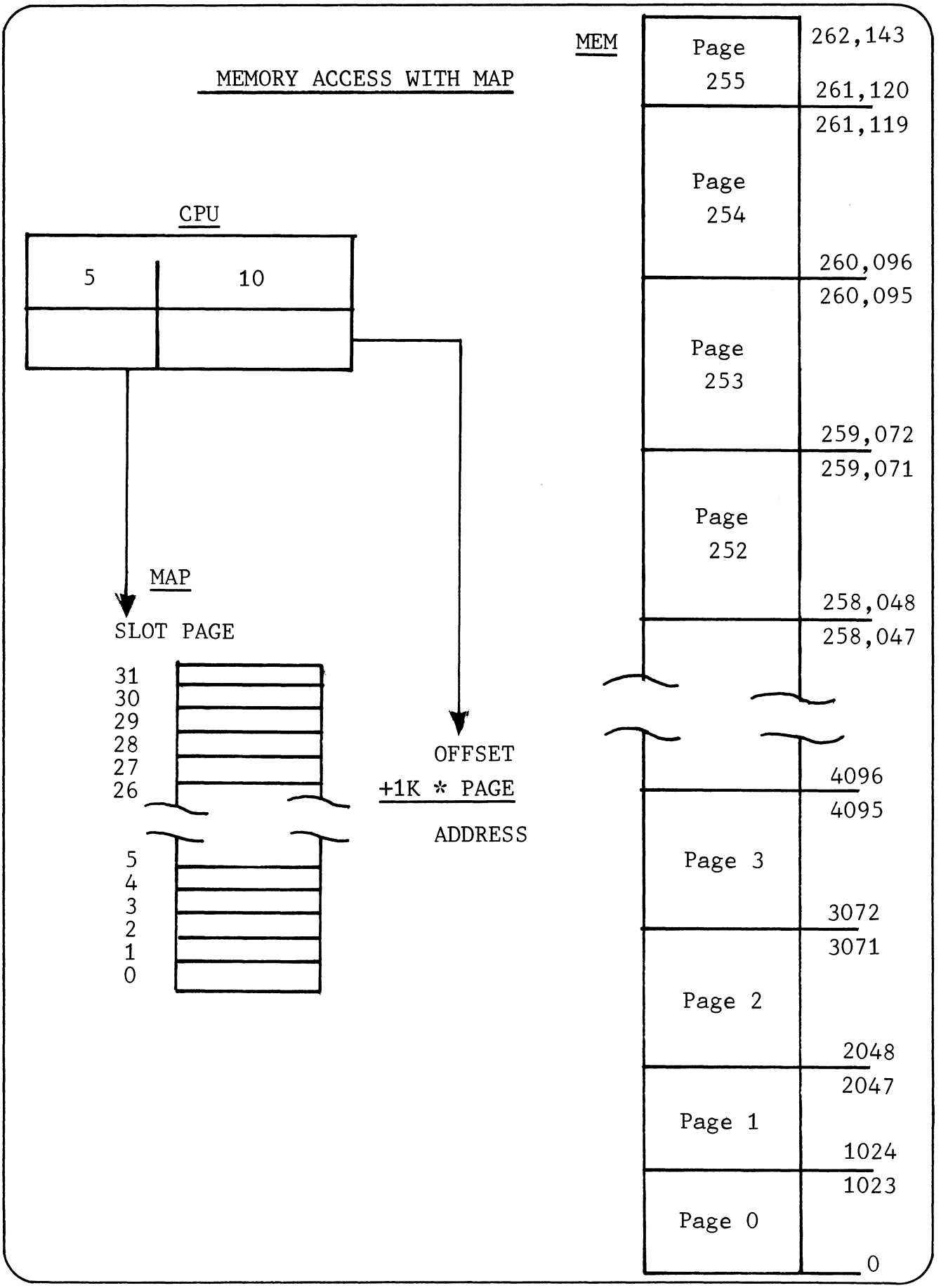
Upon successful completion of this module the student will be able to:

1. Explain address translation through the MAP
2. Describe AOS's use of the MAP to support user processes.

MEMORY ACCESS WITHOUT MAP



BITS	RANGE
5	0-31
8	0-255
10	0-1023 (1K)
15	0-32,767 (32K)
18	0-262,143 (256K)



AOS AND THE MAP

AOS CHOOSES PROCESS TO RUN

AOS SETS UP MAP SLOTS

AOS ENABLES MAP & JUMPS TO USER

USER RUNS

MAP DISABLED BY:

HARDWARE INTERRUPT

SYSTEM CALL

USER TRAP

AOS CONTAINS UNMAPPED CODE

TO HANDLE 3 CASES:

ADDRESS 1= INTERRUPT

ADDRESS 2= CALL

ADDRESS 3= TRAP

MAP PROTECTION

ACCESS PROTECT
(EMPTY SLOT)

WRITE PROTECT

MACHINE LEVEL I/O
(MAP ITSELF PROGRAMMED
THROUGH I/O COMMANDS)

ENDLESS DEFER LOOP
(16 INDIRECT ADDRESSES)

This page intentionally left blank.

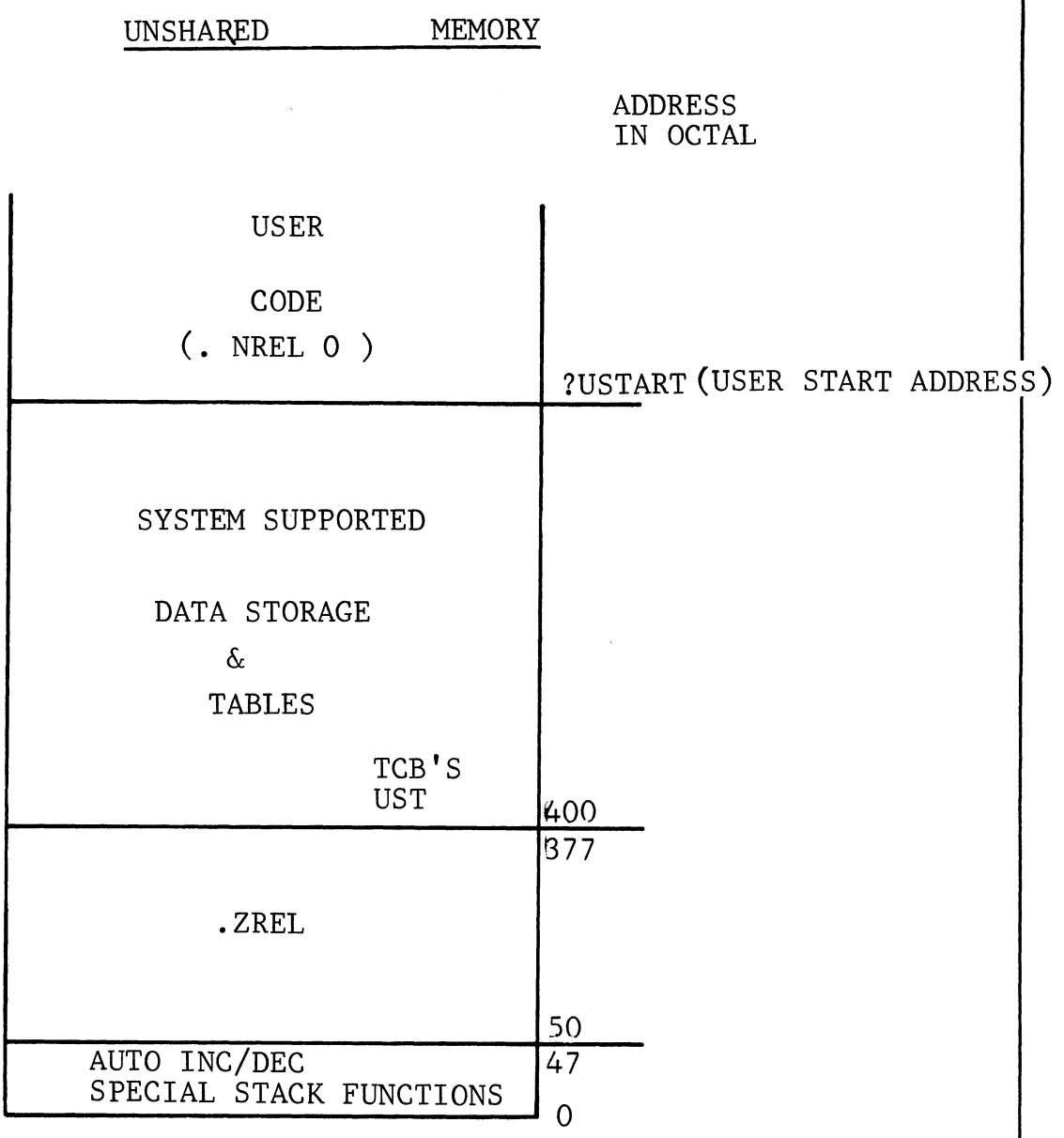
MODULE OBJECTIVES

Section IV MEMORY MANAGEMENT

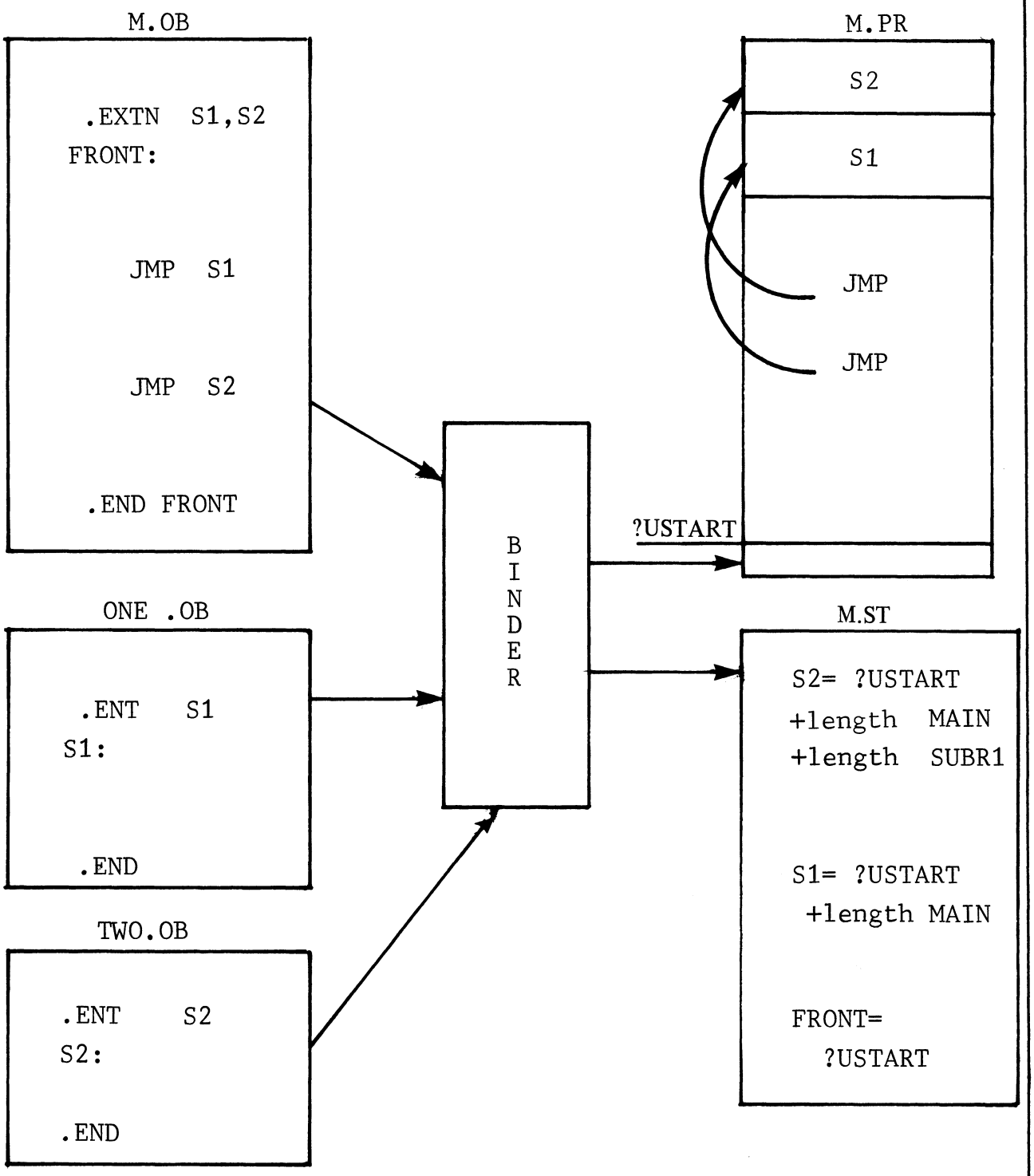
Module Title: B: UNSHARED MEMORY

Upon successful completion of this module the student will be able to:

1. Identify the portions of a program that need unshared memory.
2. Interpret Bind listing reports of unshared memory assignment.
3. List the major sections of the User Statue Table
4. Design a program using unshared overlays.



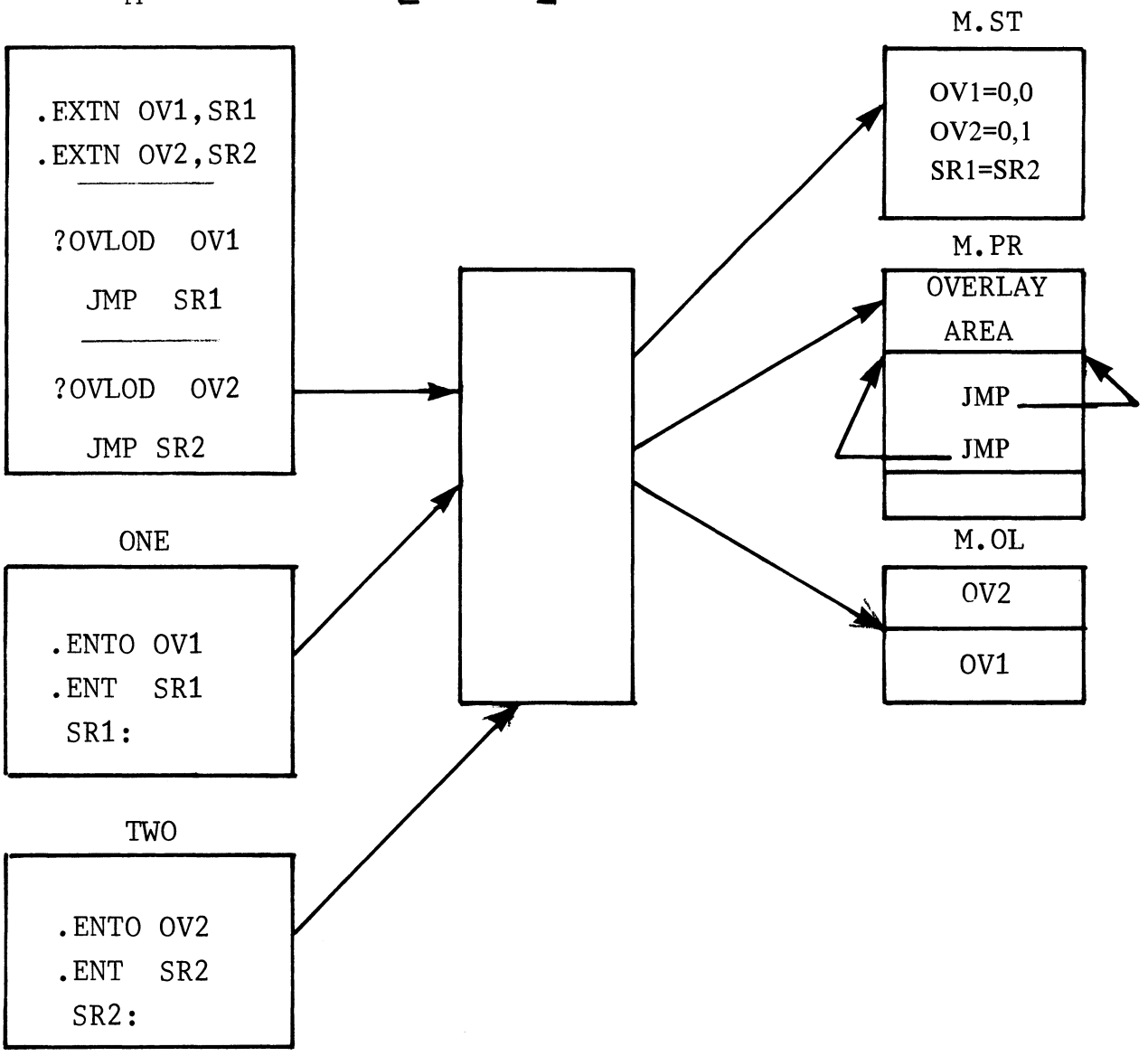
BINDING UNSHARED OBJECT CODE



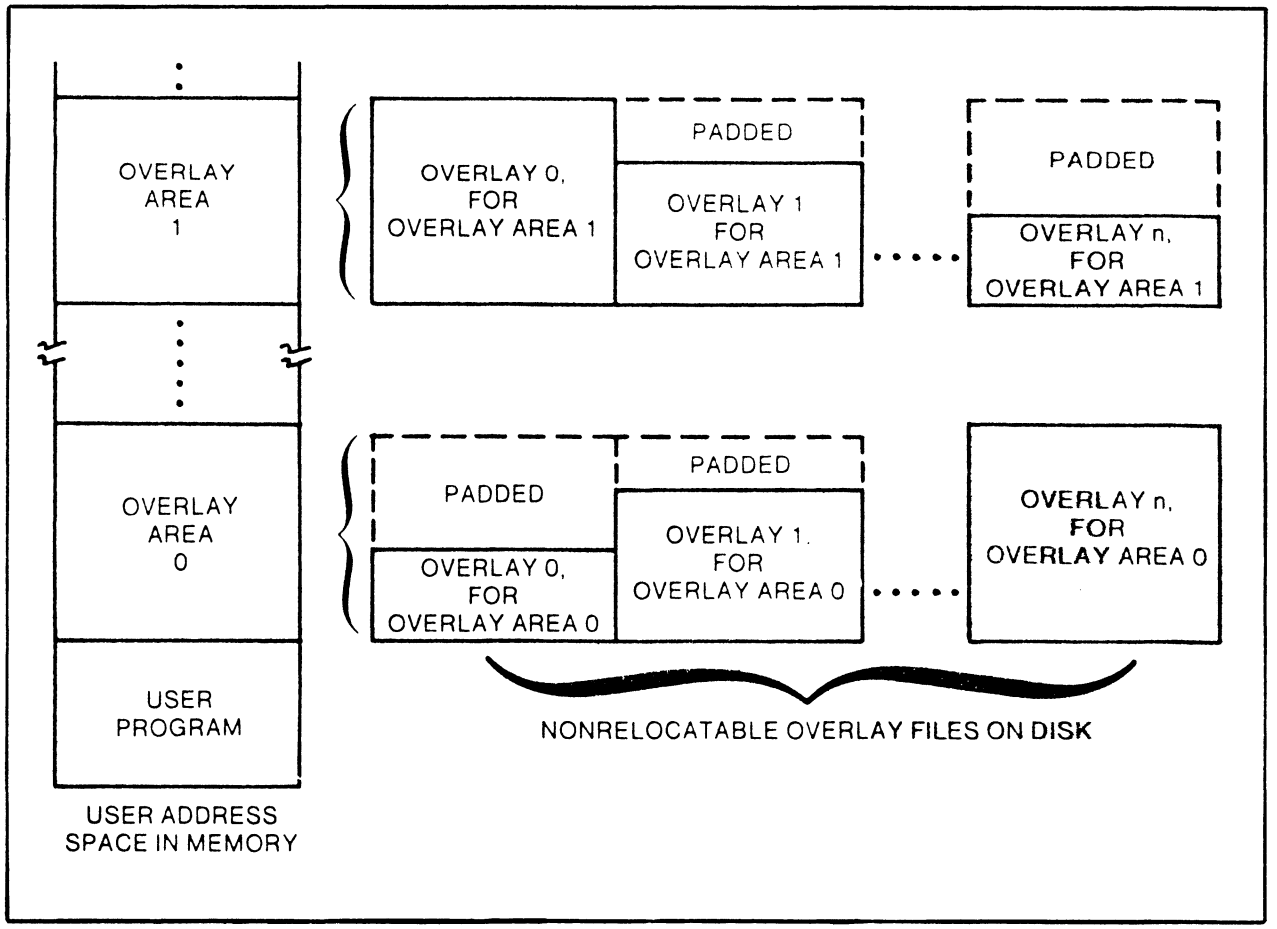
BINDING UNSHARED OVERLAYS

```
X BIND M OVS/C

OVS = [ONE, TWO]
```



UNSHARED OVERLAY STRUCTURES



OVERLAY AREAS

SIZE OF
"BASIC OVERLAY AREA"
IS ROUNDED UP FROM LARGEST
OVERLAY THAT MUST FIT
IN THAT AREA

YOU CAN BUILD A
"TOTAL OVERLAY AREA"
THAT IS A SERIES OF
BASIC AREAS

EXAMPLE: YOU HAVE 5 ROUTINES,
But SPACE FOR ONLY 4
[ONE, TWO, THREE, FOUR, FIVE] /4
ALLOWS ROOM FOR 4 OF THE 5

ONE PROGRAM CAN HAVE 63 "TOTAL" AREAS

OVERLAY CODE SHOULD BE POSITION INDEPENDENT
IF USING MULTIPLE BASIC AREAS-AOS COULD
LOAD ANY OVERLAY IN ANY BASIC AREA

LOADING OVERLAYS

?OVLOD

CONDITIONAL - IF OVERLAY
REQUESTED IS ALREADY
LOADED, USE IT

UNCONDITIONAL - ALWAYS TRANSFER
A FRESH COPY FROM DISK
(GOOD FOR INITIAL VALUES)

LOAD ONLY - START LOCATION = -1
- JUST DELIVERS OVERLAY & RETURNS
- YOU MUST JUMP TO OVERLAY
AS TO ANY CODE

LOAD AND GO - START LOCATION CONTROLS
- 0B0 FOR ADDRESS
- 1B0 FOR OFFSET

LOCKING OVERLAYS

EVERY OVERLAY LOAD MARKS
AREA AS BEING IN USE BY
OVERLAY USE COUNT (OVC)
KEPT IN AOS TABLES OF LOW CORE

REQUESTS TO LOAD DIFFERENT
OVERLAYS INTO OCCUPIED AREA
WILL BE SUSPENDED UNTIL
AREA IS FREE
(OVC = 0)

CONDITIONAL LOADS OF
THE CURRENT OVERLAY
SIMPLY INCREMENT OVC
BY ONE - NOW AN ADDITIONAL
RELEASE MUST BE DONE
TO FREE UP AREA

RELEASING OVERLAYS

YOU ARE RESPONSIBLE FOR
ISSUING ONE RELEASE FOR
EVERY OVERLAY LOAD

BEWARE – RELEASING AN AREA
MAY CAUSE AOS TO
OVERWRITE AREA IN
RESPONSE TO A SUSPEND
LOAD

?OVREL – RELEASE
– FROM OUTSIDE AREA

?OVEX – EXIT TO SAFE CODE
– FROM WITHIN AREA
– YOU PROVIDE RETURN ADDRESS

?OVKIL – KILL CURRENT TASK
– FROM WITHIN AREA
– DISABLES RETURN TO AREA

MODULE SYSTEM CALL SUMMARY

Section IV MEMORY MANAGEMENT

Module Title: B: UNSHARED MEMORY

?OVEX	-	Release an overlay area and return
?OVLOD	-	Load and go to an overlay
?OVREL	-	Release an overlay area
?OVKIL	-	Release an overlay area and kill the task

MODULE OBJECTIVES

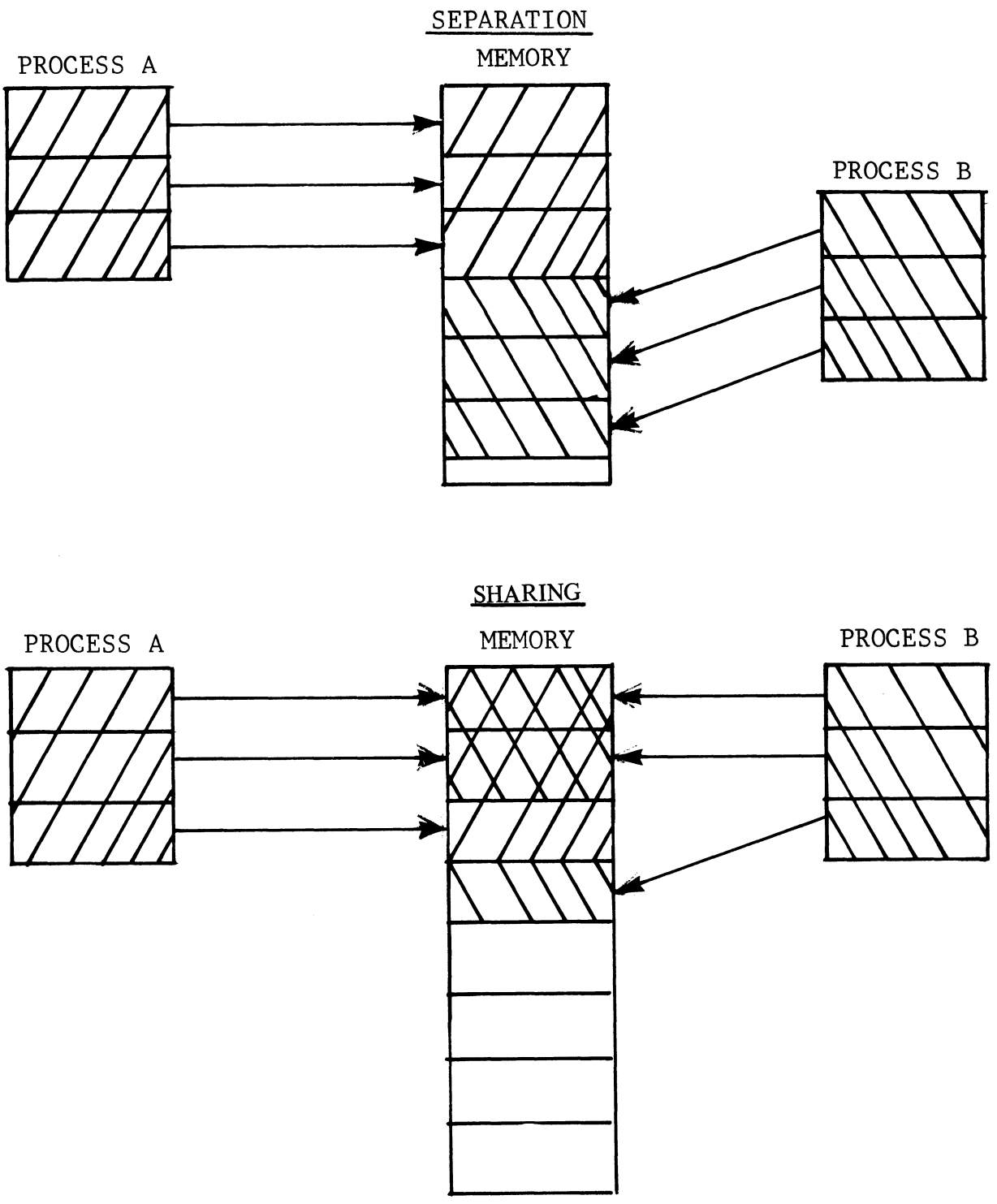
Section IV MEMORY MANAGEMENT

Module Title: C: SHARED MEMORY

Upon successful completion of this module the student will be able to:

1. Explain the need for and structures of re-entrant and position independent code.
2. Describe program applications suitable for shared code.
3. Describe the tradeoffs of using shared data pages.
4. Interpret BIND listing reports of shared structures.
5. Describe Least Recently Used replacement of shared pages.

MAP APPLICATIONS



AOS AND THE MAP

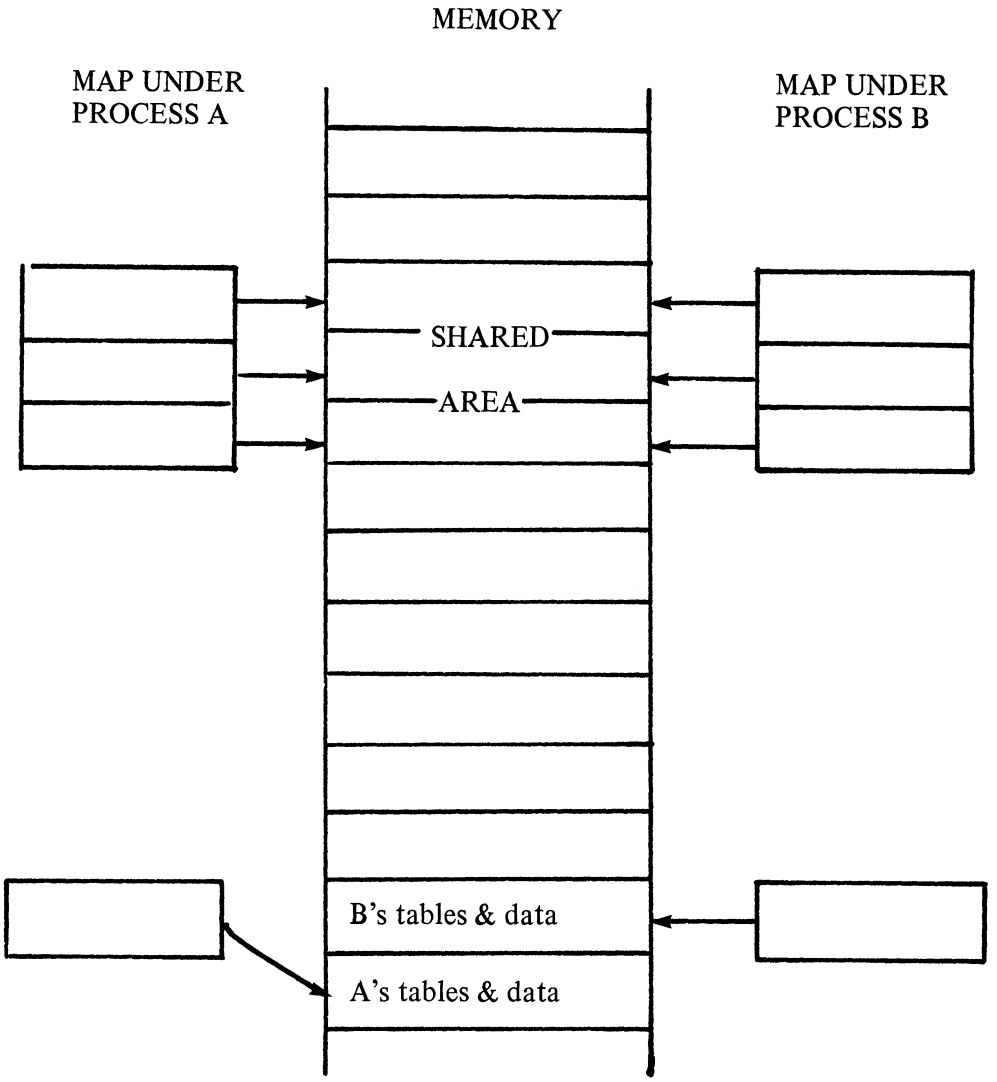
MAP IS A 64K BYTE
WINDOW THROUGH WHICH
THE CPU VIEWS MEMORY

MAP IS SET FOR ONLY
A SINGLE USER AT A TIME

SHARING IS A SOFTWARE GAME
BY WHICH AOS REUSES THE
SAME PHYSICAL MEMORY IN
MORE THAN ONE PROCESS' VIEW

TYPE OF SHARING	MATCHING PATHNAME
IN-LINE	PROGRAM
DATA	DATA FILE
OVERLAYS	PROGRAM
ROUTINE	SHARED LIBRARY

IN-LINE SHARED PAGES



X EXAMP
X EXAMP

IN-LINE SHARING

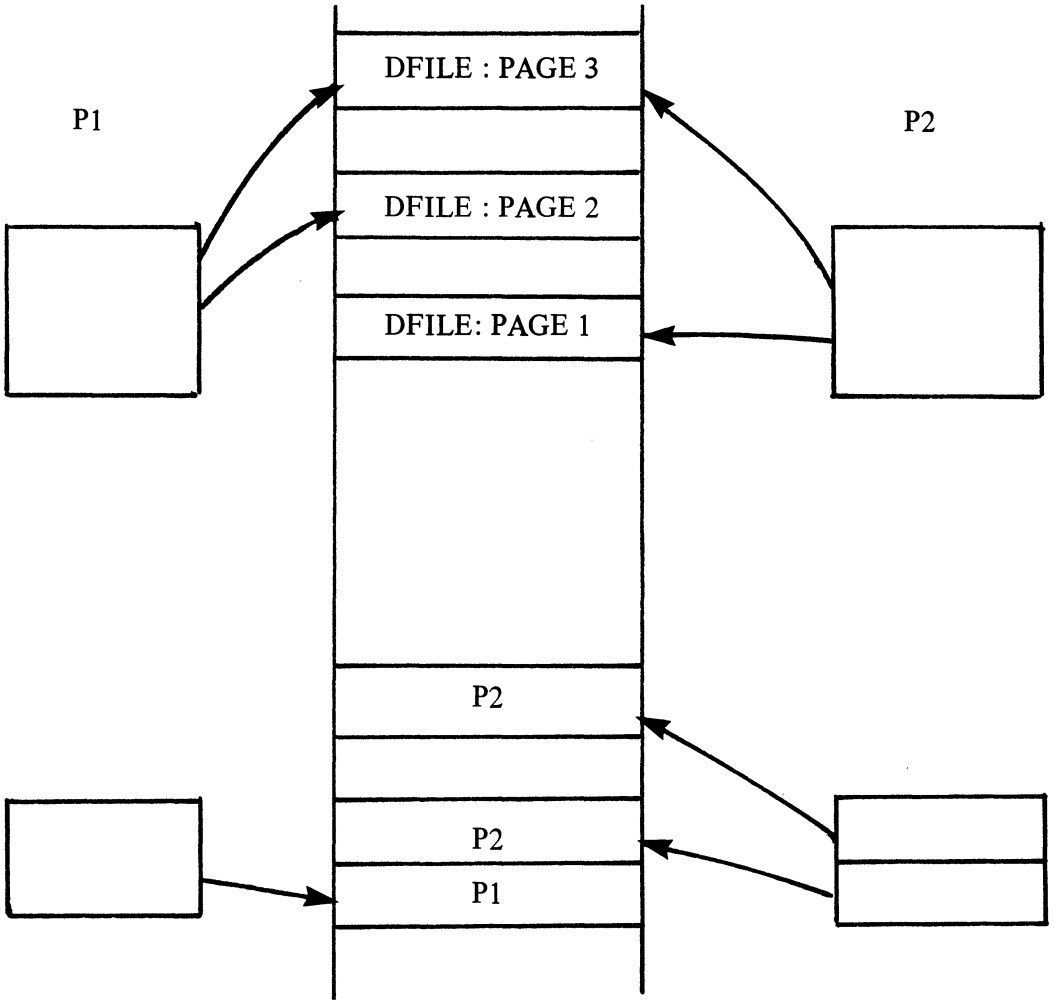
SIMPLE TO USE
- .NREL 1
- /H LOCAL BIND SWITCH
- AUTOMATIC IN HIGH LEVEL LANGUAGES

MEMORY SAVED ONLY IF SAME PROGRAM RUNS UNDER DIFFERENT PROCESSES

REENTRANT (PURE) CODE RECOMMENDED EVEN IF ONE COPY OF A PROGRAM HAS BEEN RUNNING FOR AN HOUR, A NEW COPY SHOULD BE ABLE TO USE THE SAME CODE

DATA CAN BE SHARED THIS WAY - IF ONE PROCESS CHANGES VALUES, ALL PROCESSES AFFECTED (EXAMPLE: USER RESOURCE LOCKING)

SHARED DATA



X P1
X P2

SHARED DATA

NO SPECIAL SET UP
FOR DATA FILE

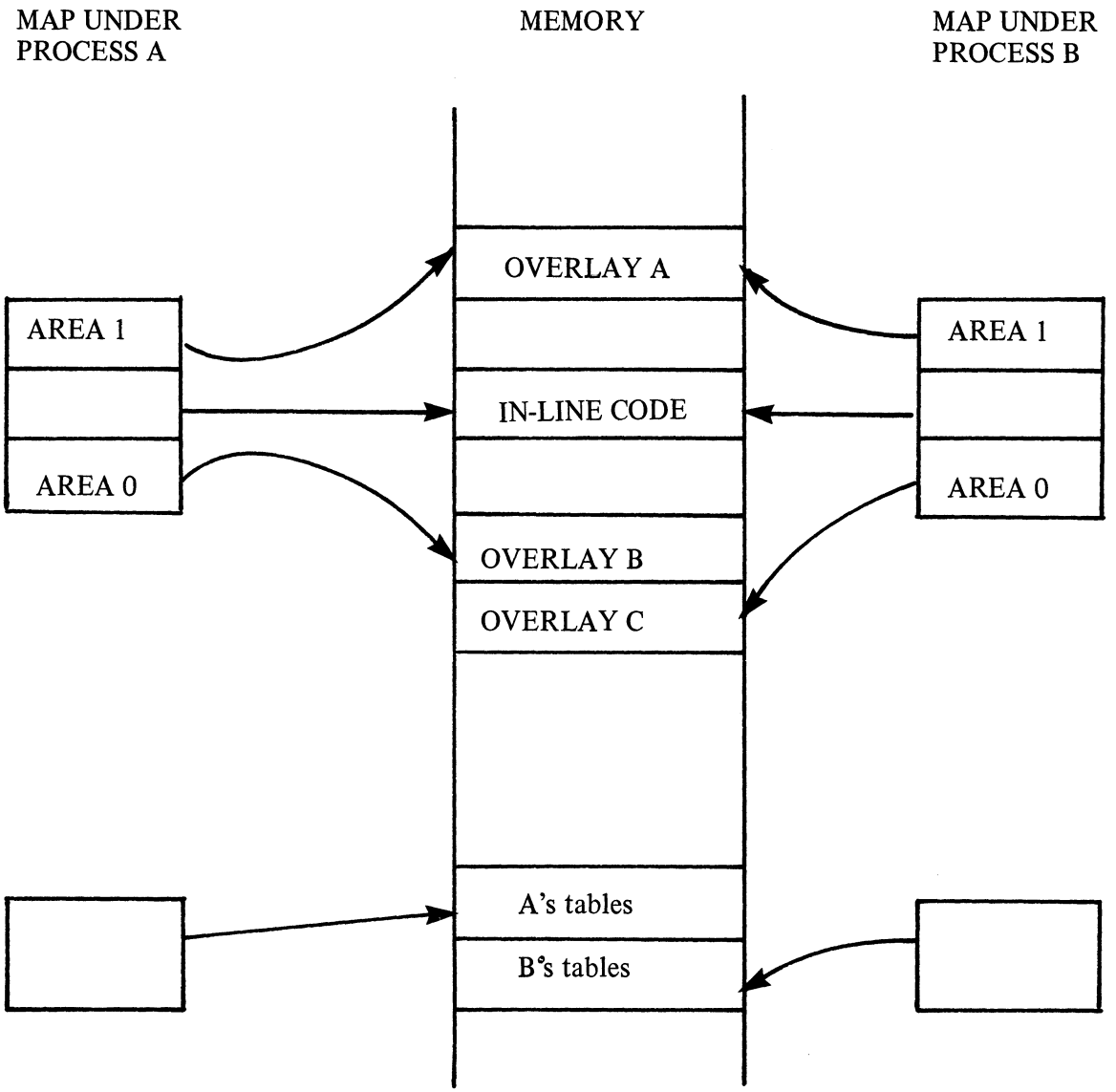
?SOPEN – SHARED OPEN
WARNS AOS TO GET READY FOR
SHARING

?SPAGE – IF PAGE YOU WANT
IS ALL READY IN CORE, YOU ONLY
WAIT FOR A REMAP

PAGES CAN BE WRITE PROTECTED
THROUGH BOTH ?SOPEN AND ?SPAGE

?FLUSH – IF YOU WRITE TO A
SHARED PAGE, YOU CAN FORCE
AOS TO COPY IT BACK TO THE
ORIGINAL FILE

SHARED OVERLAYS



X SOVLAY
X SOVLAY

SHARED OVERLAYS

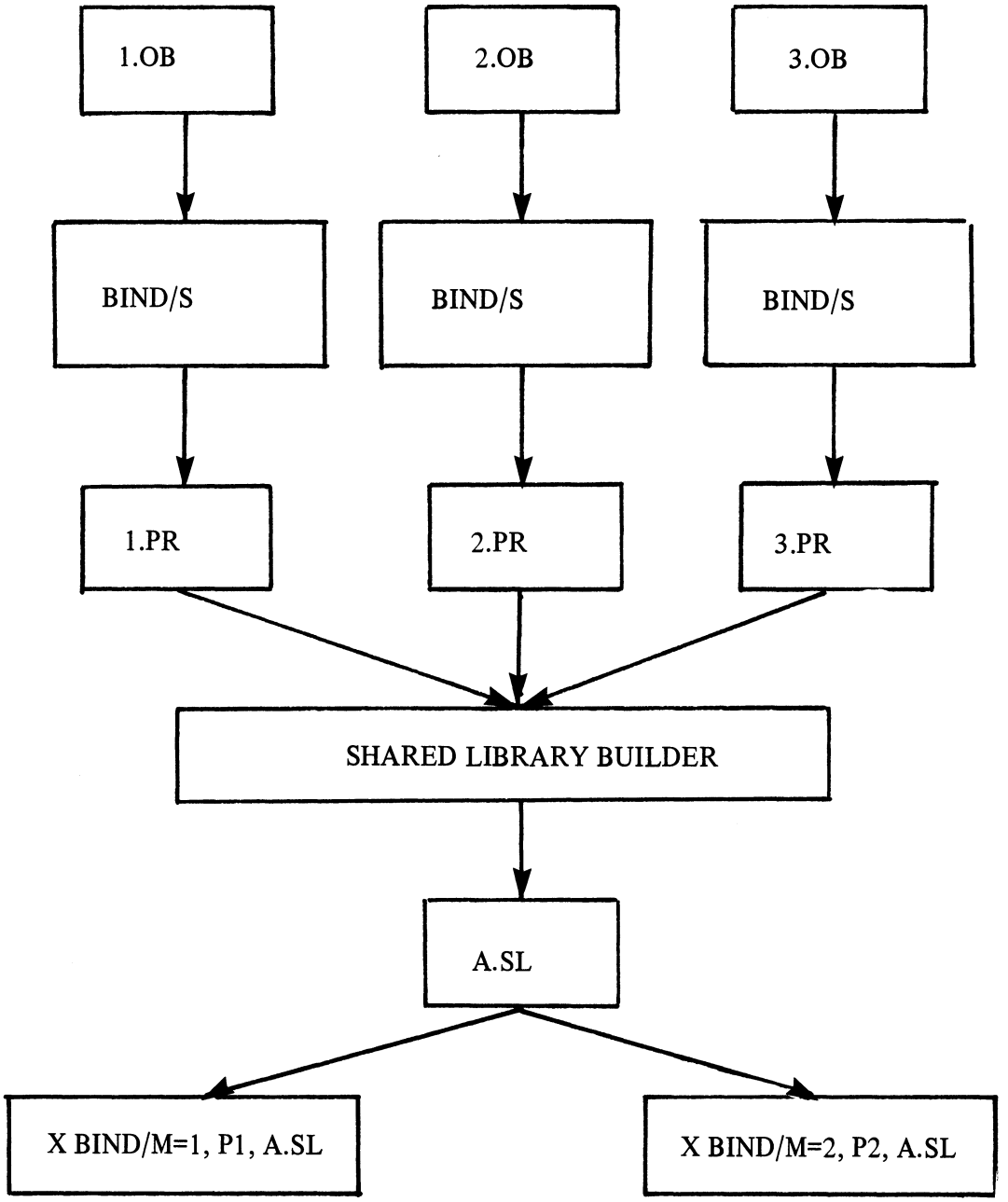
SAME AS UNSHARED EXCEPT –

* AREAS MUST BE PAGE ALLIGNED
* OVERLAYS BUILT IN MULTIPLES OF PAGES

MEMORY SAVED ONLY WHEN
MORE THAN ONE COPY OF THE PROGRAM
WANT THE SAME OVERLAY

OVC LOCKS FROM ONE PROCESS
DON'T AFFECT ANY OTHER PROCESS

SHARED ROUTINES



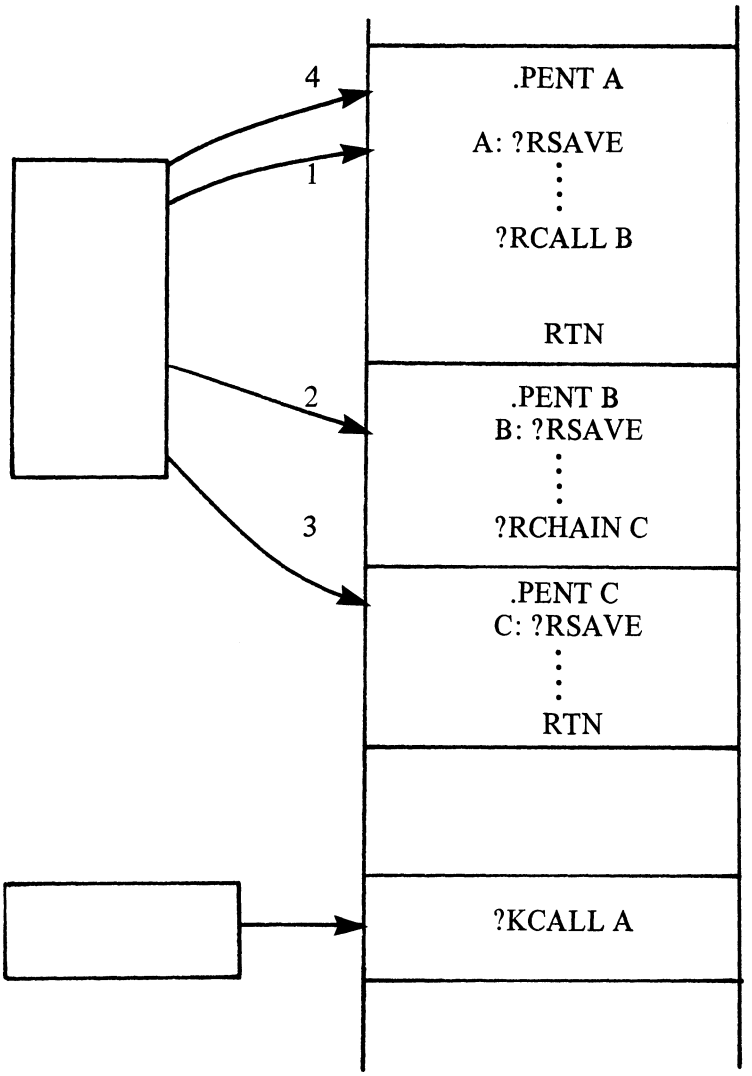
SHARED ROUTINES

SHARED ROUTINES ARE MINI-PROGRAMS
THAT BECOME A PART OF A REGULAR PROGRAM
WHEN CALLED.

CODE SHOULD BE RE-ENTRANT FOR
SHARING BETWEEN TWO PROCESSES –
CAN BE DIFFERENT PROGRAMS

CODE SHOULD BE POSITION-INDEPENDENT
SO AOS CAN LOAD IT IN ANY SHARED
PAGE THAT THE BINDER RESERVED IN THE
REGULAR PROGRAM SPACE

MANAGING SHARED ROUTINES



LOADING SHARED ROUTINES

Entry point into each routine must be declared in ".PENT" statement

Routine must start with "?RSAVE" (MASM MACRO). AOS uses stack to remember where to return after routine is done.

Most routines end with "RTN" instruction to release routine and return to routine's caller though stack information

* ?RCHAIN NEXT
  - releases routine
  - loads, locks, goes to next

* ?RCALL next
  - releases routine
  - uses stack to record current position
  - loads, locks and goes to next

* ?KCALL next
  - uses stack to record current position
  - loads, locks and goes to next

MODULE SYSTEM CALL SUMMARY

Section IV MEMORY MANAGEMENT

Module Title: C: SHARED MEMORY

?SOPEN	-	OPEN A SHARED PAGE DATA FILE
?SCLOSE	-	CLOSE DOWN A ?SOPEN
?SPAGE	-	READ A SHARED DATA PAGE
?FLUSH	-	COPY A CHANGED SHARED DATA PAGE BACK TO DISK
?RCALL	-	RELEASE ROUTINE, GO TO NEXT
?KCALL	-	KEEP ROUTINE, GO TO NEXT
?RCHAIN	-	RELEASE & FORGET ROUTINE, GO TO NEXT

MODULE OBJECTIVES

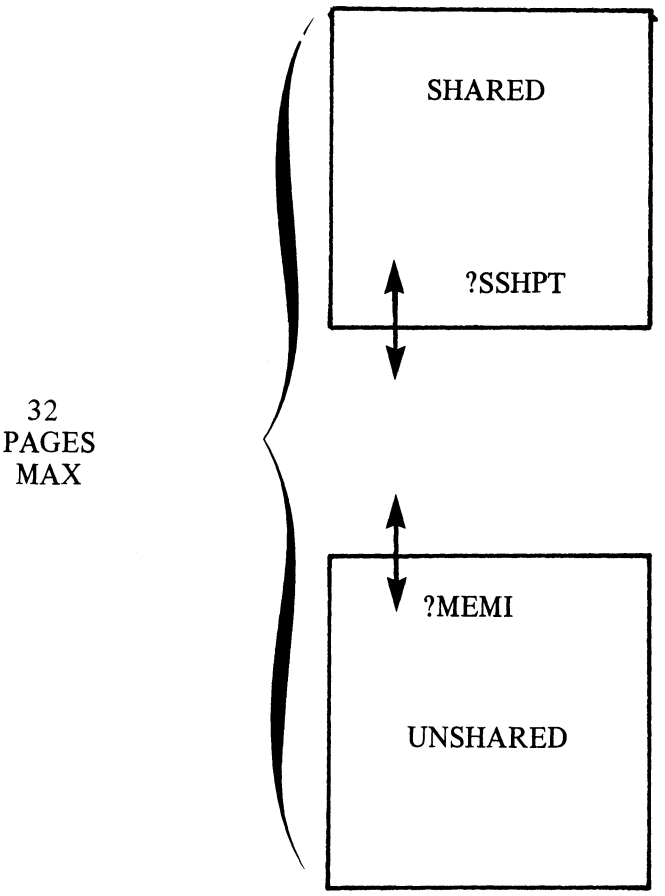
Section IV MEMORY MANAGEMENT

Module Title: D: PROGRAM STRUCTURE

Upon successful completion of this module the student will be able to:

1. Explain the tradeoffs of shared and unshared memory.
2. Implement Resource Calls to support flexible program structure.
3. Describe the mechanisms for dynamic memory allocation.

PROGRAM STRUCTURE



PROGRAM STRUCTURE

BIND BUILDS UNSHARED CODE FROM
BOTTOM UP, SHARED ALSO IN
ASSENDING ORDER BUT POSITIONED
IN HIGH CORE.

BOTH CAN GROW INTO ANY
UNCLAIMED MAP SLOTS IN THE MIDDLE

BOOKKEEPING FOR PAGES KEPT IN UST

READ PRESENT ALLOCATIONS–
- ?MEM FOR UNSHARED
- ?GSHPT FOR SHARED

CHANGE MEMORY ALLOCATIONS–
- ?MEMI FOR UNSHARED
- ?SSHPT FOR SHARED

RESTRUCTURING PROGRAMS

?RCALL, ?RCHAIN, ?KCALL
CAN BE USED TO LOAD, LOCK and GO TO
ANY ROUTINE:

* SHARED ROUTINE
* SHARED OVERLAY
* UNSHARED OVERLAY
* SHARED ROOT
* UNSHARED ROOT

(LOAD and LOCK NOT APPLICABLE TO ROOT)

IF YOU USE RESOURCE CALLS
WHENEVER YOU JUMP FROM ONE
MODULE TO ANOTHER, YOU CAN
RESTRUCTURE YOUR PROGRAM THROUGH
BIND WITHOUT CHANGING ANY CODE

BIND SUBSTITUTES THE PROPER
GO TO/RETURN FROM CODE

BIND OPTIONS

[ANY .OB]

DEFINE ANY AS AN OVERLAY

[ANY .OB] /AM=#

BASIC AREAS IN TOTAL AREA

ANY .OB/H

CHANGE ANY UNSHARED CODE INTO SHARED

ANY .OB/S

CHANGE ANY SHARED CODE TO UNSHARED

BIND/M = #

RESERVE # PAGES FOR SHARED ROUTINES

MODULE SYSTEM CALL SUMMARY

Section IV MEMORY MANAGEMENT

Module Title: D: PROGRAM STRUCTURE

?RCALL	-	Make a general procedure call and release the calling resource
?KCALL	-	Make a general procedure call, and do not release the calling resource
?RCHAIN	-	Release the current resource, acquire a new resource; the chained procedure will return control to the previous caller.
?MEM	-	Get unshared page description
?MEMI	-	Change number of unshared pages
?GSHPT	-	Get shared page description
?SSHPT	-	Change number of shared pages

MODULE OBJECTIVES

Section IV MEMORY MANAGEMENT

Module Title: E: PROCESS TRADEOFFS

Upon successful completion of this module the student will be able to:

1. Determine the outcome of contention for memory.
2. Describe the Bias Factor

This page intentionally left blank

ALLOCATING MEMORY

WHENEVER A USER PROCESS
NEEDS MEMORY AOS GETS IT FROM:

1. UNUSED PAGES

2. OVERWRITING OLD AOS OVERLAYS

3. OVERWRITING UNUSED SHARED
   PAGES BY LEAST RECENTLY USED

4. SWAPPING OUT BLOCKED PROCESSES
   BY PRIORITY

5. SWAPPING OUT LOWER PRIORITY
   ELIGIBLE PROCESSES BY PRIORITY

LOCKING SHARED PAGES

IN ADDITION TO LOCKS
WITHIN A SINGLE PROCESS
(OVERLAY AREA IN USE, SHARED ROUTINE
PAGE OCCUPIED)
AOS MAINTAINS A USE COUNT FOR
EACH SHARED PAGE IN CORE

THIS SYSTEM WIDE USE COUNT IS
THE NUMBER OF PROCESSES THAT
A PAGE IS MAPPED TO

USE COUNT INCREMENTED BY:
- *RUNNING A PROGRAM WITH IN-LINE SHARING
- *LOADING A SHARED ROUTINE OR OVERLAY
- *?SPAGE FOR A SHARED DATA PAGE

USE COUNT DECREMENTED BY:
- *SWAPPING OR TERMINATING PROCESS
- *RTN FROM A RESOURCE CALL
- *?RCHAIN OR ?RCALL
- *EXPLICIT OVERLAY RELEASE CALLS
- *?RPAGE or next ?SPAGE

LEAST RECENTLY USED (LRU) CHAIN

WHILE A SHARED PAGE HAS A NON-ZERO USE COUNT, IT WILL NOT BE OVERWRITTEN

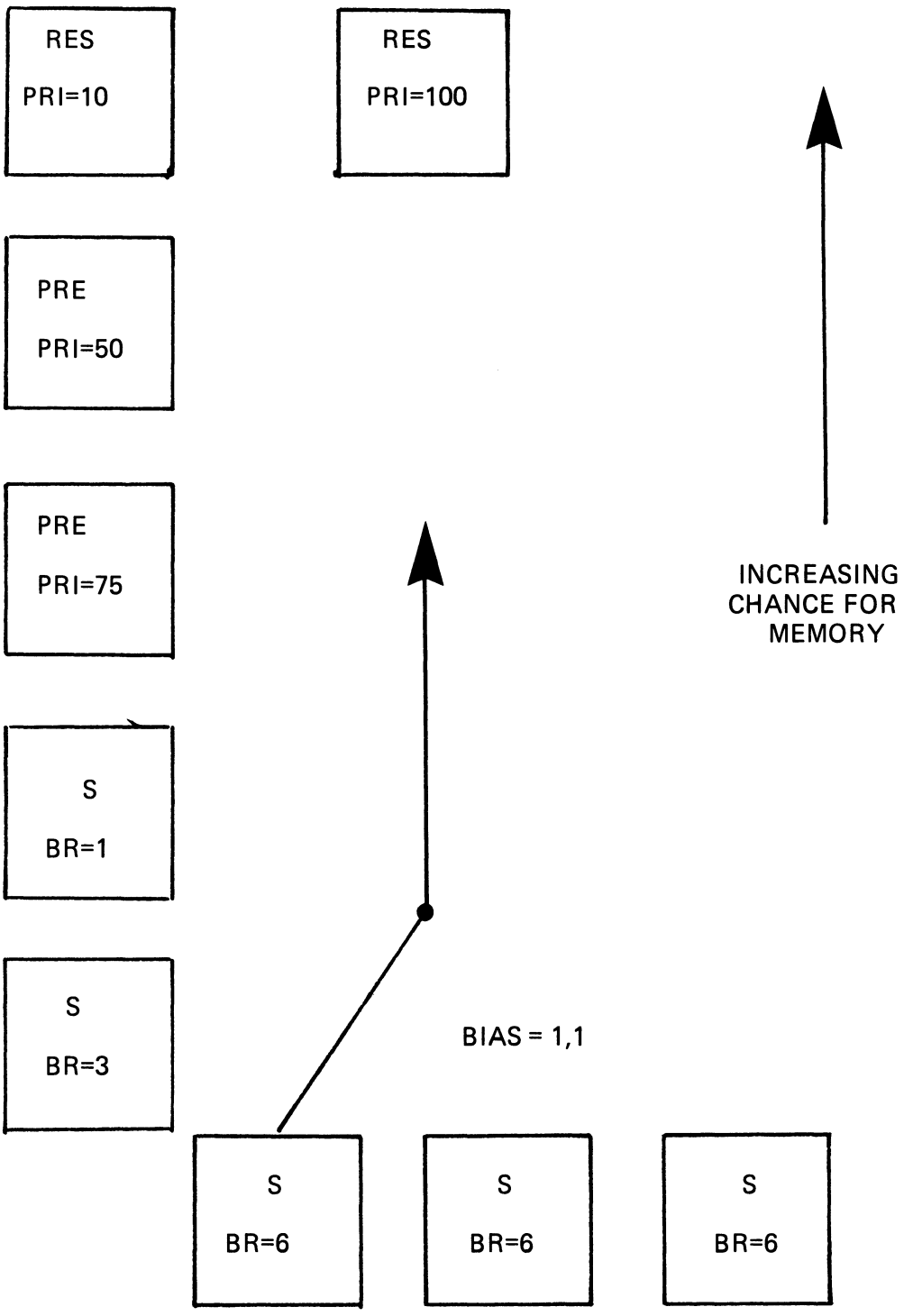
WHEN A PROCESS GETS SWAPPED, SHARED PAGES ARE NEVER PART OF THE SWAP FILE – THE ONLY EFFECT ON THE SHARED PAGE IS DECREMENT OF USE COUNT

WHEN USE COUNT HITS ZERO PAGE IS ADDED TO THE END OF A CHAIN OF UNUSED SHARED PAGES

IF AOS NEEDS MEMORY, SHARED PAGES FROM BEGINNING OF UNUSED CHAIN ARE OVERWRITTEN

SHARED DATA PAGES ARE RECOPIED TO THEIR ORIGINAL FILES IF YOU ALLOW THEM TO BE WRITTEN TO

MEMORY CONTENTION



BIAS FACTOR

AOS NORMALLY FAVORS INTERACTIVE OVER COMPUTE BOUND SWAPPABLE PROCESSES FOR BOTH MEMORY AND TIME

OPERATOR CAN OVERCOME THIS DEFAULT BIAS FOR MEMORY BY SETTING THE NUMBER OF COMPUTE BOUND SWAPPABLES TO KEEP IN MEMORY

SINCE ONLY PROCESSES IN CORE CAN BE GIVEN CPU TIME, BIAS FACTOR ALSO INCREASES THE CHANCE OF GETTING TIME

AOS PICKS JUST WHICH COMPUTE BOUND SWAPPABLE TO HOLD IN CORE, BUT OPERATOR CAN SET BOTH A MINIMUM AND MAXIMUM LIMIT

MODULE SYSTEM CALL SUMMARY

SECTION IV MEMORY MANAGEMENT

MODULE TITLE: E: PROCESS TRADEOFFS

?SBIAS	–	SET BIAS FACTOR
?GBIAS	–	GET CURRENT BIAS FACTOR
?RPAGE	–	RELEASE A SHARED DATA PAGE

This page intentionally left blank.

SECTION GOAL

Course Title: S309 SYSTEM PROGRAMMER

Section V INTERFACING

To interact with the operating system on a primitive level with regard to interrupt handling and to manipulate system data that is normally processed by the utilities.

MODULE OBJECTIVES

Section V INTERFACE

Module Title: A: DRIVERS

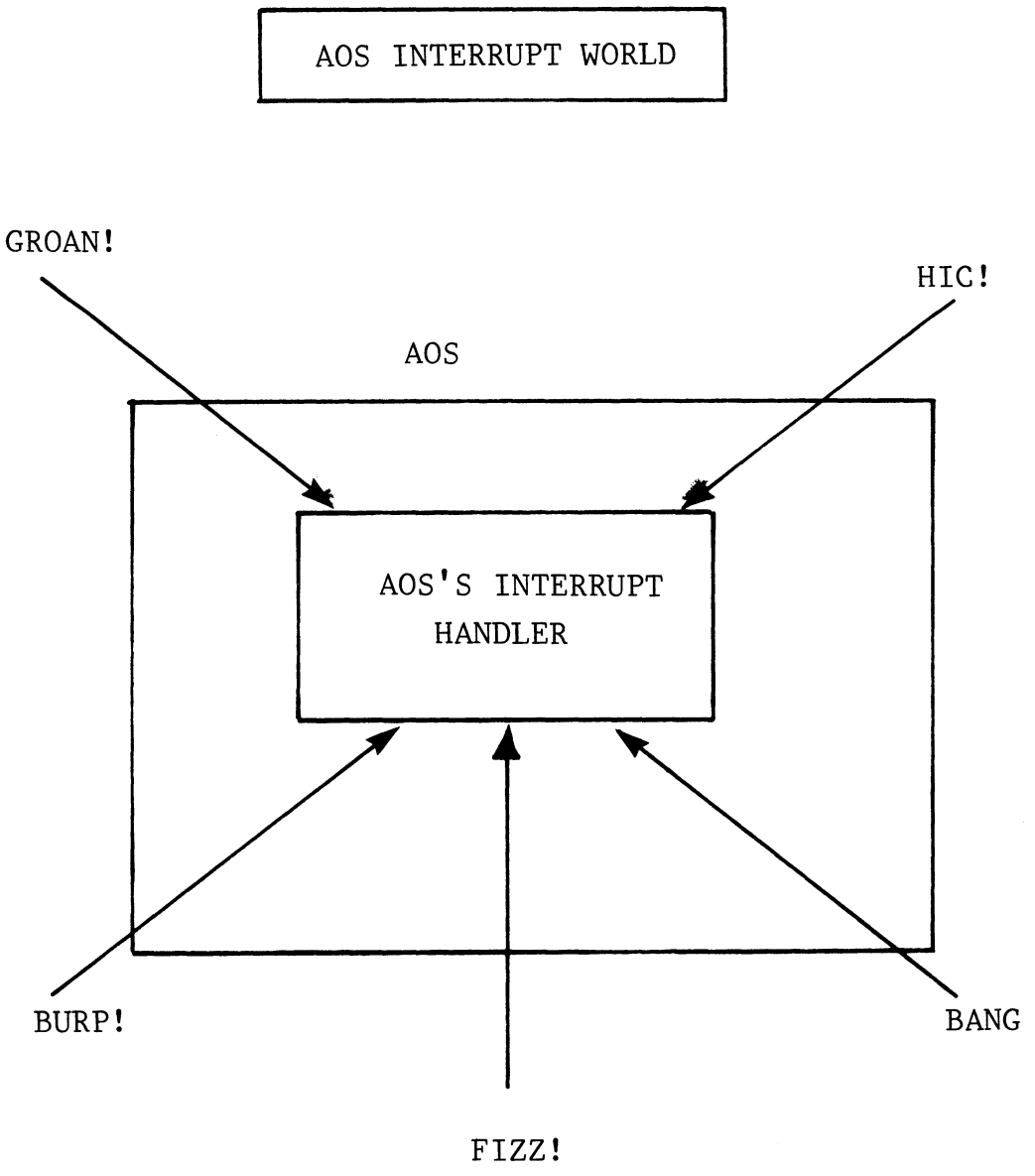
Upon successful completion of this module the student will be able to:

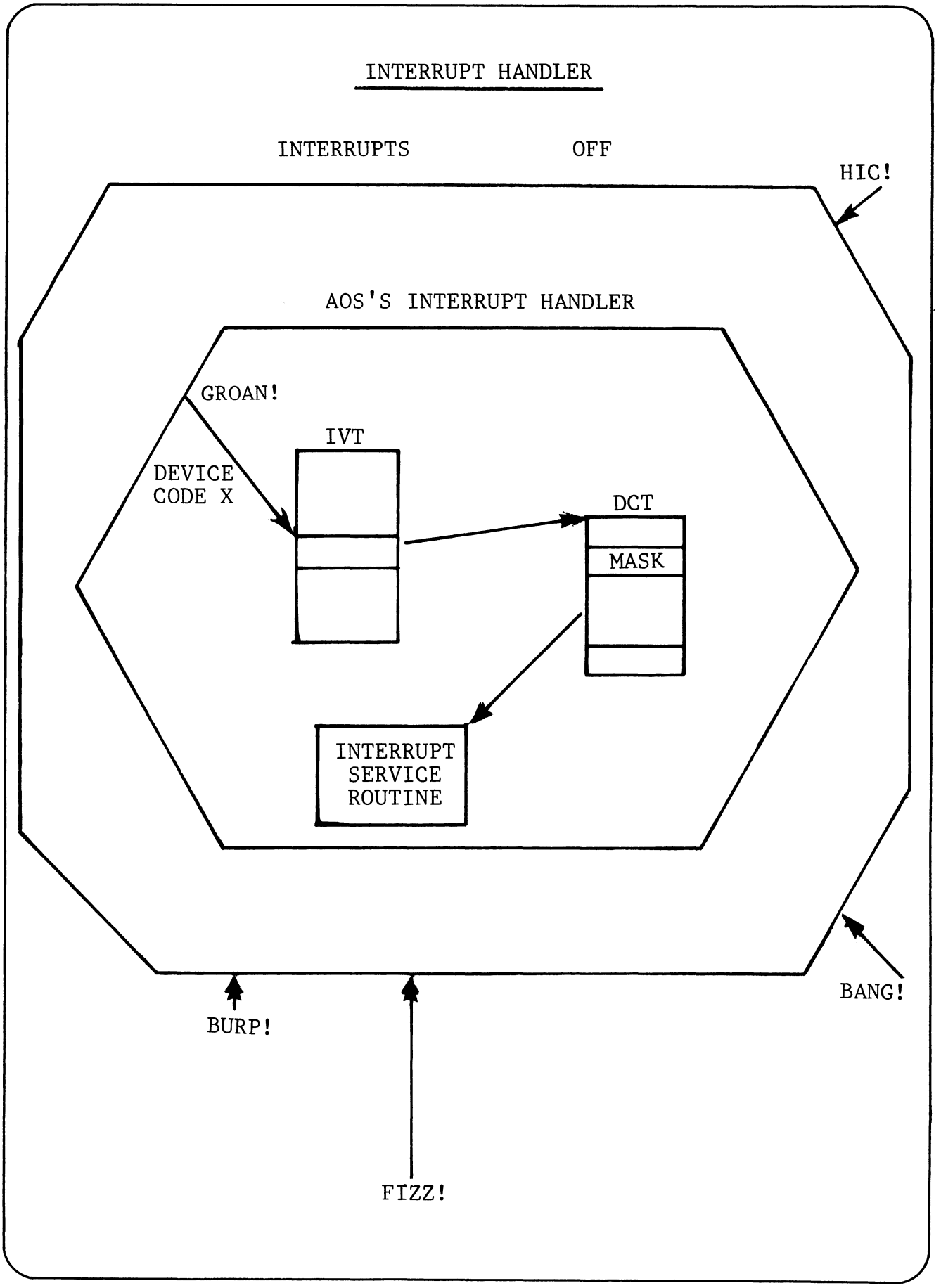
1. List the steps necessary to implement a device driven in a user process.
2. Describe the criteria for adding a device thru modifying sysgen files.
3. Describe the major phases of AOS interrupt handling.

DRIVERS

ADDING A DEVICE NOT SUPPORTED BY AOS

ADDING EXTRA, OR MODIFYING, AOS-SUPPORTED DEVICE DRIVERS.





INTERRUPT HANDLER

PROCEDURE AFTER RECEIVING AN INTERRUPT

```
STACK MACHINE STATE
INTERRUPTS OFF
GET DEVICE CODE
                THROUGH THE INTERRUPT VECTOR TABLE
                GET THE DEVICE CONTROL TABLE
                        MASK OUT, SAVE & DISABLE LEF
                        INTERRUPTS ON
                        GO TO INTERRUPT SERVICE ROUTINE

                        RETURN FROM THE INTERRUPT
                        SERVICE ROUTINE
                        INTERRUPTS OFF
RESTORE THE MACHINE STATE, PREVIOUS MASK, LEF
INTERRUPTS ON
CONTINUE
```

ADDING/REMOVING DRIVERS

ADDING:

DEFINE THE DEVICE CONTROL TABLE

INTRODUCE THE DCT ENTRY INTO
AOS'S INTERRUPT VECTOR TABLE

SET THE DATA CHANNEL MAP

WRITE TASKS TO START AND/OR TRANSFER DATA
TO/FROM THE DEVICE, AND PROCESS THE DATA

WRITE THE INTERRUPT SERVICE ROUTINE

REMOVING:

DEVICE CODE

INTERRUPT SERVICE ROUTINE

GENERAL PRINCIPLES

GET OUT FAST

MODIFY THE MASK?

SERVICE THE DEVICE

ACCESS DATA
CLEAR/START

TRANSMIT A MESSAGE TO A TASK

EXIT.

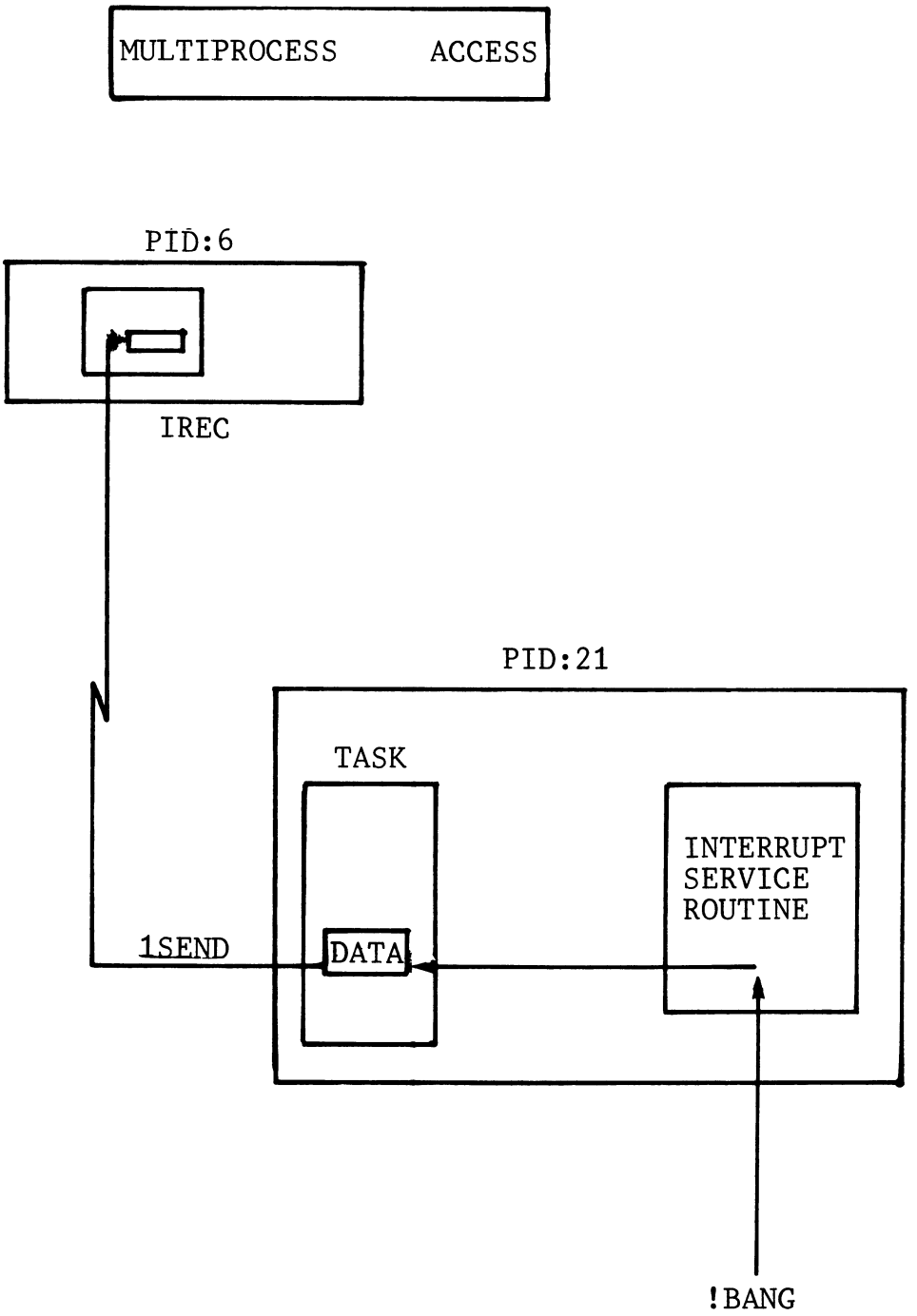
DIRECT ACCESS

TASK LEVEL

ALL DEVICES

PROGRAMMED I/O INSTRUCTIONS

LEF DISABLED



SYSTEM SUPPORTED DEVICES

AOSGEN

ADD TO THE :PER DEVICES

MODIFY ?_____.TMP FILES

PMGR

PATCH

MODULE SYSTEM CALL SUMMARY

SECTION V INTERFACE

MODULE TITLE: A: DRIVERS

?IDEF	IDENTIFY A USER DEVICE AND INTERRUPT SERVICE ROUTINE.
?IRMV	REMOVE A DEVICE'S ENTRY FROM THE INTERRUPT VECTOR TABLE.
?IXIT	EXIT FROM AN INTERRUPT SERVICE ROUTINE.
?IXMT	TRANSMIT A MESSAGE TO A TASK.
?IMSG	RECEIVE A MESSAGE TRANSMITTED FROM AN INTERRUPT SERVICE ROUTINE.
?SMSK	MODIFY THE CURRENT INTERRUPT MASK.
?STMAP	SET THE DATA CHANNEL MAP.

This page intentionally left blank

MODULE OBJECTIVES

Section V INTERFACE

Module Title: B: CONTROL CHARACTERS

Upon successful completion of this module the student will be able to:

1. Describe how to control AOS's reaction to "CONTROL C" sequences.

INTERCEPTING CTRL C

CONSOLE INTERRUPTS ENABLED BY DEFAULT

DISABLE

DEFINE A CONSOLE INTERRUPT TASK

ENABLE

MODULE SYSTEM CALL SUMMARY

SECTION V INTERFACE

MODULE TITLE: B: CONTROL CHARACTERS

?OEBL	ENABLE CONSOLE CONTROL C INTERRUPTS.
?ODIS	DISABLE CONSOLE CONTROL C INTERRUPTS.
?INTWT	DEFINE A TASK THAT IS READIED ON A CTRL C CTRL A.

This page intentionally left blank.

MODULE OBJECTIVES

Section V INTERFACE

Module Title: C: SYSTEM DATA

Upon successful completion of this module the student will be able to:

1. Design a program which communicates with EXEC.
2. Add user defined error codes.
3. Design a program that accesses the various accounting data recorded by AOS.

This page intentionally left blank.

EXEC AND SPOOL

FROM WITHIN A USER PROCESS:

EXEC CAN BE COMMANDED TO
MOUNT AND DISMOUNT MAGNETIC TAPES
AND DISK UNITS

A FILE CAN BE PLACED ON
ANY QUEUE

THE OPERATOR CAN BE REQUESTED
TO CONTROL ANY QUEUE

HELP AND ERROR FILES

HELP

CLI COMMAND

ADD NEW HELP FILES

ERROR

ERMES.SR

CAN BE ACCESSED AND EXTENDED

USER ERROR FILES

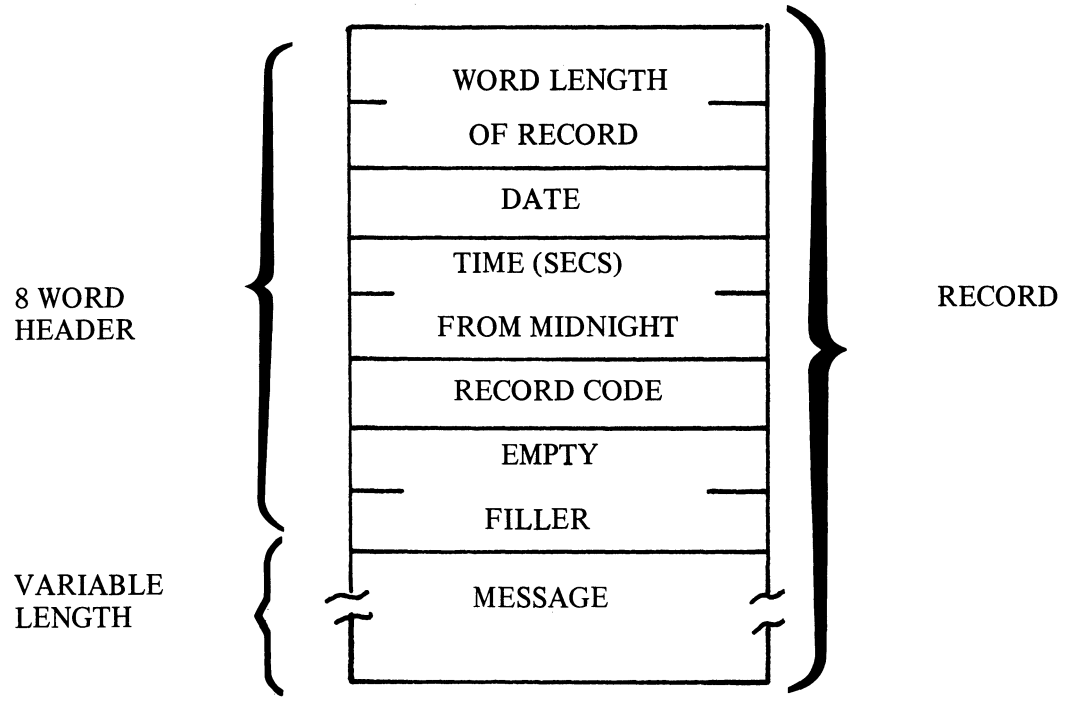
HISTOGRAM AND RUNTIME DATA

MONITORING OF PROCESS RESOURCES

HISTOGRAM
- DYNAMIC
- CPU TIME OF OWN OR OTHER PROCESS
- MEMORY INTERVALS

RUNTIME
- SNAPSHOT
- TIME ELAPSED SINCE PROCESS CREATION
- CPU TIME
- BLOCKS TRANSFERRED
- PAGE USAGE OVER TIME

SYSLOG STRUCTURE



CODE	EVENT	MESSAGE
1	LOG TURNED ON	
2	LOG TURNED OFF	
3	PROCESS TERMINATION	NAME, STATS
4	DEVICE ERROR	UNIT, STATUS, RETRYS,
6	ERCC ERROR	ERROR CODE,ADDRESS
7	POWER FAIL	
1024	CONSOLE CONNECT	USER,DEVICE,TIME
1025	DEVICE MOUNT	USER,DEVICE,TIME
1026	SUPERUSER LOGON	USER
1027	PAGES PRINTED	USER, PAGES
2048-4096	USER EVENT	USER SUPPLIED

SYSLOG

SYSLOG FILE COLLECTS ?SYSLOG

- RUNTIME STATISTICS
- HARDWARE ERRORS
- EXEC DATA
- USER MESSAGES ?LOGEV
  (FROM SUPERUSERS ONLY)

XEQ REPORT

USER ACCOUNTING AND BILLING

- ACCESS SYSLOG FILE
- REQUIRE SYSLOG STRUCTURE

MODULE SYSTEM CALL SUMMARY

SECTION V INTERFACE

MODULE TITLE: C: SYSTEM DATA

?ERMES	OBTAIN A MESSAGE ASSOCIATED WITH AN ERROR CODE.
?EXEC	PERFORM ONE OF THE EXEC FUNCTIONS.
?RUNTM	GET THE PROCESS RUNTIME STATISTICS.
?IHIST	START COLLECTING DATA FOR BUILDING A HISTOGRAM.
?KHIST	STOP COLLECTING HISTOGRAM DATA.
?OPEN ?CLOSE ?READ ?SUSER	CALLS TO ACCESS A SYSLOG FILE.
?ENQUE	PLACE A FILE IN THE SPOOL QUEUE.
?IFPU	INITIALIZE THE FLOATING POINT UNIT.
?SYSLOG	START/STOP LOGGING
?LOGEV	ADD USER MESSAGE TO SYSTEM LOG

MODULE OBJECTIVES

Section V INTERFACE

Module Title: D: SECURING MASTER CONSOLE

Upon successful completion of this module the student will be able to:

1. Describe the techniques for protecting CON0 and also for limiting its power.

CON 0

PROTECTION OF MASTER CONSOLE FORM

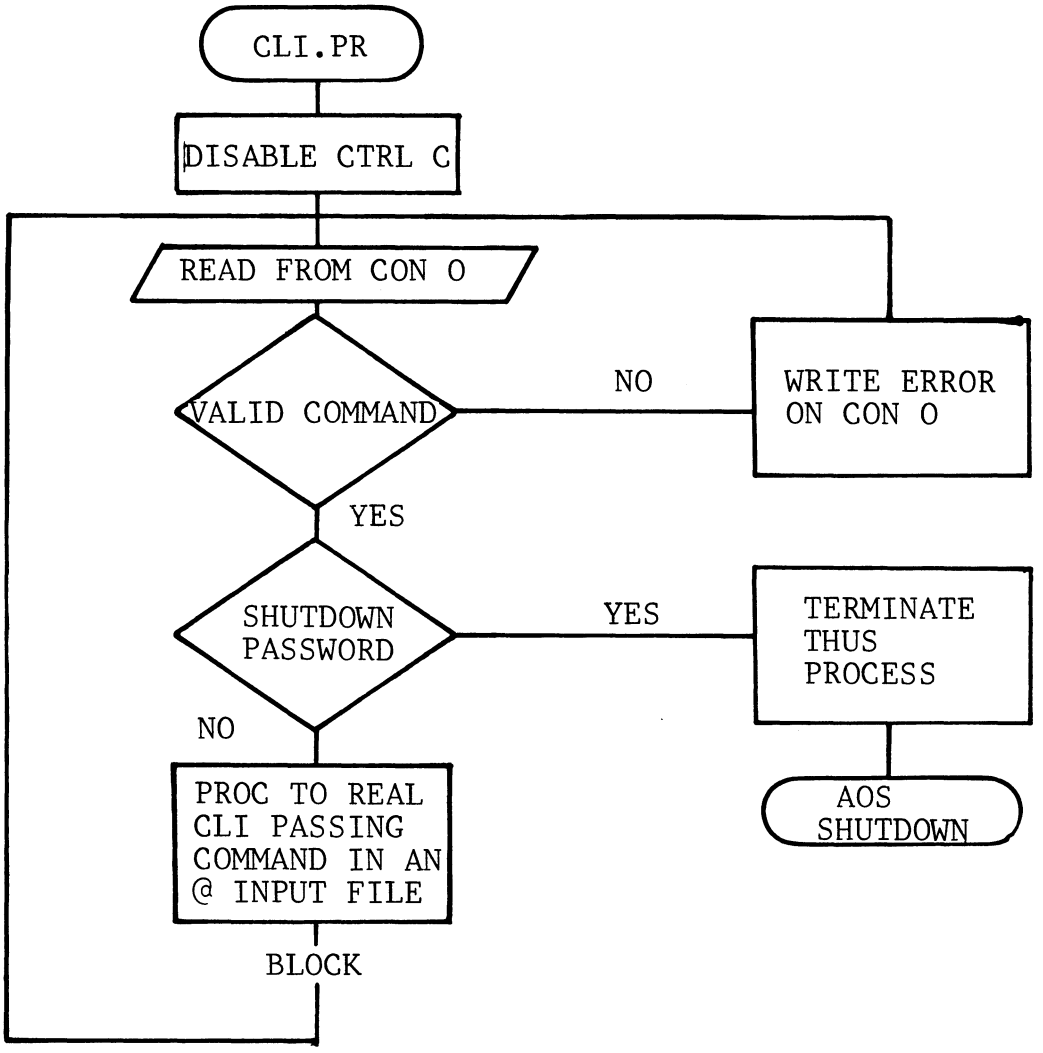
CTR C SEQUENCES
HARMFUL CLI COMMANDS
AOS SHUTDOWN

WRITE YOUR OWN CLI.PR

LOOKUP TABLE OF PERMITTED CLI COMMANDS
DISABLE CTRL C SEQUENCES
PASSWORD FOR BYE

CLI.PR

AOS INITIALIZATION



MODULE SYSTEM CALL SUMMARY

SECTION V INTERFACE

MODULE TITLE: D: SECURING THE MASTER CONSOLE

?PROC	CREATE A SUBORDINATE PROCESS.
?ODIS	DISABLE CONSOLE INTERRUPTS.

This page intentionally left blank.

SECTION GOAL

COURSE TITLE: S309 SYSTEM PROGRAMMER

SECTION VI COMMUNICATIONS

TO UNDERSTAND THE PRINCIPLES OF COMMUNICATIONS WITH EMPHASIS ON THE BINARY SYNCHRONIZATION PROTOCOL.

MODULE OBJECTIVES

Section VI COMMUNICATIONS

Module Title: A: ASYNCHRONOUS

Upon successful completion of this module the student will be able to:

1. Summarize the AOS supported capabilities of asyncronous hardware.

ALM

AOSGEN DEFINED @CON–

MULTITERMINAL SYSTEM
DEFINED SPEED AND CHARACTERISTICS

APPLICATIONS

ONE CONSOLE PER PROCESS - EXEC
ONE PROCESS MULTITASKING MANY CONSOLES

MCA 4038

HARDWARE DESCRIPTION

- MCA RECEIVER AND MCA TRANSMITTER PAIR MAKE UP UNIT
- TWO PAIRS POSSIBLE PER CPU
- UP TO 75' BETWEEN CPU'S
- FULL DUPLEX OPERATION
- DATA CHANNEL TRANSFER SPEEDS
- MCA RECEIVER CAN WAIT INDEFINITELY FOR DATA
- MCA TRANSMITTER TIMEOUT SETTABLE FROM 200 MILLISEC TO 655 SECONDS

SOFTWARE DESCRIPTION

- AOS MCA DRIVER IS A AOS OPTION
- ACCESS TO A CPU IS BY CONVENTIONAL AOS SOFTWARE CHANNEL
- ASSOCIATION WITH A CPU'S UNIT NUMBER

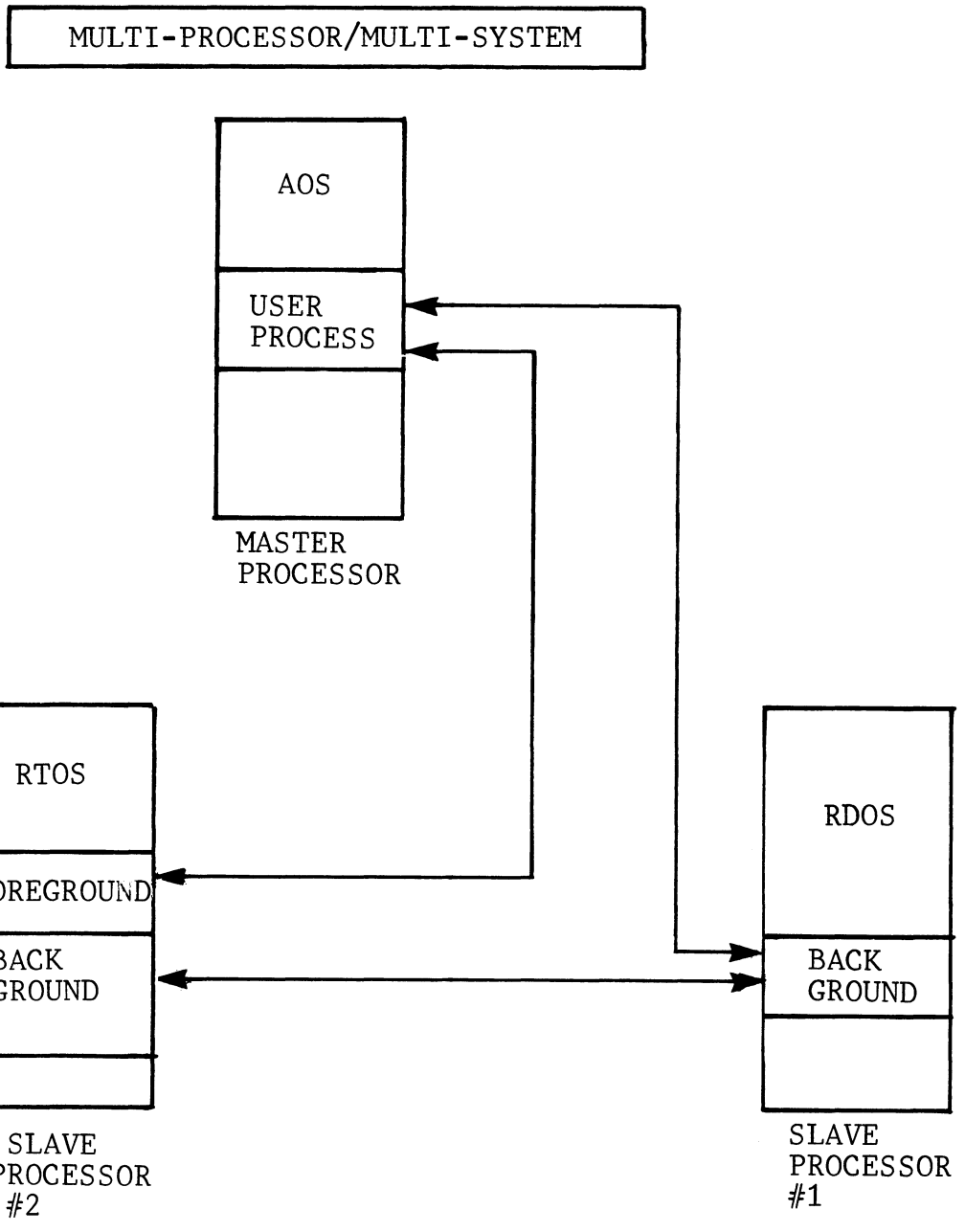
  - 1st MCA CAN ACCESS 14 OTHER CPU'S BESIDES ITSELF

    MCAR: m m m m = 1 to 15
    MCAT: m m DEVICE CODE = 6_8

  - 2nd MCA CAN ACCESS 15 OTHER CPU'S BESIDES ITSELF

    MCAR1: n n n n = Ø to 15
    MCAT1: n n DEVICE CODE: 46_8

PROTOCOL ALLOWS COMMUNICATIONS BETWEEN AOS AND RDOS.



MCA ACCESS

RECORD I/O

OBSERVE PROTOCOL

OPEN READ WRITE

BLOCK I/O

OBSERVE PROTOCOL

GOPEN READ BLOCK WRITE BLOCK

DIRECT BLOCK I/O

NO PROTOCOL

GOPEN READ BLOCK WRITE BLOCK

DATA AND LINE VALIDATION

NO QUEUEING OF MULTIPLE TRANSFERS

MODULE SYSTEM CALL SUMMARY

SECTION VI COMMUNICATIONS

MODULE TITLE: A: ASYNCHRONOUS

?CHAR, ?SCHAR	OBTAIN OR SET A DEVICES CHARACTERISTICS.
?OPEN	ASSOCIATE A CHANNEL NUMBER WITH A CONSOLE OR MCA FILE.
?CLOSE	DISASSOCIATE A CHANNEL NUMBER FROM A CONSOLE OR MCA FILE.
?READ/?WRITE	PERFORM RECORD I/O BETWEEN A PROCESS AND OONSOLE DEVICE OR MCA FILES.
?GOPEN	ASSOCIATE A CHANNEL NUMBER WITH A MCA DEVICE.
?RDB/?WRB	PERFORM BLOCK I/O BETWEEN A PROCESS AND A MCA DEVICE.

MODULE OBJECTIVES

Section VI COMMUNICATIONS

Module Title: B: BINARY SYNCHRONOUS COMMUNICATIONS

Upon successful completion of this module the student will be able:

1. Describe the methods of sending data using synchronous lines.

BSC LINE
ERROR STATISTICS

- ERROR STATISTICS FOR A GIVEN BSC LINE
- CLEARS ALL ERROR COUNTS

- CALL RETURNS

  1. TOTAL NUMBER OF NAK's
  2. NUMBER OF TIMEOUTS
  3. NUMBER OF BLOCK CHECK ERRORS

- BSC ERROR RECOVERY PROCEDURE HANDLED BY AOS

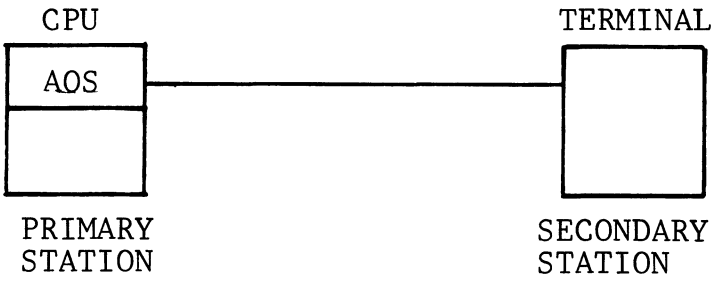
PROTOCOL

ACK 0 ACK 1	Affirmative acknowledgement (alternating)
CRC	Cyclic redundancy check
DLE	Data-link escape
DLE EOT	Disconnect sequence for a switched line in transparent mode
EOT	End of transmission
ETB	End of transmission block
ETX	End of text
ITB	End of intermediate record
NAK	Negative acknowledgement
RVI	Reverse interrupt
SOH	Start of header
STX	Start of text
SYN	Synchronous idle
SYN...	Two or more SYNs
TTD	Temporary text delay
WACK	Wait before next transmission (positive acknowledgement)

POINT - TO - POINT AND MULTIPOINT

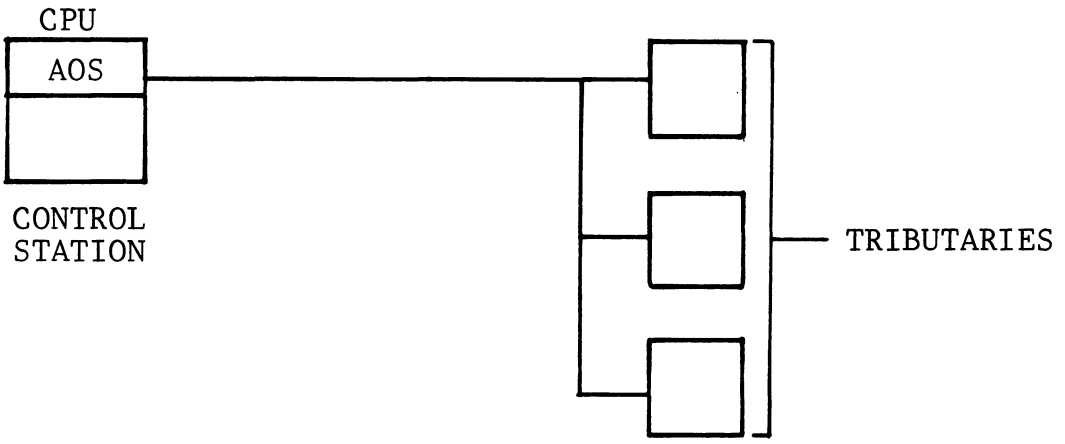
. POINT - TO - POINT

- CONTENTION SYSTEM
- LINE PROTOCOL RESOLVES CONTENTION



. MULTIPOINT

- NON-CONTENTION SYSTEM
- POLLING AND SELECTING RESOLVES CONTENTION



BI - SYNC

- TRANSMISSION OF DATA ON A TIMED BASIS BY MUTUAL AGREEMENT

- CALL CYCLE ON BSC LINE IS ANALAGOUS TO STANDARD FILE I/O

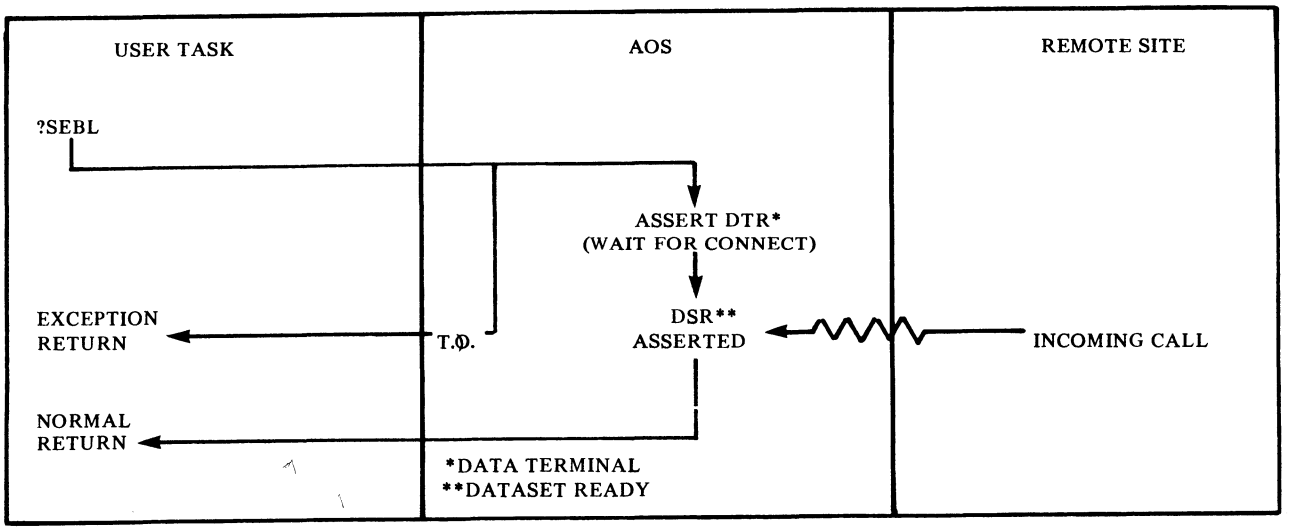
?SEBL	?OPEN
?SSND	?READ
?SRCV	?WRITE
?SDBL	?CLOSE

- PROTOCOL MESSAGE EXCHANGE

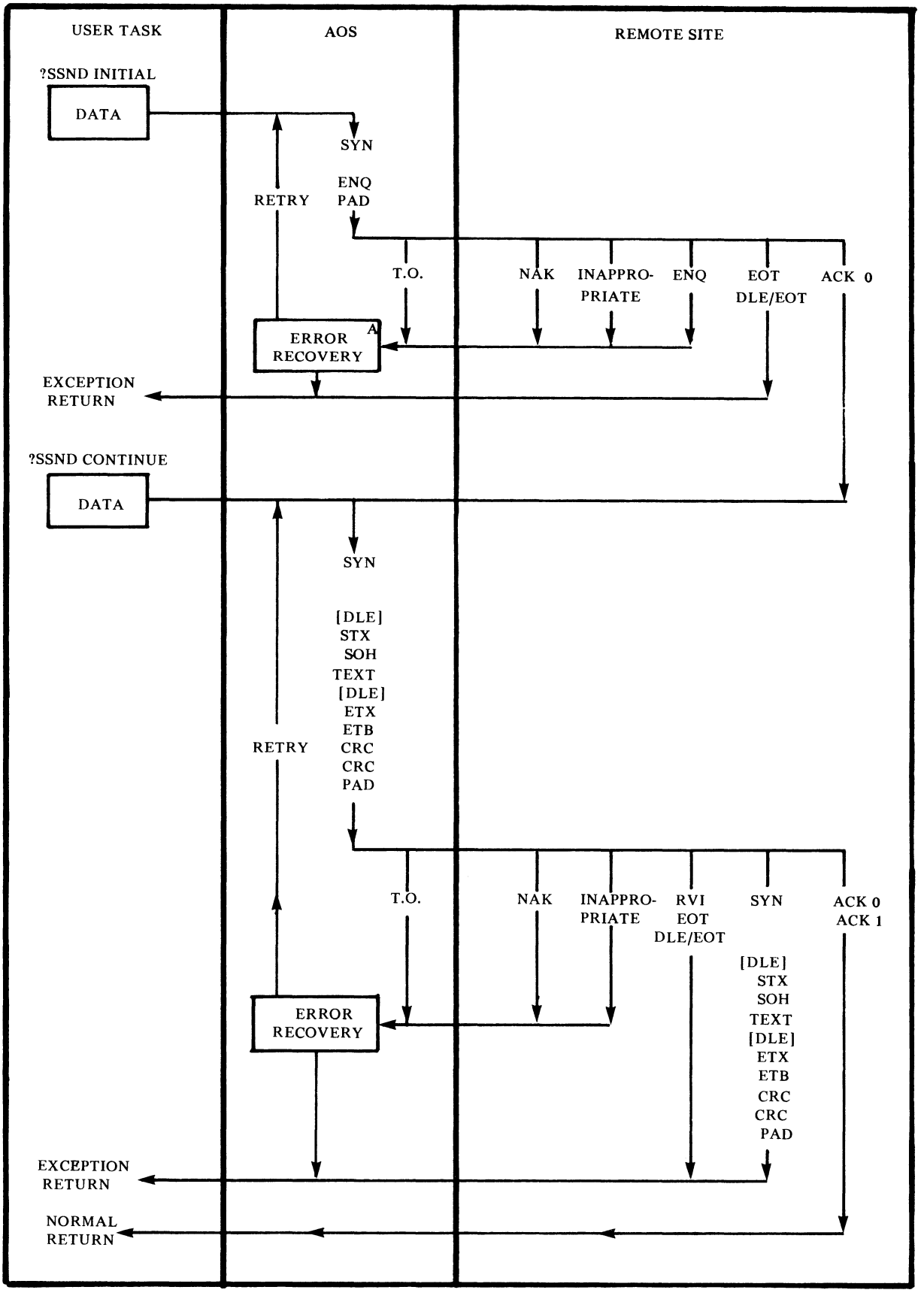
  - SET OF RULES FOR DATA TRANSMISSION MANAGEMENT

- BY-SYNC SUPPORTS DEDICATED AND SWITCHED COMMUNICATION LINES

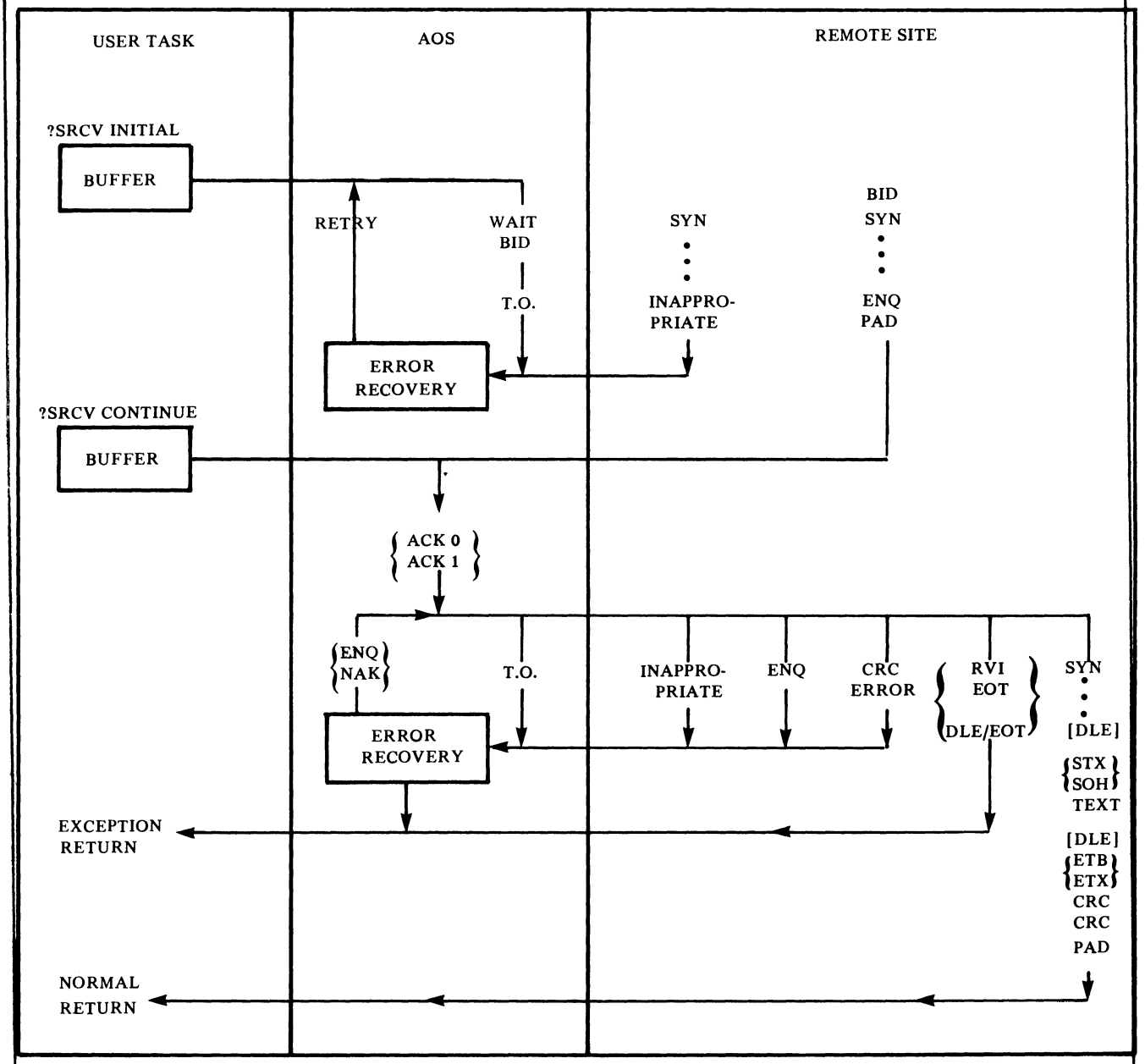
ENABLE A BSC LINE



SEND DATA



RECEIVE DATA



MODULE SYSTEM CALL SUMMARY

SECTION VI COMMUNICATIONS

MODULE TITLE: B: BINARY SYNCHRONOUS

?SEBL	ASSIGNS A CHANNEL NUMBER WITH A BSC LINE AND PERFORMS HARDWARE INITIALIZATION.
?SDBL	DISABLES THE BSC LINE ASSOCIATED WITH A CHANNEL NUMBER.
?SRCV	RECEIVES DATA OVER A BSC LINE.
?SSND	SENDS DATA OVER A BSC LINE.
?SGES	GET THE ERROR STATISTICS FOR A BSC LINE.
?SDPOL	DEFINE A POLLING LIST FOR MULTIPOINT COMMUNICATIONS.
?SDRT	DISABLE A RELATIVE TERMINAL FROM POLLING
?SERT	ENABLE A RELATIVE TERMINAL FOR POLLING