HEWLETT PACKARD

SE LEVEL II

RTE IV

student workbook

SE LEVEL II

RTE IV

data systems
training

22999-90200

August 1978

RECOMMENDED COURSE OUTLINE FOR SE Level II COURSE

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
8:00 9:00	Introduction Hardware Overview	Review Labs Operator Requests -Trace "ON,XYZ" From Keyboard to $XEQ	Resource Numbers LU Locks	Review Hw #3 Re-Entrant Processing -LIBR/LIBX -REIO SAM -Users of Sam	Review Hw #2,4 Power Fail System Library Utilities
			COFFEE BREAK		
10:00 11:00	RTE Overview RTE Modules	Program Dispatching Partition Assignment	Review Hw #1 Program States -State Diagram -$List	-SAM Management I/O Drivers -Initialization -Continuation -Completion -Privileged	Performance Measurement Lab Seminar
12:00			LUNCH		
1:00 2:00	DMS -Phy./Log. Memory -RTE Maps Boot Process -Trace From Front Panel Thru $STRT	I/O Processing Overview Exec Calls -Trace Exec 2 Call from MP Thru I/O	TBG Time Tick -Trace From Interr. to $XEQ		EMA -EMA in Fortran -EMA in Assembler -EMAST, MMAP -.EMAP,.EMIO
			COFFEE BREAK		
3:00 4:00 5:00	CMM4/DBUGR Lab	Completion Parity Errors Lab	Class I/O MTM -Trace From Keyboard Thru RPN Lab	Lab	Exam

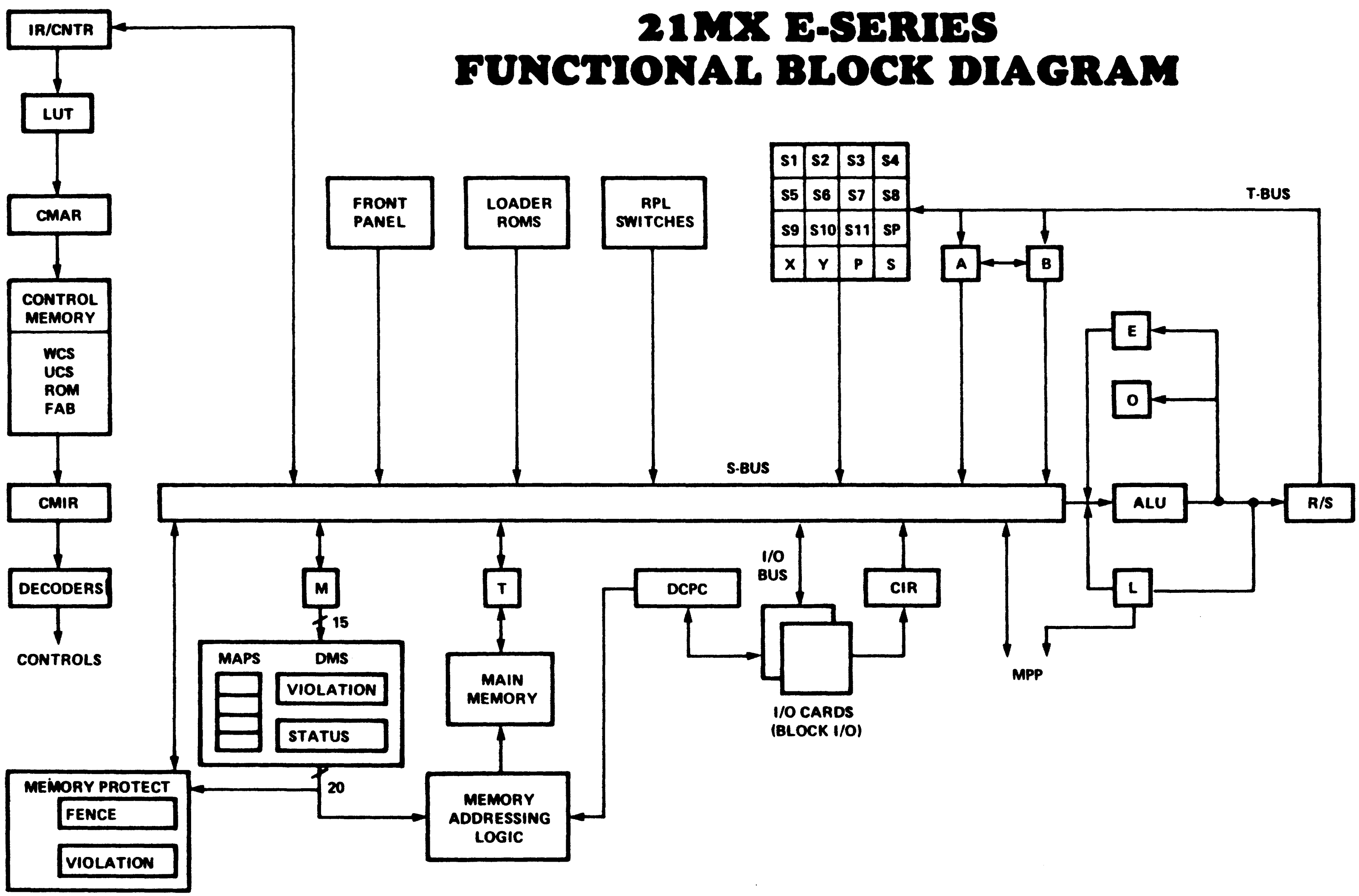
PREFACE

This student work book is to be used with the SE Level II training course and consists of the following sections:

SECTION NUMBERS	TOPICS
1	HARDWARE OVERVIEW
2	RTE OVERVIEW
3	RTE MODULES
4	DMS
5	BOOT PROCESS
6	DBUGR/CMM4
7	OPERATOR REQUESTS
8	PROGRAM DISPATCHING/PARTITION ASSIGNMENT
9	I/O PROCESSING
10	EXEC CALL PROCESSING
11	PARITY ERRORS
12	RESOURCE NUMBERS
13	LOGICAL UNIT LOCK
14	PROGRAM STATES
15	TBG TIME TICK
16	CLASS I/O
17	MULTI-TERMINAL MONITOR (MTM)
18	RE-ENTRANT PROCESSING
19	SYSTEM AVAILABLE MEMORY (SAM)
20	I/O DRIVERS
21	POWER FAIL
22	SYSTEM LIBRARY
23	UTILITIES
24	PERFORMANCE MEASUREMENT
25	EMA
A	SYSTEM TABLES/LISTS
H	HOMEWORK & LAB ASSIGNMENTS

HARDWARE OVERVIEW

21MX E-SERIES FUNCTIONAL BLOCK DIAGRAM



21MX-E SECTIONS

- CONTROL PROCESSOR
  Controls all other sections with microinstructions
- ARITHMETIC/LOGIC
  ALU, L, R/S, O, E, A, B, and 16 RAM registers
- MAIN MEMORY
  M and T registers
- INPUT/OUTPUT
  CIR and I/O cards
- OPERATOR PANEL
  Microprogrammed front panel
- MEMORY PROTECT
- DYNAMIC MAPPING
  Optional
- DUAL CHANNEL PORT CONTROLLER
  Two channels assignable

INTERRUPT SYSTEM

Vectored priority interrupt system with distinct interrupt levels. Each level is associated with a corresponding interrupt location (trap cell) in memory.

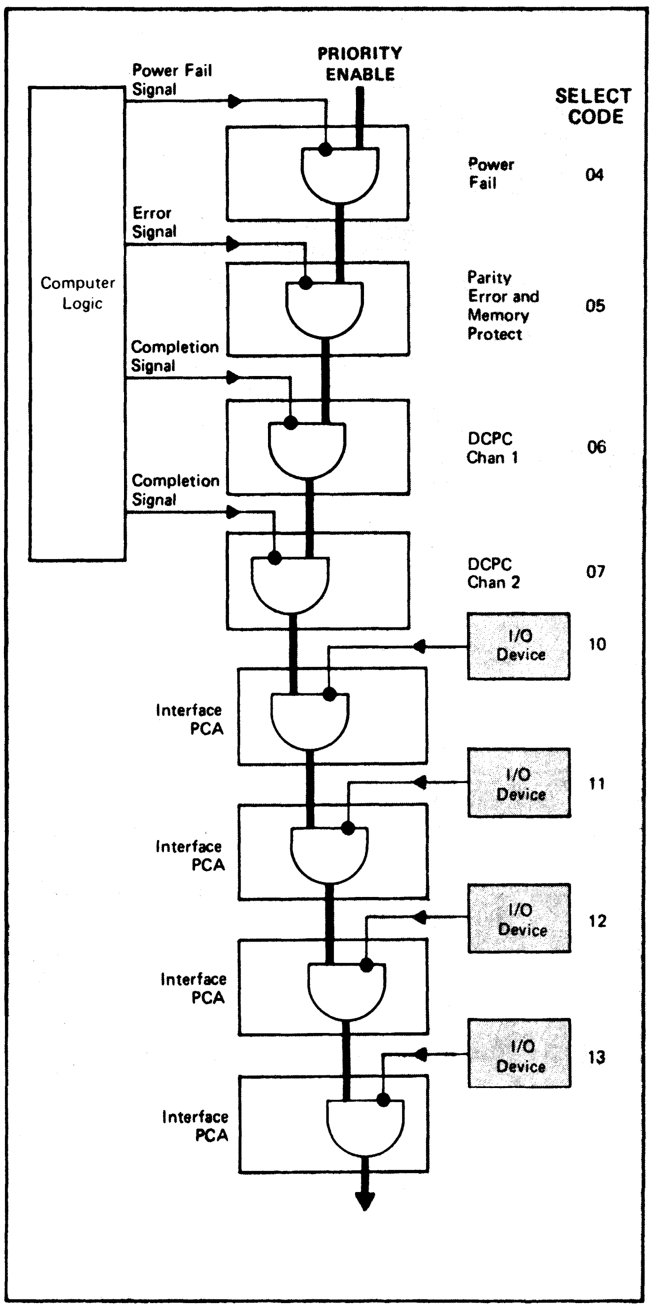
CHANNEL OCTAL	INTERRUPT LOCATION	ASSIGNMENT
00	NONE	INTERRUPT SYSTEM DISABLE/ENABLE
01	NONE	SWITCH REGISTER OR OVERFLOW
02	NONE	DCPC CHANNEL 1 INITIALIZE
03	NONE	DCPC CHANNEL 2 INITIALIZE
04	04	POWER FAIL INTERRUPT/CIR
05	05	MEMORY PARITY/MEMORY PROTECT/DMS INTERRUPT
06	06	DCPC CHANNEL 1 COMPLETION INTERRUPT
07	07	DCPC CHANNLE 2 COMPLETION INTERRUPT
10	10	I/O DEVICE (HIGHEST PRIORITY)
thru	thru	thru
77	77	I/O DEVICE (LOWEST PRIORITY)

An interrupt causes the instruction in the "TRAP CELL" to be executed. The interrupt select code is stored the CIR.

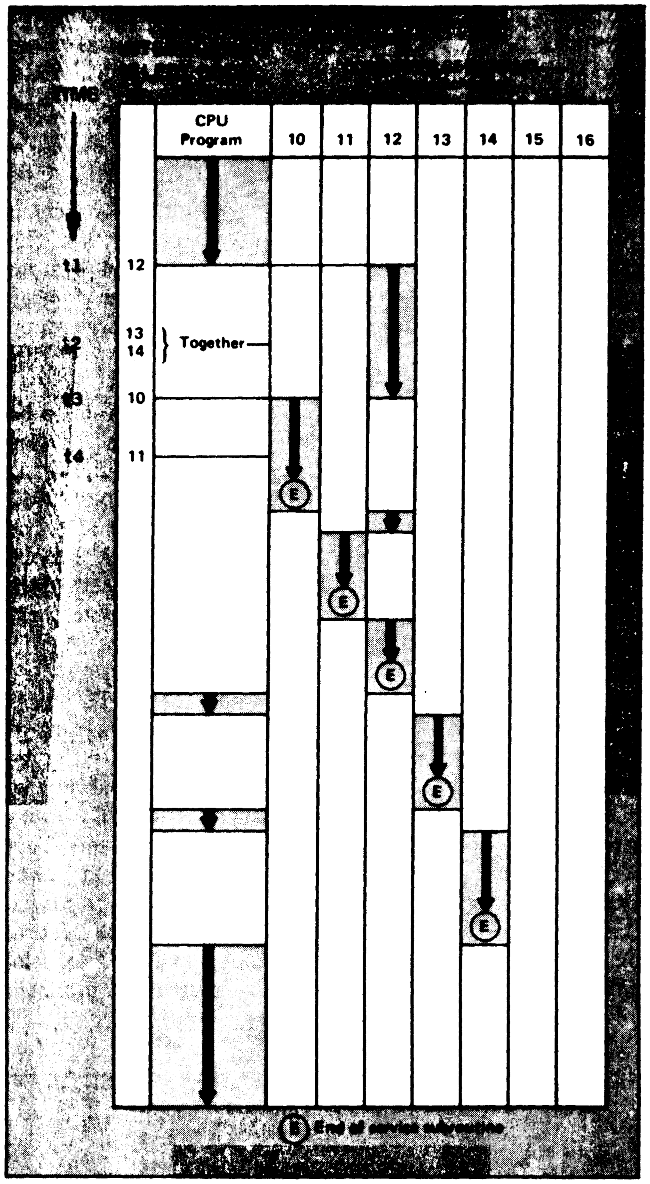
INTERRUPT PRIORITY

Interrupt priority decreases with increasing select code.

A series-linked priority structure allows any device to interrupt and hold off interrupts from devices with higher select codes (SC).



EXAMPLE INTERRUPT SEQUENCE



NOTE: RTE turns off the interrupt system while a service routine or driver is executing unless a privileged interface card is present.

REQUIREMENTS FOR AN INTERRUPT

1. Interrupt system is enabled. (STF 0)
2. Device flag flip-flop is set.
3. Device control flip-flop is set. (STC)
4. Device has priority.
5. Interrupt recognition is enabled.

- Interrupt system is enabled/disabled with a STF 0/CLF 0.
- Flag bit is used by a device to request service from the computer.
- Control bit is used to enable/disable a device.
- Interrupts are inhibited until the succeeding instruction is executed for:

JMP indirect	STC	CLC	SFS (E series)
JSB indirect	STF	CLF	SFC (E series)

DCPC

Provides a direct path, software switchable, between memory and I/O devices. Two DCPC channels are available that operate on a cycle-stealing basis in the following priority:

DCPC1
DCPC2
CPU

DCPC OPERATION:

1. Initialize the DCPC channel with the I/O devices select code, transfer direction, buffer address, and word count.

2. Data transfer is accomplished on a word-by-word basis under automatic control of the DCPC hardware. This eliminates interrupting to a device driver after each word transfer.

3. DCPC completion interrupt is generated when the data transfer is finished. Optionally, the device also generates an interrupt upon completion of the data transfer.

MEMORY PROTECT

Protects a selected block of memory from a settable fence address downward. An interrupt on select code (SC) 5 is generated when:

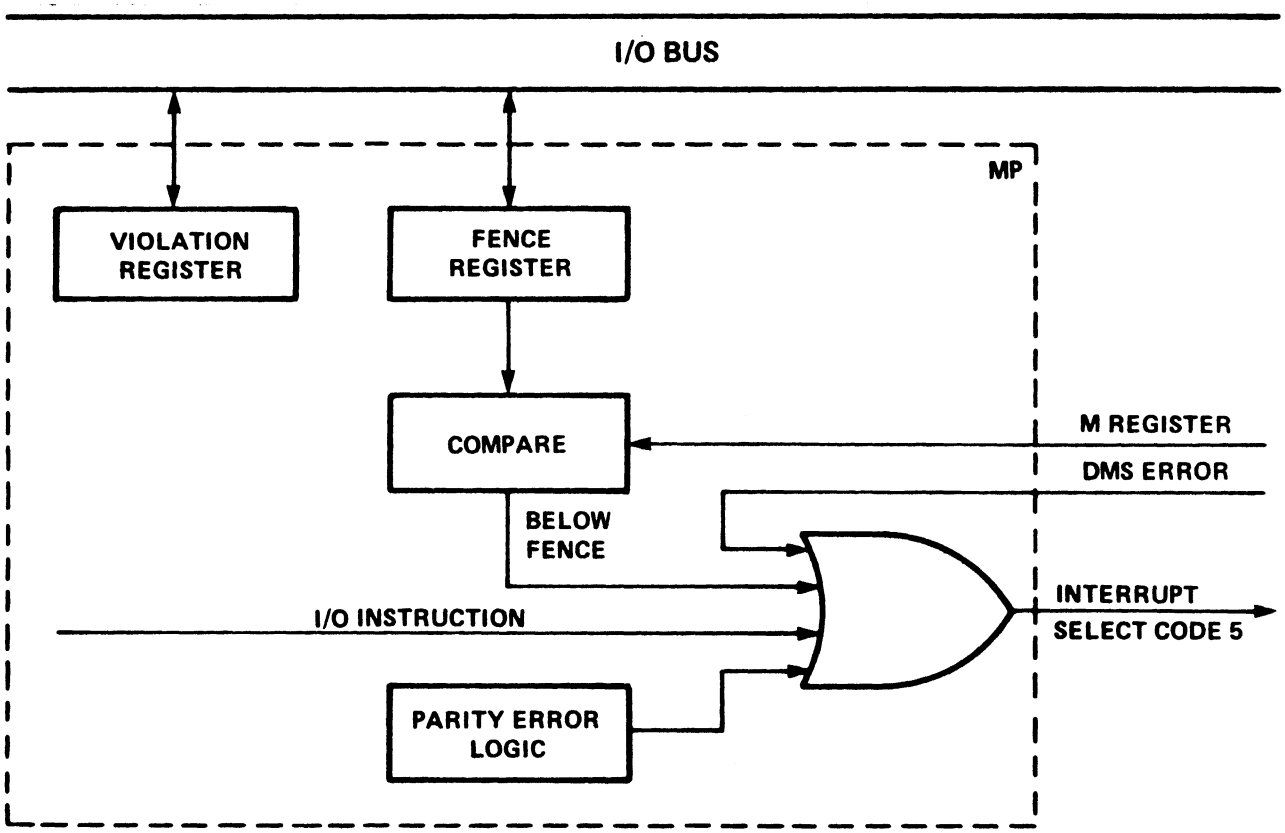
- One of the following instructions directly or indirectly modifies or enters a memory location below the fence:

  DST,ISZ,JLY,JMP,JPY,JSB,MVB,MVW,SAX,SAY,SBX,SBY,STA,STB,STX,STY

- Any I/O instruction is attempted except instructions to select code (SC) 1.
- HLT instruction is attempted.
- A DMS instruction is attempted. (Many of the DMS instructions are allowed if the system map is enabled.)

The violation register will contain the address of the illegal instruction.

MEMORY PROTECT LOGIC



THE CAUSE OF A SC (SELECT CODE) 5 INTERRUPT CAN BE DETERMINED BY:

MP – BIT 15 OF VIOLATION REGISTER IS CLEAR*
PE – BIT 15 OF VIOLATION REGISTER IS SET*
DMS – FLAG ON SC 5 IS SET (SFS 5 OR SFC 5)

*VIOLATION REGISTER IS READ WITH: LIA/B 5

RTE
OVERVIEW

```
                            RTE IV
PROG TYPES  *-----------------------------------------*
    0       *         DRIVER PARTITION #n             *
            *-----------------------------------------*
            *                  .                      *
            ~                  .                      ~
            ~                  .                      ~
            *-----------------------------------------*
    0       *         DRIVER PARTITION #2             *
            *-----------------------------------------*
   (16)     *           SAM ($CNFG)                   *
            *-----------------------------------------*-+
            *              PERR4                      * |
            *-----------------------------------------* |
            *              OCMD4                      * |
            *-----------------------------------------* |
            *              $ALC                       * |
            *-----------------------------------------* |
            *              SCHD4                      * |
            *-----------------------------------------* |   MEMORY
    0       *              $TRN4                      * |-RESIDENT
            *-----------------------------------------* |   SYSTEM
            *              EXEC4                      * |
            *-----------------------------------------* |
            *              RTIO4                      * |
            *-----------------------------------------* |
            *              $ASC4                      * |
            *-----------------------------------------* |
            *              RTIME                      * |
            *-----------------------------------------* |
            *              DISP4                      *-+
            *-----------------------------------------*-+
            *                                         * |  TAT,MATA,MPFT,
            *                                         * |  KEYWORD,ID
    13      *          TABLE AREA II                  * |  SEG, CLASS,
            *                                         * |  RN'S, LU SWTCH
            *-----------------------------------------*-+
     0      *        SYSTEM DRIVER AREA               *
            *-----------------------------------------*
            *            BG COMMON                    *
            *-----------------------------------------*
            *            RT COMMON                    *
            *-----------------------------------------*
    30      *              SSGA                       *
            *-----------------------------------------*
    0       *         DRIVER PARTITION #1             *
            *-----------------------------------------*-+
            *              sam                        * | INTERRUPT,
            *                                         * | DRT,DVMP,
    15      *          TABLE AREA I                   * | EQT,TB3X
            *-----------------------------------------*-+ 2000
            *         SYSTEM BASE PAGE                *
            *-----------------------------------------*    0
```

RTE IV

```
PROG TYPES   *--------------------------------------* 1000K MAX
             *                                      *
             *                                      *
 2  3  4  5  *       DISC RESIDENT PARTITION        *
10 11 12 18  *                 #n                   *
19 20 26 27  *--------------------------------------*
28           *     DISC RES. PART. n BASE PAGE      *
             *--------------------------------------*
             *                                      *
          -----                                  -----
          -----                                  -----
             *                                      *
             *--------------------------------------*
             *                                      *
             *       DISC RESIDENT PARTITION        *
             *                 #1                   *
             *                                      *
             *--------------------------------------*
             *                                      *
             *           DRP 1 BASE PAGE            *
             *                                      *
             *--------------------------------------*
             *                                      *
             *                                      *
             *    SYSTEM AVAILABLE MEMORY           *
             *           EXTENSION                  *
             *             (SAM)                    *
             *                                      *
             *                                      *
             *--------------------------------------*
             *                                      *
             *                                      *
             *                                      *
             *                                      *
             *                                      *
 1  9        *              MEMORY                  *
17 25        *              RESIDENT                *
             *                                      *
             *                                      *
             *                                      *
             *                                      *
             *--------------------------------------*
             *                                      *
  6  14      *      MEMORY RESIDENT LIBRARY         *
             *                                      *
             *--------------------------------------*
             *      MEMORY RESIDENT BASE PAGE       *
             *--------------------------------------*
             *                                      *
             *                                      *
```

DRIVER PARTITIONS

Driver partitions contain one or more device drivers. All driver partitions are the same size and only the partition containing the driver currently being used is included in the user's logical map. The minimum partition size is two pages.

SYSTEM DRIVER AREA (SDA)

This area contains all drivers not allocated to a driver partition. SDA is not included in the large BG disc resident map and is optional in the memory resident map. Drivers should be put into SDA for the following reasons:

A. Drivers greater than 2K words would be included in SDA to reduce the size of driver partitions and thus increase the potential size of type 4 programs.

B. Privileged drivers are put into SDA since they must always be present in the system map. This results from the fact that an interrupt from a privileged device enables the system map and then jumps directly to the privileged section of the driver.

C. Drivers that do their own mapping must also be put into SDA. Since RTE enters self-mapping drivers with the system map enabled these drivers (like privileged drivers) must always be present in the system map.

There is a restriction to placing drivers in SDA that do not do their own mapping. Drivers in SDA may only be used for class I/O or buffered output requests from programs that do not have SDA in their logical maps. This includes all type 4 programs and possibly memory resident programs.

```
                   SYSTEM DISC
                       LU2
*-----------------------------------------*  255
*              FMP DIRECTORY              *
*-----------------------------------------*
*                                         *
*                                         *
-----                                 -----
-----                                 -----
*                                         *
*                                         *
*          FMP-OWNED TRACKS               *
*              (FMGR)                     *
*                                         *
*                                         *
*                                         *
*-----------------------------------------*  100
*                                         *
*        SYSTEM AVAILABLE TRACKS*         *
*                                         *
*               SWAPPING                  *
*              USER TRACKS                *
*             GLOBAL TRACKS               *
*         ON-LINE LOADED PROGS            *
*-----------------------------------------*  25
*                                         *
*          ENTRY POINT DIRECTORY          *
*                                         *
*-----------------------------------------*
*                                         *
*   DISC RESIDENT LIBRARY SUBROUTINES     *
*         (TYPE (6), 7, AND 14)           *
*                                         *
*-----------------------------------------*
*                                         *
*       GENERATION LOADED PROGRAMS        *
*                                         *
*-----------------------------------------*
*                                         *
*               RTE SYSTEM                *
*                                         *
*-----------------------------------------*
*            SYSTEM BASE PAGE             *
*-----------------------------------------*
*             BOOT EXTENSION              *
*-----------------------------------------*  0
```

* SYSTEM ALLOCATES FROM THE TOP TRACK DOWN; USER ALLOCATES FROM THE BOTTOM TRACK UP

ENTRY POINT DIRECTORY

- Addresses of memory resident system modules, tables, lists, and drivers

examples:	$XEQ	$CIC	EXEC
	$MATA	$IOUP	I.05
	$XSIO	$RQST	C.05
	$LIST	$ALC	I.12

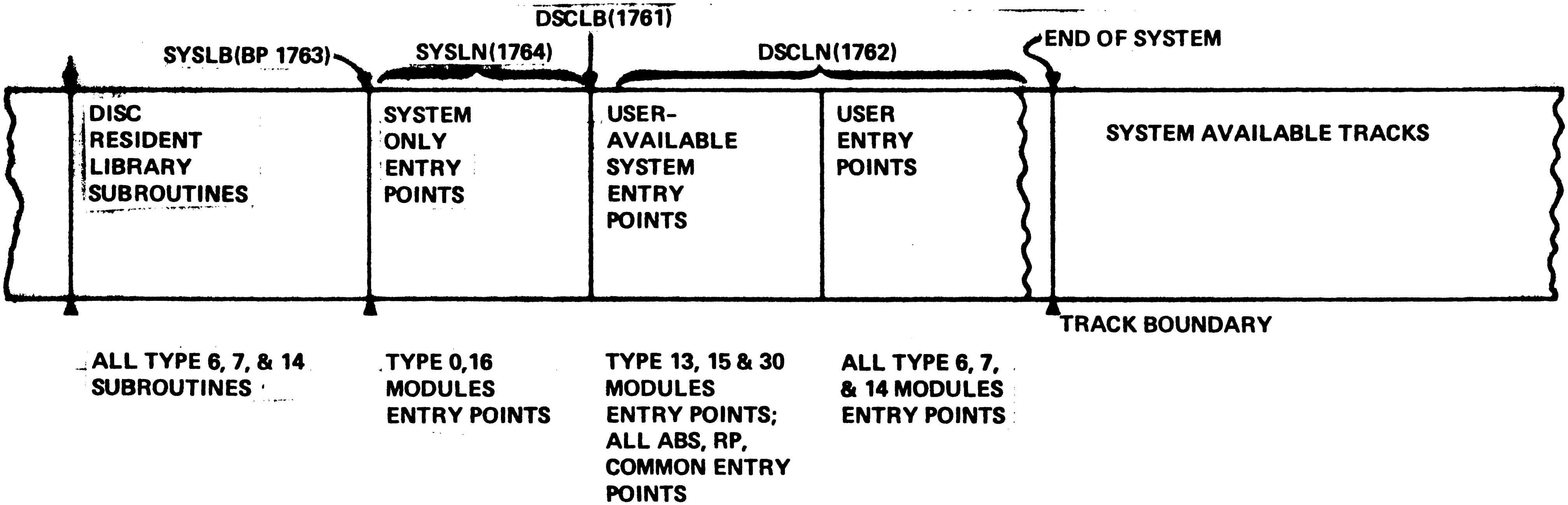
- Track and sector of disc resident library subroutines

examples:	RNRQ	READF	COS
	RMPAR	ABREG	ABS
	PARSE	DBUGR	DMAX1
	CLOSE	REIO	CSQRT

- Microcode replacement values

examples:	FLOAT	.MVW	.DLD
	IFIX	.DIV	.GOTO

ENTRY POINT DIRECTORY



NOTE: SYSLN & DSCLN CONTAIN THE NUMBER OF 4-WORD ENTRIES

LIBRARY ENTRY POINTS LIST

FORMAT:	word 1	name 1,2
	word 2	name 3,4
	word 3	name 5, flag bits
	word 4	value
flag bits:	000	memory resident entry point
	001	disc resident subroutine
	010	common entry point
	011	absolute
	100	replace

LGTAT PRINTOUT
(LONG FORM)

:RU,LGTAT,,1

TRACK ASSIGNMENT TABLE   & =PROG ^ =SWAP

TRACK	0	1	2	3	4	5	6	7	8	9
0	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	SYSTEM	JOB &	AUTOR&
10	LOADR&	GASP &	FMGR &	FMGR0&	FMGR2&	FMGR4&	FMGR6&	FMGR8&	TVST4&	ASMB0&
20	XREF &	SYSTEM	F4.4 &	F4.0 &	DRSTR&	DRSTR&	DSAVE&	LIBRY	LIBRY	LIBRY
30	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY	LIBRY
40	LIBRY	ENTS	SYSTEM	MEAS7&	FMGR	--	GLOBA	--	--	--
50	--	--	--	--	--	--	--	--	--	--
60	--	--	--	--	--	--	--	--	--	--
70	--	--	--	--	--	--	--	--	--	--
80	--	--	--	--	--	--	--	--	--	--
90	--	--	--	--	--	--	--	--	--	--
100	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
110	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
120	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
130	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
140	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
150	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
160	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
170	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
180	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
190	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP	FMP
200	FMP	FMP	D.RTR							

SYSTEM OWNED TRACK

GLOBALLY OWNED TRACK

SYSTEM ENTRY POINTS

RELOCATABLE SYSTEM LIBRARY

LOADED PROGRAMS "F4.4" AND "MEAS7" STORED HERE (PROGRAM NAME FOLLOWED BY "&")

AUXILIARY DISC

```
  0     --     --     --     --     --     --     --     --     --     --
 10     --     --     --     --     --     --     --     --     --     --
 20     --     --     --     --     --     --     --     --     --     --
 30     --     --     --     --     --     --     --     --     --     --
 40     --     --     --     --     --     --     --     --     --     --
 50     --     --     --     --     --     --     --     --     --     --
 60     --     --     --     --     --     --     --     --     --     --
 70     --     --     --   FMG07^ FMG07^ FMG21^ FMG21^ FMGR  ^ FMGR  ^ X..07^
 80     LG     LG     LG     LG     LG     LG     LG     LG     LG     LG
 90     LG     LG     LG     LG     LG     LG     LG     LG     LG     LG
100    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
110    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
120    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
130    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
140    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
150    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
160    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
170    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
180    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
190    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
200    FMP   D.RTR  D.RTR

THE LS TRACK(S) START AT TRACK   44   OF LU    2

TOTAL AVAILABLE TRACKS =        127
LARGEST CONTIGUOUS TRACK BLOCK =    73
```

FREE TRACK

DIRECTORY TRACK

FILE MANAGER TRACK

LOAD-AND-GO TRACK

SWAPPED PROGRAMS "FMG21" AND "X . . 07" STORED HERE (PROGRAM NAME FOLLOWED BY " ∧ ")

NOTE:

The "&" suffix denotes the original loaded copy of a program created by the generator or loader. The " ∧ " suffix denotes a copy of a suspended program which has been swapped by RTE.

SYSTEM BASE PAGE COMMUNICATION AREA

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
01645	XIDEX	Address of current program's ID extension
01646	XMATA	Address of current program's MAT entry
01647	XI	Address of index register save area
01650	EQTA	FWA of Equipment Table
01651	EQT#	Number of EQT entries
01652	DRT	FWA of Device Reference Table, word 1
01653	LUMAX	Number of logical units in DRT
01654	INTBA	FWA of Interrupt Table
01655	INTLG	Number of Interrupt Table Entries
01656	TAT	FWA of Track Assignment Table
01657	KEYWD	FWA of keyword block
I/O MODULE/DRIVER COMMUNICATION		
01660	EQT1	
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EQT5	Addresses of first 11 words of
01665	EQT6	current EQT entry (see 01771 for
01666	EQT7	last four words
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11	
01673	CHAN	Current DCPC channel number
01674	TBG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY
SYSTEM REQUEST PROCESSOR/EXEC COMMUNICATION		
01676	RQCNT	Number of request parameters -1
01677	RQRTN	Return point address
01700	RQP1	
01701	RQP2	
01702	RQP3	Addresses of request parameters (set
01703	RQP4	for a maximum of nine parameters)
01704	RQP5	
01705	RQP6	
01706	RQP7	
01707	RQP8	
01710	RQP9	

SYSTEM BASE PAGE COMMUNICATION AREA (continued)

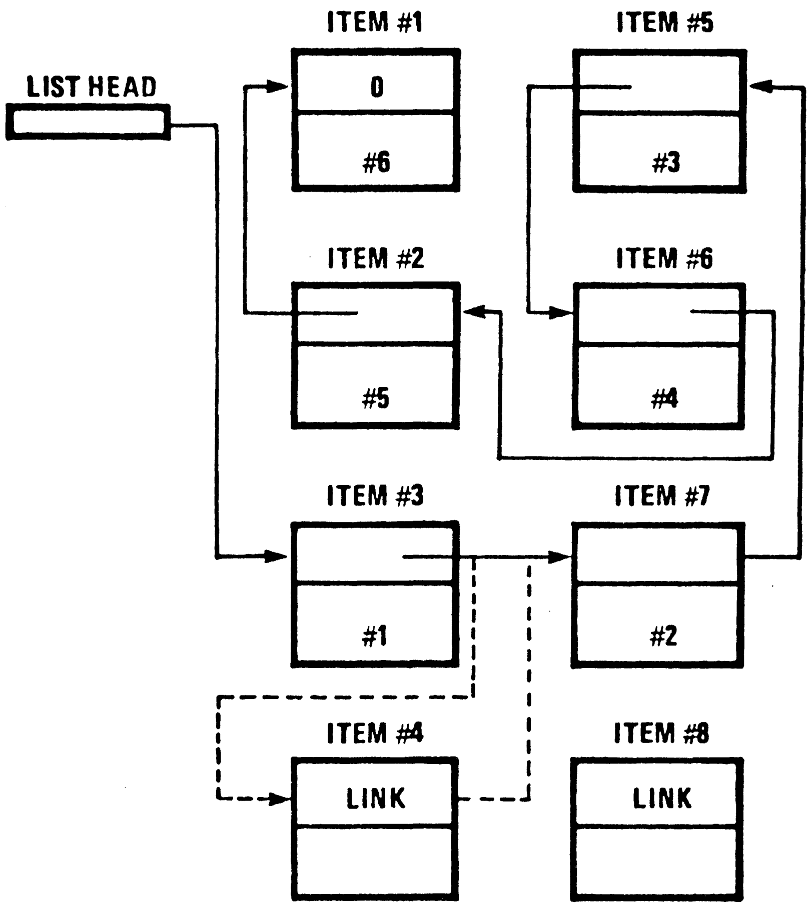
Octal Location	Contents	Description
SYSTEM LISTS ADDRESSES		
01711	SKEDD	Schedule list
01713	SUSP2	Wait Suspend list
01714	SUSP3	Available Memory list
01715	SUSP4	Disc Allocation list
01716	SUSP5	Operator Suspend list
PROGRAM ID SEGMENT DEFINITION		
01717	XEQT	ID segment address of current program
01720	XLINK	Linkage
01721	XTEMP	Temporary (five words)
01726	XPRIO	Priority word
01727	XPENT	Primary entry point
01730	XSUSP	Point of suspension
01731	XA	A-register at suspension
01732	XB	B-register at suspension
01733	XEO	E and overflow register suspension
SYSTEM MODULE COMMUNICATION FLAGS		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy interface flag
01740	IDSDA	Disc address of first ID segment
01741	IDSDP	Position within disc sector
MEMORY ALLOCATION BASES DEFINITION		
01742	BPA1	FWA user base page link area
01743	BPA2	LWA user base page link area
01744	BPA3	FWA user base page link
01745	LBORG	FWA of resident library area
01746	RTORG	FWA of real-time COMMON
01747	RTCOM	Length of real-time COMMON
01750 D	RTDRA	FWA of real-time partition
01751 D	AVMEM	LWA+1 of real-time partition
01752	BGORG	FWA of background COMMON
01753	BGCOM	Length of background COMMON
01754 D	BGDRA	FWA of background partition

SYSTEM BASE PAGE COMMUNICATION AREA (continued)

Octal Location	Contents	Description
UTILITY PARAMETERS		
01755	TATLG	Negative length of track assignment table
01756	TATSD	Number of tracks on system disc
01757	SECT2	Number of sectors/track on LU2 (system)
01760	SECT3	Number of sectors/track on LU3 (aux.)
01761	DSCLB	Disc address of user available library entry points
01762	DSCLN	Number of user available library entry points.
01763	SYSLB	Disc address of system library entry points
01764	SYSLN	Number of system library entry points
01765	LGOTK	LGO: LU#, starting trrack, number of tracks (same format as ID segment word 28)
01766	LGOC	Current LGO track/sector address (same format as ID segment word 26)
01767	SFCUN	LS: LU# and disc address (same format as ID segment word 26)
01770	MPTFL	Memory protect ON/OFF flag (0/1)
01771	EQT12 \	
01772	EQT13 \	Address of last four
01773	EQT14 /	words of current EQT
01774	EQT15 /	
01775 D	FENCE	Memory protect fence address
01777	BGLWA	LWA memory background partition

D letter indicates the contents of the location are set dynamically by the dispatcher.

LINKED LISTS



Linked lists provide a mechanism for quickly ordering and accessing blocks of memory and their constants. **It consists of a** list head or 'starting point', and a word in each entry pointing to the next entry.

New entries can be included and old entries removed by re-setting only 1 link.

SYSTEM TABLES

- ID segments, long, short, & extensions
- Equipment Table
- Device Reference Table
- Interrupt Table
- Track Assignment Table
- Class Table
- LU Switch Table
- Resource Number Table
- Keyword Block
- ID Extension Table
- Memory Allocation Table
- Memory Protect Fence Table
- Driver Mapping Table
- Track Map Table

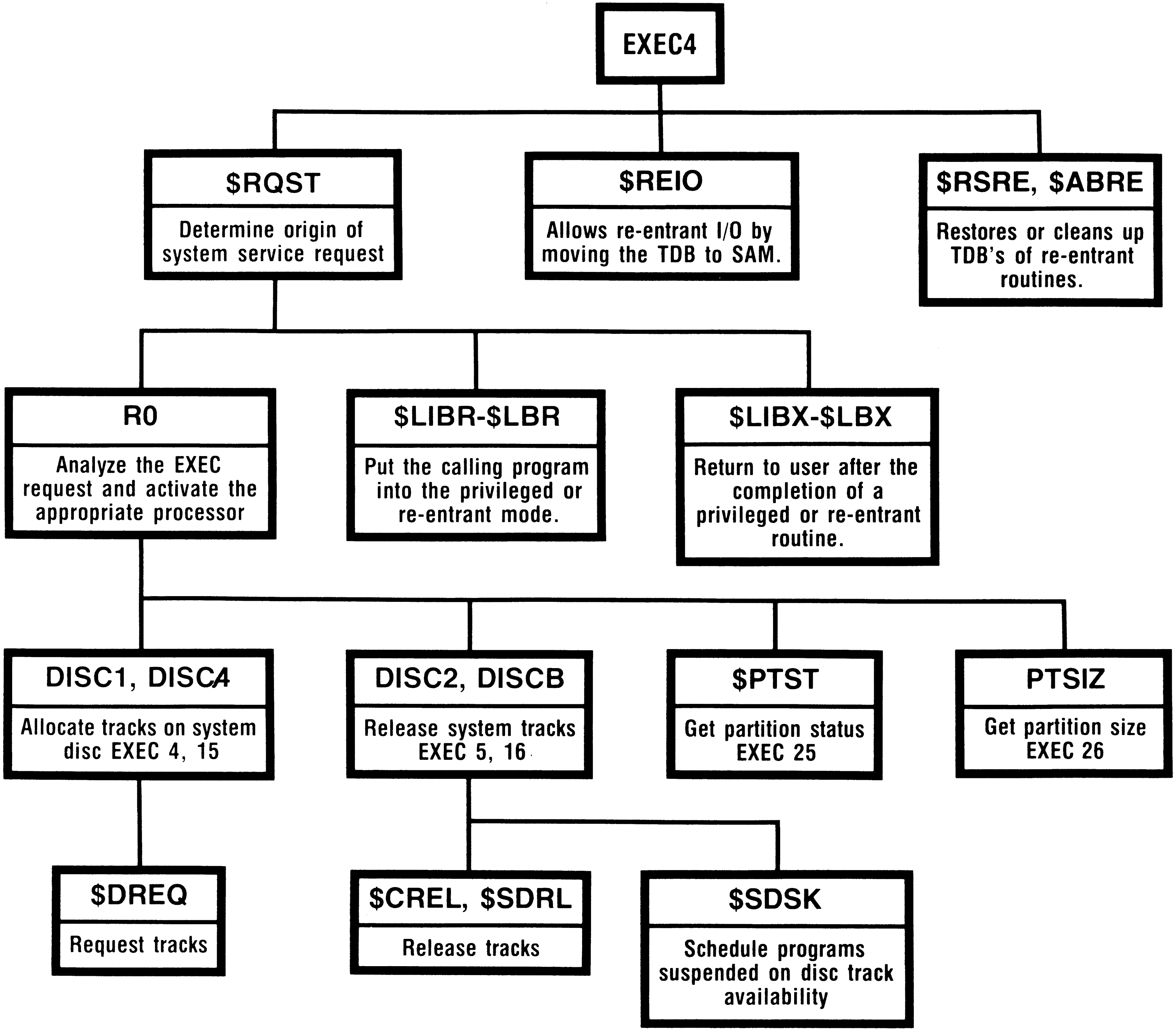
SYSTEM LISTS

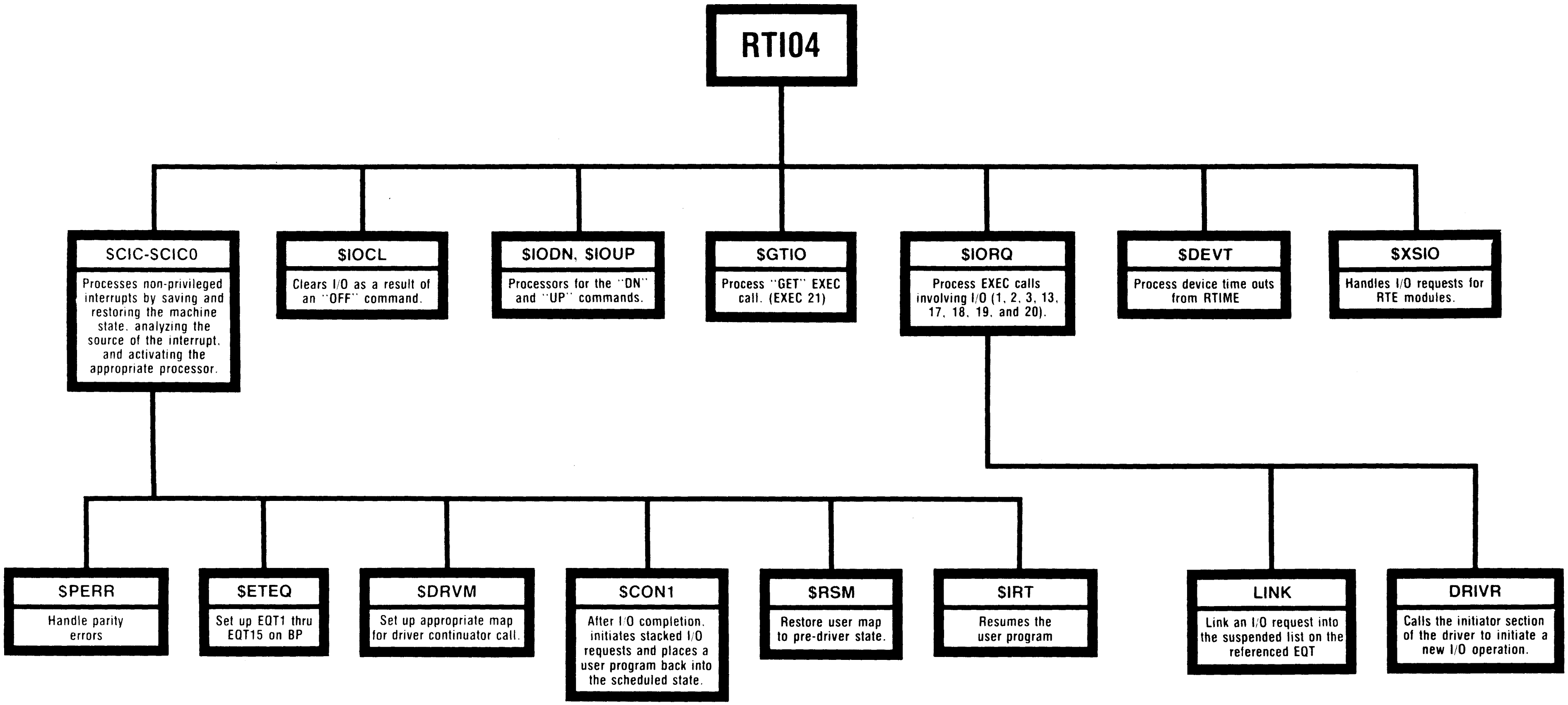
- Schedule List
- General Wait List
- Available Memory Suspend List
- Disc Allocation Suspend List
- Operator Suspend List
- I/O Suspend Lists
- Free SAM List

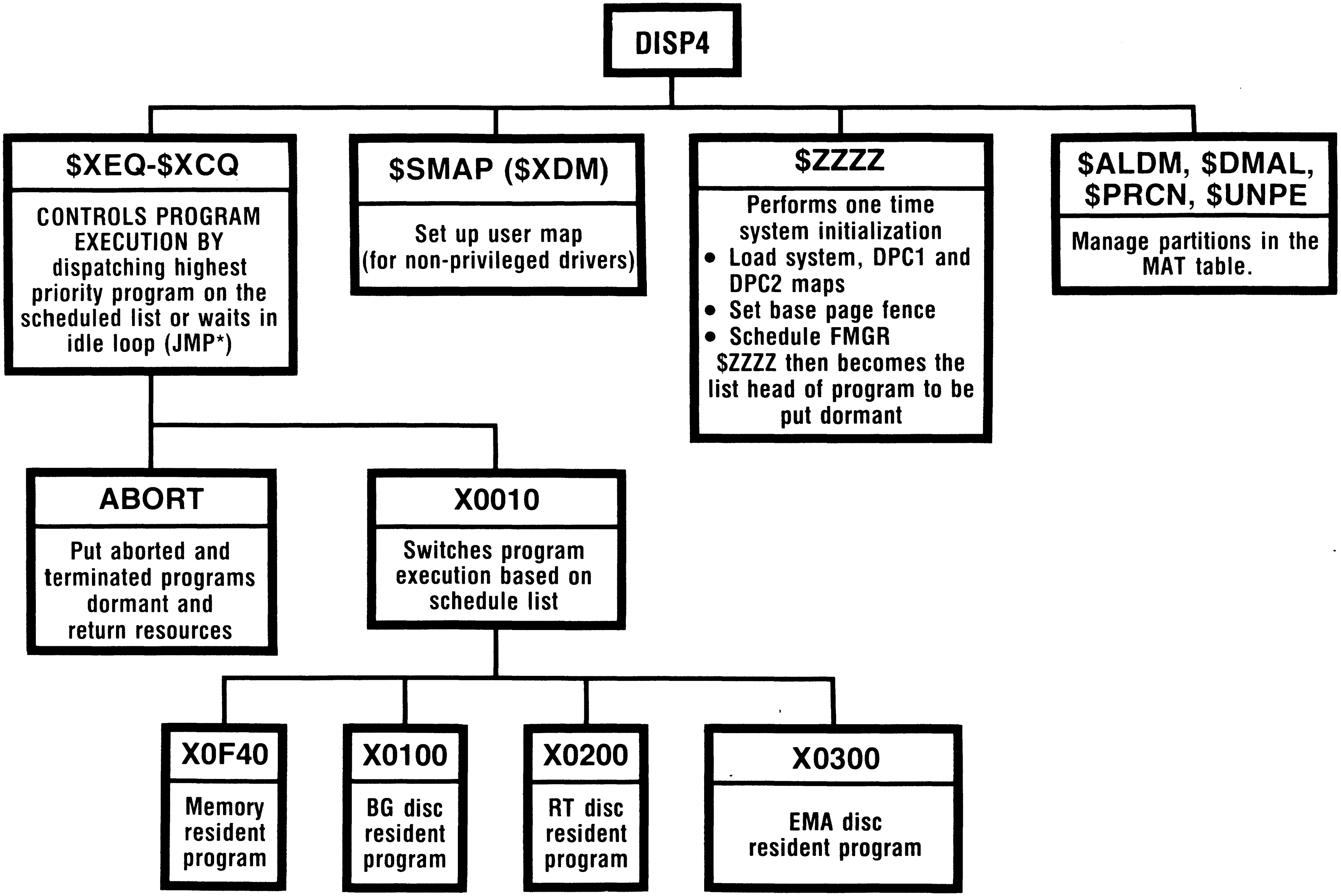
OVERVIEW CHART

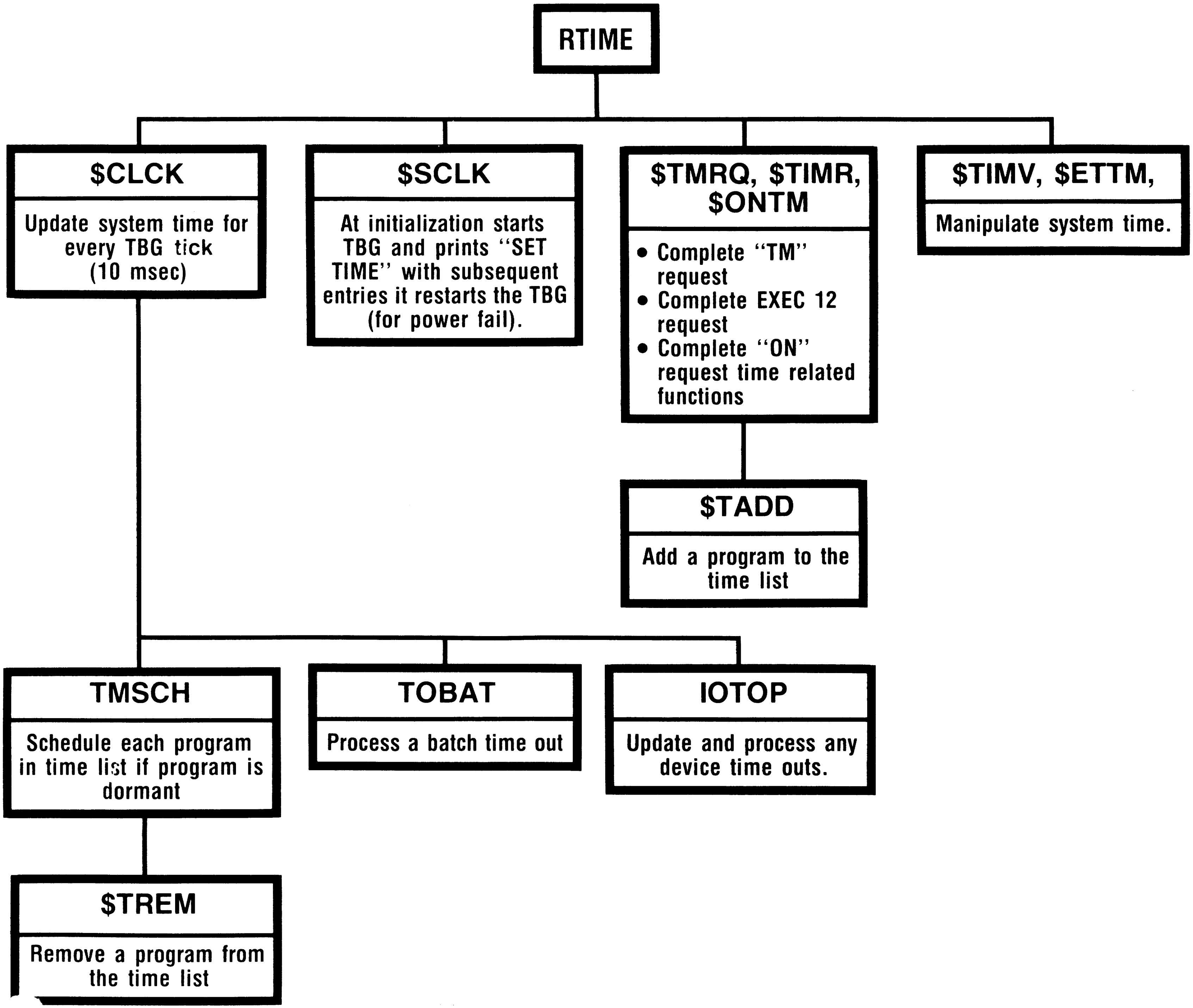
OF RTE TABLES/LISTS

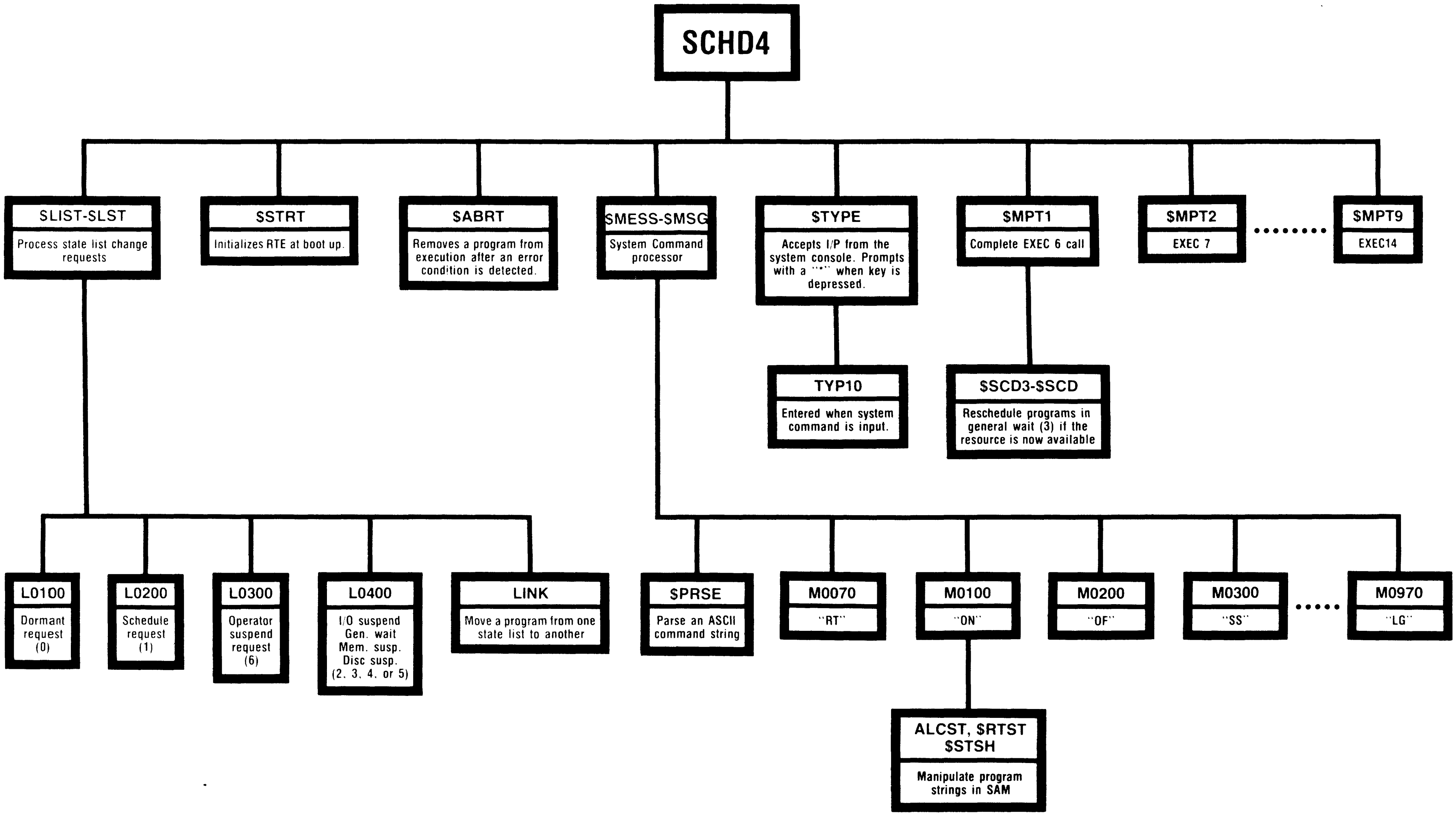
RTE
MODULES

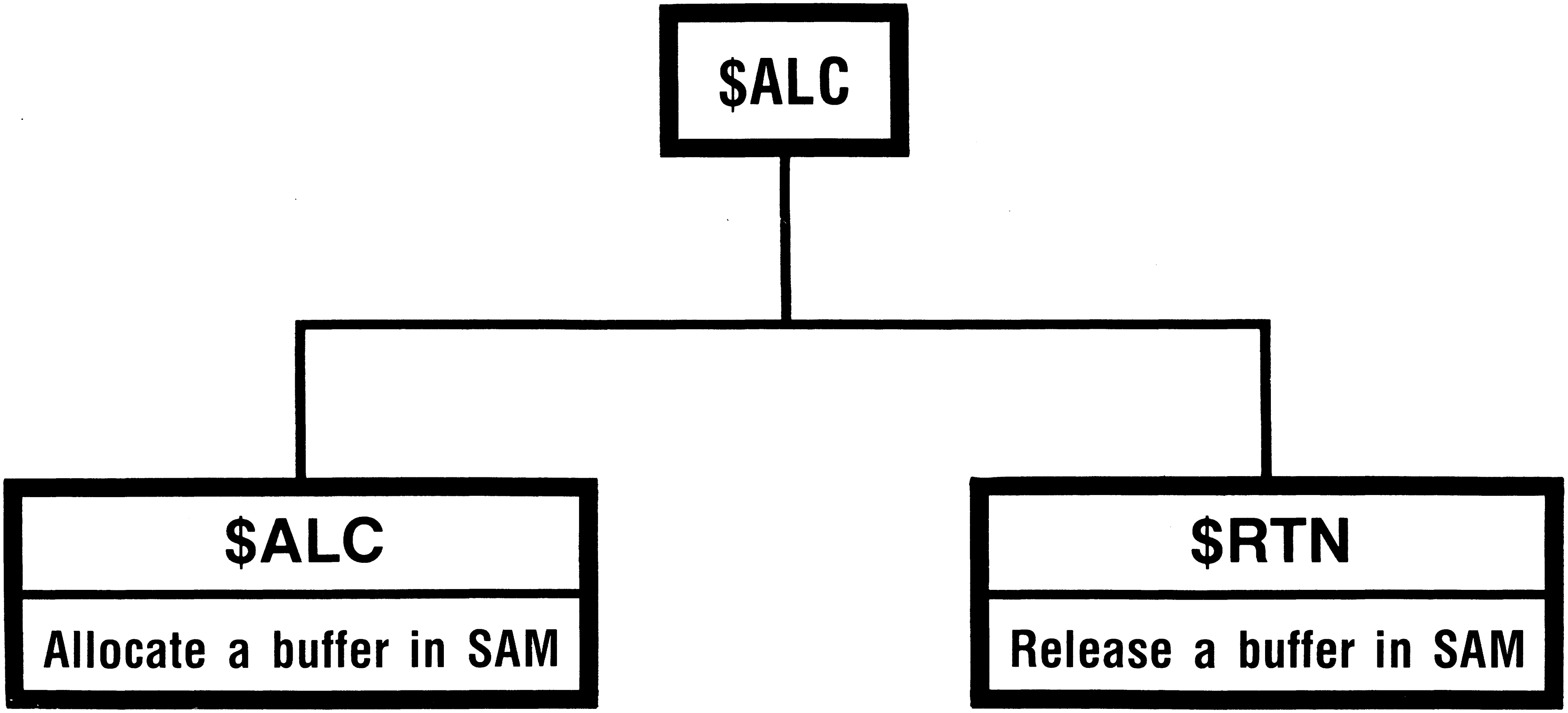

EXEC4
$RQST
Determine origin of system service request
$REIO
Allows re-entrant I/O by moving the TDB to SAM.
$RSRE, $ABRE
Restores or cleans up TDB's of re-entrant routines.
R0
Analyze the EXEC request and activate the appropriate processor
$LIBR-$LBR
Put the calling program into the privileged or re-entrant mode.
$LIBX-$LBX
Return to user after the completion of a privileged or re-entrant routine.
DISC1, DISCA
Allocate tracks on system disc EXEC 4, 15
DISC2, DISCB
Release system tracks EXEC 5, 16
$PTST
Get partition status EXEC 25
PTSIZ
Get partition size EXEC 26
$DREQ
Request tracks
$CREL, $SDRL
Release tracks
$SDSK
Schedule programs suspended on disc track availability

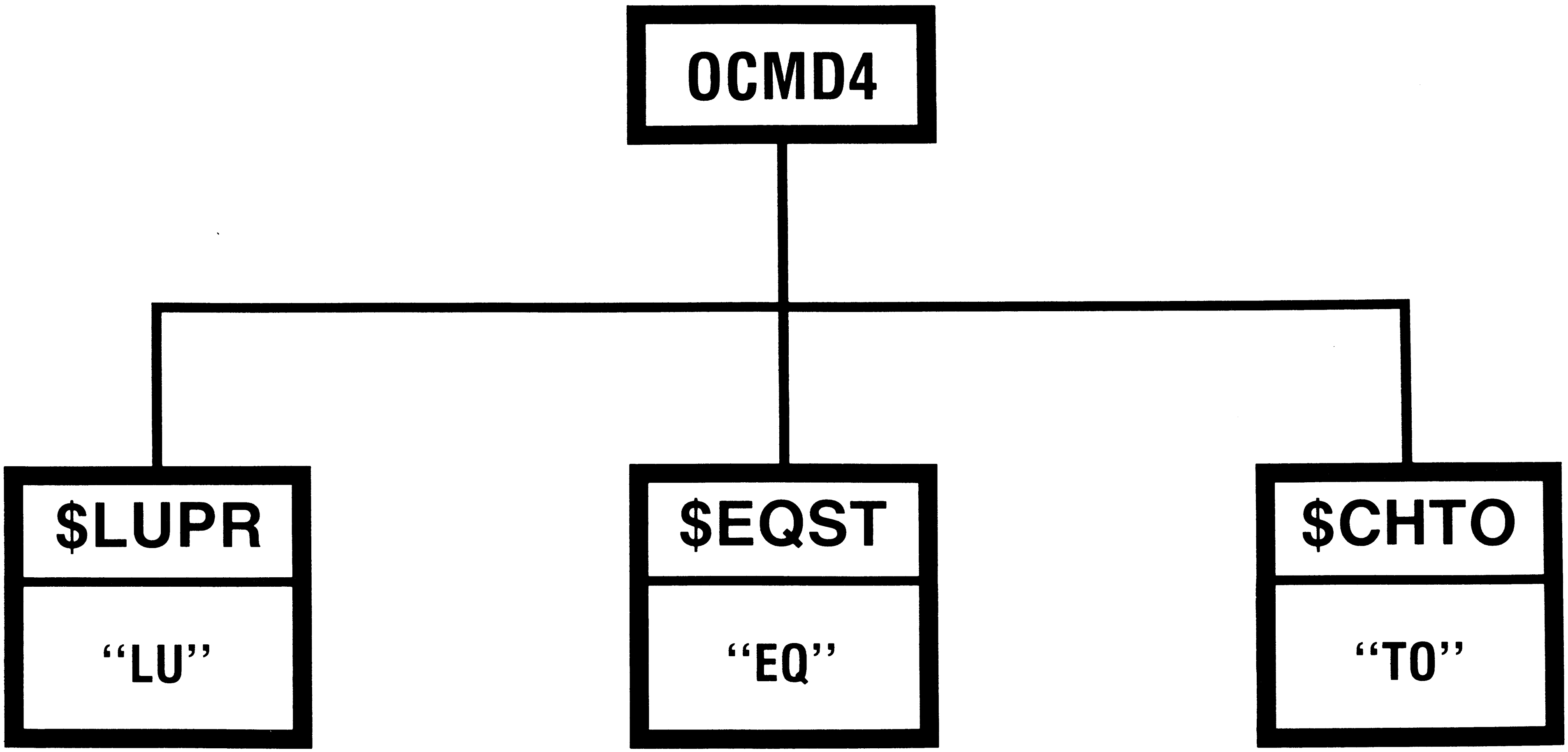

RTI04
SCIC-SCICO
Processes non-privileged interrupts by saving and restoring the machine state, analyzing the source of the interrupt, and activating the appropriate processor.
$IOCL
Clears I/O as a result of an "OFF" command.
$IODN, $IOUP
Processors for the "DN" and "UP" commands.
$GTIO
Process "GET" EXEC call. (EXEC 21)
$IORQ
Process EXEC calls involving I/O (1, 2, 3, 13, 17, 18, 19, and 20).
$DEVT
Process device time outs from RTIME
$XSIO
Handles I/O requests for RTE modules.
SPERR
Handle parity errors
$ETEQ
Set up EQT1 thru EQT15 on BP
SDRVM
Set up appropriate map for driver continuator call.
SCON1
After I/O completion, initiates stacked I/O requests and places a user program back into the scheduled state.
$RSM
Restore user map to pre-driver state.
$IRT
Resumes the user program
LINK
Link an I/O request into the suspended list on the referenced EQT
DRIVR
Calls the initiator section of the driver to initiate a new I/O operation.


DISP4
$XEQ-$XCQ
CONTROLS PROGRAM EXECUTION BY dispatching highest priority program on the scheduled list or waits in idle loop (JMP*)
$SMAP ($XDM)
Set up user map (for non-privileged drivers)
$ZZZZ
Performs one time system initialization
• Load system, DPC1 and DPC2 maps
• Set base page fence
• Schedule FMGR
$ZZZZ then becomes the list head of program to be put dormant
$ALDM, $DMAL, $PRCN, $UNPE
Manage partitions in the MAT table.
ABORT
Put aborted and terminated programs dormant and return resources
X0010
Switches program execution based on schedule list
X0F40
Memory resident program
X0100
BG disc resident program
X0200
RT disc resident program
X0300
EMA disc resident program


RTIME
$CLCK
Update system time for every TBG tick (10 msec)
$SCLK
At initialization starts TBG and prints "SET TIME" with subsequent entries it restarts the TBG (for power fail).
$TMRQ, $TIMR, $ONTM
• Complete "TM" request
• Complete EXEC 12 request
• Complete "ON" request time related functions
$TIMV, $ETTM,
Manipulate system time.
$TADD
Add a program to the time list
TMSCH
Schedule each program in time list if program is dormant
TOBAT
Process a batch time out
IOTOP
Update and process any device time outs.
$TREM
Remove a program from the time list


SCHD4
SLIST-SLST
Process state list change requests
$STRT
Initializes RTE at boot up.
$ABRT
Removes a program from execution after an error condition is detected.
$MESS-$MSG
System Command processor
$TYPE
Accepts I/P from the system console. Prompts with a "*" when key is depressed.
$MPT1
Complete EXEC 6 call
$MPT2
EXEC 7
$MPT9
EXEC14
TYP10
Entered when system command is input.
$SCD3-$SCD
Reschedule programs in general wait (3) if the resource is now available
L0100
Dormant request (0)
L0200
Schedule request (1)
L0300
Operator suspend request (6)
L0400
I/O suspend Gen. wait Mem. susp. Disc susp. (2, 3, 4, or 5)
LINK
Move a program from one state list to another
$PRSE
Parse an ASCII command string
M0070
"RT"
M0100
"ON"
M0200
"OF"
M0300
"SS"
M0970
"LG"
ALCST, $RTST $STSH
Manipulate program strings in SAM


$ALC
$ALC
Allocate a buffer in SAM
$RTN
Release a buffer in SAM


OCMD4
$LUPR
"LU"
$EQST
"EQ"
$CHTO
"TO"

SYSTEM VARIABLES

I. TABLE AREA I

```
$PVCN - Level count for privileged calls
$PVST - DMS status for privileged calls
$MTM - Class number for PRMPT/$R$PN
$OPSY - Operating system identification
```

II. TABLE AREA II

```
$MATA - Address of MAT table
$MCHN - Max. mother partition size
$MBGP - Max. BG partition size
$MRTP - Max. RT partition size
$DLTH - No. pages per driver partition
$DVPT - Driver ptn. map register number (org 0)
$TIME - System time (10's ms (2 words), day/year (1 word))
$BATM - Batch timer (2 words 10's ms)
$DLP  - Logical address of type 4 programs for loading
$PLP  - Logical address of type 2+3 programs for loading
$ENDS - No. of pages in system up to the SAM extension
$MPFT - Address of memory protect fence table
$BGFR - Listhead of free BG partitions
$RTFR - Listhead of free RT partitions
$CFR  - Listhead of free Mother partitions
$IDEX - Address of ID extension table
$MRMP - Address of MP map
$MPSA - No. of pages and starting page of SAM
$MPS2 - No. of pages and starting page of SAM extension
$SDA  - Starting page of SDA (org. 0)
$SDT2 - No. of pages in SDA and Table Area II
$CMST - Start page of common (org. 0)
$COML - No. of pages in common
$MNP  - Max. number of partitions
$DVMP - Address of driver mapping table
$EMRP - LWA of MR program area
$RLB  - Logical start page of MR library
$RLN  - No. pages in MR library
$SBTB - Disc address of driver partitions #2 onward
        # of pages for driver partitions #2 onward
        Disc address of memory resident base page
        # of pages for memory resident base page
        Disc address of memory resident lib/programs
        # of pages for memory resident lib/programs
```

```
MODULE                     MODULE  SIZE (OCTAL)   **MXREF**
NAME    MODULE IDENT.    BPAG  MAIN  COMM
-----------------------------------------------------------
FILE NAME: %CR4S1:: 32767:       5:    112:      0 IS ON LU    35

SCSY4     92067-16014 REV.1805 780125
                         00000 00000 00000

DISP4     92067-16014 REV.1805 780317
                         00000 05363 00000 ENT= $RENT $BRED $ZZZZ $XCQ
                                           ENT= $ALDM $DMAL $SMAP $PRCN
                                           ENT= $XDM  $MAXP $UNPE
                                           EXT= $MRMP $MATA $MPFT $BGFR
                                           EXT= $RTFR $EMRP $RSRE $ABRT
                                           EXT= $XSIO $DREQ $WATR $TIME
                                           EXT= $DREL $TRRN $SZIT $ABRE
                                           EXT= $LIST $RTST $SGAF $ERMG
                                           EXT= $MCHN $MBGP $MRTP $CFR
                                           EXT= $WORK $IOCL $IRT  $IDLE
                                           EXT= $DVPT $IDEX $CMST $SDA
                                           EXT= $SDT2 $MNP  $XDMP
                                           END=  05214

RTIME     92067-16014 REV.1805 780104
                         00000 00605 00000 ENT= $TADD $CLCK $TREM $TIMV
                                           ENT= $ETTM $TIMR $ONTM $TMRQ
                                           ENT= $SCLK
                                           EXT= $INER $DEVT $LIST $XEQ
                                           EXT= $ERMG $MSEX $SYMG $IDSM
                                           EXT= $WORK $BATM $TIME
                                           END=  00216

$ASC4     92067-16014 REV.1805 780125
                         00000 00073 00000 ENT= $OPER $ERIN $NOPG $ILST
                                           ENT= $NOLG $LGBS $NMEM
                                           END=  00050

RTIO4     92067-16014 REV.1805 780310
                         00002 05321 00000 ENT= $CIC0 $XSIO $SYMG $IORQ
                                           ENT= $IOUP $IODN $ETEQ $IRT
                                           ENT= $DEVT $CXC  $CYC  $GTIO
                                           ENT= $UP   $CVEQ $DMS  $BLLO
                                           ENT= $BLUP $BITB $UNLK $XXUP
                                           ENT= $DLAY $DMEQ $CKLO $CON1
                                           ENT= $CON2 $CON3 $DRVM $RSM
                                           ENT= $IOCL
                                           EXT= $RQST $CLCK $XEQ  $TYPE
                                           EXT= $LIST $ALC  $RTN  $LUSW
                                           EXT= $SCD3 $RNTB $CNV3 $ERMG
                                           EXT= $CNV1 $CLAS $REIO $ABRT
                                           EXT= $INER $ZZZZ $PDSK $UCON
                                           EXT= $UIN  $CIC  $PERR $ERAB
                                           EXT= $IDNO $SMAP $MATA $MRMP
                                           EXT= $MVBF $DVPT $DLTH $DVMP
                                           EXT= $SDA
                                           END=  00000

EXEC4     92067-16014 REV.1805 780310
                         00000 02357 00000 ENT= $ERMG $RQST $OTRL $USER
```

```
                                                  ENT= $DREQ $DREL $SDRL $SDSK
                                                  ENT= $ERRA $REIO $CREL $RSRE
                                                  ENT= $ABRE $PDSK $ABXY $PWR5
                                                  ENT= $MVBF $SGAF $LEND $DHED
                                                  ENT= $LBR  $LBX  $XEX
                                                  EXT= $CNV3 $SYMG $LIST $XEQ
                                                  EXT= $PVCN EXEC  $LIBR $LIBX
                                                  EXT= $IDLE $PVST $RENT $CVEQ
                                                  EXT= $ABRT $DMS  $TRRN $SCLK
                                                  EXT= $ALC  $RTN  $MATA $IDNO
                                                  EXT= $MRMP $PBUF $MNP  $MPFT
                                                  EXT= $PERR $CNV1 $IORQ $MPT1
                                                  EXT= $MPT2 $MPT3 $MPT4 $MPT5
                                                  EXT= $MPT6 $MPT7 $MPT9 $GTIO
                                                  EXT= $MPT8
                                                  END=  00000
FILE NAME: %CR4S2:: 32767:       5:     106:        0 IS ON LU      35

 $TRN4     92067-16014 REV.1805 780104
                               00000 00153 00000 ENT= $TRRN $CRN# $ULU
                                                  EXT= $RNTB $IDNO $SCD3 $SCLK
                                                  EXT= $ULLU $CGRN
                                                  END=  00000

 SCHD4     92067-16014 REV.1805 780317
                               00000 05160 00000 ENT= $ABRT $TYPE $PRSE $CNV1
                                                  ENT= $CNV3 $OP   $MPT1 $MPT2
                                                  ENT= $MPT3 $MPT4 $MPT5 $MPT6
                                                  ENT= $STRT $INER $MPT7 $ASTM
                                                  ENT= $WATR $SZIT $MPT8 $IDSM
                                                  ENT= $PBUF $MPT9 $RTST $CVWD
                                                  ENT= $STRG $MSEX $LSTM $LST
                                                  ENT= $SCD  $ID#  $MSG
                                                  EXT= $XSIO $IOUP $IODN $ERMG
                                                  EXT= $DREQ $DLP  $PLP  $MPFT
                                                  EXT= $MEU  $CMST $COML $SDA
                                                  EXT= $SDT2 $RLB  $RLN  $MPSA
                                                  EXT= $MPS2 $IDEX $IOCL $OTRL
                                                  EXT= $DREL $CHTO $LUPR $EQST
                                                  EXT= $MESS $LIST $IDNO $SCD3
                                                  EXT= $CNFG $ERAB $ZZZZ $TIME
                                                  EXT= $PVCN $MNP  $ERIN $NOPG
                                                  EXT= $OPER $ILST $NOLG $LGBS
                                                  EXT= $NMEM $XEQ  $TMRQ $ONTM
                                                  EXT= $ALC  $RTN  $WORK $BRED
                                                  EXT= $TIMR $ETTM $TIMV $TREM
                                                  EXT= $RNTB $CREL $SYMG $SDRL
                                                  EXT= $ALDM $DMAL $MATA $PRCN
                                                  EXT= $MBGP $MRTP $MCHN $MAXP
                                                  EXT= $BLLO $BLUP
                                                  END=  00047

 $ALC      92067-16014 REV.1805 741120
                               00000 00206 00000 ENT= $ALC  $RTN  $PNTR
                                                  EXT= $LIST $WORK
                                                  END=  00000

 OCMD4     92067-16014 REV.1805 771102
                               00000 01142 00000 ENT= $LUPR $EQST $CHTO
                                                  EXT= $CVEQ $CNV1 $CNV3 $UNLK
```

```
                                          EXT= $XXUP $DLAY $DMEQ $SCD3
                                          EXT= $ETEQ $CKLO $BITB $INER
                                          EXT= $XCQ  $MSEX
                                          END=  00000

PERR4    92067-16014 REV.1805 780227
                        00001 00741 00000 ENT= $PERR $PETB
                                          EXT= $CNV1 $CNV3 $SYMG $ERMG
                                          EXT= $XCQ  $UNPE $MAXP $MATA
                                          EXT= $DMS  $ABXY $CIC
                                          END=  00000

$CNFG    92067-16014 REV.1805 770112
                        00000 04637 00000 ENT= $CNFG $EXIT $PCHN $WRRD
                                          ENT= $USRS $ABDP $SMTB $TRTB
                                          ENT= $TREN $NPGQ $GDPG $SAVE
                                          EXT= $SBTB $XSIO $CMST $ENDS
                                          EXT= $MRMP $XCQ  $LIST $CNV3
                                          EXT= $PRSE $PLP  $MATA $MNP
                                          END=  00312

$STB1    92067-16014 REV.1805 780223
                        00000 00123 00000 ENT= $ERAB $PVCN EXEC  $LIBR
                                          ENT= $LIBX $PVST $UPIO $CIC
                                          ENT= $XCIC $YCIC $UIN  $UCON
                                          ENT= $XEQ  $XDMP $IDLE $SCD3
                                          ENT= $IDNO $MEU  $LIST $MESS
                                          ENT= $WORK $SOP  $ULLU $CGRN
                                          ENT= $MTM  $OPSY
                                          EXT= $ERRA $LBR  $LBX  $XEX
                                          EXT= $UP   $CIC0 $CXC  $CYC
                                          EXT= $CON1 $CON2 $CON3 $XCQ
                                          EXT= $XDM  $SCD  $ID#  $LST
                                          EXT= $MSG  $IDSM $OP   $ULU
                                          EXT= $CRN#
                                          END=  00122

$STB2    92067-16014 REV.1805 771107
                        00000 00045 00000 ENT= $MATA $MCHN $MBGP $MRTP
                                          ENT= $DLTH $DVPT $TIME $BATM
                                          ENT= $DLP  $PLP  $ENDS $MPFT
                                          ENT= $BGFR $RTFR $IDEX $MRMP
                                          ENT= $MPS2 $EMRP $MPSA $SDA
                                          ENT= $SDT2 $CMST $COML $CFR
                                          ENT= $MNP  $DVMP $RLB  $RLN
                                          ENT= $SBTB
                                          END=  00000
               TOTAL    00003 33056 00000
```

```
MODULE  LEVEL   MODULES WHERE USED
-------------------------------------------------------
$$TB1    100    $TRN4 RTIO4 PERR4 SCHD4 DISP4 EXEC4 RTIME $ALC  $CNFG
                OCMD4
$$TB2    100    RTIME DISP4 SCHD4 $CNFG RTIO4 EXEC4 PERR4
$ALC     100    RTIO4 EXEC4 SCHD4
$ASC4    100    SCHD4
$CNFG    100    SCHD4
$CSY4      1
$TRN4    100    $$TB1 DISP4 EXEC4
DISP4    100    SCHD4 PERR4 EXEC4 RTIO4 OCMD4 $CNFG $$TB1
EXEC4    100    DISP4 PERR4 SCHD4 RTIME RTIO4 $$TB1
OCMD4    100    SCHD4
PERR4    100    RTIO4 EXEC4
RTIME    100    RTIO4 SCHD4 EXEC4 $TRN4
RTIO4    100    OCMD4 SCHD4 $$TB1 EXEC4 RTIME PERR4 DISP4 $CNFG
SCHD4    100    DISP4 RTIO4 EXEC4 OCMD4 PERR4 $CNFG $$TB1 RTIME
```

ENTRY	DEFN-MOD	MODULES WHERE USED
$$OP	$STB1	
$ABDP	$CNFG	
$ABRE	EXEC4	DISP4
$ABRT	$CHD4	DISP4 RTIO4 EXEC4
$ABXY	EXEC4	PERR4
$ALC	$ALC	RTIO4 EXEC4 $CHD4
$ALDM	DISP4	SCHD4
$ASTM	$CHD4	
$BATM	$$TB2	RTIME
$BGFR	$$TB2	DISP4
$BITB	RTIO4	OCMD4
$BLLO	RTIO4	$CHD4
$BLUP	RTIO4	SCHD4
$BRED	DISP4	SCHD4
$CFR	$$TB2	DISP4
$CGRN	$$TB1	STRN4
$CHTO	OCMD4	SCHD4
$CIC	$$TB1	RTIO4 PERR4
$CIC0	RTIO4	$$TB1
$CKLO	RTIO4	OCMD4
$CLCK	RTIME	RTIO4
$CMST	$$TB2	DISP4 SCHD4 $CNFG
$CNFG	$CNFG	$CHD4
$CNV1	SCHD4	RTIO4 EXEC4 OCMD4 PERR4
$CNV3	SCHD4	RTIO4 EXEC4 OCMD4 PERR4 $CNFG
$COML	$$TB2	SCHD4
$CON1	RTIO4	$$TB1
$CON2	RTIO4	$$TB1
$CON3	RTIO4	$$TB1
$CREL	EXEC4	SCHD4
$CRN#	STRN4	$$TB1
$CVEQ	RTIO4	EXEC4 OCMD4
$CVWD	SCHD4	
$CXC	RTIO4	$$TB1
$CYC	RTIO4	$$TB1
$DEVT	RTIO4	RTIME
$DHED	EXEC4	
$DLAY	RTIO4	OCMD4
$DLP	$$TB2	SCHD4
$DLTH	$$TB2	RTIO4
$DMAL	DISP4	SCHD4
$DMEQ	RTIO4	OCMD4
$DMS	RTIO4	EXEC4 PERR4
$DREL	EXEC4	DISP4 SCHD4
$DREQ	EXEC4	DISP4 SCHD4
$DRVM	RTIO4	
$DVMP	$$TB2	RTIO4
$DVPT	$$TB2	DISP4 RTIO4
$EMRP	$$TB2	DISP4
$ENDS	$$TB2	$CNFG
$EQST	OCMD4	$CHD4
$ERAB	$$TB1	RTIO4 SCHD4
$ERIN	$ASC4	SCHD4
$ERMG	EXEC4	DISP4 RTIME RTIO4 SCHD4 PERR4
$ERRA	EXEC4	$$TB1
$ETEQ	RTIO4	OCMD4
$ETTM	RTIME	SCHD4
$EXIT	$CNFG	

$GDPG	$CNFG							
$GTIO	RTIO4	EXEC4						
$ID#	SCHD4	$$TB1						
$IDEX	$$TB2	DISP4	SCHD4					
$IDLE	$$TB1	DISP4	EXEC4					
$IDNO	$$TB1	RTIO4	EXEC4	$TRN4	SCHD4			
$IDSM	SCHD4	RTIME	$$TB1					
$ILST	$ASC4	SCHD4						
$INER	SCHD4	RTIME	RTIO4	OCMD4				
$IOCL	RTIO4	DISP4	SCHD4					
$IODN	RTIO4	SCHD4						
$IORQ	RTIO4	EXEC4						
$IOUP	RTIO4	SCHD4						
$IRT	RTIO4	DISP4						
$LBR	EXEC4	$$TB1						
$LBX	EXEC4	$$TB1						
$LEND	EXEC4							
$LGBS	$ASC4	SCHD4						
$LIBR	$$TB1	EXEC4						
$LIBX	$$TB1	EXEC4						
$LIST	$$TB1	DISP4	RTIME	RTIO4	EXEC4	SCHD4	$ALC	$CNFG
$LST	SCHD4	$$TB1						
$LSTM	SCHD4							
$LUPR	OCMD4	SCHD4						
$MATA	$$TB2	DISP4	RTIO4	EXEC4	SCHD4	PERR4	$CNFG	
$MAXP	DISP4	SCHD4	PERR4					
$MBGP	$$TB2	DISP4	SCHD4					
$MCHN	$$TB2	DISP4	SCHD4					
$MESS	$$TB1	SCHD4						
$MEU	$$TB1	SCHD4						
$MNP	$$TB2	DISP4	EXEC4	SCHD4	$CNFG			
$MPFT	$$TB2	DISP4	EXEC4	SCHD4				
$MPS2	$$TB2	SCHD4						
$MPSA	$$TB2	SCHD4						
$MPT1	SCHD4	EXEC4						
$MPT2	SCHD4	EXEC4						
$MPT3	SCHD4	EXEC4						
$MPT4	SCHD4	EXEC4						
$MPT5	SCHD4	EXEC4						
$MPT6	SCHD4	EXEC4						
$MPT7	SCHD4	EXEC4						
$MPT8	SCHD4	EXEC4						
$MPT9	SCHD4	EXEC4						
$MRMP	$$TB2	DISP4	RTIO4	EXEC4	$CNFG			
$MRTP	$$TB2	DISP4	SCHD4					
$MSEX	SCHD4	RTIME	OCMD4					
$MSG	SCHD4	$$TB1						
$MTM	$$TB1							
$MVBF	EXEC4	RTIO4						
$NMEM	$ASC4	SCHD4						
$NOLG	$ASC4	SCHD4						
$NOPG	$ASC4	SCHD4						
$NPGQ	$CNFG							
$ONTM	RTIME	SCHD4						
$OP	SCHD4	$$TB1						
$OPER	$ASC4	SCHD4						
$OPSY	$$TB1							
$OTRL	EXEC4	SCHD4						
$PBUF	SCHD4	EXEC4						
$PCHN	$CNFG							

$PDSK	EXEC4	RTIO4
$PERR	PERR4	RTIO4 EXEC4
$PETB	PERR4	
$PLP	$$TB2	SCHD4 SCNFG
$PNTR	$ALC	
$PRCN	DISP4	SCHD4
$PRSE	SCHD4	SCNFG
$PVCN	$$TB1	EXEC4 SCHD4
$PVST	$$TB1	EXEC4
$PWR5	EXEC4	
$REIO	EXEC4	RTIO4
$RENT	DISP4	EXEC4
$RLB	$$TB2	SCHD4
$RLN	$$TB2	SCHD4
$RQST	EXEC4	RTIO4
$RSM	RTIO4	
$RSRE	EXEC4	DISP4
$RTFR	$$TB2	DISP4
$RTN	$ALC	RTIO4 EXEC4 SCHD4
$RTST	SCHD4	DISP4
$SAVE	SCNFG	
$SBTB	$$TB2	SCNFG
$SCD	SCHD4	$$TB1
$SCD3	$$TB1	RTIO4 STRN4 SCHD4 OCMD4
$SCLK	RTIME	EXEC4 STRN4
$SDA	$$TB2	DISP4 RTIO4 SCHD4
$SDRL	EXEC4	SCHD4
$SDSK	EXEC4	
$SDT2	$$TB2	DISP4 SCHD4
$SGAF	EXEC4	DISP4
$SMAP	DISP4	RTIO4
$SMTB	SCNFG	
$STRG	SCHD4	
$STRT	SCHD4	
$SYMG	RTIO4	RTIME EXEC4 SCHD4 PERR4
$ZIT	SCHD4	DISP4
$TADD	RTIME	
$TIME	$$TB2	DISP4 RTIME SCHD4
$TIMR	RTIME	SCHD4
$TIMV	RTIME	SCHD4
$TMRQ	RTIME	SCHD4
$TREM	RTIME	SCHD4
$TREN	SCNFG	
$TRRN	STRN4	DISP4 EXEC4
$TRTB	SCNFG	
$TYPE	SCHD4	RTIO4
$UCON	$$TB1	RTIO4
$UIN	$$TB1	RTIO4
$ULLU	$$TB1	STRN4
$ULU	STRN4	$$TB1
$UNLK	RTIO4	OCMD4
$UNPE	DISP4	PERR4
$UP	RTIO4	$$TB1
$UPIO	$$TB1	
$USER	EXEC4	
$USRS	SCNFG	
$WATR	SCHD4	DISP4
$WORK	$$TB1	DISP4 RTIME SCHD4 $ALC
$WRRD	SCNFG	
$XCIC	$$TB1	

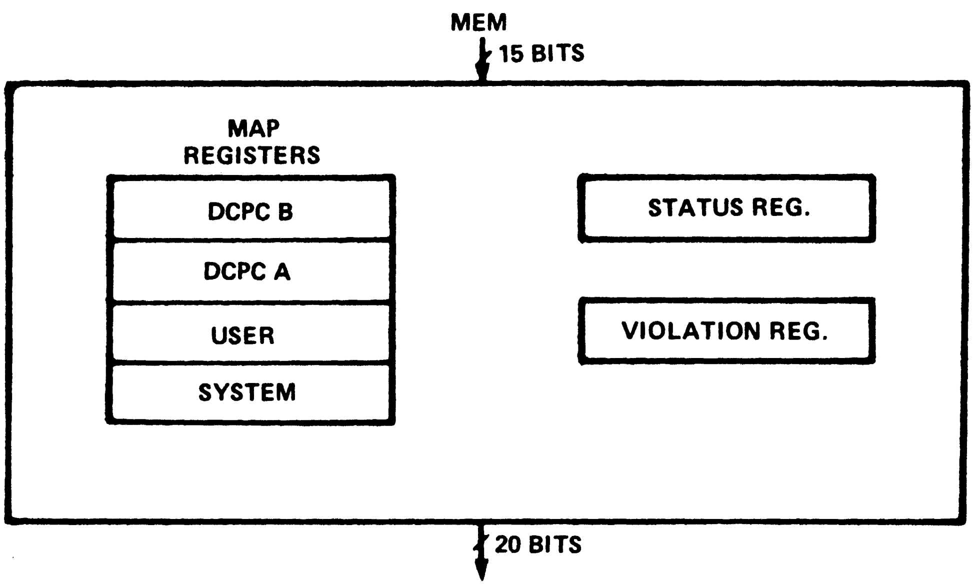
$XCQ	DISP4	OCMD4	PERR4	$CNFG	$STB1
$XDM	DISP4	$STB1			
$XDMP	$STB1	DISP4			
$XEQ	$STB1	RTIME	RTIO4	EXEC4	SCHD4
$XEX	EXEC4	$STB1			
$XSIO	RTIO4	DISP4	SCHD4	$CNFG	
$XXUP	RTIO4	OCMD4			
$YCIC	$STB1				
$ZZZZ	DISP4	RTIO4	SCHD4		
EXEC	$STB1	EXEC4			

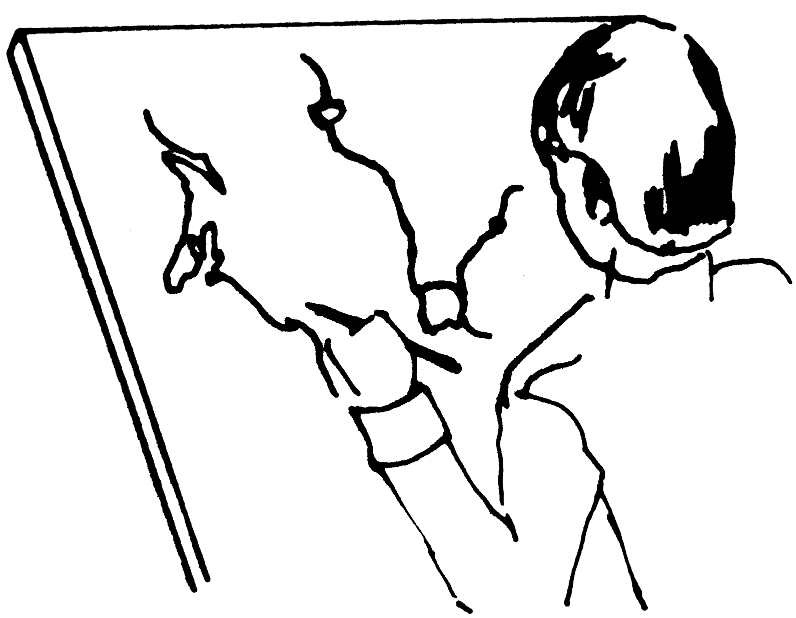
```
UNRESOLVED EXT  MODULES WHERE USED
-------------------------------------------------------------
 $CLAS           RTIO4
 $LUSW           RTIO4
 $RNTB           RTIO4 $TRN4 SCHD4
END OF CROSS REF
```

DYNAMIC MAPPING

DYNAMIC MAPPING SYSTEM (DMS)

- DMS CONSISTS OF:
  1) MEMORY PROTECT (MP)
  2) DMS INSTRUCTIONS (ROM)
  3) MEMORY EXPANSION MODULE (MEM)





15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	INSTR.				IZ/C	← ADDRESS →									
	L	D	A						5	7	3_8				

– MEMORY ADDRESSING REQUIRES 15 BITS!

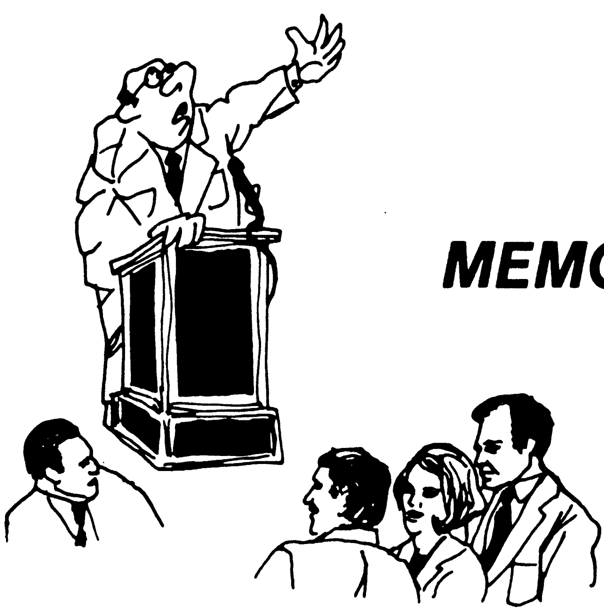
– THE INSTRUCTION PROVIDES 10

→ WHERE DO THE OTHER 5 COME FROM?

a) IF BIT 15 IS SET (INDIRECT ADDRESSING), THEY'RE TAKEN FROM THE FINAL ADDRESS. OTHERWISE,

b) IF BIT 10 IS CLEAR, THEY'RE SET = 0

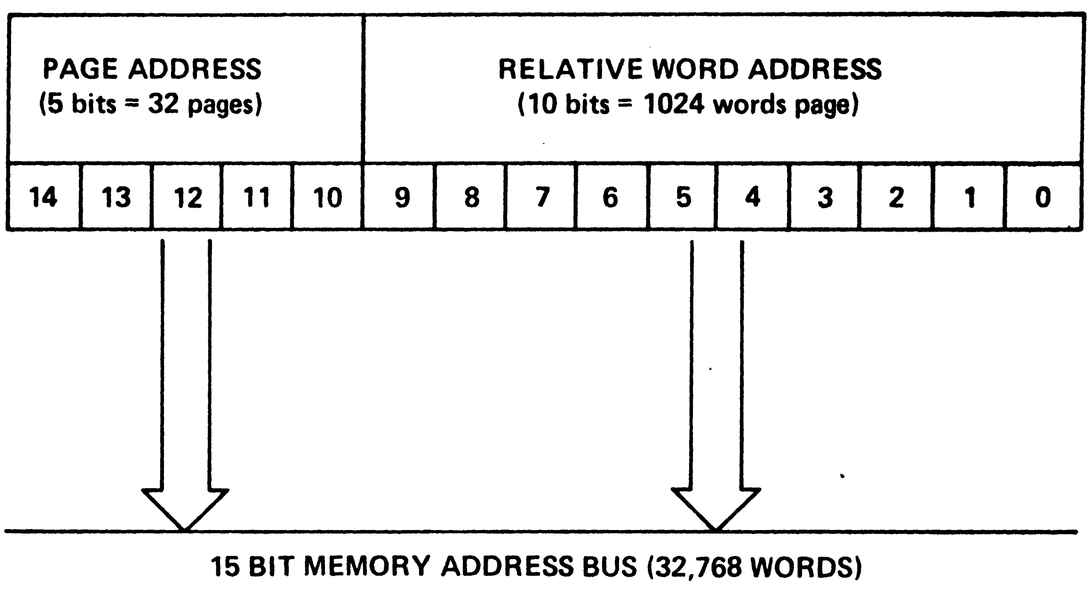
c) IF BIT 10 IS SET, THEY'RE SET = TO THE UPPER 5 BITS SPECIFYING THE ADDRESS OF THE INSTRUCTION ("P" REGISTER)



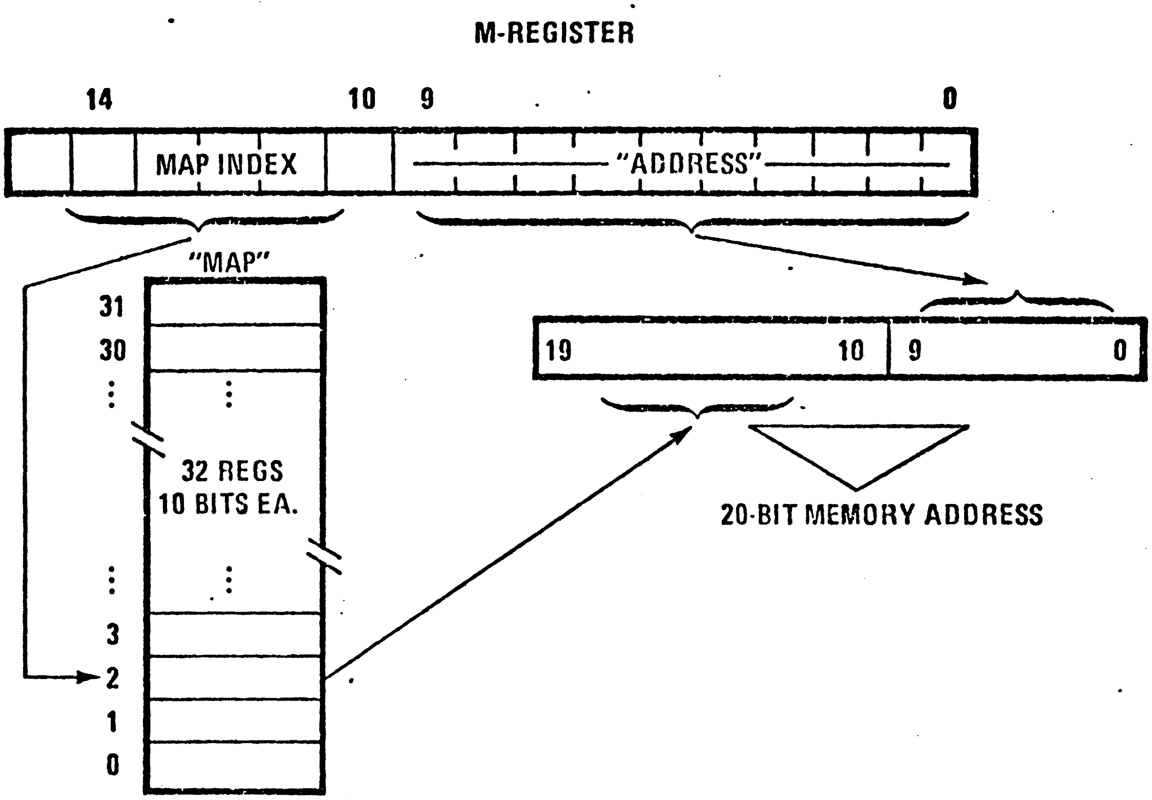
MEMORY ADDRESSING

* BASIC 21MX
ADDRESSES 32K WORDS

M-REGISTER (15 BITS)



21MX WITH DMS ADDRESSES UP TO 1 MEGA WORD



1024 WORDS/PAGE

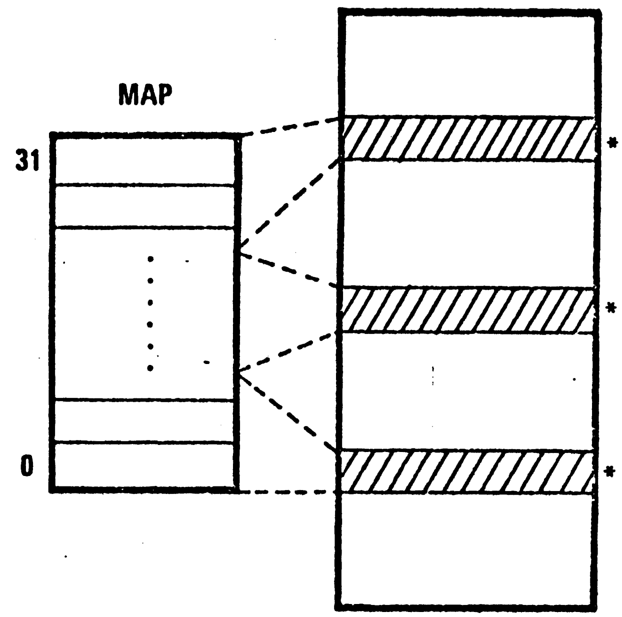
32 PAGES/MAP

PHYSICAL MEMORY

NOTE:

THE 5-BIT "MAP INDEX" CAN ONLY SPECIFY UP TO 32 REGISTERS. THUS, WITH A GIVEN MAP WE CAN STILL ONLY ADDRESS (ACCESS) 32 PAGES OF MEMORY.

THIS 32K SUB-SET IS OUR "LOGICAL ADDRESS SPACE" OR "LOGICAL MEMORY"



*32 PAGES TOTAL

21 MX DYNAMIC MAPPING

a) TRANSPARENT TO USER INSTRUCTIONS

b) A HARDWARE/FIRMWARE SCHEME

c) 12-BIT MAP REGISTERS ALLOW READ/WRITE PROTECTION

d) STATUS AND VIOLATION REGISTERS

e) USES 4 INDEPENDENT MAPS

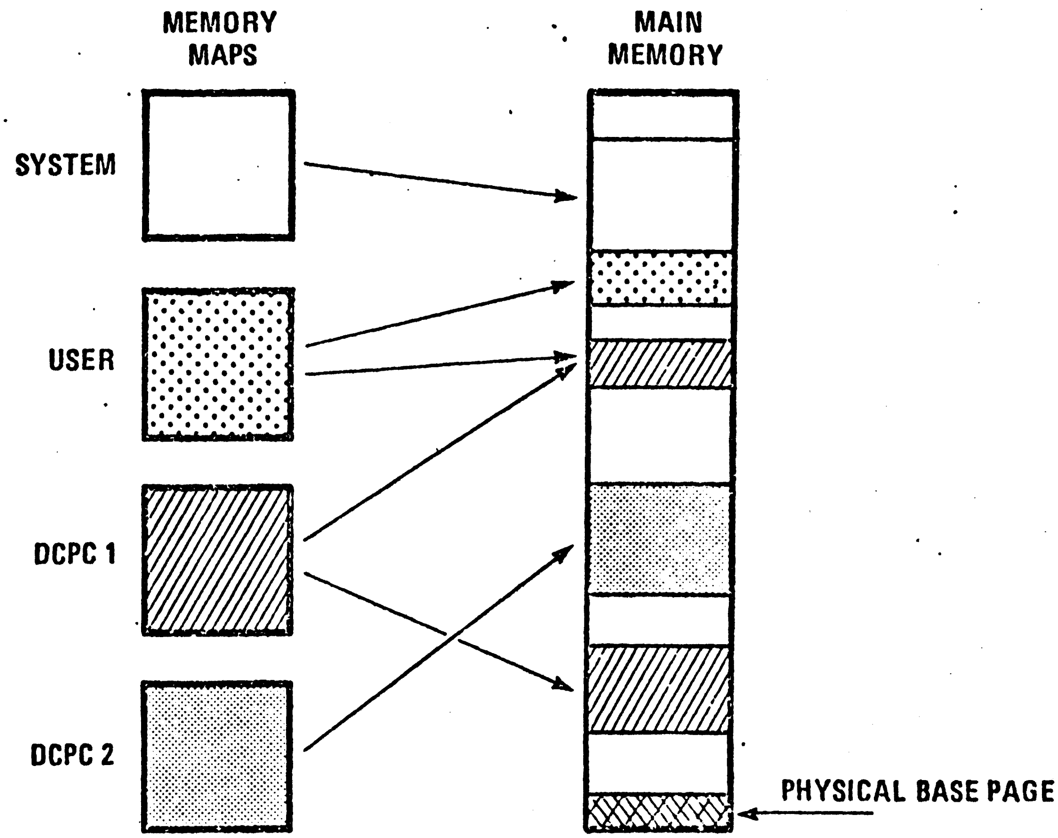
f) MAP CONTENTS ARE PROGRAMMABLE (SYA, USA, etc.)

g) MAPPING CAN BE ENABLED/DISABLES

h) SPECIAL INSTRUCTIONS ALLOW "CROSS-MAP-MOVES" BETWEEN SYSTEM AND USER MAPS (XLA, XSA, etc.)

i) PROGRAMMABLE BASE PAGE FENCE (LFA, etc.)

MAPS DYNAMICALLY CONFIGURE MEMORY

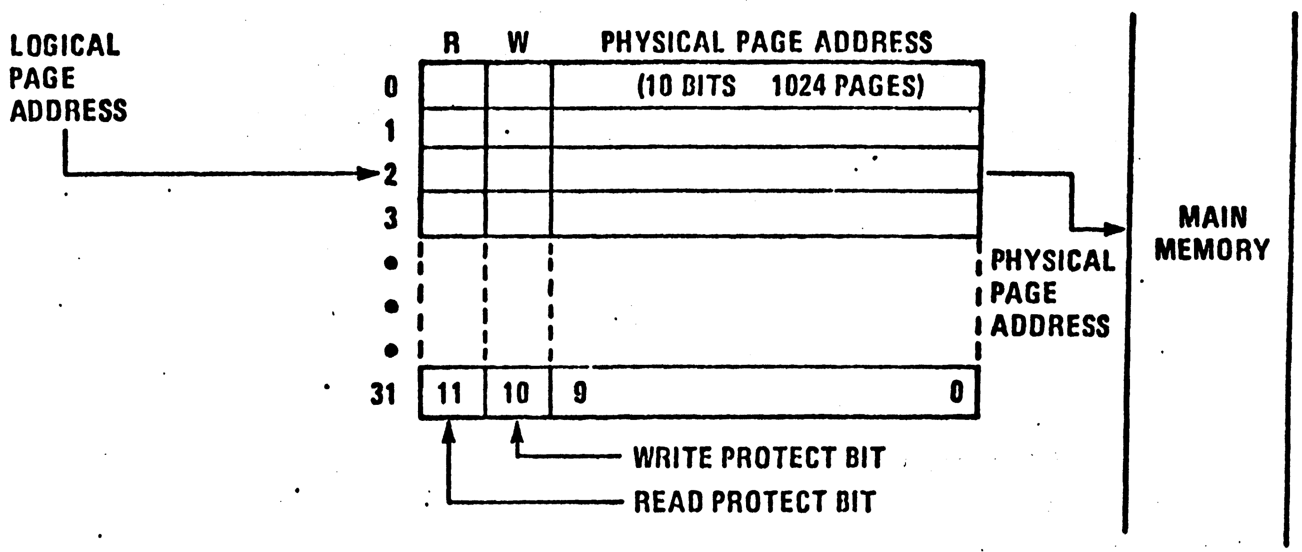


ALL MAPS INCLUDE PART OF THE PHYSICAL BASE PAGE

MAP SEGMENTATION

Octal	Map	Decimal
171_8	PORT B MAP (32 REGISTERS)	127_{10}
140		96
137	PORT A MAP (32 REGISTERS)	95
100		64
77	USER MAP (32 REGISTERS)	63
40		32
37	SYSTEM MAP (32 REGISTERS)	31
0		0

MAP REGISTER CONTENTS:



UNUSED MAP REGISTERS HAVE BOTH "R" AND "W" BITS SET

DMS (MEM) REGISTERS

MEM Status Register Format

BIT	SIGNIFICANCE
15	0 = MEM disabled at last interrupt 1 = MEM enabled at last interrupt
14	0 = System map selected at last interrupt 1 = User map selected at last interrupt
13	0 = MEM disabled currently 1 = MEM enabled currently
12	0 = System map selected currently 1 = User map selected currently
11	0 = Protected mode disabled currently 1 = Protected mode enabled currently
10	Portion mapped*
9	Base page fence bit 9
8	Base page fence bit 8
7	Base page fence bit 7
6	Base page fence bit 6
5	Base page fence bit 5
4	Base page fence bit 4
3	Base page fence bit 3
2	Base page fence bit 2
1	Base page fence bit 1
0	Base page fence bit 0

*Bit 10	Mapped Address (M)
0	Fence $\leq$ M $<$ 2000_8
1	1 $<$ M $<$ Fence

Note: The base page fence separates the reserved (mapped) memory from the shared (unmapped) memory. Bit 10 specifies which area is reserved (mapped).

MEM Violation Register Format

BIT	SIGNIFICANCE
15	Read violation*
14	Write violation*
13	Base page violation*
12	Privileged instruction violation*
11	Reserved
10	Reserved
9	Reserved
8	Reserved
7	0 = ME bus disabled at violation 1 = ME bus enabled at violation
6	0 = MEM disabled at violation 1 = MEM enabled at violation
5	0 = System map enabled at violation 1 = User map enabled at violation
4	Map address bit 4
3	Map address bit 3
2	Map address bit 2
1	Map address bit 1
0	Map address bit 0

*Significant when associated bit is set.

If MP is enabled any of these 4 violations will cause an interrupt of SC 5.

These registers are read with RSA/B or RVA/B instructions.

DMS ERRORS

DMS errors generate an interrupt on SC5 along with MP errors and memory parity errors. DMS errors are caused by:

- Reading a read protected page.
- Writing into a write protected page
- Base page fence violations
- Attempts to alter the DMS registers while memory protect is enabled.

DMS errors only occur when MEM and memory protect are both enabled.

RTE-IV
AND
DYNAMIC MAPPING

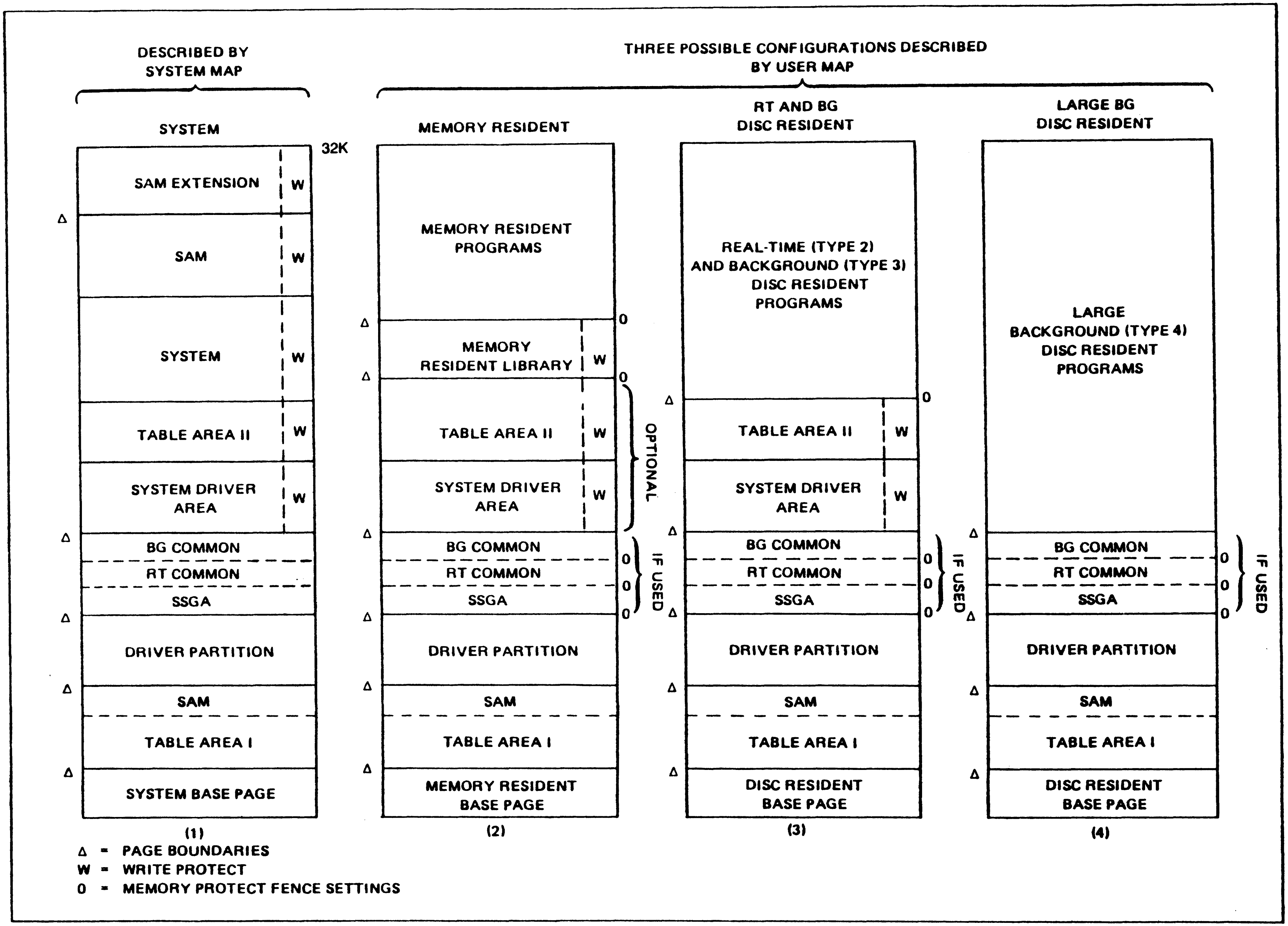
USES:

- Allow many partitions in systems with >32K memory.
- Increase memory space for user programs by removing most of RTE, device drivers, SAM, memory resident programs, memory resident librarys and optionally common from the user's logical address space (or user map).
- Provide additional space for each program's base page links by using the base page fence.

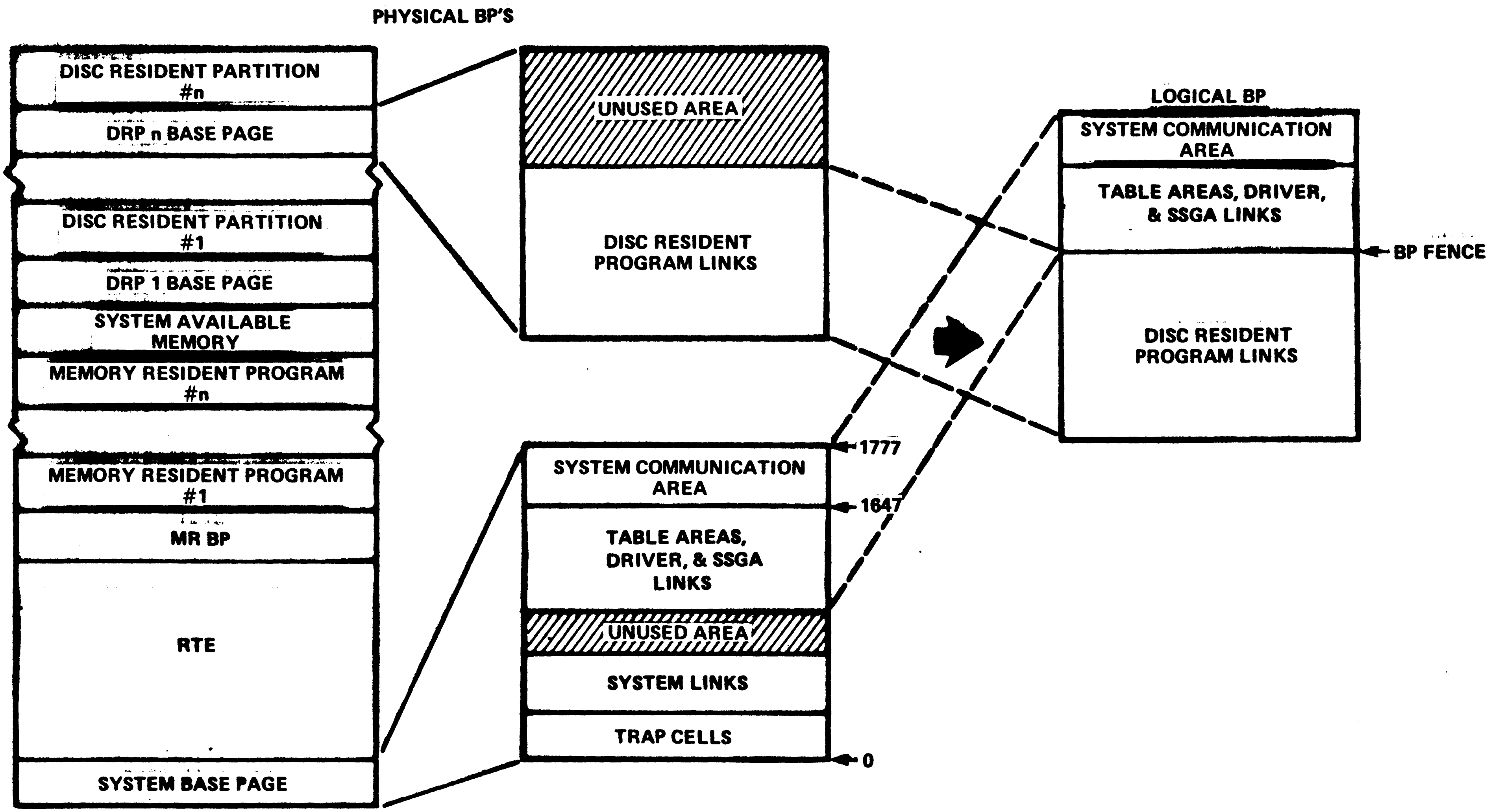
IMPLEMENTATION:

- System map is fixed after boot up.
- The user map is built when a program is dispatched for the first time or dispatched for the first time after a swap-in. For a context switch, RTE saves and restores the user map in the unused portion of the partition base page. A copy of the memory resident map is kept in the system ($MRMP).
- Separate map (a copy of the system or user map) for each DCPC channel.
- The base page fence register allows each user's logical base page to include part of the system base page.
- System map is automatically enabled upon all interrupts.
- DCPC maps are enabled/disabled on a word by word basis during DCPC processing.

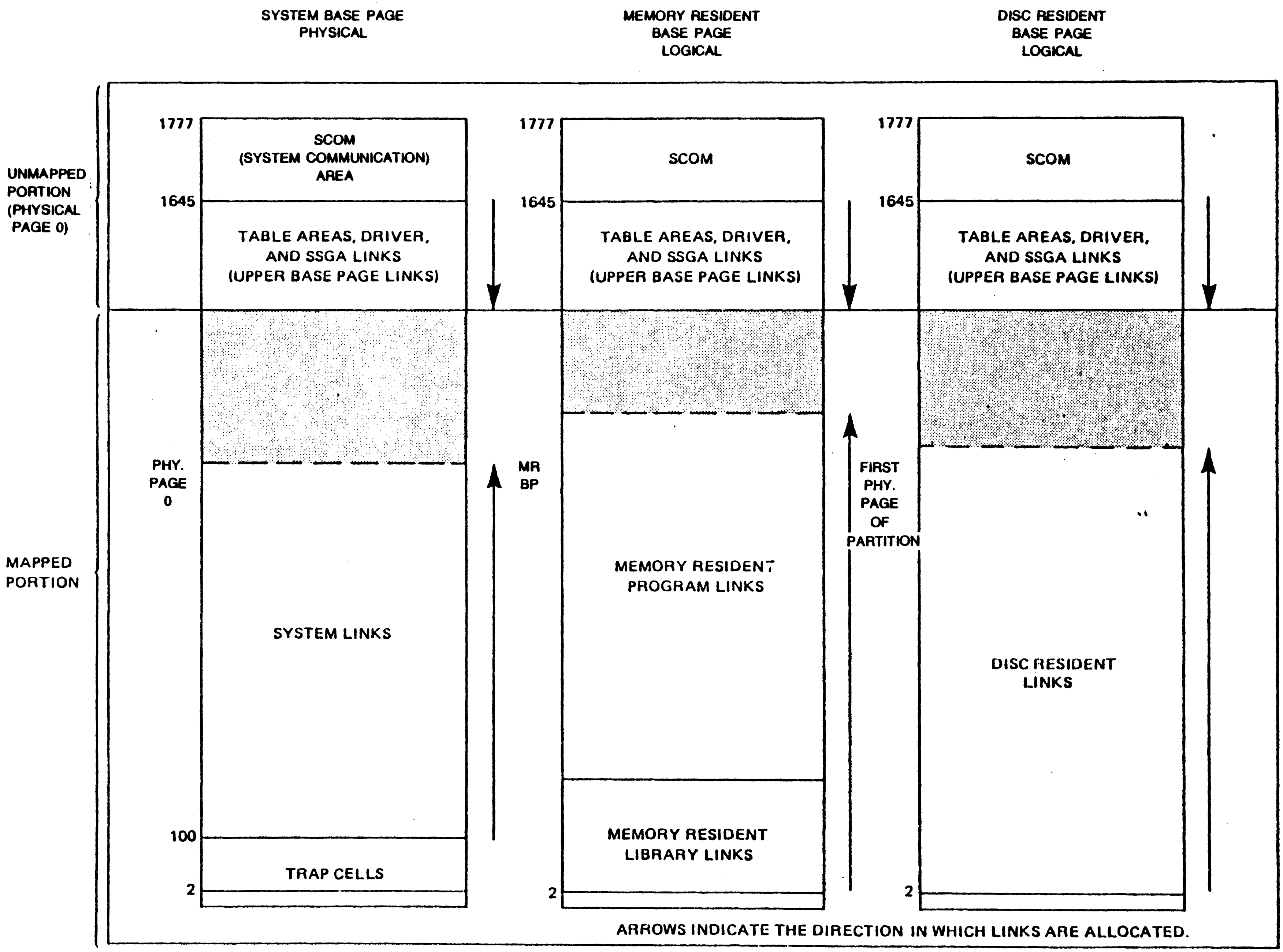
RTE-IV LOGICAL MEMORY MAPS



USER LOGICAL BASE PAGE



BASE PAGES


SYSTEM BASE PAGE PHYSICAL
MEMORY RESIDENT BASE PAGE LOGICAL
DISC RESIDENT BASE PAGE LOGICAL
UNMAPPED PORTION (PHYSICAL PAGE 0)
MAPPED PORTION
1777
1645
100
2
SCOM (SYSTEM COMMUNICATION) AREA
TABLE AREAS, DRIVER, AND SSGA LINKS (UPPER BASE PAGE LINKS)
PHY. PAGE 0
SYSTEM LINKS
TRAP CELLS
MR BP
SCOM
MEMORY RESIDENT PROGRAM LINKS
MEMORY RESIDENT LIBRARY LINKS
FIRST PHY. PAGE OF PARTITION
DISC RESIDENT LINKS
ARROWS INDICATE THE DIRECTION IN WHICH LINKS ARE ALLOCATED.

- Set M.P. fence from MPFT index in word 21 of I.D. seg.
- Set up appropriate user map: ($SMAP)

  1) Load base page register from MAT entry, word #3 (start page)

  2) Load system registers to map Table Area I and II, SDA, Driver Partition, and common based upon the program type (MPFI). The starting physical page is #1 for these areas and the ending page depends on the variables:

     $CMST - starting logical page number of common (org. 0)
     $SDA - starting logical page number of SDA (org. 0)
     $SDT2 - # of pages in SDA and Table Area II

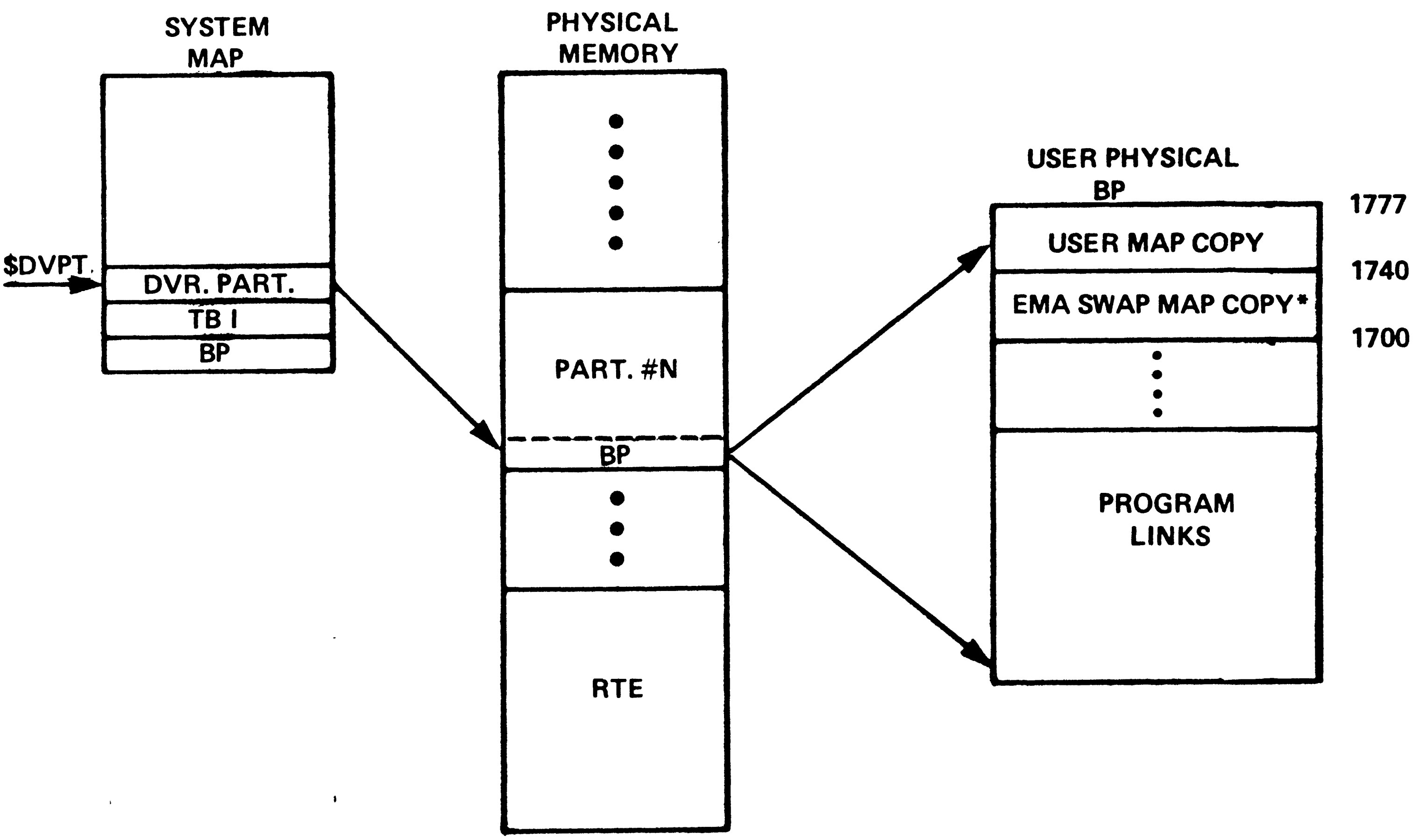
  3) At the next register map in the remainder of the partition by incrementing and loading until register number specified by ID word 21 is reached.

  4) Set remaining registers read/write protected.

- User map gets copied directly into DCPC maps when DCPC is used (swapping, DMA I/O)
- At boot-up $STRT sets up the system map. $ZZZZ initializes the remaining maps and sets the BP fence address.

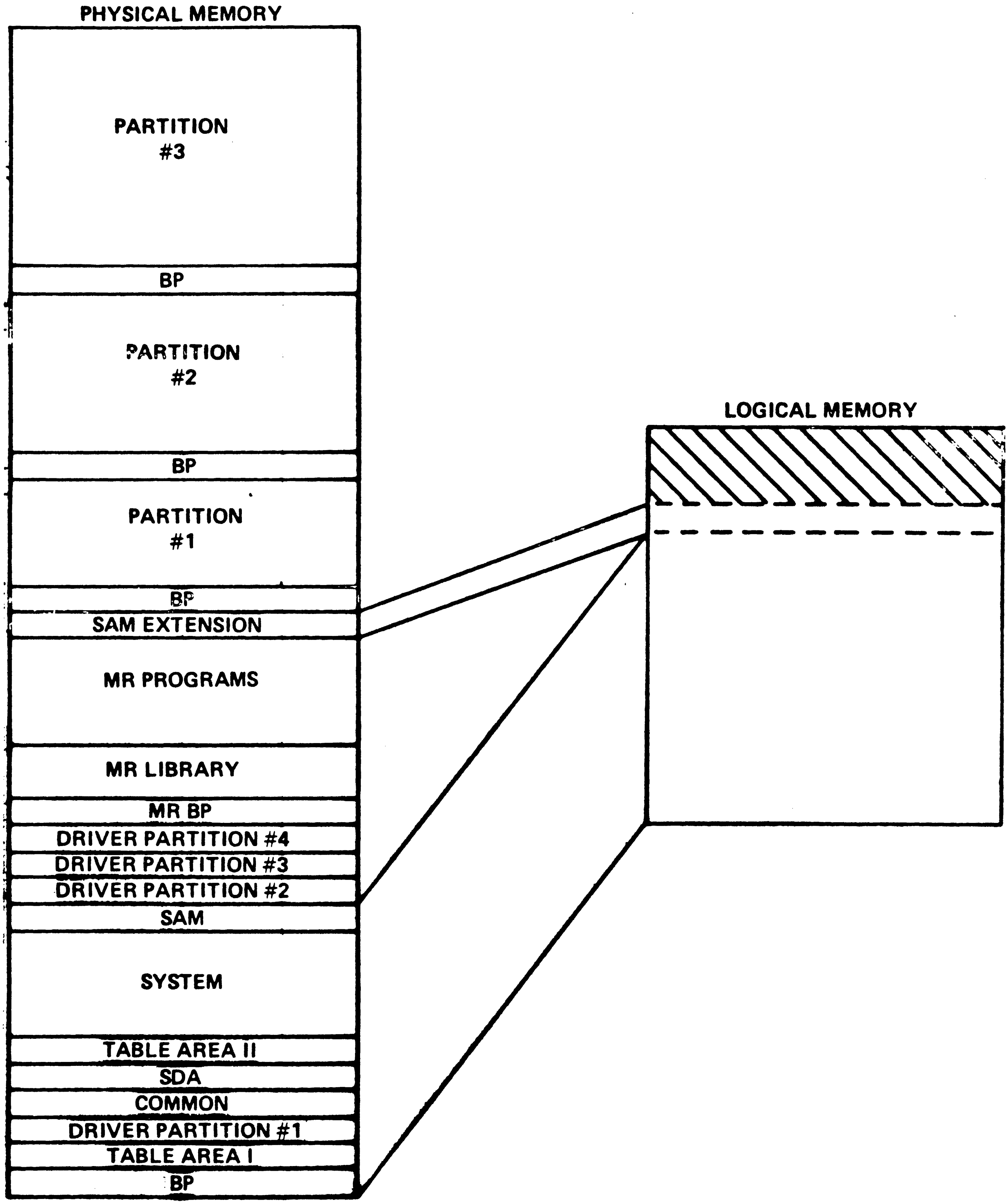
USER MAP SAVE AREA

DISP4 & RTIO4 USE THE UPPER PORTION OF THE PHYSICAL BASE PAGE OF EACH PARTITION TO SAVE AND RESTORE EACH USER'S MAP. THE DRIVER PARTITION MAP REGISTER ($DVPT) IS USED TO ACCESS THE USER'S PHYSICAL BP.

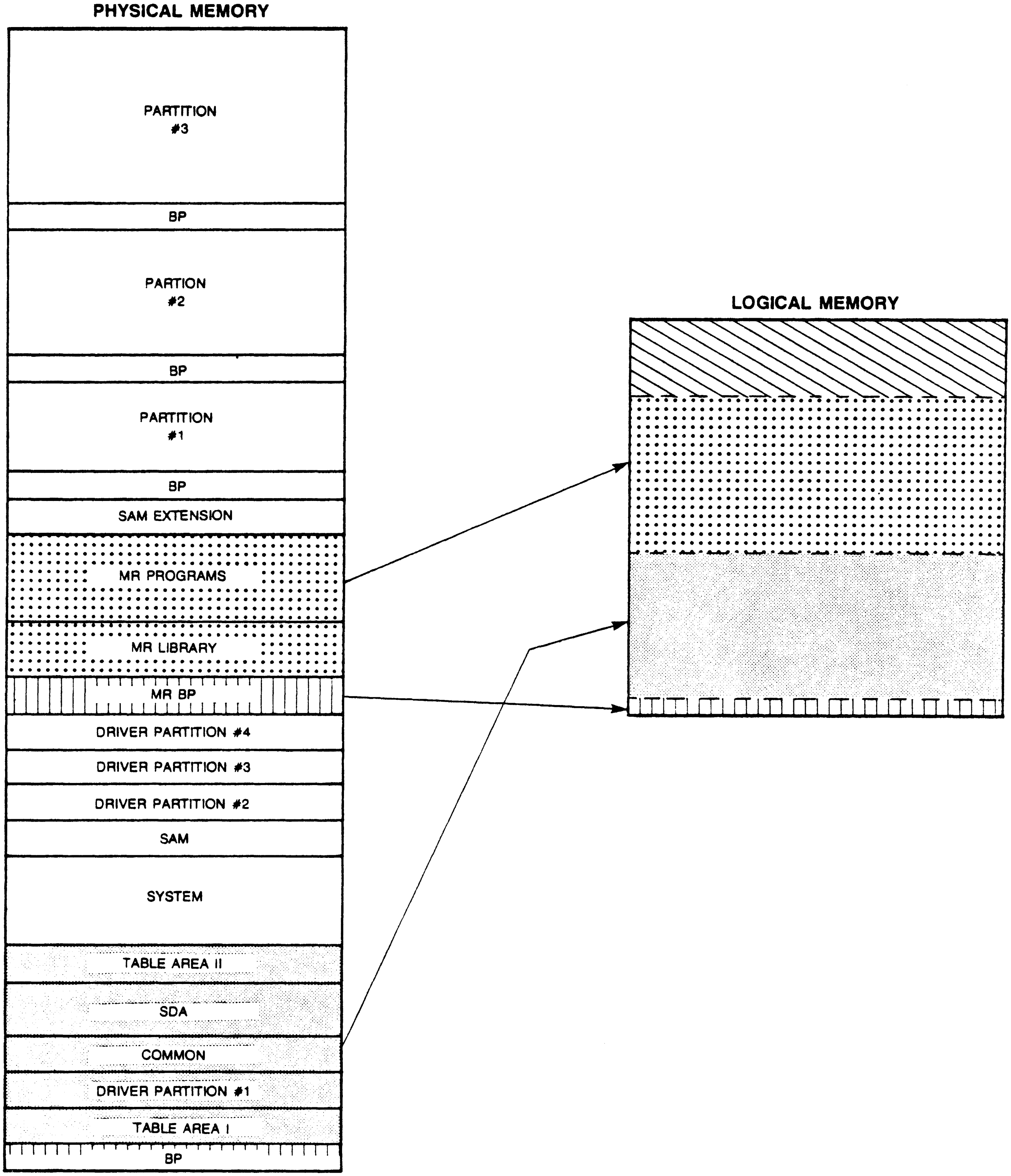


* USED ONLY DURING THE SWAPPING OF THE EMA PORTION OF A PROGRAM. THIS PREVENTS THE DESTRUCTION OF THE ORIGINAL COPY OF THE USER MAP. AFTER THE PROGRAM HAS BEEN SWAPPED, THE USER MAP IS MODIFIED TO SWAP EMA CHUNKS EQUAL IN SIZE TO THE LOGICAL ADDRESS SPACE OF TYPE 4 PROGRAMS (27K MAX.).

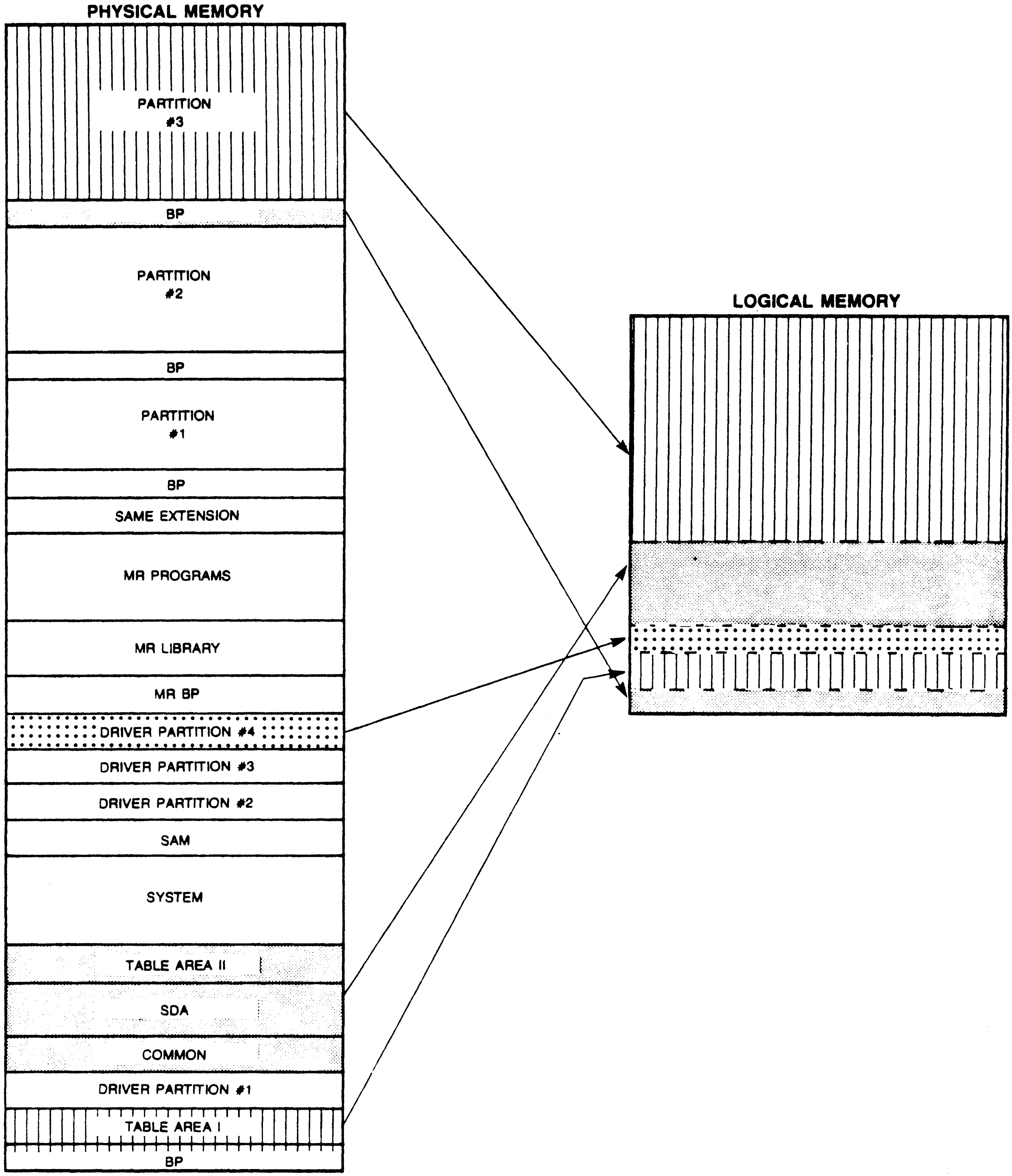
SAMPLE SYSTEM MAP



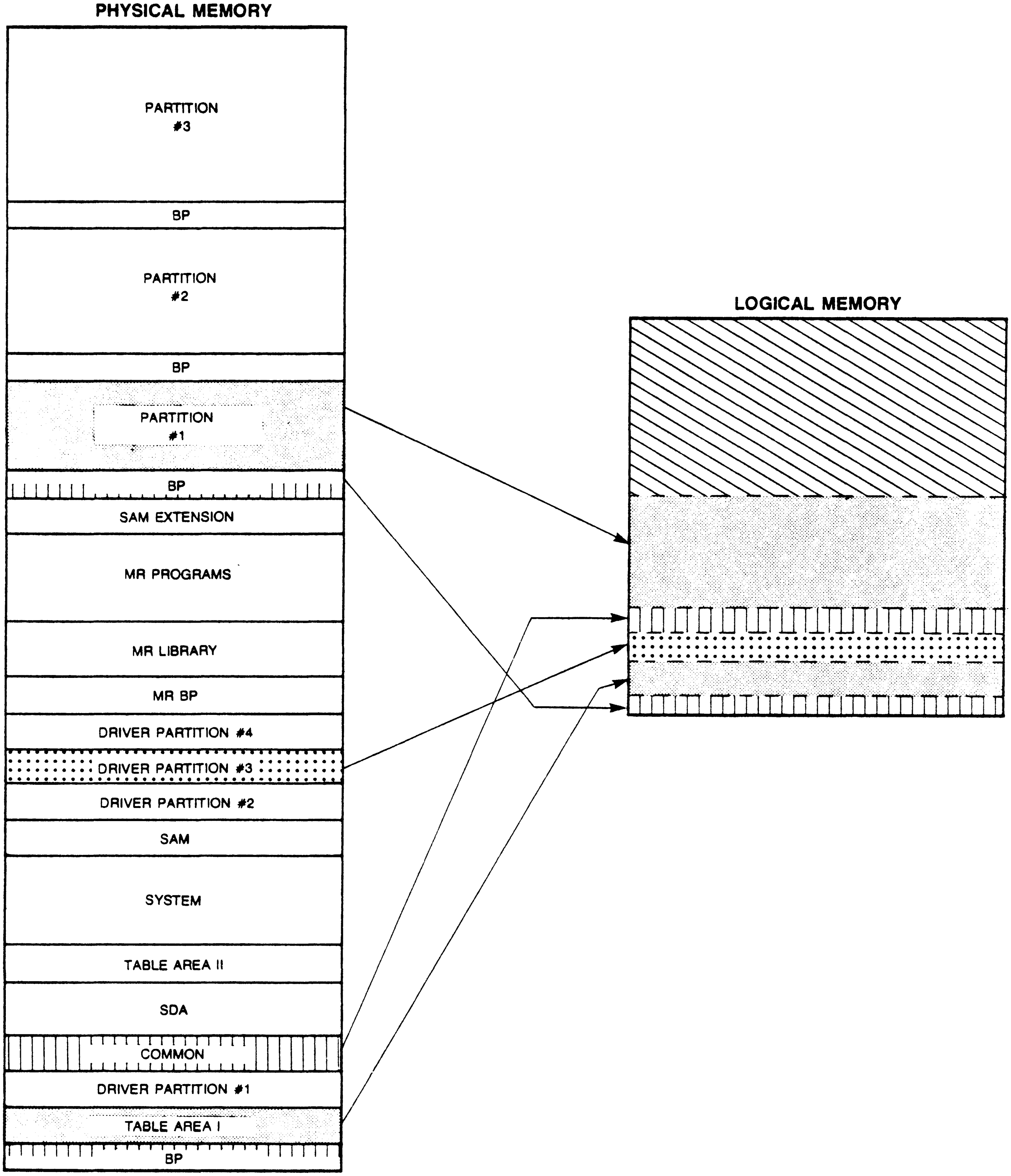
SAMPLE USER MAP
MEMORY RESIDENT PROGRAM WITH COMMON



SAMPLE USER MAP
DISC RESIDENT PROGRAM WITH OR WITHOUT COMMON
(TYPE 2 OR 3)



SAMPLE USER MAP
LARGE BG DISC RESIDENT PROGRAM WITH COMMON
(TYPE 4)



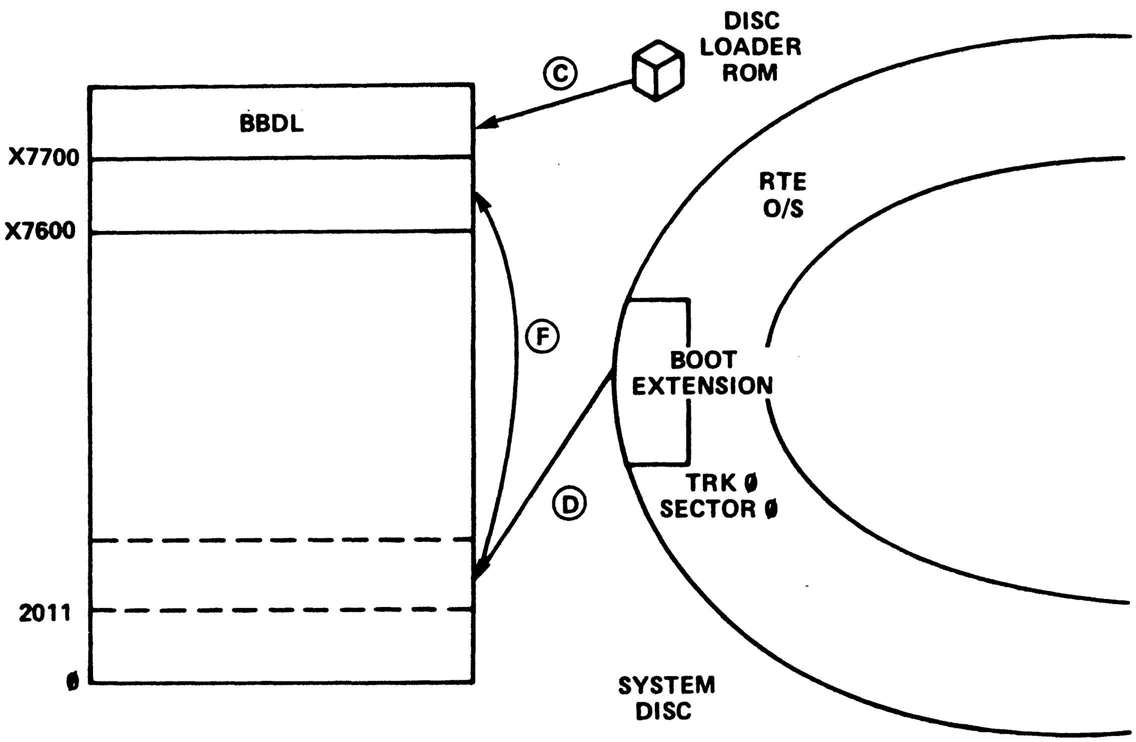
```
LIST OF DYNAMIC MAPPING REGISTERS FOR SYSTEMMAP

000000  000001  000057  000003  000004  040005  040006  040007
040010  040011  040012  040013  040014  040015  040016  040017
040020  040021  040022  040023  040024  040025  040026  040027
040054  040031  040032  040033  040034  040035  040036  040037

LIST OF DYNAMIC MAPPING REGISTERS FOR USER  MAP

000057  000001  000002  000003  000004  040005  040006  040007
040010  040011  040012  000060  000061  000062  140000  140001
140002  140003  140004  140005  140006  140007  140010  140011
140012  140013  140014  140015  140016  140017  140020  140021

LIST OF DYNAMIC MAPPING REGISTERS FOR DCPC1 MAP

000057  000001  000002  000003  000004  040005  040006  040007
040010  040011  040012  000060  000061  000062  140000  140001
140002  140003  140004  140005  140006  140007  140010  140011
140012  140013  140014  140015  140016  140017  140020  140021

LIST OF DYNAMIC MAPPING REGISTERS FOR DCPC2 MAP

000000  000001  000002  000003  000004  040005  040006  040007
040010  040011  040012  040013  040014  040015  040016  040017
040020  040021  040022  040023  040024  040025  040026  040027
040054  040031  040032  040033  040034  040035  040036  040037

STAT=  173446     VIOL=  000140
```

USER PROGRAM TYPE = 3
$CMST = 4

$SDA = 5
$STD2 = 6

RTE
BOOT PROCESS

RTE BOOT PROCESS



A. SET THE S REGISTER.

B. PRESS PRESET TO DISABLE THE INTERRUPT SYSTEM.

C. PRESS IBL TO CAUSE THE LOADER MICROPROGRAM TO READ THE DISC LOADER ROM (BBDL) INTO THE END OF UNMAPPED MEMORY.

D. PRESS RUN AND BBDL WILL READ THE BOOT EXTENSION FROM DISC INTO MEMORY AND TRANSFER EXECUTION TO THE BOOT EXTENSION AT 2055 (OCTAL).

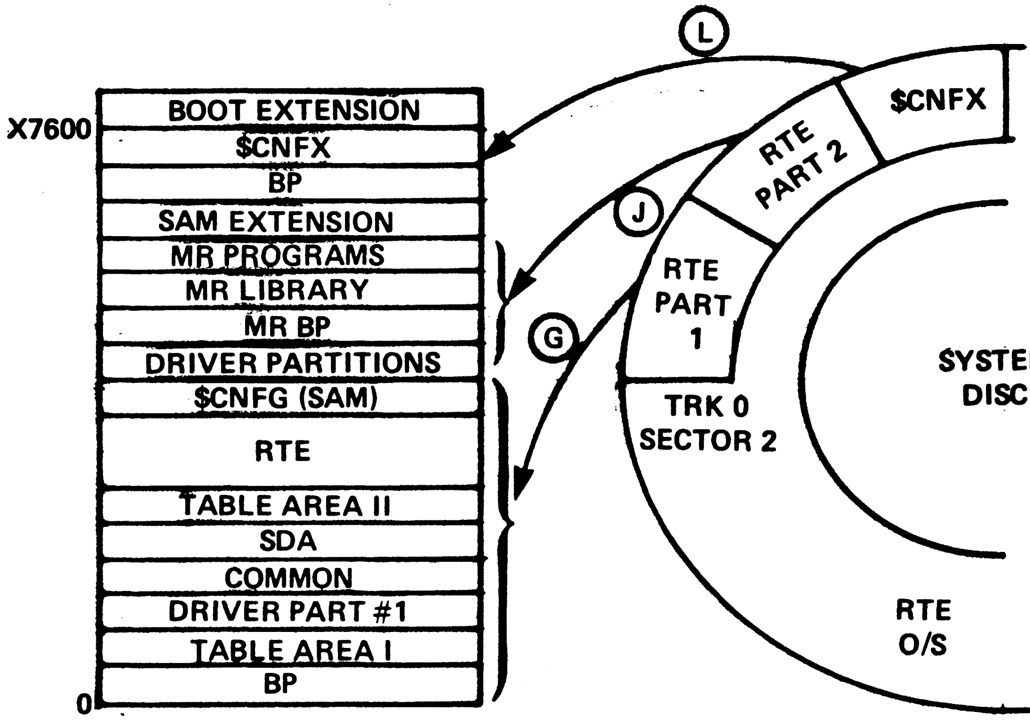
E. BOOT EXTENSION OPTIONALLY HALTS WITH A 102077 IF S REGISTER BIT 5 IS SET REQUESTING RECONFIGURATION. ENTER SELECT CODES OF SYSTEM CONSOLE AND DISC. PRESS RUN TO CONTINUE.

F. BOOT EXTENSION MOVES ITSELF TO END OF UNMAPPED MEMORY.

```
                 7905/20 DISC BOOT LOADER (12992B) - RPL COMPATIBLE

                         ASMB,A,B,L
      07700                    ORG 7700B
*******************************************************
*                                                *
*      REVISION          05 AUG 77'              *
*      PART NUMBER        12992-80002            *
*      PRODUCT NUMBER     12992B                 *
*                                                *
*******************************************************
*
*  SWITCH REGISTER USAGE
*
*   15-14     LOADER SELECT
*   13        UNUSED
*   12        =0/1=RPL/MANUAL BOOT
*   11-6      DISC SELECT CODE
*    5-3      RESERVED
*    2-0      SUBCHANNEL NUMBER
*
      00010          DC     EQU 10B
*
      07700 017727   START JSB STAT       GET STATUS
      07701 002021         SSA,RSS        IS DRIVE READY ?
      07702 027742         JMP DMA          YES, SET UP DMA
      07703 013714         AND B20          NO, CHECK STATUS BITS
      07704 002002         SZA            IS DRIVE FAULTY OR HARD DOWN ?
      07705 102030         HLT 30B          YES, HALT 30B, "RUN" TO TRY AGAIN
      07706 027700         JMP START        NO, TRY AGAIN FOR DISC READY
*
*  CONSTANTS
*
      07707 102011   ADDR1 OCT 102011   BOOT EXTENSION LOAD ADDRESS
      07710 102055   ADDR2 OCT 102055
      07711 164000   CNT   DEC -6144
      07712 000007   D7    OCT 7            START ADDRESS
      07713 001400   STCMD OCT 1400
      07714 000020   B20   OCT 20
      07715 017400   STMSK OCT 17400
*   9 NOP'S
                           LST
*
      07727 000000   STAT  NOP            STATUS CHECK SUBROUTINE
      07730 107710         CLC DC,C       SET STATUS COMMAND MODE
      07731 063713         LDA STCMD      GET STATUS COMMAND
      07732 102610         OTA DC         OUTPUT STATUS COMMAND
      07733 102310         SFS DC         WAIT FOR STATUS#1 WORD
      07734 027733         JMP *-1
      07735 107510         LIB DC,C         B-REG = STATUS#1 WORD
      07736 102310         SFS DC         WAIT FOR STATUS#2 WORD
      07737 027736         JMP *-1
      07740 103510         LIA DC,C         A-REG = STATUS#2 WORD
```

```
07741 127727         JMP STAT,I     RETURN
*
* SET UP DMA CHANNEL
*
07742 067776  DMA    LDB DMACW      GET DMA CONTROL WORD
07743 106606         OTB 6          OUTPUT DMA CONTROL WORD
07744 067707         LDB ADDR1      GET MEMORY ADDRESS
07745 106702         CLC 2          SET MEMORY ADDRESS INPUT MODE
07746 106602         OTB 2          OUTPUT MEMORY ADDRESS TO DMA
07747 102702         STC 2          SET WORD COUNT INPUT MODE
07750 067711         LDB CNT        GET WORD COUNT      ONE TRACK
07751 106602         OTB 2          OUTPUT WORD COUNT TO DMA
*FALL THRU
*  7905/20 COLD LOAD COMMAND
*
07752 106710  CLDLD  CLC DC         SET COMMAND INPUT MODE
07753 102501         LIA 1          LOAD SWITCH
07754 106501         LIB 1          REGISTER SETTINGS
07755 013712         AND D7         ISOLATE HEAD NUMBER
07756 005750         BLF,CLE,SLB    BIT 12=0?
07757 027762         JMP *+3        NO,MANUAL BOOT
07760 002002         SZA            YES,RPL BOOT. HEAD#=0?
07761 001000         ALS            NO,HEAD#=1, MAKE HEAD#=2
07762 001720         ALF,ALS        FORM COLD LOAD
07763 001000         ALS            COMMAND WORD
07764 103706         STC 6,C        ACTIVATE DMA
07765 103610         OTA DC,C       OUTPUT COLD LOAD COMMAND
07766 102310         SFS DC         IS COLD LOAD COMPLETED ?
07767 027766         JMP *-1          NO, WAIT
07770 017727         JSB STAT         YES, GET STATUS
07771 060001         LDA 1
07772 013715         AND STMSK      A-REG = STATUS BITS OF STATUS#1 WORD
07773 002002         SZA            IS TRANSFER OK ?
07774 027700         JMP START        NO,TRY AGAIN
07775 117710  EXIT   JSB ADDR2,I      YES, EXECUTE LOADED PROGRAM @ 2055B
*FALL THRU
*  THE NEXT 2 WORDS MUST BE THE LAST 2 WORDS
*   IN THE BOOTSTRAP LOADER IN THE LAST 2 MEMORY LOCATIONS
07776 000010  DMACW  ABS DC
07777 170100         ABS -START
                     END
```

BOOT PROCESS CONTINUED



G. BOOT EXTENSION LOADS RTE FROM BASE PAGE THRU $CNFG.

H. CONTROL IS TRANSFERRED TO RTE AT $STRT WITH A JMP 3,I

I. $STRT SETS UP AND ENABLES THE SYSTEM MAP

J. $CNFG LOADS DRIVER PARTITIONS, MEMORY RESIDENT BP, MR LIBRARY, AND MR PROGRAMS INTO MEMORY.

K. $CNFG MAKES SELECT CODE RE-ASSIGNMENTS, IN TRAP CELLS, INTERRUPT TABLE, AND EQTS.

L. $CNFG SETS UP USER MAP FOR $CNFX AS A TYPE 3 PROGRAM AND LOADS $CNFX.

```
0002*
0003*                        7905 BOOT EXTENSION
0004*
0006  07600 063725  START LDA HIGH        HIGH CORE ADDRESS(INIT. AT 20118)
0007  07601 003300        CMA,CCE         SET DIRECTION BIT
0008  07602 073741        STA RECNT       INIT COUNT
0009  07603 005500        ERB             100000B IS LOW CORE ADDRESS
0010  07604 106702        CLC 2           WITH DIRECTION BIT SET
0011  07605 106602        OTB 2           SET MEMORY ADDRESS REGISTER
0012  07606 063733        LDA SC
0013  07607 002003        SZA,RSS         COMING FROM PAPER TAPE BOOT?
0014  07610 102501        LIA 1           YES,GET CONTENTS OF S REGISTER
0015  07611 101046        LSR 6
0016  07612 013753        AND B77         MASK SELECT CODE OF DISC
0017  07613 073733        STA SC          SAVE IT
0018  07614 163731  LOOP  LDA HDA,I       CONFIGURE I/O INSTR FROM STIO
0019  07615 167731        LDB HDA,I
0020  07616 013751        AND IOMSK       MASK OUT LOWER 6 BITS IN INSTR
0021  07617 047733        ADB SC          CONFIGURE INSTR FOR DISC SC
0022  07620 053746        CPA IOG         IS THIS INSTR IN I/O GROUP?
0023  07621 177731        STB HDA,I       YES, THEN STORE IT BACK
0024  07622 037731        ISZ HDA         MOVE ON TO NEXT INSTR
0025  07623 063731        LDA HDA
0026  07624 053734        CPA HDA3        ALL DISC I/O INSTR CONFIGURED?
0027  07625 002405        CLA,INA,RSS     YES, SET A TO 1 FOR SECTOR #
0028  07626 027614        JMP LOOP        NO THEN CONFIGURE NEXT ONE
0029  07627 073755  SLOAD STA BENT
0030  07630 063754        LDA T#ACK
0031  07631 006400        CLB
0032  07632 100400        DIV #HDS        GET RELATIVE TRACK/HEAD
      07633 007747
0033  07633         DDIV  EQU *-1
0034  07634 043743        ADA TBASE       ADD TRACK ZERO TO GET ABS. TRACK
0035  07635 073730        STA CYLA1       SAVE FOR ADDRESSING
0036  07636 073733        STA CYLA3       SAVE FOR ADDRESSING
0037  07637 047750        ADB BHD#        ADD THE BASE HEAD ADDRESS
0038  07640 063755        LDA BENT        GET SECTOR
0039  07641 005727        BLF,BLF         PUT HEAD IN HIGH B AND
0040  07642 047755        ADB BENT        ADD THE SECTOR
0041  07643 002001        RSS             SKIP OVER ADDRESS OF BENT
0042  07644 002166        OCT 2166        DEFINE ADDR. OF BENT(INIT. AT 205
0043  07645 077731        STB HDA         SET THE HEAD/SECTOR ADDRESSES
0044  07646 077734        STB HDA3
0045  07647 100047        LSL 7           SECTOR TIMES 128
0046  07650 003004        CMA,INA         AND SUBTRACT FORM
0047  07651 043740        ADA #WDTK       NUMBER OF WORDS PER TRACK
0048  07652 073724        STA P#WDS       SET POSITIVE # WORDS
0049  07653 003004        CMA,INA         AND
0050  07654 073725        STA N#WDS       NEG. # WORDS THIS TRACK
0051  07655 063741        LDA RECNT       GET # LEFT
0052  07656 002021        SSA,RSS         IF POSITIVE
0053  07657 124003        JMP 3,I         DONE, SO EXIT
0054*
0055  07660 043724        ADA P#WDS       ELSE SET TO READ
0056  07661 073741        STA RECNT       SAVE REMAINING COUNT
0057  07662 002020        SSA             NEXT TRACK
0058  07663 002400        CLA             USE MIN. OF # ON TRACK OR
0059  07664 043725        ADA N#WDS       NUMBER LEFT
```

```
07665 102702        STC 2          SET DMA FOR WORD COUNT
07666 102602        OTA 2          AND SENT IT
07667 067742        LDB D#PRM      GET THE COMMAND
07670 160001  SLOOP LDA 1,I
07671 001275        RAL,CLE,SLA,ERA IF SIGN BIT SET
07672 106700  DSK10 CLC 0          SEND COMMAND IS COMMING
07673 103600  DSK11 OTA 0,C        SEND THE COMMAND
07674 057744        CPB A#DMA      IF DMA
07675 103706        STC 6,C        START IT
07676 102700  DSK12 STC 0          ALLOW ATTENTION
07677 006045        SEZ,INB,RSS    IF NOT A COMMAND
07700 027703        JMP STDMA      DON'T WAIT FOR FLAG
*
07701 102300  DSK13 SFS 0          WAIT FOR THE FLAG
07702 027701        JMP *-1
07703 102106  STDMA STF 6          STOP DMA IF NEEDED
07704 057745        CPB A#END      END OF LOOP?
07705 002001        RSS            SKIP IF END
07706 027670        JMP SLOOP      NOT END AROUND WE GO
*
07707 103500  DSK14 LIA 0,C        GET STATUS 1
07710 102300  DAK15 SFS 0          WAIT FOR FLAG
07711 027710        JMP *-1
07712 107500  DKS16 LIB 0,C        GET STATUS 2
07713 013723        AND C174B      ISOLATE
07714 002003        SZA,RSS        IF NO ERRORS
07715 027721        JMP OK         CONTINUE
*
07716 101100        SWP            SWITCH A AND B REG. CONTENTS
07717 102031  HLT31 HLT 31B        ELSE HALT
07720 027717        JMP HLT31      TRY AGAIN
*
07721 037754  OK    ISZ T#ACK      STEP THE TRACK ADDRESS
07722 027627        JMP SLOAD      GO LOAD(A=0=SECTOR ADDRESS)
*                    DATA AREA
07723 017400  C174B OCT 17400
07724 177600  P#WDS DEC -128
07725 077477  N#WDS OCT 77477
07725         HIGH  EQU N#WDS
07726 113000  WAK   OCT 113000
07727 101200  SKCMD OCT 101200
07730 077600  CYLA1 OCT 77600
07731 077672  HDA   OCT 77672
07732 106000  AD#RC OCT 106000
07733 000000  CYLA3 NOP
07733         SC    EQU CYLA3
07734 077713  HDA3  OCT 77713
07735 107404  FILM# OCT 107404
07736 102400  R#CMD OCT 102400
07737 101400  S#TAC OCT 101400
07740 014000  #WDTK DEC 6144
07741 077600  RECNT OCT 77600      CONFIGURED TO BBL ADDRESS
07742 077726  D#PRM OCT 77726
07743 000000  TBASE NOP            FIRST TRACK#-MUST BE AT START+143
07744 077736  A#DMA OCT 77736
07745 077740  A#END OCT 77740
07746 102000  IOG   OCT 102000
07747 000002  #HDS  DEC 2          # SURFACES
07750 000000  BHD#  NOP            STARTING HEAD #
07751 172076  IOMSK OCT 172076
```

```
0120   07752 002011  SPCAD OCT 2011
0121   07753 000077  B77   OCT 77
0122   07754 000000  T#ACK NOP
0123*
0124*               THE FOLLOWING CODE IS EXECUTED WHEN THE
0125*                BOOT EXTENSION IS ENTERED AT 2011B.
0126*
0127   07755 000000  BENT  NOP                JSB HERE FROM BBDL(INIT. AT 2166B
0128   07756 102106        STF 6              CLEAN UP DMA
0129   07757 107700        CLC 0,C            AND THE I/O SYSTEM
0130   07760 006400        CLB                ELIMINATE HLT 77B
0131   07761 102501        LIA 1              READ S REG.
0132   07762 072144        STA SC             SAVE S REG. CONTENTS
0133   07763 101045        LSR 5
0134   07764 002011        SLA,RSS            WAS BIT 5 OF S REG. SET?
0135   07765 026201        JMP NORCN          NO, THEN RECONFIG. NOT REQD
0136   07766 102077        HLT 77B            YES, THEN HALT TO LET USER SET S
0137   07767 026202        JMP DRBOT          RELOCATE THE REST OF THIS BOOT
0138   07770 106601  NORCN OTB 1              CLEAR S REG.
0139*
0140   07771 162163  DRBOT LDA SPCAD,I        MOVE 128 WORDS TO BBL-128
0141   07772 172152        STA RECNT,I
0142   07773 036163        ISZ SPCAD
0143   07774 036152        ISZ RECNT
0144   07775 036135        ISZ P#WDS          DONE?
0145   07776 026202        JMP DRBOT          NO, GET NEXT WORD
0146*
0147   07777 126141        JMP CYLA1,I        YES,GO EXECUTE THE BOOT
0148*
0150                       END START
```

BOOT PROCESS
COMPLETION
(SYSTEM START UP)

A. $CNFX - Redefine and remap SAM extension including up to 5 bad pages. Accept changes to partitions, program sizes, and partition assignments. Record in memory and optionally on disc.

B. $STRT - Set up SAM by calling $RTN. EQT 1 thru EQT 6 on BP contain SAM block addresses and sizes. EQT1&2 are SAM default block. EQT3&4 are SAM extension and EQT5&6 are SAM in Table Area I.

C. $ZZZZ - Disable interrupt system

D. $ZZZZ - Subroutine MPINT sets up remaining three maps, base page fence, and MAT table linked lists.

E. $ZZZZ - Schedule FMGR to set up the file system.

F. $STRT - Save ID segment addresses of FMGR, D.RTR, EDIT, and SMP.

G. $SCLK - Start the TBG and print "SET TIME".

H. $XEQ - Idle loop.

BOOT EXTENSION

ON DISC

LOADED AT 2011B

```
LU    2  TRK       0  SECTR  =      0
063725 003300 073741 005500 106702 106602 063733 002003*          @
102501 101046 013753 073733 163731 167731 013751 047733* A &            O
053746 177731 037731 063731 053734 002405 027614 073755*W    ?    W    /
063754 006400 100400 077747 043743 073730 073733 047750*          G     O
063755 005727 047755 002001 002166 077731 077734 100047*     O           !
003004 043740 073724 003004 073725 063741 002021 124003*  G
043724 073741 002020 002400 043725 102702 102602 067742*G        G
160001 001275 106700 103600 057744 103706 102700 006045*          +      %
```

BOOT START ADDRESS (2055B)

JMP 3,I to $STRT

LWA OF RTE FROM BP THRU $CNFG

```
LU    2  TRK       0  SECTR  =      1
027703 102300 027701 102106 057745 002001 027670 103500*/    /  F+   /  @
102300 027710 107500 013723 002003 027721 101100 102031*  /  @     /  @
027717 037754 027627 017400 177600 057622 113000 101200*/ ? /      +
077600 077672 106000 000000 077713 107404 102400 101400*
014000 077600 077726 000000 077736 077740 102000 000001*
000002 172076 002011 000077 000000 000000 102106 107700*    >   ?     F
006400 102501 072144 101045 002011 026201 102077 026202*    A   %  ,  ?,
106601 162163 172152 036163 036152 036135 026202 126141*       <  <  <],
```

BOOT EXTENSION
RELOCATION ADDRESS

START
(2166B)

RECONFIGURATOR HALT
(BIT 5 SET)

BASE PAGE COPY

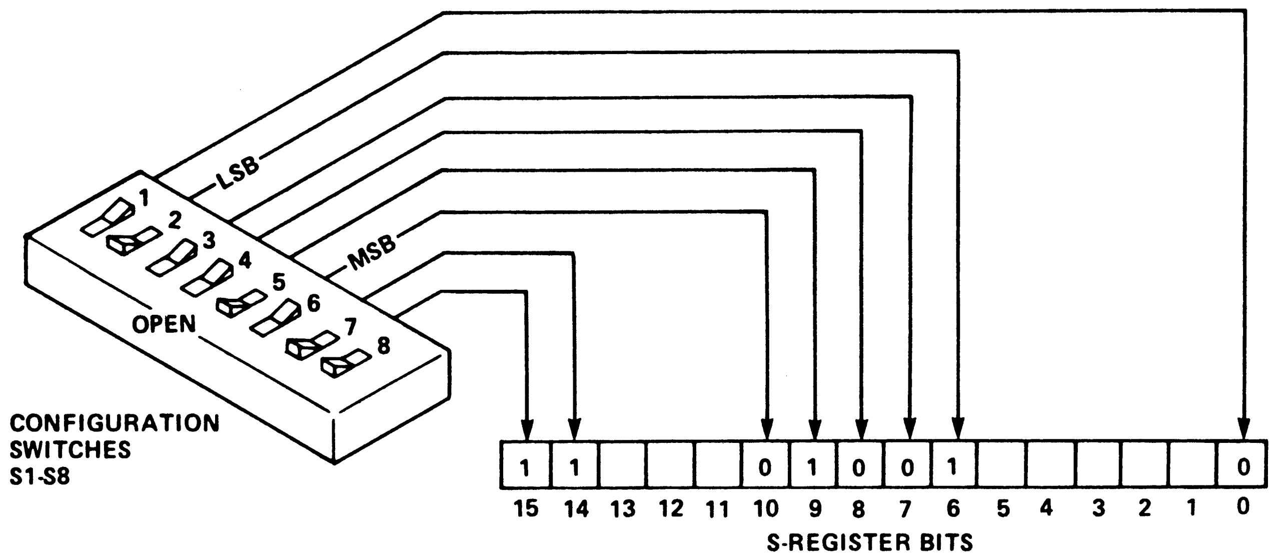
```
LU    2  TRK       0  SECTR  =      2
115644 115644 124003 045056 115621 115644 115644 115644*         J.
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 115644*
115644 115644 115644 115644 115644 115644 115644 000000*
```

$STRT ADDRESS

REMOTE PROGRAM LOADING
RPL

RTE IS AUTOMATICALLY BOOTED WHEN POWER IS TURNED ON AND THE LOCK/OPERATE SWITCH IS IN THE LOCK POSITION.

S-REGISTER BOOT INFORMATION IS OBTAINED FROM THE CONFIGURATION SWITCHES MOUNTED ON THE CPU BOARD.



SWITCH	FUNCTION
S1	7905/7920 DISC OPEN POSITION SELECTS HEAD 000, REMOVABLE PLATTER CLOSED POSITION SELECTS HEAD 010, FIXED PLATTER ON 7905 9885 FLEXIBLE DISC MUST BE IN CLOSED POSITION. ALL LOADING STARTS AT TRACK 0, SECTOR 1 7900/7901 DISC OPEN POSITION SELECTS FIXED PLATTER CLOSED POSITION SELECTS REMOVABLE PLATTER
S2 S3 S4 S5 S6	DECODES THE FIVE BIT OCTAL SELECT CODE OF THE INPUT DEVICE (S.C. 10-37) SWITCHES SHOWN SET TO SELECT CODE 11_8 (01001_2)
S7	IF RPL ENABLED, OPEN POSITION SELECTS ROM 10; CLOSED POSITION SELECTS ROM 11
S8	OPEN POSITION RPL NOT ENABLED CLOSED POSITION RPL ENABLED

DBUGR/CMM4

DBUGR FEATURES

Used to debug programs under RTE-IV

* Symbolic or octal printout
* Register examination and change
* Memory search
* Memory clear
* Breakpoint
* Map examination.

DBUGR EXECUTION

* Relocating with program to be debugged

```
IN-LINE:     FTN4,L,M
                     PROGRAM PROGA
                     CALL DBUGR [or CALL DBUGR(LU)]
                     .
                     .
                     .
                       Where LU = lu # of console if omitted,
                       LU is defaulted to terminal from which
                       the program was scheduled.
          OR

WITH LOADR:     *RU,LOADR,,%PROGA,,DB
```

* Starting DBUGR

```
   *RU,PROGA

   START DBUGR ←  (DBUGR PRINTOUT)

   (DBUGR IS READY TO ACCEPT COMMANDS)
```

DBUGR SAMPLE PROGRAM

(SOURCE)

This program will be used for the DBUGR command examples

```
FTN4,L,M
      PROGRAM PROGA
      DIMENSION IARY(10)
      CALL DBUGR
    5 I = 10
      J = 3
      JJ = 2HAB
      X = 2.3
      DO 10 I = 1,10
      IARY(I) = I
   10 CONTINUE
      GO TO 5
      END
```

DBUGR
SAMPLE PROGRAM
(MIXED LISTING)

```
0002        PROGRAM PROGA
0003        DIMENSION IARY(10)
0004        CALL DBUGR
                                 IARY    BSS 00012B
         00012 *000000                   NOP
         00013  000001X                  JSB CLRIO
         00014  000015R                  DEF *-2+00003B
0005      5 I = 10
         00015  000002X                  JSB DBUGR
         00016  000017R                  DEF *-4+00005B
0006        J = 3
         00017  000050R @5               LDA 00050B
         00020  000051R                  STA I
0007        JJ = 2HAB
         00021  000053R                  LDA 00053B
         00022  000052R                  STA J
0008        X = 2.3
         00023  000055R                  LDA 00055B
         00024  000054R                  STA JJ
0009        DO 10 I = 1,10
         00025  000003X                  JSB .DLD
         00026  000060R                  DEF 00060B
         00027  000004X                  JSB .DST
         00030  000056R                  DEF X
         00031  000062R                  LDA 00062B
         00032  000051R                  STA I
0010        IARY(I) = I
         00033  000051R                  LDA I
         00034  000063R                  ADA 00063B
0011     10 CONTINUE
         00035  000064R                  STA A.001
         00036  000051R                  LDA I
         00037  100064R                  STA A.001,I
0012        GO TO 5
         00040  000051R @10              LDA I
         00041  000062R                  ADA 00062B
         00042  000051R                  STA I
         00043  003004                   CMA,INA
         00044  000050R                  ADA 00050B
         00045  002021                   SSA,RSS
         00046  000033R                  JMP 00033B
0013        END
         00047  000017R                  JMP @5
         00050  000012                   OCT 000012
                                 I       BSS 00002B
         00053  000003                   OCT 000003
                                 JJ      BSS 00001B
         00055  040502                   OCT 040502
                                 X       BSS 00002B
         00060  044631                   OCT 044631
         00061  115004                   OCT 115004
         00062  000001                   OCT 000001
         00063  177777R                  DEF 77777B
                                 A.001   BSS 00001B
```

*NOTE: These offsets maybe used along with the program's load address to determine the final relocated address of each instruction. For example the final address of the "STA A.001" instruction would be 26002 (see PROGA load map) +35 or 26037.

DBUGR
SAMPLE PROGRAM
(SYMBOL TABLE)
FROM COMPILER

```
SYMBOL TABLE

NAME        ADDRESS      USAGE                        TYPE       LOCATION


@10           00040R     STATEMENT NUMB
@5            00017R     STATEMENT NUMB
CLRIO         00001X     STATEMENT FUNCTION   REAL       EXTERNAL
DBUGR         00002X     STATEMENT FUNCTION   REAL       EXTERNAL
I             00051R     VARIABLE             INTEGER    LOCAL
IARY          00000R     ARRAY(*)             INTEGER    LOCAL
J             00052R     VARIABLE             INTEGER    LOCAL
JJ            00054R     VARIABLE             INTEGER    LOCAL
X             00056R     VARIABLE             REAL       LOCAL
```

(LOAD MAP)

```
PROGA   26002 26066

CLRIO   26067 26075   750701   24998-16041
DBUGR   26076 33734   92067-16075 REV.1805 780214
LOGLU   33735 34004   92067-16035 REV.1805 771117
IFBRK   34005 34034   92067-16035 REV.1805 770621

 5 PAGES RELOCATED      5 PAGES REQ'D     NO PAGES EMA
 /LOADR:PROGA   READY

 /LOADR:$END
```

NOTE: The address of each variable maybe found by adding PROGA's load address (26002) to the variables offset from the symbol table.
For example:

J	26002+52	= 26054
IARY	26002+0	= 26002

CONVENTIONS FOLLOWED IN THE DBUGR EXAMPLES

——	User inputs are underlined.
CR	Carriage return
LF	Line feed ("Control J" on 2645/2648)
[ ]	Input control character which is not echoed
\	Escape (or ALT MODE on some terminals)
377	Octal number
377.	Decimal number

DBUGR COMMANDS

(BREAK POINTS)

n\B	Set the breakpoint at location n and clear the previous breakpoint. When the breakpoint location is encountered, control is transferred to DBUGR prior to execution of the instruction. Only one breakpoint is allowed at a time.
\P	Proceed with program execution until the next breakpoint
n\P	Proceed until the breakpoint is executed n octal times.
\T	Execute or trace the current instruction and break (single-step)
n\T	Trace (single-step) the next n octal instructions and break.
n\G	Continue execution at location n
\B	Remove the breakpoint

where,
\ = ESCAPE

DBUGR EXAMPLES
(BREAK POINTS)

```
*RU,PROGA

START DBUGR

26002+34\B              - set breakpoint in "do loop"
\P                      - proceed
26036(ADA 26065)  1     115004    160010    114011    0  [CR]

(break-)(instruction)  (A reg.)  (B reg.)  (X reg.)  (Y reg.)  (EO reg.)
(point address)

                        printout when breakpoint encountered

3\P                     - break after 3 executions of the breakpoint instruction
26036(ADA 26065)  4     115004    160010    114011    2  [CR]
                        breakpoint encountered again
\T                      - execute one instruction
26037(STA 26066)  26005    115004    160010    114011    2  [CR]
3\T                     - execute three instructions
26040(LDA 26053)  26005    115004    160010    114011    2
26041(STA 26066,I)  4    115004    160010    114011    2
26042(LDA 26053)  4     115004    160010    114011    2  [CR]
26002+20\B              - reset breakpoint
26002+17/G              - begin execution at 26021
26022(STA 26053)  12    115004    160010    114011    2  [CR]
\B                      - remove breakpoint
\P                      - proceed
END DBUGR               - program continues execution no longer under DBUGR
                          control
```

DBUGR COMMANDS
(MEMORY MODIFICATION)

n < s:	Define the symbol s as the value n
n/	Print and open for modification location n
[CR]	Close the open location
n [CR]	Store n (assembly instruction or octal constant) in the open location
[LF]	Print and open the next location
n [LF]	Same as n [CR] and also print and open the next location.
n "[CR] or n" [LF]	Same as above except n is interpreted as one or two ASCII characters.

EXAMPLES
(MEMORY MODIFICATION)

```
:RU,PROGA

START DBUGR

26002<R1:[CR]                         - set "R1" equal to PROGA's load address
                                        (26002)
R1+34\B                               - set breakpoint at an offset of 34 into PROGA
                                        (26036)
\                                     - proceed
R1+34(ADA R1+63)  1    115004    160010    114011    2   [CR]
R1+52/  3  7  [CR]                    - display and change "J" to 7
R1+52/  7  [CR]                       - display "J"
R1/  0  [LF]                          - display "IARY(1)"

R1+1/  0  2\P                         - display "IARY(2)" and break after 2 breakpoint
                                        instructions
R1+34(ADA R1+63)  3    115004    160010    114011    2   [CR]
R1/  1  7  [LF]                       - change "IARY(1)" to 7
R1+1/  2  [CR]                        - display "IARY(2)"

26002/  7  CD"  [LF]                  - change "IARY(1)" to "CD"
R1+1/  2  [CR]                        - display "IARY(2)"
R1+21/  LDA  R1+53  CLA,INA  [LF]  - change to CLA,INA instruction
R1+22/  STA  R1+52  STA  R1+54  [CR]  - change to store into "JJ"
R1+21/  CLA,INA                       - verify change
```

DBUGR COMMANDS

^	Print and open the previous location
*/ or ./	Print and open the current location
0/ or 1/	Print and open the A or B register
\M+1/	Print and open E and O registers
\M+3/	Print and open X register
\M+4/	Print and open Y register

EXAMPLES

R1 1/ 2 [LF] — display "IARY (2)"

R1 + 2/ 0 3 [LF] — set "IARY (3)" to 3

R1 + 3/ 0 4 [CR] — set "IARY (4)" to 4

∧ — display previous location

R1 + 2/ 3 ∧ — "IARY (3)" and previous location

R1 + 1/ 2 [CR] — "IARY (2)"

1/ JSB 1004,I JSB 1003 [CR] — display and change B register

./ JSB 1003 [CR] — display current location

\M+4/ JSB 11,I — display Y register

COMMANDS
(PRINT MODE)

\S	Set print mode to symbolic instruction (default)
!	Print the last quantity typed as an instruction
\C	Set mode to constant
=	Print last quantity as a constant
\H	Set mode to ASCII characters
'	Print last quantity as two ASCII characters
n\R	Change the output radix to n

EXAMPLES
(PRINT MODE)

\M 3/ LDA 10,I [CR]	— display X register
= 160010 [CR]	— display X register as a constant
R1+54/ ADA 502 [CR]	— display "JJ"
' AB" [CR]	— display "JJ" in ASCII
\C	— change mode to constant
R1 54/ 40502 ' AB" ! ADA 502	— display "JJ" all three modes where 40502 octal = "AB" = ADA 502.
16 = 16 16. = 20 [CR]	— convert 20. to octal
10.\R	— set output radix to 10
16 = 14. 16. = 16. [CR]	— convert 16 to decimal
555 = 365.	— convert octal to decimal

```
INPUT  FUNCTION
 ID  LIST ID SEGMENT
 EQ  LIST EQT AND EXTENTS
 DR  LIST DEV REF TABLE
 LM  LIST MEMORY
 XL  LIST MEMORY   (SYSTEM MAP)
 IN  LIST INTERUPT TABLE
 TA  LIST TRACK ASSIGNMENT TABLE
 TR  TRACE LIST
 XT  TRACE LIST (SYSTEM MAP)
 LP  LIST DISC RES PROGRAM
 DP  DISPLAY INPUT IN OCTAL DECIMAL & ASCII
 PG  LIST ANY LOCATION IN PHYS MEMORY
 PP  MODIFY ANY LOCATION IN PHYSICAL MEMORY
 LL  CHANGE LIST DEVICE
 PM  PATCH MEMORY
 XP  PATCH MEMORY    (SYSTEM MAP)
 F/  FIND A VALUE IN MEMORY
 XF  FIND A VALUE IN MEMORY  (SYSTEM MAP)
 LI  LIST ENTRY POINT
 DI  REPORT DISC DICTIONARY ADDRESS OF  AN ENTRY POINT
 LE  LIST ALL ENTRY POINTS IN SYS
 DL  LIST DISC SECTOR
 DM  DISC MOD  ANY LU
 DS  DISC SEARCH
 MS  MOVES DISC SECTORS TO ANOTHER DISC AREA
 NS  SET #  OF SECTRS PER TRACK
 FP  DISPLAY PAST DISK MODS
 /E OR EN OR EX TO EXIT
 FOR MORE INFO DO A ??,INPUT
 A PK AFTER THE INPUT GIVES A PACKED LISTING
```

```
ID,PROGRAM NAME
ID,SEGMENT NAME
ID,NUMBR = ALL ID'S IN SYSTEM
OR USE  IDPK,
EQ,NUMBR
EQ,NUMBR,NUMBR  GIVES EQTS INCLUSIVE
OR USE  EQPK,
DR,NUMBR
DR,NUMBR,NUMBR  GIVES DRT ENTRIES INCLUSIVE
OR USE  DRPK,
LM,ADDRESS
LM,ADDRESS,# OF WORDS
OR USE  LMPK,
XL,ADDRESS     (SYSTEM MAP)
XL,ADDRESS,# OF WORDS   (SYSTEM MAP)
OR USE  XLPK,
IN,NUMBR
IN,NUMBR,NUMBR  GIVES INT TABLE ENTRIES INCLUSIVE
OR USE  INPK,
TA
TA,LU #
TA,LU #,TRK #, # OF TRKS
OR USE  TAPK,
TR,START LOCATION,LIST DELIMITER
XT,START LOCATION,LIST DELIMITER
LP,PROG NAME,REL ADDRESS
OR USE  LPPK,
DP,VALUE
DP,VALUE,*,VALUE
DP,VALUE,/,VALUE
DP,VALUE,+,VALUE
DP,VALUE,-,VALUE
PG, PG#,OFFSET,# OF WORDS
OR USE  PGPK,
PP, PG#, OFFSET, NEW VALUE
OR USE  PPPK,
LL,LIST LU#
PM,ADDRESS,NEW VALUE
XP,ADDRESS,VALUE     (SYSTEM MAP)
F/,VALUE TO FIND,START ADDRESS,# OF WORDS
XF,VALUE TO FIND,START ADDRESS,# OF WORDS
LI,ENTRY POINT NAME
DI,ENTRY POINT NAME
LE
DL,LU,TRK,SECTR, # OF SECTORS
OR USE  DLPK,
DM    DISC MOD    <INTERACTIVE>
DS,LU,TRK,VALUE TO FIND
      SOURCE IS:    DESTINATION IS:
MS,   LU,TRK,SECTR,  LU,TRK,SECTR,#  OF SECTRS
```

OPERATOR REQUESTS

ENVIRONMENT BEFORE A KEY IS PRESSED ON THE SYSTEM CONSOLE

1. System console is in input mode.
2. System console select code (SC) is 15.
3. System console driver is DVR05.
4. RTE is idle (no executing programs).

EXECUTION	SUBROUTINE CALLS	NOTES
$CIC(RTIOC)		Entered by depressing a key on the console, since DVR05 (and DVR00) leaves each terminal with its device control flip-flop set. Interrupt causes instruction in trap cell location 15 to be exexuted (JSB $CIC,I). Save machine state, turn off interrupts, etc. Use interrupt and EQT tables to find the driver continuation/completion address.

DISCUSS TRAP CELLS & INTERRUPT TABLE

EXECUTION	SUBROUTINE CALLS	NOTES
	C.05(DVR05)	Set "OPATN" to tell system that operator wants service.
$CIC		If console is not busy (OPFLG):
$TYPE(SCHED)		
	$XSIO(RTIOC)	Request DVR05 to output "*".
	$XSIO	Request DVR05 to read operator input and return to TYP10 upon completion.
$XEQ(DISP)		RTE idle loop (or resume execution of the interrupted programs).

Prompt ("*") is output
Operator inputs: "ON,XYZ"

EXECUTION	SUBROUTINE CALLS	NOTES
$CIC		Save machine state, etc.
	C.05(DVR05)	Detect end of input line and completion of I/O request.
IOCOM ($CON1)		Remove I/O request and jump to completion address.

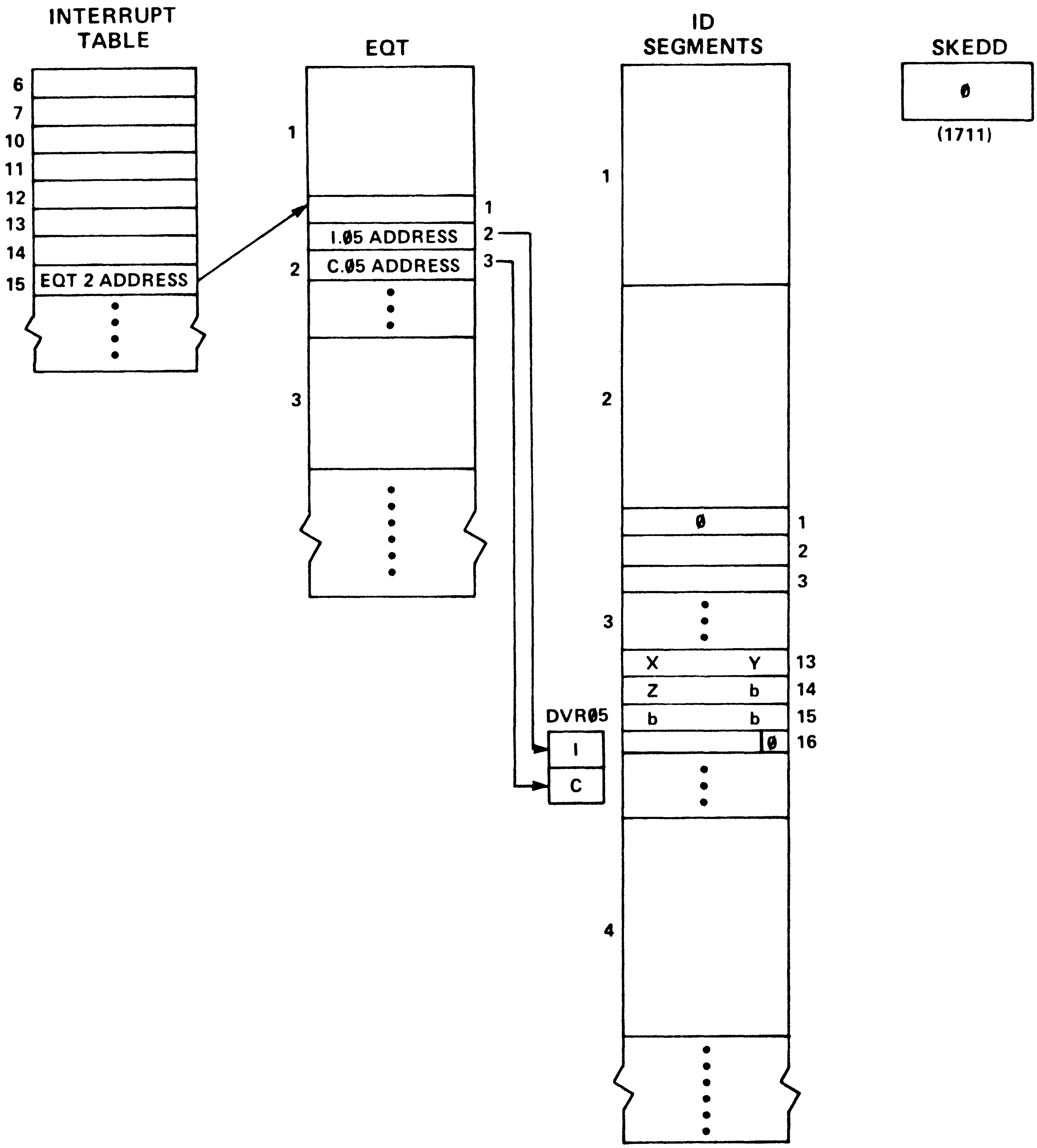
EXECUTION	SUBROUTINE CALLS	NOTES
TYP10		Clear "OPFLG" (system console not busy)
	$MESS($MSG-SCHED)	Operator command processor
	$PRSE	Parse the request "ON,XYZ"

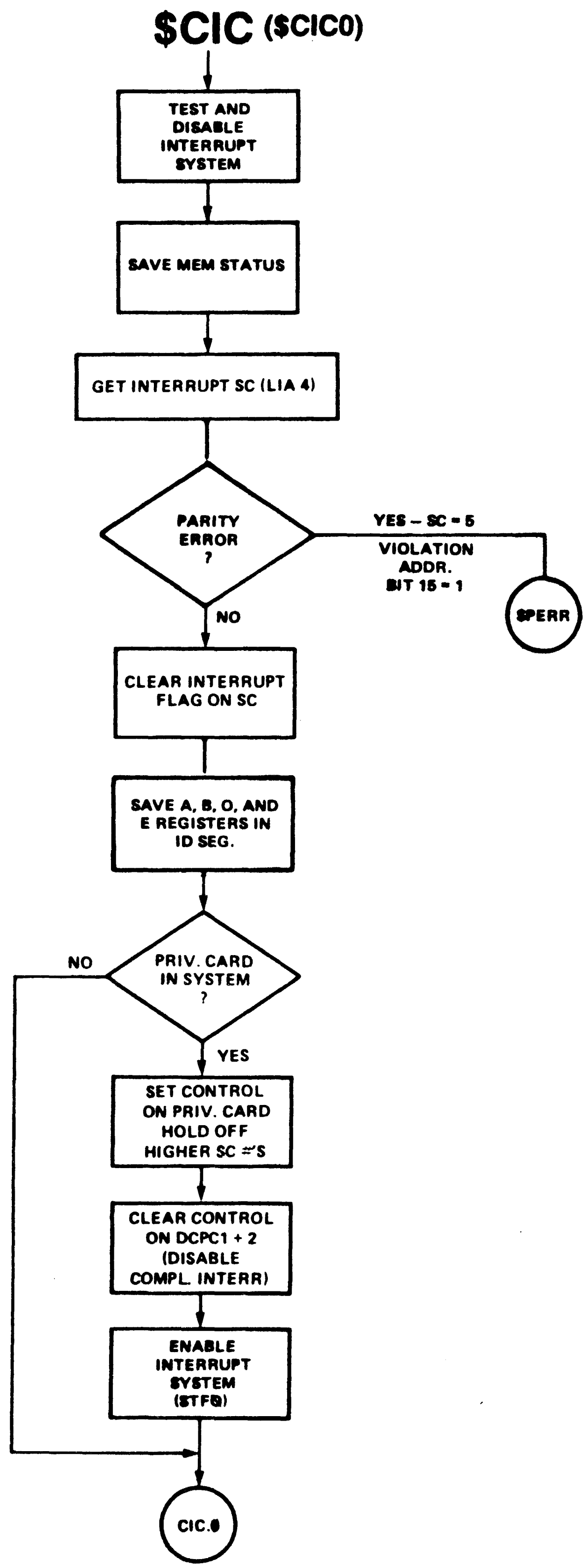
DISCUSS KEYWORD BLOCK

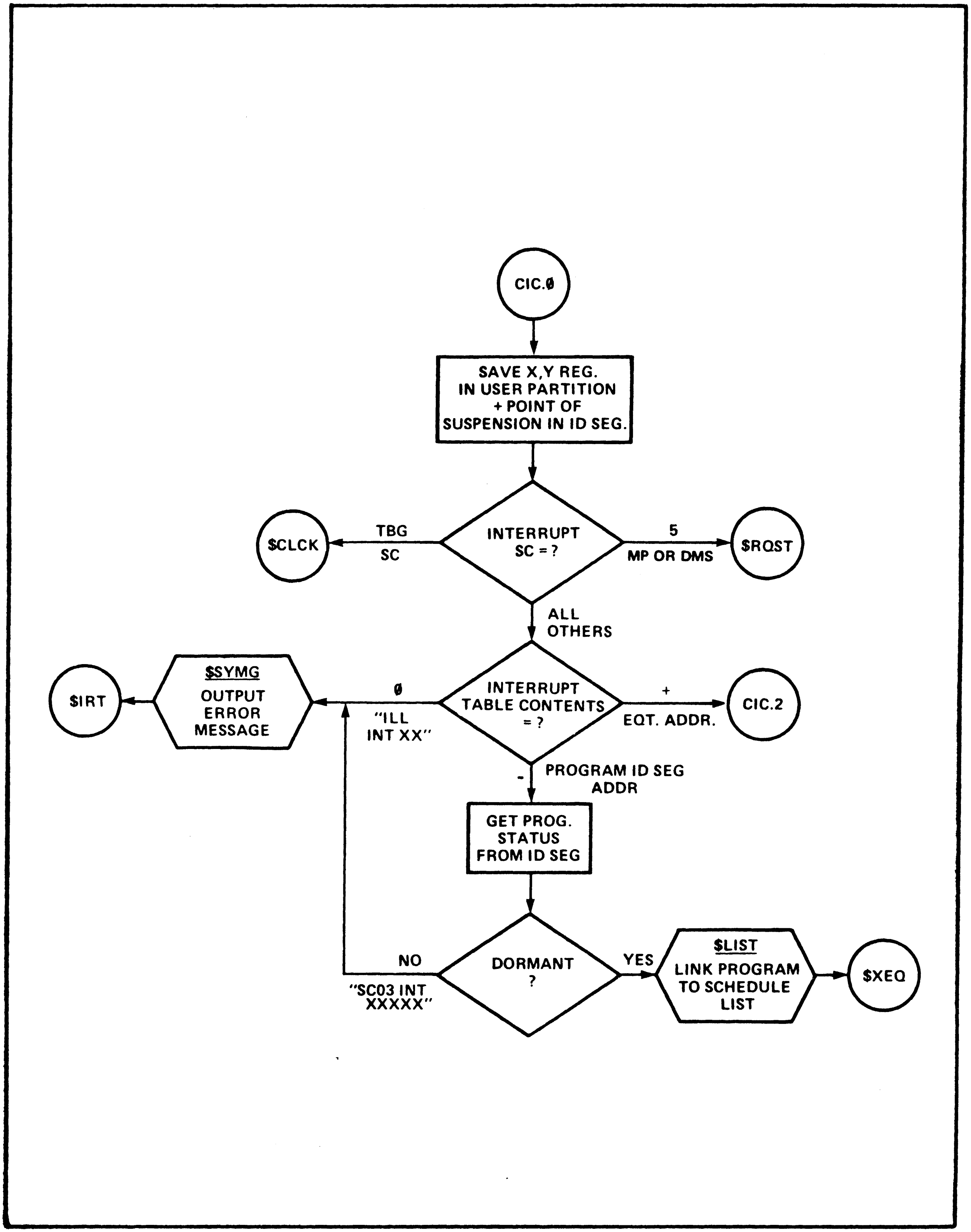
	M0100	ON,XXXX processor
	TTNAM	Find ID segment address of XYZ
	M0100	See if program is dormant
	PLOAD	Store parameters (P1-P5) into ID segment.
	$LIST(SCHED)	Schedule* the program by changing its state and adding to the scheduled list.
TYP10		
	$XSIO	Output message if necessary
$XEQ		See if any programs are ready to be executed.
	X0010	Dispatch* program XYZ for execution.

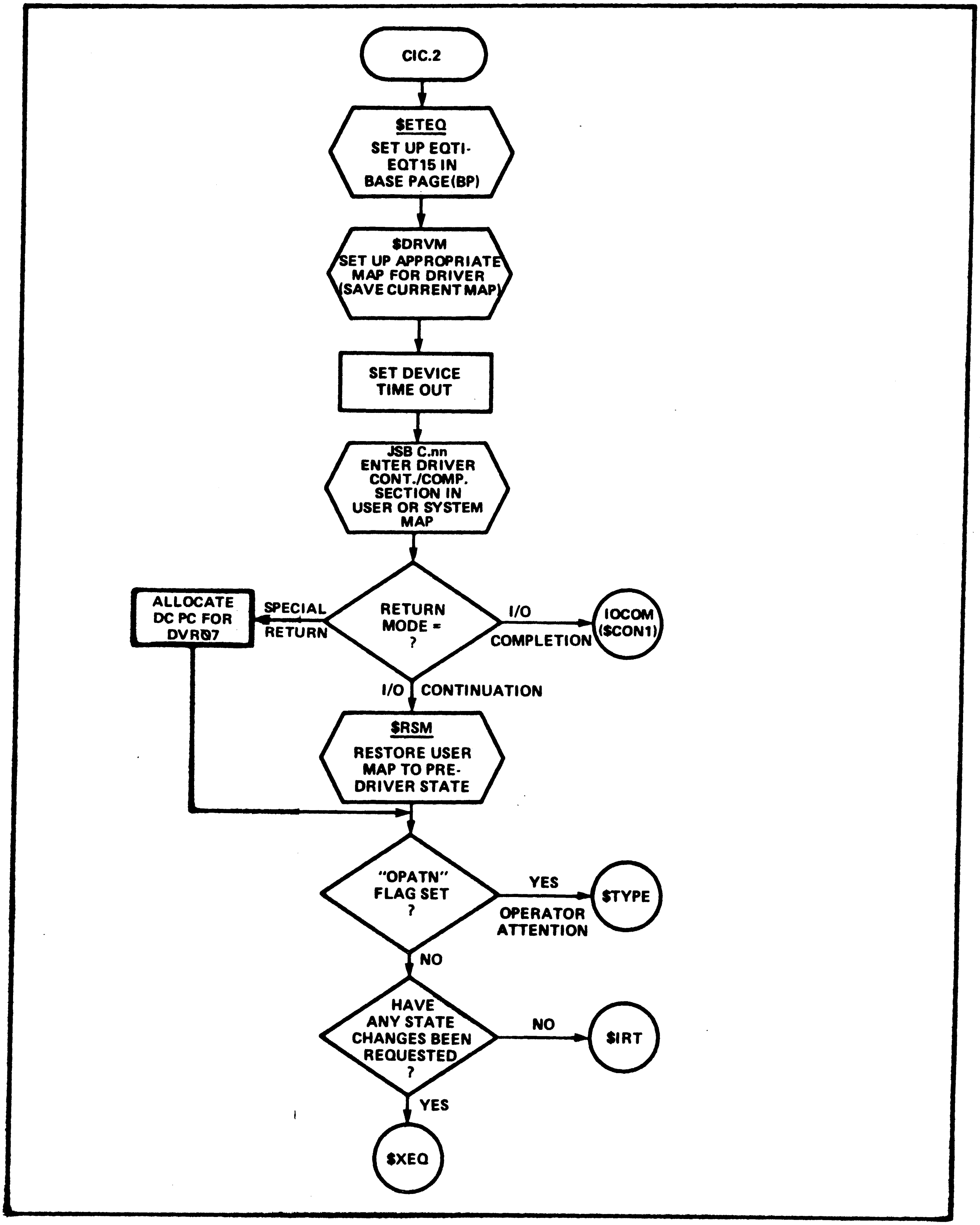
*Note the distinction made in RTE between scheduling and dispatching a program.

"ON,XYZ" TABLES/LISTS (INITIAL CONDITIONS)

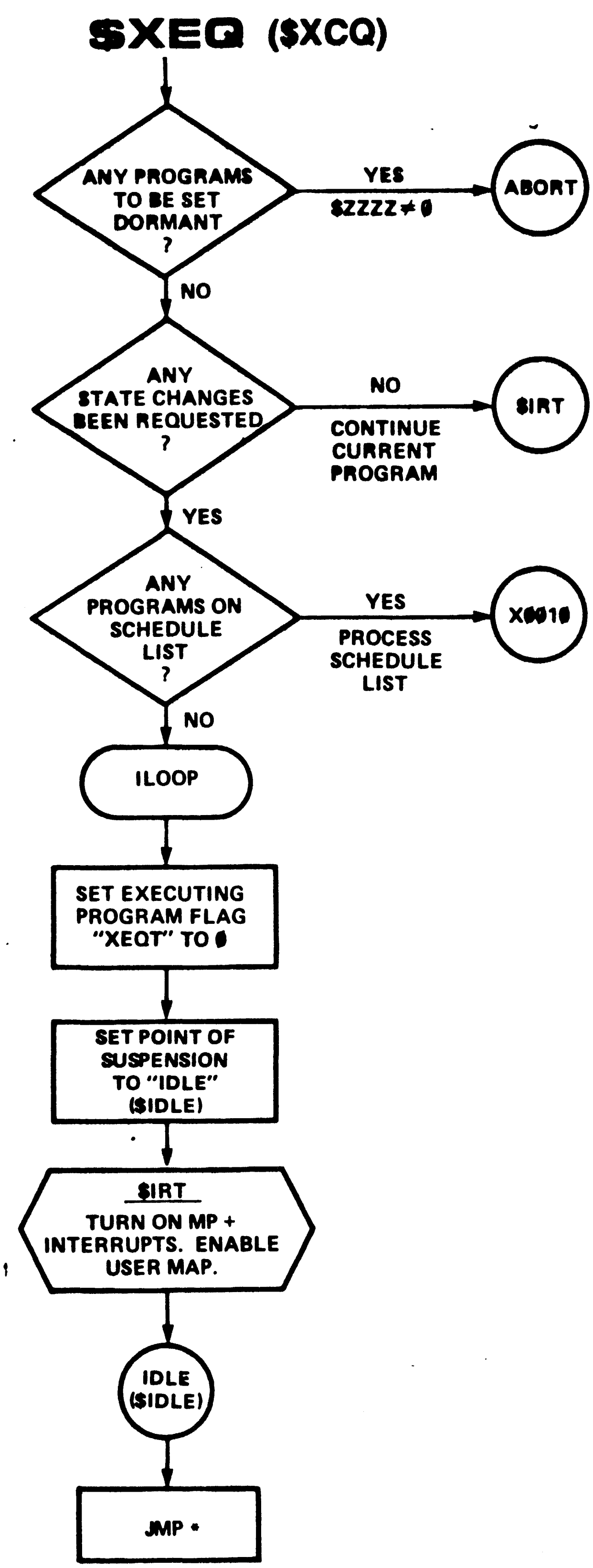



$CIC ($CIC0)
TEST AND DISABLE INTERRUPT SYSTEM
SAVE MEM STATUS
GET INTERRUPT SC (LIA 4)
PARITY ERROR ?
YES – SC = 5
VIOLATION ADDR. BIT 15 = 1
$PERR
NO
CLEAR INTERRUPT FLAG ON SC
SAVE A, B, O, AND E REGISTERS IN ID SEG.
PRIV. CARD IN SYSTEM ?
NO
YES
SET CONTROL ON PRIV. CARD HOLD OFF HIGHER SC #'S
CLEAR CONTROL ON DCPC1 + 2 (DISABLE COMPL. INTERR)
ENABLE INTERRUPT SYSTEM (STFO)
CIC.0


CIC.Ø
SAVE X,Y REG. IN USER PARTITION + POINT OF SUSPENSION IN ID SEG.
INTERRUPT SC = ?
TBG SC
$CLCK
5
MP OR DMS
$RQST
ALL OTHERS
INTERRUPT TABLE CONTENTS = ?
Ø
"ILL INT XX"
$SYMG
OUTPUT ERROR MESSAGE
$IRT
+
EQT. ADDR.
CIC.2
-
PROGRAM ID SEG ADDR
GET PROG. STATUS FROM ID SEG
DORMANT ?
NO
"SC03 INT XXXXX"
YES
$LIST
LINK PROGRAM TO SCHEDULE LIST
$XEQ


CIC.2
$ETEQ
SET UP EQTI-
EQT15 IN
BASE PAGE(BP)
$DRVM
SET UP APPROPRIATE
MAP FOR DRIVER
(SAVE CURRENT MAP)
SET DEVICE
TIME OUT
JSB C.nn
ENTER DRIVER
CONT./COMP.
SECTION IN
USER OR SYSTEM
MAP
RETURN
MODE =
?
SPECIAL
RETURN
ALLOCATE
DC PC FOR
DVR07
I/O
COMPLETION
IOCOM
($CON1)
I/O CONTINUATION
$RSM
RESTORE USER
MAP TO PRE-
DRIVER STATE
"OPATN"
FLAG SET
?
YES
OPERATOR
ATTENTION
$TYPE
NO
HAVE
ANY STATE
CHANGES BEEN
REQUESTED
?
NO
$IRT
YES
$XEQ

```
*
*******************************************************************
*
*      ON[IH],XXXXX
*      ON[IH],XXXXX,NOW
*      ON[IH],XXXXX,P1,...,P5
*      ON[IH],XXXXX,NOW,P1,...,P5
*
*         THE ON REQUEST FUNCTIONS AS FOLLOWS:
*               IF NO RESOLUTION CODE, THEN PROGRAM SCHEDULED.
*               IF -NOW- OPTION, THEN ENTER PROGRAM INTO TIME LIST
*                    AND SET TIME VALUES TO CURRENT TIME PLUS 10 MSC
*               IF NOT ONE OF ABOVE, AND TIME VALUES ARE ZERO THEN
*                    PROGRAM FUNCTIONS SAME AS -NOW- OPTION.
*               IF NOT ONE OF ABOVE, AND TIME VALUES ARE PRESENT,
*                    THEN PROGRAM IS ADDED TO TIME LIST.
*               NOTE: 1)ALL THE ABOVE OPTIONS ALLOW PARAMETERS TO BE
*                       PASSED TO THE PROGRAM. THESE MUST BE ASCII
*                       DECIMAL NUMBERS WHICH ARE CONVERTED TO BINARY
*                       AND STORED IN ID SEGMENT TEMP AREA. UPON
*                       EXECUTION, THE B REGISTER WILL POINT TO TEMP.
*                       UP TO 5 PARAMETERS MAY BE INPUT. IF NO PARA-
*                       METERS ARE INPUT, THE TEMP AREA ARE ZEROS BUT
*                       B REGISTER WILL STILL POINT TO TEMP. AREA
*                    2) THE ABOVE OPTIONS WILL ALLOW THE ORIGINAL
*                       SCHEDULING STRING TO BE SAVED(UNLESS 'IH'
*                       IS SPECIFIED OR THERE ARE NO PARAMETERS).
*                       THE SCHEDULED PROGRAM MAY RECOVER THIS STRING
*                       WITH AN EXEC 14 CALL.
*
*********************************************************************
*
 01164 017404R M0100 JSB TTNAM    FIND ID SEGMENT ADDR
 01165 166605R       LDB WSTAT,I  IF NO PARAMETERS
 01166 005222        RBL,RBL       BIT IS SET, THEN
 01167 006021        SSB,RSS        ILLEGAL STATUS.
 01170 002002        SZA          CHECK IF PROGRAM DORMANT
 01171 027470R       JMP M0405    ILLEGAL STATUS ERROR
 01172 017550R       JSB $SZIT    CHECK OUT THE PROGRAM SIZE
 01173 002002        SZA          IS IT OK ?
 01174 026670R       JMP MSEX     NO, FLUSH HIM !
*
 01175 017267R       JSB PLOAD     GO TO PROCESS CONTROL PRAMETERS
 01176 066057X       LDB WORK
 01177 046026R       ADB D17      COMPUTE RES/T/MULT ADDR
 01200 160001        LDA B,I
 01201 001723        ALF,RAR
 01202 012022R       AND D7       CHECK RESOLUTION CODE
 01203 002002        SZA          NONE, SO GO TO SCHED NOW
 01204 027210R       JMP M0110
 01205 016032X M0105 JSB $LIST    SCHEDULE PROGRAM
 01206 000301        OCT 301
 01207 026670R       JMP MSEX     RETURN
 01210 006004  M0110 INB           SET B FOR $ONTM
 01211 062036R       LDA CP2       IF ASCII
```



$SXEQ ($XCQ)
ANY PROGRAMS TO BE SET DORMANT ?
YES
$ZZZZ ≠ 0
ABORT
NO
ANY STATE CHANGES BEEN REQUESTED ?
NO
CONTINUE CURRENT PROGRAM
$IRT
YES
ANY PROGRAMS ON SCHEDULE LIST ?
YES
PROCESS SCHEDULE LIST
X0010
NO
ILOOP
SET EXECUTING PROGRAM FLAG "XEQT" TO 0
SET POINT OF SUSPENSION TO "IDLE" ($IDLE)
$IRT
TURN ON MP + INTERRUPTS. ENABLE USER MAP.
IDLE ($IDLE)
JMP *

OPERATOR COMMAND PROCESSORS

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED		REMARKS
RT	M0070	$OTRL		SETS CALL TO $SDRL
			$SDRL	SCANS TAT, RELEASES TRACKS
ON	M0100	$LIST		IF NO TIME PRAMS THEN SCHEDULE PROGRAM, IF AVAILABLE.
		$ONTM		IF TIME PRAMS THEN SET START TIME
			$TADD	ADDS PROG TO TIME LIST
OF(,0)	M0200	SABRT		SOFT ABORT
			$TREM	REMOVE FROM TIME LIST
			$LIST	SET DORMANT
OF(,1) or		$IOCL		IF I/O SUSP, CLEAR I/O
OF(,8)		$ABRT		HARD ABORT(OF,1)
			$TREM	REMOVE FROM TIME LIST
			$SDRL	RELEASE DISC TRACKS
			$SYMG	ABORT MESSAGE
		$DREL		RELEASE PROG´S TRACKS (OF,8 ONLY)
SS	M0300	$LIST		PUT IN SUSPEND LIST
GO	M0400	$ILST		ILLEGAL STATUS MESSAGE
		$LIST		OR PUT IN SCHEDULED LIST
ST	M0500	$CNV1		FORMAT MESSAGE PARAMETERS
PR	M0650	$INER		ERROR MESSAGES
		$LIST		RE-LINK THE PROGRAM
IT	M0600	$INER		ERROR MESSAGES
		$ETTM		SET TIME VALUES IN ID SEG
		$TREM		REMOVE FROM T-LIST IF RES =0
TM	M0700	$TMRQ		DOES ALL THE WORK
			$INER	ERROR MESSAGES
			$ETTM	SETS TIME VALUES IN CLOCK
DN	M0800	$IODN		DOES THE WORK
	$IODN	$INER		ERROR MESSAGES
		$CVEQ		GET EQT ADDRESS
		$LIST		WAIT LIST FOR PROGS USING DEV.

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
UP	$IOUP	$INER	ERRORS
		$CVEQ	GETS EQT ADDRESS
		$SCD3	RE-SCHEDULE WAITING PROGS
		$LIST	PUT IN SCED. LIST
		XUPIO	UP LU'S ASSOCIATED WITH THIS EQT
		$XXUP	LINK AN I/O QUEUE ONTO THIS EQT
		LINK	LINK AN I/O REQUEST ON THIS EQT
		DRIVR	INITIATE WAITING I/O
		$SYMG	DIAG. MESSAGE IF STILL N/A
LS	M0960	$INER	ERRORS
LG	M0970	$CREL	RELEASE CURRENT TRACKS
		$DREQ	ALLOCATE NEW TRACKS
TI	M0750	$TIMV	GETS TIME VALUES
		$CNV1	FORMATS INTO OUTPUT MESSAGE
BR	M0725		DOES ALL THE WORK
AB	M0950	$ABRT	SOFT ABORT, ENTERS OF OR BR
		$TREM	REMOVE FROM TIME LIST, REFER TO OF OR BR PROCESSING
		$LIST	SET IN DOMANT LIST
RU	M0408	$SZIT	CHECK PROGRAM VS. MAX PARTITION SIZE.
		$LIST	SCHEDULE PROGRAM
BL	BLIM	$CNV1	FORMAT MESSAGE

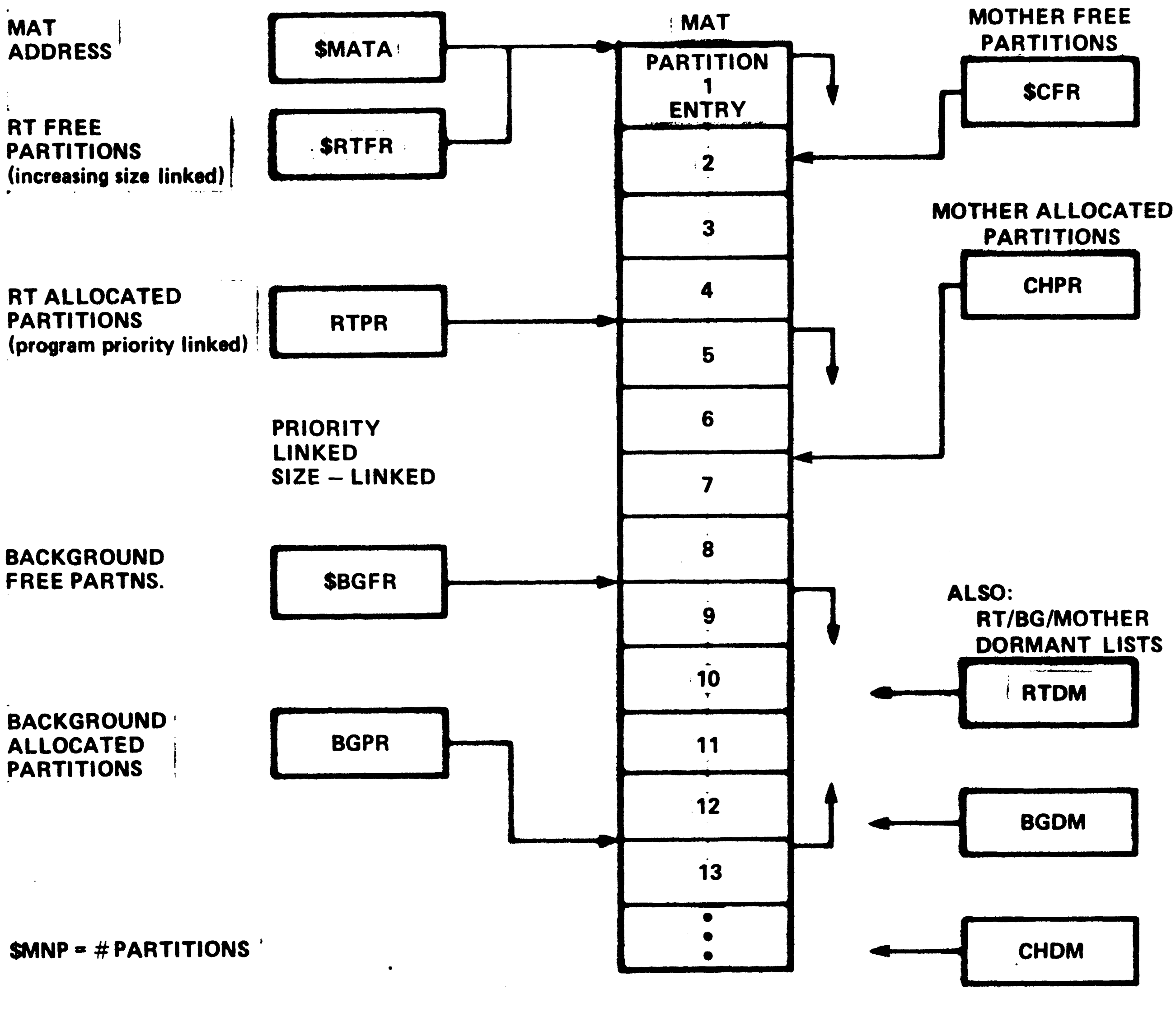
OPERATOR COMMAND PROCESSORS (cont'd)

COMMAND	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
LU	M0850	$LUPR	PROCESSED IN OCMD4 MODULE
EQ	M0900	$EQST	PROCESSED IN OCMD4 MODULE
TO	M0990	$CHTO	PROCESSED IN OCMD4 MODULE
SZ	SIZE	SZCHK	CHECK SIZE CHANGE GET
		$SZIT	GET PROGRAM SIZE PARAMETERS
AS	ASIGN	SZCHK	VERIFY ASSIGNMENT
		$SZIT	GET PROGRAM SIZE PARAMETERS
UR	URESV		DOES IT ALL

PROGRAM DISPATCHING

PARTITION ASSIGNMENT

MEMORY ALLOCATION TABLE (MAT)



(NOTE: listheads are kept in Table Area 2 and DISPM)

- last entry points to start of allocated list
- priority linked

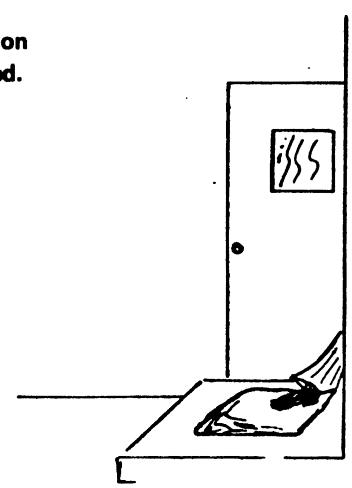
MAT ENTRY

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Linkage pointer to next entry (–1 if undefined partition)															
1	Priority of current resident program															
2	Current resident's ID-segment address															
3	M		D				Beginning page of partition									
4	R	C					Number of pages in partition (–1)									
5	RT													S		
6	Subpartition Link Word (SLW)															

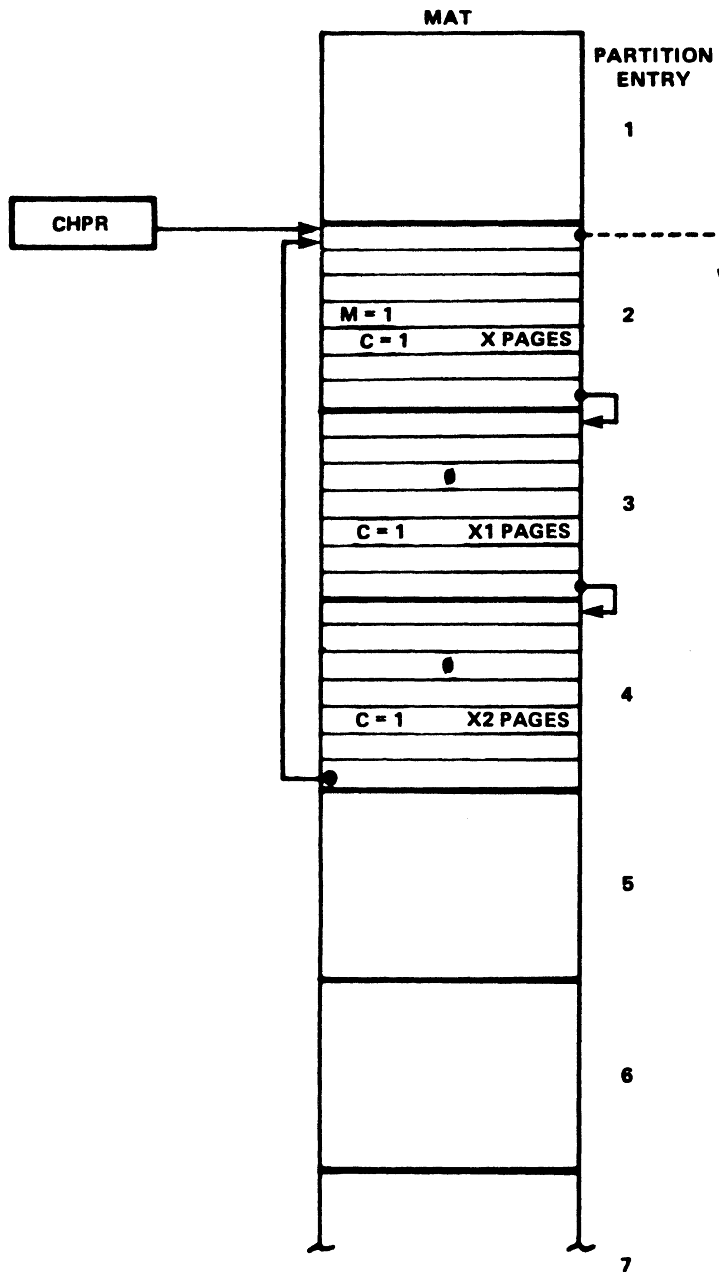
D = Resident is dormant — save resources, serially reusable, or operator suspended
M = Mother partition
R = Partition is reserved
C = Partition is part of a chained mother partion
RT = REAL TIME PARTITION
S = Program's dispatching status
- 0 - Read in progress
- 1 - Program is resident
- 2 - SWAP out or segment load in progress
- 3 - SWAP out complete but program still resident
- 4 - Subpartition swap-out started for mother partition
- 5 - Subpartition swap-out completed. Mother cleared.

"NORMAL" sequence: 2,3,0,1

SLW
- = 0 - Partition not a subpartition
- = Next subpartition
- = Mother partition if partition is last subpartition

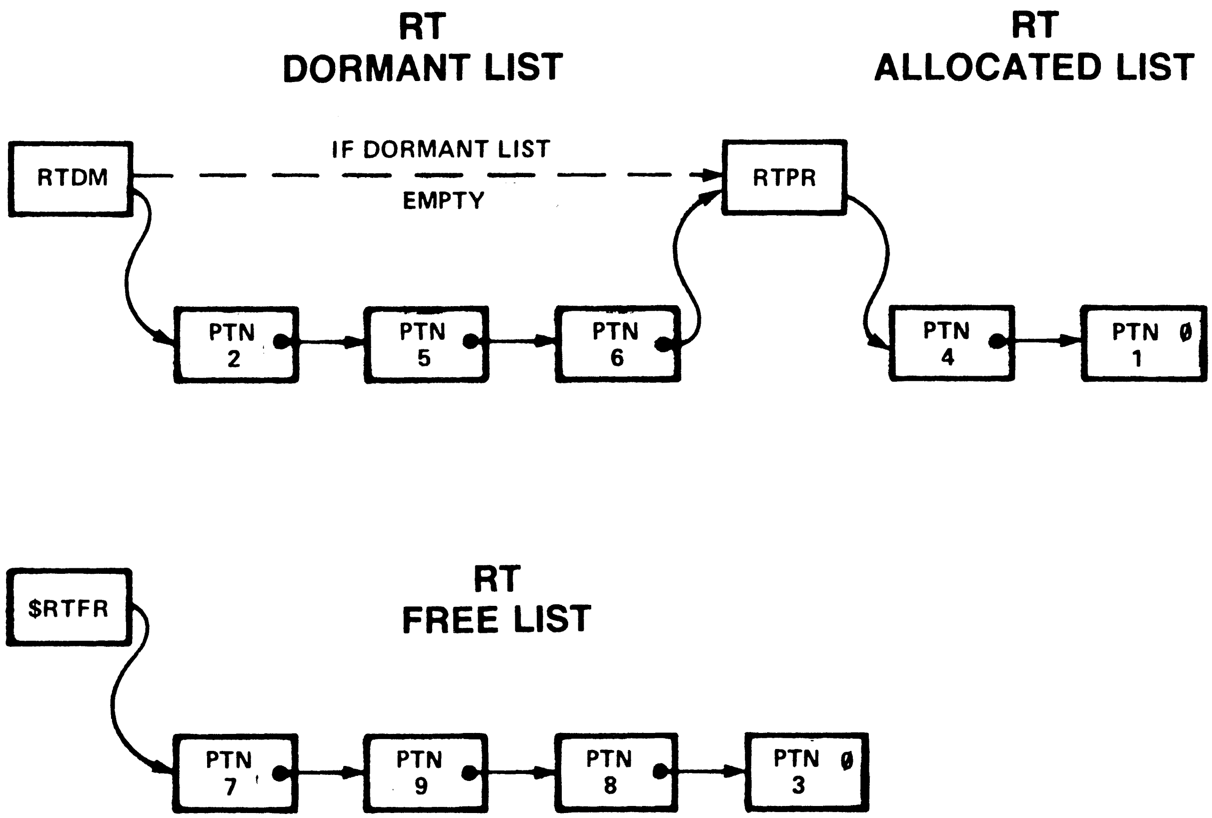


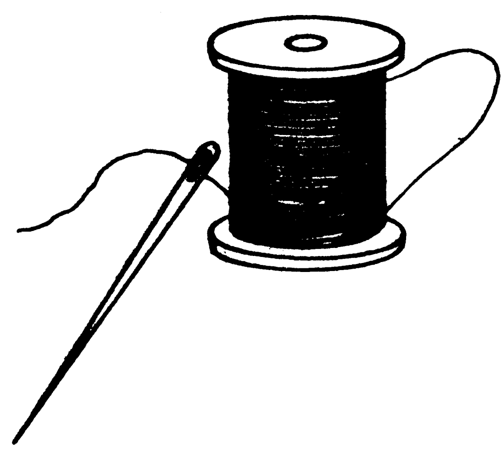
SAMPLE MOTHER/SUBPARTITION LIST THREADING IN MAT



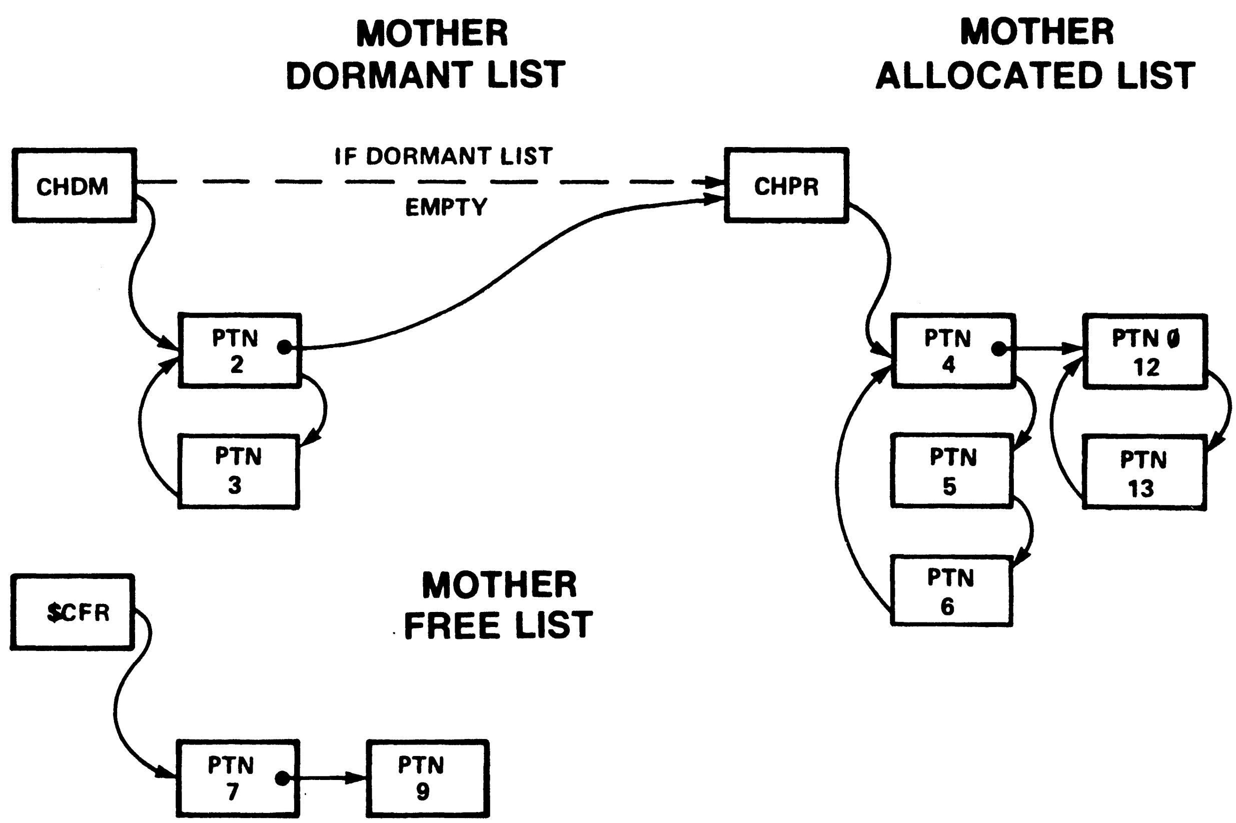
NOTE: X > X1 + X2

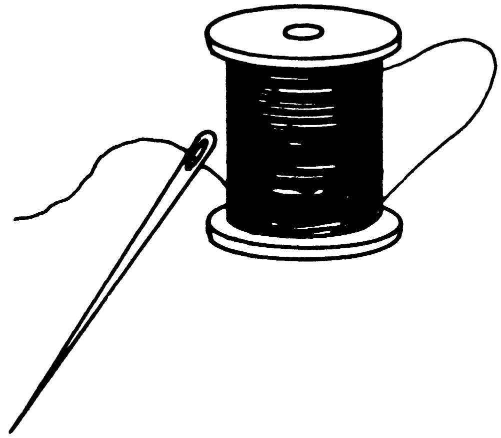
SAMPLE LIST THREADING IN MAT





SAMPLE LIST THREADING IN MAT





MAT EXAMPLE

WORD	LOCATION	VALUE(8)	
1	24317	23500	$MATA

WORD	LOCATION	VALUE(8)	
1	24336	23534	$BGFR
2	24337	0	$RTFR

WORD	LOCATION	VALUE(8)	
1	24350	23525	$CFR
2	24351	17	$MNP

WORD	LOCATION	VALUE(8)	
1	27274	23552	BGDM
2	27275	23500	BGPR
3	27276	24336	
4	27277	27275	
5	27300	27274	
6	27301	27302	RTDM
7	27302	0	RTPR
8	27303	24350	
9	27304	27307	
10	27305	27306	
11	27306	27307	CHDM
12	27307	0	CHPR

WORD	LOCATION	VALUE(8)
1	23500	23543
2	23501	77777
3	23502	17612
4	23503	55
5	23504	1
6	23505	1
7	23506	0
8	23507	0
9	23510	1
10	23511	17551
11	23512	57
12	23513	4
13	23514	1
14	23515	0
15	23516	23507
16	23517	132
17	23520	17714
18	23521	64
19	23522	12
20	23523	1
21	23524	0
22	23525	0
23	23526	62
24	23527	0
25	23530	100077
26	23531	100
27	23532	1
28	23533	23534
29	23534	23561
30	23535	62
31	23536	0
32	23537	77
33	23540	6
34	23541	1
35	23542	23543
36	23543	23516
37	23544	132
38	23545	17141
39	23546	106
40	23547	6
41	23550	1
42	23551	23552
43	23552	27275
44	23553	24
45	23554	17202
46	23555	20115
47	23556	25
48	23557	1
49	23560	23561
50	23561	0
51	23562	0
52	23563	0
53	23564	143
54	23565	34
55	23566	0
56	23567	23525
57	23570	177777
58	23571	0
59	23572	0
60	23573	0
61	23574	0
62	23575	0
63	23576	0
64	23577	177777
65	23600	0
66	23601	0
67	23602	0
68	23603	0
69	23604	0
70	23605	0
71	23606	177777
72	23607	0
73	23610	0

MAT EXAMPLE
(EMA PROGRAM RUNNING)

WORD	LOCATION	VALUE(8)	
1	24317	23500	$MATA

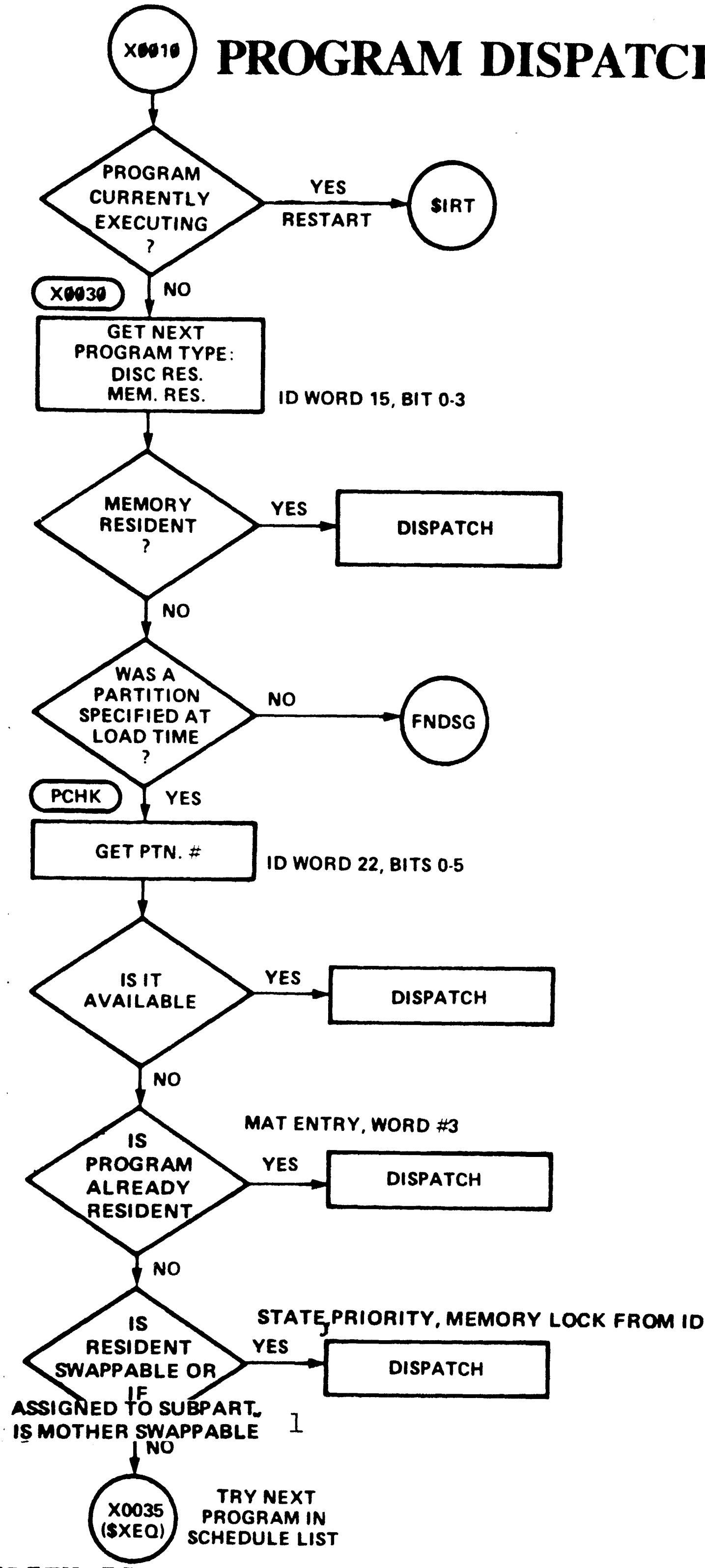
WORD	LOCATION	VALUE(8)	
1	24336	0	$BGFR
2	24337	0	$RTFR

WORD	LOCATION	VALUE(8)	
1	24350	0	$CRF
2	24351	17	$MNP

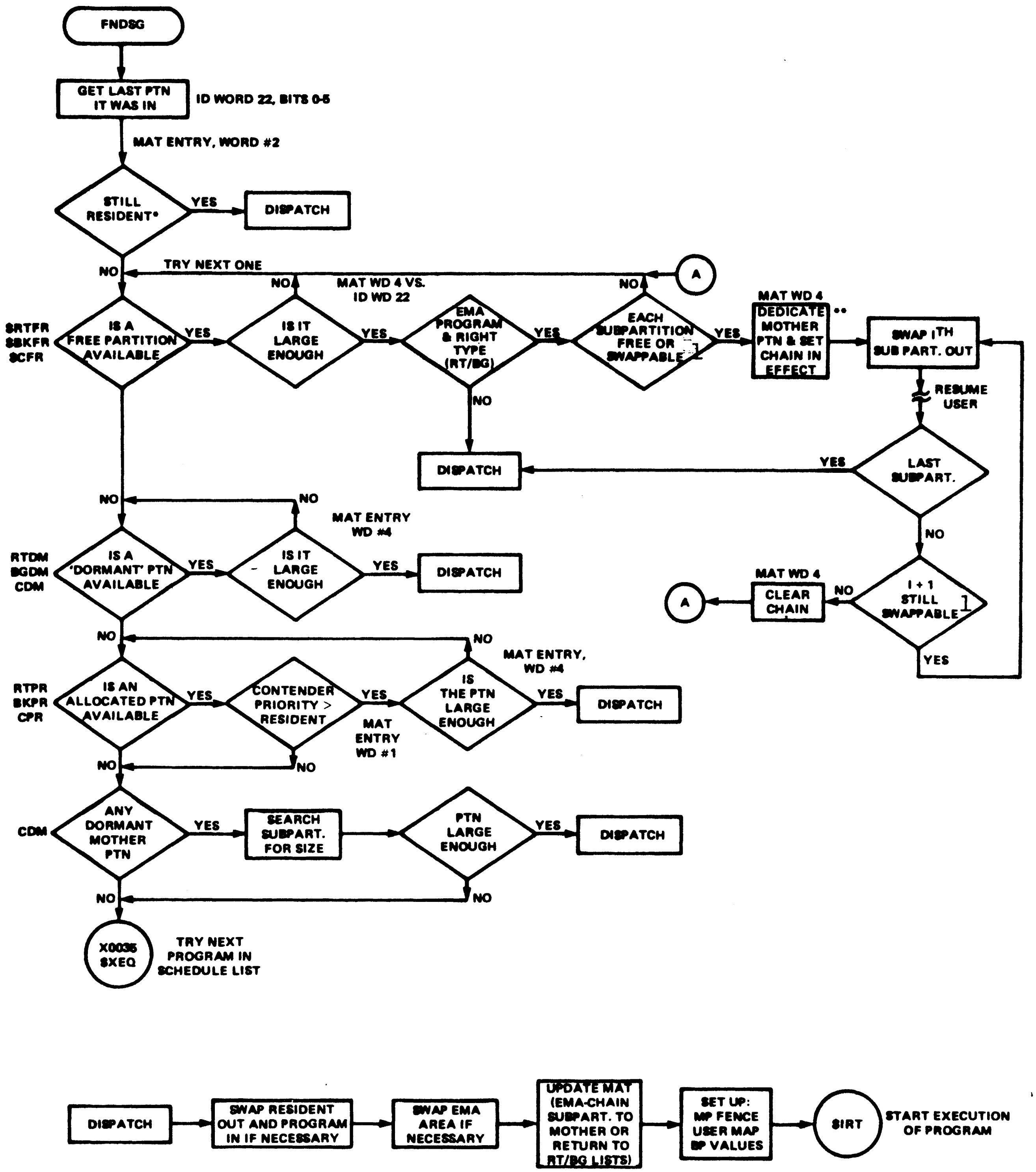
WORD	LOCATION	VALUE(8)	
1	27274	27275	BGDM
2	27275	23500	BGPR
3	27276	24336	
4	27277	27275	
5	27300	27274	
6	27301	27302	RTDM
7	27302	0	RTPR
8	27303	24350	
9	27304	27307	
10	27305	27306	
11	27306	27307	CHDM
12	27307	23525	CHPR

WORD	LOCATION	VALUE(8)
1	23500	23516
2	23501	77777
3	23502	17612
4	23503	55
5	23504	1
6	23505	1
7	23506	0
8	23507	0
9	23510	1
10	23511	17551
11	23512	57
12	23513	4
13	23514	1
14	23515	0
15	23516	23567
16	23517	132
17	23520	17714
18	23521	64
19	23522	12
20	23523	1
21	23524	0
22	23525	0
23	23526	62
24	23527	20016
25	23530	100077
26	23531	40100
27	23532	1
28	23533	23534
29	23534	23567
30	23535	132
31	23536	0
32	23537	77
33	23540	40006
34	23541	0
35	23542	23543
36	23543	27276
37	23544	24
38	23545	0
39	23546	20106
40	23547	40006
41	23550	0
42	23551	23552
43	23552	23561
44	23553	120
45	23554	0
46	23555	115
47	23556	40025
48	23557	0
49	23560	23561
50	23561	0
51	23562	0
52	23563	0
53	23564	143
54	23565	40034
55	23566	0
56	23567	23525
57	23570	177777
58	23571	0
59	23572	0
60	23573	0
61	23574	0
62	23575	0
63	23576	0
64	23577	177777
65	23600	0
66	23601	0
67	23602	0
68	23603	0
69	23604	0
70	23605	0
71	23606	177777
72	23607	0

PROGRAM DISPATCHING



1. SWAPPABILITY IS DETERMINED BY STATE, PRIORITY, MEMORY LOCK FROM ID, &SWAP DELAY.



*MUST BE:

- SERIALLY REUSABLE OR
- SAVING RESOURCES OR
- OPERATOR SUSPEND

**A PROGRAM IN A SUBPART. WILL NOT BE RESUMED IF MAT "C" BIT IS SET

I/O
PROCESSING

GENERAL OPERATION

- I/O operations are performed concurrently with program computation.
- I/O transfer can be broken into three phases:
  - Initiation
  - Continuation
  - Completion
- User programs are involved only in the initiation and completion phases.
- I/O request types include:
  - User Normal Operation
  - User Automatic Buffering
  - User Class I/O
  - System Requests
- I/O drivers operate under control of IOC and $CIC system modules.
- I/O drivers are composed of two sections:
  - Initiation
  - Continuation/Completion
- $XSIO handles all I/O requests from the system modules.
- EQTs associate drivers with devices.

I/O INITIATION

- User program makes an EXEC call to initiate I/O transfers.
- Request specifies LU, buffer location, buffer length, and request type (read, write, or control).
- IOC calls initiation section of driver to start the data transfer.
- User program will be suspended or restarted depending on the I/O request type.

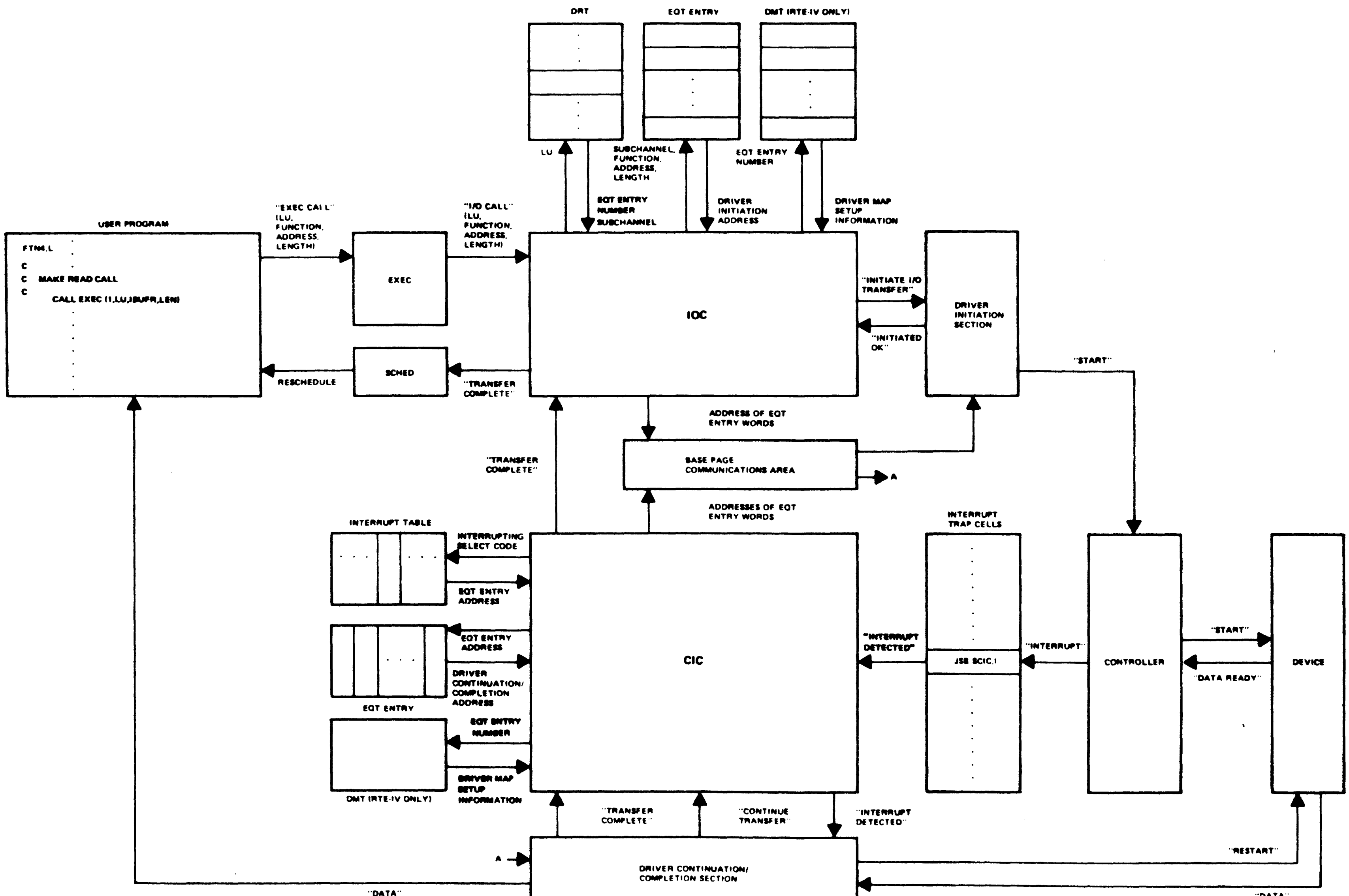
I/O CONTINUATION

- Device will interrupt after completion of first data transfer
- CIC calls the continuation/completion section of the driver.
- The driver completes the first transfer and starts the next data transfer.
- The currently executing program is restarted.

I/O COMPLETION

- After the last interrupt the driver notifies $CIC that the I/O transfer has completed.
- IOCOM is called to terminate the I/O request and initiate the next transfer if any requests are stacked on the EQT.
- The supended user program is then placed on the scheduled list.

UNBUFFERED EXEC READ REQUEST FLOW



I/O REQUEST STACKING

Requests are stacked in lists fo each device according to priority. The formats of the four types of requests as they appear in the I/O lists are:

1. USER (NORMAL OPERATION)

The parameters from the request are stored in the temporary area of the program ID segment. The link word of the segment is used to link into the I/O list.

```
WORD          CONTENTS
----          --------

 1       < LINKAGE WORD                     >
 2       <T, CONTROL INFO, REQUEST CODE     >
 3       <L, BUFFER ADDRESS(L=1 IF IN SAM)>
 4       <BUFFER LENGTH                     >
 5       <DISC TRACK ADDR OR ZERO           >
 6       <DISC SECTOR ADDR OR ZERO          >
 7       <PROGRAM PRIORITY                  >
 .       -REMAINDER OF ID SEGMENT           .
```

2. USER (AUTOMATIC OUTPUT BUFFERING)

Requests of this type are constructed in system available memory.

```
WORD          CONTENTS
----          --------

 1       <LINKAGE WORD                  >
 2       <T, CONTROL INFO, REQUEST CODE>
 3       <PRIORITY OF REQUESTOR         >
 4       <TOTAL BLOCK LENGTH WORDS      >
 5       <USER BUFFER LENGTH            >
 6       <OPTIONAL PARAMTER 1           >
 7       <OPTIONAL PARAMETER 2          >
 8       <WORD 1 OF USER BUFFER         >
 .          .       .        .
 .          .       .        .
N+7      <WORD N OF USER BUFFER         >
```

3. USER (CLASS INPUT/OUTPUT)

Requests of this type are constructed in system available memory.

```
WORD           CONTENTS
----           --------

 1       <LINKAGE WORD                    >
 2       <T, CONTROL INFO, REQUEST CODE>
 3       <PRIORITY OF REQUESTOR           >  (CHANGED TO STATUS AT COMP.)
 4       <TOTAL BLOCK LENGTH WORDS        >
 5       <CLASS ID WORD                   >
 6       <USER BUFFER LENGTH              >  (CHANGED TO TLOG AT COMP.)
 7       <OPTIONAL PARAMETER 1            >
 8       <OPTIONAL PARAMETER 2            >
 9       <WORD 1 OF USER BUFFER           >
 .          .        .         .
 .          .        .         .
N+8      <WORD N OF USER BUFFER           >
```

4. SYSTEM REQUEST

The system request is linked into the I/O list by using word 4 of the call as a link word. A system request asumes the priority level of zero (highest priority).

```
WORD           CONTENTS
----           --------

 1       <    JSB $XSIO                   >
 2       <  LOGICAL UNIT #                >
 3       <COMPLETION ROUTINE ADDR         >
 4       <    LINKAGE WORD                >
 5       <T, CONTROL INFO, REQUEST CODE>
 6       <BUFFER ADDR OR DISC CNTL        >
 7       <BUFFER LENGTH OR PRIORITY       >
 8       <MAP WORD WHERE:                 >
```

BIT 15	BITS 0-14	SYSTEM MAP
0	0	SYSTEM MAP
0	ID SEG.	USER MAP
1	0	CURRENT USER MAP
1	ID SEG.	MODIFIED USER MAP (I.E. EMA SWAP)

<T> FIELD:

The <T> field (bits 15-14 in control word) identifies the request type as:

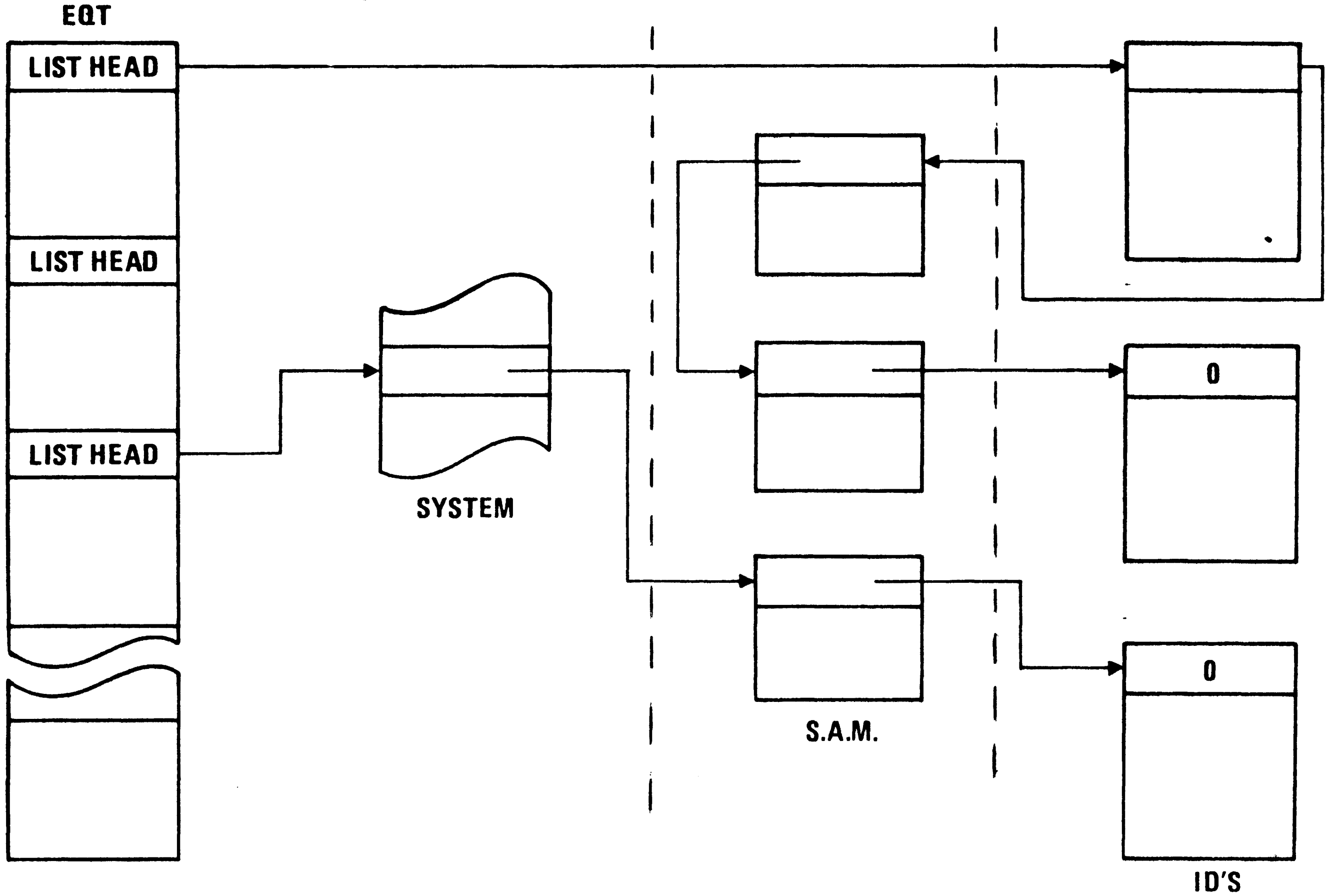
00 USER (NORMAL OPERATION)
01 USER (AUTOMATIC BUFFERING)
10 SYSTEM
11 CLASS I/O

<REQUEST CODE> FIELD:
----

1 READ
2 WRITE
3 CONTROL

<CONTROL INFO> FIELD INCLUDES SUBSCHANNEL NUMBER

I/O REQUEST LINKING



SYSTEM I/O REQUEST PROCESSOR
-$XSIO-

- $XSIO allows RTE modules to call for I/O operations without the overhead of a user I/O request.
- Error checking is not performed
- System request has priority 0
- System disc call can specify a series of transfers
- A completion routine can be specified

EXEC CALL PROCESSING

```
CALL EXEC(2,2,IBUF,LEN,50,0)
```

```
     JSB EXEC
     DEF RET      Return address
     DEF N2       Request code
     DEF N2       Disk LU
     DEF IBUF     Output buffer
     DEF LEN      Buffer length
     DEF N50      Track
     DEF ZERO     Sector
RET  EQU *
```

ENVIRONMENT

1. Calling program previously allocated track 50.
2. Disc (unbuffered) has no other requests in progress or stacked.
3. Disc is a 7905 using DVR32 on select code 11.

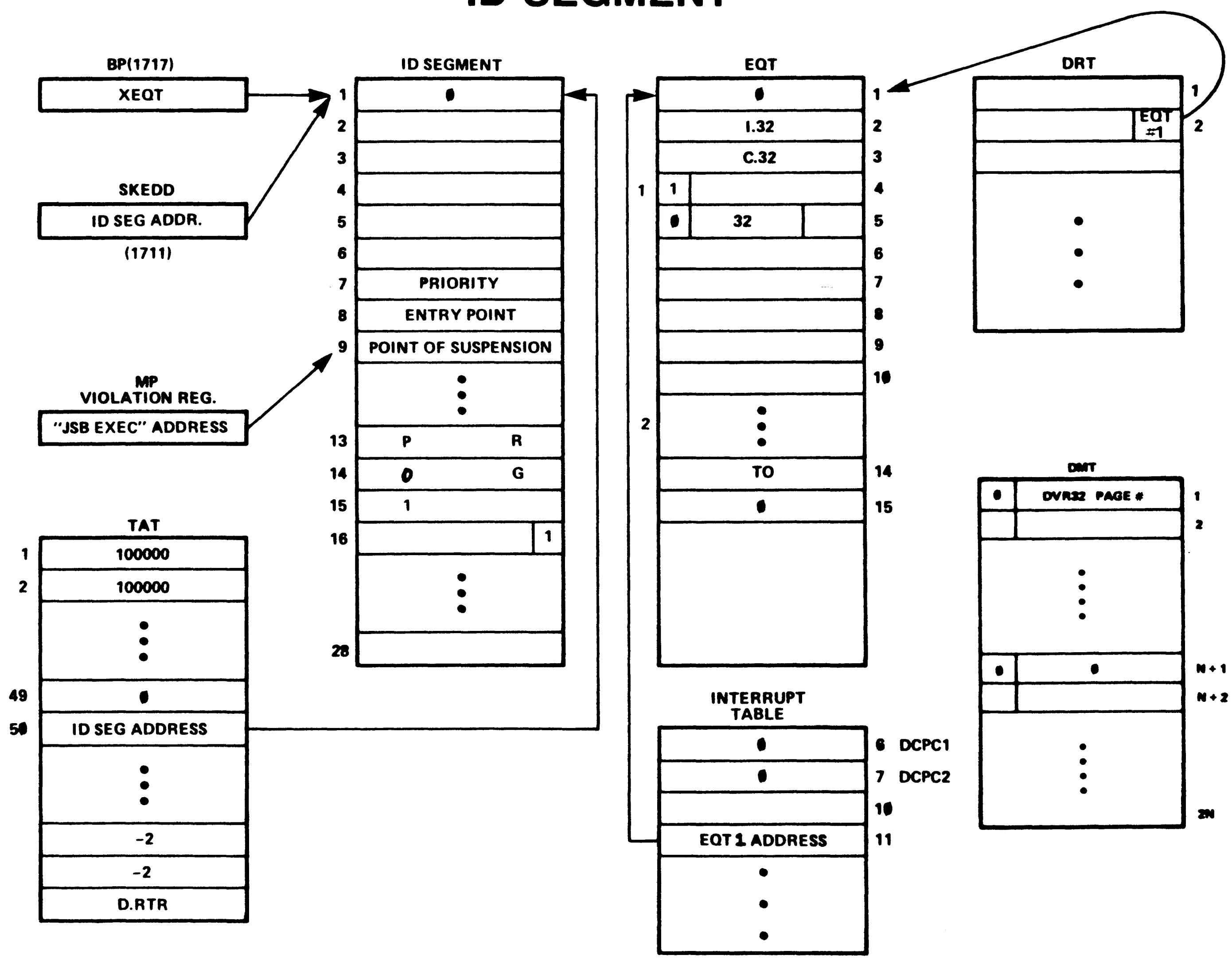
EXEC 2 PROCESSING

EXECUTION	SUBROUTINE CALLS	NOTES
MEMORY PROTECT		Caused by JSB EXEC in user program
TRAP CELL 5(JSB $CIC,I)		MP violation generates an interrupt on select code 5.
$CIC($TB1)		Enable system map and jump to $CIC ($CIC0).
$CIC0(RTIOC)		Save machine state, turn off interrupts, etc. Detect MP violation (CIR=5).
$RQST(EXEC)		Save violation address in program's ID segment.
	DISCUSS ID SEGMENT	
		Detect MP error and not DMS violation
		Check that violating instruction was a JSB or JSB,I.
		Verify that destination address was "EXEC".
R0($RQST)		Check return address and number of parameters in request. (1<Pn<9) Store addresses of request parameters in BP. Verify request code and that address of each parameter to be used for storage is above MP fence. Jump to the request processor.
$IORQ(RTIOC)		Verify LU number. Get EQT entry number from DRT.
	DISCUSS DRT	

EXECUTION	SUBROUTINE CALLS	NOTES
	$CVEQ STADV	Transfer EQT entry addresses into BP. Ensure that EQT entry and LU are both up. If not suspend the program (state 3) by calling $LIST.
L.01($IORQ)		Check that disc request has 5 request parameters. Verify track and sector numbers. Check TAT to insure that the user program owns track 50.
	DISCUSS TAT	
L.10($IORQ)		Setup ID segment words 2 thru 6: 2 - T, control, code 3 - Buffer address 4 - Buffer length 5 - Track 6 - Sector
	$LIST	Put the program in I/O suspend state (state 2)
L.13($IORQ)	LINK	Link ID segment into the EQT entry.
	DRIVR	Assign a DCPC channel to the device EQT.
	DRVMP	Set up appropriate map for driver (DVR32). In this example the current user map will be modified to include DVR32's driver partition. In general the system map is used when: - driver is in SDA and does own mapping - I/O request is a buffered, class, or system request (driver ptn. must be mapped.) User map is used when: - driver is in SDA and program is type 3 - I/O request is unbuffered (driver ptn. must be mapped) Transfer request parameter into EQT entry (words 4-10, and 15).
	DISCUSS DMT	

EXECUTION	SUBROUTINE CALLS	NOTES
	I.32(DVR32)	Enter driver initiator section under system or user map. Start DCPC data transfer.
		DISCUSS TRACK MAP TABLES
	DRIVR	Set EQT entry availability (AV) to busy (2).
		DISCUSS EQT
$XEQ		Schedule next program.
DCPC CHANNEL		DCPC moves words between user program's buffer and disc track 50.
DEVICE COMPLETION INTERRUPT		
$CIC(RTIOC)		Save machine state, etc.
	C.32(DVR32)	Detect end of data transfer.
IOCOM or $CON1 (RTIOC)		Return DCPC channel and clear TO. Unlink request from EQT entry.
L.51(IOCOM)	$LIST	Place program into schedule list. Set EQT entry availability (AV) to 0 (available).
L.68(IOCOM)	DRIVR	If EQT entry has I/O requests stacked initiate next request.
IOCX		Assign available DCPC channel. If more than one device is waiting for a channel, the order of priority for assignment is the order of the Positions in the Equipment Table. There are two exceptions to this scheme: 1. If the first entry in the EQT is waiting for a DCPC, the channel is assigned to that device, which is assumed to be the system disc. 2. If the first entry encountered (other than entry #1) just released a DCPC channel, then the next lower priority device waiting for DMA is used. This allows for a "switching" operation in the allocation of a DMA channel.
$XEQ(DISPM)		Dispatch the next program.

CALL EXEC 2 TABLES/LISTS
(INITIAL CONDITIONS)
ID SEGMENT



ASSUME THE DISC IS BUSY
(A DISC I/O REQUEST IS IN PROGRESS)

In this case: EQT entry 1 word 5 (AV field) would be 2(busy). Word 1 would be the link address of the request being processed.

EXECUTION	SUBROUTINE CALLS	NOTES
		Flow of the EXEC 2 call remains the same as previously discussed (pg. 10-2 thru 10-3) until:
L.13 ($IORQ) $XEQ	LINK	Link ID segment into EQT entry. Since the device is busy (EQT entry 1 word #0), schedule the next program.

Once the current request is completed and $CIC has called IOCOM:

L.68 (IOCOM)	DRIVR	DRIVR is called since EQT entry 1, word 1 #0.

From this point on the flow is the same as the previous discussion following the first call to DRIVR from L.13 (pg. 10-3).

FLOW OF EXEC 2 REQUEST
FOR A BUFFERED DEVICE
(NON DISC DEVICE)

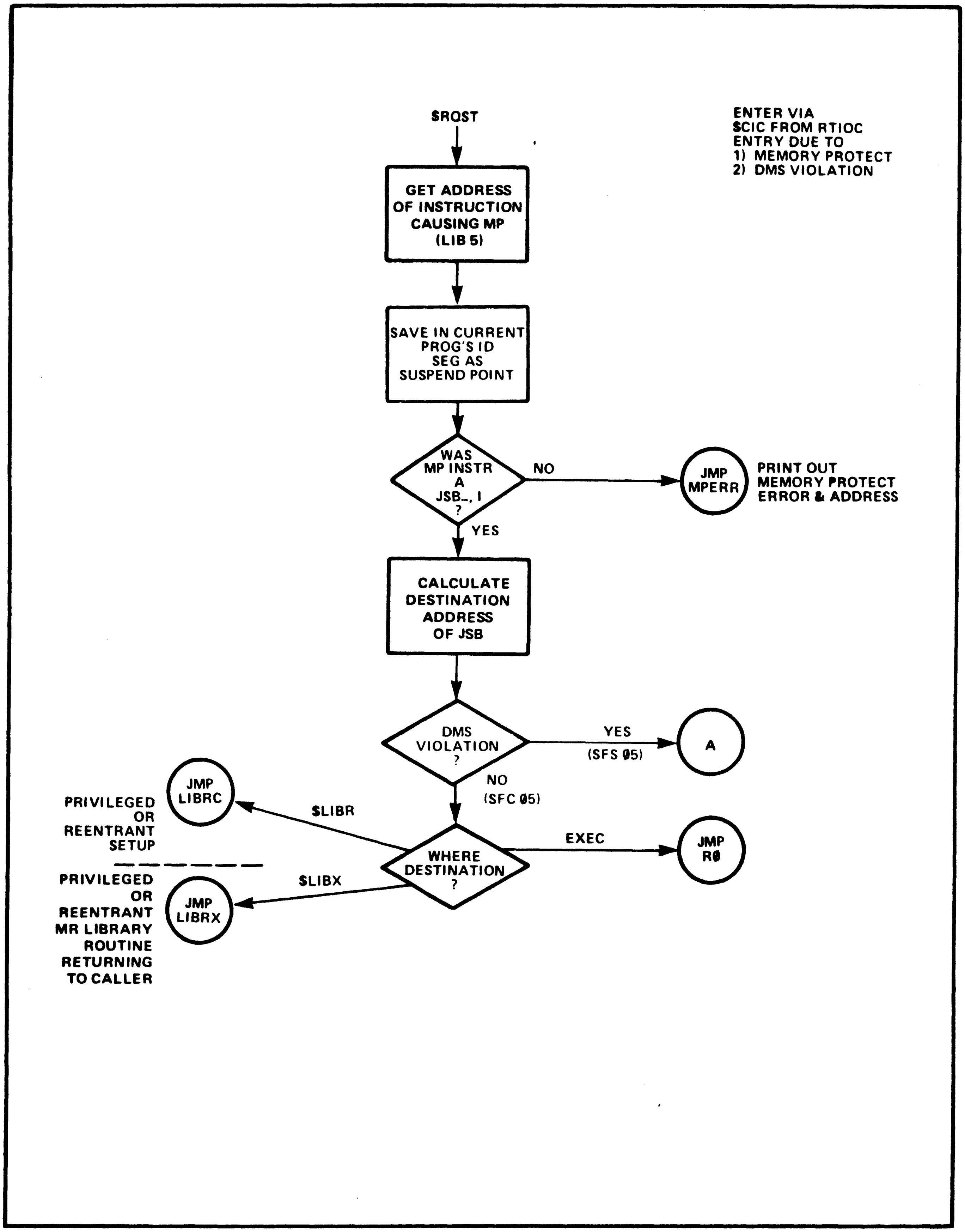
EXECUTION	SUBROUTINE CALL	NOTES
Flow is the same as the disc request example until (pg. 10-3):		
L.01($IORQ(RTIOC))		Detect that it is not a disc request (DVR30,31,32, or 33).
L.02($IORQ)		Check for LU lock.
L.027($IORQ)		Calculate SAM buffer size needed for this request (ILEN+7)
	QCHK	Check buffer limit
	$ALC	Request SAM for buffer.
L.06($IORQ)		Store request parameter into SAM buffer: 2 T(=1), control, code 3 Priority of requestor 4 Block length 5 User buffer length 6 Optional parameter 1 7 Optional parameter 2 Move data from user buffer into SAM buffer
L.13($IORQ)	LINK	Link SAM buffer into the EQT entry.
	DFIVR	Assign a DCPC channel if required
	DRVMP	Set up appropriate map for driver. In this case system map would be used and driver's ptn. would be mapped. Transfer request parameters into EQT entry (words 4-10, and 15).

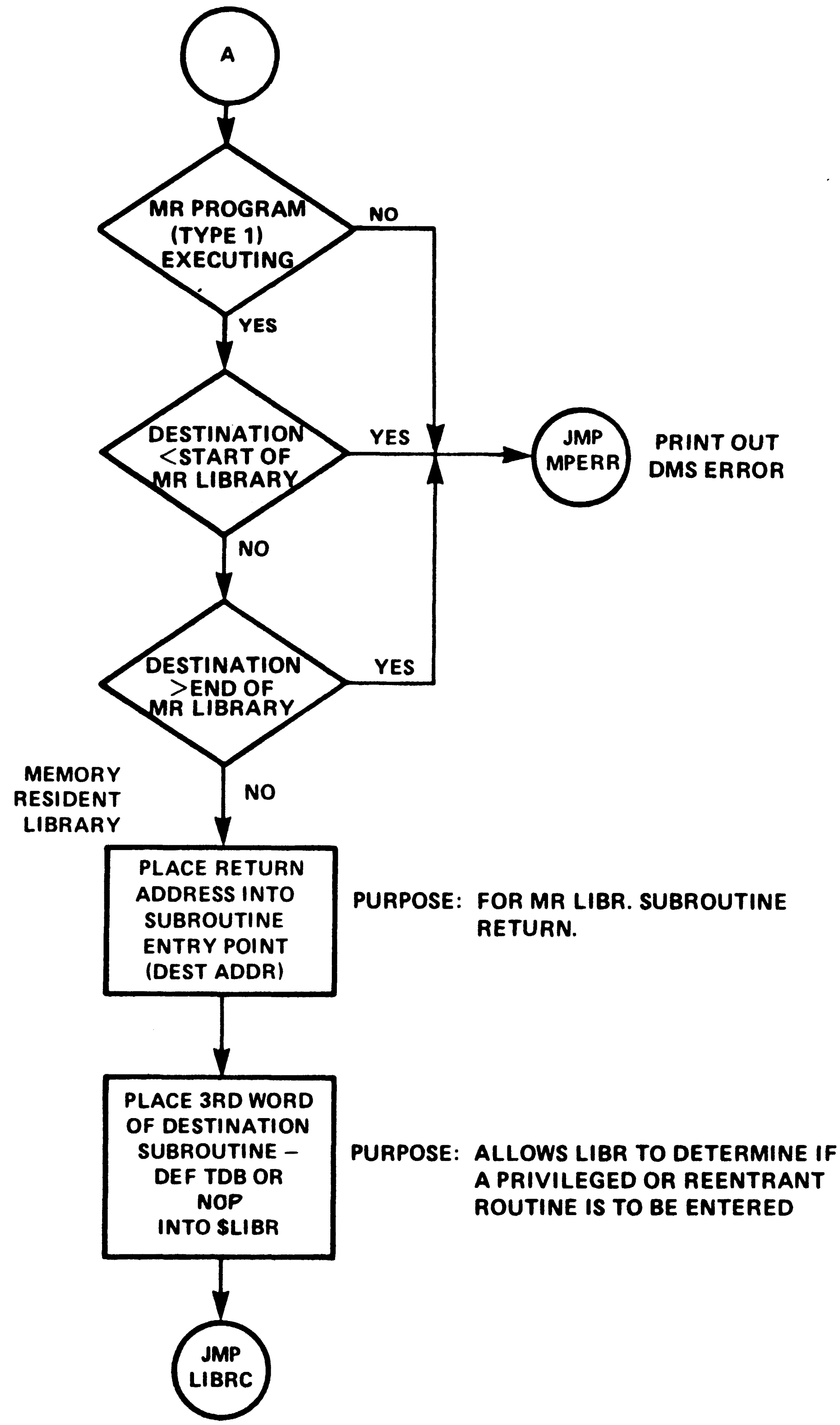
EXECUTION	SUBROUTINE CALLS	NOTES
	Ixnn	Enter driver initiator section under system or user map.
	DRIVR	Set EQT availability to busy (2).
$XEQ		Continue execution of user program.

CONTINUATION INTERRUPTS

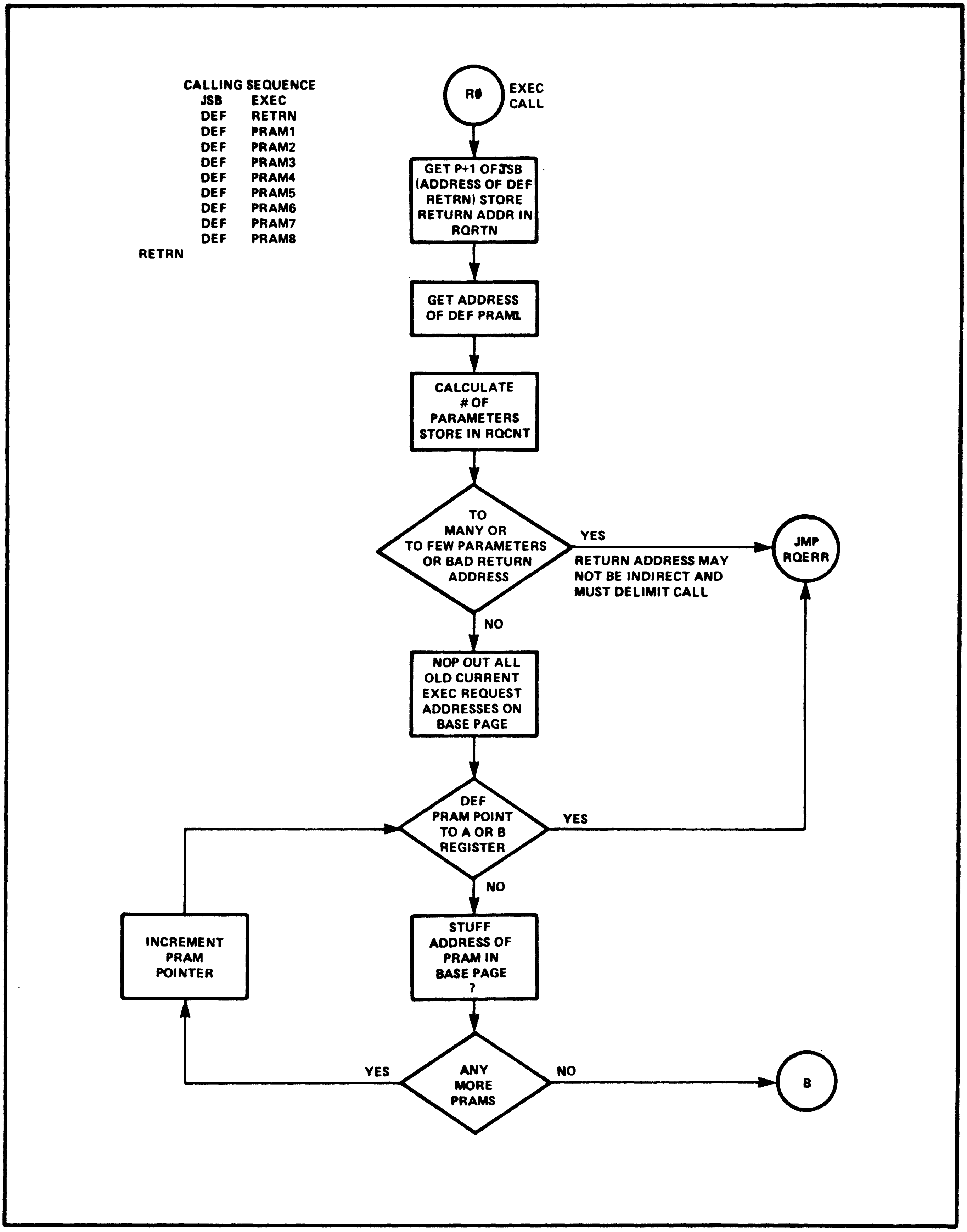
COMPLETION INTERRUPT

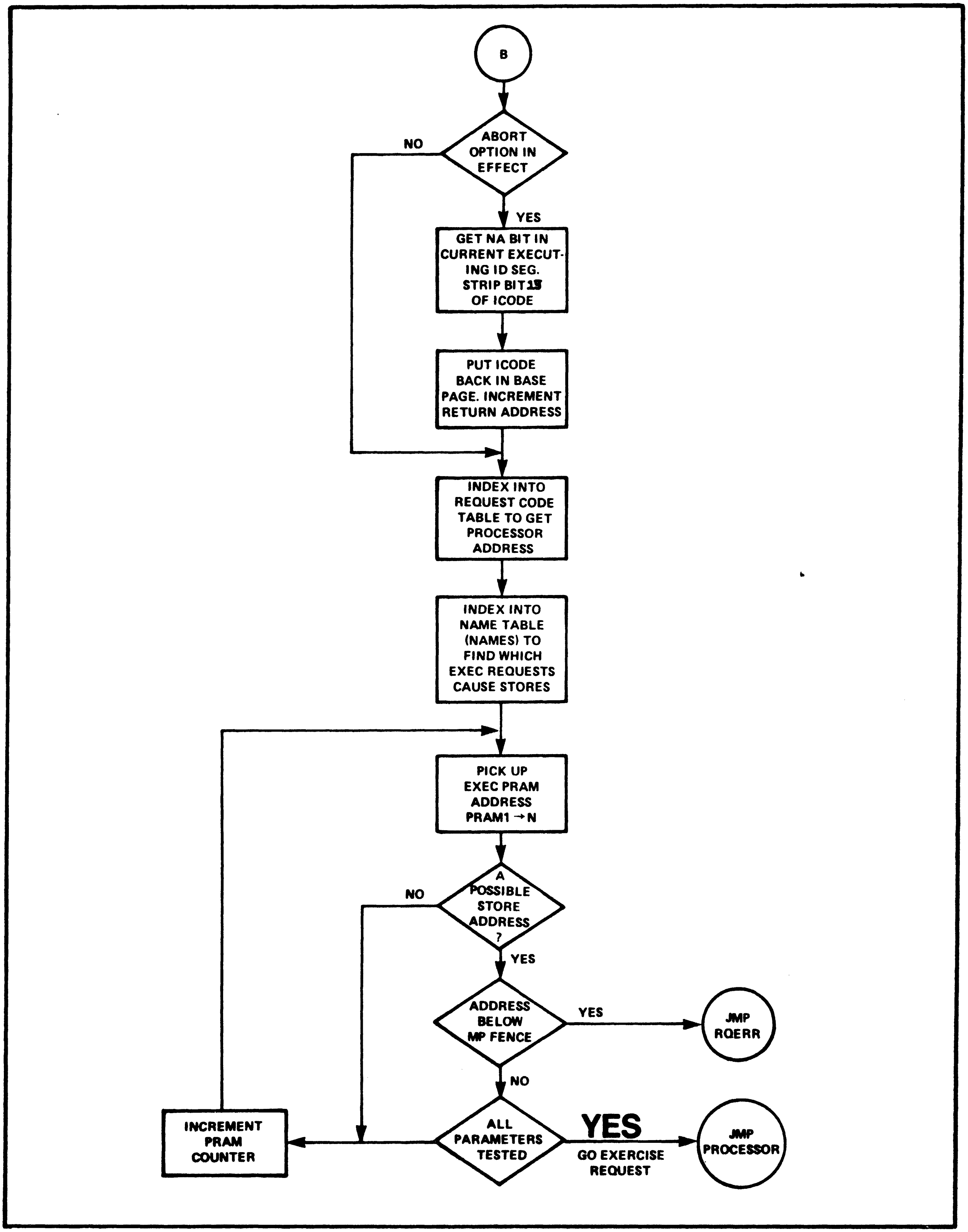
EXECUTION	SUBROUTINE CALLS	NOTES
$CIC(RTIOC)		Save machine state, etc.
	Cxnn	Detect end of data transfer
IOCOM or $CON1		Clear TO and unlink request from EQT entry.
	$RTN	Return buffer to SAM
	$CKLO	Check lower buffer limit and schedule any waiting programs.
L.54(IOCOM)		Set EQT entry to available (AV=0).
L.68(IOCOM)	DRIVR	If EQT entry has I/O requests stacked, initiate next request.
IOCX		Assign DCPC channel if available.
$XEQ		Continue execution of current user program.


$RQST
ENTER VIA
$CIC FROM RTIOC
ENTRY DUE TO
1) MEMORY PROTECT
2) DMS VIOLATION
GET ADDRESS OF INSTRUCTION CAUSING MP (LIB 5)
SAVE IN CURRENT PROG'S ID SEG AS SUSPEND POINT
WAS MP INSTR A JSB_, I ?
NO
JMP MPERR
PRINT OUT MEMORY PROTECT ERROR & ADDRESS
YES
CALCULATE DESTINATION ADDRESS OF JSB
DMS VIOLATION ?
YES
(SFS Ø5)
A
NO
(SFC Ø5)
WHERE DESTINATION ?
EXEC
JMP RØ
$LIBR
JMP LIBRC
PRIVILEGED OR REENTRANT SETUP
$LIBX
JMP LIBRX
PRIVILEGED OR REENTRANT MR LIBRARY ROUTINE RETURNING TO CALLER



NOTE: DMS VIOLATION IS LEGAL IF A MR PROGRAM CALLS A MR LIBRARY PROGRAM (USEABLE ONLY BY MR PROGRAMS). THE PHYSICAL ADDRESS OF THE LIBRARY WILL BE ABOVE THE MP FENCE IF THE MR PROGRAM IS USING COMMON: HOWEVER, THE PAGES CONTAINING THE LIBRARY ARE WRITE PROTECTED.


CALLING SEQUENCE
JSB EXEC
DEF RETRN
DEF PRAM1
DEF PRAM2
DEF PRAM3
DEF PRAM4
DEF PRAM5
DEF PRAM6
DEF PRAM7
DEF PRAM8
RETRN
R0
EXEC CALL
GET P+1 OF JSB (ADDRESS OF DEF RETRN) STORE RETURN ADDR IN RQRTN
GET ADDRESS OF DEF PRAM1
CALCULATE # OF PARAMETERS STORE IN RQCNT
TO MANY OR TO FEW PARAMETERS OR BAD RETURN ADDRESS
YES
RETURN ADDRESS MAY NOT BE INDIRECT AND MUST DELIMIT CALL
JMP RQERR
NO
NOP OUT ALL OLD CURRENT EXEC REQUEST ADDRESSES ON BASE PAGE
DEF PRAM POINT TO A OR B REGISTER
YES
NO
INCREMENT PRAM POINTER
STUFF ADDRESS OF PRAM IN BASE PAGE ?
YES
ANY MORE PRAMS
NO
B


B
ABORT OPTION IN EFFECT
NO
YES
GET NA BIT IN CURRENT EXECUT-ING ID SEG. STRIP BIT 15 OF ICODE
PUT ICODE BACK IN BASE PAGE. INCREMENT RETURN ADDRESS
INDEX INTO REQUEST CODE TABLE TO GET PROCESSOR ADDRESS
INDEX INTO NAME TABLE (NAMES) TO FIND WHICH EXEC REQUESTS CAUSE STORES
PICK UP EXEC PRAM ADDRESS PRAM1 → N
A POSSIBLE STORE ADDRESS ?
NO
YES
ADDRESS BELOW MP FENCE
YES
JMP RQERR
NO
ALL PARAMETERS TESTED
YES
GO EXERCISE REQUEST
JMP PROCESSOR
INCREMENT PRAM COUNTER

EXEC CALL PROCESSORS

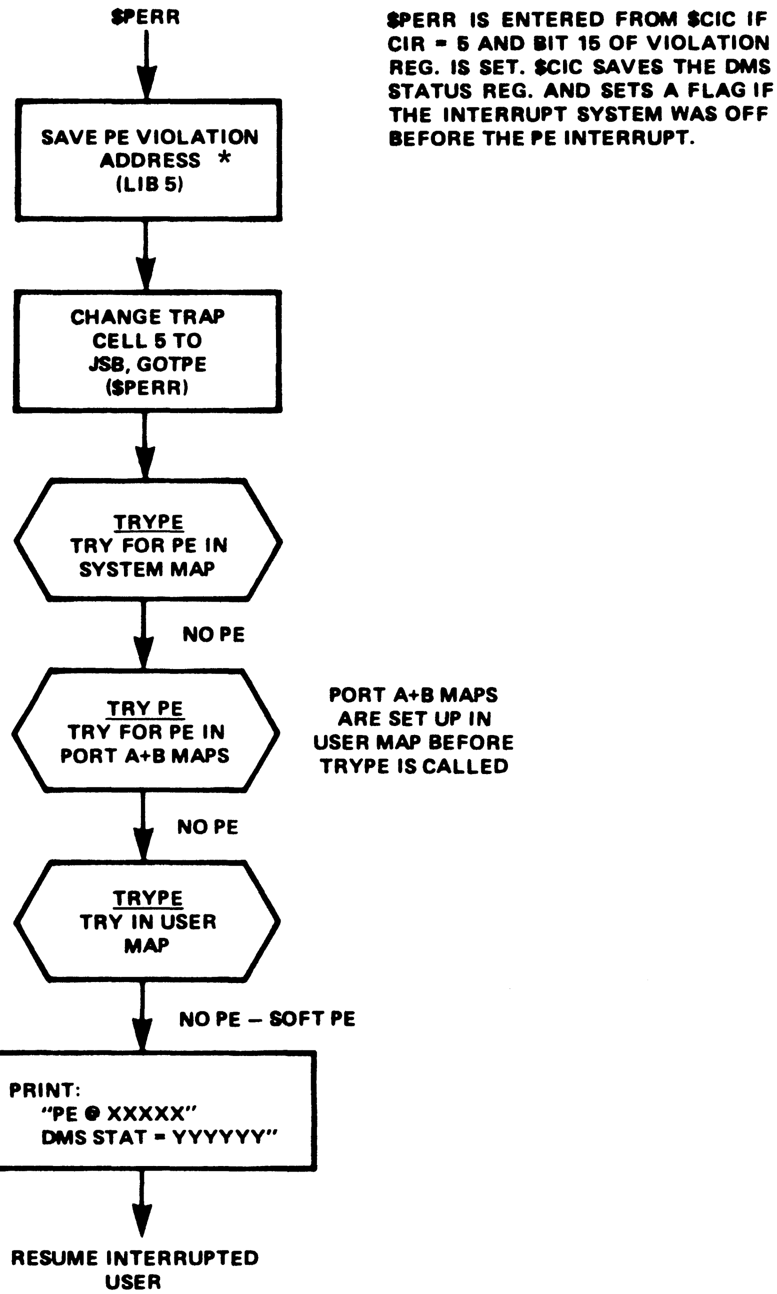
RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
1			
2, 3			
17, 18	$IORQ	$LIST	SUSPEND IF LOCKED
19			NO BUFFER AVAIL.
20			OR SET I/O SUSPEND
		$ALC	ALLOCATE BUFFER FOR OUTPUT
		$REIO	MOVE TDB FOR RE-ENTRANT I/O
		$SYMG	ERROR MESSAGES
		DRIVR	INITIATE I/O
		$XEQ	RETURN
4,15	DISC1	$DREQ	ALLOCATE DISC TRACKS
		$LIST	SUSPEND IF NOT AVAILABLE
		$XEQ	RETURN
5,16	DISC2	$CREL	RELEASE GLOBAL TRACKS
		$SDSK	SCHEDULE DISC-SUSP PROGS
		$XEQ	RETURN
6 (0)	$MPT1	MPT1B	STANDARD TERMINATION
		$LIST	SET DORMANT, SCHEDULE FATHER
		$XEQ	RETURN
(1)		MPID	SAVE RESOURCES TERMINATION
		$WATR	FIND IF ANY WAITING PROGRAMS
		$SCD3	IF SO, SCHEDULE 'EM
		$LIST	SET PROC IN DORMANT LIST,SCHED PO
		$XEQ	RETURN
(2)		SABRT	SOFT ABORT(SEE 'OF,0')
		$TREM	REMOVE FROM TIME LIST
		$LIST	SET PROG IN DORMANT LIST
		$XEQ	RETURN
(3)			SEE 'OF,X'
7	$MPT2	$LIST	SET SUSPENDED
		$XEQ	RETURN
8	$MPT3	$BRED	SET UP TO READ SEGMENT
		$XSIO	SYSTEM I/O REQUEST
		$LIST	SET PROG SUSPENDED
		$CVEQ	I/O ROUTINES...
		DRIVR	
		$LIST	SET PROG I/O SUSPEND
		$XEQ	RETURN

EXEC CALL PROCESSORS (cont'd)

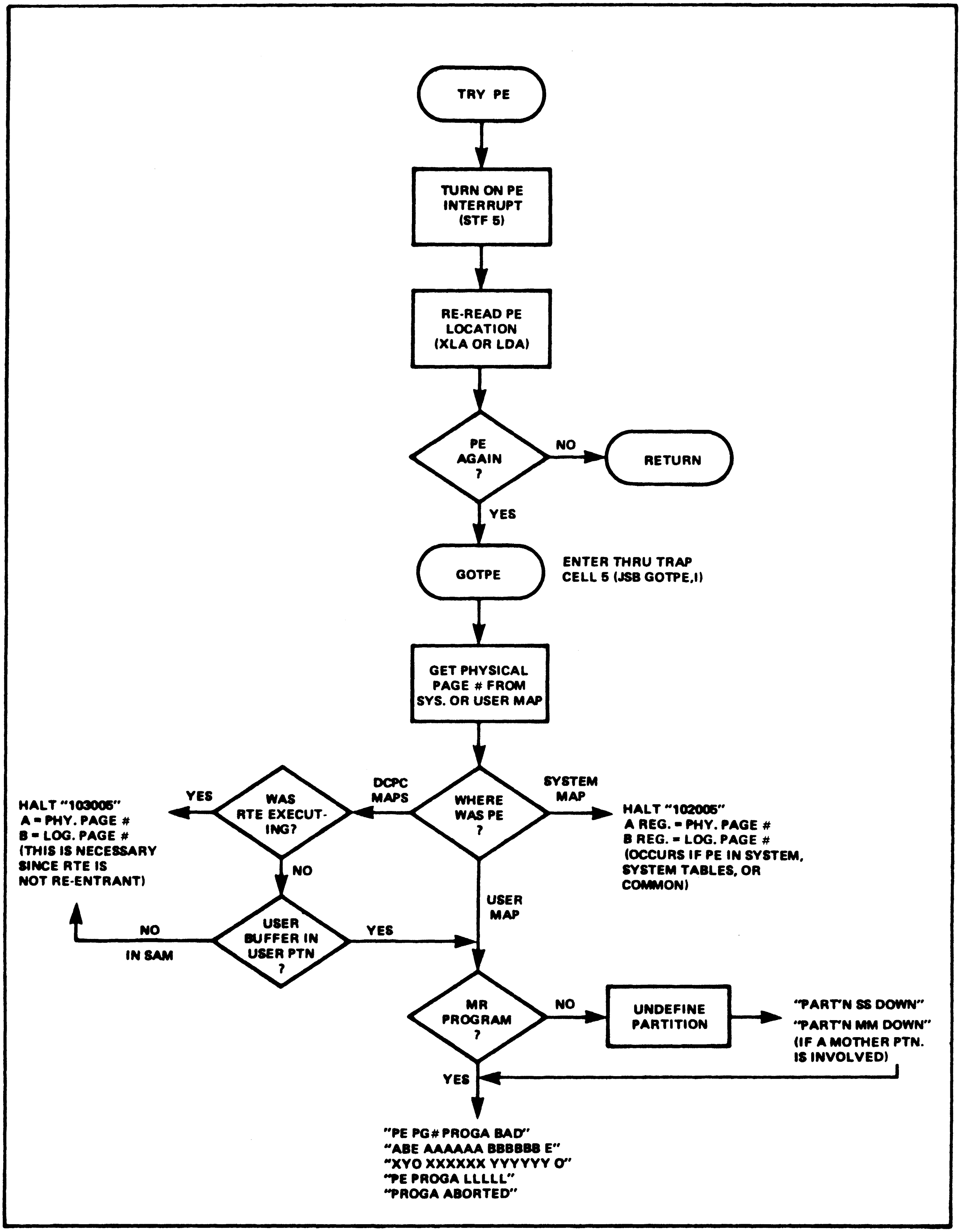
RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
9,23			
	$MPT4	IDCKK	
		$IDNO	GET ID SEG NUMBER
			SET POP POINTER.
		ALCST	SAVE PARAMETER STRING
		$RTST	RETURN SAM USED FOR STRINGS
		$RTN	RETURN MEMORY
		$LIST	SCHEDULE MEMORY WAITERS
		$ALC	ALLOCATE MEMORY.
		$LIST	SCHEDULE 'SCHEULEE'
			PASS 'BATCH' FLAG,.
			PLACE SCHEDULER DORMANT
10,24			
	$MPT5	IDCKK	
		$IDNO	GET ID SEG. NUMBER
			SET POP POINTER
		ALCST	SAVE PARAMETER STRING.
		$RTST	RETURN SAM USED FOR STRINGS
		$RTN	RETURN MEMORY
		$LIST	SCHEDULE MEMORY WAITERS
		$ALC	ALLOCATE MEMORY
		$LIST	SCHEDULE 'SCHEDULEE'
11	$MPT6	$TIME	ACCESS CURRENT TIME
		$TIMV	CONVERT & MOVE TO USER
12	$MPT7	$TIMR	DOES THE WORK
		$LIST	SET DORMANT IF CURRENT PROG
		$TADD	SET IN TIME LIST
		$XEQ	RETURN
13	$IORQ	$CVEQ	GET EQT ADDRESS
		$XEQ	RETURN
22	$MPT8		DOES THE WORK
		$XEQ	RETURN
14(1)	$MPT9		GET (RETRIEVE) STRING
		$STSH	GET ADDRESS OF STRING
		$RTST	RETURN STRING MEMORY
		$RTN	RETURN MEMORY
		$LIST	SCHEDULE MEM.WAIT PROGS.

RQ CODE	PROCESSOR	ENTRY POINTS ACCESSED	REMARKS
(2)		MPT9W	PUT(WRITE)STRING TO FATHER
		ALCST	SAVE PARAMETER STRING.
		$RTST	RETURN STRING MEMORY
		$RTN	RETURN MEMORY
		$LIST	SCHEDULE MEM.WAIT PROGS.
25	$PTST		
		$IDNO	GET ID SEG. OF OCCUPANT
26	PTSIZ		DOES THE WORK

PARITY ERRORS



* ON A PE THE VIOLATION REGISTER CONTAINS A COPY OF THE M REGISTER.


TRY PE
TURN ON PE INTERRUPT (STF 5)
RE-READ PE LOCATION (XLA OR LDA)
PE AGAIN ?
NO
RETURN
YES
GOTPE
ENTER THRU TRAP CELL 5 (JSB GOTPE,I)
GET PHYSICAL PAGE # FROM SYS. OR USER MAP
WHERE WAS PE ?
DCPC MAPS
SYSTEM MAP
HALT "102005"
A REG. = PHY. PAGE #
B REG. = LOG. PAGE #
(OCCURS IF PE IN SYSTEM, SYSTEM TABLES, OR COMMON)
WAS RTE EXECUT-ING?
YES
HALT "103005"
A = PHY. PAGE #
B = LOG. PAGE #
(THIS IS NECESSARY SINCE RTE IS NOT RE-ENTRANT)
NO
USER BUFFER IN USER PTN ?
NO
IN SAM
YES
USER MAP
MR PROGRAM ?
NO
UNDEFINE PARTITION
"PART'N SS DOWN"
"PART'N MM DOWN"
(IF A MOTHER PTN. IS INVOLVED)
YES
"PE PG# PROGA BAD"
"ABE AAAAAA BBBBBB E"
"XYO XXXXXX YYYYYY O"
"PE PROGA LLLLL"
"PROGA ABORTED"

RESOURCE NUMBERS

RESOURCE MANAGEMENT

(RESOURCE NUMBERING)

ALLOWS COOPERATING PROGRAMS A METHOD OF SHARING RESOURCES

CALL RNRQ (ICODE, IRN, ISTAT)

	15	14	5	4	3	2	1	0
	WAIT OPTION		ALLOCATE OPTION			LOCK OPTION		
ICODE =	NO WAIT	NO ABORT	CLEAR	GLOBAL	LOCAL	CLEAR	GLOBAL	LOCAL

IRN = RESOURCE NUMBER. RETURNED ON ALLOCATE; REQUIRED OTHERWISE.

ISTAT (RETURNED) =
0 NORMAL DEALLOCATE
1 RN IS CLEAR (UNLOCKED)
2 RN IS LOCKED LOCALLY TO CALLER
3 RN IS LOCKED GLOBALLY
4 NO RN AVAILABLE NOW
5 —
6 RN IS LOCKED LOCALLY TO ANOTHER PROGRAM
7 RN WAS LOCKED GLOBALLY WHEN REQUEST WAS MADE

NOTE: STATUS 4, 6, AND 7 ARE RETURNED ONLY IF "NO WAIT" BIT IS SET.

RESOURCE NUMBER TABLE

```
      +-----------------+
$RNTB |     # of RNs    | 1
      +--------+--------+
 RN 1 | OWNER1 |LOCKER1 | 2
      +--------+--------+
 RN 2 | OWNER2 |LOCKER2 | 3
      +--------+--------+
      |        .        |
      |        .        |
      ~        .        ~
      ~        .        ~
      |        .        |
      |        .        |
      +--------+--------+
 RN N | OWNERn | LOCKERn| N+1
      +--------+--------+
```

OWNER/LOCKER ENTRY:
- ID segment number in keyword block of owner/locker program
- 377B if globally owned/locked
- 0 if not owned or locked

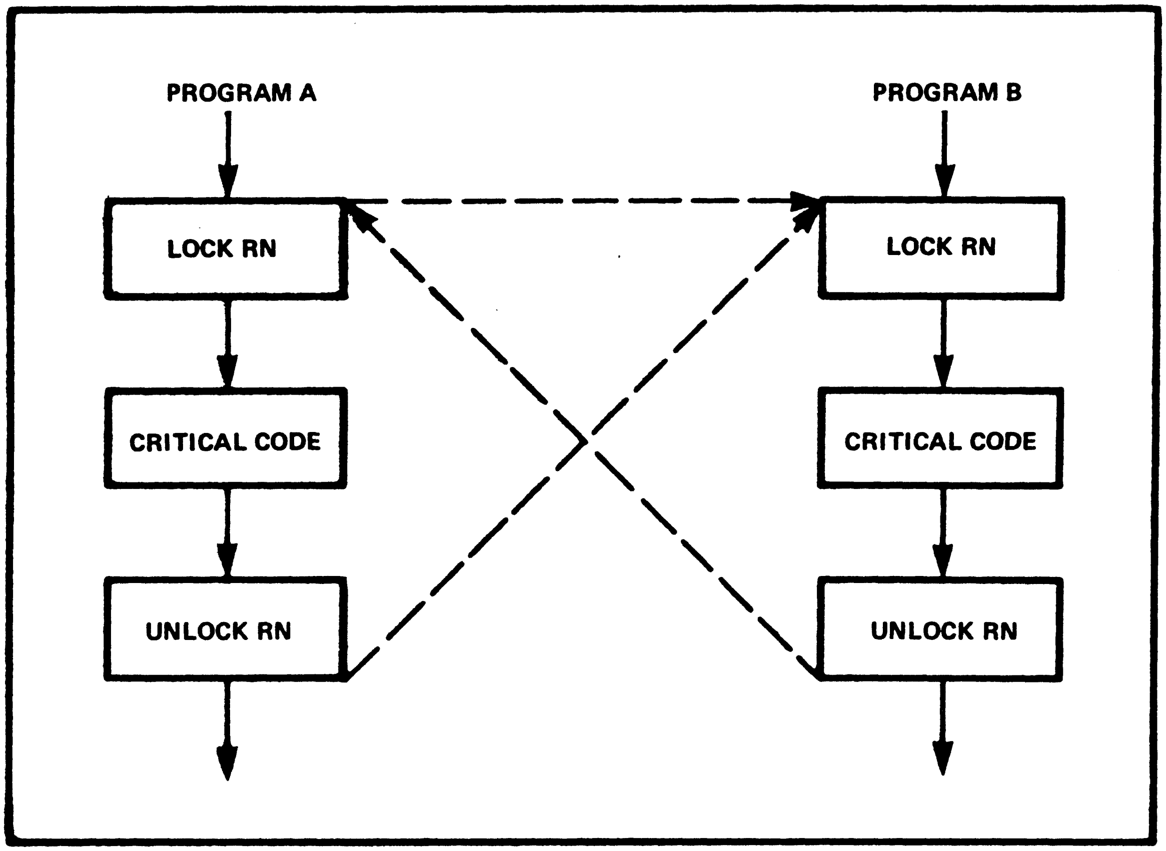
RESOURCE NUMBER FORMAT

```
15                8 7                0
+------------------+-----------------+
| OWNER ENTRY FROM | RESOURCE NUMBER |
|     RN TABLE     |      (1-N)      |
+------------------+-----------------+
```

RN MANAGEMENT

- RNRQ manages RN´s
- "RN" bit in ID segment word 21 is set when a program is a RN owner or locker.
- RNRQ is a type 6 utility subroutine
- Programs attempting to lock a locked RN are put into general wait (3)
- When a program aborts or terminates, the system ($TRRN):
  - Releases the program´s local RN locks
  - Deallocates its local RN´s
  - Reschedules waiting programs

RN APPLICATIONS



RESOURCE NUMBER (RN) LOCKING ALLOWS TWO OR MORE COOPERATING PROGRAMS TO ACCESS SENSITIVE AREAS OF THEIR CODE ON A ONE-AT-A-TIME-ONLY BASIS

CRITICAL CODE MIGHT REFERENCE SHARED:

1. COMMON AREAS
2. DATA BASE
3. PERIPHERAL DEVICES
4. DISC FILES

LOGICAL UNIT LOCK

LOGICAL UNIT LOCK

ALLOWS A PROGRAM TO EXCLUSIVELY DOMINATE (LOCK) A GROUP OF I/O DEVICES

DIMENSION LUARY (number of LU's to be locked)

CALL LURQ (IOPTN, LUARY, NOLU)

- IOPTN:
  - 0 0 0 0 0 0 B = UNLOCK SPECIFIED LU'S
  - 1 0 0 0 0 0 B = UNLOCK ALL LU's
  - 0 0 0 0 0 1 B = LOCK WITH WAIT
  - 1 0 0 0 0 1 B = LOCK WITHOUT WAIT
  - 4 = NO ABORT ON ERRORS
- LUARY: ARRAY OF LU'S TO BE LOCKED
- NOLU: NUMBER OF LU'S TO BE LOCKED

ON RETURN:

"A" REGISTER = 0 LU LOCK SUCCESSFUL
=>1 LU LOCK NOT SUCCESSFUL
= -1 NO RESOURCE NUMBER AVAILABLE
= 1 ONE OR MORE OF THE LU'S ALREADY LOCKED

THIS CALL USES RESOURCE NUMBERS

LU LOCK MANAGEMENT

- LURQ (a type 6 utility subroutine) manages LU locks
- LURQ flow is:
  - Allocate a local RN for the calling program
  - Locally lock the RN to the caller
  - Enter RN into DRT entry for each LU
- Programs attempting to use or lock a locked LU are put into general wait (3)
- A maximum of 31 programs may simultaneously lock LUs
- LU locks are removed by $TRRN when the program:
  - Terminates
  - Terminates serially reusable
  - Aborts

LU LOCK EXAMPLE

```
PROGRAM XYZ
    .
    .
    .
LUARY(1)=5
LUARY(2)=7
CALL LURQ(1,LUARY,2)
    .
    .
    .
```

SYSTEM TABLES WOULD BE:

```
            DRT                                     RN TABLE
          PART 1
  +-----------------+                      +----------------+
  |        .        |           $RNTB      |       N        |
  |        .        |                      |                |
  |        .        |                      +-------+--------+
  |        .        |                      |       |        | 1
  +-----+-----+-----+                      |       |        |
5 |     |  2  |     |                      +-------+--------+
  |     |     |     |                      |  15*  |   15   | 2
  +-----+-----+-----+                      |       |        |
6 | SUB | RN  | EQT |                      +-------+--------+
  |CHAN |     |     |                      |       .        |
  +-----+-----+-----+                      |       .        |
7 |     |  2  |     |                      ~       .        ~
  |     |     |     |                      ~       .        ~
  +-----+-----+-----+                      |       .        |
  |        .        |                      |       .        |
  ~        .        ~                      |       .        |
  ~        .        ~                      |       .        |
  |        .        |                      |       .        |
  |        .        |                      |       .        |
  +-----------------+                      +----------------+ N+1
```

*Assumes that Program XYZ's keyword block index to its ID segment is 15.

PROGRAM STATES

PROGRAM STATES

•	Dormant		0
•	Scheduled		1
•	Executing		1
•	Suspended:	I/O suspend	2
		unavailable memory	4
		disc allocation	5
		operator suspend	6
•	General Wait		3

The General Wait state is implimented to indicate a waiting state in which a program is swappable. Reasons for wait include waiting for:

a. buffered I/O
b. class I/O "Get"
c. Resource Number lock/availability
d. I/O class availability
e. program scheduled with wait
f. scheduling in queue
g. downed I/O device
h. LU lock
i. buffer limit exceeded

Dormant substates include:

a. terminate sarring resources
b. terminate serial re-useable
c. truly not active

RTE STATE CHART

(handout)

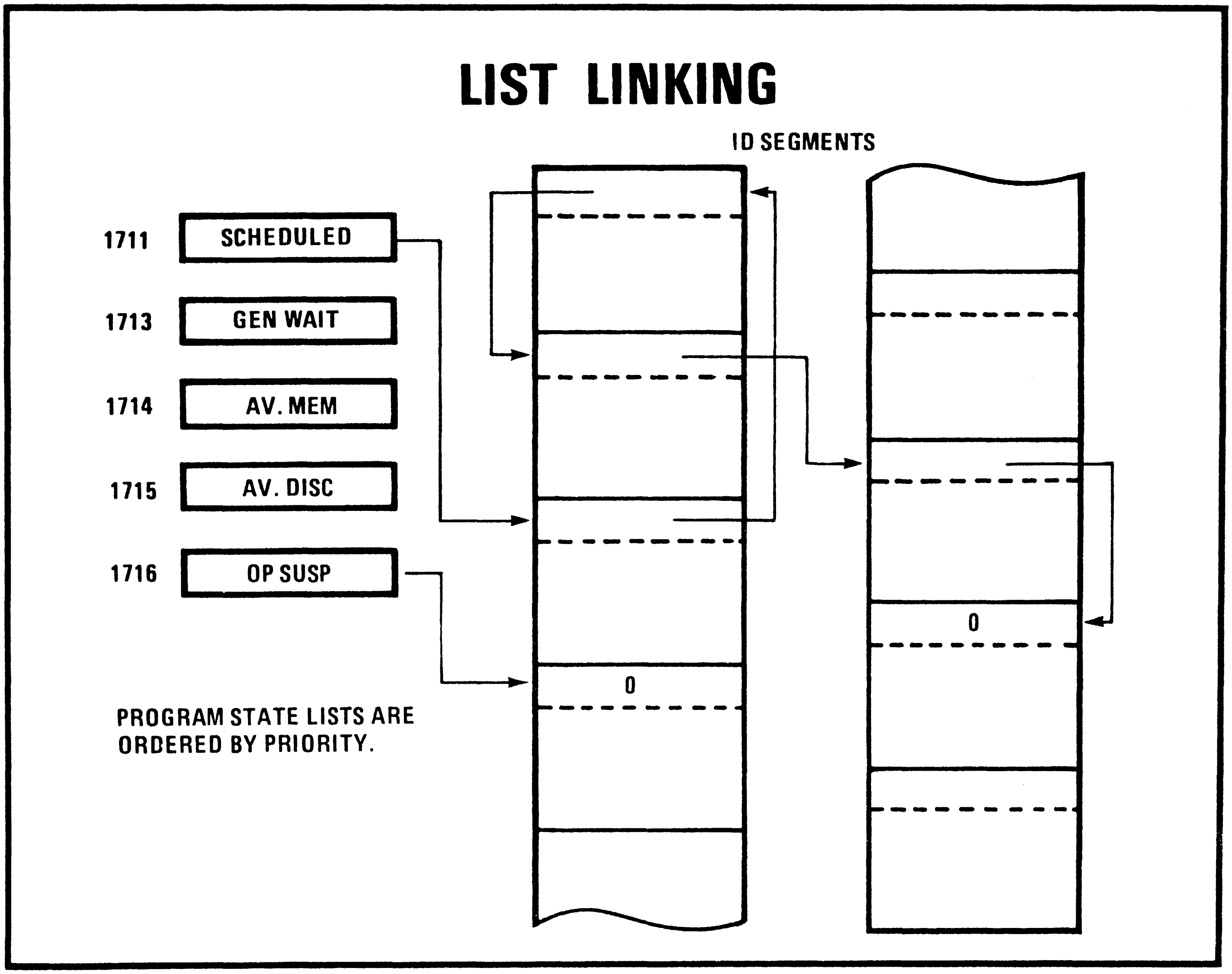
RTE moves programs from state to state by linking and unlinking the programs ID segment between the appropriate state lists.

Programs in each state are grouped as follows:

STATE	LISTHEAD
- Dormant (0)	none
- Scheduled (1)	SKEDD(1711)*
- I/O Suspend (2)	each EQT entry
- General Wait (3)	SUSP2(1713)
- Memory (4)	SUSP3(1714)
- Disc Allocation (5)	SUSP4(1715)
- Operator Suspend (6)	SUSP5(1716)

* Octal base page addresses

LIST LINKING



. ID link word is word =1.

STATE REPRESENTATION IN PROGRAM'S ID SEGMENT

ID SEGMENT WORD(BITS)	CONTENTS
1	Linkage(or 0)
2	I/O Suspend(2): EXEC call request parameters in ID2 thru ID6.

General Wait(3):

a. ID segment address of program waiting to schedule

b. ID segment address of son waiting to complete

c. Address of RN table ($RNTB) if waiting on RN allocation.

d. Address of RN if waiting on a RN lock. $RNTB < ID(2) < $RNTB + ($RNTB)

e. "4" if waiting on a downed device. Also ID(3) will be the LU# of the device.

f. Address of class table ($CLAS) if waiting on class allocation.

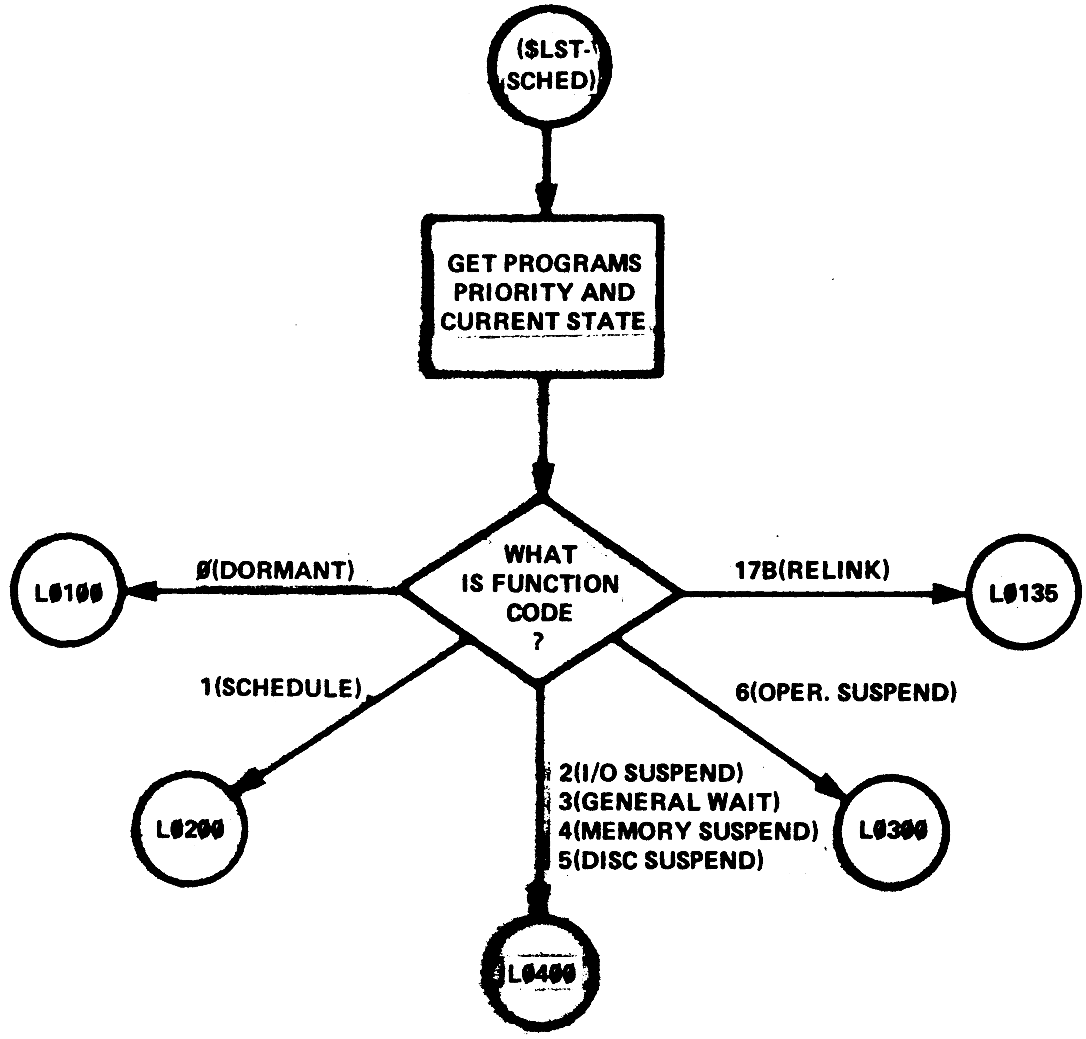
g. Address of class number if waiting on a class "GET". $CLAS < ID(2) < $CLAS + ($CLAS).

h. Address of RN if waiting on an LU lock. DRT will also contain the RN.

i. EQT entry address on which the buffer limit has been exceeded.

STATE REPRESENTATION INn
PROGRAM´S ID SEGMENT (cont´d)

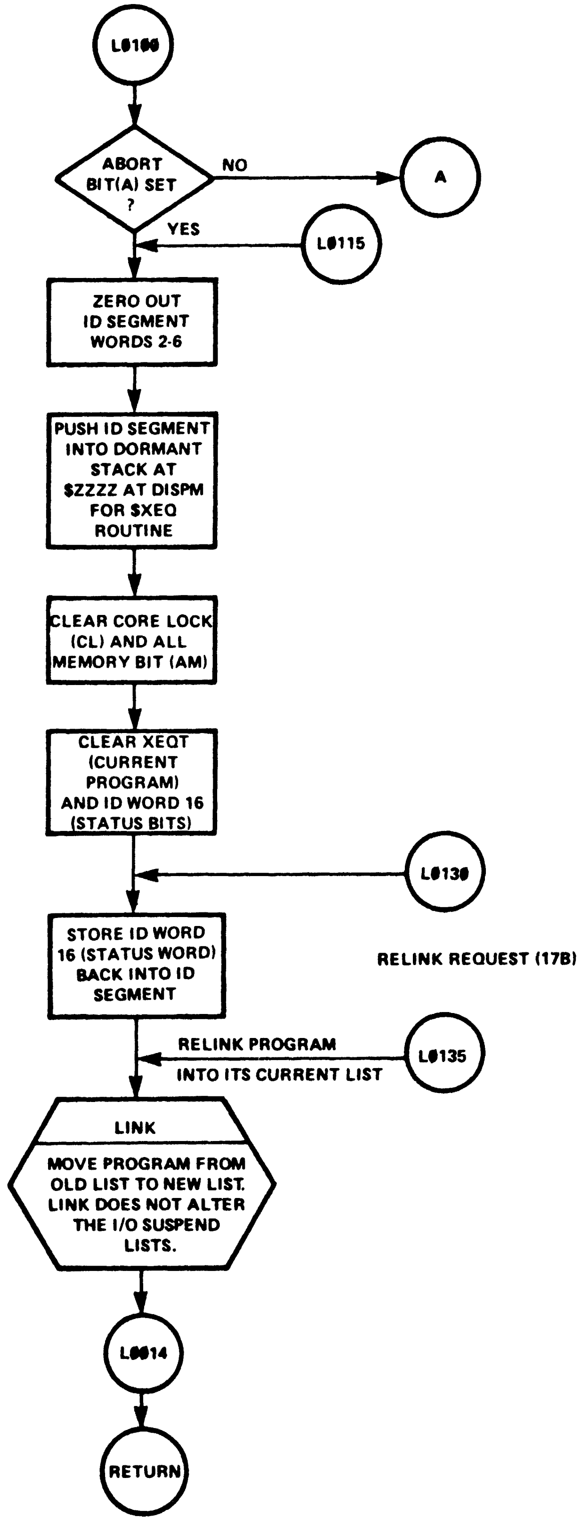
WORD (BITS)	CONTNENTS
2 cont´d	Memory(4): Number of SAM words requested
16(3-0)	Program state(0-6)
16(6)	Dormant (D) bit: set program dormant on next schedule attempt.
16(7)	Save resources (R) bit: program wants to save it´s resources when it goes dormant. (R bit is cleared when set dormant.)
16(9)	Operator suspend(O) bit: suspend program as soon as feasible.
16(11)	Abort (A) bit: abort program and set dormant as soon as feasible.
16(12)	Wait (W) bit: this program is waiting to schedule another program.

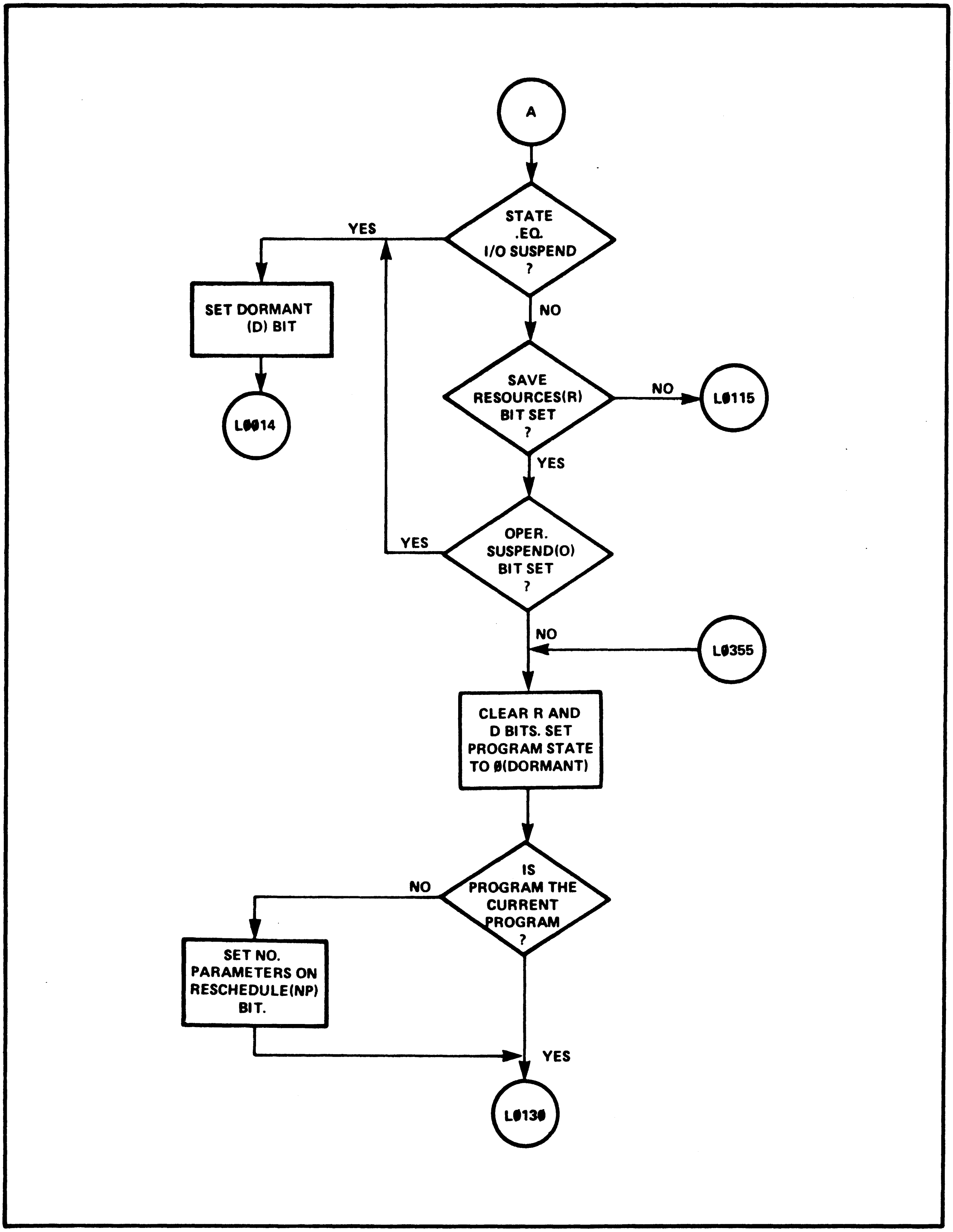


CALLING SEQUENCE

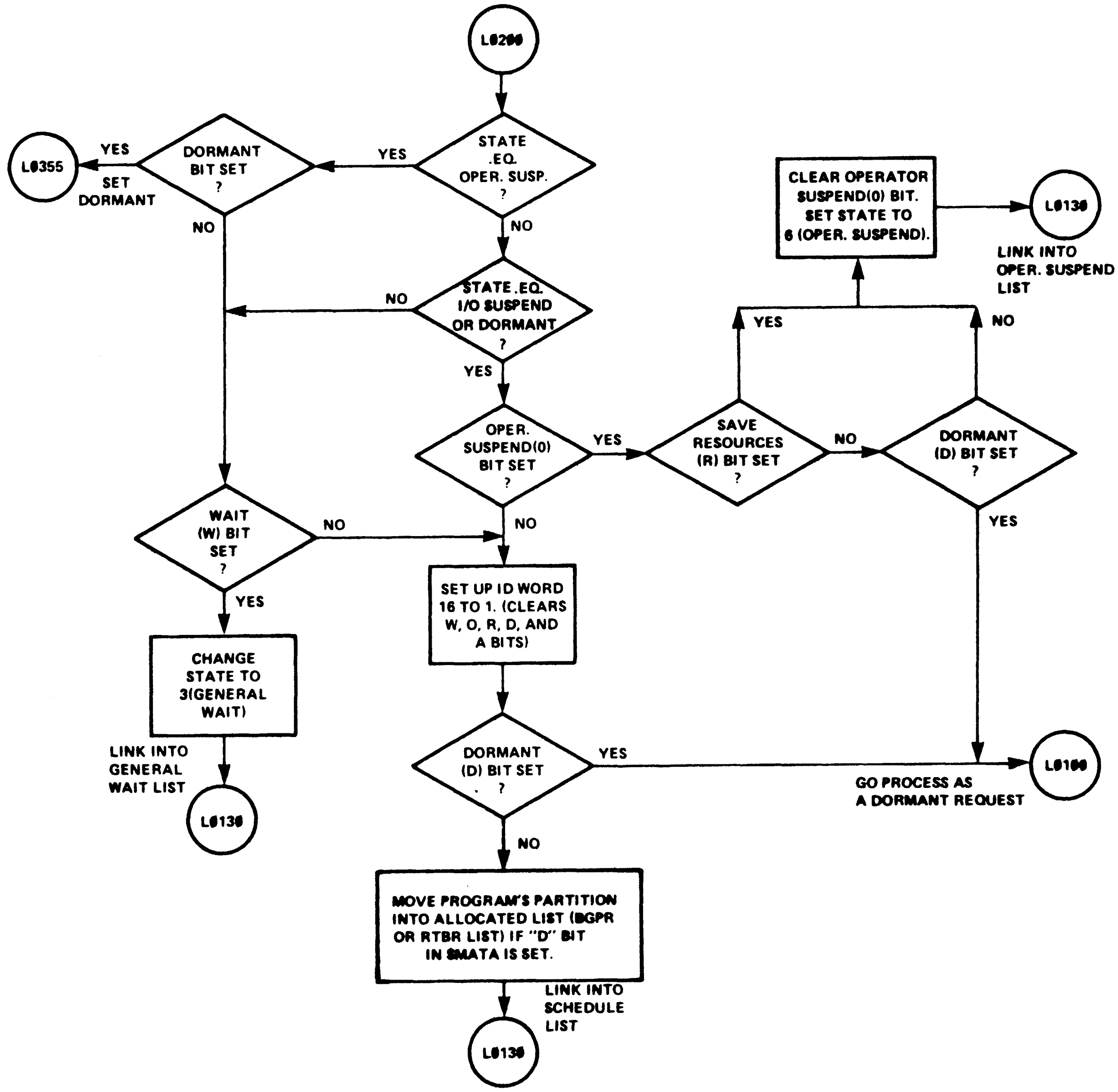
JSB	$LIST
OCT	(ADDRESS CODE) (FUNCTION CODE: 0-6 AND 17B)
DEF	ID SEGMENT ADDRESS

DORMANT REQUEST (0)

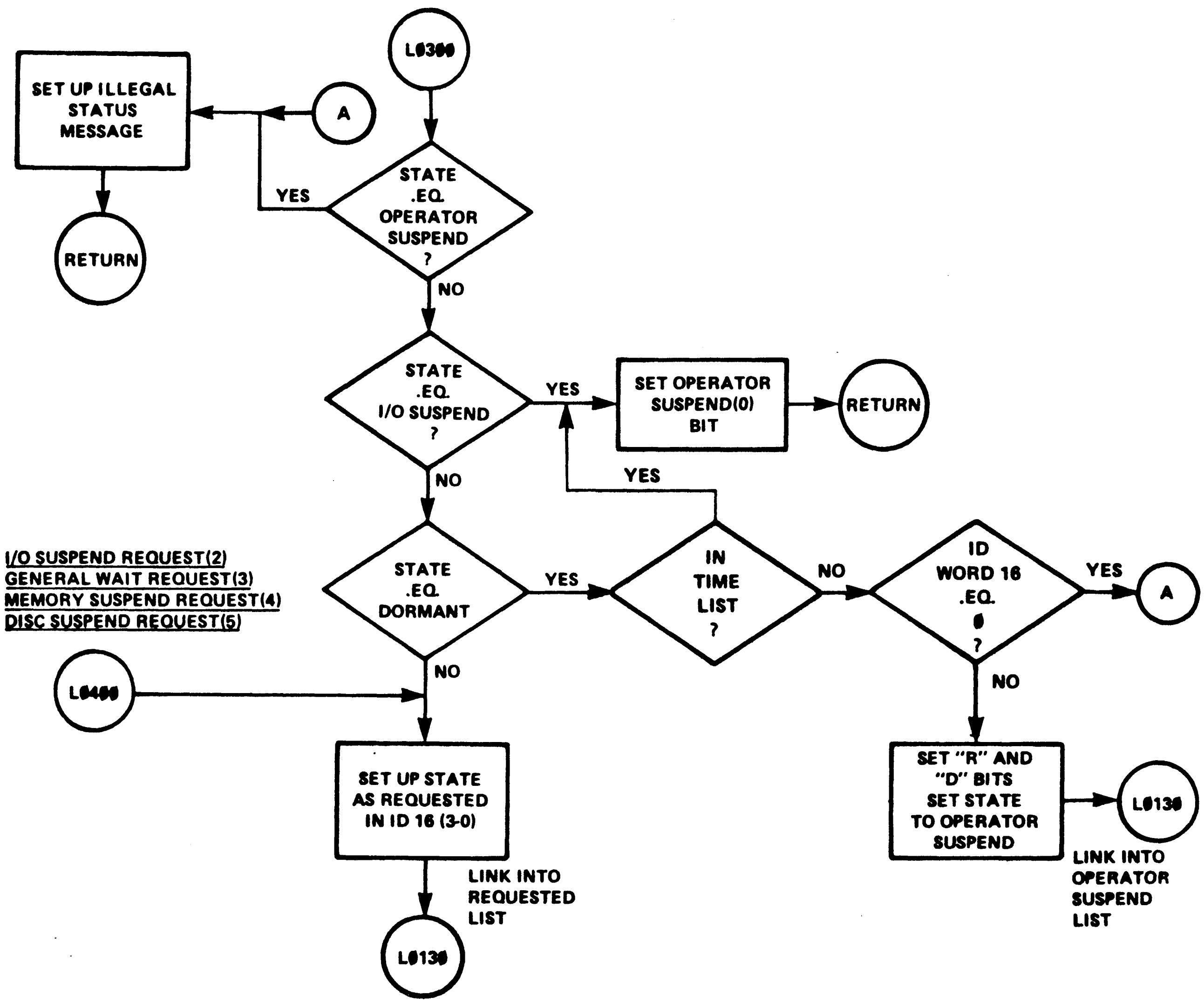



A
STATE .EQ. I/O SUSPEND ?
YES
NO
SET DORMANT (D) BIT
L0014
SAVE RESOURCES(R) BIT SET ?
NO
L0115
YES
OPER. SUSPEND(O) BIT SET ?
YES
NO
L0355
CLEAR R AND D BITS. SET PROGRAM STATE TO 0(DORMANT)
IS PROGRAM THE CURRENT PROGRAM ?
NO
SET NO. PARAMETERS ON RESCHEDULE(NP) BIT.
YES
L0130

SCHEDULE REQUEST (1)



OPERATOR SUSPEND REQUEST ⑥



```
*THE FOLLOWING IS A SAMPLE OUTPUT OF THIS PROGRAM:
*                 ON,WHZAT,LU
*
* 09:51:50:710
* *********************************************************************
* PT SZ PRGRM,T ,PRIOR*DRMT*SCHD*I/O *WAIT*MEMY*DISC*OPER * NEXT TIME  *
* *********************************************************************
*  0 ** MEM   *1 *09000 ***** 1
*  0 ** R$PN$*1 *00010 **************** 3,CL  032
*  3  5 PROGA*3 *00097 ********************************** 6
*  4  5 PROGB*3 *00097B**************** 3,LULK 40,LKPRG=PROGA
*  5 17 PROGC*3E*00097 **************** 3,RN  031,LKPRG=PROGD
*  3A27 PROGD*4 *00097 **************** 3,RESOURCE
*  5  7 PROGE*3 *00097 **************** 3,CLASS #
*  2  4 QUIKR*3 *00099  0 *****************************************00:00:00:000
*  6  7 FMGR *3 *00090 **************** 3,EDITR'S QUEUE
*  3  7 EDITR*3 *00050 *************************** 5
*  6 15 ASMB *3 *00099 **************** 3,LU,EQ DN   6,  5(0[00000000])
*  4A 6 TIMEL*4 *00090 **************** 3,LU,EQ DN   6,  5(0[00000000])
*  4A 6 TIMEL*4 *00090 **************** 3,LU,EQ DN *********00:00:00:000
*  7  7 FMG07*3 *00050 **************** 3,BL,EQT 7
*  2  3 WHZAT*4 *00001 ***** 1
*  0 ** RENSB*1 *00060 ******************** 4
*  3  6 PROGF*4 *00096 **************** 3,RN  031,LKPRG=GLOBL
*  6  7 EU26 *3 *00050 ********** 2, 16(2[00000010])
* *********************************************************************
* DOWN LU'S,  6, 14
**********************************************************************
* DOWN EQT'S,  5,  6
* *********************************************************************
* 09:51:50:710
*
```

Reason for I/O Suspend: EQT entry number (STATUS field of EQT entry word 5[binary content of STATUS])

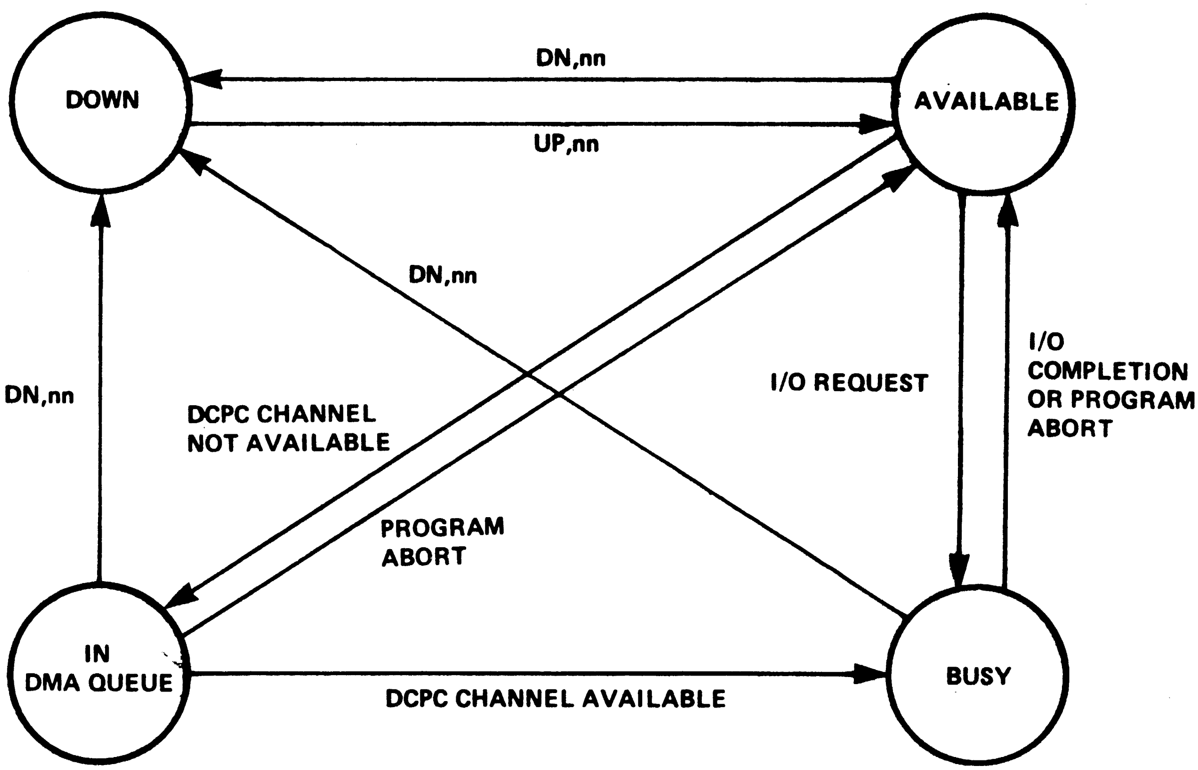
Reason for wait:

BL,EQT eqt	Buffer limit exceeded on the controller in EQT entry eqt
CL ccc	Waiting for class number ccc to complete GET
CLASS #	Waiting for a class number
LU/EQ DN	A device or controller is down. Look at DOWN LU´s or DOWN EQT´S list at bottom of report
LULK lu,LKPRG= prog name	Logical unit with number lu is locked to named program
program name	Waiting for named program to comlete
program´S QUEUE	Waiting to schedule named program which is busy
RESOURCE	Waiting for resource number
RN nnn,LKPRG= prog name	Resource number nnn is locked to named program

PT SZ	COLUMN HEADING (PARTITION NUMBER AND PARTITION SIZE)
0 **	MEMORY RESIDENT PROGRAM
5 8	PARTITION #5 IS USED AND 8 PAGES IN USE
11	SCHEDULED PROGRAM IS NOT YET IN PARTITION

´A´ AFTER THE PARTITION # MEANS THE PROGRAM WAS ASSIGNED
´E´ AFTER THE PROGRAM´S TYPE MEANS IT IS AN EMA PROGRAM
´B´ AFTER THE PROGRAM´S PRIORITY MEANS RUNNING UNDER BATCH

EQT ENTRY STATE DIAGRAM



EQT entry is kept in EQT entry word 5 in the AV field where:

- 0 = Available
- 1 = Down (only with a "DN" command
- 2 = Busy
- 3 = Waiting for DCPC channel

HANDLING OF "DISPLACED" I/O BUFFERS DUE TO A DOWNED DEVICE

Subchannels of a device are downed because of:

- Device-time out
- Device not ready
- Parity error
- "DN,,LU" command

I/O requests queued to the subchannel's EQT entry are removed and the follow action is taken depending on the I/O request type:

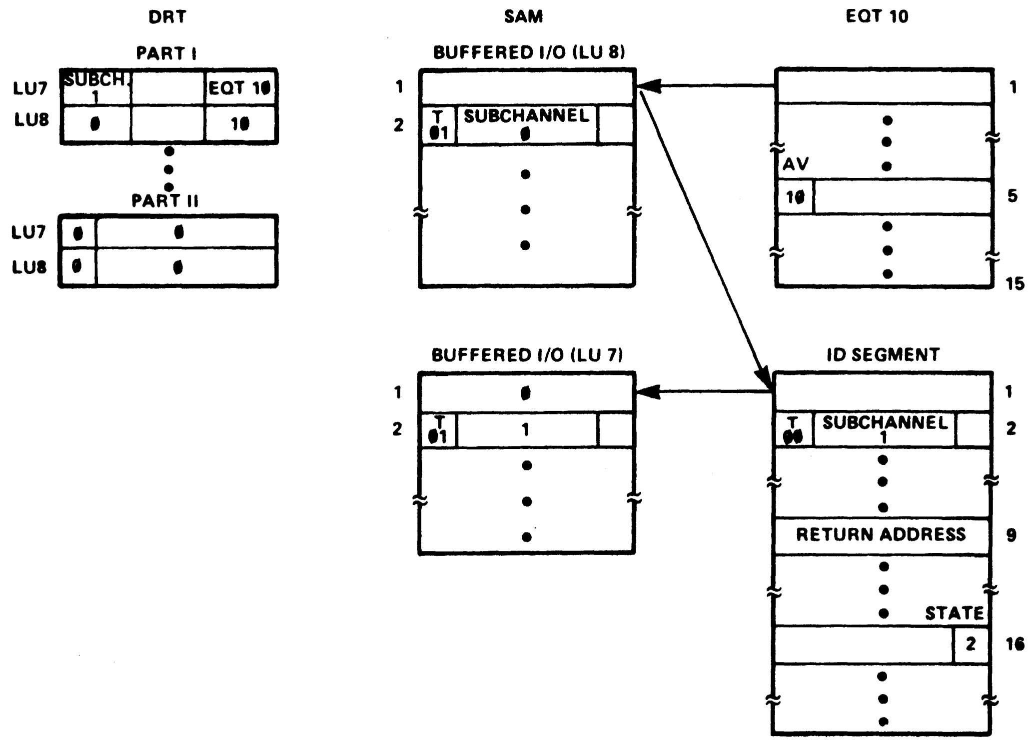
USER NORMAL OPERATION

The program's point of suspension (ID word 9) is backed up from the return address to the address of the "JSB EXEC" (saved in ID word 10). A "4" is stored into ID word 2, the program's state is changed to general wait (3), and its ID segment is linked into the general wait list.

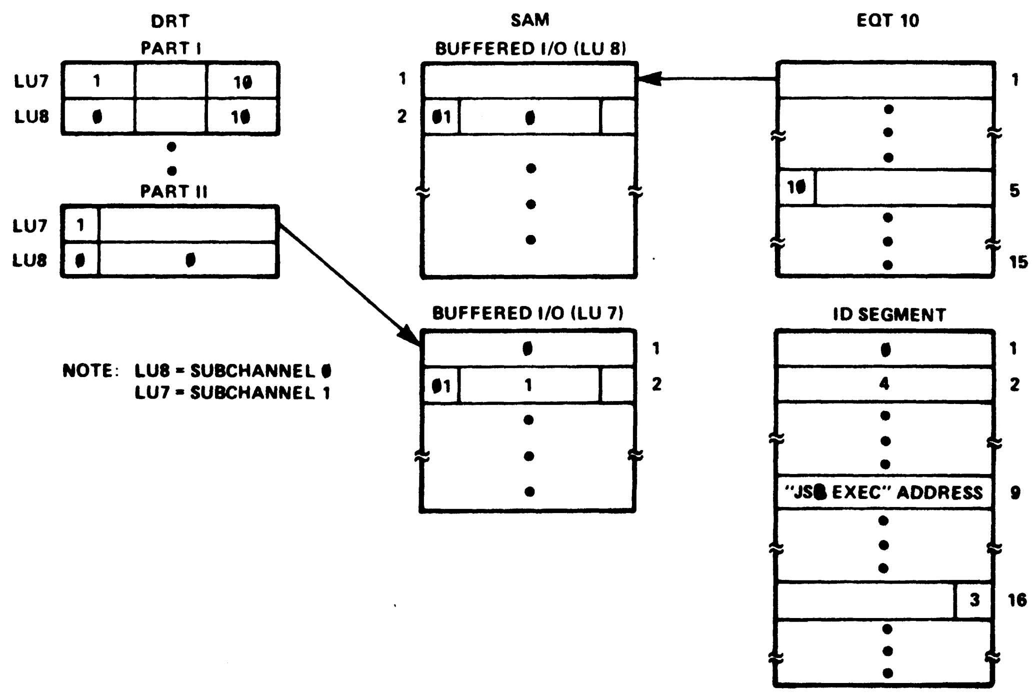
USER AUTOMATIC OUTPUT BUFFERING

The buffer is unlinked from the EQT entry and relinked into the second half of the DRT table on the associated LU. Bit 15 of the DRT is set to indicate the LU in down. CLASS I/O and SYSTEM I/O request are handled in the same manner.

BUFFER HANDLING EXAMPLE



*DN,,7



TBG TIME TICK

ENVIRONMENT BEFORE INTERRUPT:

1. TBG (Time Base Generator) on select code 11

2. System time is ten milliseconds before 8:00 a.m.

3. PROGA is in the time list and scheduled to run at 8:00 a.m.

4. PROGB is also in the time list and scheduled to run at 9:00 a.m.

5. PROGC is currently executing.

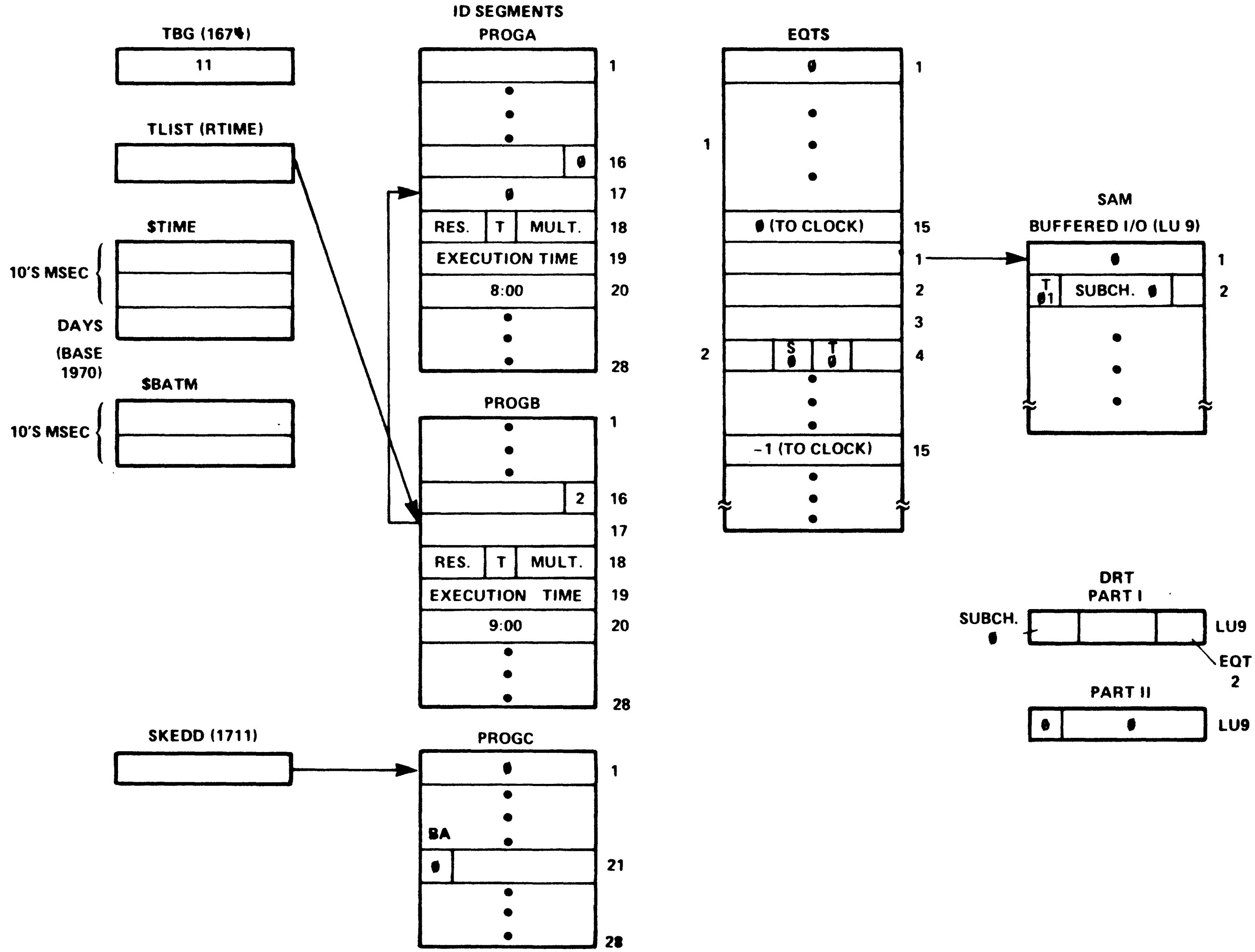
TIME TICK PROCESSING

EXECUTION	SUBROUTINE CALLS	NOTES
TBG interrupt		TBG generator interrupts every 10 milliseconds (msec)
TRAP CELL 11(JSB $CIC,I)		Interrupt causes the instruction in trap cell 11 to be executed
$CIC(RTIOC)		Saves the machine state and turns of the interrupt system. Compare select code of interrupt (CIR=11) to base page (BP) TBG word (1674B).
$CLCK(RTIME)		Step the system time kept in RTIME ($TIME) in a double word integer in 10's of msec. The first word is stepped ($TIME) and if it goes to zero, the second word ($TIME+1) is stepped. If the second word goes to zero it is midnight and the double integer is reset for the next day and the day word ($TIME+2) is stepped. Days are kept in one word referenced to the base year (1970).

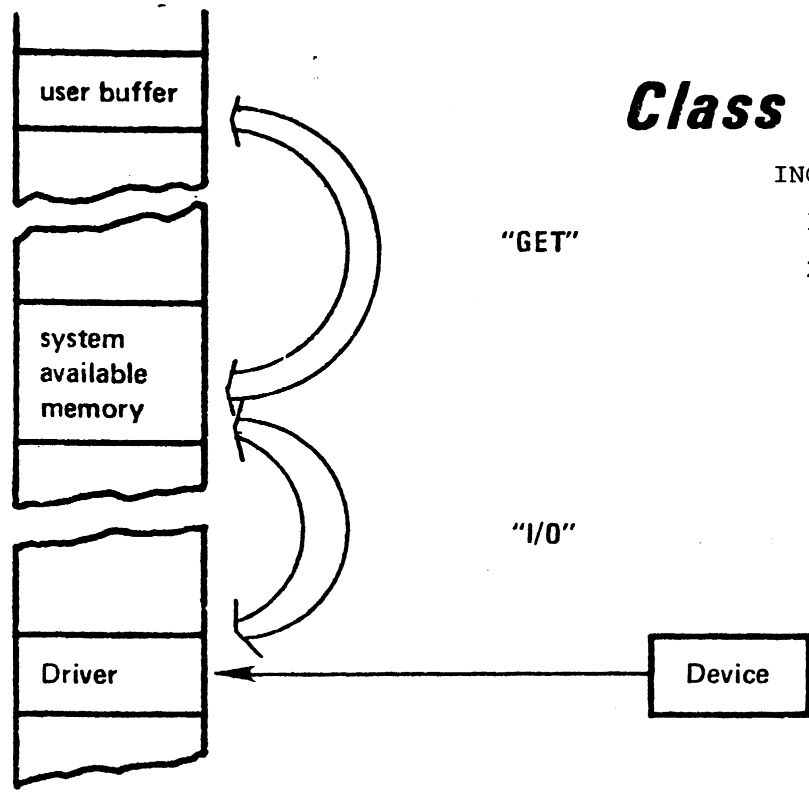
EXECUTION	SUBROUTINE CALLS	NOTES
CL010($CLCK)		Compare system time to next execution time (ID words 19 and 20) of each program on time list (threaded thru ID word 17).
	TMSCH($CLCK)	If times are equal, the program should be scheduled.
	$LIST	If the program is in the dormant state (ID word 16), schedule it.
	$TREM	If the program's MULTIPLE value (ID word 18) is zero, remove the program from the time list.
	TUDAT	If the MULTILE value is not zero, use it and the RESOLUTION value (ID word 18) to calculate the program next execution time.
TOBAT($CLCK)		Step the batch time ($BATM) if the currently executing program is a a batch program (ID word 21-BA bit) and not SMP or D.RTR.
	$ERMG	If batch time is zero, abort current batch program.

EXECUTION	SUBROUTINE CALLS	NOTES
IOTOP($CLCK)		Step the time out clock (EQT entry word 15) in each EQT entry which has a time out in progress. (EQT word 15 not zero) If none of the time out clocks go to
$XEQ(DISPM)		zero then dispatch the next program
$DEVT(RTIOC)		else the EQT entry has timed out. Set the time out bit (T) in EQT entry word 4.
CIC.6($CIC)		If the driver will handle the time outs (EQT entry word 4-S bit); enter the driver's continuation/completion section (Cxnn).
IOCOM-$CON1(RTIOC)		If the driver is not to handle the time out, a driver completion return is simulated by transfering to the I/O completion routines with an error code of 4 (time out). A time out message will be output and the associated LU's will be set down.
XEQ(DISPM)		Dispatch the next program.

TIME TICK TABLES/LISTS (INITIAL CONDITIONS)



CLASS I/O



Class I/O

INCLUDES:

1. I/O WITHOUT WAIT
2. PROGRAM-TO-PROGRAM I/O

In Class I/O, the user area is buffered in System Available Memory in a block of memory identified by a "Class Number". The user is thus swappable. The data is retrieved with a "Class Get" call to the appropriate class number.

NOTE NOTE NOTE

All Class I/O is double-call I/O: One call to initiate the operation
and
One call to complete the operation

CLASS I/O-READ/WRITE

TRANSFERS DATA TO OR FROM AN I/O DEVICE. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

DIMENSION IBUFR (size)
ICODE = (17, READ; 18, WRITE; 20, WRITE THEN READ)
ICLAS = 0, ALLOCATE A CLASS NUMBER;
1 - 255, A CLASS NUMBER TO USE

CALL EXEC (ICODE, $\underbrace{\text{ICNWD, IBUFR, IBUFL,}}_{\text{SAME AS FOR ICODE = 1 OR 2 (IBUFR IS A DUMMY VARIABLE FOR ICODE = 17)}}$ $\underbrace{\text{IPRM1, IPRM2, ICLAS}}_{\text{USER INFORMATION PASSED TO GET CALL}}$)

SAME AS FOR ICODE = 1 OR 2
(IBUFR IS A DUMMY
VARIABLE FOR ICODE = 17)

USER INFORMATION
PASSED TO GET CALL

ICLAS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NO WAIT BIT			CLASS NUMBER												

NO WAIT BIT:
= 0, PROGRAM IS PUT IN GENERAL WAIT LIST (STATE 3) IF MEMORY OR CLASS NUMBER NOT AVAILABLE.
= 1, "A" REGISTER = -1, NO CLASS NUMBER AVAILABLE
"A" REGISTER = -2, NO MEMORY AVAILABLE
"A" REGISTER = Ø, SUCCESSFUL CALL

ON RETURN FROM CALL

* WRITE/READ IS USED WITH LU-Ø FOR PROG. TO PROG. COMMUNICATION

CLASS I/O - I/O CONTROL

TO PERFORM VARIOUS I/O CONTROL OPERATIONS SUCH AS BACKSPACE, WRITE END-OF-FILE, REWIND, ETC.. THE CALLING PROGRAM NORMALLY DOES NOT WAIT FOR THE CALL TO COMPLETE.

$$\text{CALL EXEC (19, }\underbrace{\text{ICNWD, IPRAM}}_{\substack{\text{SAME AS FOR}\\ \text{STANDARD I/O}\\ \text{CONTROL CALL}\\ \text{(REQUEST}\\ \text{CODE = 3)}}}\text{, }\underbrace{\text{ICLAS}}_{\substack{\text{SAME AS FOR}\\ \text{CLASS I/O –}\\ \text{READ/WRITE}}}\text{)}$$

CLASS I/O-GET

COMPLETES THE DATA TRANSFER BETWEEN THE SYSTEM AND USER PROGRAM THAT WAS PREVIOUSLY INITIATED BY A CLASS REQUEST.

CALL EXEC (21, ICLAS, IBUFR, IBUFL, IRTN1, IRTN2, IRTN3)

ICLAS

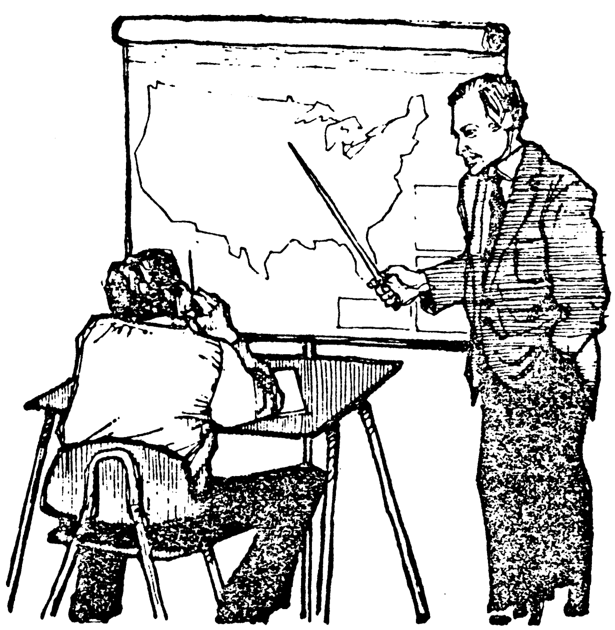
15	14	13	12	...	2	1	0

- BITS 2–0 (AND ON TO 12): CLASS NUMBER
- BIT 13: DO NOT DE-ALLOCATE CLASS NUMBER
- BIT 14: SAVE CLASS BUFFER
- BIT 15: NO WAIT BIT

NO WAIT BIT
- = 0, PROGRAM PUT IN GENERAL WAIT LIST (STATE 3) IF NO CALLS HAVE COMPLETED FOR THIS CLASS.
- = 1, RETURN IMMEDIATELY EVEN IF NO CALL HAS COMPLETED.

A AND B REGISTERS ON RETURN:

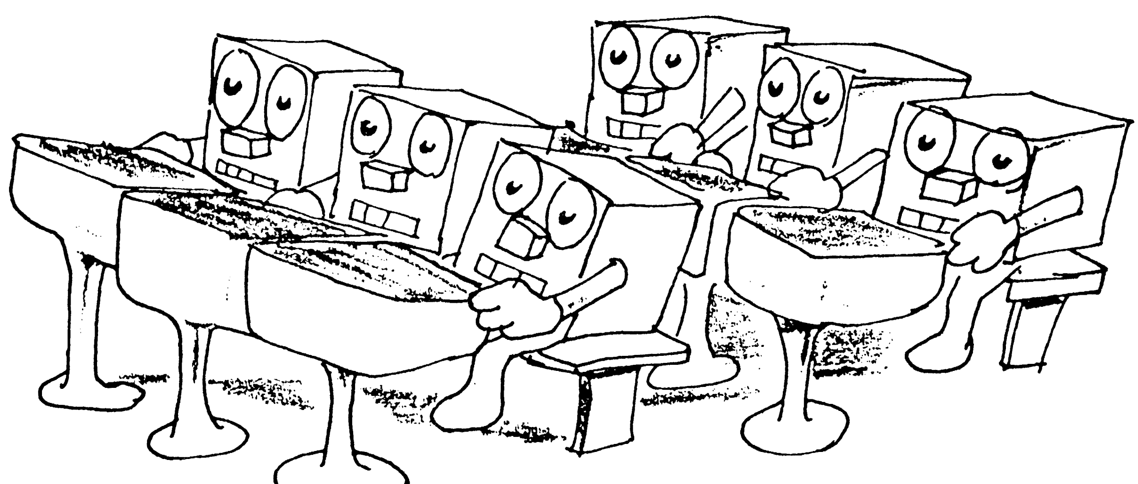
SUCCESSFUL GET	UNSUCCESSFUL GET
A REG, BIT 15 = 0	A REG, BIT 15 = 1
A REG = STATUS	A REG = NEGATIVE OF (NUMBER + 1) OF REQUESTS NOT COMPLETED FOR THIS CLASS
B REG = TRANSMISSION LOG	



CLASS I/O-GET (cont.)

IBUFR	DATA IS RETURNED HERE FROM CLASS READ (17) OR WRITE/READ (20) CALLS. IT IS A DUMMY VARIABLE FOR CLASS WRITE (18) AND CONTROL (19) CALLS.
IBUFL	DATA BUFFER LENGTH; WORDS (+), CHARACTERS (–)
IRTN1, IRTN2	USER INFORMATION PASSED FROM CLASS READ, WRITE OR WRITE/READ CALLS
IRTN3	REQUEST CODE RECEIVED BY DRIVER RETURNED HERE

ORIGINAL REQUEST CODE	VALUE RETURNED IN IRTN3
17/20 (READ, WRITE/READ)	1
18 (WRITE)	2
19 CONTROL	3



CLASS TABLE:

THE CLASS TABLE ENTRY CAN BE IN ONE OF FOUR DIFFERENT STATES:

```
 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
| 0   0  0  0   0  0  0   0  0  0   0  0  0   0  0  0|
------------------------------------------------------
STATE 1:  CLASS DEALLOCATED, AVAILABLE

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01  00
-------------------------------------------------------
| 0 |    A D D R E S S   O F  F I R S T   E N T R Y  |
-------------------------------------------------------
STATE 2:  POINTER TO FIRST ENTRY IN CLASS QUEUE

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
| 1   0  X| SECURITY CODE  | NUMBER OF PENDING REQS. |
           (5 LSB OF ID SEG)
------------------------------------------------------
STATE 3:  CLASS ALLOCATED, NO ONE WAITING ON CLASS
          NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
| 1   1  X| SECURITY CODE  | NUMBER OF PENDING REQS. |
------------------------------------------------------
STATE 4:  CLASS ALLOCATED, SOMEONE WAITING (SUSPENDED)
          NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255
```

CLASS QUEUE FORMAT:

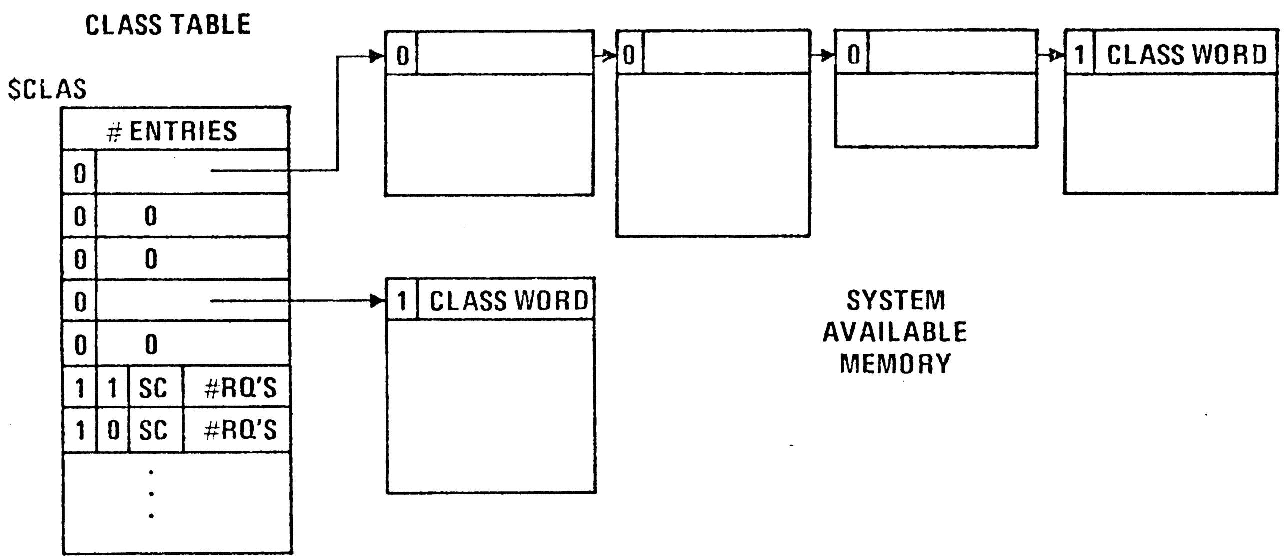
```
    WORD         CONTENTS
    ----         --------
     1    < LINKAGE WORD          >
     2    <T, CONTROL INFO, CODE  >
     3    <PRIORITY OF REQUESTOR  > (CHANGED TO STATUS AT COMP.)
     4    <TOTAL BLOCK LENGTH WORDS>
     5    <CLASS ID WORD          >
     6    <USER BUFFER LENGTH     > (CHANGED TO TLOG AT COMP.)
     7    < OPTIONAL PARAMETER 1  >
     8    < OPTIONAL PARAMETER 2  >
     9    <WORD 1 OF USER BUFFER  >
     .        .       .          .
     .        .       .          .
    N+8   <WORD N OF USER BUFFER  >
```

THE <T> FIELD (BITS 15-14 IN CONTROL WORD)
IDENTIFIES THE REQUEST TYPE AS:

```
00   USER (NORMAL OPERATION)
01   USER (AUTOMATIC BUFFERING)
10   SYSTEM
11   CLASS I/O
```

```
SPECIFICATION:    $CLAS = # ENTRIES IN CLASS TABLE
                          HEADS CLASS TABLE
```

CLASS I/O LINKING



sc = Security code (low 5 bits of owning program's index into keyword block)

#RQ'S = Number of outstanding class I/O segments (READ,WRITE, OR WRITE/READ).

CLASS NUMBER

1. BITS 0-7: index into class table
2. BITS 8-12: programs keyword block index (see SC above)

SYSTEM HANDLING OF CLASS I/O EVENTS

The action taken by RTE depends on the type of class I/O event and the state of the class queue listhead (or class table entry) as follows:

CLASS I/O REQUESTS:

STATE 1. STATE 3 IS SET UP, SECURITY CODE IS LOW 5 BITS OF PROGRAM ID NUMBER. COUNTER IS SET TO 1.
STATE 2. THE COUNTER AT END OF QUEUE IS INCREMENTED BY 1
STATE 3. THE COUNTER IS INCREMENTED BY 1.
STATE 4. THE COUNTER IS INCREMENTED BY 1.

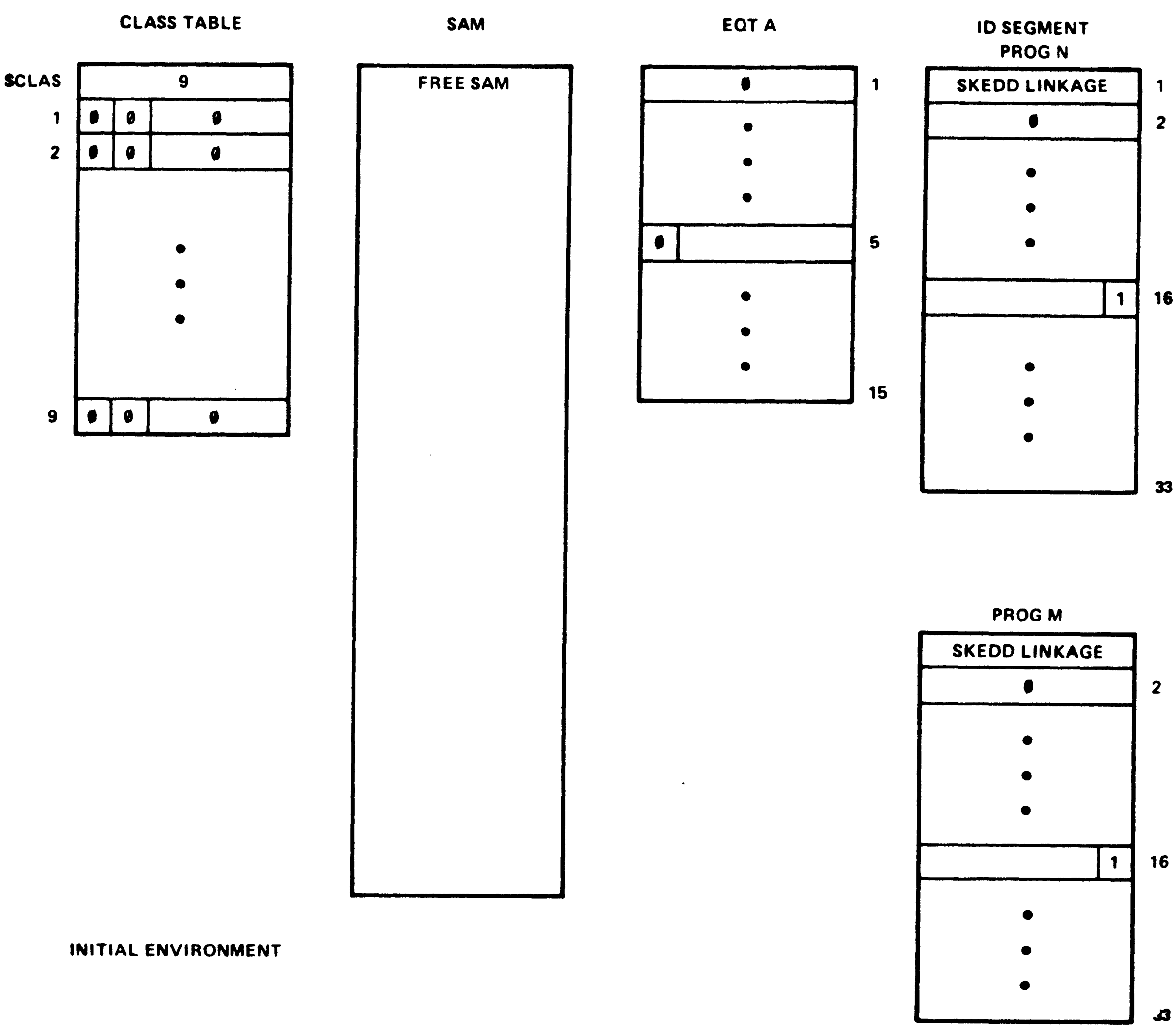
ON COMPLETION OF CLASS I/O REQUESTS:

STATE 1. ILLEGAL—SHOULD NEVER HAPPEN--BUFFER IS RETURNED AND THE COMPLETION IS IGNORED.
STATE 2. THE NEW DATA IS ADDED AT THE END OF THE LIST (FIFO) AND THE COUNTER IS DECREMENTED BY 1.
STATE 3. THE NEW DATA IS ADDED AT THE END OF THE LIST (FIFO) AND THE COUNTER IS DECREMENTED BY 1.
STATE 4. THE WAITING PROGRAM IS SCHEDULED AND THE COUNTER IS DECREMENTED BY 1 AND THE SOMEONE WAITING BIT(BIT14) IS CLEARED.

GET REQUESTS:

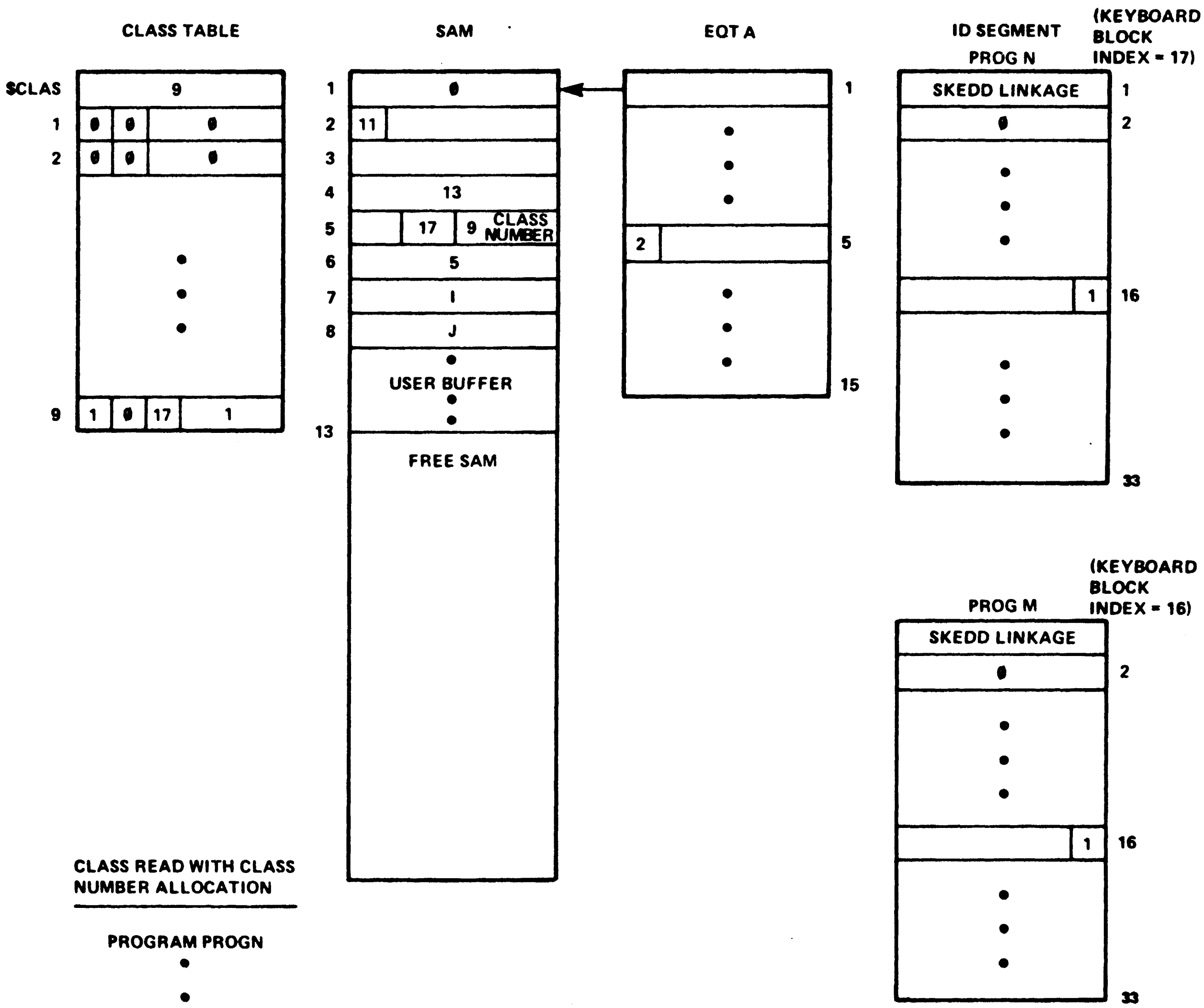
STATE 1. ABORT THE PROGRAM IO00. NO CLASS.
STATE 2. RETURN THE DATA FROM CLASS BUFFER
STATE 3. SET THE SOMEONE WAITING BIT(BIT14). SUSPEND PROGRAM
STATE 4. ABORT THE PROGRAM IO00. ONLY ONE PROGRAM MAY BE SUSPENDED PER CLASS.

CLASS I/O EXAMPLE



INITIAL ENVIRONMENT

CLASS I/O EXAMPLE



CLASS READ WITH CLASS NUMBER ALLOCATION

PROGRAM PROGN

•

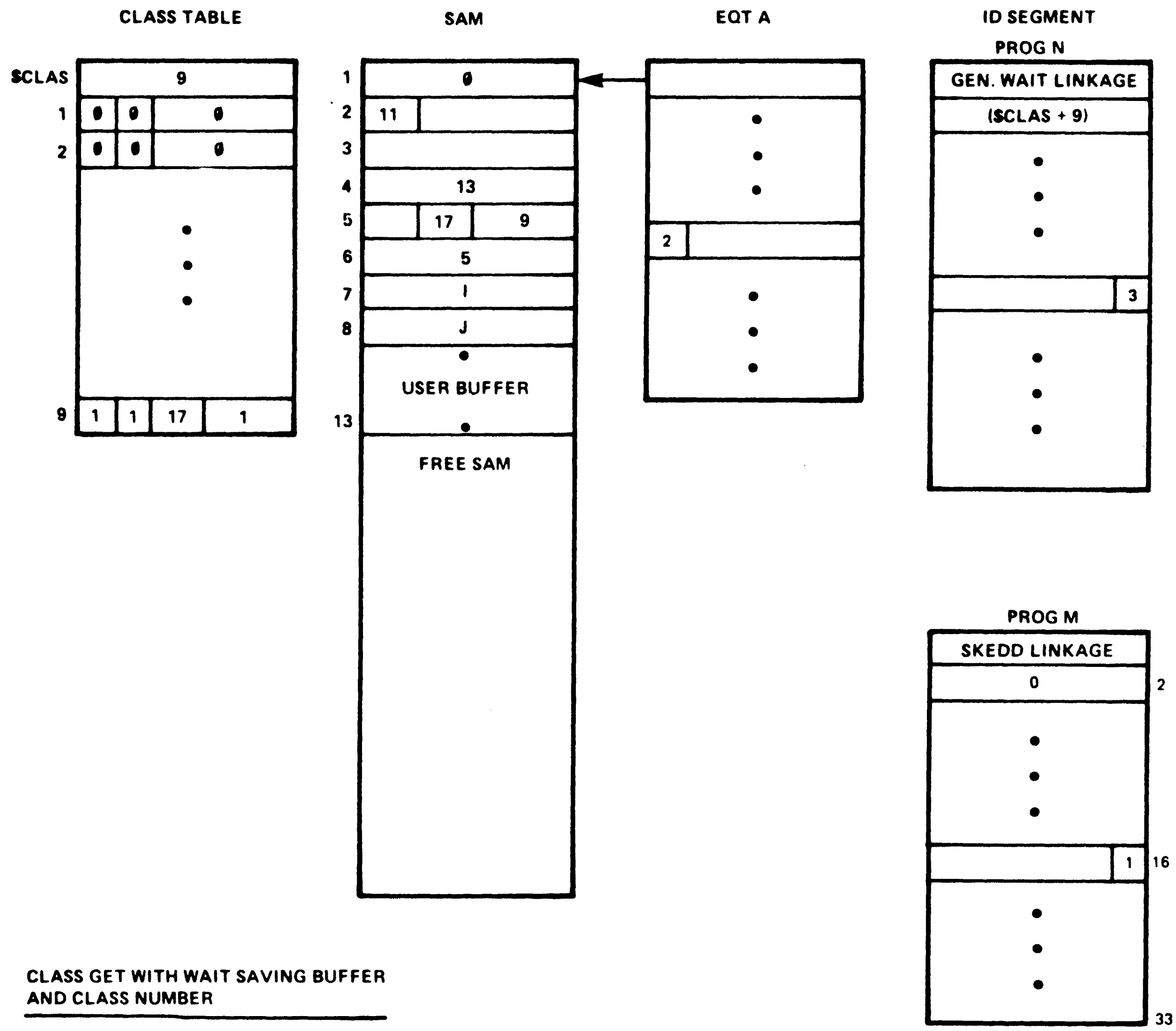
•

•

ICLAS = Ø

CALL EXEC (17,LU,IBUF,5,I,J,ICLAS)

CLASS I/O EXAMPLE



CLASS GET WITH WAIT SAVING BUFFER
AND CLASS NUMBER

PROGRAM PROGN

•

•

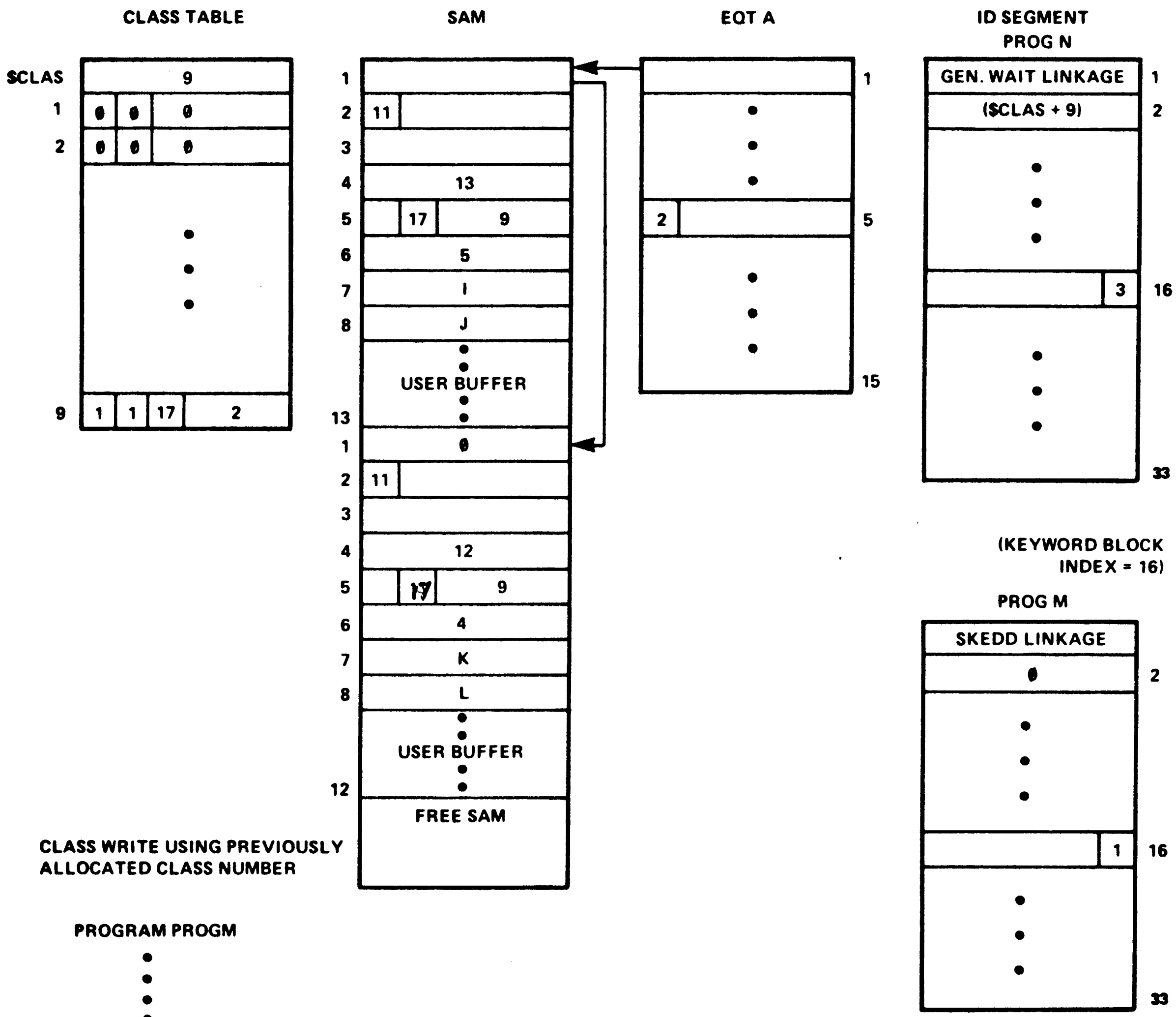
```
ICLAS = ICLAS + 60000B
CALL EXEC (21,ICLAS,IBUF,5,IRTN,JRTN,ICODE)
```

•

•

•

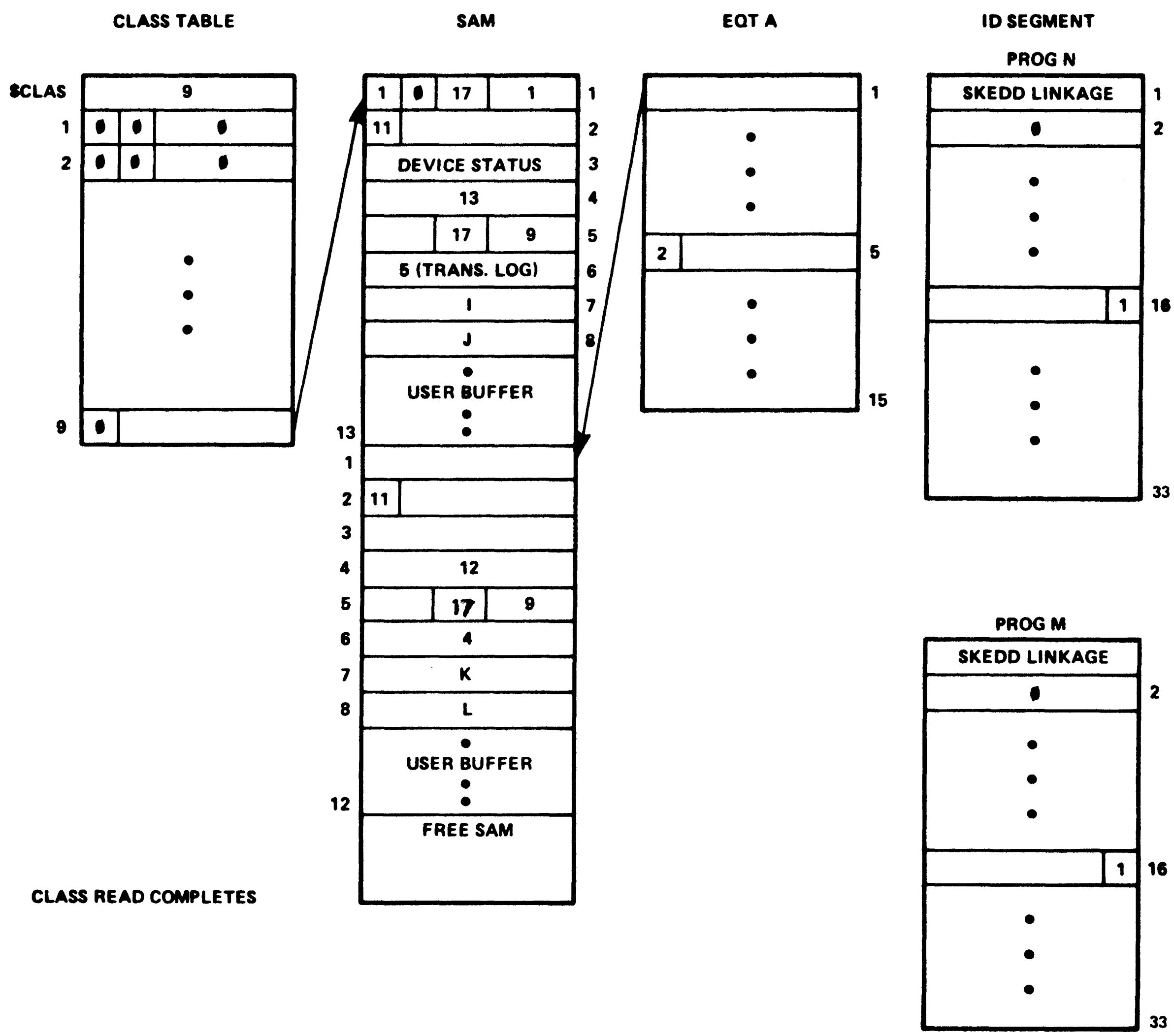
CLASS I/O EXAMPLE



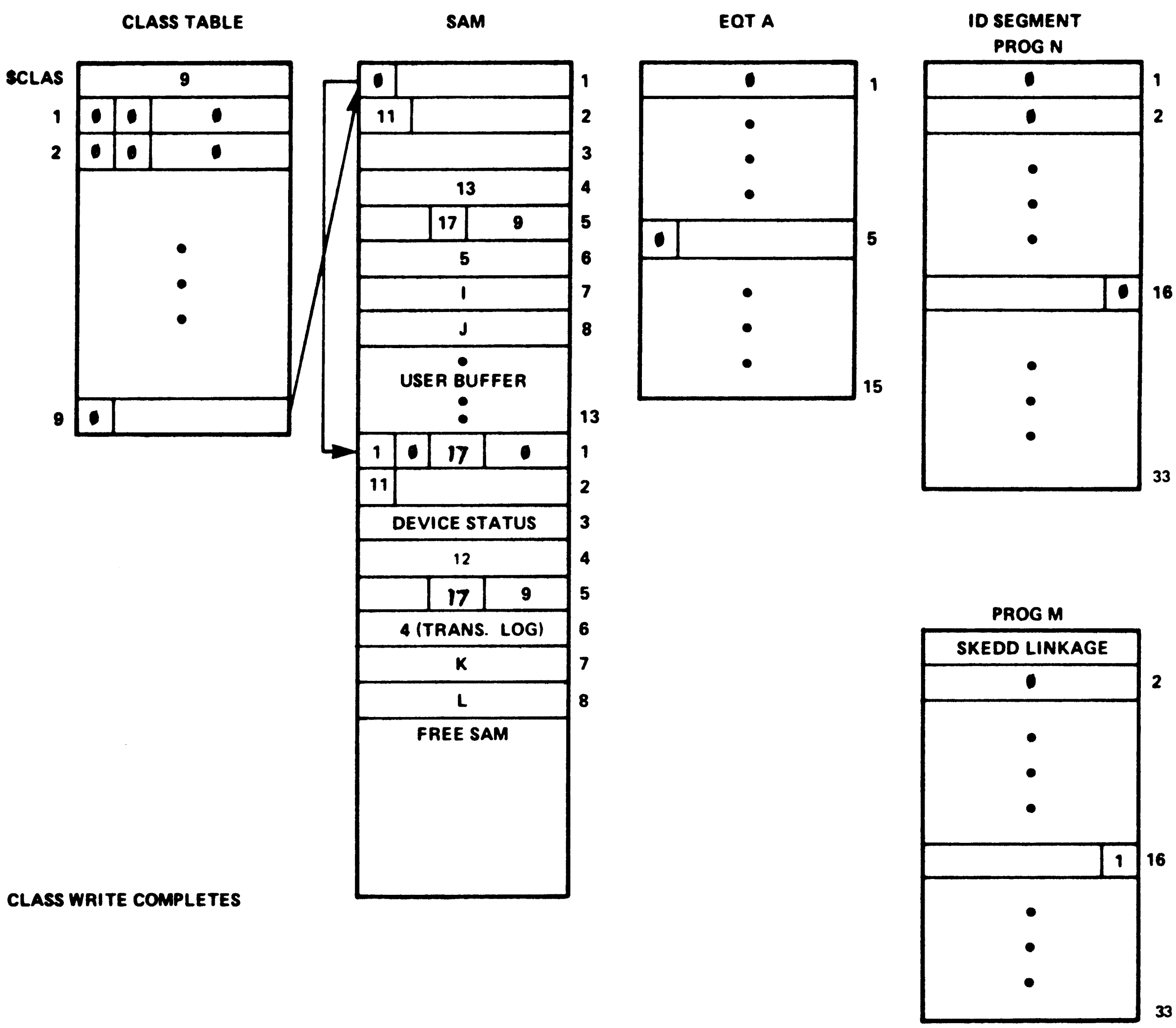
CLASS WRITE USING PREVIOUSLY ALLOCATED CLASS NUMBER

PROGRAM PROGM
•
•
•
•
CALL EXEC (18,LU,IBUF,4,K,L,ICLAS)

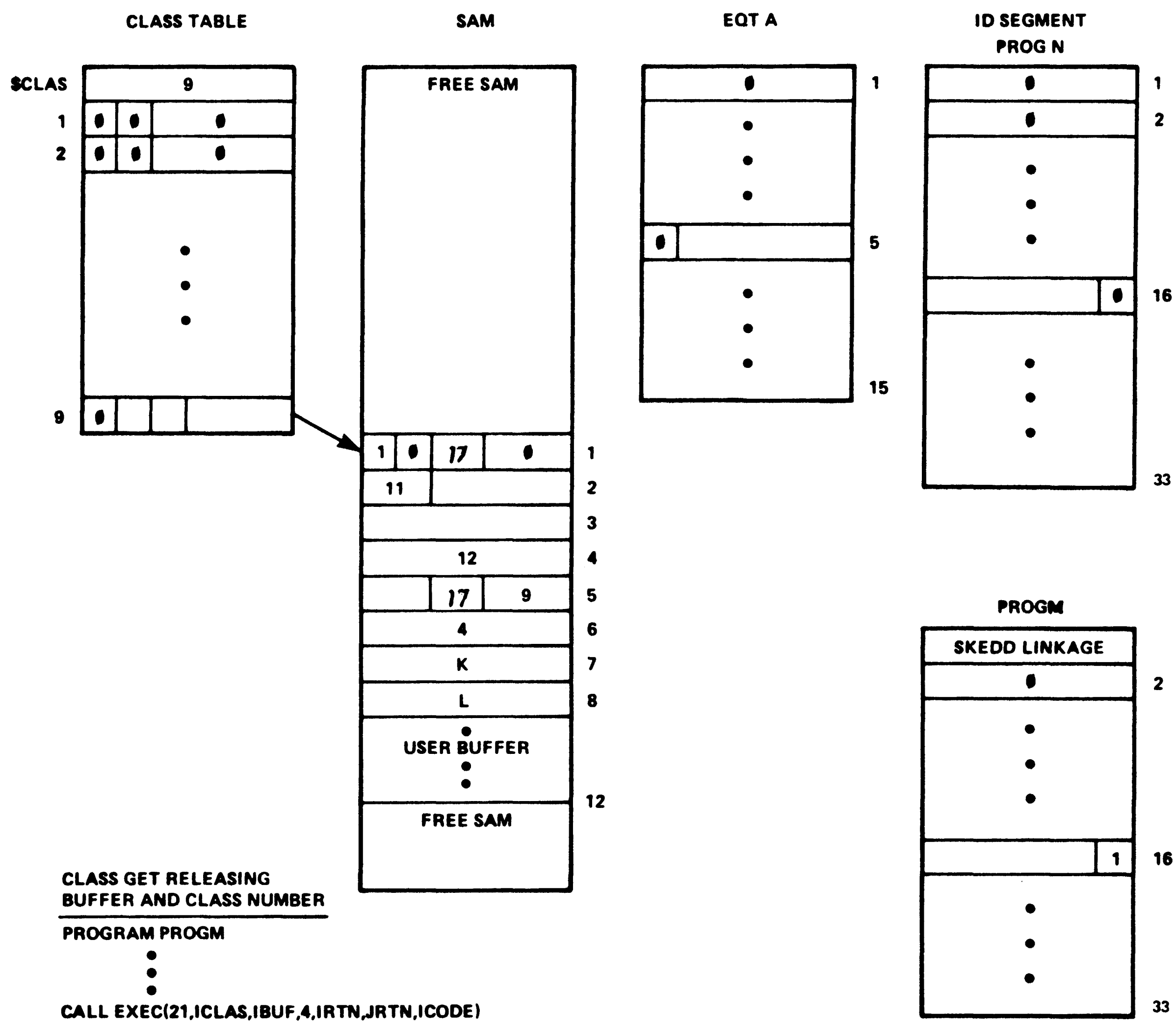
CLASS I/O EXAMPLE



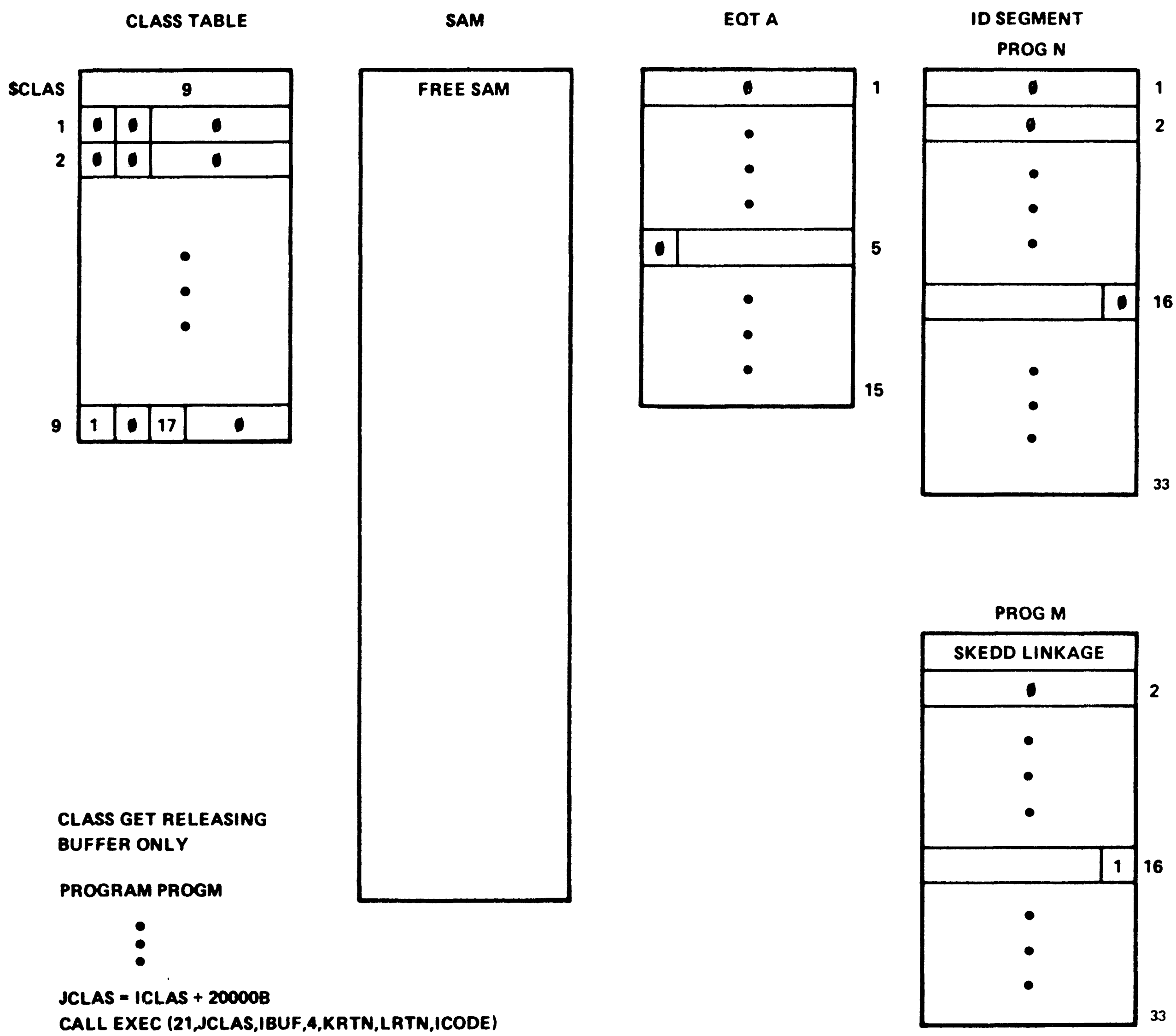
CLASS I/O EXAMPLE



CLASS I/O EXAMPLE



CLASS I/O EXAMPLE



CLASS GET RELEASING
BUFFER ONLY

PROGRAM PROGM

•
•
•

JCLAS = ICLAS + 20000B
CALL EXEC (21,JCLAS,IBUF,4,KRTN,LRTN,ICODE)

RTE MODULES HANDLING
CLASS I/O REQUESTS

Class I/O processing is handled in RTIOC in the following modules:

$IORQ - Initiates class READ, WRITE, CONTROL, and READ/WRITE requests (EXEC 17, 18, 19, and 20)

IOCOM($CON1) - Unlinks the completed request from it's EQT entry, checks for errors and branches to a subsection based on the I/O request type.

C.01(IOCOM) - Links the class request buffer into its class queue and reschedules the program, if any, waiting on the class number with a class GET request.

$GTIO - Processes class GET requests (EXEC 21)

```
CLASS: CLASS TABLE IS AT 040227 WITH 12 ENTRIES!

CLASS: FOLLOWING COMMANDS ARE ACCEPTED:
 DISPLAY,N1,N2,LU  -  DISPLAY STATUS OF CLASS TABLE FOR
                      CLASS NUMBERS N1 THROUGH N2
 LIST,LU           -  LIST CONTENTS OF CLASS TABLE ON LU

 CLEAR             -  CLEAR OUT PENDING CLASS BUFFERS
                      (CLASS NUMBER REQUESTED LATER)
 END               -  END

CLASS: TASK:

                                    GET PROG OR BUFFER PRAMS
CLASS  POSSIBLE OWNERS SECU  #RQ  SIZE  OPT1   OPT2
                                  OCTAL

  12  PRMPT               001    RSPNS
  11  SMP                 010    SPOUT
  10   ** AVAILABLE **
   9   ** AVAILABLE **
   8   ** AVAILABLE **
   7   ** AVAILABLE **
   6   ** AVAILABLE **
   5   ** AVAILABLE **
   4   ** AVAILABLE **
   3   ** AVAILABLE **
   2   ** AVAILABLE **
   1   ** AVAILABLE **
```

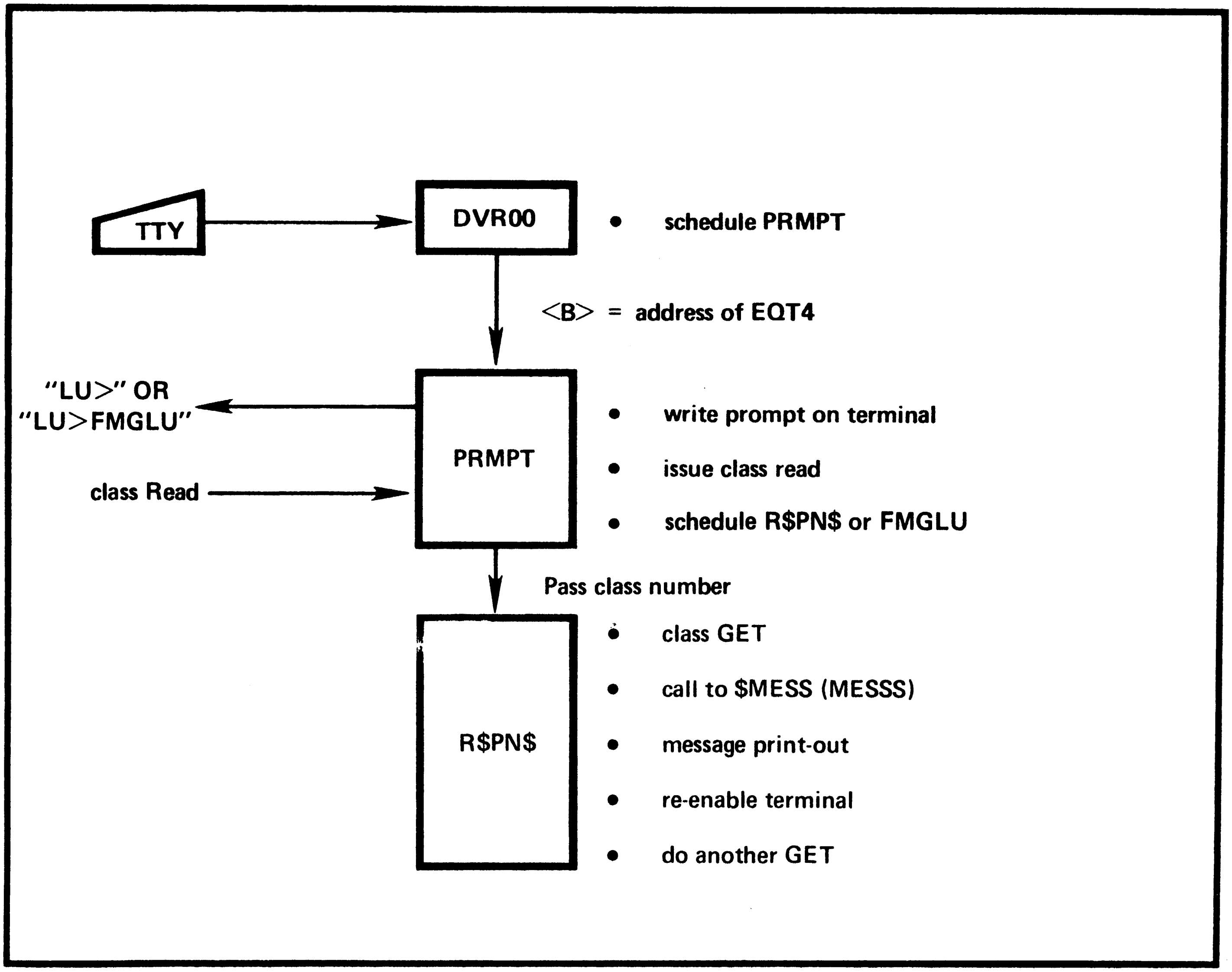
* AVAILABLE IN LOCUS OR SOFTWARE SERVICE KIT

MULTI-TERMINAL MONITOR
(MTM)

MULTI-TERMINAL-MONITOR

- System Console Capability at Peripheral Keyboards

- PRMPT — schedule by interrupt
- RPN — process command input


TTY
DVR00
• schedule PRMPT
<B> = address of EQT4
"LU>" OR
"LU>FMGLU"
class Read
PRMPT
• write prompt on terminal
• issue class read
• schedule RPN or FMGLU
Pass class number
RPN
• class GET
• call to $MESS (MESSS)
• message print-out
• re-enable terminal
• do another GET

MTM FLOW

(Flow may also be easily followed in the listing.)

1. At generation: - Set "PRG,PRMPT" as interrupt table entry for each terminal.

2. At first call to each terminal ("CN,LU,20B", etc.) the terminal driver (DVR00,DVR05,etc.):

   - Moves PRMPT's ID segment number into the terminal's EQT entry.
   - Put the EQT entry address into the interrupt table.
   - Set EQT entry word 5 (bit 1) if the terminal was enabled.

3. Operator depresses a key and the terminal driver's continuation/completion section is entered thru $CIC:

   - Checks that PRMPT is dormant.
   - Puts the address of EQT entry word 4 into ID segment word 11 (XB) of PRMPT.
   - Call $LIST to schedule PRMPT

4. PRMPT is entered:

   - Calls library routine "TRMLU" (also checks for interactive LU-DVR00, 05, or 07) to get the terminals LU from the address of EQT entry word 4.

   - Check that LU and EQT are both up.

   - If LU is locked, configure 9th parameter for EXEC I/O requests to write thru the LU lock. The parameter contains the RN# owner from the RN table and RN# from DRT.

   - Disables the terminal with an EXEC 3 call. This inhibits PRMPT from being scheduled by the terminal´s driver.

   - Issue a zero-length record

   - If FMGXX exists, schedule it, output "LU>FMGXX", and re-enable the terminal.

   - If no FMGXX exists, output "LU> " and get the saved class number from $MTM in Table Area I and issue a class READ on the terminal. Optional parameters are set to terminal LU and address of EQT entry word 4. Schedule RPN immediatley without wait (EXEC 10). It´s probably in class GET suspend so scheduling errors are ignored.

   - Terminate saving resources (pass the class number).

5. RPN is entered:

   - If RPN was dormant, it picks up the class number passed in PRMPT´s schedule request.

```
+-- -  Issues a class GET with wait saving the class number and releasing
|      the class buffer.
|
|   -  Like PRMPT, check for down LU, down EQT, or locked LU.
|
|   -  Check for and process "FL", "BR", or "AB" commands.
|
|   -  Calls $MESS in SCHED to process the operator request.
|
|   -  Prints out any system response from "$MESS" with a class write.
|   -  Enables the terminal with an EXEC 3 call. (re-checks LU in case it
|      was re-assigned)
|
+-- -  Issue another class GET on the same class number.
```

RE-ENTRANT PROCESSING

RE-ENTRANT SUBROUTINES

WHAT?

Subroutines that do not modify their own instructions or local data and therefore may be called before completing its current task.

WHY?

Many executing programs can reference the same re-entrant subroutine on a priority basis (in RTE) and thus save memory space. A shared re-entrant subroutine can be used to manage a shared resource.

HOW?

Variable data associated with the re-entrant subroutine must be stored in separate private areas. This data normally includes arguments, return addresses, and temporary variables. Accomplished in RTE with $LIBR, $LIBX and temporary data blocks (TDB´s).

WHEN?

In subroutines with execution times exceeding one millisecond. For shorter execution times, the overhead time RTE user in saving and restoring temporary data blocks (TDB´s) makes the re-entrant structure unreasonable. In these cases, privileged subroutines should be used.

WHERE?

Normally in shared memory resident library subroutines which are only accessible by MR programs. (Type 6 or 14 programs in RTE). Re-entrant subroutines may also be shared by disc resident programs by placing them in SSGA (Type 30).

WHAT NOT?

Re-entrant subroutines in RTE may not call themselves (recursive).

FORMAT OF RTE RE-ENTRANT
SUBROUTINE

```
      NAM   ENTRY
      EXT   $LIBR,$LIBX

ENTRY NOP          Entry point of routine.
      JSB  $LIBR Call RTE-III to save temporary data.
      DEF  TDB   Address of temporary data.
       .
       .
       .           Program instructions
       .
       .

EXIT  JSB  $LIBX Call RTE-III to restore data.
      DEF  TDB
      DEC  m     m is for routines with two return points in
                 the calling program;  0 specifies the error-
                 point return and 1 the normal return.  For
                 routines with only one return point, m = 0.

TDB   NOP        System control word.
                 0 = Subroutine not in use else;
                     Points to word 2 of ID extension if TDB not in SAM.
                     Points to original TDB location in re-entrant
                     subroutine if TDB has been moved into SAM.

      DEC  n+3   Total length of current block.
      NOP        Return address to calling program.


T1
  .
  ..................Temporary data (n words).
  .
Tn
```

LIMITATIONS:
1. Re-entrant subroutines cannot call themselves (recursion).
2. Re-entrant subroutines can call other re-entrant subroutines
   (and other provileged subroutines).

MAKING A SUBROUTINE RE-ENTRANT UNDER RTE

Suppose we have a subroutine, SUB, with the calling sequence:

```
CALL SUB(A,B,C)
```

and performs the operation:

```
C=(A*5)+(B*3)
```

Non re-entrant version of SUB:

```
        NAM     SUB,T
        ENT     SUB
        EXT     .ENTR
A       NOP
B       NOP
C       NOP
SUB     NOP
        JSB     .ENTR
        DEF     A
        LDA     A,I     GET A
        MPY     D5      A*5
        STA     ATEMP
        LDA     B,I     GET B
        MPY     D3      B*3
        ADA     ATEMP   (A+5)+(B+3)
        STA     C,I     SET C
        JMP     SUB,I   RETURN
        D3      DEC     3
        D5      DEC     5
        ATEMP   NOP
                END
```

To make SUB re-entrant:

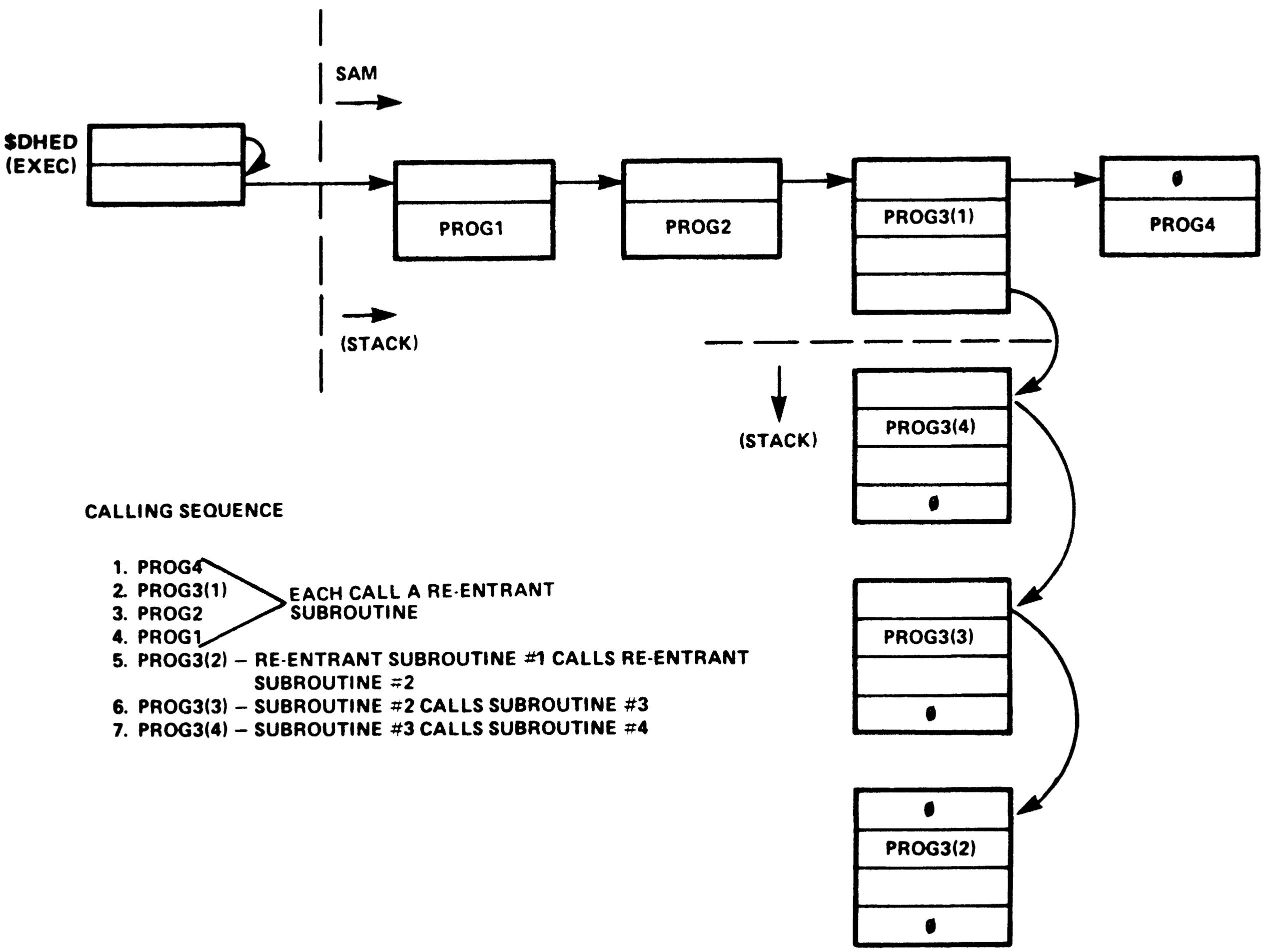
```
          NAM    SUB,14
          ENT    SUB
          EXT    .ENTP,$LIBR,$LIBX
TDB       NOP    (re-entrant data block)
          DEC    1+3+3          (TDB length)
RET       NOP    (return address)
ATEMP     NOP    (local variables)
A         NOP
B         NOP    (argument addresses)
C         NOP
SUB       NOP
          JSB    $LIBR          (go re-entrant)
          DEF    TDB
          JSB    .ENTP          (get argument addresses)
          DEF    A              (argument addresses in TDB)
          STA    RET            (save return address)
          LDA    A,I
          MPY    D5
          STA    ATEMP
          LDA    B,I
          MPY    D3
          ADA    ATEMP
          STA    C,I
          JSB    $LIBX
          DEF    TDB
          DEC    0              (return address adjustment)
          D3     DEC            3
          D5     DEC            5
                 END
```

1. $LIBR - accepts requests to start a re-entrant or privileged subroutine.
2. $LIBX - returns a user to normal processing after a re-entrant or privileged subroutine completes.
3. $LIBR/$LIBX message:
   a. TDB's - when a re-entrant subroutine is re-entered $LIBR moves the TDB into SAM. When the subroutine exits the SAM copy of the TDB is moved back into the subroutine's TDB.
   b. Calling program's ID segment (word 21),
      RE BIT: re-entrant subroutine is now in control
      RM BIT: re-entrant memory (TDB in SAM) must be moved before program is dispatched.
   c. Re-entrant table - an re-entrant table is created in SAM by $LIBR each time a re-entrant call is made.

WORD	RE-ENTRANT TABLE FORMAT (listhead at $DHED+1 in EXEC)
1	Link to next 4 word re-entrant table in SAM (0 = end of list)
2*	ID segment address of user program making re-entrant call
3	Pointer to TDB. Sign bit set if TDB has been moved to SAM.
4	Used to link re-entrant table of first re-entrant subroutine to a group of nested re-entrant subroutines.

* Sign bit set if K+1 words of SAM allocated instead of K words as requested.

RE-ENTRANT TABLE LIST FORMAT



CALLING SEQUENCE

1. PROG4
2. PROG3(1)
3. PROG2
4. PROG1

(1–4: EACH CALL A RE-ENTRANT SUBROUTINE)

5. PROG3(2) – RE-ENTRANT SUBROUTINE #1 CALLS RE-ENTRANT SUBROUTINE #2
6. PROG3(3) – SUBROUTINE #2 CALLS SUBROUTINE #3
7. PROG3(4) – SUBROUTINE #3 CALLS SUBROUTINE #4

RE-ENTRANT PROCESSING EXAMPLE

1. PROGA calls re-entrant subroutine, SUB:

```
PROGRAM PROGA
 .
 .
 .
 .
 .
 .
CALL SUB(A,B,C)
 .
 .
 .
 .
END
```

2. PROGB also calls the re-entrant subroutine, SUB:

```
PROGRAM  PROGB
 .
 .
 .
 .
 .
CALL SUB(A,B,C)
 .
 .
 .
 .
END
```

3. SUB is a re-entrant subroutine which calls $LIBR/$LIBX

4. Sequence of execution will be:
   a. PROGA calls SUB
   b. While in SUB, PROGB begins execution and eventually calls SUB.
   c. PROGB exits SUB and terminates
   d. PROGA resumes execution in SUB and terminates.

NOTE: Both PROGA & PROGB must be MR programs with SUB in the MR library. (Or SUB is in SSGA.)

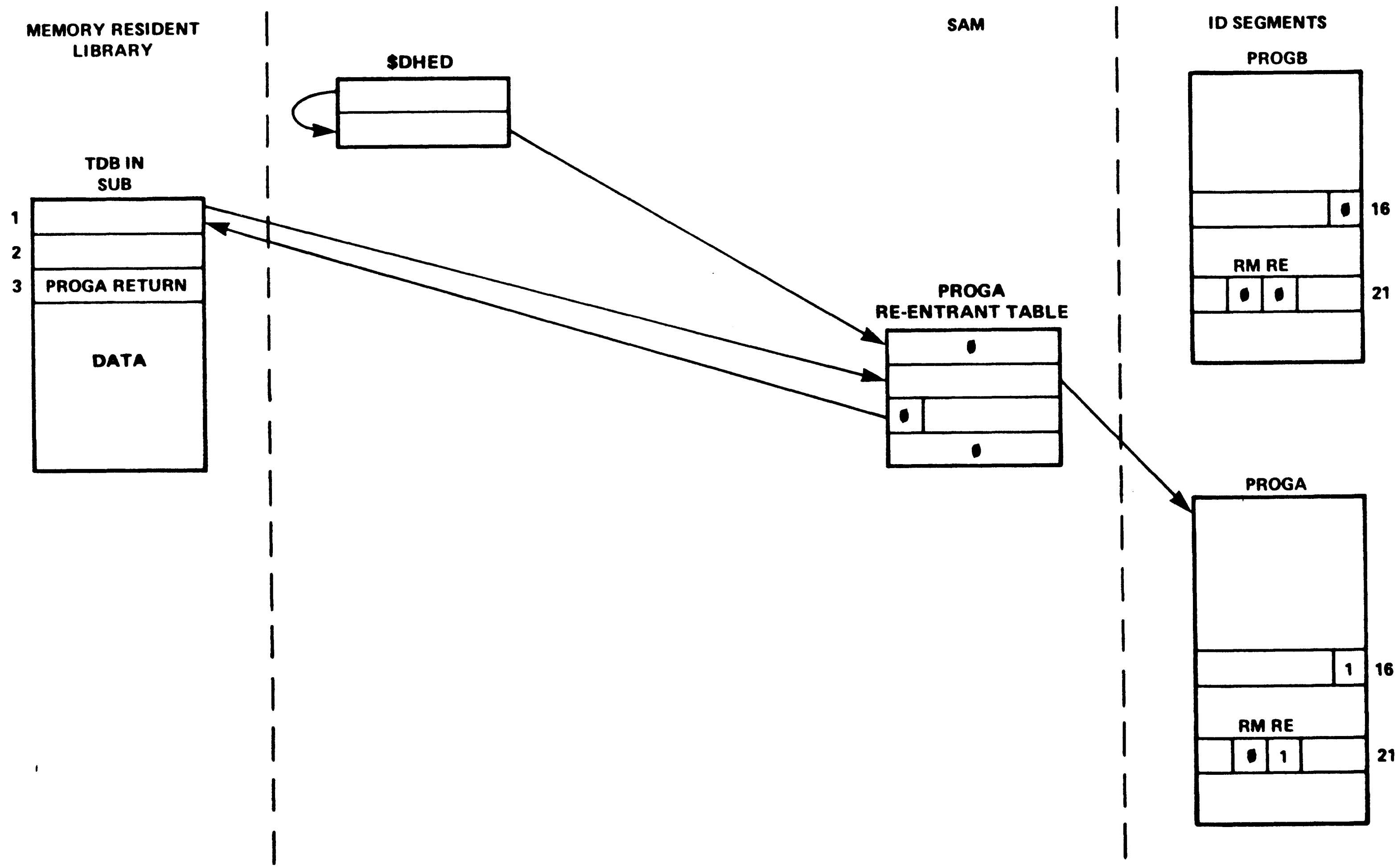
RE-ENTRANT PROCESSING
EXAMPLE FLOW

EXECUTION	NOTES
PROGA	PROGA starts execution.
CALL SUB(A,B,C)	SUB is called by PROGA.
MEMORY PROTECT	Since SUB is in MR library.
TRAP CELL 5(JSB CIC,I)	
$CIC(RTIOC)	Detects MP violation on select code 5. (CIR=5)
$RQST(EXEC)	Detect MP error and not DMS violation. Check that violating instruction was JSB or JSB,I. Determine that destination was MR library and calling program (PROGA) in MR.
LIBRC(EXEC-$LIBR)	Detect that first word of TDB =0 and RE bit=0; therefore SUB is not being re-entered.
($ALC)	Allocate four word re-entrant table. Set up re-entrant table block and link into re-entrant list. Set RE bit in ID segment word 21. Set up words 1 and 3 of TDB.
$RENT(DISPM)	Lower MP fence to start of MR library. Turn on memory protect and interrupt system. Restore SUB's registers and continue execution of SUB at "SUB NOP"+3. (Therefore "JSB $LIBR" never executed!!!).

(The figure on 18-9 summarizes the state of tables/lists at this point)

SUB	SUB continues execution.
PROGB	Begins execution before SUB completes.

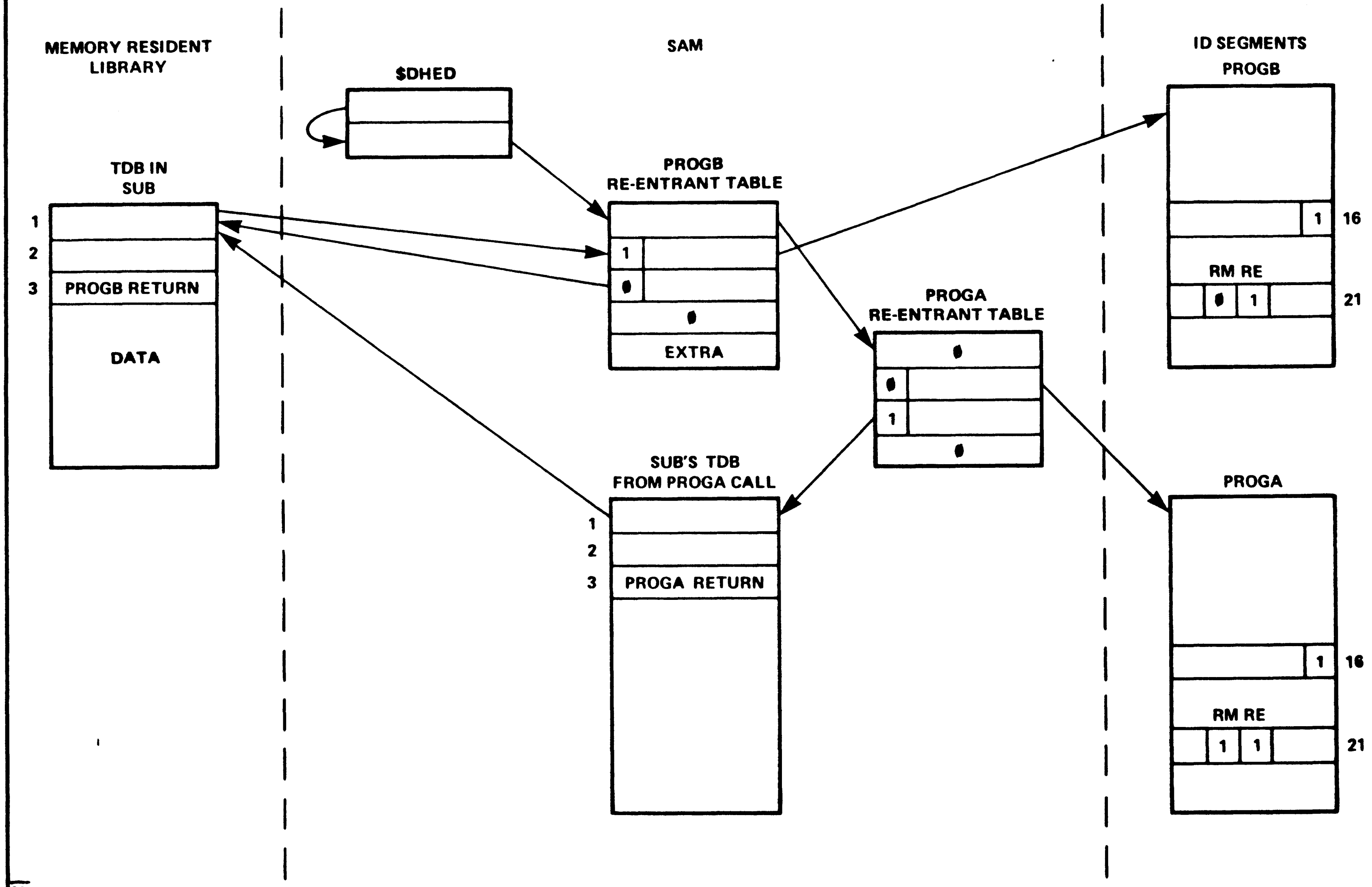
RE-ENTRANT PROCESSING TABLES/LISTS



EXECUTION	NOTES
CALL SUB(A,B,C)	SUB is called by PROGB
MEMORY PROTECT	Since SUB is in MR library
$CIC(RTIOC)	Detect MP violaiton
$RQST(EXEC)	Determine that destination was MR library.
LIBRC(EXEC)	Make sure not a recursive call (TDB word 1,I = PROGBs ID segment address) Detect that TDB word 1 is not zero and therefore
($ALC)	SUB is being re-entered. Allocate four word re-entrant table and buffer for copy of current TDB in SAM.
	Set up re-entrant table.
(MTDB)	Copy current TDB into SAM and set RM bit in PROGA's ID segment.
	Set RE bit in PROGB's ID segment.
	Setup words 1 and 3 of TDB in SUB for PROGB's call.
$RENT(DISPM)	Continue execution of SUB (for PROGB's call) at "SUB NOP"+3.

(See figure 18-11 for current system state)

RE-ENTRANT PROCESSING TABLES/LISTS



18-11

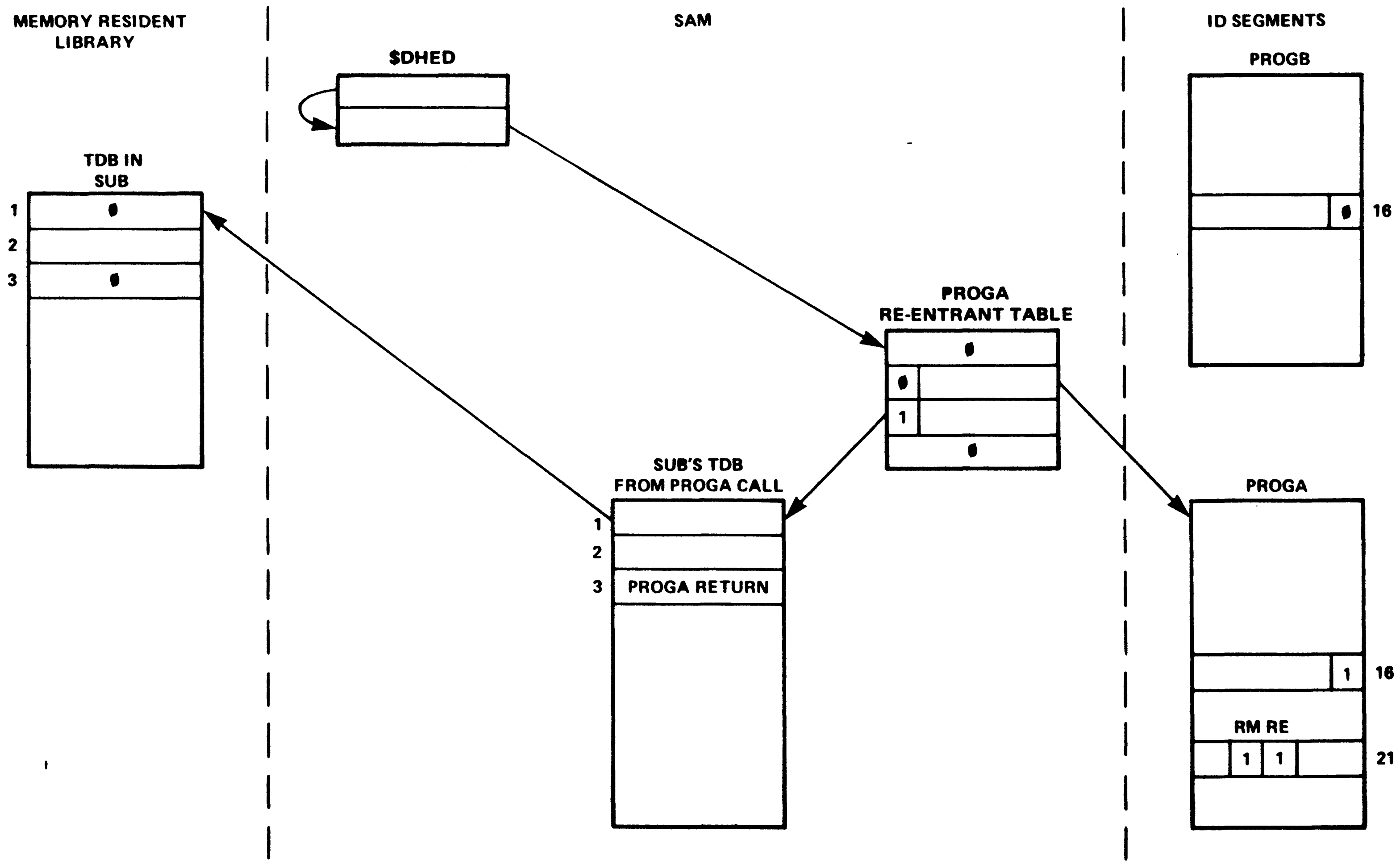
EXECUTION	NOTES
SUB	Complete execution from PROGB call
JSB $LIBX	SUB calls RTE to terminate re-entrant processing
MEMORY PROTECT	
$CIC	
$RQST	Determines that destination was $LIBX
LIBXC(EXEC) ($RTN)	Clear RE bit in PROGB's ID segment. Unlink and return re-entrant table. Set word 1 of SUB's TDB to zero.
$RENT	Continue execution of PROGB after SUB call.
PROGB	PROGB completes

(See figure 18-13 for current system state)

SUB (PROGA)	PROGA is re-scheduled after its TDB is moved back from SAM.

(See figure 18-9 for current system state)

RE-ENTRANT PROCESSING TABLES/LISTS



18-13

RE-ENTRANT I/O

1. Any user program may do re-entrant I/O by issuing an I/O request from a re-entrant subroutine.

2. RTE processes a re-entrant I/O request as follows:

   - RTIOC notes that the RE bit in the ID segment is set and calls $REIO in EXEC.

   - $REIO* verifies that the user's buffer is totally within a TDB and requests SAM for the TDB from $ALC.

   - $REIO moves the TDB into SAM and returns to RTIOC.

   - RTIOC puts the caller in I/O suspend (2) and calls LINK to link the request buffer on the EQT entry.

3. Note that the request buffer address in ID segment word 3 has the "L" bit set to indicate that the buffer is in SAM and not in the user's program.

*Different from the utility subroutine, REIO.

REIO UTILITY SUBROUTINE

CALL REIO(ICODE,ICNWD,IBUFR,IBUFL)

REIO:
Sets up IBUFR in a TDB
Goes re-entrant
Issues the EXEC I/O request
The I/O request is handled by RTE as re-entrant

LIMITIATIONS:
- Read/Write calls only
- Optional parameters not available
- Buffer (IBUFL) 129 words or less.
- Buffer address at least 5 words above the program's load point.

PRIVILEGED SUBROUTINES

- Execute with interrupt system and memory protect turned off
- Used in programs that execute in less that one millisecond
- Format:

```
             NAM ENTRY,6
             EXT $LIBR,$LIBX

      ENTRY NOP                  Entry points to the routine.
            JSB $LIBR            Call RTE-III to disable the interrupt
                                 system and memory protect fence.
            NOP                  Denotes privileged format.
             .
             .
             .
      EXIT  JSB $LIBX            Call RTE-III to return to calling
                                 program and enable interrupts and
                                 memory protect fence.
            DEF ENTRY            Location of return address.
```

- Privileged routines may call other privileged routines but not re-entrant routines.

.ZPRV/.ZRNT

The externals .ZPRV and .ZRNT are treated as "special" entry points in the RTE Disc-Based Operating Systems, in RTE-II, RTE-III and RTE-IV. The RTE Generator modifies the code that is loaded for subroutines that reference these externals, the changes made depend on whether or not the code is loaded into the core resident library (and hencce may be shareable) or if the code is loaded with the program (not shareable), in the latter case the externals are satisfied by replacing the calls to .ZPRV or .ZRNT with an RSS (i.e., .ZPRV,RP,2001). These RP´s are passed to the on-line loader in the same manner as an operators RP command at RTGEN time, thus, the on-line loader can perform the same functions as the RTE-Generator with respect to the externals .ZPRV, .ZRNT, $LIBR, and $LIBX. The following examples should help to illustrate how an assembled subroutine is modified.

NOTE - The capability of handling calls to REIO must also be added for compatibility reasons since the new library references this routine.

The code of .ENTP and .ENTR is included.

AS ASSEMBLED	WHEN "SUB" IN CORE RESIDENT LIBRARY	WHEN "SUB" NOT IN CORE RESIDENT LIBRARY
N O R M A L	P R I V I L E G E D	R O U T I N E
SUB NOP	SUB NOP	SUB NOP
JSB .ZPRV	JSB $LIBR	RSS
DEF LIBX	NOP	DEF LIBX
...	...	...
...	...	...
LIBX JMP SUB,I	LIBX JSB $LIBX	LIBX JMP SUB,I
DEF SUB	DEF SUB	DEF SUB

.ZPRV/.ZRNT CALLING SEQUENCES

P R I V I L E G E D   W I T H   " . E N T R "

```
PARM1 NOP                  PARM1 NOP                 PARM1 NOP
PARM2 NOP                  PARM2 NOP                 PARM2 NOP
SUB   NOP                  SUB   NOP                 SUB   NOP
      JSB .ZPRV                  JSB $LIBR                 RSS
      DEF LIBX                   NOP                       DEF LIBX
      JSB .ENTP                  JSB .ENTP                 JSB .ENTP
      DEF PARM1                  DEF PRAM1                 DEF PRAM1
      ...                        ...                       ...
      ...                        ...                       ...
LIBX  JMP SUB,I            LIBX  JSB $LIBX           LIBX JMP SUB,I
      DEF SUB                    DEF SUB                   DEF SUB
```

N O R M A L   R E - E N T R A N T   R O U T I N E

```
SUB   NOP                  SUB   NOP                 SUB   NOP
      JSB .ZRNT                  JSB $LIBR                 RSS
      DEF LIBX                   DEF TDB                   DEF LIBX
      ...                        ...                       ...
      ...                        ...                       ...
      ISZ SUB                    ISZ SUB                   ISZ SUB
      ISZ TDB+2                  ISZ TDB+2                 ISZ TDB+2
      NOP                        NOP                       NOP
      ...                        ...                       ...
LIBX  JMP SUB,I            LIBX  JSB $LIBX           LIBX  JMP SUB,I
      DEF TDB                    DEF TDB                   DEF TDB
      DEC 0                      DEC 0                     DEC 0
```

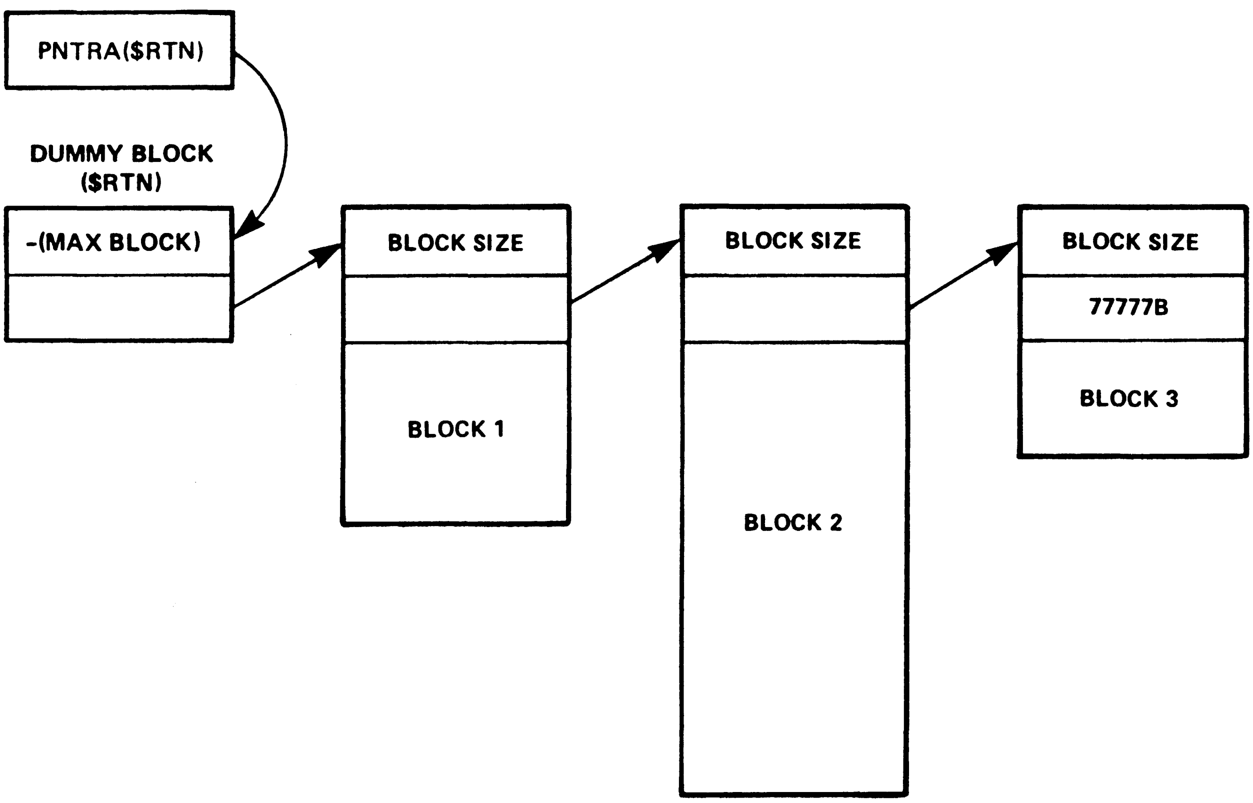
R E - E N T R A N T   W I T H   " . E N T R "

```
PRAM1 NOP                  PRAM1 NOP                 PRAM1 NOP
PRAM2 NOP                  PRAM2 NOP                 PRAM2 NOP
SUB   NOP                  SUB   NOP                 SUB   NOP
      JSB .ZRNT                  JSB $LIBR                 RSS
      DEF LIBX                   DEF TDB                   DEF LIBX
      JSB .ENTP                  JSB .ENTP                 JSB .ENTP
      DEF PRAM1                  DEF PRAM1                 DEF PRAM1
      STA TDB+2                  STA TDB+2                 STA TDB+2
      ...                        ...                       ...
      ...                        ...                       ...
LIBX  JMP TDB+2,I          LIBX  JSB $LIBX           LIBX  JMP TDB+2,I
      DEF TDB                    DEF TDB                   DEF TDB
      DEC 0                      DEC 0                     DEC 0
```

SYSTEM AVAILABLE MEMORY
(SAM)

- RTE-IV generator/configurator sets in base page EQT1 thru EQT6 the address and size of each of the 3 possible pieces of SAM. (SAM default block, SAM extenion, and SAM in Table Area I.)
- At boot-up $STRT maps SAM in contiguous map register allowing up to 5 bad pages in SAM.
- $STRT links each piece of SAM into the SAM free list.
- PNTRA in $ALCM is the free SAM listhead.
- $ALC in $ALCM allocates requested contiguous blocks of free SAM.
- $RTN ($ALCM) returns blocks to the free SAM list.
- As blocks of SAM are returned, only the highest priority program on the memory suspend list (4) is checked for scheduling (1).

FREE SAM LIST



$ALC CALLING SEQUENCE:

```
(P)       JSB $ALC
(P+1)     (# OF WORDS NEEDED)
(P+2)     -RETURN NO MEMORY EVER (A)=-1, (B)=MAX EVER
(P+3)     -RETURN NO MEMORY NOW (A)=0,   (B)=MAX NOW
(P+4)     -RETURN OK   (A)=ADDR,  (B)=SIZE OR SIZE+1
```

TO FIND OUT HOW LARGE A BLOCK MAY EVER BE ALLOCATED:

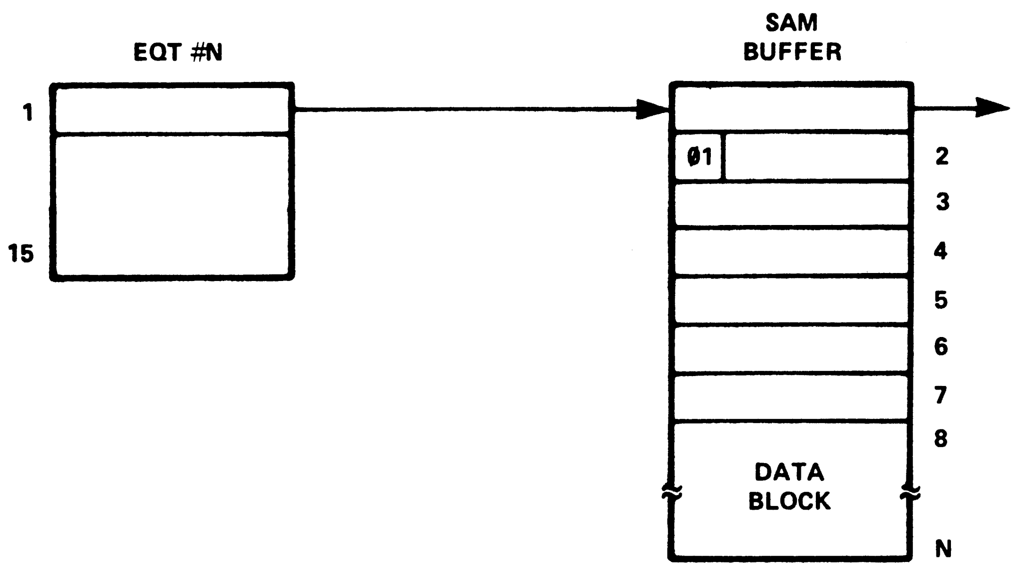
```
JSB $ALC
DEC 32767
```

$RTN CALLING SEQUENCE:

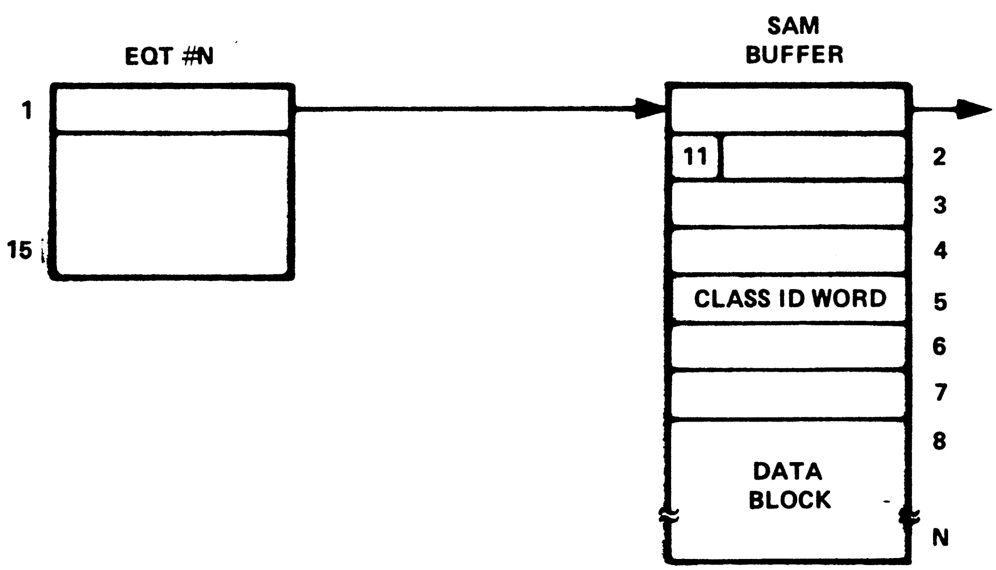
```
(P)       JSB $RTN
(P+1)     (FWA OF BUFFER)
(P+2)     (# OF WORDS RETURNED)
(P+3)     -RETURN-  (ALL REGISTERS DESTROYED)
```

SAM USERS

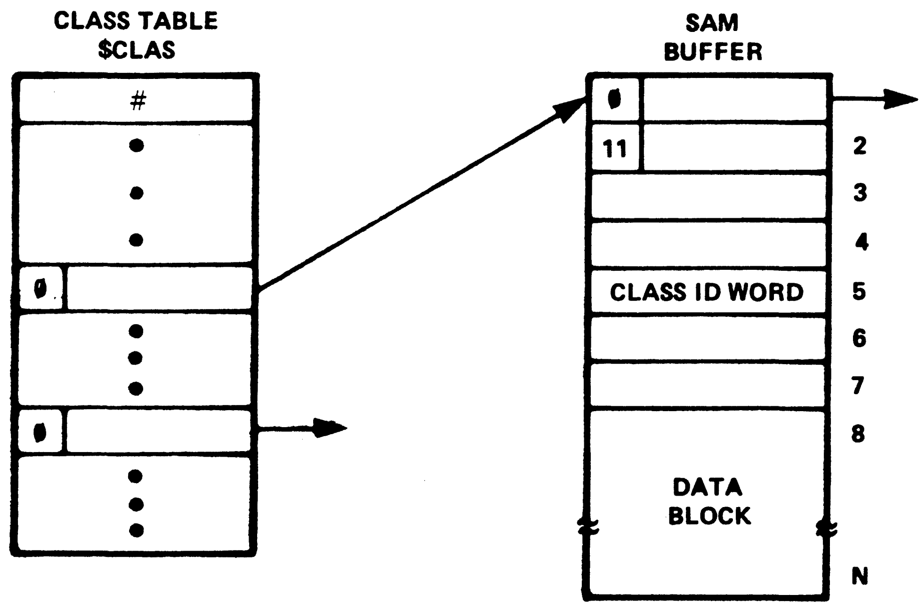
I. AUTOMATIC OUTPUT BUFFERING I/O REQUESTS



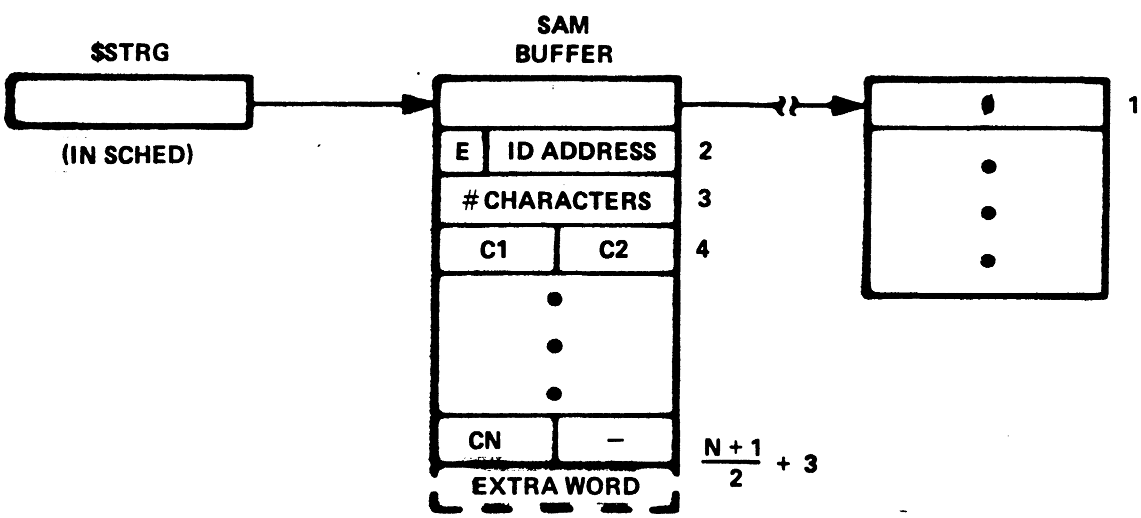
II. CLASS I/O REQUESTS



III. CLASS QUEUE



IV. PARAMETER STRINGS FROM "ON" OR "RU" COMMANDS

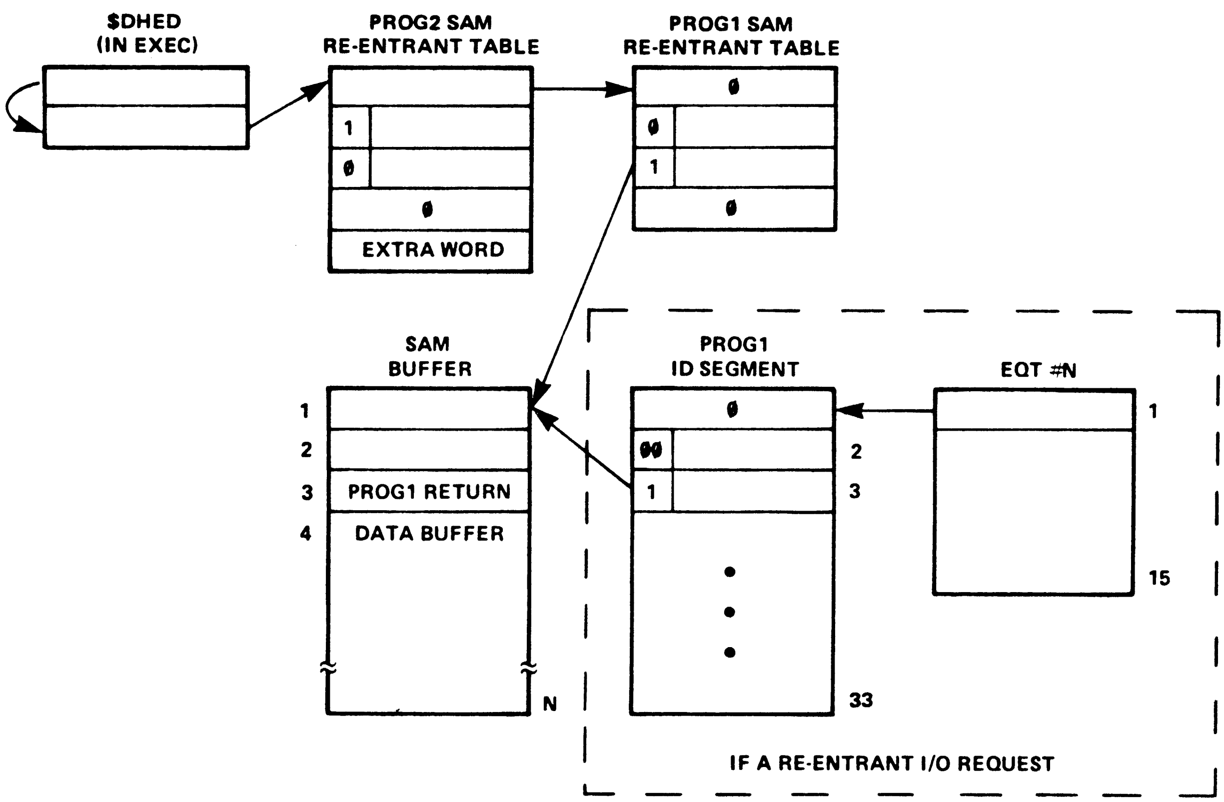


E = 1 IF $ALC HAD TO ALLOCATE AN EXTRA WORD IN SAM

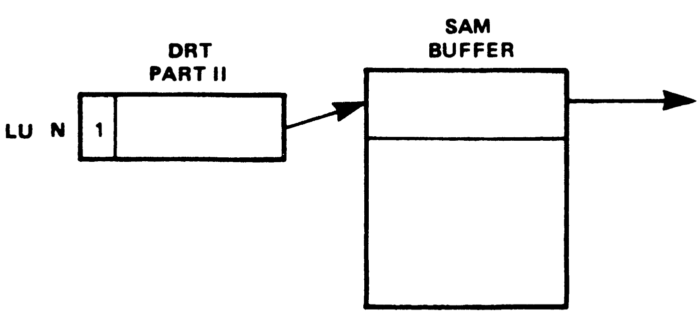
V. RE-ENTRANT PROCESSING

a. ID EXTENSIONS CREATED EACH TIME A RE-ENTRANT ROUTINE IS CALLED.

b. MOVED TDB'S (TEMPORARY DATA BLOCK) BECAUSE OF RE-ENTRANT SUBROUTINES BEING RE-ENTERED OR AN I/O REQUEST FROM A RE-ENTRANT ROUTINE WITH THE REQUEST BUFFER IN THE TDB.



VI. DOWN DEVICE BUFFERS



SAM USAGE PERMITS

- I/O WITHOUT WAIT
- PROGRAM TO PROGRAM COMMUNICATION
- PROGRAM SWAPPING WHILE PROGRAM WAITS FOR I/O COMPLETION
- A MEMORY POOL USEABLE BY MULTIPLE USERS ON AN "AS NEEDED" BASIS

I/O DRIVERS

WHAT IS A DRIVER?

A DRIVER IS A ROUTINE WHICH RESIDES IN THE SYSTEM AND IS RESPONSIBLE FOR ALL DATA TRANSFERRED BETWEEN THE I/O DEVICE AND COMPUTER DURING THE INITIATION, CONTINUATION, AND COMPLETION PHASES OF I/O.

HOW IS A DEVICE STARTED?

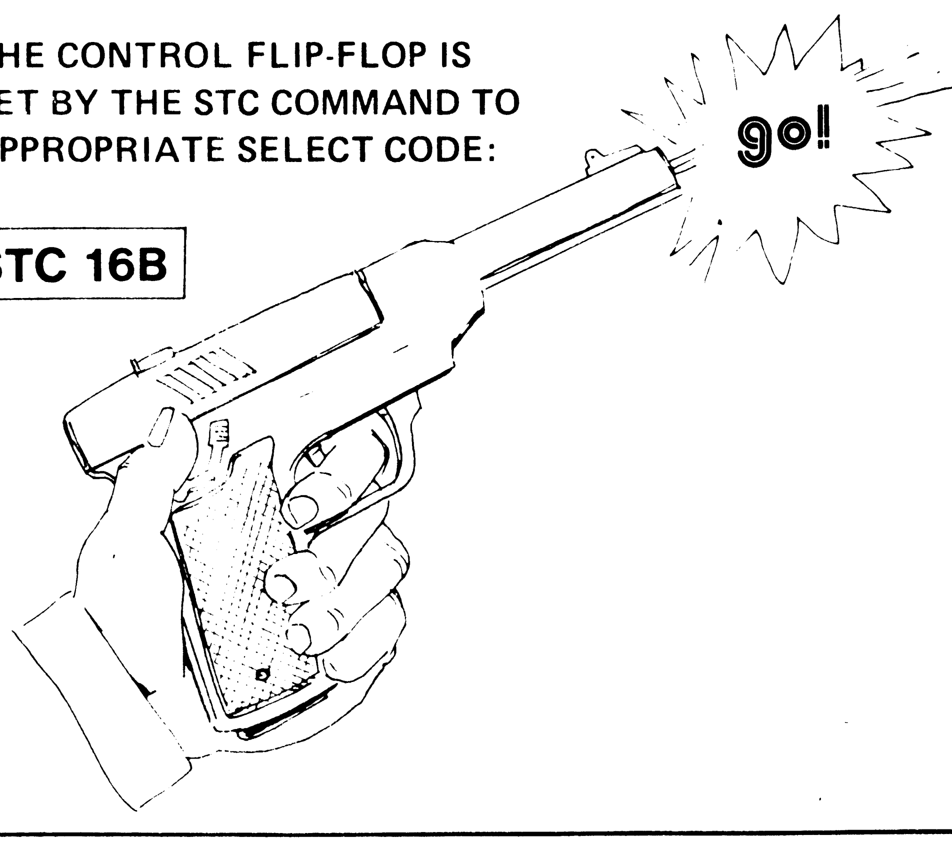
* BY A CONTROL FLIP-FLOP ON THE INTERFACE CARD.

* WHEN SET, THE CONTROL FLIP-FLOP GENERATES A START COMMAND.

* THE DEVICE THEN PERFORMS ONE OPERATION CYCLE.

* THE CONTROL FLIP-FLOP IS SET BY THE STC COMMAND TO APPROPRIATE SELECT CODE:

STC 16B



WHAT HAPPENS TO DATA ?

- THE INTERFACE CARD HAS A STORAGE BUFFER.

- TO INPUT DATA TO THE PROCESSOR FROM THE BUFFER USE:

  LIA 16B

- TO OUTPUT DATA FROM THE PROCESSOR TO THE BUFFER USE:

  OTA 16B

WHEN IS DATA READY ?

- WHEN THE DEVICE CYCLE IS COMPLETE, A FLAG FLIP-FLOP IS SET ON THE I/O CARD.

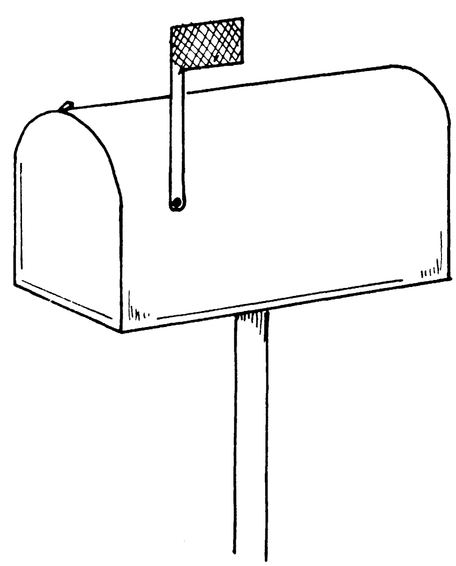
  TO CHECK THE FLAG FLIP-FLOP USE

  SFS 16B

- IF THE FLAG IS ALREADY SET – CLEAR THE FLAG BEFORE STARTING AN OPERATION.

- TO ELIMINATE TIMING CONFLICTS USE "SET CONTROL" AND "CLEAR FLAG" IN THE SAME COMMAND.

  STC 16B, C



AN INPUT AND OUTPUT EXAMPLE

ASSUME SELECT CODE 16 FOR INPUT AND 17
FOR OUTPUT:

INPUT

```
STC   16B,C     CLEAR FLAG AND START OPERATION
SFS   16B       OPERATION FINISHED ?
JMP   *-1       NO, KEEP CHECKING
LIA   16B       YES, GET DATA
```

OUTPUT

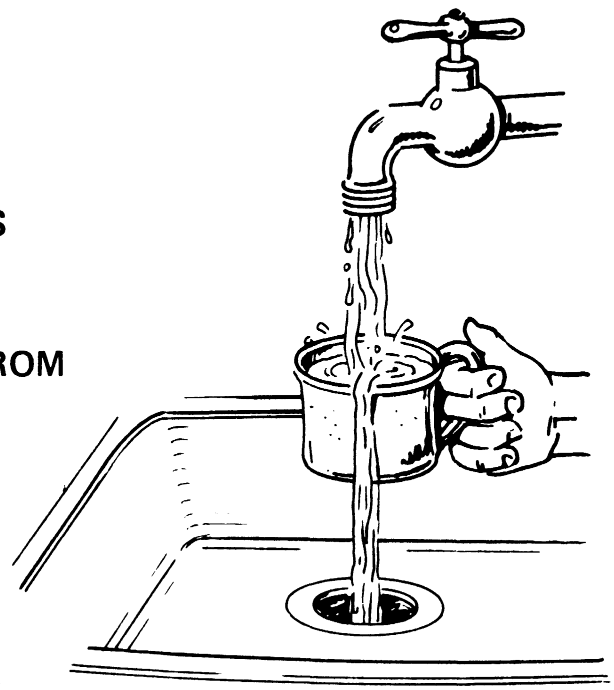
```
OTA   17B       OUTPUT DATA TO I/O CARD
STC   17B,C     CLEAR FLAG AND START OPERATION
SFS   17B       FINISHED ?
JMP   *-1       NO, KEEP CHECKING
  •             YES, CONTINUE
  •
  •
```

ASYNCHRONOUS DRIVERS

- ASYNCHRONOUS DEVICES MAKE ONE DATA TRANSFER AND WAIT FOR COMPUTER ACKNOWLEDGEMENT (HANDSHAKE).

- DATA IS TRANSFERRED ONLY AS FAST AS THE DRIVER REQUESTS IT. (CRT TERMINAL OR LINE PRINTER)

SYNCHRONOUS DRIVERS



▶SYNCHRONOUS DEVICES DO NOT WAIT FOR DATA TRANSFER ACKNOWLEDGEMENT FROM THE COMPUTER.

▶IF THE DRIVER DOESN'T KEEP UP, DATA IS LOST.

▶SINCE THE DEVICE KEEPS RUNNING ONCE IT HAS BEEN STARTED, DCPC SHOULD BE USED. (MAGNETIC TAPE OR DISC)

STANDARD RTE DRIVERS

- TWO SECTIONS

  INITIATION

  CONTINUATION/COMPLETION

- INTERFACE WITH SYSTEM MODULES

  $CIC

  IOC

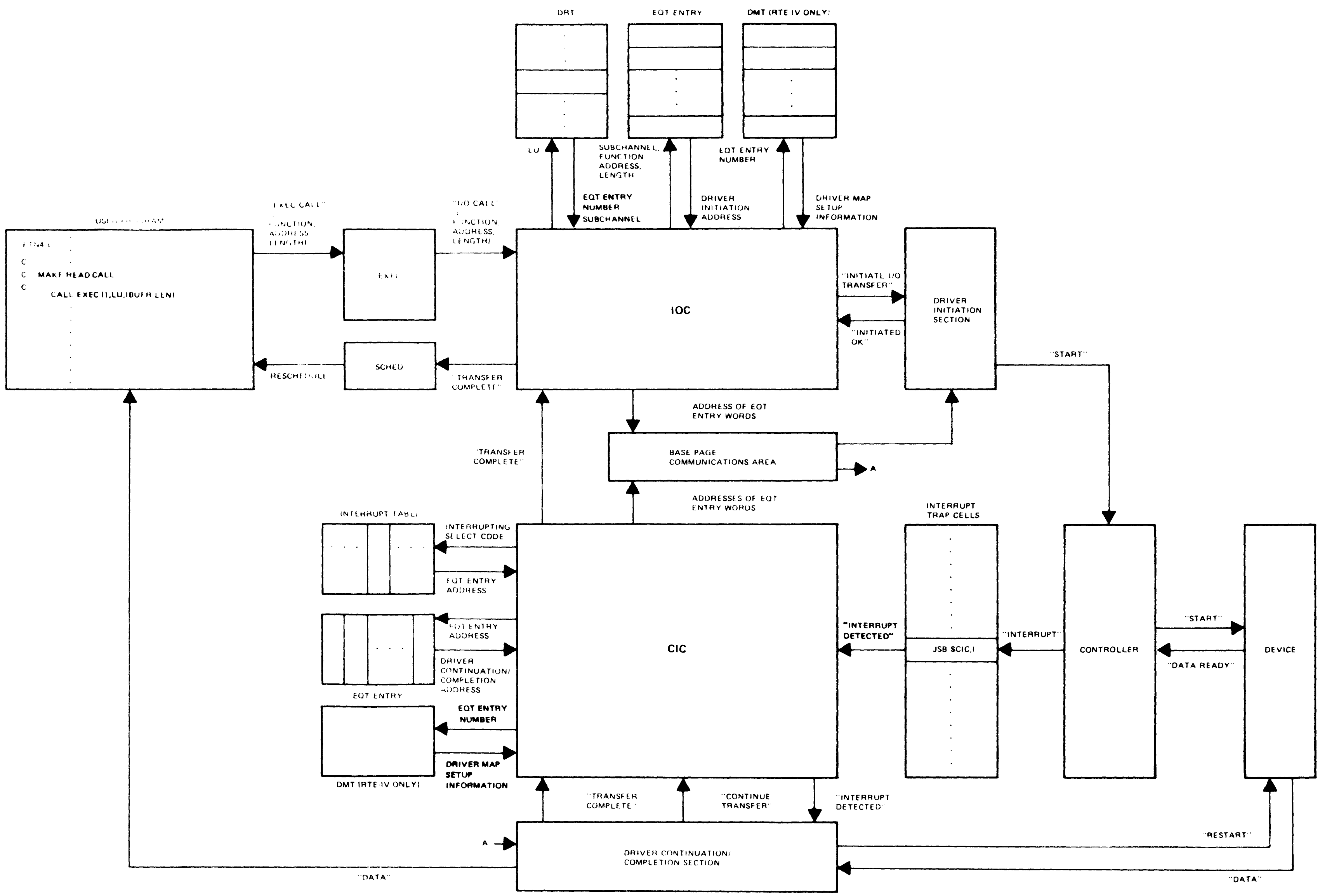
- USE DATA AREAS

  EQUIPMENT TABLE

  SYSTEM BASE PAGE COMMUNICATION AREA

- NORMALLY OPERATE WITH INTERRUPT SYSTEM OFF

UNBUFFERED EXEC READ REQUEST FLOW



WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O REQUEST LIST POINTER <C>														
2	R	DRIVER "INITIATION" SECTION ADDRESS <A>														
3	R	DRIVER "CONTINUATION/COMPLETION" SECTION ADDRESS <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	SUBCHANNEL #<C>					I/O SELECT CODE # <A>					
5	AV <F>		EQUIPMENT TYPE CODE <A>						STATUS <E>							
6	CONWD (CURRENT I/O REQUEST WORD) <C>															
7	REQUEST BUFFER ADDRESS <C>															
8	REQUEST BUFFER LENGTH <C>															
9	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
10	TEMPORARY STORAGE <D> OR OPTIONAL PARAMETER <C>															
11	TEMPORARY STORAGE FOR DRIVER <D>															
12	TEMPORARY STORAGE FOR DRIVER <D> OR EQT EXTENSION SIZE, IF ANY <A>															
13	TEMPORARY STORAGE FOR DRIVER <D> OR EQT EXTENSION STARTING ADDRESS, IF ANY <A>															
14	DEVICE TIME-OUT RESET VALUE <B>															
15	DEVICE TIME-OUT CLOCK <C>															

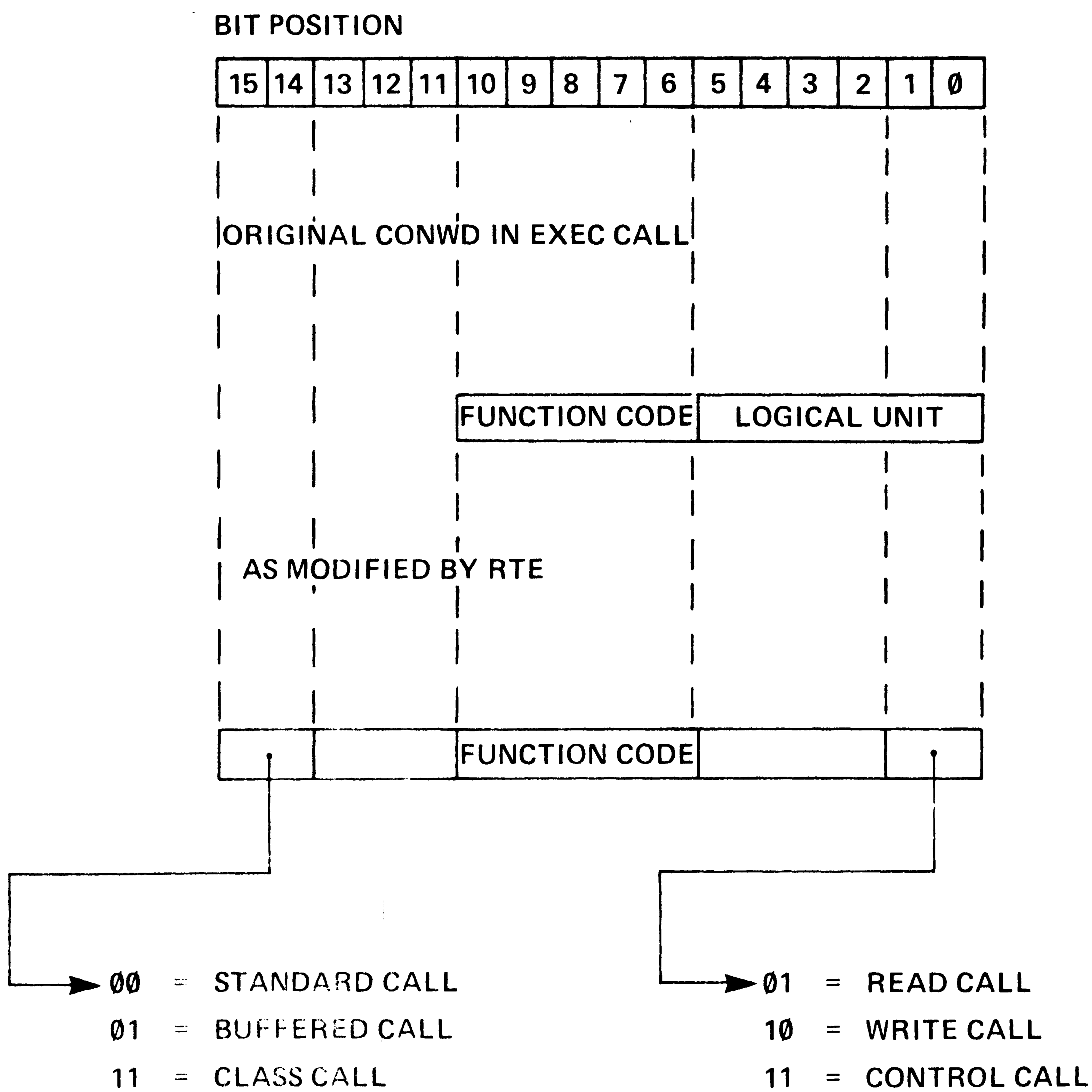
WHERE THE LETTERS IN BRACKETS (<>) INDICATE THE NATURE OF EACH DATA ITEM, AS FOLLOWS:

- <A> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); NEVER CHANGES.
- <B> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); CAN BE CHANGED ON-LINE.
- <C> = SET UP OR MODIFIED AT EACH I/O INITIALIZATION.
- <D> = AVAILABLE FOR USE AS TEMPORARY STORAGE BY DRIVER.
- <E> = CAN BE SET BY DRIVER.
- <F> = MAINTAINED BY SYSTEM.

AND WHERE:

- R = (RESERVED FOR SYSTEM USE)
- I/O REQUEST LIST POINTER = POINTER TO LIST OF REQUESTS QUEUED UP ON THIS EQT ENTRY. FIRST ENTRY IN LIST IS CURRENT REQUEST IN PROGRESS; ZERO IF NO REQUESTS.
- D = 1 IF DCPC REQUIRED
- B = 1 IF AUTOMATIC OUTPUT BUFFERING USED
- P = 1 IF DRIVER IS TO PROCESS POWER FAIL
- S = 1 IF DRIVER IS TO PROCESS TIME-OUT
- T = 1 IF DEVICE TIMED OUT (SYSTEM SETS TO ZERO BEFORE EACH I/O REQUEST)
- SUBCHANNEL # = LAST SUBCHANNEL ADDRESSED

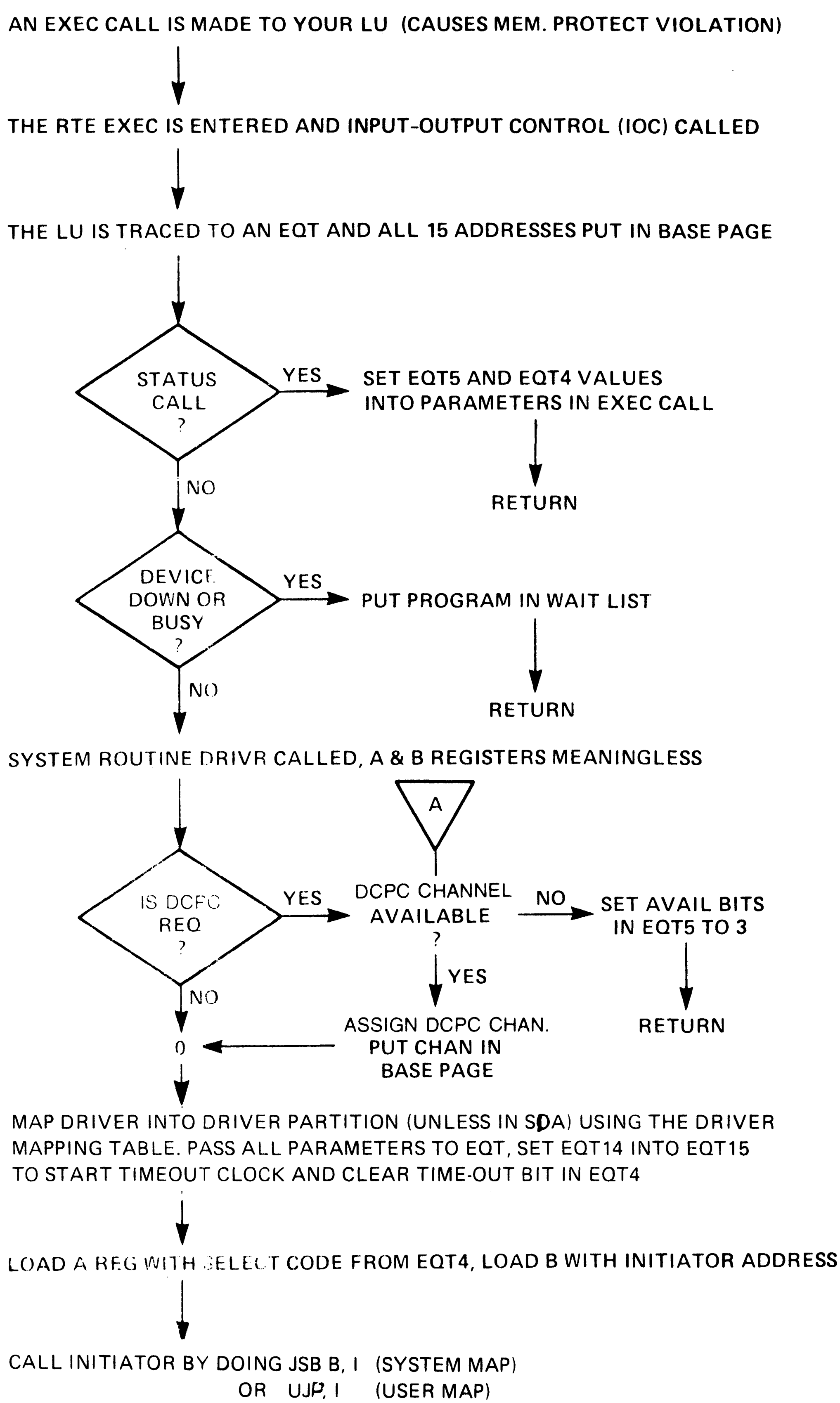
STRUCTURE OF EQT WORD #6
AS SET BY RTE



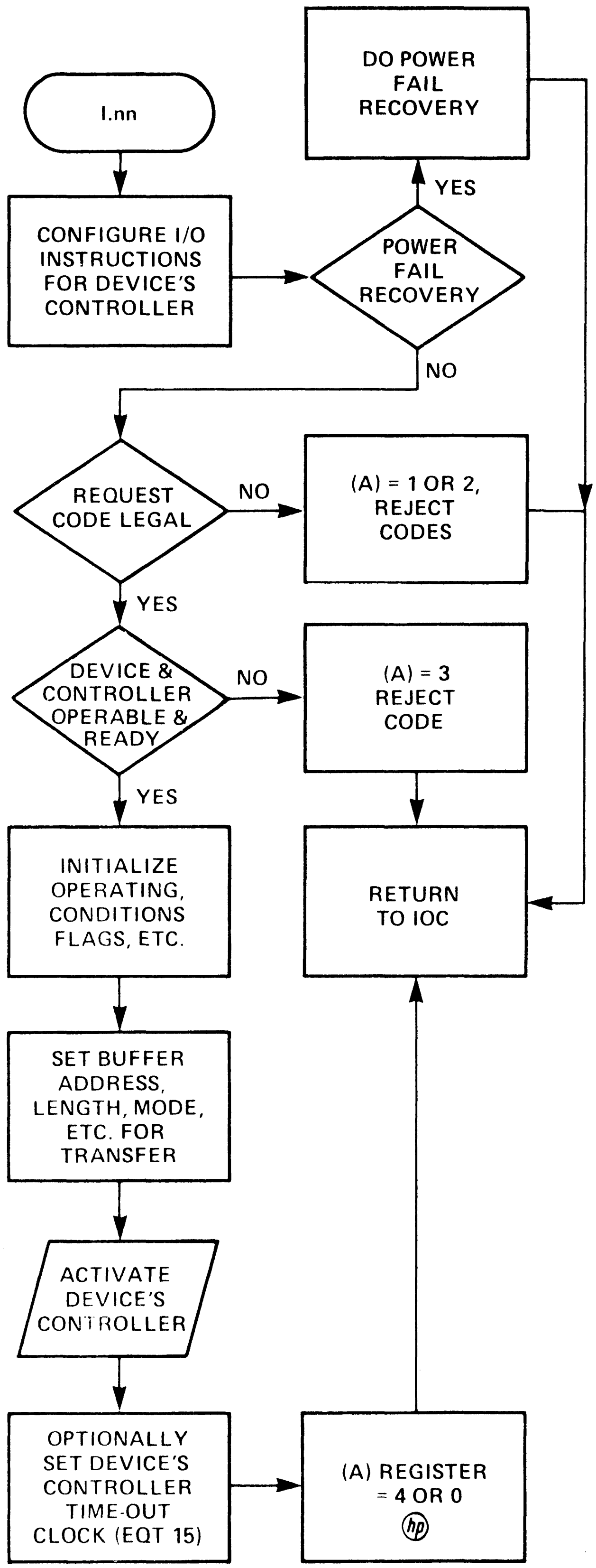
BASE PAGE COMM. AREA CONTENTS

OCTAL LOCATION	CONTENTS	DESCRIPTION
• • •		
01650	EQTA	Address of Equipment Table (EQT)
01651	EQT#	Number of EQT entries
01652	DRT	Address of Device Reference Word 1 Table
01653	LUMAX	Number of logical units (in Device Reference Table)
01654	INTBA	Address of Interrupt Table
01655	INTLG	Number of Interrupt Table entries
01656	TAT	Address of Track Assignment Table (disc-based systems only)
01657	KEYWD	Address of keyword block
01660 01661 01662 01663 01664 01665 01666 01667 01670 01671 01672	EQT1 EQT2 EQT3 EQT4 EQT5 EQT6 EQT7 EQT8 EQT9 EQT10 EQT11	Addresses of first 11 words of current EQT entry (see location of 01771 for last 4 words)
01673	CHAN	Current DCPC Select Code (6 or 7)
• • •		
01717	XEQT	ID segment address of current program
• • •		
01737	DUMMY	I/O channel of privileged interrupt card (0 if none)
• • •		
01770	MPTFL	Memory Protect On/Off (0/1) flag.
01771 01772 01773 01774	EQT12 EQT13 EQT14 EQT15	Addresses of last 4 words of current EQT entry

WHAT HAPPENS WHEN A PROGRAM CALLS YOUR DRIVER?



RTE DRIVER INITIATION SECTION



(hp) IF A = 4 SET B = TRANSMISSION LOG

DRIVER RULES FOR INITIATOR RETURN

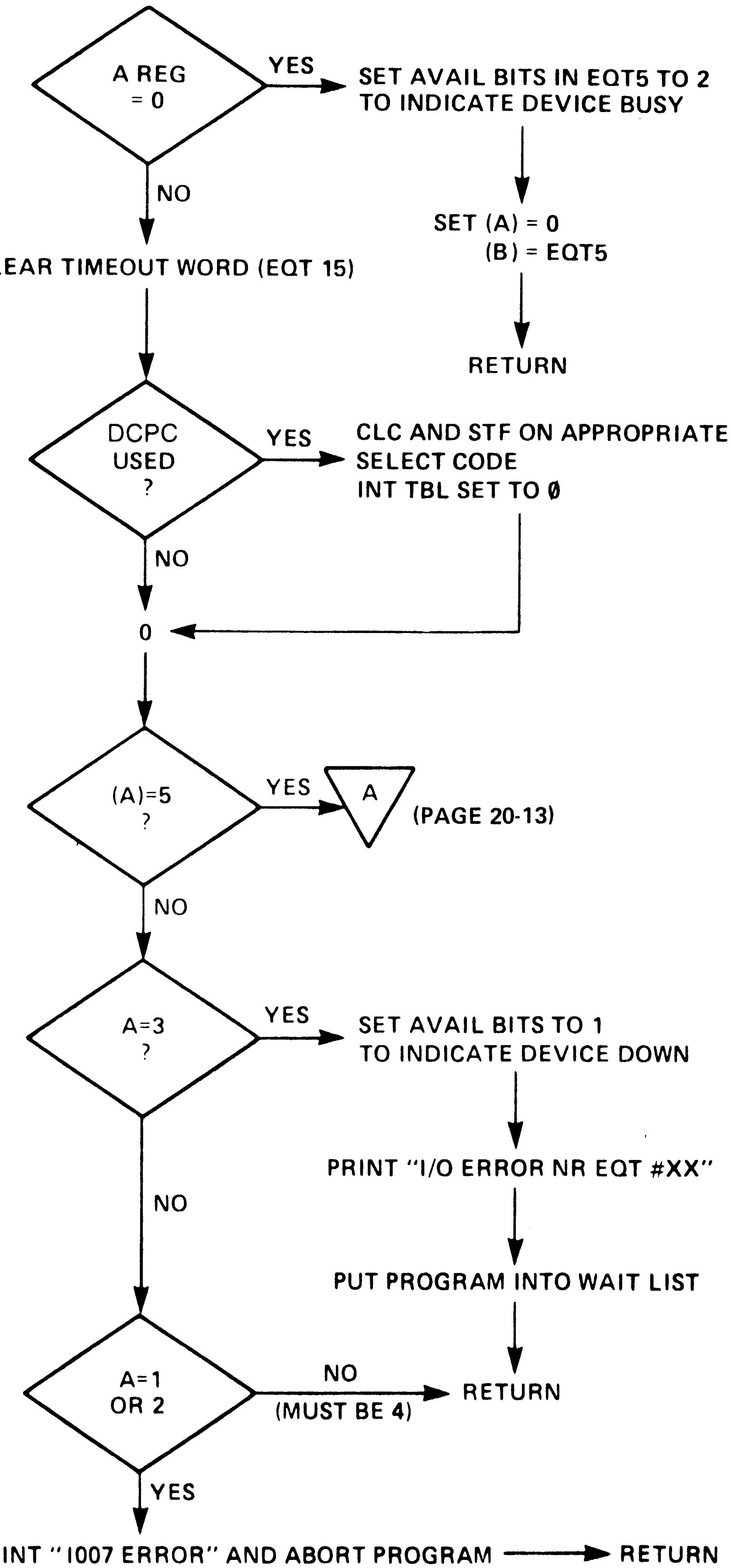
COMMUNITY CHEST

RETURN TO JAIL.
DO NOT PASS GO.
DO NOT COLLECT $200.

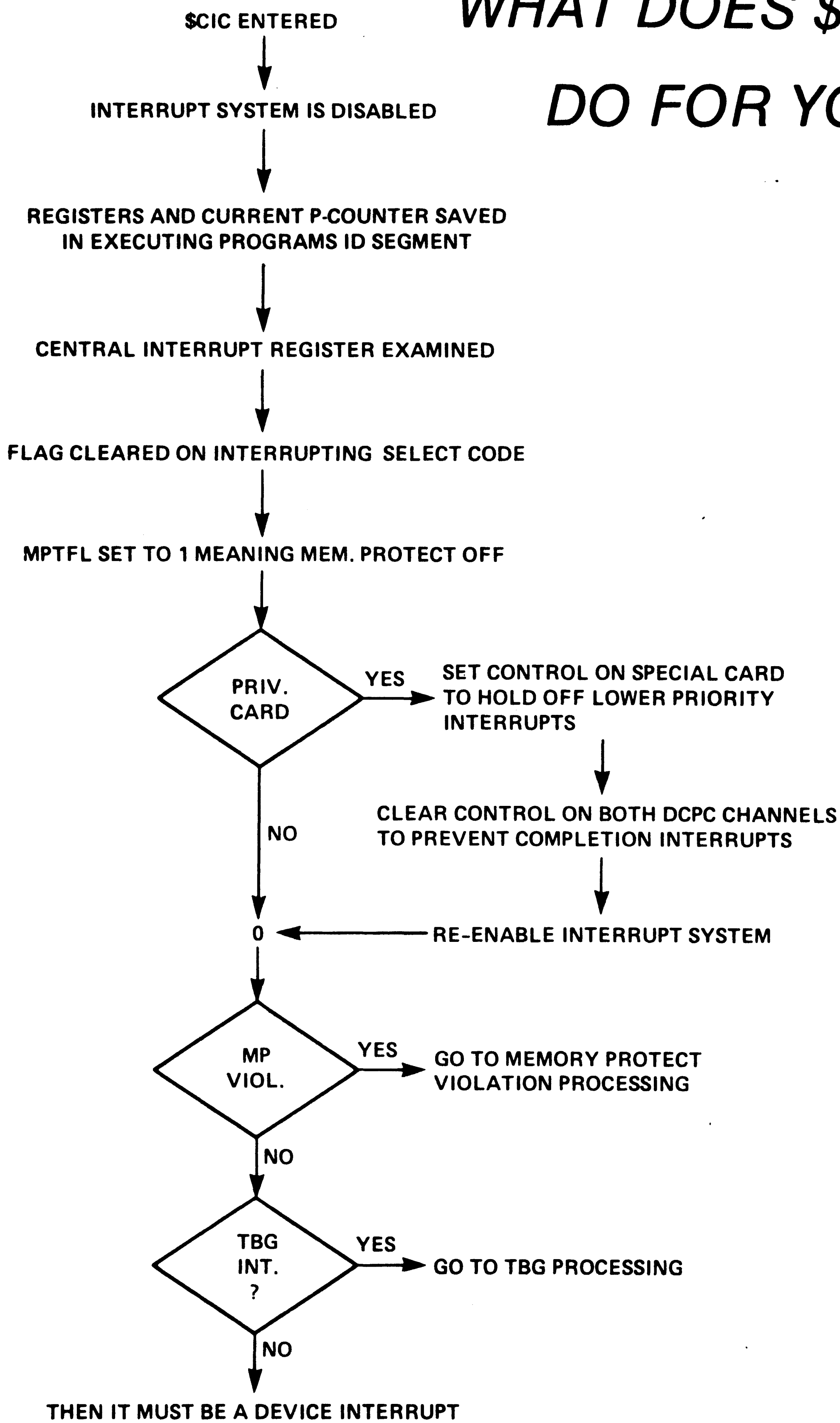
SET THE A-REGISTER TO INDICATE INITIATION OR REJECTION

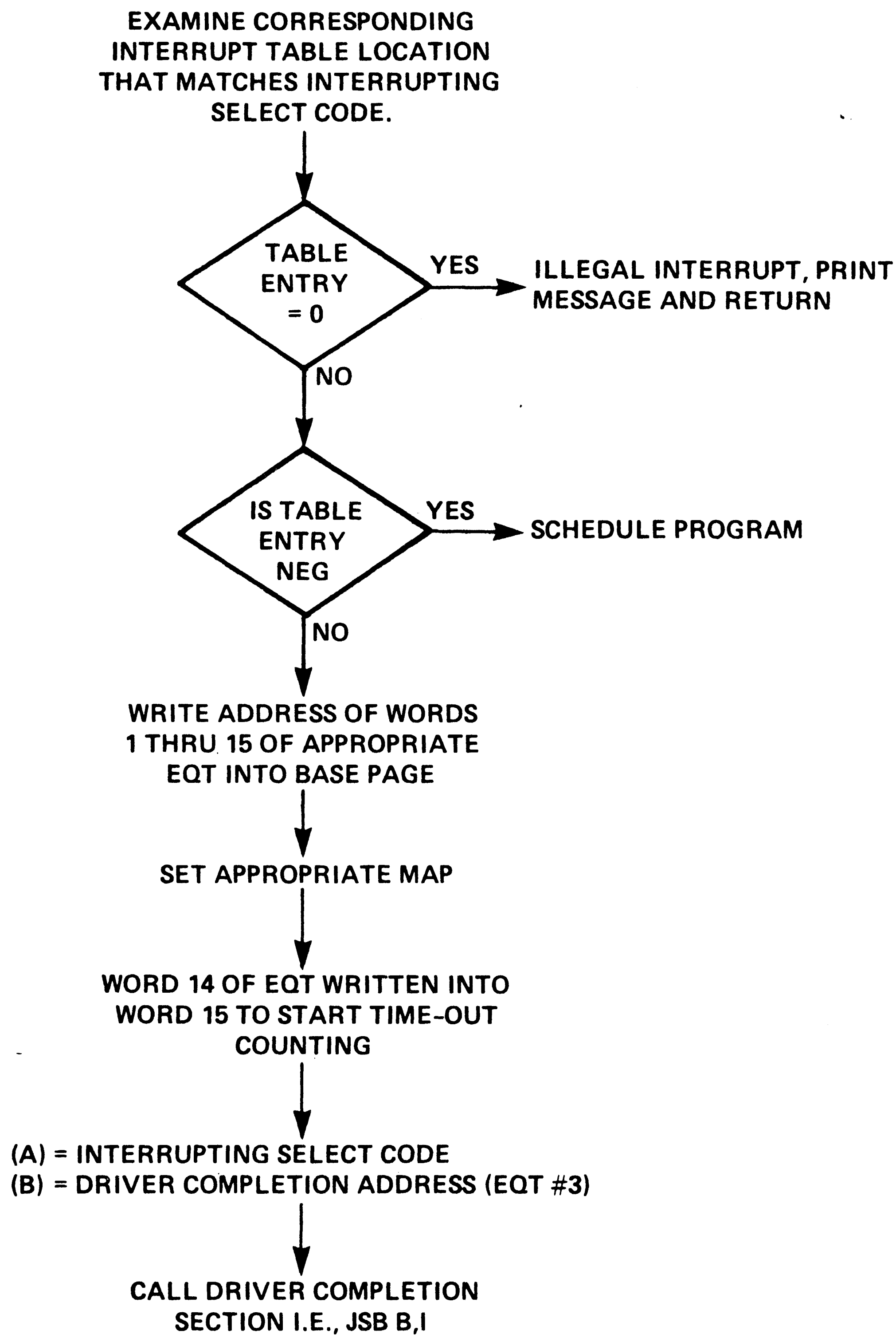
A = 0	OPERATION INITIATED
= 1	READ OR WRITE ILLEGAL
= 2	CONTROL REQUEST ILLEGAL
= 3	EQUIPMENT NOT READY
= 4	IMMEDIATE COMPLETION
= 5	DCPC CHANNEL REQUIRED
= 6-99	PROGRAM MAKING I/O REQUEST IS ABORTED & MESSAGE PRINTED
6-59	HP DRIVERS
60-99	USER WRITTEN DRIVERS

ON RETURN FROM YOUR DRIVER, WHAT HAPPENS?



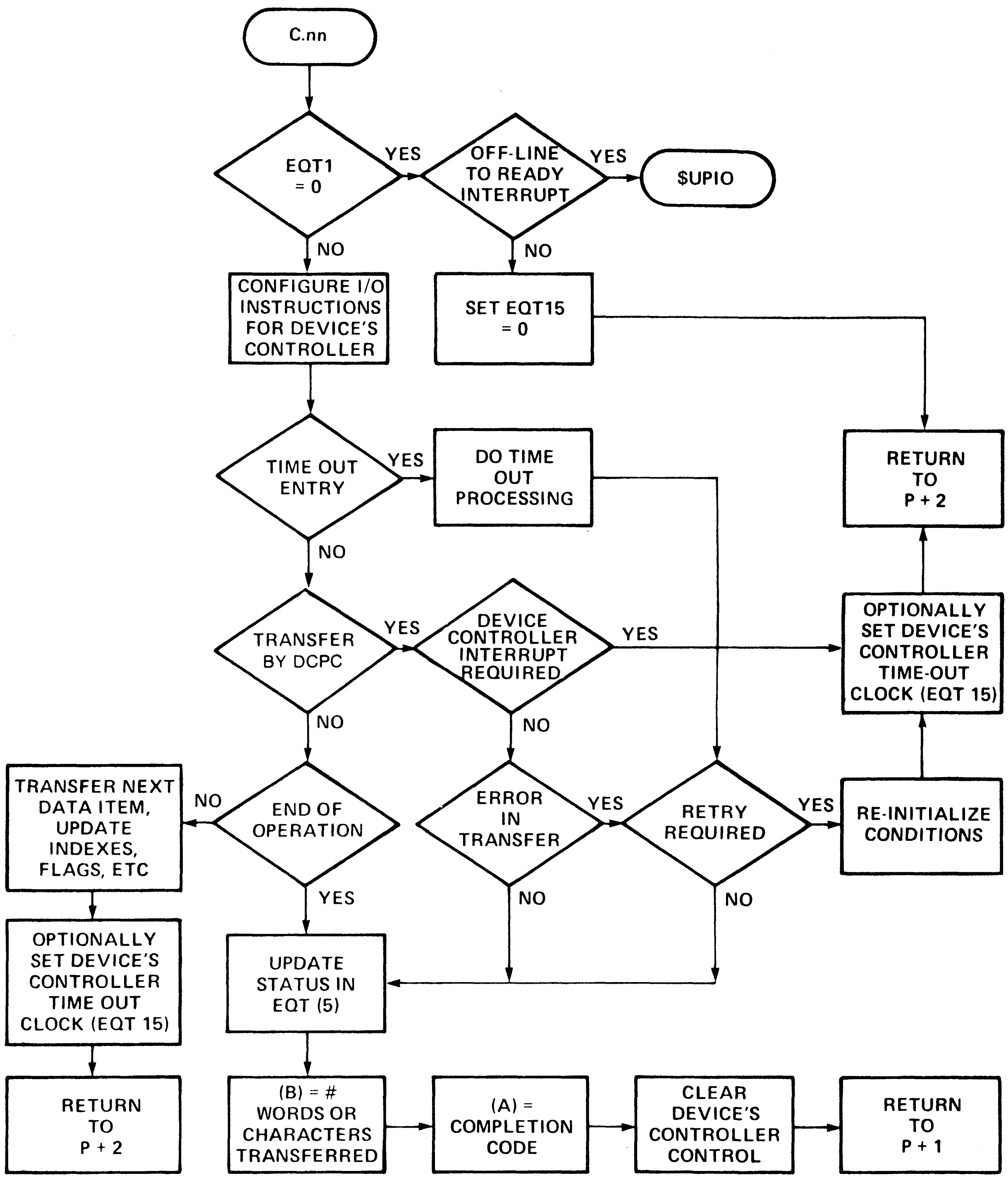
WHAT DOES $CIC DO FOR YOU?





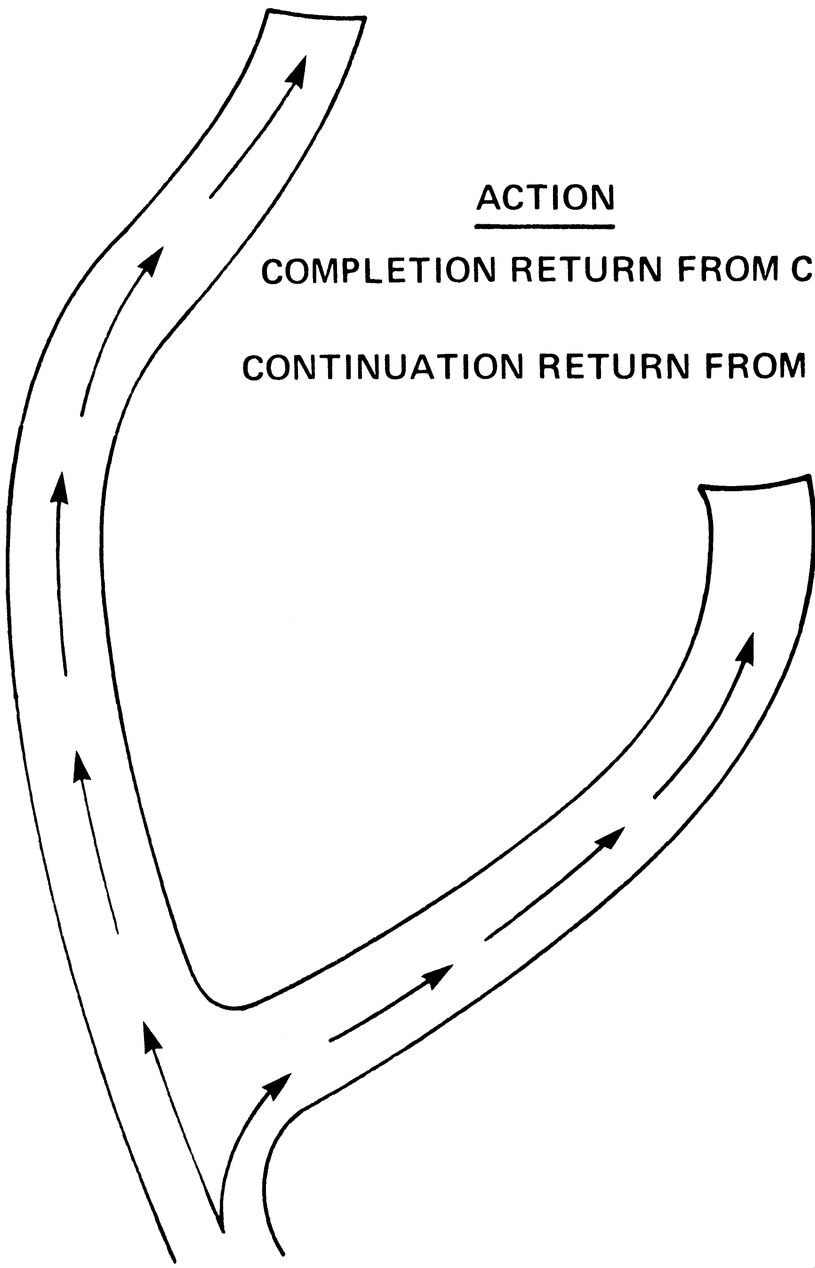
WHAT DOES $CIC DO FOR YOU (CONT'D)

RTE DRIVER COMPLETION SECTION



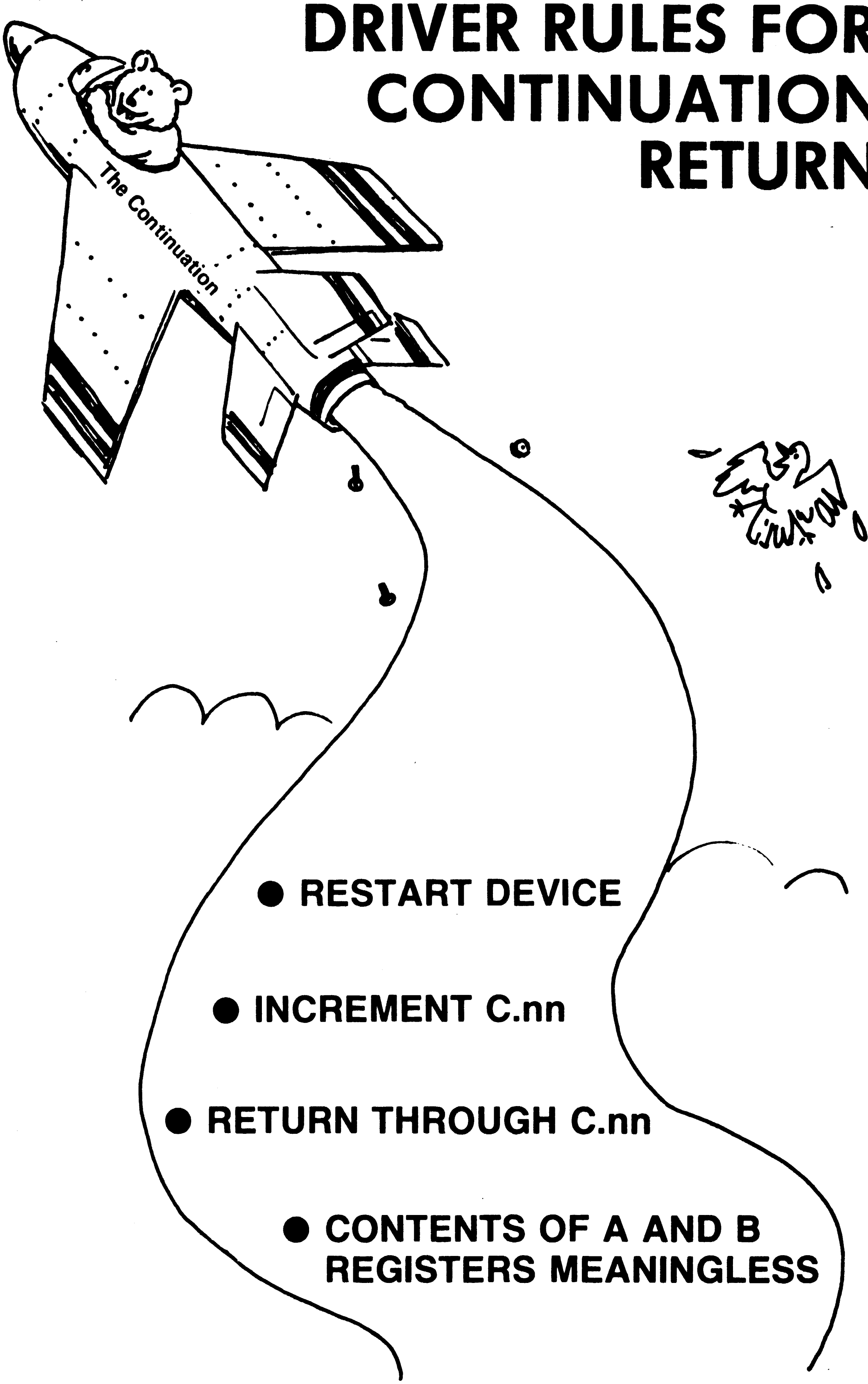
RTE DRIVERS MUST INDICATE TO THE EXECUTIVE WHEN TRANSFER IS COMPLETE. THIS IS ACCOMPLISHED BY EXITING THE DRIVER IN ONE OF TWO WAYS

DRIVER RETURN TO RTE	ACTION
I. P + 1	COMPLETION RETURN FROM C.nn
II. P + 2	CONTINUATION RETURN FROM C.nn



TWO WAYS TO RETURN FROM THE COMPLETION SECTION

DRIVER RULES FOR CONTINUATION RETURN



- **RESTART DEVICE**
- **INCREMENT C.nn**
- **RETURN THROUGH C.nn**
- **CONTENTS OF A AND B REGISTERS MEANINGLESS**

DRIVER RULES FOR COMPLETION RETURN

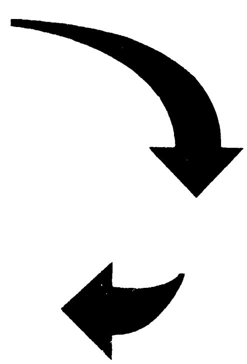
- CLC ON DEVICE
- PUT COMPLETION STATUS IN A-REGISTER

  (A) = 0 FOR SUCCESSFUL COMPLETION
  = 1 DEVICE NOT READY
  = 2 END OF TRANSMISSION (UNEXPECTED)
  = 3 TRANSMISSION PARITY ERROR
  = 4 DEVICE TIME-OUT

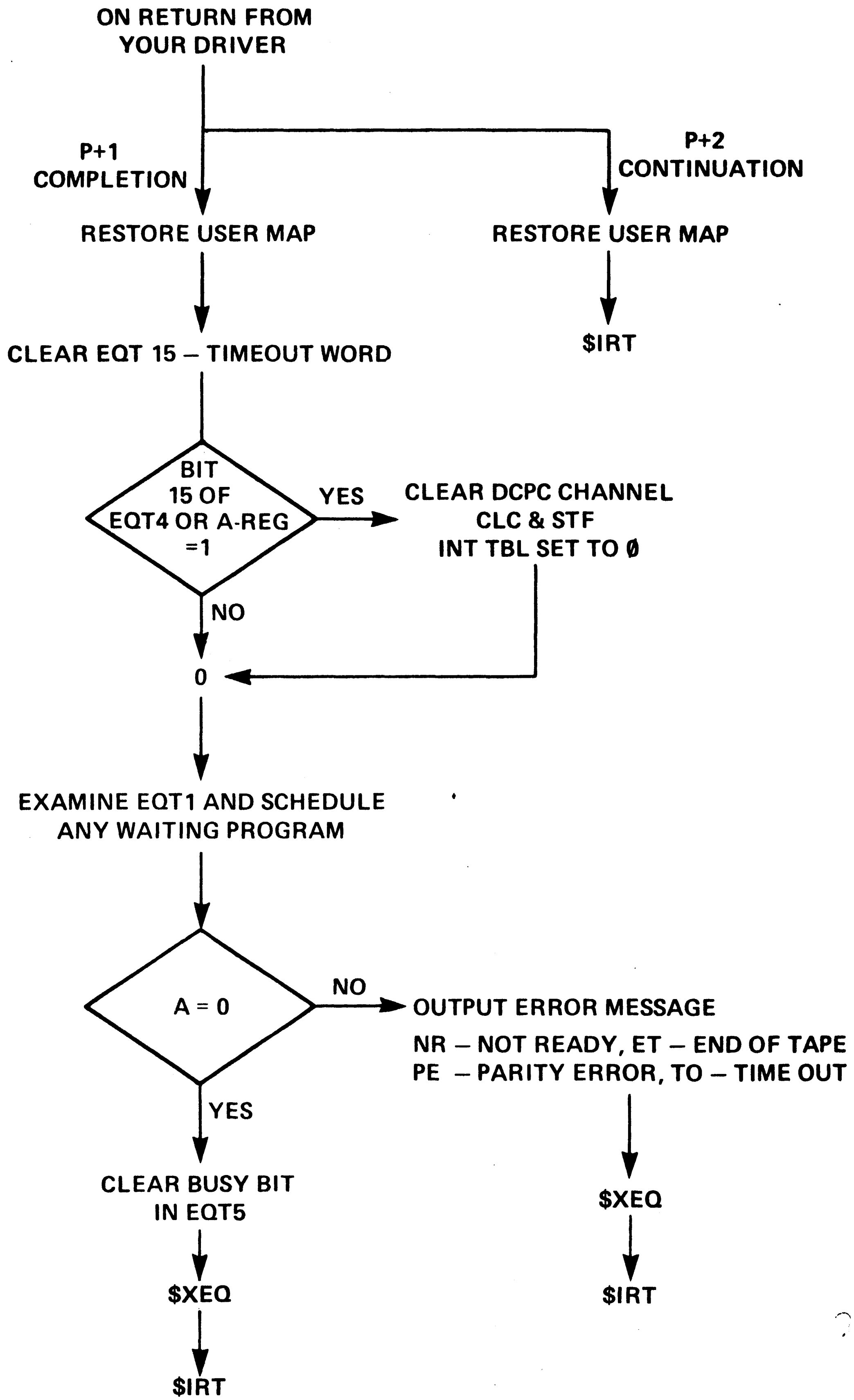
- NOTE: (A) #0 PRODUCES AN ERROR MESSAGE ON THE SYSTEM CONSOLE.

STATUS IN A	ERROR MESSAGE
1	I/O NR En Ln Sn
2	I/O ET En Ln Sn
3	I/O PE En Ln Sn
4	I/O TO En Ln Sn

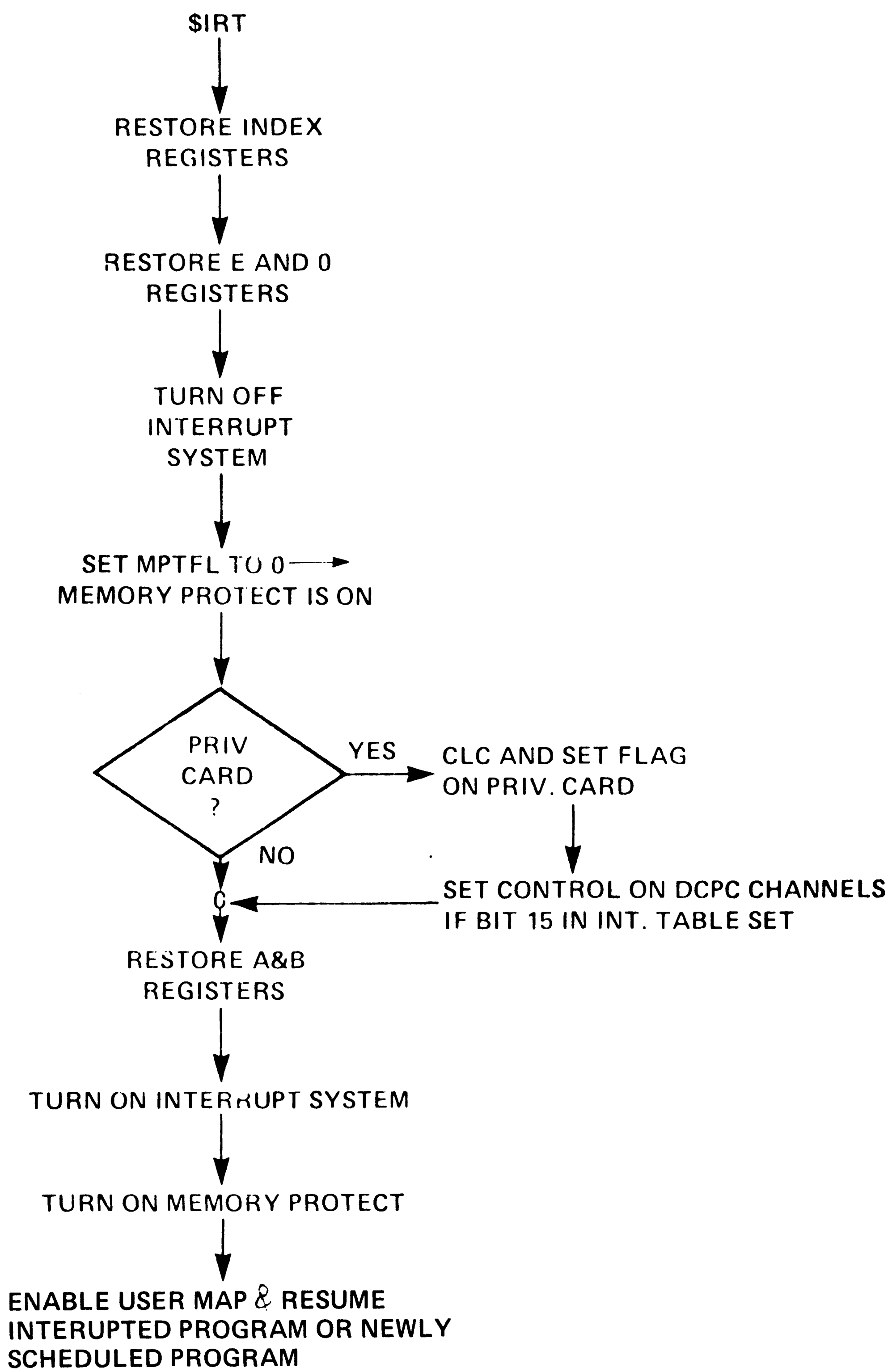
- PUT THE TRANSMISSION LOG IN THE B-REGISTER
- MODIFY STATUS BITS 0 THROUGH 7 OF EQT WORD 5.



WHAT DOES $CIC DO AFTER YOUR DRIVER FINISHES?



$CIC RETURN SEQUENCE



```
                   ASMB,L
     00000              NAM DVR70   ** STANDARD RTE DRIVER EXAMPLE **
*
*
                        ENT I.70,C.70
*
*
* DRIVER 70 OPERATES UNDER THE CONTROL OF THE I/O CONTROL (IOC)
* AND THE CENTRAL INTERRUPT CONTROL (CIC) MODULES OF RTE.
* THIS DRIVER IS RESPONSIBLE FOR CONTROLLING OUTPUT
* TRANSMISSION TO A 16 BIT EXTERNAL DEVICE.
* I.70 IS THE ENTRY POINT FOR THE *INITIATION* SECTION
* AND C.70 IS THE ENTRY POINT FOR THE *CONTINUATION/COMPLETION*
* SECTION.
*
* NOTE THAT THIS DRIVER DOES NOT PROCESS TIME-OUTS OR
* POWER FAIL.  THESE PROCEDURES ARE LEFT ENTIRELY UP TO
* THE SYSTEM.
*
* REMEMBER THAT RTE SETS THE ADDRESSES OF EACH WORD OF
* THE 15 WORD EQT ENTRY FOR THE DEVICE BEING SERVICED INTO
* THE BASE PAGE COMMUNICATIONS AREA ON EACH ENTRY TO THE
* DRIVER.
* THIS DRIVER REFERENCES THESE ADDRESSES THROUGH VARIABLES
* EQT1 THROUGH EQT15.
*
* **********************
* * INITIATION SECTION *
* **********************
*
* THE INITIATION SECTION IS CALLED FROM I/O CONTROL (IOC) TO
* INITIALIZE A DEVICE AND INITIATE AN OUTPUT OPERATION
*
* THE CALLING SEQUENCE FOR THE INITIATION SECTION IS:
*
*                  (SET A = SELECT CODE OF I/O DEVICE)
*          P       JSB I.70
*          P+1     (RETURN POINT)
*
*          ON RETURN, A REGISTER INDICATES STATUS, AS FOLLOWS:
*
*                  A = 0,   OPERATION SUCCESSFULLY INITIATED
*                  A NOT 0, OPERATION REJECTED FOR THE FOLLOWING
*                           REASON:
*
*                           A = 1 = ILLEGAL READ REQUEST
*                           A = 2 = ILLEGAL CONTROL REQUEST
*
* (NOTE, HOWEVER, THAT A "CLEAR" CONTROL REQUEST FROM THE
*  SYSTEM WILL BE PROCESSED BY THE DRIVER, AS REQUIRED.)
*
* ***********************************
* * CONTINUATION/COMPLETION SECTION *
* ***********************************
*
* THE CONTINUATION/COMPLETION SECTION IS CALLED BY CENTRAL
```

```
* INTERRUPT CONTROL (CIC) TO CONTINUE OR COMPLETE AN OPERATION WHEN
* AN INTERRUPT IS DETECTED ON THE DEVICE
*
* THE CALLING SEQUENCE FOR THE COMPLETION SECTION IS:
*
*                   (SET A = SELECT CODE OF I/O DEVICE)
*           P       JSB C.70
*           P+1     COMPLETION RETURN
*           P+2     CONTINUATION RETURN
*
*           ON RETURN, A & B REGISTERS INDICATE STATUS, AS FOLLOWS:
*
*                   ON A COMPLETION RETURN:
*
*                   A = 0, SUCCESSFUL COMPLETION, WITH
*                          B = NUMBER OF WORDS TRANSMITTED
*
*                   A = 2, TRANSMISSION ERROR DETECTED
*
*                   ON A CONTINUATION RETURN, THE REGISTERS ARE
*                   MEANINGLESS
*
* RECORD FORMAT:
*
*     THIS DRIVER PROVIDES A 16 BIT BINARY WORD
*     TRANSFER ONLY.
*
```

```
*
*                                 ***********************
*                                 * INITIATION SECTION *
*                                 ***********************
*
      00000 000000  I.70   NOP                ENTRY FROM IOC
*
      00001 016100R        JSB SETIO          CONFIGURE I/O INSTRUCTIONS FOR DEVICE
*
      00002 161665         LDA EQT6,I         GET CONTROL WORD OF REQUEST, AND
      00003 012115R        AND =B3            ISOLATE THE REQUEST TYPE
*
      00004 052116R        CPA =B1            IF REQUEST IS FOR INPUT
      00005 126000R        JMP I.70,I         THEN REJECT IT (A = 1 = ILLEGAL READ)
      00006 052117R        CPA =B2            IF REQUEST IS FOR OUTPUT
      00007 026017R        JMP D.X1           THEN GO PROCESS WRITE REQUEST
*
* CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
*
      00010 161665         LDA EQT6,I         ACCESS CONTROL WORD
      00011 012120R        AND =B3700         ISOLATE SUBFUNCTION
      00012 002002         SZA                "CLEAR" REQUEST?
      00013 026015R        JMP REJCT          NO, SO REJECT REQUEST AS ILLEGAL
*
      00014 106700  I.0    CLC SC             YES, CLEAR DEVICE AND RETURN
*
* REQUEST ERROR - CAUSE REJECT RETURN TO IOC
*
      00015 062117R REJCT  LDA =B2            SET A = 2 FOR ILLEGAL CONTROL REQUEST
      00016 126000R        JMP I.70,I         AND RETURN (A = 2 = ILLEGAL CONT. REQ.)
*
* WRITE REQUEST PROCESSING
*
      00017 161666  D.X1   LDA EQT7,I         GET REQUEST BUFFER ADDRESS
      00020 171670         STA EQT9,I         AND SET IT AS CURRENT ADDRESS
      00021 161667         LDA EQT8,I         GET REQUEST BUFFER LENGTH
      00022 003004         CMA,INA            MAKE NEGATIVE AND
      00023 171671         STA EQT10,I        AND SAVE AS REMAINING BUFFER LENGTH
      00024 002002         SZA                IS BUFFER LENGTH = 0?
      00025 026031R        JMP D.X3           NO, PROCESS AS USUAL
      00026 062121R        LDA =B4            YES, SO MAKE IMMEDIATE COMPLETION RETURN
      00027 006400         CLB                SET TRANSMISSION LOG = 0 INTO B
      00030 126000R        JMP I.70,I         AND RETURN (A = 4 = IMMED. COMPLETION)
*
* CALL THE CONTINUATION/COMPLETION SECTION TO WRITE FIRST WORD
*
      00031 062114R D.X3   LDA P2             ADJUST RETURN ADDRESS SO WILL
      00032 072036R        STA C.70           RETURN HERE (INITIATION SECTION)
      00033 026047R        JMP D.X2           GO TO COMPLETION SECTION
*
      00034 002400  IEXIT  CLA                NOW RETURN TO IOC WITH
      00035 126000R        JMP I.70,I         OPERATION INITIATED (A = 0 = OK)
*
```

```
*
*                                ***********************************
*                                * CONTINUATION/COMPLETION SECTION *
*                                ***********************************
*
      00036 000000   C.70  NOP             CONTINUATION/COMPLETION ENTRY POINT
*
      00037 016100R        JSB SETIO       CONFIGURE I/O INSTRUCTIONS
*
      00040 161660         LDA EQT1,I      CHECK FOR SPURIOUS INTERRUPT
      00041 012122R        AND =B77777     ISOLATE I/O REQUEST LIST PTR (15 BITS)
      00042 002002         SZA             IS A REQUEST IN PROGRESS?
      00043 026047R        JMP D.X2        YES, GO PROCESS REQUEST
*
      00044 171774         STA EQT15,I     NO, SPURIOUS INTERRUPT-ZERO TIME-OUT CLK
      00045 036036R        ISZ C.70        ADJUST RETURN TO P+2 (CONTINUATION)
      00046 126036R        JMP C.70,I      MAKE CONTINUATION RETURN TO CIC
*
      00047 002400   D.X2  CLA             IF CURRENT BUFFER LENGTH  = 0,
      00050 151671         CPA EQT10,I     THEN GO TO STATUS
      00051 026063R        JMP I.3         SECTION. (I.E., TRANSFER DONE NOW)
*
      00052 165670         LDB EQT9,I      GET CURRENT BUFFER ADDRESS
*
      00053 135670         ISZ EQT9,I      ADD 1 FOR NEXT WORD
      00054 160001         LDA B,I         GET WORD TO BE WRITTEN TO DEVICE
      00055 135671         ISZ EQT10,I     INCREMENT WORD COUNT ALSO
      00056 000000         NOP             IGNORE P+1 SKIP IF LAST WORD
*
      00057 102500   I.1   OTA SC          OUTPUT WORD TO INTERFACE
      00060 103700   I.2   STC SC,C        TURN DEVICE ON
*
      00061 036036R        ISZ C.70        ADJUST RETURN TO P+2 (CONTINUATION)
      00062 126036R        JMP C.70,I      MAKE CONTINUATION RETURN
*
* STATUS AND COMPLETION SECTION
*
      00063 102500   I.3   LIA SC          GET STATUS WORD FROM DEVICE
      00064 012123R        AND =B77        STRIP OFF UNUSED BITS
      00065 070001         STA B           SAVE IN B TEMPORARILY
      00066 161664         LDA EQT5,I      REMOVE PREVIOUS STATUS
      00067 012124R        AND =B177400    BITS IN EQT WORD 5
      00070 030001         IOR B           OR IN NEW BITS
      00071 171664         STA EQT5,I      AND RESET INTO EQT WORD 5
*
      00072 002400         CLA             SET A = 0 = OK RETURN CODE
      00073 056121R        CPB =B4         ERROR STATUS BIT ON?
      00074 062117R        LDA =B2         YES, SET A = 2 = ERROR RETURN
*
      00075 165667         LDB EQT8,I      SET B = TRANSMISSION LOG
*
      00076 106700   I.4   CLC SC          CLEAR DEVICE CONTROLLER
*
      00077 126036R        JMP C.70,I      MAKE COMPLETION RETURN TO CIC
*
```

```
*
*                                 *********************
*                                 * SUBROUTINE SETIO *
*                                 *********************
*
* SUBROUTINE <SETIO> CONFIGURES ALL I/O INSTRUCTIONS IN DRIVER
*
  00100 000000  SETIO NOP                ENTRY POINT
*
  00101 032113R       IOR LIA            COMBINE LIA WITH I/O
  00102 072063R       STA I.3            SELECT CODE AND SET IN CODE
*
  00103 042125R       ADA =B100          CONSTRUCT OTA INSTRUCTION
  00104 072057R       STA I.1
*
  00105 042126R       ADA =B1100         CONSTRUCT STC,C INSTRUCTION
  00106 072060R       STA I.2
*
  00107 032127R       IOR =B4000         CONSTRUCT CLC INSTRUCTION
  00110 072014R       STA I.0
  00111 072076R       STA I.4
*
  00112 126100R       JMP SETIO,I        RETURN
*
```

```
0222*
0223*                                 **************
0224*                                 * DATA AREA *
0225*                                 **************
0226*
0227* CONSTANT AND STORAGE AREA
0228*
0229  00000              A       EQU 0              A-REGISTER
0230  00001              B       EQU 1              B-REGISTER
0231*
0232  00000              SC      EQU 0              DUMMY I/O SELECT CODE NUMBER
0233  00113 102500       LIA     LIA 0              CODE FOR LIA INSTRUCTION
0234  00114 000033R      P2      DEF IEXIT-1        RETURN POINT IN INITIATION SECTION
0235*
0236* ** BASE PAGE COMMUNICATIONS AREA DEFINITIONS **
0237*
0238  01650              .       EQU 1650B
0239*
0240  01660              EQT1    EQU .+8
0241  01661              EQT2    EQU .+9
0242  01662              EQT3    EQU .+10
0243  01663              EQT4    EQU .+11
0244  01664              EQT5    EQU .+12
0245  01665              EQT6    EQU .+13
0246  01666              EQT7    EQU .+14
0247  01667              EQT8    EQU .+15
0248  01670              EQT9    EQU .+16
0249  01671              EQT10   EQU .+17
0250  01672              EQT11   EQU .+18
0251  01771              EQT12   EQU .+81
0252  01772              EQT13   EQU .+82
0253  01773              EQT14   EQU .+83
0254  01774              EQT15   EQU .+84
0255*
0256*
      00115 000003
      00116 000001
      00117 000002
      00120 003700
      00121 000004
      00122 077777
      00123 000077
      00124 177400
      00125 000100
      00126 001100
      00127 004000
0257                                 END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

PURPOSE — DCPC TRANSFERS DATA DIRECTLY BETWEEN MEMORY AND HIGH SPEED AND/OR SYNCHRONOUS DEVICES.

DCPC
DUAL CHANNEL PORT CONTROLLER

THE TRANSFER IS BEGUN BY THE INITIATOR PORTION OF DRIVER. OPERATION IS CONTROLLED BY COMPUTER HARDWARE.

FUNCTION OF THE DCPC INITIATOR

* SETS UP DCPC HARDWARE

* SPECIFIES DIRECTION OF TRANSFER
  (TO OR FROM MEMORY)

* SPECIFIES WHERE IN MEMORY TO READ OR STORE DATA

* DECIDES WHICH DEVICE SELECT CODE IS TO BE USED

* CONTROLS HOW MUCH DATA IS TO BE TRANSFERRED

DCPC CONTROL WORDS

CONTROL WORD 1 (DEVICE CONTROL)

STC* IF SET		CLC* IF SET	(NOT USED)	DEVICE SELECT CODE (BITS 5 - Ø)
BIT 15		BIT 13		

CONTROL WORD 2 (MEMORY CONTROL)

INPUT IF SET	MEMORY ADDRESS
BIT 15	

CONTROL WORD 3 (BLOCK LENGTH CONTROL)

WORD COUNT (2's COMPLEMENT)

*STC
CLC } IF SET IN CONTROL WORD 1, DCPC WILL ISSUE STC FOR EACH WORD TRANSFER AND/OR CLC UPON COMPLETION.

DCPC USAGE IN RTE DRIVERS

A DRIVER OBTAINS A DCPC CHANNEL FOR USE BY:

- DCPC BIT SET IN EQT AT SYSTEM GENERATION TIME

OR

- THE DRIVER CAN DYNAMICALLY REQUEST A DCPC CHANNEL IN THE INITIATOR.

  A REG = 5 ON RETURN

DCPC ASSIGNMENT BY RTE

Before calling the driver initiation section:

- CHAN is setukp on the base page
- Words 1 and 2 of the interrupt table are setup where,

```
+-------------------------------------+
|   DCPC Channel 1 Assignment Word    |     Interrupt Table Word 1
+-------------------------------------+      (I/O SELECT CODE 6)
|   DCPC Channel 2 Assignment Word    |     Interrupt Table Word 2
+-------------------------------------+      (I/O SELECT CODE 7)
```

Where each DCPC Channel Assignment Word has the format:

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+-----------------------------------------------+
| F!            Address                         |
+-----------------------------------------------+
```

Where:

F = 1, if the driver assigned to the channel needs the DCPC completion interrupt (set only in systems with a privileged interrupt card).

= 0, otherwise

Address = the address of the EQT entry of the driver to which the DCPC channel is assigned.

= 0, if the DCPC channel is currently not assigned.

DYNAMIC DCPC REQUEST

- Once the driver determines that it needs a DCPC channel for a request, it requests DCPC by:

```
CHDCP EQU  *          Executes this code if DCPC required
      DLD  INTBA,I    Access DCPC Channel Assignment Words
      CPA  EQT1       Is DCPC channel 1 assigned to this driver?
      JMP  CH1        Yes, configure and initiate transfer on channel 1
      CPB  EQT1       Is DCPC channelo 2 assigned to this driver?
      JMP  CH2        Yes, configure and initiate transfer on channel 2
      LDA  =B5        No.  A DCPC channel is not assigned.  Set
      JMP  Ixnn,I     A = 5 to request one from IOC, and return.
```

- When the request is completed, the DCPC channel is returned by:

```
      LDA  COMCD      Set A = completion code determined earlier
      IOR =B100000    Set sign bit to indicate dynamic DCPC assignment
      JMP  Cxnn,I     Return to CIC
```

DCPC COMPLETION INTERRUPT

- BOTH THE DCPC AND DEVICE CAN GENERATE INTERRUPTS ON COMPLETION

- IF YOUR DRIVER NEEDS ONLY A DEVICE INTERRUPT, CLEAR CONTROL ON THE DCPC CHANNEL AFTER INITIALIZATION

  NO FURTHER PROCESSING IS REQUIRED

- IF A DCPC COMPLETION INTERRUPT IS REQUIRED, THEN SPECIAL PROCESSING IS REQUIRED IN YOUR DRIVER

SPECIAL PROCESSING IF DCPC COMPLETION INTERRUPT IS REQUIRED

```
     CLF 0                  Disable the interrupt system

     STC DCPC,C             Initiate transfer on DCPC channel

     CLA                    Bypass section below if
     CPA DUMMY              DUMMY = 0(non-privileged system)
     JMP X                  and special processing not needed.

     CLC DCPC               Clear DCPC control to inhibit DCPC
     LDB INTBA              interrupt.  Set B = address of the appropriate
     LDA CHAN               DCPC Channel Assignment word in the
     CPA = D7               Interrupt Table
     INB

     LDA B,I                Set bit 15 of DCPC channel assignment entry
     IOR = B1000000         equal to 1 as flag to system to turn DCPC
     STA B,I                interrupts back on later.  Reenable the
     STF 0                  interrupt system.

X    EQT *                  Continue processing.
```

PRIVILEGED DRIVERS

20-39

WHAT IS A PRIVILEGED INTERRUPT?

* RTE NORMALLY HAS THE INTERRUPT SYSTEM OFF WHILE A DRIVER SERVICES AN I/O REQUEST.

* SOME DEVICES CANNOT BE DELAYED AND SHOULD HAVE THE "PRIVILEGE" OF GENERATING AN INTERRUPT AT ANY TIME.

* THIS REQUIRES A I/O CARD KNOWN AS A PRIVILEGED FENCE.

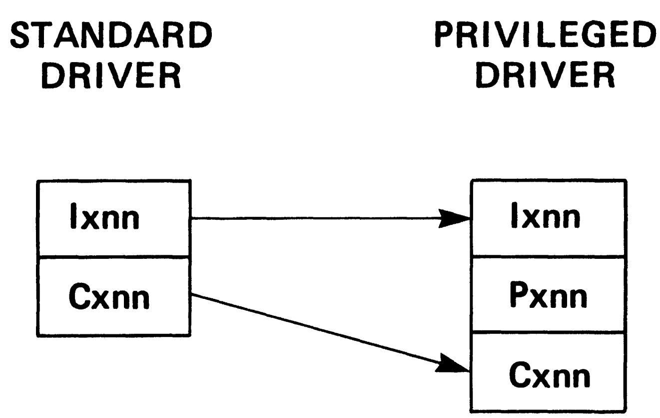
* THIS FENCE PHYSICALLY SEPARATES THE PRIVILEGED DEVICE INTERRUPTS FROM REGULAR DEVICE INTERRUPTS.

* THE SELECT CODE OF FENCE USED IS STORED IN BASE PAGE LOCATION 1737_8 LABELED "DUMMY".

* RTE OPERATES WITH AN INTERRUPT SYSTEM ON FOR DRIVER SERVICING BUT INTERRUPTS ARE HELD OFF FOR THOSE DEVICES AFTER THE FENCE.

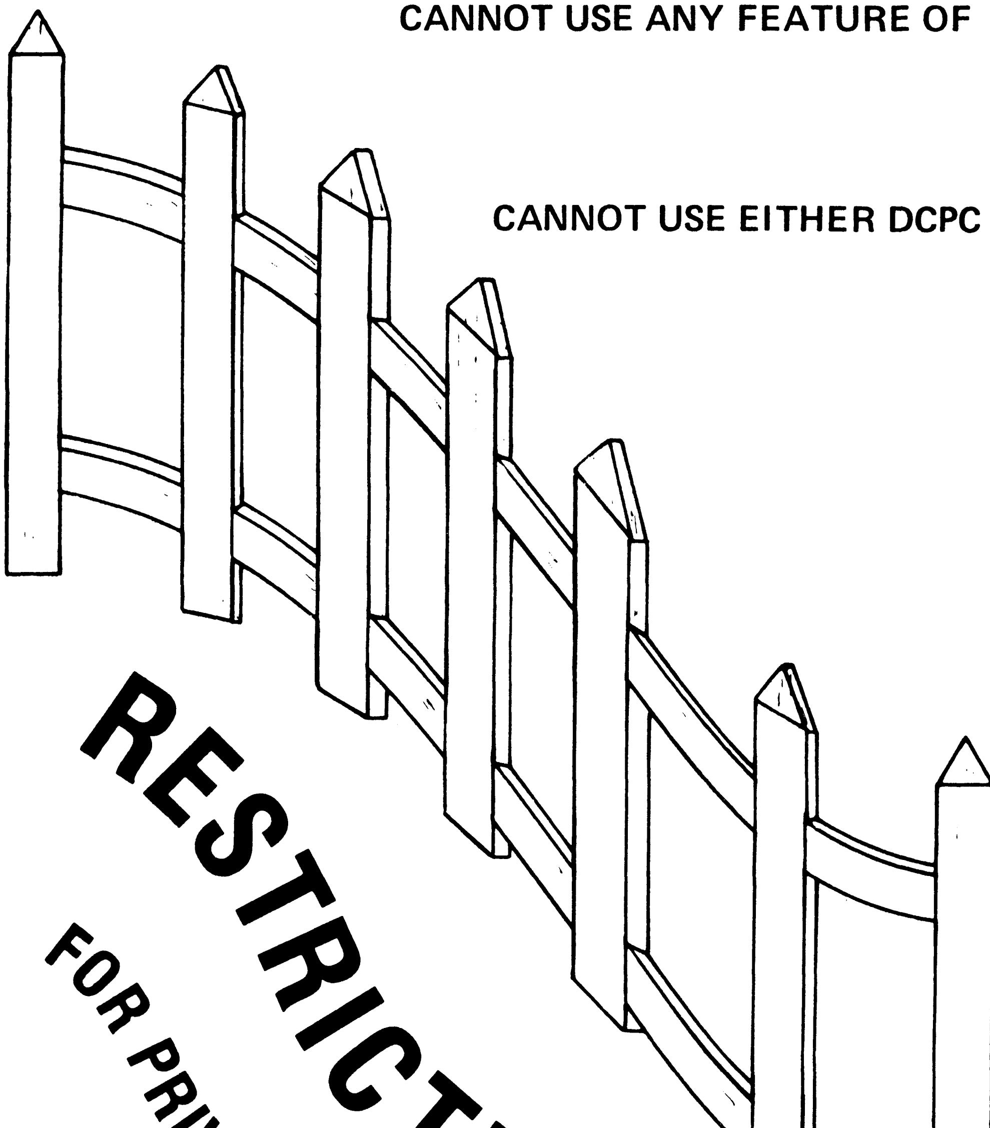
HOW ARE PRIVILEGED INTERRUPTS PROCESSED?

- PRIVILEGED DRIVER
  - USER PROGRAM CALLS ARE THE SAME AS ANY I/O CALL
  - THE DRIVER, IN GENERAL, HAS THE SAME STRUCTURE AS A REGULAR DRIVER PLUS A PRIVILEGED PORTION.
- PRIVILEGED ROUTINE
  - TRAP CELL SET TO JSB XXX,I DURING SYSGEN.



CANNOT USE ANY FEATURE OF **EXEC**

CANNOT USE EITHER DCPC CHANNEL.



RESTRICTIONS

FOR PRIVILEGED ROUTINES

DATA TRANSFER

- USE SYSTEM COMMON

- CLASS I/O –
  SYSTEM AVAILABLE MEMORY

- REIO – SYSTEM AVAILABLE MEMORY

- **IF THESE ARE USED, THEN YOUR DRIVER NEVER NEEDS TO USE MAP SWITCHING IN RTE-IV**

PRIVILEGED DRIVER CONCEPTS

- CALLED BY EXEC OR REIO I/O CALL
- CALLING PROGRAM PLACED INTO I/O SUSPENSION
- DEVICE TRAP CELL CHANGED FROM

  $\boxed{\text{JSB \$CIC,I}}$ TO $\boxed{\text{JSB P.XX,I}}$

  NOW RTE BYPASSED ON INTERRUPTS.

- SYSTEM IS NOTIFIED OF COMPLETION BY:

  PRIVILEGED PORTION OF DRIVER SETS
  TIMEOUT IN EQT AND EXITS.
  ON TIMEOUT, RTE ENTERS CONTINUATOR

- $\boxed{\text{C.XX}}$ RETURNS TRANSMISSION LOG AND STATUS
- SUSPENDED PROGRAM RESUMES

- CHECKS FOR VALID REQUEST CODE
- SINCE DRIVER CONTROLS ONLY <u>ONE</u> DEVICE, CONFIGURE ONCE AND SET A SWITCH TO PREVENT RE-EXECUTING.
- TRAP CELL MODIFIED <u>ONCE</u>
- COUNT AND BUFFER ADDRESS SAVED WITHIN THE DRIVER
- START DEVICE
- RETURN

INITIATOR SECTION

PRIVILEGED SECTION

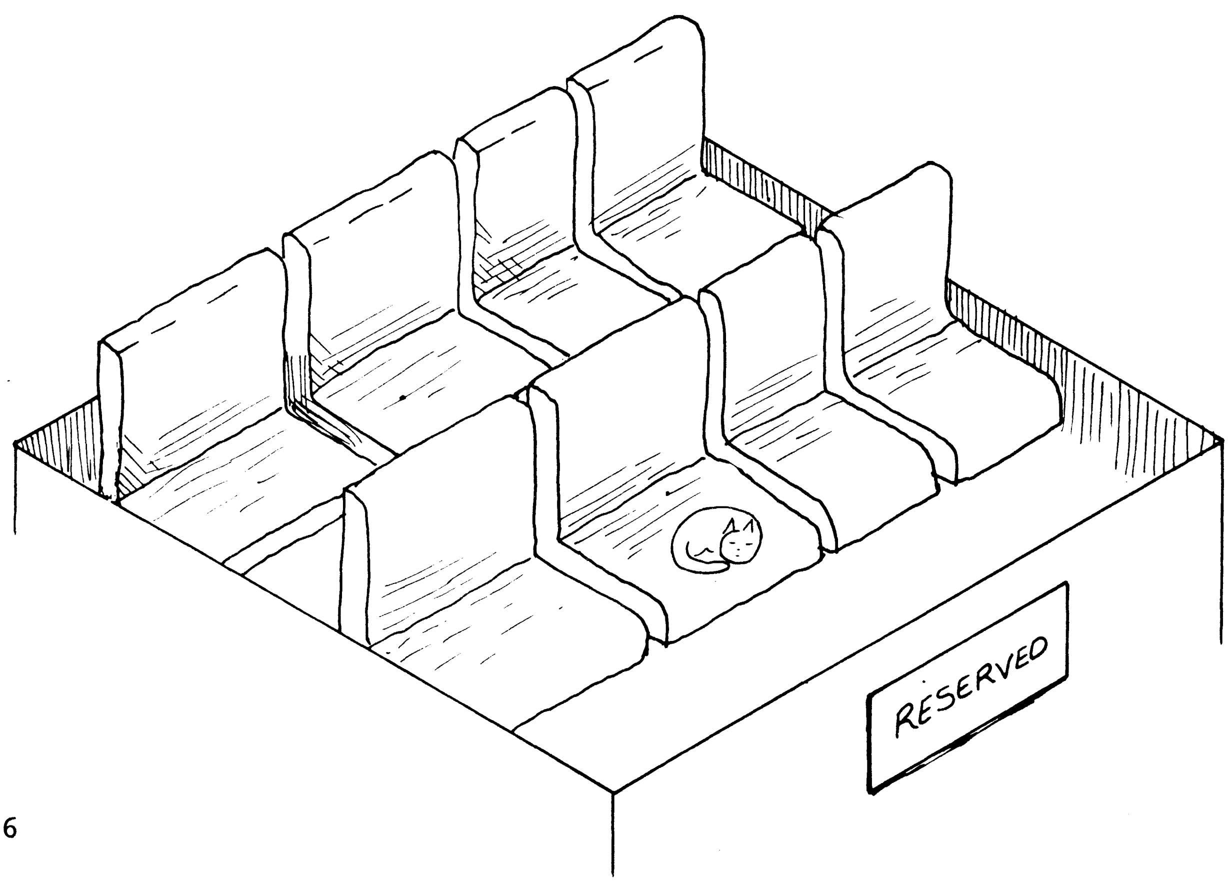
- ENTRY IS AUTOMATIC, BYPASSING RTE

TURN OFF INTERRUPT SYSTEM

- SAVE ALL REGISTERS TO BE USED
- DISABLE DCPC INTERRUPTS
- SAVE MEMORY PROTECT STATUS
- SAVE DMS STATUS

ENABLE INTERRUPT SYSTEM

- TRANSFER DATA



MORE DATA TO BE PROCESSED

- DISABLE INTERRUPT SYSTEM
- START DEVICE
- REENABLE DCPC COMPLETION INTERRUPT IF:

  MP WAS ON AND A STANDARD DRIVER REQUIRES A DCPC COMPLETION INTERRUPT.

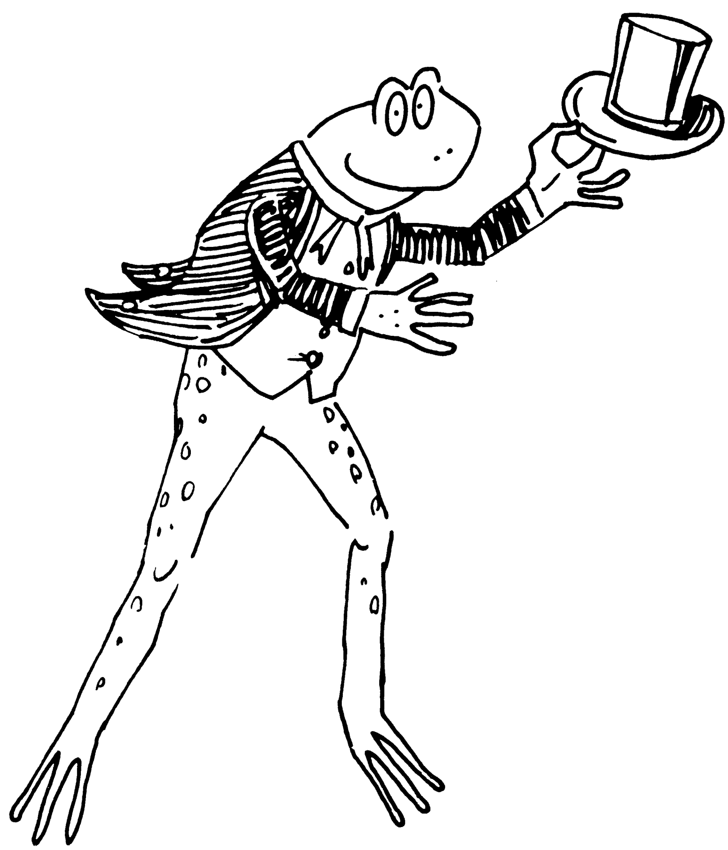
- RESTORE MEMORY PROTECT AND ITS FLAG (MPTFL) TO ITS ORIGINAL STATE
- RESTORE REGISTERS
- ENABLE INTERRUPT SYSTEM
- RESTORE DMS
- RETURN

LAST DATA HAS BEEN PROCESSED

- DISABLE INTERRUPT SYSTEM
- CLC ON DEVICE
- SET UP 10 MSEC TIMEOUT (STORE −1 IN EQT 15)
- REENABLE DCPC COMPLETION INTERRUPT
- RESTORE MEMORY PROTECT
- RESTORE REGISTERS
- ENABLE INTERRUPT SYSTEM
- RESTORE DMS STATUS
- RETURN

COMPLETION SECTION

- ENTERED ONLY AFTER TIMEOUT
- SET RETURN PARAMETERS IN A & B REGISTERS
- MODIFY STATUS BITS 0-7 IN EQTS
- RETURN



```
                ASMB,L,C
*
      00000            NAM DVYNN   ** RTE DMS PRIVILEGED DRIVER EXAMPLE **
                       SUP
*
                       ENT IXNN,CXNN
*
*************************************************************
* SAMPLE RTE PRIVILEGED DRIVER DVYNN - FOR DMS SYSTEMS *
*************************************************************
*
* HANDLES USER PROGRAM REQUESTS TO READ FROM A PRIVILEGED
* CONTROLLER
*
* USER PROGRAM CALLING SEQUENCE:
*
*   JSB EXEC        CALL EXEC
*   DEF *+5         RETURN POINT
*   DEF RCODE       REQUEST CODE (MUST BE READ REQUEST)
*   DEF CONWD       CONTROL WORD
*   DEF BUFFR       ADDRESS OF BUFFER (MUST BE IN SYSTEM COMMON)
*   DEF LENTH       LENGTH OF BUFFER
*
* CAUTION:
*
* THIS DRIVER WILL NOT WORK WITH MORE THAN ONE PRIVILEGED
* CONTROLLER.  IF MORE THAN ONE PRIVILEGED CONTROLLER
* EXISTS IN A SYSTEM, DVYNN MUST BE
* RE-ASSEMBLED WITH ALL NAMES CONTAINING "NN" CHANGED SO
* THAT EACH COPY OF THE DRIVER HAS UNIQUE ENTRY POINTS.
* THEN ONE DRIVER PER CONTROLLER MUST BE PUT
* INTO THE SYSTEM AT GENERATION TIME.
*
* NOTE:
*
* 1.) THE DESIGN OF THIS DRIVER ASSUMES THAT THE I/O
*     BUFFER BEING PROCESSED IS LOCATED IN SYSTEM COMMON.
*     THIS CAUSES THE DRIVER TO BE ENTERED WITH THE
*     SYSTEM MAP ENABLED.  THIS IS NECESSARY FOR THE
*     CORRECT OPERATION OF THE TRAP CELL MODIFICATION
*     TECHNIQUE ILLUSTRATED BELOW.  IN ADDITION, THE
*     BUFFER IN SYSTEM COMMON ALLOWS THE DRIVER TO PUT THE
*     DATA VALUES DIRECTLY INTO THE BUFFER, WITHOUT
*     THE NEED FOR MAP SWITCHING
*
* 2.) THIS DRIVER DOES NOT PROCESS POWER FAIL INTERRUPTS.
*
* 3.) THIS DRIVER DOES NOT PROCESS ANY TIME-OUTS EXCEPT
*     FOR THE TIME-OUT THAT IT CREATES AS A MEANS TO
*     COMPLETE THE I/O REQUEST AND RETURN TO IOC
*
```

```
*
*                        ***********************
*                        * INITIATION SECTION *
*                        ***********************
*
      00000 000000  IXNN  NOP              INITIATION SECTION ENTRY POINT
      00001 072200R       STA SCODE        SAVE SELECT CODE OF CONTROLLER
*
      00002 066203R       LDB FIRST        ACCESS FIRST TIME THROUGH FLAG
      00003 006002        SZB              IS THIS THE FIRST TIME THRU?
      00004 026020R       JMP INIT         NO, SO SKIP CONFIGURATION CODE
*
* CONFIGURE I/O INSTRUCTIONS
*
      00005 032217R       IOR LIA          CREATE LIA INSTRUCTION
*                              .                  .
*                              .                  .
*                              .                  .
*
* MODIFY TRAP CELL
*
      00006 060000B       LDA $JSB         SET TRAP CELL TO
      00007 172200R       STA SCODE,I      JSB $JPNN,I ($JPNN = ADDR OF PXNN)
*
* SAVE EQT ADDRESSES
*
      00010 061774        LDA EQT15        SAVE EQT15
      00011 072215R       STA EQ15
      00012 061663        LDA EQT4         EQT 4
      00013 072214R       STA EQ4
      00014 061660        LDA EQT1         AND EQT1
      00015 072213R       STA EQ1          ADDRESSES
*
      00016 002404        CLA,INA          SET FLAG TO PREVENT CONFIGURING ON
      00017 072203R       STA FIRST        SUBSEQUENT INITIATIONS
*
* CLEAR THE "DRIVER PROCESSES TIME-OUT" BIT TO ALLOW
* NORMAL TIME-OUT OPERATION
*
      00020 161663  INIT  LDA EQT4,I       ACCESS EQT WORD 4
      00021 012221R       AND =B167777     CLEAR BIT 12
      00022 171663        STA EQT4,I       AND RESET EQT WORD 4
*
* CHECK THE REQUEST CODE
*
      00023 161665        LDA EQT6,I       ACCESS REQUEST CODE
      00024 012222R       AND =B3          ISOLATE REQUEST TYPE
      00025 052223R       CPA =B1          READ REQUEST?
      00026 026041R       JMP PROC         YES, GO PROCESS READ REQUEST
*
      00027 052222R       CPA =B3          CONTROL REQUEST?
      00030 026033R       JMP CNTRL        YES, GO PROCESS CONTROL REQUEST
*
      00031 002404        CLA,INA          NO, SO REJECT AS ILLEGAL WRITE REQUEST
      00032 126000R       JMP IXNN,I
*
```

```
* CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
*
  00033 161665  CNTRL LDA EQT6,I     ACCESS CONTROL WORD
  00034 012224R       AND =B3700     ISOLATE SUBFUNCTION
  00035 002002        SZA            "CLEAR" REQUEST?
  00036 026037R       JMP REJCT      NO, SO REJECT AS ILLEGAL CONTROL REQUEST
*
*                      .
*                      .             EXECUTE CODE TO CLEAR CONTROLLER
*                      .
*
  00037 062225R REJCT LDA =B2        REJECT AS ILLEGAL CONTROL REQUEST
  00040 126000R       JMP IXNN,I
*
* SET UP FOR THE DATA TRANSFER
*
  00041 161667  PROC  LDA EQT8,I     ACCESS # OF CONVERSIONS REQUIRED
  00042 003004        CMA,INA        NEGATE FOR CONVERSION COUNTER
  00043 072201R       STA CVCTR      AND SAVE
  00044 002021        SSA,RSS        REJECT IF
  00045 026037R       JMP REJCT      NUMBER <0
  00046 161666        LDA EQT7,I     SAVE DATA BUFFER ADDRESS
  00047 072202R       STA DAPTR      FOR PXNN
*
* INITIATE A READ AND RETURN
*
  00050 016053R       JSB READ       START A READ
  00051 103700  I.1   STC SC,C       ENCODE DEVICE
  00052 126000R       JMP IXNN,I     RETURN TO IOC
*
* SUBROUTINE TO INITIATE A READ
*
  00053 000000  READ  NOP            ROUTINE CONTAINING
*                      .             CONFIGURED I/O
*                      .             INSTRUCTIONS TO
*                      .             SET UP THE DEVICE
*                      .             TO INITIATE ONE READING
  00054 126053R       JMP READ,I
```

```
*
*                        ***********************
*                        * PRIVILEGED SECTION *
*                        ***********************
*
* SAVE STATE OF COMPUTER AT INTERRUPT
*
 00055 000000  PXNN  NOP              PRIVILEGED SECTION ENTRY POINT
*
 00056 103100        CLF 0            TURN OFF INTERRUPT SYSTEM
*
 00057 106706        CLC 6            TURN OFF DCPC COMPLETION INTERRUPTS
 00060 106707        CLC 7
*
 00061 072204R       STA ASV          SAVE REGISTERS
 00062 076205R       STB BSV
 00063 001520        ERA,ALS
 00064 102201        SOC
 00065 002004        INA
 00066 072206R       STA EOSV
 00067 105743        STX XSV          SAVE X REGISTER
 00071 105753        STY YSV          SAVY Y REGISTER
 00073 105714        SSM DMSTS        SAVE DYNAMIC MAPPING SYSTEM STATUS
*
 00075 061770        LDA MPTFL        SAVE OLD MEMORY PROTECT FLAG
 00076 072212R       STA MPFSV
 00077 002404        CLA,INA          SET MEMORY PROTECT FLAG TO OFF
 00100 071770        STA MPTFL        SINCE MEMORY PROTECT IS NOW OFF
*
 00101 102100        STF 0            TURN INTERRUPT SYSTEM BACK ON
*
* CHECK FOR SPURIOUS INTERRUPT
*
 00102 162213R       LDA EQ1,I        ACCESS REQUEST LIST POINTER WORD
 00103 012226R       AND =B77777      ISOLATE REQUEST LIST POINTER
 00104 002002        SZA              IS A REQUEST IN PROGRESS?
 00105 026111R       JMP PREAD        YES, GO PROCESS INTERRUPT
*
 00106 103100        CLF 0            NO, TURN OFF INTERRUPT SYSTEM
 00107 107700  I.2   CLC SC,C         RESET CONTROLLER, AND
 00110 026121R       JMP EXIT         IGNORE SPURIOUS INTERRUPT BY RETURNING
*
* PROCESS READ REQUEST
*
 00111         PREAD EQU *
*                          .          LOAD IN DATA FROM DEVICE
*                          .          VIA CONFIGURED I/O INSTRUCTIONS
*                          .
*
 00111 172202R       STA DAPTR,I      STORE WORD IN DATA BUFFER
 00112 036201R       ISZ CVCTR        IS THIS THE LAST CONVERSION?
 00113 002001        RSS              NO
 00114 026164R       JMP DONE         YES, GO SET UP TO TERMINATE CALL
*
 00115 036202R       ISZ DAPTR        NO, SET UP FOR NEXT CONVERSION
 00116 016053R       JSB READ         INITIATE IT
```

```
*
* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
*
      00117 103100          CLF 0          TURN OFF INTERRUPT SYSTEM TEMPORARILY
*
      00120 103700   I.3    STC SC,C       ENCODE DEVICE
*
      00121 062212R EXIT    LDA MPFSV      ACCESS PREVIOUS STATE OF MEMORY PROTECT
      00122 002002          SZA            WAS MEMORY PROTECT ON?
      00123 026134R         JMP EXIT1      NO, SO DO NOT TURN ON DCPC INTERRUPTS
*
      00124 065654          LDB INTBA      YES, TURN DCPC COMPLETION INTERRUPTS
      00125 160001          LDA B,I        BACK ON IF THEY WERE ON INITIALLY.
      00126 002020          SSA            ON/OFF STATUS IS INDICATED BY BIT 15
      00127 102706          STC 6          OF EACH DCPC ASSIGNMENT WORD IN THE
      00130 006004          INB            INTERRUPT TABLE
      00131 160001          LDA B,I
      00132 002020          SSA
      00133 102707          STC 7
*
      00134 062206R EXIT1   LDA EOSV       RESTORE E AND O REGISTERS
      00135 103101          CLO
      00136 000036          SLA,ELA
      00137 102101          STF 1
      00140 066205R         LDB BSV        RESTORE B-REGISTER
      00141 105745          LDX XSV        RESTORE X REGISTER
      00143 105755          LDY YSV        RESTORE Y REGISTER
*
      00145 062212R         LDA MPFSV      RESTORE MEMORY PROTECT FLAG
      00146 071770          STA MPTFL      IN BASE PAGE
      00147 002002          SZA            WAS MEMORY PROTECT ON AT INTERRUPT?
      00150 026157R         JMP EXIT2      NO
*
      00151 062204R         LDA ASV        YES, RESTORE A-REGISTER
      00152 102100          STF 0          TURN ON INTERRUPT SYSTEM
      00153 102705          STC 5          SET MEMORY PROTECT ON
      00154 105715          JRS DMSTS PXNN,I     RESTORE DMS STATUS AND RETURN
*                                                (NOTE: EXECUTION OF A "JRS"
*                                                INSTRUCTION AFTER TURNING THE
*                                                MEMORY PROTECT FENCE ON IS
*                                                ALLOWED ONLY IF THE SYSTEM MAP
*                                                IS CURRENTLY ENABLED.  THIS
*                                                DRIVER HAS BEEN DESIGNED SUCH
*                                                THAT THIS IS ALWAYS THE CASE.
*
      00157 062204R EXIT2   LDA ASV        NO,RESTORE A-REGISTER
      00160 102100          STF 0          TURN ON INTERRUPTS
      00161 105715          JRS DMSTS PXNN,I     RESTORE DMS STATUS AND RETURN
*
* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
*
      00164 103100  DONE    CLF 0          TURN OFF THE INTERRUPT SYSTEM
      00165 106700  I.4     CLC SC         TURN OFF PRIVILEGED DEVICE
      00166 003400          CCA            SET TIME OUT FOR
      00167 172215R         STA EQ15,I     ONE TICK AND SET
      00170 162214R         LDA EQ4,I      BIT12 IN EQT4 SO
```

```
00171 032216R        IOR BIT12      RTIOC WILL
00172 172214R        STA EQ4,I      CALL CXNN ON TIME-OUT
00173 026121R        JMP EXIT       GO TO EXIT ROUTINE

                     **********************
                     * COMPLETION SECTION *
                     **********************

00174 000000   CXNN  NOP            COMPLETION SECTION ENTRY POINT

00175 002400         CLA            SET A = 0 = NORMAL RETURN
00176 165667         LDB EQT8,I     SET B = TRANSMISSION LOG
00177 126174R        JMP CXNN,I     RETURN TO IOC

```

```
*
* CONSTANT AND STORAGE AREA
*
00000                 A      EQU 0
00001                 B      EQU 1
00000                 SC     EQU 0            DUMMY I/O SELECT CODE NUMBER
*
00200 000000          SCODE  BSS 1
00201 000000          CVCTR  BSS 1
00202 000000          DAPTR  BSS 1
00203 000000          FIRST  BSS 1
00204 000000          ASV    BSS 1
00205 000000          BSV    BSS 1
00206 000000          EOSV   BSS 1
00207 000000          XSV    BSS 1
00210 000000          YSV    BSS 1
00211 000000          DMSTS  BSS 1
00212 000000          MPFSV  BSS 1
00213 000000          EQ1    BSS 1
00214 000000          EQ4    BSS 1
00215 000000          EQ15   BSS 1
00216 010000          BIT12  OCT 10000
00217 102500          LIA    LIA 0
*
* BASE PAGE COMMUNICATIONS AREA DEFINITION
*
01650                 .      EQU 1650B
01654                 INTBA  EQU .+4
01660                 EQT1   EQU .+8
01663                 EQT4   EQU .+11
01665                 EQT6   EQU .+13
01666                 EQT7   EQU .+14
01667                 EQT8   EQU .+15
01774                 EQT15  EQU .+84
01770                 MPTFL  EQU .+80
*
* CODE TO SET UP JSB $JPNN,I INSTRUCTION ON BASE PAGE
*
00220 000055R         $JPNN  DEF PXNN         PRIV. SECTION ENTRY POINT ADDR
*
00000                        ORB              RESET LOCATION COUNTER TO BASE PAGE
00000 116220R         $JSB   JSB $JPNN,I      JSB INSTR. TO PRIV. SECTION, INDIRECT
*
                             END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

POWER FAIL

A POWER FAIL INTERRUPT

OCCURS WHEN THE PRIMARY POWER DROPS BELOW A PREDETERMINED LEVEL.

IF a power fail interrupt **THEN**
- it interrupts to location 4 in memory
- the system map is enabled when DMS is in the system

ON POWER UP

IF auto restart switch in main CPU board is in the 'ON' position **THEN** the CPU will resume after ~1/2 sec.

POWER FAIL SYSTEM COMPONENTS

- Power-sensing circuits

- Memory sustaining battery (1.75 to 4.25 hours depending on memory size)

- Power fail/automatic restart driver: DVP43 ($POWR, CP43 and IP43 entries) generated into SDA.

- Automatic restart program (AUTOR)

- Automatic restart switch (ARS)

POWER FAIL EVENTS

POWER DROP

⬇

POWER FAIL INTERRUPT

⬇

JSB $POWR,I

⬇

*DVP43 POWER DOWN SEQUENCE ($POWR)

⬇

HALT CPU

⬇

POWER RESTORED

⬇

POWER FAIL INTERRUPT

⬇

*DUP43 POWER UP SEQUENCE ($POWR)

⬇

RESTART PROCESS SUSPENDED AT POWER FAIL

⬇

POWER FAIL EQT TIMES OUT (FIRST TBG TICK)

⬇

*DVP43 RESTART I/O SEQUENCE (CP43)

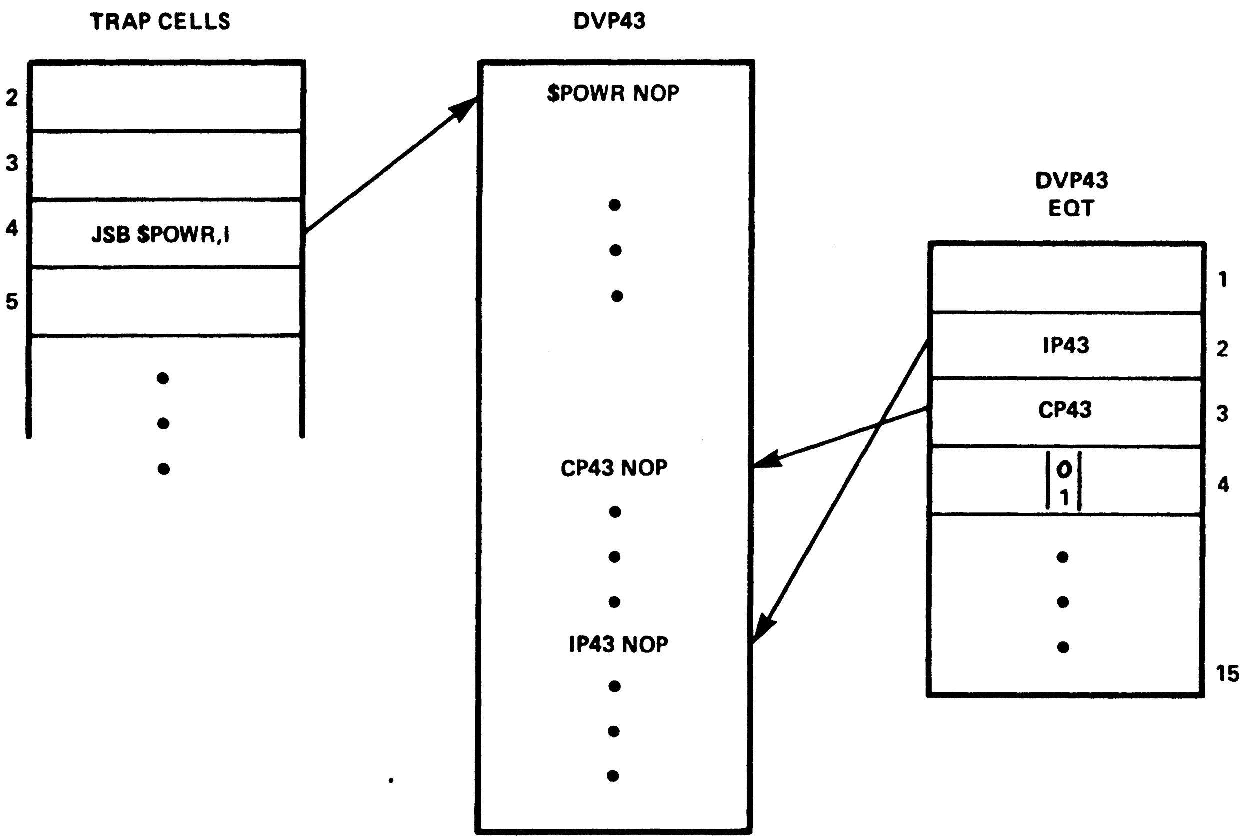
⬇

*AUTOR

⬇

SYSTEM IS READY

POWER FAIL LINKAGES



DVP43 POWER DOWN SEQUENCE*
($POWR)

```
      NAM DVP43,0
       .
       .
       .
$POWR NOP                  POWER UP/DOWN ENTRY
      SFC 4                UP?
      JMP UP               YES GO DO UP THING.

      JMP DOWN,1           GO TO DOWN ROUTINE
       .
       .
       .
DWN   STF 6B               ABORT DCPC TRANSFERS
      STF 7B
      STA ASAVE--+
      STB BSAVE  |
      ERA,ALS    |
      SOC        |
      INA        | ---- SAVE REGISTERS AND INTERRUPT LOCATION
      STA EOSAV  |
      LDA $POWR  |
      STA PSAVE  |
      STX XSAVE--+
      STY YSAVE
      LIA 5------+
      LIB 5      |
      CPB 5      | ---- SAVE MP VIOLATION ADDRESS
      STA $CIC   |
      STA $PWR5--+
      LIA 2------+
      STA SDMA1  |
      LIA 3      | ---- SAVE DCPC WORD COUNTS
      STA SDMA2--+
      LIA 1                SAVE S REGISTER
      STA SSAVE
      RSA--------+
      STA MEMST  |
      CLA        |
      LDB SMAPA  | ---- SAVE DMS STATUS AND MAP REGISTERS
      LDX MD128  |
      XMM -------+
      CLC 4                WAIT FOR POWER UP INTERRUPT
      HLT 0
```

* POWER DOWN ROUTINE MUST EXECUTE IN 500 MICRO-SECONDS.

DVP43 POWER UP SEQUENCE
($PCWR)

- Set switch so that another power fail interrupt will halt.

- Reenable power fail hardware

- Restore DMS and MP registers

- Save power fail time

- Set DVP43's time out (EQT entry word 15) to one tick (-1) and set time-out bit (EQT entry word 4).

- Restart system clock ($SCLK sets the clock up for an immediate interrupt)

- Restore registers and DCPC channels

- Return to point of power fail interrupt

DVP43 RESTART I/O SEQUENCE
(CPL43)

- For each EQT entry:

  1. If the EQT entry was busy (AV=2) and its power fail bit ("P") set, enter the driver at InXX. The driver's initiator section notes that AV=2 and thus the request is for a power-failure.

  2. If the EQT entry was waiting for a DCPC channel (AV=3), no action is taken.

  3. $UPIO is called for all other EQT entries to restart requests that were in progress or were pending by calling each driver at InXX.

- Abort and then schedule AUTOR.

- Call power fail LU to get power fail time

- Sends power fail messages to each LU which is a DVR00 or DVR05 subchannel 0 device.

- Reenable each terminal

- Calls power fail LU a second time to indicate that the recovery process is complete.

SYSTEM LIBRARY

LIBRARY SUBROUTINE STRUCTURES

- RE-ENTRANT - TYPE 6
- PRIVILEGED - TYPE 6
- UTILITY - TYPE 7

MEMORY RESIDENT LIBRARY

- TYPE 6 SUBROUTINES WHICH ARE:

  A. REFERENCED BY MEMORY RESIDENT PROGRAMS (TYPE 1, 9, 17, OR 25).

  B. REFEFENCED BY OTHER LIBRARY SUBROUTINES (TYPE 6).

- TYPE 14 SUBROUTINES WHICH ARE FORCE LOADED INTO THE MEMORY RESIDENT LIBRARY.

SUBROUTINES IN THE MR LIBRARY ARE ONLY CALLABLE

************FROM PROGRAMS IN THE MR AREA*************

DISC RESIDENT LIBRARY

- TYPE 7 SUBROUTINES

- TYPE 6 SUBROUTINES

SAMPLE LIBRARY ROUTINES

- PARSE ($PARS)
  PARSE AN ASCII STRING USING COMMAS OR DELIMITERS

- CNUMD ($CVT3) OR CNUMO ($CVT3)
  CONVERT A BINARY NUMBER TO ASCII DECIMAL OR ASCII OCTAL

- MESSS
  ISSUE A SYSTEM COMMAND

- EQLU
  FIND LU NUMBER OF AN INTERRUPTING DEVICE FROM ITS EQT ADDRESS

- IFBRK
  TESTS PROGRAMS BREAK FLAG

- INPRS
  DOES A REVERSE PARSE

- NAMR
  PARSES A FMGR NAMR

- IFTTY
  DETERMINES IF LU IS INTERACTIVE OR NOT

- LOGLU
  RETURNS LU OF TERMINAL THAT SCHEDULED PROGRAM

22-4

UTILITIES

UTILITY PROGRAMS

- LOCUS - library of contributed user software
- Software Service Kit

MAPIO(LUPRN)

CONTRIBUTION #: 598	PART NUMBER: 22682-18919
CLASS: 103	PRICE: $20
CONTRIBUTOR: LARRY SMITH	DATE CODE: 1617
HP - NEELY SANTA CLARA	LANG: ASSEMBLY RELOCATABLE
	OP SYS: RTE

MAPIO

THIS PROGRAM PRINTS A COMPACT TABLE BY LOGICAL UNIT OF ANY RTE I/II OR III I/O CONFIGURATION. THE MAP INCLUDES THE LU, EQT, SELECT CODE, SUB-CHANNEL AND CHANNEL, EQT ADDRESS, DRIVER ADDRESS, AND DEVICE TYPE. THE OUTPUT CAN BE SENT TO ANY DEVICE AND DEVICE NAMES CAN BE CHANGED BY THE USER.

HARDWARE REQUIREMENTS: TERMINAL OR PRINT DEVICE

ORDER #22682-13319 SOURCE ON CASSETTE $35.00

********************** DOCUMENTATION FILE *****************************

********************** SOFTWARE SERVICE KIT ***************************

MATERIAL LIST

PART #	REV	DESCRIPTION	
24999-16048	1727	JSAVE DK LU UTL	
24999-16049	1727	JRSTR DK LU UTL	
24999-16050	1651	SDLS4 DISK UTIL	BINARIES
24999-16051	1712	MXREF XREF ON BN	
24999-16052	1651	CMM3 MEM/DK MOD	
24999-16053	1646	FGETR GETR FILE	
24999-16055	1651	CLASS I/O UTIL	
24999-16044	1650	RECON	
24999-16163	1727	JVRFY DK LU UTIL	
24999-16167	1731	MLOAD RELOCATABLE	
24999-16168	1731	MDUMP RELOCATABLE	
24999-16171	1736	LTAT RELO - LIST TRK ASSGN TABLE.	
24999-18065	1727	JSAVE DK LU UTL	
24999-18066	1727	JRSTR DK LU UTL	
24999-18067	1651	SDLS4 DISK UTIL	
24999-18068	1712	MXREF XREF ON BN	
24999-18069	1651	CMM3 MEM/DISK MOD	
24999-18070	1646	FGETR GET FILE	SOURCES
24999-18083	1651	CLASS I/O UTIL	
24999-18052	1650	RECON	
24999-18071	1727	KIT DOCUM.	
24999-18163	1727	JVRFY DK LU UTIL	
24999-18167	1731	&MLOAD SOURCE	
24999-18168	1731	&MDUMP SOURCE	
24999-16171	1736	&LTAT SOURCE LIST TRK ASSGN TABLE.	

TABLE OF CONTENTS

PERFORMANCE MEASUREMENT

TYPES OF PROCESSES TO MEASURE

I. PROCESSES WITHOUT WAIT

examples: library functions (SIN, COS, etc.)
obtaining system time
locking an LU
going privileged
scheduling a son program

II. PROCESSES WITH WAIT

examples: I/O transfers
scheduling of disc resident programs

RTE OVERHEAD

- CONSISTS OF RTE TIME TO SERVICE TBG INTERRUPTS

- MEASURED BY EXECUTING A FIXED NUMBER OF INSTRUCTIONS IN A KNOWN TIME (F.TIME) AND CALCULATING THE ELAPSED TIME (E.TIME) OF THE INSTRUCTIONS

- RTE OVERHEAD CAN THEN BE CALCULATED:

$$TBG\% = \frac{E.TIME - F.TIME}{E.TIME} \times 100$$

- SAMPLE RTE-II OVERHEAD TIMES:

  2100A - 1.43%
  M SERIES - 3.31%
  E SERIES - 1.21%

PROGRAM TO MEASURE RTE OVERHEAD TIME

```
FTN4,L
C      6/03/76 WEIMAN
       PROGRAM SYSOH
C
C
C      MEASURES BASIC SYSTEM OVERHEAD ON RTE SYSTEMS.
C      MAY BE RUN ALONE, OR USED IN CONJUNCTION WITH
C      ANOTHER PROGRAM TO MEASURE THE SYSTEM OVERHEAD
C      THAT PROGRAM INTRODUCES.
C
C
C
C      SCHEDULE PARAMETERS:
C              #1-TTY LU FOR MESSAGE OUTPUT.
C              #2-CPU: <2 = 2100
C                       2 = 21MX
C                      >2 = 21XE
C
C      SEQUENCE OF OPERATIONS:
C      1)GET START TIME
C      2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
C      3)GET FINISHED TIME.
C      4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME AND
C        EXECUTION TIME, AS A PERCENTAGE OF ELAPSED TIME.
C
       INTEGER STIME(5),FTIME(5),IPRAM(5),IOFF,NTIME,LU
       INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
       INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
       INTEGER CPU
       REAL XTIME
C      EXECUTION TIME = DATA STORED IN "XTIME"
       EQUIVALENCE (IPRAM(1),LU)
       EQUIVALENCE (IPRAM(2),CPU)
       EQUIVILENCE (IPRAM(3),NCHAR)
       EQUIVALENCE (FTIME(1),FTIME1)
       EQUIVALENCE (FTIME(2),FTIME2),  (FTIME(3),FTIME3)
       EQUIVALENCE (FTIME(4),FTIME4),  (FTIME(5),FTIME5)
       EQUIVALENCE (STIME(1),STIME1),  (STIME(2),STIME2)
       EQUIVALENCE (STIME(3),STIME3),  (STIME(4),STIME4)
       EQUIVALENCE (STIME(5),STIME5)
       DATA XTIME/56.85568/
C
C
C
1      FORMAT(" CPU IS 2100A")
2      FORMAT(" CPU IS 21MX")
3      FORMAT(" CPU IS 21XE")
C
C      GET SCHEDULE PARAMETERS
       CALL PMPAR(IPRAM)
C      DEFAULT TTY-LU
       IF(LU .LT. 1) LU=1
C
C      USE PROPER EXECUTION TIME FOR COMPUTER BEING USED.
C
       IF(CPU .LT. 2) WRITE(LU,1)
       IF(CPU .LT. 2) XTIME=56.85568
       IF(CPU .EQ. 2) XTIME=64.40780
       IF(CPU .EQ. 2) WRITE(LU,2)
       IF(CPU .GT. 2) WRITE(LU,3)
       IF(CPU .GT. 2) XTIME=34.91070
C
C
C
C      GFT START TIME
500    CALL EXEC(11,STIME)
C      LOOP
       DO 1000 I=1,1000
       DO 1000 J=1,1000
       DO 1000 L=1,3
1000   CONTINUE
C      GET FINISHED TIME.
       CALL EXEC(11,FTIME)
C
C      COMPUTE ELAPSED TIME
C
       ETIME=(FTIME/STIME)*.01 +(FTIME2-STIME2) +
      1 (FTIME3-STIME3)*60. +(FTIME4-STIME4)*3600.
       IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
C
C      PRINT ELAPSED TIME  CPU LOAD AS PERCENTAGE
C      OF ELAPSED TIME.
2000   FORMAT(" ELAPSED TIME="F8.2"SECS.CPU LOAD="F6.3"%")
       CPULOD=(ETIME-XTIME)/ETIME *100.0
       WRITE(LU,2000) ETIME, CPULOG
2400   CONTINUE
C
C
2700   CONTINUE
       CALL EXEC(3,1100B+LU,-1)
3000   END
```

MEASURING PROCESSES WITHOUT WAIT

- WRITE A PROGRAM TO EXECUTE THE PROCESS (SIN,COS, ETC.) A LARGE NUMBER OF TIMES (1,000 TO 10,000)

- RECORD THE ELAPSED TIME OF EXECUTING THE PROCESS

- PROCESS SERVICE TIME WILL EQUAL:

$$\frac{\text{ELAPSED TIME } (1-\text{TBG}\%/100)}{\text{NUMBER OF EXECUTION TIMES}}$$

MEASURING PROCESSES WITH WAIT

- WRITE TWO PROGRAMS:

  OVRHD program executes in a fixed amount of time
  PROC program to repeatedly perform the process

- ENSURE THAT EACH PROGRAM WILL HAVE A SEPARATE PARTITION AND THAT PRIORITY OF "PROC" > "OVRHD"

- THE PROGRAMS ARE RUN SIMULTANEOUSLY WITH "OVRHD" RUNNING WHENEVER "PROC" IS WAITING

- WHEN "OVRHD" COMPLETES, IT RECORDS ELAPSED TIME AND "PROC" RECORDS NUMBERS OF PROCESSES COMPLETED

- PROCESS SERVICE TIME WILL EQUAL:

$$\frac{\text{ELAPSED TIME}(1-\text{TBG\%}/100) - \text{FIXED TIME}}{\text{NUMBER OF EXECUTION TIMES}}$$

EXAMPLE MEASUREMENT OF A PROCESS WITH WAIT

Suppose we need to know the CPU time consumed outputting characters to a terminal:

1. Program OVRHD is shown on page 24-7.
2. Process program T0002 is shown on page 24-8.
3. System common is used to:
   - Count number of EXEC 2 calls (T0002)
   - Set a start flag (T0002)
   - Set a stop flag (OVRHD)
4. Program OVRHD schedules program T0002.
5. When OVRHD completes, the time for each EXEC 2 call will equal:

$$\frac{\text{ELAPSED TIME}(1-\text{TBG\%}/100) - \text{FIXED TIME}}{\text{NUMBER OF EXEC 2 CALLS}}$$

OVRHD PROGRAM

```
FTN4,L
C       5/17/76 WEIMAN
        PROGRAM OVRHD
        COMMON IBUSY,ICOUNT,ICNTR2,IOPT,ISTOP
C
C
C       PROGRAM TO MAKE SYSTEM OVERHEAD MEASUREMENTS
C       ON RTE-II SYSTEMS
C       COMMON COMMUNICATION:  IBUSY=FLAG, SET BY
C       OVRHD WHEN IT WANTS THE FOREGROUND PROGRAM
C       TO DO SOMETHING.  IT IS CLEARED WHEN THAT
C       TASK IS DONE.
C       ICOUNT,ICNTR2 FORM A TWO-WORD COUNTER
C       IOPT=NUMBER OF WORDS/CHARACTERS
C
C
C       SCHEDULE PARAMETERS:
C               #1=TTY LU FOR MESSAGE OUTPUT.
C               #2=CPU: <2 = 2100
C                        2 = 21MX
C                       >2 = 21XE
C              #3= NUMBER OF CHARACTERS PRINTED.
C                 + = WORDS, -=CHARS.
C       SEQUENCE OF OPERATIONS:
C       1)GET START TIME
C       2)EXECUTE A FIXED TIME'S WORTH OF INSTRUCTIONS.
C       3)GET FINISHED TIME.
C       4)PRINT THE DIFFERENCE BETWEEN ELAPSED TIME
C         AND  EXECUTION TIME, AS A PERCENTAGE OF
C         ELAPSED TIME.
        INTEGER STIME(5),FTIME(5),IPRAM(5),LU
        INTEGER STIME1,STIME2,STIME3,STIME4,STIME5
        INTEGER FTIME1,FTIME2,FTIME3,FTIME4,FTIME5
        INTEGER IPROG(3)
        INTEGER CPU
        REAL CLOCK,XTIME
C       EXECUTION TIME = DATA STORED IN "XTIME"
        EQUIVALENCE (IPRAM(1),LU)
        EQUIVALENCE (IPRAM(2),CPU)
        EQUIVALENCE (IPRAM(3),NCHAR)
        EQUIVALENCE (FTIME(1),FTIME1)
        EQUIVALENCE (FTIME(2),FTIME2), (FTIME(3),FTIME3)
        EQUIVALENCE (FTIME(4),FTIME4), (FTIME(5),FTIME5)
        EQUIVALENCE (STIME(1),STIME1), (STIME(2),STIME2)
        EQUIVALENCE (STIME(3),STIME3), (STIME(4),STIME4)
        EQUIVALENCE (STIME(5),STIME5)
        DATA XTIME/56.85568/
C
C       SET THE FOREGROUND PROGRAM'S NAME
C
        DATA IPROG/2HT0,2H00,2H2 /
C
C
C       GET SCHEDULE PARAMETERS
        CALL PMPAR(IPRAM)
1       FORMAT(" CPU IS 2100A")
2       FORMAT(" CPU IS 21MX")
3       FORMAT(" CPU IS 21XE")
C       DEFAULT TTY LU
        IF(LU .LT. 1) LU=1
        IOPT=NCHAR
401     FORMAT(" # CHARACTERS="I5)
        WRITE(LU,401) IOPT
C
C       USE PROPER EXECUTION TIME FOR COMPUTER BEING
C       USED.
        IF(CPU .GE. 2) GOTO 5
C                             COMPUTER IS 2100
        WRITE(LU,1)
        XTIME=56.85568
        GOTO 15
5       CONTINUE
        IF(CPU .GT. 2) GOTO 6
C                             COMPUTER IS 21MX
        WRITE(LU,2)
        XTIME=64.40780
        GOTO 15
6       CONTINUE
C                             COMPUTER IS 21MX-E SERIES
        WRITE(LU,3)
        XTIME=34.91070
15      CONTINUE
C
C
C       SCHEDULE "SLAVE" TASK PROGRAM
C
        IBUSY=0
        ISTOP=0
100     CALL EXEC(10,IPROG,IOPT)
C
C       WAIT FOR IT TO COME IN FROM THE DISC
C
        IF(IBUSY .EQ. 0) GOTO 100
C
C       IF PROGRAM HAS ALREADY FINISHED, SKIP WAIT
C       LOOP.
        IF(IBUSY .EQ. -2) GOTO 1050
C
C
C       GET START TIME
500     CALL EXEC(11,STIME)
C       LOOP
        DO 1000 I=1,1000
        DO 1000 J=1,1000
        DO 1000 L=1,3
1000    CONTINUE
C       GET FINISHED TIME.
        CALL EXEC(11,FTIME)
C       GET # OF COUNTS
C
C       CONVERT NUMBER USING "ICOUNT" AS LOW 16 BITS,
C       AND "ICNTR2" AS HIGH 15 BITS.
C
1050    CONTINUE
        IIJ=ICOUNT
        IIK=ICNTR2
C
C       IF PROGRAM ALREADY STOPPED, SKIP WAIT
C
        IF(IBUSY .EQ. -2) GOTO 1020
C
C       SIGNAL PROGRAM TO TERMINATE
C
        ISTOP=-1
1010    IF(ISTOP .GE. 0) GOTO 1010
C
C       CONVERT COUNTERS
C
1020    CONTINUE
        III=IAND(IIJ,77777B)
        FTN=III
        IF(IIJ .LT. 0) FIN=FIN+32768.0
        FIN=FIN+IIK*65536.0
C
C       COMPUTE ELAPSED TIME
C
        FTIME=(FTIME-STIME)*.01 +(FTIME2-STIME2) +
       1 (FTIME3-STIME3)*60. +(FTIME4-STIME4)*3600.
        IF(FTIME5 .NE. STIME5) ETIME=ETIME+86400.
C
C       PRINT ELAPSED TIME,CPU LOAD AS PERCENTAGE
C       OF ELAPSED TIME, AND NUMBER OF EVENTS.
2000    FORMAT(" ELAPSED TIME="F8.2"SECS.CPU
        LOAD="F6.3 "% #EVENTS="F8.0)
        CPULOD=(ETIME-XTIME)/ETIME *100.0
        WRITE(LU,2000) ETIME,CPULOG,FIN
2400    CONTINUE
C
C
C
        CALL EXEC(3,1100B+LU,-1)
        END
```

T0002 PROGRAM

```
                      ASMB,L
      00000                  NAM T0002,2,70 5 17 76 3:30 PM
                             SUP
                             COM BUSY,COUNT,CNTR2,IOPT,ISTOP
                             EXT EXEC
      00000           T0002 EQU *
      00000 002400           CLA        ZERO THE
      00001 072001C          STA COUNT  COUNTER
      00002 072002C          STA CNTR2
      00003 002004           INA
      00004 072000C          STA BUSY
      00005           LOOP   EQU *
      00005 062004C          LDA ISTOP
      00006 002020           SSA
      00007 026023R          JMP STOP
      00010 016001X          JSB EXEC   OUTPUT SOME
      00011 000016R          DEF *+5
      00012 000032R          DEF D2       CHARACTERS
      00013 000031R          DEF D1
      00014 000033P          DEF MSG
      00015 000003C          DEF IOPT
      00016 036001C          IS7 COUNT  INCREMENT THE COUNTER
      00017 026005R          JMP LOOP
      00020 036002C          ISZ CNTR2  ROLLED OVER. BUMP
                                        OTHER COUNTER
      00021 000000           NOP
      00022 026005R          JMP LOOP
*
      00023           STOP   EQU *
      00023 002400           CLA        SET "STOPPED"
      00024 072004C          STA ISTOP   FLAG
      00025 016001X          JSB EXEC   TERMINATE
      00026 000030R          DEF *+2
      00027 000030R          DEF D6
      00030 000006    D6     DEC 6
      00031 000001    D1     DEC 1
      00032 000002    D2     DEC 2
      00033 040523    MSG    ASC 28,ASASDFASDFASDFASDFASDF
                             ASDFASDFASDFASDF
                             END T0002
** NO ERRORS *TOTAL **RTE ASMB 760924**
```

SAMPLE RTE-IV PERFORMANCE MEASUREMENT

MEASUREMENT	E-SERIES, 7905, HS-MEMORY
Program Schedule	
INTERR->MR	1.655 ms.
MR->MR	2.216
BG->MR	2.416
BG->BG	3.114
BG->EMA	3.304
TBG Overhead (0 programs in T.L.&20EQTs)	295 usec.
Overhead to go Privileged	290 usec.
Max. I/O Throughput (500 word buffer)	UB(B)
Standard	3,554(3,581) words/sec
DCPC	158,227(112,612)
Privileged	6,251(12,518)

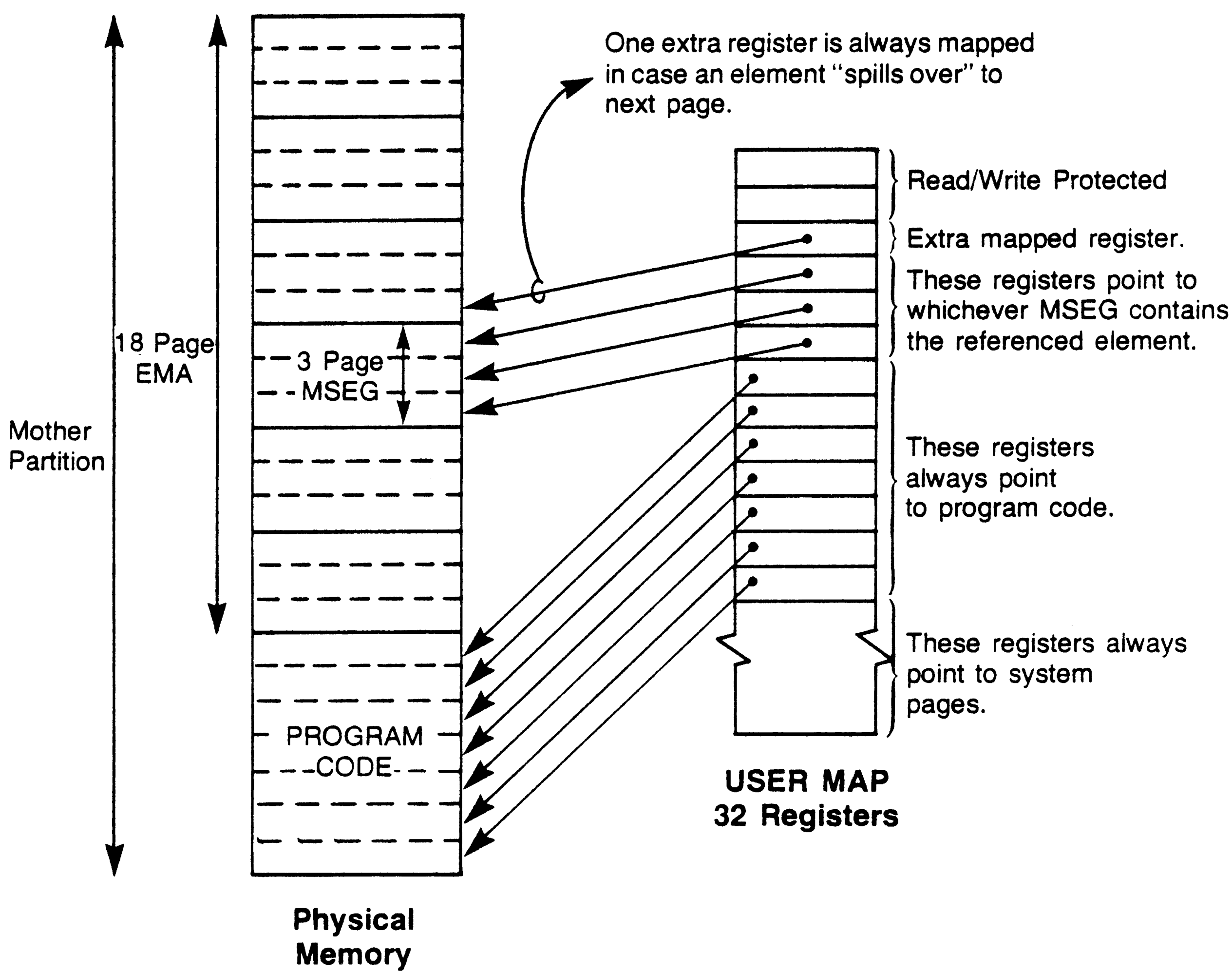
---

* See SE Note 101(4-18-78). For RTE-II see SA Note 156

EXTENDED MEMORY AREA

EMA IN A NUTSHELL

- User map registers change to point to different physical pages as needed to reference data.



(System Pages + Program Size + MSEG size + EXTRA PAGE) ≤ 32.

PROGRAMMING WITH EMA

USING EMA IN FORTRAN PROGRAMS

$EMA(blockname , mseg)

The $EMA statement listed above must be the first non-comment statement of the program. The "$" must appear in column one.

- blockname — is the name of a common block to be further defined in a named COMMON statement.
- mseg — is the size in pages of the MSEG. Specify 0 to default this size to the largest possible.

EXAMPLE EMA DECLARATION

```
FTN4,L
$EMA(XYZ,0)
         PROGRAM EXMPL
         COMMON/XYZ/IA(1000,100),IB(32767),REAL(10000)
```

The above declarations allocate 132,767 words for integer variables and 20000 words for real storage for a total of 152,767 words in EMA. The MSEG size will be defaulted.

EMA SIZE

- Refers to number of pages of physical memory necessary to contain all EMA data.

- EMA size for FORTRAN programs is set by the compiler.
  EMA size = total number of pages necessary to contain all
  variables declared to be in EMA.

EXAMPLE

```
EMA SIZE = 32 PAGES;MSEG size = default
FTN4,L
$EMA(ABC,0)
         PROGRAM EMA2
         COMMON/ABC/IA(20000),IB(12000)
                  .
                  .
                  .
                  .
```

MSEG

- MSEG refers to the pages of EMA that are currently mapped into a program's logical address space. The MSEG is the "window" of data currently accessible to the program under the current map registers.

- MSEG size is defined in a FORTRAN program with the following statement, which should be the first non-comment statement:

  $EMA (blockname, mseg)

  where

  blockname is a named common block further defined later in the program

  mseg is the MSEG size

EXAMPLE

```
FTN4,L
$EMA(XYZ,2)
      PROGRAM EMA1
      COMMON/XYZ/IA(1000,200), IB(20000)
          .
          .
          .
```

This defines arrays IA and IB to be in EMA,MSEG size equals 2.

DEFAULTING MSEG

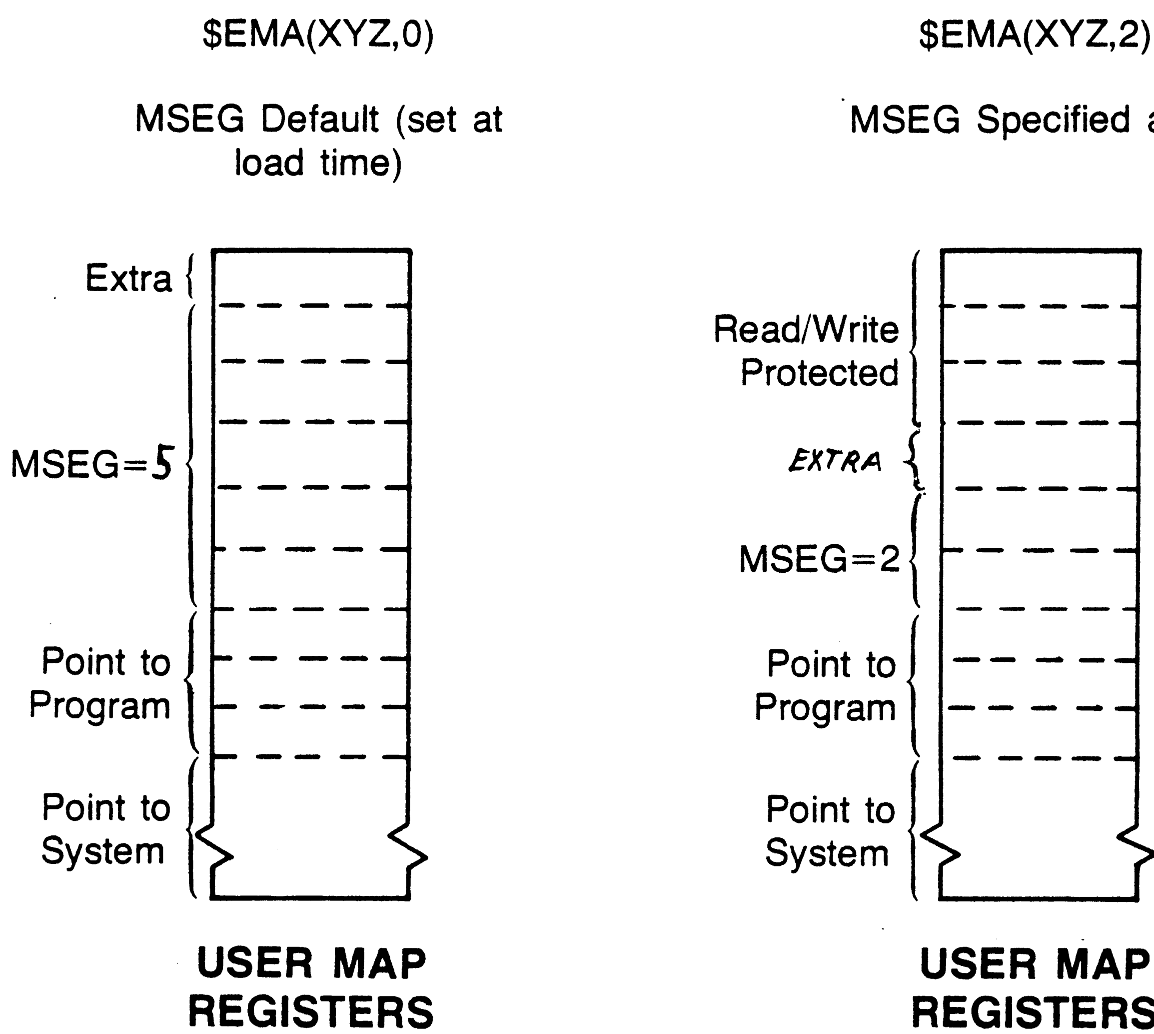
- MSEG size can be defaulted by specifying 0.

  Largest MSEG possible is used then.

  All user map registers not pointing to program and system are used to point to MSEG.

  If default used, MSEG size can be modified on-line with SZ command.

  Default value is set at load time.
  Default value = (32 − #mapped system pages − program size −1).



CONSIDERATIONS OF MSEG SIZE

- In an all-FORTRAN program executing with the EMA firmware, MSEG size makes absolutely no difference.

- The EMA firmware always maps two pages (one page containing the data and one extra page). Therefore the MSEG size makes no difference.

- The Assembly Language programmer may want to specify MSEG size since subroutines .EMIO and MMAP use the MSEG size.

LOADING/UNLOADING DATA INTO EMA IN FORTRAN

- To load/unload data into an EMA array, load a non-EMA buffer in the user program, then copy the buffer into/out of EMA. See the example program on the following pages.

- Use formatted or unformatted READ/WRITE statements

- Neither FMP or EXEC calls may be made in FORTRAN with EMA variables.

PASSING EMA VARIABLES TO SUBROUTINES

- EMA variables must be passed to subroutines using call-by-value.
  Call-by-value implies the subroutine cannot modify the actual parameter since the subroutine is passed the value and not the address of the variable.
  Call-by-value is indicated in one of two ways:

  1) Enclosing the actual parameter in an extra layer of parentheses

     ```
     CALL FUN((UA(2)))
     ```

  2) Making the actual parameter part of an arithmetic expression

     ```
     CALL FUN(IA(2)+0)
     ```

- Subroutines can only modify EMA variables by declaring the same EMA variables and then accepting the subscripts as parameters. See the example on the following pages.

```
&TEST  T=00004 IS ON CR00002 USING 00008 BLKS R=0044

C*********************************************************
C THIS PROGRAM LOADS CONSECUTIVE INTEGERS INTO A 1-D EMA
C ARRAY. THEN IT LOADS DATA INTO A 2-D EMA ARRAY OFF THE
C DISC. THEN IT CALLS A SUBROUTINE TO SQUARE THE (100,125)
C ELEMENT OF THE 2-D ARRAY.
C**********************************************************
FTN4,L
$EMA(XYZ,0)
      PROGRAM TEST
      COMMON/XYZ/IA(20000),IB(128,150)
      DIMENSION IBUF(128),IDCB(144),NAME(3)
      DATA NAME/2HFI,2HLE,2HXX/
C     ****************************************************
C     LOAD CONSECUTIVE INTEGERS INTO THE 1-D ARRAY
      DO 10 J=1,20000
      IA(J)=J
10    CONTINUE
C     ***************************************************
C     OPEN IDCB AND READ DATA FROM DISC INTO SUCCESSIVE
C     COLUMNS OF 2-D ARRAY.
      CALL OPEN(IDCB,IERR,NAME)
      IF (IERR .LT. 0) GO TO 9000
      DO 30 K=1,150
                    CALL READF(IDCB,IERR,IBUF)
                    IF(IERR .LT. 0) GO TO 9100
                    DO 20 L=1,128
                    IB(L,K)=IBUF(L)
20                  CONTINUE
30    CONTINUE
C     ********************************************************
C     CALL SUBROUTINE TO SQUARE THE (100,125) ELEMENT OF 2-D ARRAY
      CALL SQRE(100,125)
      GO TO 9999
C     ******************************************************
9000  WRITE(LU,9010) IERR
9010  FORMAT(" /TEST: OPEN ERROR,IERR=",I6)
      GO TO 9999
C     *****************************************************************
9100  WRITE(LU,9110) IERR
9110  FORMAT(" /TEST: READF ERROR,IERR=",I6)
      GO TO 9999
C     *******************************************************
9999  CALL CLOSE(IDCB)
      END
```

```
&SQRE  T=00004 IS ON CR00002 USING 00002 BLKS R=0006

FTN4,L
$EMA(XYZ,0)
      SUBROUTINE SQRE(ISUB1,ISUB2)
      COMMON/XYZ/IA(20000),IB(128,150)
      IB(ISUB1,ISUB2)=IB(ISUB1,ISUB2)*IB(ISUB1,ISUB2)
      END
```

EMA IN ASSEMBLY LANGUAGE

EMA IN ASSEMBLY LANGUAGE

- Why use Assembly Language EMA?
- EMA Statement
- EMA Subroutines
- EMAST - Returns Information about EMA
- MMAP - Maps Physical Pages Into Logical Address Space
- Table Definition for .EMIO and .EMAP
- .EMAP - Resolves References to EMA Elements
- .EMIO - Used for I/O from EMA Arrays

WHY USE ASSEMBLY LANGUAGE EMA?

- Increase speed by doing mapping only when necessary.
- Use EXEC reads and writes to perform I/O quickly to/ from EMA arrays.
- Call .EMIO to do fast I/O for non-standard buffers.

EMA STATEMENT

The following statement is required to use EMA in Assembly Language:

label EMA m1,m2

where:

label = the name assigned to the EMA area (may only be referenced directly; indirects and offsets are not allowed).

m1 = EMA size in pages .$0 \leq m1 \leq 1023$. Specifying 0 defaults EMA size.

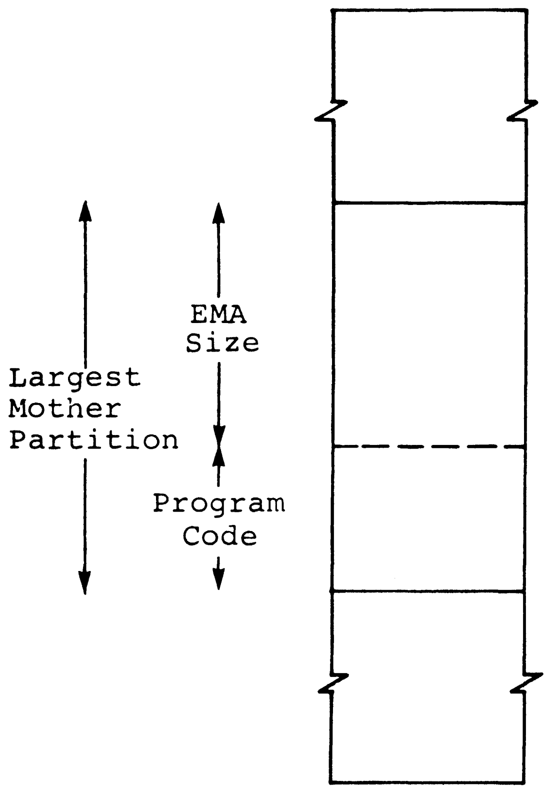
m2 = MSEG size.$0 \leq m2 \leq 31$. Specifying 0 defaults MSEG size.

- Only one EMA pseudo-op per program is allowed.
- References to EMA labels are processed as indirect addresses through a base page link at load time. This is similar to external references, except EMA labels can't be used with indirect or offset.

Refer to the RTE-IV Assembler Manual for more information.

DEFAULTING EMA SIZE

- If the EMA size is defaulted, it will be set at dispatch time to the size of the largest mother partition in the system. This default may be taken only for Assembly Language programs.



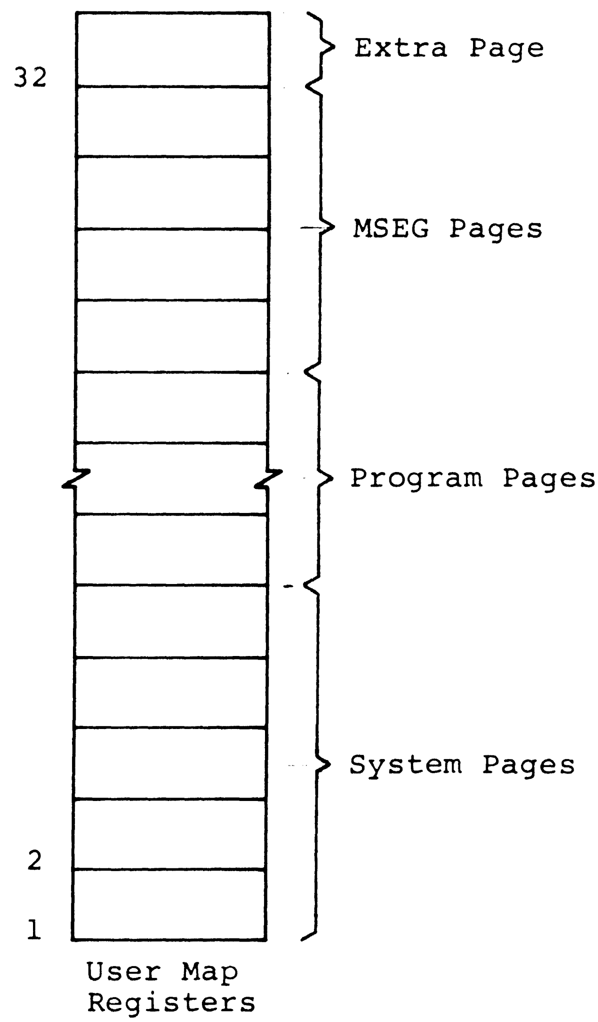
Physical Memory

DEFAULTING MSEG SIZE

- If the MSEG size is defaulted, it is set at load time according to the following formula:

  Default MSEG = 32-System Pages-Program Pages-1

- This allocates all the remaining registers to the MSEG except one that points to the extra page.



EMA SUBROUTINES

EMA is implemented by four subroutines:

1) EMAST
2) MMAP
3) .EMAP
4) .EMIO

EMAST — RETURNS INFORMATION ABOUT EMA

CALL EMAST (NEMA, NMSEG, IMSEG)

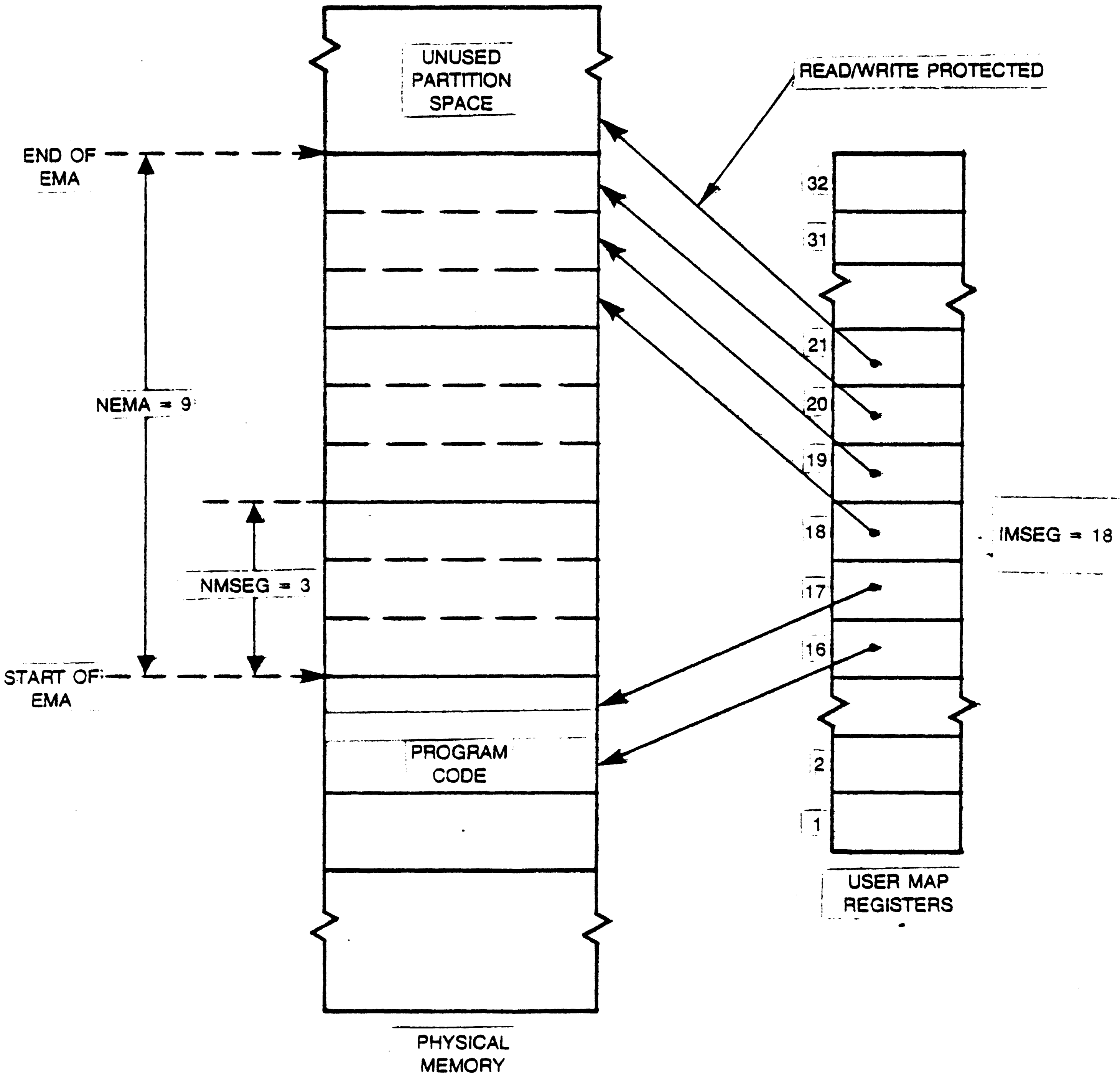
where (all values are returned):
NEMA = total size of EMA
NMSEG = size of MSEG
IMSEG = starting logical page of EMA

```
     JSB EMAST
     DEF RTN
     DEF NEMA
     DEF NMSEG
     DEF IMSEG
RTN —
```


7-6

EMAST EXAMPLE

```
      NAM   PROG
      ENT   PROG
XYZ   EMA   9,3
```



MMAP - MAPS PHYSICAL PAGES INTO LOGICAL ADDRESS SPACE

```
CALL MMAP (IPGS,NPGS)            JSB MMAP
                                 DEF RTN
                                 DEF IPGS
                                 DEF NPGS
                             RTN —
```

where:

IPGS = page offset from start of EMA to segment being mapped (origin = 0)

NPGS = number of pages to map.

- MMAP maps one more page than the NPGS specified in case an element to be referenced "spills over" to the next page above the mapped portion.
- When mapping near the top of EMA, MMAP read/write protects pages above top of EMA.
- MMAP will not map more pages than the MSEG size.

EXAMPLE MMAP CALL: CALL MMAP (6,3)

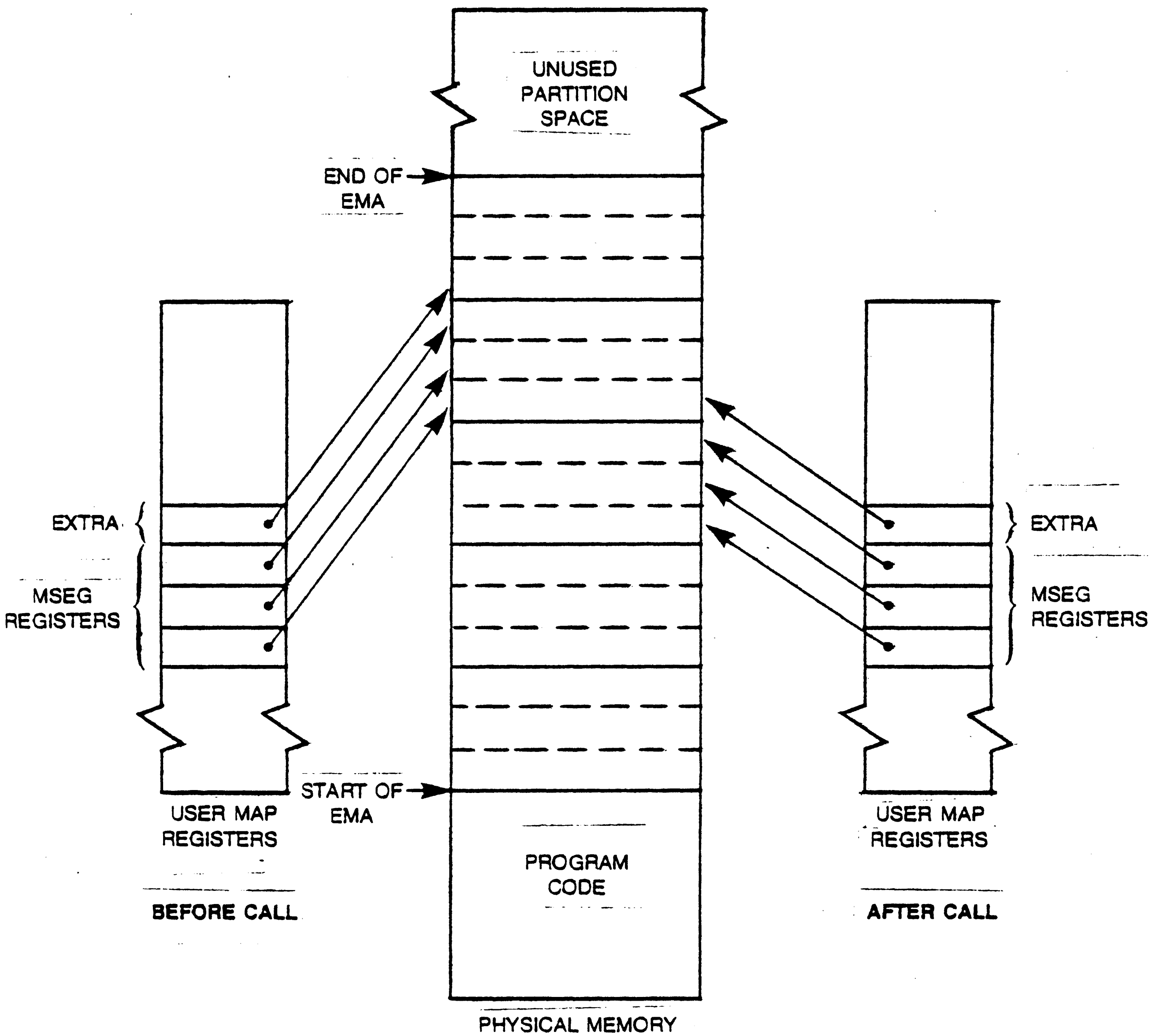


TABLE DEFINITION FOR .EMIO AND .EMAP

- The Assembly Language programmer must build a table containing information about the EMA array to make calls to .EMIO and .EMAP.

- FTN4 compiler develops the table automatically.

- The form of the table is:

TABLE:	DEC n	number of dimensions
	DEC −L(n)	negative of lower bound of n^{th} dimension
	DEC d(n−1)	number of elements in (n−1) dimension
	DEC −L(n−1)	
	DEC d(n−2)	
	DEC L(n−2)	
	•	
	•	
	•	
	•	
	DEC d(2)	
	DEC −L(2)	
	DEC d(1)	
	DEC −L(1)	
	DEC p	number of words per element
	DEC offset 1	(bits 15-0)
	DEC offset 2	(bits 31-16)

where: L(i) is the lower bound of the i^{th} dimension.
d(i) is the number of elements in the i^{th} dimension.
offset 1 and 2 specifies the number of words between the start of EMA and this array.

.EMAP — RESOLVES REFERENCES TO EMA ELEMENTS

- .EMAP maps the referenced EMA element into the program's logical address space and returns the element's logical address in the B-register.

	JSB .EMAP	
	DEF RTN	
	DEF ARRAY	name of start of EMA array
	DEF TABLE	table containing array parameters
	DEF A_n	actual subscript for n^{th} dimension
	DEF A_{n-1}	actual subscript for (n−1)st dimension
	.	
	.	
	.	
	DEF A_2	
		actual subscript for 2nd dimension
	DEF A_1	
		actual subscript for 1st dimension
RTN	-error return-	A-reg = 15 (ASCII) B-reg = EM (ASCII)
	normal return	B-reg = logical address of element in current map

where: ARRAY is the name of the start of the EMA array.
TABLE is as previously defined.

- FORTRAN compiler emits calls to .EMAP to resolve EMA references.

.EMAP - SOFTWARE AND FIRMWARE DIFFERENCES

SOFTWARE .EMAP:

Checks whether referenced element is already mapped.

YES - returns logical address of element in current map.

NO - maps in complete MSEG containing element, then returns logical address of element in new map.

FIRMWARE .EMAP:

Always maps two pages, then returns logical address of element in new map. First page contains element; second page is mapped in case element "spills over" to next page.

7-12

```
&EMALD T=00004 IS ON CR01001 USING 00003 BLKS R=0027

ASMB,L,T
*********************************************************
* EXAMPLE ASSEMBLER PROGRAM, USING .EMAP, TO STORE      *
* 9000 CONSECUTIVE INTEGERS INTO AN EMA ARRAY OF SIZE
* 9K, WITH AN MSEG SIZE OF 2K.                          *
*********************************************************
      NAM EMALD
      EXT .EMAP
      EXT DBUGR
      EXT EXEC
EMALB EMA 9,2        EMA SIZE=9 PAGES, MSEG SIZE=2 PAGES
LOOP  NOP            BEGIN EXECUTION
**********************************************************
* CALL .EMAP TO MAP ELEMENT AND RETURN PTR IN B-REG     *
**********************************************************
      JSB .EMAP      CALL .EMAP TO MAP ELEMENT
      DEF RTN        DEF RETURN ADDRESS
      DEF EMALB      DEF EMA AREA
      DEF TABLE      DEF TABLE FOR USE BY .EMAP
      DEF COUNT      DEF COUNTER FOR INDEX OF EMA ARRAY
RTN   JSB DBUGR      ERROR RETURNED
**********************************************************
* NORMAL RETURN- B-REG HOLDS PTR TO ELEMENT IN CURRENT *
* LOGICAL ADDRESS SPACE                                 *
**********************************************************
      LDA COUNT      GET INDEX OF ARRAY

      STA 1,I        STORE COUNT IN EMA ARRAY
      CPA TOP        CHECK TO SEE IF DONE
      JMP DONE       YES, EXIT
      ISZ COUNT      ADD ONE TO COUNT
      JMP LOOP       CONTINUE PROCESS WITH COUNT=COUNT+1
**********************************************************
* DATA STORAGE                                          *
**********************************************************
COUNT DEC 1
TOP   DEC 9000       STORE 9000 ELEMENTS
TABLE DEC 1          # OF DIMENSIONS IN ARRAY
      DEC -1         NEGATIVE OF LOWER BOUND OF FIRST DIM
      DEC 1          # WORDS ELEMENT
      DEC 0
      DEC 0
DONE  JSB EXEC       TERMINATE EXECUTION
      DEF *+2
      DEF D6
D6    DEC 6          EXEC REQUEST CODE FOR TERMINATION
      END LOOP
```

EXAMPLE PROGRAM FOR .EMAP AND MMAP

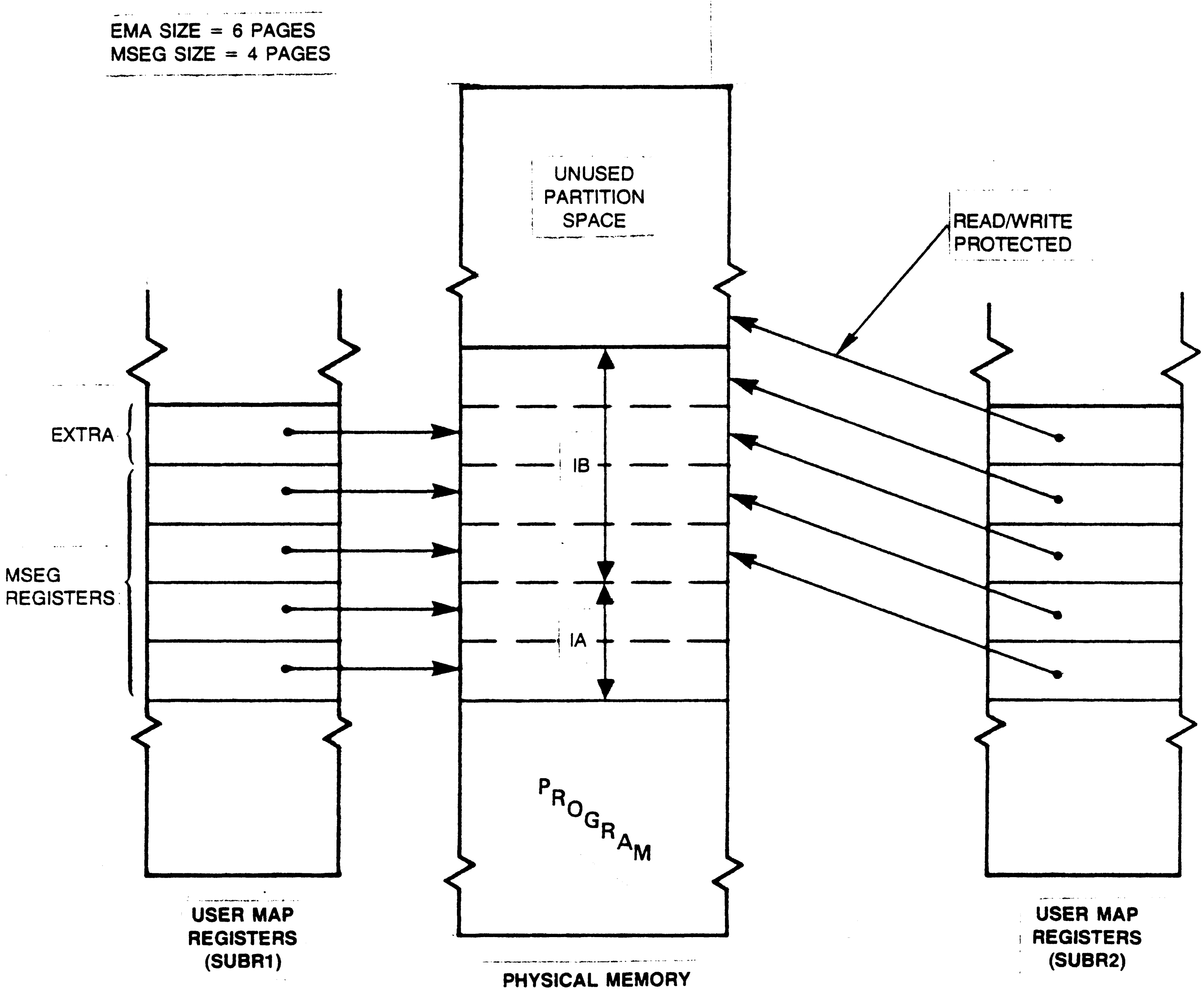
The following program illustrates the use of .EMAP and MMAP to load and manipulate EMA data.

SUBR1 and SUBR2 illustrate two different ways to access EMA variables. SUBR1 uses .EMAP to load 2048 consecutive values into IA. SUBR2 does its own mapping and uses the EMA label XYZ to load 4096 consecutive values into IB.

SUBR3 uses both techniques to add IA(2000) to IB(4000). Note especially the array table for IB in SUBR3 which contains an offset of 2048 (lines 60-61). This needs to be specified to reference IB since IB starts 2048 elements above the start of EMA.

7-

DIAGRAM FOR PROGRAM EMAP



```
C*******************************************************
C PROGRAM ILLUSTRATING THE USE OF .EMAP. THIS PROGRAM
C LOADS CONSECUTIVE VALUES INTO TWO EMA ARRAYS. ONE LOAD
C USES .EMAP AND THE OTHER LOAD USES THE EMA LABEL AND DOES
C ITS OWN CALL TO MMAP.
C*******************************************************
FTN4,L,T
$EMA(XYZ,4)
      PROGRAM EMAP
      COMMON/XYZ/IA(2048),IB(4096)
      DIMENSION IPRAM(5)
C     *****************************************************
      CALL RMPAR(IPRAM)
      LU=IPRAM(1)
      IF (LU .LE. 0) LU=1
C     ***************************************************
C     LOAD CONSECUTIVE VALUES INTO IA USING .EMAP.
      CALL SUBR1
      DO 10 J=1,2048
      IF (IA(J) .NE. J) CALL DBUGR
10    CONTINUE
C     **************************************************
C     LOAD CONSECUTIVE VALUES INTO IB USING THE EMA LABEL
      CALL SUBR2
      WRITE(LU,900) (IB(K),K=1,10)
900   FORMAT(" EMAP OUTPUT"//" FIRST 10 VALUES OF IB=",10I4)
      DO 20 K=1,4096
      IF (IB(K) .NE. K) CALL DBUGR
20    CONTINUE
C     *************************************************
C     SUBR3 CALLS MMAP AND .EMAP TO ADD IA(2000) TO IB(4000)
      ITEMP=IA(2000)+IB(4000)
      WRITE(LU,2000) ITEMP
2000  FORMAT(/" BEFORE SUBR3,IA(2000)+IB(4000)=",I6)
      CALL SUBR3
      WRITE(LU,3000) IB(4000)
3000  FORMAT(" AFTER SUBR3,IB(4000)=",I6)
      IF (IB(4000) .NE. ITEMP) CALL DBUGR
C     *************************************************
      WRITE(LU,1000)
1000  FORMAT(//" /EMAP:ENDING NOW"//)
      END
```


7-15

```
ASMB,R,L
***************************************************************
* SUBR1 LOADS 2048 CONSECUTIVE VALUES INTO THE ARRAY NAMED
* IA. IA'S ARRAY TABLE IS LISTED AT TABLE.
***************************************************************
      NAM SUBR1,7
      ENT SUBR1
      EXT XYZ
      EXT .EMAP
      EXT DBUGR
SUBR1 NOP
LOOP  NOP
************************************************************
* CALL .EMAP TO MAP ELEMENT AND RETURN PTR IN B-REG
************************************************************
      JSB .EMAP
      DEF RTN1
      DEF XYZ
      DEF TABLE
      DEF COUNT
RTN1  JSB DBUGR
***************************************************************
* NORMAL RETURN- B-REG HOLDS PTR TO ELEMENT IN CURRENT
* LOGICAL ADDRESS SPACE.
***************************************************************
      LDA COUNT
      STA 1,I
      CPA TOP
      JMP DONE
      ISZ COUNT
      JMP LOOP
DONE  NOP
      JMP SUBR1,I
***************************************************************
* DATA STORAGE STARTS HERE
***************************************************************
COUNT DEC 1
TOP   DEC 2048
TABLE DEC 1            # OF DIMENSIONS IN ARRAY
      DEC -1           NEGATIVE OF LOWER BOUND OF FIRST DIM
      DEC 1            # WORDS/ELEMENT
      DEC 0
      DEC 0
      END
```

```
ASMB,L,R
*****************************************************************
* SUBR2 LOADS 4096 CONSECUTIVE INTEGERS INTO ARRAY IB, USING
* THE EMA LABEL OF XYZ.
*****************************************************************
      NAM SUBR2,7
      ENT SUBR2
      EXT XYZ
      EXT DBUGR
      EXT MMAP
ADXYZ DEF XYZ
SUBR2 NOP
***********************************************************
* CALL MMAP TO MAP IN EMA PAGES CORRESPONDING TO IB ARRAY
      JSB MMAP
      DEF RTN1
      DEF IPGS
      DEF NPGS
RTN1  CPA MIN1
      JSB DBUGR
**********************************************************
* LOAD PTR TO EMA INTO B-REG
      LDB ADXYZ
      RBL,CLE,SLB,ERB GET RID OF SIGN BIT
      LDB 1,I       RESOLVE ONE INDIRECT
***********************************************************
* LOAD COUNT INTO A-REG, STORE IT THROUGH THE B-REG
LOOP  LDA COUNT
      STA 1,I
      CPA TOP
      JMP DONE
***********************************************************
* BUMP B-REG, BUMP COUNT, LOOP BACK **
      ADB D1
      ISZ COUNT
      JMP LOOP
**********************************************************
* RETURN
DONE  JMP SUBR2,I
***********************************************************
* DATA VALUES START HERE
D1    DEC 1
MIN1  DEC -1
COUNT DEC 1         COUNTS HOW MANY VALUES LOADED
TOP   DEC 4096      TOP VALUE LOADED
IPGS  DEC 2         PAGE OFFSET FOR MMAP CALL
NPGS  DEC 4         # OF PAGES MAPPED IN CALL TO MMAP
      END
```

```
ASMB,L,R
**************************************************************
* SUBR3 ADDS IA(2000) TO IB(4000) AND STORES THE RESULT IN
* IB(4000)
**************************************************************
      NAM SUBR3,7
      ENT SUBR3
      EXT XYZ
      EXT .EMAP
      EXT MMAP
      EXT DBUGR
ADXYZ DEF XYZ
SUBR3 NOP
*************************************************************
* GET IA(2000) USING MMAP TO DO YOUR OWN MAPPING
      JSB MMAP
      DEF RTN1
      DEF IPGS
      DEF NPGS
RTN1  CPA MIN1
      JSB DBUGR
**************************************************************
* LOAD A PTR TO IA(2000) INTO B-REG
      LDB ADXYZ
      RBL,CLE,SLB,ERB
      LDB 1,I
      ADB D1999     ADD IN 1999 OFFSET TO GET IA(2000)
**************************************************************
* PUT IA(2000) INTO A-REG,STORE IT AT ARG1
      LDA 1,I
      STA ARG1
**************************************************************
* USE .EMAP TO GET PTR TO IB(4000)
      JSB .EMAP
      DEF RTN2
      DEF XYZ
      DEF TBL2
      DEF D4000
RTN2  JSB DBUGR
*************************************************************
* NORMAL RTN, B-REG PTS TO IB(4000)
* LOAD THE SECOND ARG INTO A-REG.
      LDA 1,I
*************************************************************
* ADD IN ARG1 AND STORE BACK INTO IB(4000)
      ADA ARG1
      STA 1,I
      JMP SUBR3,I
*************************************************************
* DATA STORAGE STARTS HERE.
ARG1  NOP           VALUE OF IA(2000) STORED HERE
IPGS  DEC 0         MMAP MAPS STARTING AT 0 OFFSET
NPGS  DEC 4         MMAP MAPS 4 PAGES
MIN1  DEC -1        CHECKS ERROR RETURN FOR MMAP CALL
D1999 DEC 1999      OFFSET INTO ARRAY
D4000 DEC 4000      ELEMENT # FOR .EMAP CALL
TBL2  DEC 1         # OF DIMENSIONS IN ARRAY
      DEC -1        NEG OF LOWER BOUND OF FIRST DIMENSION
      DEC 1         # WORDS/ELEMENT
      DEC 2048      OFFSET FROM START OF EMA FOR THIS ARRAY
      DEC 0         HIGH BITS OF OFFSET
      END
```

```
EMAP OUTPUT

FIRST 10 VALUES OF IH=    1    2    3    4    5    6    7    8    9   10

BEFORE SUBR3,IA(2000)+IB(4000)=   6000
AFTER SUBR3,IB(4000)=   6000


/EMAP:ENDING NOW
```


7-19
6A-10

.EMIO - USED FOR I/O FROM EMA ARRAYS

- .EMIO - used to ensure that an entire memory buffer is in a program's logical address space (possibly for I/O purposes).
- Only callable in Assembly language.
- Allows EXEC I/O and FMP calls with EMA buffers.
- .EMIO does the following:

  1) Checks if buffer fits into any possible mapping
     YES - Go to Step 2
     NO - Error Return

  2) Check if buffer fits into standard MSEG
     YES - map in the MSEG, return logical address of buffer.
     NO - go to Step 3

  3) Map in a non-standard MSEG and return logical address of buffer in the non-standard MSEG.

- The user buffer plus a possible page offset of the start of the buffer should be less than the MSEG size. A good rule of thumb is to always make sure that the buffer is at least one page smaller than the MSEG size.
- .EMIO always maps an MSEG size number of pages.

.EMIO CALLING SEQUENCE

- .EMIO callable only from Assembly Language.

```
EXT  .EMIO
  :
JSB .EMIO
DEF RTN
DEF BUFL              LENGTH OF BUFFER
DEF TABLE             TABLE DESCRIBING EMA ARRAY
DEF a(n)
DEF a(n-1)
   .
   .                  ARRAY SUBSCRIPT VALUES FOR START
   .                  OF BUFFER
DEF a(2)
DEF a(1)
```

```
RTN - error return -  A-reg = 16 (ASCII) B-req = EM(ASCII)
      normal return   B-reg = logical address of the element
                              in current map.
                      A-reg = meaningless.
```

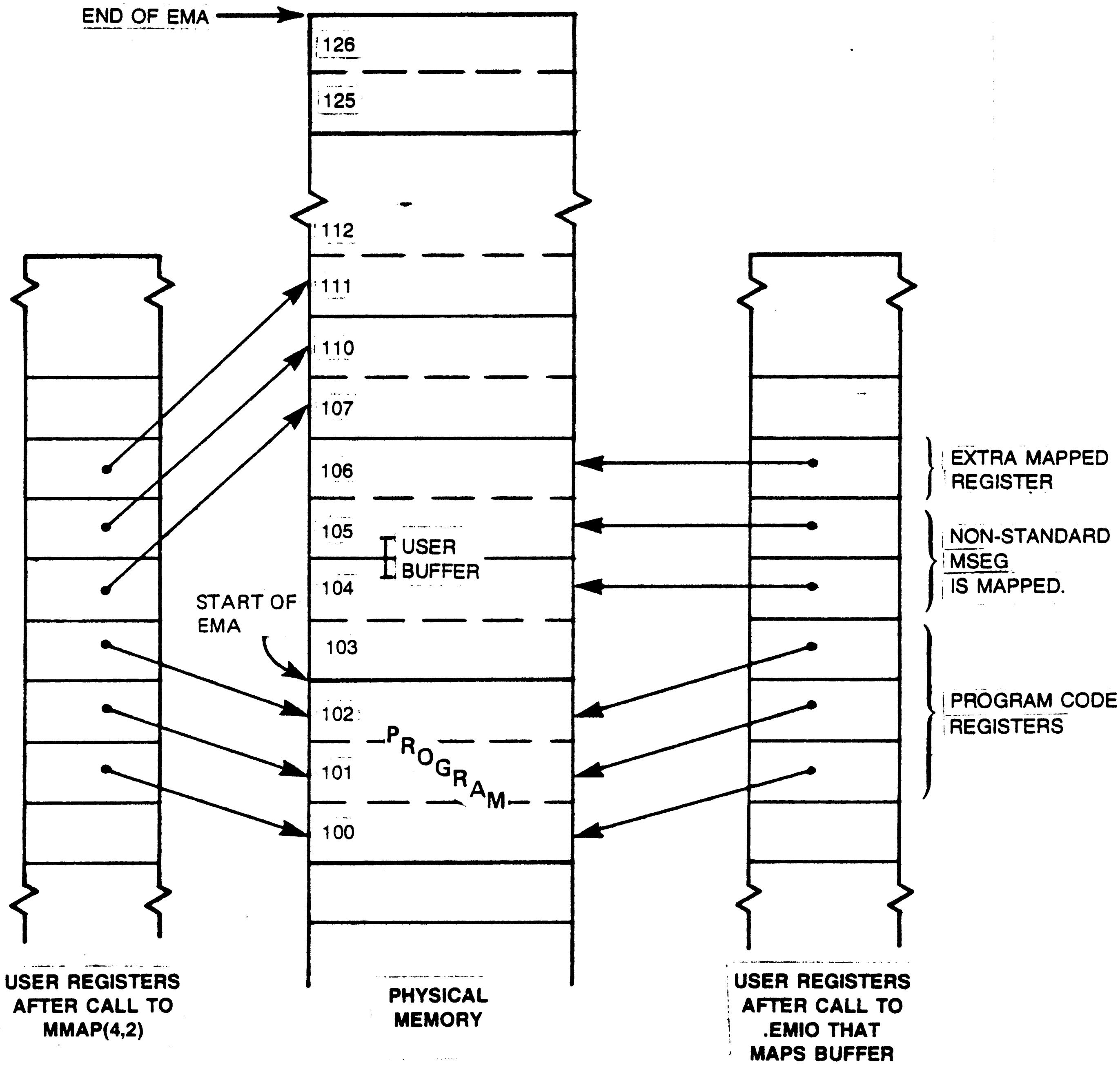
EXAMPLE PROGRAM FOR .EMIO CALL

The following example program demonstrates the use of .EMIO to output a buffer that crosses an MSEG boundary. First it loads the buffer with ASCII data (lines 20 to 26), puts out some explanatory material (lines 29-42), then calls SUBR to output the buffer.

SUBR calls .EMIO to map in a non-standard MSEG, then calls EXEC to output the desired buffer. After SUBR returns, program EMIO outputs a listing of the user registers (lines 50-52).
Note expecially registers 14 and 15 in the printed output that reflect the non-standard MSEG that was mapped in.

DIAGRAM EXPLAINING .EMIO EXAMPLE PROGRAM

REGISTERS 14 AND 15 POINT TO EMA AREA; USER WANTS TO OUTPUT BUFFER THAT CROSSES MSEG BOUNDARY.



Note: One extra register is always mapped in case an element overflows to the next page.

```
C*****************************************************
C PROGRAM TO SHOW USE OF .EMIO. THIS PROGRAM CALLS A
C SUBR THAT CALLS .EMIO TO MAP IN A BUFFER THAT CROSSES
C AN MSEG BOUNDARY.
C*************************************************
FTN4,L
$EMA(XYZ,2)
      PROGRAM EMIO
      COMMON/XYZ/IA(20000)
      DIMENSION IMAP(32),IPRAM(5)
C     ************************************
      CALL RMPAR(IPRAM)
      LU=IPRAM(1)
      IF (LU .LE. 0) LU=1
      WRITE(LU,1400)
1400  FORMAT(///" OUTPUT FROM EMIO DEMO PROGRAM"///)
C     ***********************************************
C     PUT MESSAGE INTO BUFFER
C     NOTE:MESSAGE CROSSES AN MSEG BOUNDARY
      DO 20 K=2038,2058,5
      IA(K)=2HEM
      IA(K+1)=2HIO
      IA(K+2)=2HWO
      IA(K+3)=2HRK
      IA(K+4)=2HS!
20    CONTINUE
C     **********************************************
C     IMSEG=FIRST REGISTER THAT POINTS TO EMA
      CALL EMAST(NEMA,NMSEG,IMSEG)
      WRITE(LU,1500) IMSEG
1500  FORMAT(/" /EMIO: THIS REG POINTS TO FIRST PAGE MSEG",I6/)
C     ******************************************
C     MAP TWO PAGES OF EMA STARTING AT OFFSET OF 4 PAGES
      CALL MMAP(4,2)
C     ***********************************************
C     IMAP IS A COPY OF THE USER MAP REGISTERS
      WRITE(LU,1600)
1600  FORMAT(/" THE 32 USER REGISTERS ARE:")
      CALL EXEC(26,IFPG,ILMEM,NPGS,IMAP)
      WRITE(LU,1000) (IMAP(J),J=1,32)
1000  FORMAT(10O7)
      WRITE(LU,9000)
C     ********************************************
C     CALL SUBR THAT CALLS .EMIO AND OUTPUTS A BUFFER
C     THAT CROSSES AN MSEG BOUNDARY.
      CALL SUBR
      WRITE(LU,9000)
C     ********************************************
C     CALL EXEC TO GET INFO ON USER MAP AGAIN
      CALL EXEC(26,IFPG,ILMEM,NPGS,IMAP)
      WRITE(LU,2000)(IMAP(J),J=1,32)
2000  FORMAT(10O7)
C     ******************************************
C     FINISH UP THE PROGRAM
      WRITE(LU,3000)
3000  FORMAT(//" ENDING PROGRAM EMIO"/)
9000  FORMAT(//)
      END
```

```
ASMB,R,L
***********************************************************
* THIS SUBR CALLS .EMIO TO MAP IN A NONSTANDARD MSEG
* THAT CROSSES AN MSEG BOUNDARY. THEN IT CALLS EXEC
* TO OUTPUT A BUFFER THAT CROSSES THE MSEG BOUNDARY
***********************************************************
      NAM SUBR,7
      ENT SUBR
      EXT .EMIO
      EXT ERROR
      EXT EXEC
SUBR  NOP
***************************************************************
* SET UP MSEG TO POINT TO ELEMENTS 2038-2058 IN EMA
****************************************************************
      JSB .EMIO
      DEF RTN1
      DEF BUFL
      DEF TABLE
      DEF A1
RTN1  JSB ERROR
**********************************************************
* NORMAL RETURN, B-REG HOLDS LOGICAL ADDRESS OF ELEMENT A1
      STB ADDR
**********************************************************
* JSB EXEC TO OUTPUT BUFFER TO LU 6
      JSB EXEC
      DEF RTN2
      DEF ICODE
      DEF ICNWD
      DEF ADDR,I
      DEF BUFL
RTN2  NOP
**********************************************************
* RETURN POINT
      JMP SUBR,I
******************************************************
* PARAMETERS FOR CALLS TO .EMIO AND EXEC
BUFL  DEC 20          BUFFER LENGTH
A1    DEC 2038        INDEX OF ARRAY ELT
ICODE DEC 2           EXEC WRITE ICODE
ICNWD OCT 206         OUTPUT TO LU 6,PRINT COLUMN ONE
****************************************************
* TABLE FOR CALL TO .EMIO
TABLE DEC 1           # OF DIMENSIONS
      DEC -1          NEGATIVE OF LOWER BOUND OF FIRST DIMENSION
      DEC 1           # OF WORDS PER ELT
      DEC 0           OFFSET WORD #1
      DEC 0           OFFSET WORD #2
*****************************************************
* ADDR HOLDS LOGICAL ADDRESS OF DESIRED ELT AFTER CALL
* TO .EMIO AND STB INSTRUCTION
ADDR  DEC 0
***************************************************
      END SUBR
```

```
)UTPUT FROM EMIO DEMO PROGRAM


/EMIO: THIS REG POINTS TO FIRST PAGE MSEG      14


THE 32 USER REGISTERS ARE:
000077 000001 000002 000003 000004 040005 040006 040007 040010 040011
040012 000100 000101 000102 000107 000110 000111 140001 140002 140003
140004 140005 140006 140007 140010 140011 140012 140013 140014 140015
140016 140017
```

EMA Registers

Extra Register

```
EMIOWORKS!EMIOWORKS!EMIOWORKS!EMIOWORKS!


000077 000001 000002 000003 000004 040005 040006 040007 040010 040011
040012 000100 000101 000102 000104 000105 000106 140001 140002 140003
140004 140005 140006 140007 140010 140011 140012 140013 140014 140015
140016 140017

:NDING PROGRAM EMIO
```


7-2.
6A-76

FUNCTION	NUMBER OF SUBSCRIPTS							
	STANDARD MEMORY (μs)				HIGH SPEED MEMORY (μs)			
	0D	1D	2D	3D	0D	1D	2D	3D
21MX Software								
.EMAP no map map	326 1,080+ 16.8M	429 1,182+ 16.8M	529 1,283+ 16.8M	629 1,384+ 16.8M				
.EMIO no map map	321 1,083+ 16M	424 1,186+ 16M	525 1,288+ 16M	624 1,386+ 16M				
21MX-E μCode								
.EMAP	35	45	55	65	32	40	49	58
.EMIO no map map	44 87+1.3M	54 96+1.3M	64 106+1.3M	74 115+1.3M	40 82+1.2M	49 92+1.2M	57 98+1.2M	66 109+1.2M
M = #MSEG								

APPENDIX A

SYSTEM TABLES

- ID segments, long, short, & extensions
- Equipment Table
- Device Reference Table
- Interrupt Table
- Track Assignment Table
- Class Table
- LU Switch Table
- Resource Number Table
- Keyword Block
- ID Extension Table
- Memory Allocation Table
- Memory Protect Fence Table
- Driver Mapping Table
- Track Map Table

SYSTEM LISTS

- Schedule List
- General Wait List
- Available Memory Suspend List
- Disc Allocation Suspend List
- Operator Suspend List
- I/O Suspend Lists
- Free SAM List

SYSTEM BASE PAGE COMMUNICATION AREA

Octal Location	Contents	Description
SYSTEM TABLE DEFINITION		
01645	XIDEX	Address of current program's ID extension
01646	XMATA	Address of current program's MAT entry
01647	XI	Address of index register save area
01650	EQTA	FWA of Equipment Table
01651	EQT#	Number of EQT entries
01652	DRT	FWA of Device Reference Table, word 1
01653	LUMAX	Number of logical units in DRT
01654	INTBA	FWA of Interrupt Table
01655	INTLG	Number of Interrupt Table Entries
01656	TAT	FWA of Track Assignment Table
01657	KEYWD	FWA of keyword block
I/O MODULE/DRIVER COMMUNICATION		
01660	EQT1 \	
01661	EQT2 |	
01662	EQT3 |	
01663	EQT4 |	
01664	EQT5 \	Addresses of first 11 words of
01665	EQT6 /	current EQT entry (see 01771 for
01666	EQT7 |	last four words
01667	EQT8 |	
01670	EQT9 |	
01671	EQT10 |	
01672	EQT11 /	
01673	CHAN	Current DCPC channel number
01674	TBG	I/O address of time-base card
01675	SYSTY	EQT entry address of system TTY
SYSTEM REQUEST PROCESSOR/EXEC COMMUNICATION		
01676	RQCNT	Number of request parameters -1
01677	RQRTN	Return point address
01700	RQP1 \	
01701	RQP2 |	
01702	RQP3 |	Addresses of request parameters (set
01703	RQP4 \	for a maximum of nine parameters)
01704	RQP5 /	
01705	RQP6 |	
01706	RQP7 |	
01707	RQP8 |	
01710	RQP9 /	

SYSTEM BASE PAGE COMMUNICATION AREA (continued)

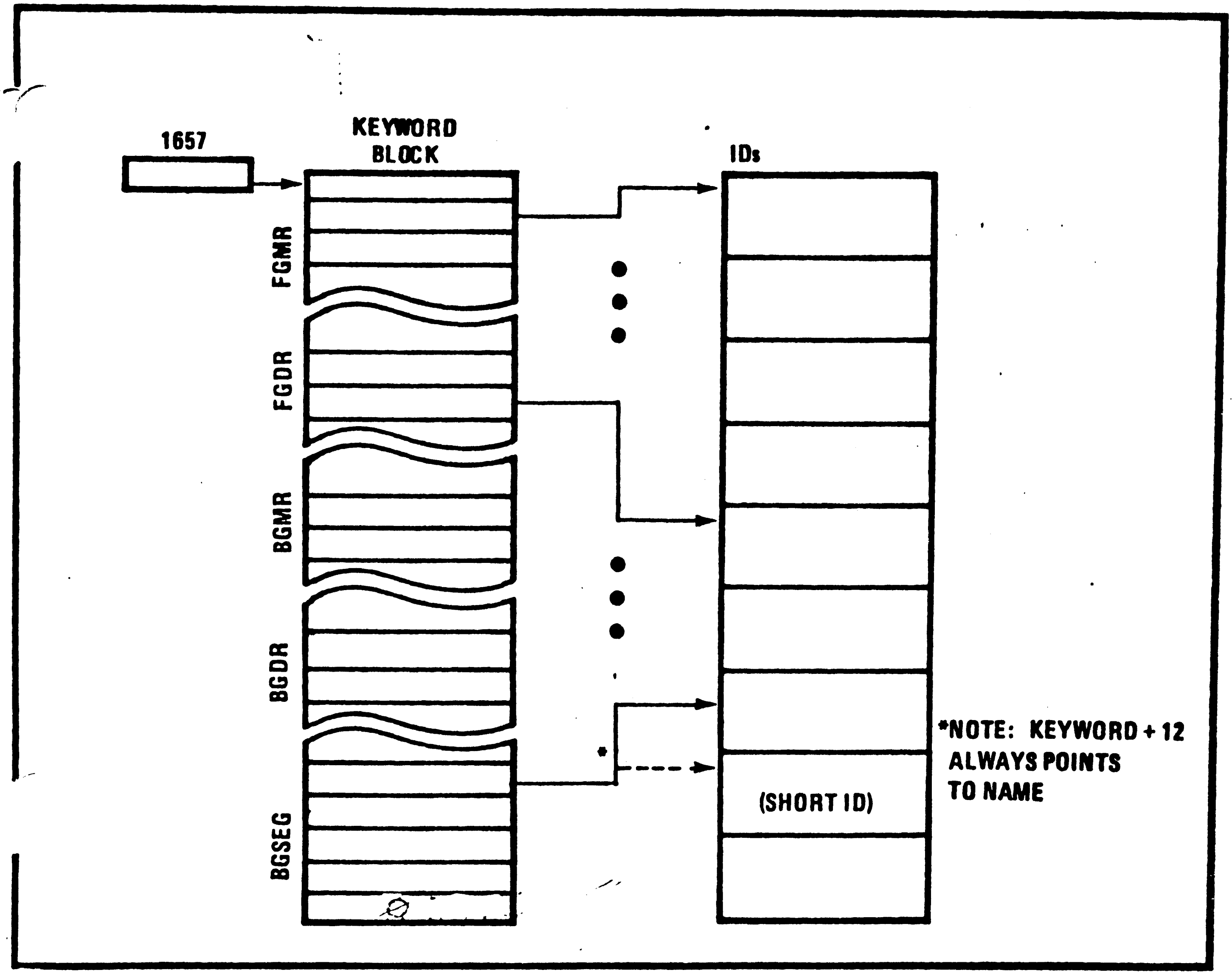
Octal Location	Contents	Description
SYSTEM LISTS ADDRESSES		
01711	SKEDD	Schedule list
01713	SUSP2	Wait Suspend list
01714	SUSP3	Available Memory list
01715	SUSP4	Disc Allocation list
01716	SUSP5	Operator Suspend list
PROGRAM ID SEGMENT DEFINITION		
01717	XEQT	ID segment address of current program
01720	XLINK	Linkage
01721	XTEMP	Temporary (five words)
01726	XPRIO	Priority word
01727	XPENT	Primary entry point
01730	XSUSP	Point of suspension
01731	XA	A-register at suspension
01732	XB	B-register at suspension
01733	XEO	E and overflow register suspension
SYSTEM MODULE COMMUNICATION FLAGS		
01734	OPATN	Operator/keyboard attention flag
01735	OPFLG	Operator communication flag
01736	SWAP	RT disc resident swapping flag
01737	DUMMY	I/O address of dummy interface flag
01740	IDSDA	Disc address of first ID segment
01741	IDSDP	Position within disc sector
MEMORY ALLOCATION BASES DEFINITION		
01742	BPA1	FWA user base page link area
01743	BPA2	LWA user base page link area
01744	BPA3	FWA user base page link
01745	LBORG	FWA of resident library area
01746	RTORG	FWA of real-time COMMON
01747	RTCOM	Length of real-time COMMON
01750 D	RTDRA	FWA of real-time partition
01751 D	AVMEM	LWA+1 of real-time partition
01752	BGORG	FWA of background COMMON
01753	BGCOM	Length of background COMMON
01754 D	BGDRA	FWA of background partition

SYSTEM BASE PAGE COMMUNICATION AREA (continued)

Octal Location	Contents	Description
UTILITY PARAMETERS		
01755	TATLG	Negative length of track assignment table
01756	TATSD	Number of tracks on system disc
01757	SECT2	Number of sectors/track on LU2 (system)
01760	SECT3	Number of sectors/track on LU3 (aux.)
01761	DSCLB	Disc address of user available library entry points
01762	DSCLN	Number of user available library entry points.
01763	SYSLB	Disc address of system library entry points
01764	SYSLN	Number of system library entry points
01765	LGOTK	LGO: LU#, starting trrack, number of tracks (same format as ID segment word 28)
01766	LGOC	Current LGO track/sector address (same format as ID segment word 26)
01767	SFCUN	LS: LU# and disc address (same format as ID segment word 26)
01770	MPTFL	Memory protect ON/OFF flag (0/1)
01771	EQT12 \	
01772	EQT13 \	Address of last four
01773	EQT14 /	words of current EQT
01774	EQT15 /	
01775 D	FENCE	Memory protect fence address
01777	BGLWA	LWA memory background partition
D letter indicates the contents of the location are set dynamically by the dispatcher.		

BASE PAGE EXAMPLE

```
LOCATIONS    1647 THROUGH     1746
036306 002023 000023 002762 000051 003104 000072 023717*<          ) D :'
016071 002136 002137 002140 002141 002142 002143 002144* 9 ↑ ←
002145 002146 002147 002150 000006 000015 002042 000003*                "
051474 000002 047337 050534 051534 000000 000000 000000*S<  N Q\S\
000000 000000 016535 000000 016406 000000 000000 000000*       ]
016535 016535 016536 016537 016540 016541 016542 016543* ] ] ↑ ←
016544 016545 016546 016547 016550 000000 000000 031017*                 2
000000 000224 000057 000002 001445 000002 026000 011107*      /    %  ,  G
LOCATIONS    1747 THROUGH     1777
000144 052654 052654 011253 000525 052654 177400 000400*  U U     UU
000140 000000 010422 001224 004646 000234 000000 000000*
000000 000000 002151 002152 002153 002154 030000 000000*               0
052654                                                 *U
```



I.D. definition:

Location 1657B on base page specifies the first entry in the keyword block. The keyword block in turn contains 1-word entries, each pointing to an ID segment*, Last entry = 0. Keyword Block entries are ordered at generation by program type.

The keyword block entry (ID address) + 12 always points to the name-word. Thus, keyword entries for short ID's don't point to the first word of the ID.

ID SEGMENT

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--+
| List Linkage                                  |   Word  0  \
|-----------------------------------------------|            |
| TEMP 1                                        |         1  |
| TEMP 2                                        |         2  |
| TEMP 3                                        |         3  |
| TEMP 4                                        |         4  |
| TEMP 5                                        |         5  |
|-----------------------------------------------|            |
| Priority                                      |         6  |
| Primary Entry Point                           |         7 *|
|-----------------------------------------------|            |
| Point of Suspension                           |         8  |
| A-Register                                    |         9  |
| B-Register                                    |        10  |
| EO-Registers                                  |        11  |
|-----------------------|-----------------------|            | Memory
| Name 1                | Name 2                |        12 *\ Resident
| Name 2                | Name 4                |        13 */ Programs
|                       |--|--|--|--|-----------|            |
| Name 3                |TM|ML|//|SS| Type      |        14 *|
|--|--|--|--|--|--|--|--|--|--|--|--|-----------|            |
|NA|//|NP| W| A|//| O|//| R| D|/////| Status    |        15  |
|--|--|--|--|--|--|--|--|--|--|-----|-----------|            |
| Time List Linkage                             |        16  |
|--------|--|-----------------------------------|            |
| RES    | T| Multiple                          |        17  |
|--------|--|-----------------------------------|            |
| Low Order 16 Bits of Time                     |        18  |
|-----------------------------------------------|            |
| High Order 16 Bits of Time                    |        19  |
|--|--|--|--|--|--|--|--|-----------------------|            |
|BA|FW| M|AT|RM|RE|PW|RN| Father ID Segment No. |        20  |
|--|--|--|--|--|--|--|--|--|--|-----------------|            |
|RP|#pgs. (no BP) | MPFI    |//| Partition No. -1|        21  |
|--|--------------|---------|--|-----------------|            |
| Low Main Address                              |        22 *|
|-----------------------------------------------|            |

|-----------------------------------------------|
| High Main Address + 1                         |        23 *|
|-----------------------------------------------|            |
| Low Base Page Address                         |        24 *|
|-----------------------------------------------|            |
| High Base Page Address + 1                    |        25 */
|--|--------------------|-----------------------|
|LU| Program: Track     | Sector                |        26 *
|--|--------------------|-----------------------|
|LU| Swap:    Track     | No. Tracks            |        27
|--|--------------|-----|-----------------------|
|ID Extension No. | EMA Size                    |        28
|-----------------|-----------------------------|
| High Address + 1 of Largest Segment           |        29
|-----------------------------------------------|
| Reserved                                      |        30\
|-----------------------------------------------|           \Memory
| Reserved                                      |        31
|-----------------------------------------------|           /Residents
| Negative MTM LU number                        |        32/
|-----------------------------------------------|
```

where:

* = words used in short ID segments for program segments

TM = temporary load (copy of ID segment is not on the disc)

ML = memory lock (program may not be swapped)

SS = short segment (indicates a nine-word ID segment)

Type = specified program type (1-5)

NA = no abort (instead, pass abort errors to program)

NP = no parameters allowed on reschedule

W = wait bit (waiting for program whose ID segment address is in word 2)

A = abort on next list entry for this program

O = operator suspend on next schedule attempt

R = resource save (save resources when setting dormant)

D = dormant bit (set dormant on next schedule attempt)

Status = current program status

T = time list entry bit (program is in the time list)

BA = batch (program is running under batch)

FW = father is waiting (father scheduled with wait)

M = Multi-Terminal Monitor bit

AT = attention bit (operator has requested attention)

RM = reentrant memory must be moved before dispatching program

RE = reentrant routine now has control

PW = program wait (some other program wants to schedule this one)

RN = Resource Number either owned or locked by this program

RP = reserved partition (only for programs that request it)

MPFI = memory protect fence index

LONG ID SEGMENT

ID SEG OF LUMAP

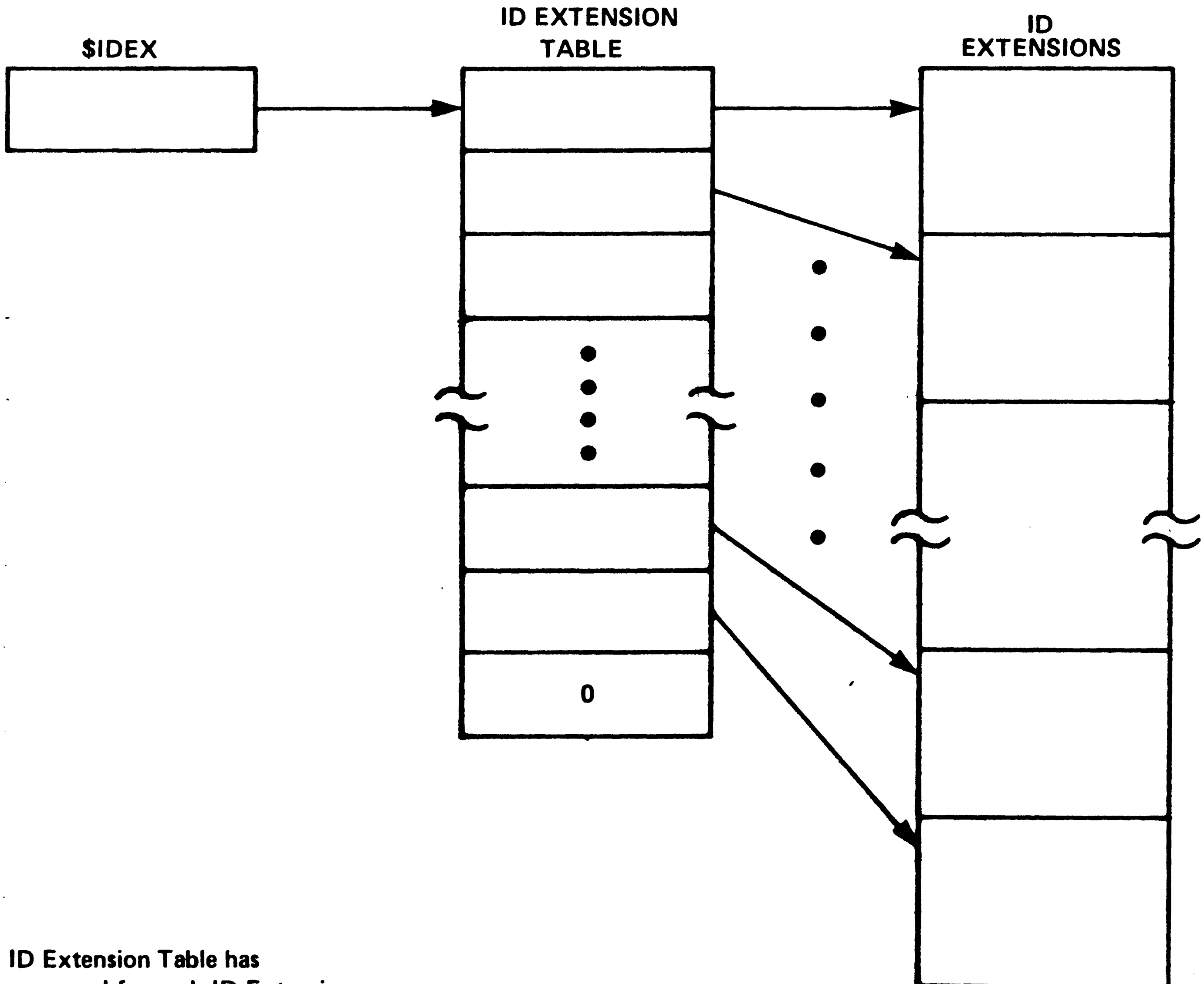
WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	17202	17551	8041	
2	17203	15	13	
3	17204	103750	-30744	
4	17205	177776	- 2	
5	17206	0	0	
6	17207	0	0	
7	17210	24	20	
8	17211	26120	11344	,P
9	17212	26221	11409	,
10	17213	0	0	
11	17214	17203	7811	
12	17215	126214	-21364	
13	17216	46125	19541	LU
14	17217	46501	19777	MA
15	17220	50003	20483	P
16	17221	0	0	
17	17222	0	0	
18	17223	0	0	
19	17224	25000	10752	*
20	17225	177574	- 132	
21	17226	0	0	
22	17227	11205	4741	
23	17230	26000	11264	,
24	17231	34056	14382	8.
25	17232	2	2	
26	17233	23	19	
27	17234	4122	2130	P
28	17235	0	0	
29	17236	0	0	
30	17237	0	0	
31	17240	0	0	
32	17241	0	0	
33	17242	177777	- 1	

SHORT ID SEGMENT

ID SEG OF FMGR0

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	22653	32770	13816	5
2	22654	43115	17997	FM
3	22655	43522	18258	GR
4	22656	30025	12309	0
5	22657	32770	13816	5
6	22660	41032	16922	B
7	22661	45	37	%
8	22662	105	69	E
9	22663	2714	1484	

ID EXTENSION TABLE



ID Extension Table has
one word for each ID Extension.

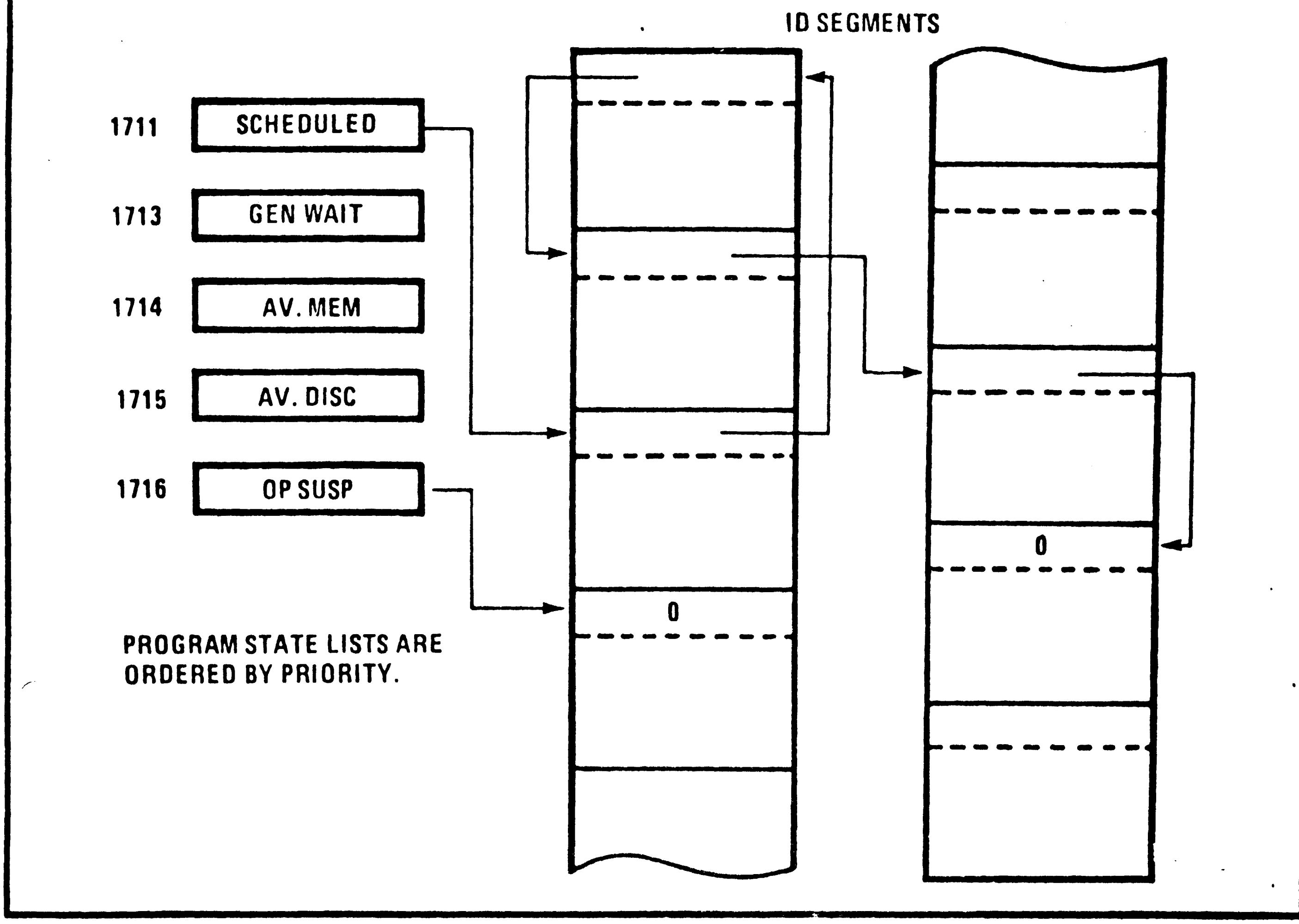
ID EXTENSION

```
+----------------------------------------------------------------------+
|                                                                      |
|    15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0                   |
|    +--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|-+                  |
|    |  |                          |                |    Word 0        |
|    |NS|     Current MSEG No.     | # Pages MSEG|                     |
|    |--|----------|--|------------|------------|                      |
|    | MSEG Start  |DE| (Physical) EMA Start    |    Word 1            |
|    | Page (logic.)|  | Page                    |                      |
|    |--|----------|--|-------------------------|                      |
|    |//////////////////| # Tracks for EMA Swap  |    Word 2            |
|    |//////////////////|                        |                      |
|    +-----------------|------------------------+                      |
|                                                                      |
|    where:                                                            |
|                                                                      |
|      NS = 0 if the MSEG is pointing to a standard segment of         |
|              the EMA (set up by .EMAP)                               |
|                                                                      |
|         = 1 if the MSEG is pointing to a non-standard segment        |
|              (set up by .EMIO or .EMAP)                              |
|                                                                      |
|      DE = 0 if the EMA size was specified by the user                |
|                                                                      |
|         = 1 if the EMA size is allowed to default to the             |
|               maximum size available to the system                   |
|                                                                      |
+----------------------------------------------------------------------+
```

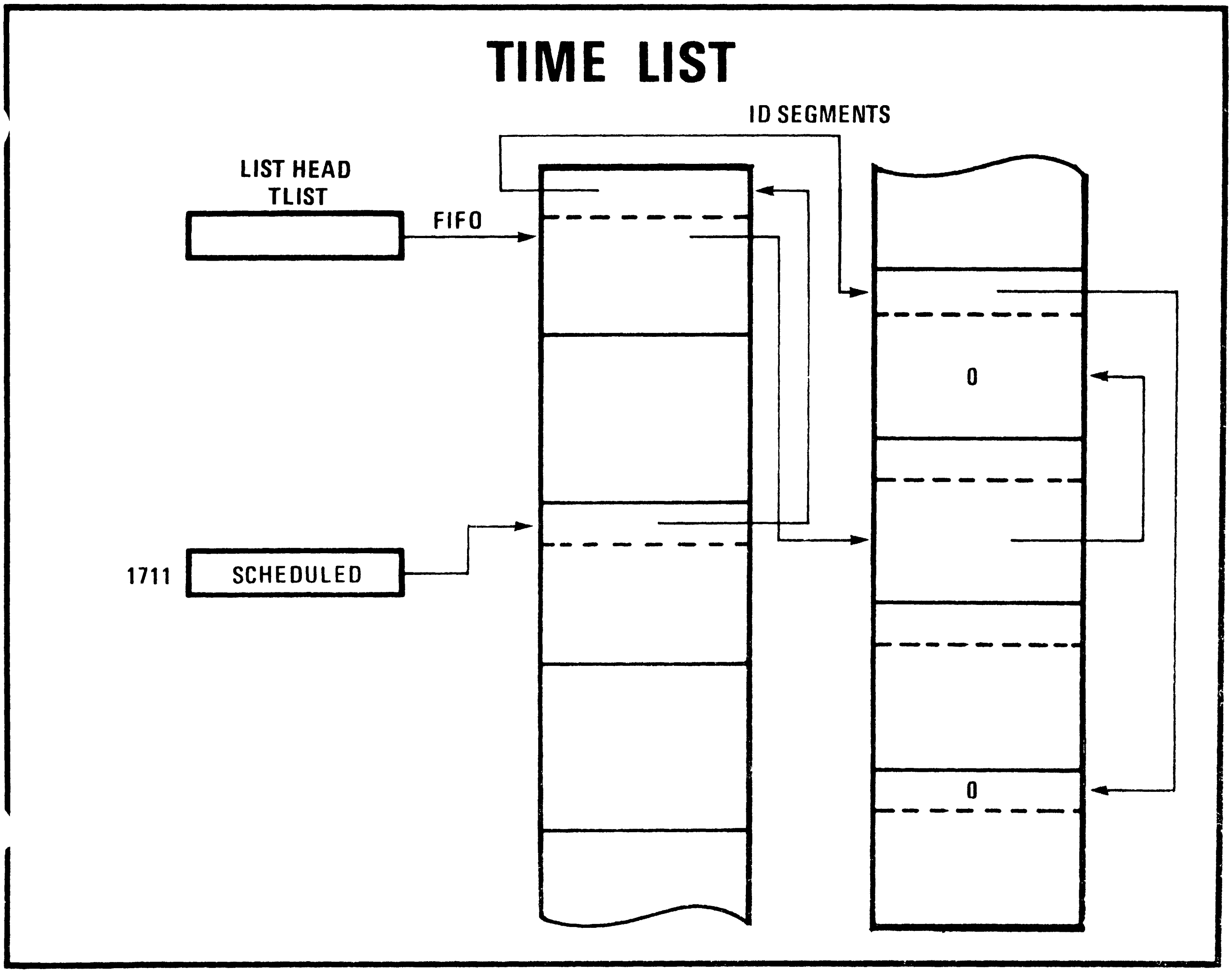
A program's ID segment word 28 contains its ID extension number.

PROGRAM STATES

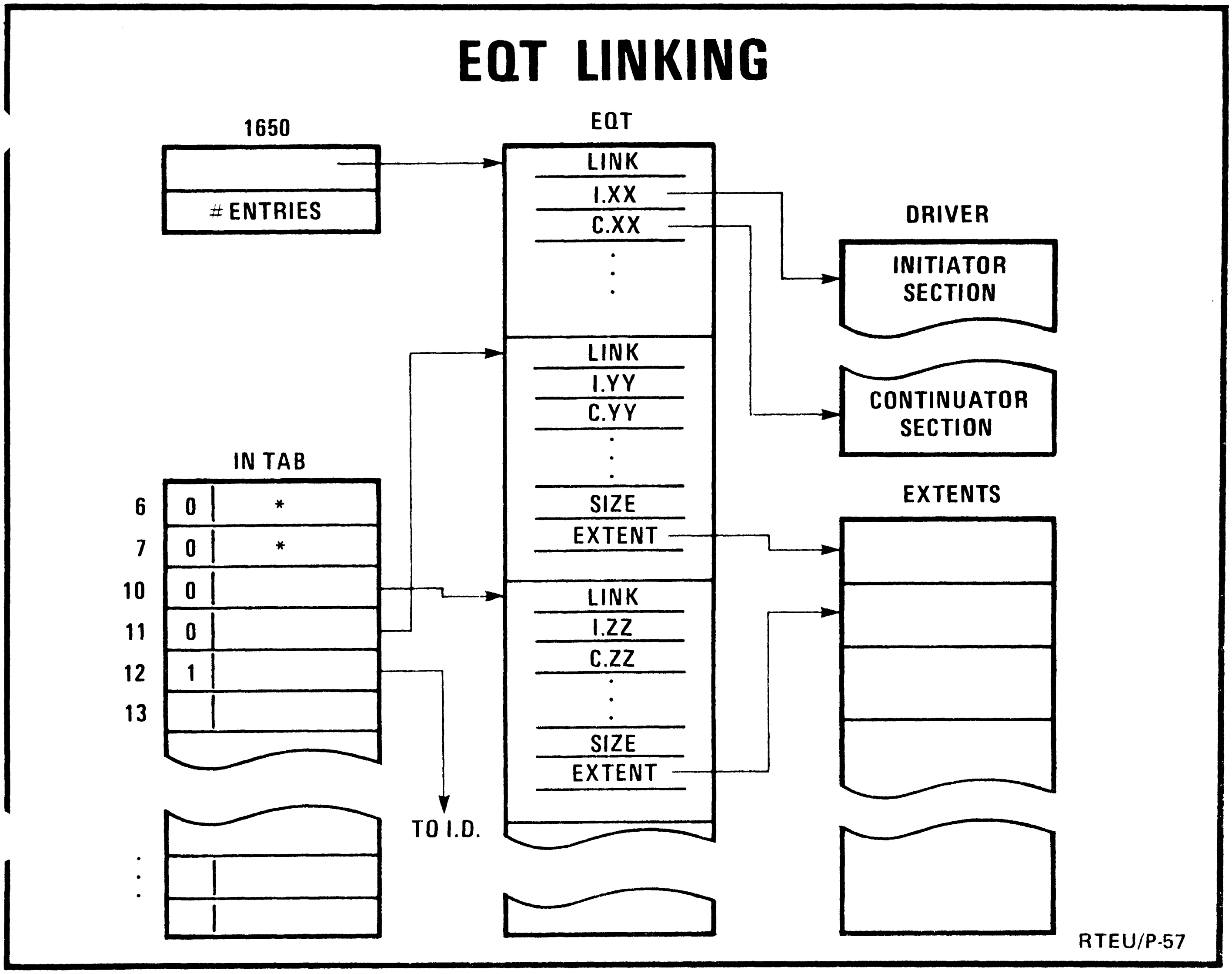
LIST LINKING



A linked list of ID segments is maintained for each of the major program states (accept I/O, to be seen later). The lists are prioritized and have their heads on Base Page. ID link word is word #1.



The time list's head is at TLIST in RTIME, and is linked thru ID word 17. It threads independently of the state lists, and is FIFO (vs prioritized).



* INTAB entries 6 and 7 are dynamic, reflecting the assignment of DMA channels to various EQT's as needed.

\+ = EQT
\- = PRG/ENT
0 = not used

EQUIPMENT TABLE:

```
EQUIPMENT TABLE:
----------------

        WORD              CONTENTS
        ----    -----------------------------
          1     * I/O LIST . LINK POINTER   *
          2     *DRIVER *INITIATION ADDRESS*
          3     *DRIVER *COMPLETION ADDRESS*
          4     *DBPOT/----UNIT#--CHANNEL #*
          5     *AV-TYPE CODE- UNIT  STATUS*
          6     *REQUEST   CONTROL    WORD *
          7     *REQUEST  BUFFER   ADDRESS *
          8     *REQUEST  BUFFER   LENGTH  *
          9     *TEMPORARY OR DISC TRACK # *
         10     *TEMPORARY OR DISC SECTOR #*
         11     *DRIVER   TEMPORARY STORAGE*
         12     *  "         "        "    *   (EXT. SIZE)
         13     *  "         "        "    *   (EXT. ADR.)
         14     * DEVICE CLOCK RESET VALUE *
         15     *    "      "  WORKING  "  *


D: =1 IF A DMA CHANNEL REQUIRED FOR TRANSFER
B: =1 IF AUTOMATIC OUPUT BUFFERING DESIRED
P: =1 IF DRIVER TO HANDEL POWER FAIL RECOVERY.
O: =1 IF DRIVER TO HANDEL TIME OUT.
T: DEVICE TIME-OUT BIT - CLEARED BEFORE EACH
            IO INITIATION; SET IF DEVICE TIMES-OUT.
UNIT#: OPTIONAL FOR DEVICES REQUIRING
        SUB-CHANNEL DESIGNATION
CHANNEL#: I/O SELECT CODE (LOWER # IF
          MULTI-BOARD INTERFACE)
AV (AVAILABILITY INDICATOR):          (SEE ALSO DRT FOR LU'S)
   =0, UNIT AVAILABLE FOR OPERATION
   =1, UNIT DISABLED
   =2, UNIT CURRENTLY IN OPERATION
   =3, UNIT WAITING FOR DMA CHANNEL
TYPE CODE: CODE IDENTIFYING TYPE OF I/O DEVICE
UNIT STATUS: ACTUAL OR SIMULATED UNIT STATUS
            AT END OF OPERATION


  SPECIFICATION

      LOCATION       BASE PAGE  1650
      # OF ENTRIES   BASE PAGE  1651
```

EQT EXAMPLE

EQT 13 DVR 05

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	35400	0	0	
2	35401	25042	10786	*"
3	35402	25132	10842	*Z
4	35403	40026	16406	@▮
5	35404	2400	1280	
6	35405	0	0	
7	35406	0	0	
8	35407	0	0	
9	35410	0	0	
10	35411	0	0	
11	35412	0	0	
12	35413	15	13	
13	35414	36127	15447	<W
14	35415	150437	-12001	
15	35416	0	0	

EXTENT

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	36127	0	0	
2	36130	0	0	
3	36131	0	0	
4	36132	0	0	
5	36133	0	0	
6	36134	0	0	
7	36135	0	0	
8	36136	0	0	
9	36137	0	0	
10	36140	0	0	
11	36141	0	0	
12	36142	0	0	
13	36143	0	0	

DEVICE REFERENCE TABLE (DRT)

THE DEVICE REFERENCE TABLE PROVIDES LOGICAL ADDRESSING OF PHYSICAL UNITS DEFINED IN THE EQUIPMENT TABLE.
THE 'DRT' ENTRIE FOR THE LU'S CONSISTS OF 2- TABLES.
THE LENGHT OF EACH TABLE IS DEFINED IN 'LUMAX'.
TABLE 1 IS A 1- WORD ENTRY CORRESPONDING TO THE RANGE OF USER-SPECIFIED "LOGICAL" UNITS, 1 TO N WHERE N IS LT OR = TO 63(10). THE CONTENTS OF THE WORD CORRESPONDING TO A LOGICAL UNIT IS THE RELATIVE POSITION OF THE EQT ENTRY DEFINING THE ASSIGNED PHYSICAL UNIT:

TABLE 2 HAS A 1-WORD ENTRIE FOR EACH LU DEFINED IN TABLE 1 AND CONTAINS THE STATUS OF THE CORRESPONDING LU.HIS POSITION IN MEMORY IS RIGHT AFTER TABLE 1.
EACH ENTRY MAY REPRESENT 4 STATES.

	BIT 15	BITS 0-14	
STATE 1	0	0	=LU UP
STATE 2	1	0	=LU DOWN (NO STACKED I/O)
STATE 3	1	ADR	=LU DOWN STACKED I/O
STATE 4	1	LU	=LU DOWN I/O STACKED ON 2. LU

TABLE 1 ENTRIE:

15 11	10 6	5 0
SUB CHANNEL	LOCKING RN #	EQT NUMBER

TABLE 2 ENTRY:

15	14 0
UP/DN	ADDRESS OR LU POINTER OR 0

SPECIFICATION:

LOCATION	BASE PAGE 1652	DRT
LENGTH	BASE PAGE 1653	LUMAX

INTERRUPT TABLE:
----

1 WORD PER SPECIFIED SELECT CODE. CONTENTS:

(+) = ADDRESS OF FIRST WORD OF EQT ENTRY

(-) = ADDRESS OF PROGRAM ID SEGMENT

0 = NO ENTRY

SPECIFICATION:

LOCATION	BASE PAGE	1654	INTBA
#ENTRIES	BASE PAGE	1655	INTLG

FIRST ENTRY IS FOR S.C. = 6 (DMA 1)
THE FIRST TWO WORDS ARE DYNAMIC (AS DMA ASSIGNMENT CHANGES)

INTERRUPT TABLE EXAMPLE

INT TABLE WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
6	36545	0	0	
7	36546	0	0	
10	36547	35323	15059	
11	36550	35342	15074	
12	36551	35361	15089	
13	36552	35246	15014	
14	36553	0	0	
15	36554	35152	14954	:J
16	36555	35152	14954	;J
17	36556	35265	15029	
20	36557	35265	15029	
21	36560	35133	14939	:[
22	36561	35114	14924	:L
23	36562	35171	14969	;Y
24	36563	35227	14999	
25	36564	35210	14984	
26	36565	140670	-15944	
27	36566	35417	15119	;■
30	36567	140670	-15944	
31	36570	140670	-15944	
32	36571	35645	15269	
33	36572	35664	15284	
34	36573	0	0	
35	36574	140670	-15944	
36	36575	140670	-15944	
37	36576	140670	-15944	
40	36577	140670	-15944	
41	36600	140670	-15944	
42	36601	140670	-15944	
43	36602	35626	15254	
44	36603	0	0	
45	36604	35304	15044	
46	36605	0	0	
47	36606	0	0	
50	36607	0	0	
51	36610	0	0	
52	36611	0	0	
53	36612	0	0	
54	36613	0	0	
55	36614	0	0	
56	36615	0	0	
57	36616	0	0	
60	36617	0	0	
61	36620	0	0	
62	36621	0	0	
63	36622	0	0	
64	36623	0	0	
65	36624	0	0	
66	36625	0	0	
67	36626	0	0	
70	36627	35703	15299	
71	36630	35722	15314	
72	36631	35741	15329	
73	36632	35760	15344	
74	36633	35777	15359	
75	36634	36016	15374	<■
76	36635	36035	15389	<■
77	36636	36054	15404	<,

TRACK ASSIGNMENT TABLE (TAT):

1 WORD PER TRACK ON LU 2 AND LU 3:

CONTENTS:

- ID SEGMENT ADDRESS OF PROGRAM-OWNER
- 077777 FOR GLOBAL ASSIGNMENT
- 077776 FOR FMP ASSIGNMENT
- 100000 FOR SYSTEM ASSIGNMENT
- 0 FOR AVAILABLE

SPECIFICATION:

LOCATION	BASE PAGE 1656	TAT
# ENTRIES	BASE PAGE 1755	TATLG
# TRACKS ON LU2	BASE PAGE 1756	TATSD
# SECTORS/TRACK, LU2	BASE PAGE 1757	SECT2
# SECTORS/TRACK, LU3	BASE PAGE 1760	SECT3

CLASS TABLE:

THE CLASS TABLE ENTRY CAN BE IN ONE OF FOUR DIFFERENT STATES:

```
 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
! 0   0  0  0   0  0  0   0  0  0   0  0  0   0  0  0!
------------------------------------------------------
STATE 1:  CLASS DEALLOCATED, AVAILABLE

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01  00
-------------------------------------------------------
! 0 !    A D D R E S S   O F   F I R S T   E N T R Y  !
-------------------------------------------------------
STATE 2:  POINTER TO FIRST ENTRY IN CLASS QUEUE

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
! 1   0  X! SECURITY CODE  ! NUMBER OF PENDING REQS. !
------------------------------------------------------
STATE 3:  CLASS ALLOCATED, NO ONE WAITING ON CLASS
          NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255

 15  14 13 12  11 10 09  08 07 06  05 04 03  02 01 00
------------------------------------------------------
! 1   1  X! SECURITY CODE  ! NUMBER OF PENDING REQS. !
------------------------------------------------------
STATE 4:  CLASS ALLOCATED, SOMEONE WAITING (SUSPENDED)
          NUMBER OF PENDING REQUESTS COUNTER MAY BE 0-255
```

CLASS QUEUE FORMAT:

```
WORD         CONTENTS
----         --------
 1    < LINKAGE WORD           >
 2    <T, CONTROL INFO, CODE   >
 3    <PRIORITY OF REQUESTOR   > (CHANGED TO STATUS AT COMP.)
 4    <TOTAL BLOCK LENGTH WORDS>
 5    <CLASS ID WORD           >
 6    <USER BUFFER LENGTH      > (CHANGED TO TLOG AT COMP.)
 7    < OPTIONAL PARAMETER 1   >
 8    < OPTIONAL PARAMETER 2   >
 9    <WORD 1 OF USER BUFFER   >
 .        .        .       .
N+8   <WORD N OF USER BUFFER   >
```

THE <T> FIELD (BITS 15-14 IN CONTROL WORD)
IDENTIFIES THE REQUEST TYPE AS:

```
00   USER (NORMAL OPERATION)
01   USER (AUTOMATIC BUFFERING)
10   SYSTEM
11   CLASS I/O
```

SPECIFICATION: $CLAS = # ENTRIES IN CLASS TABLE
                  HEADS CLASS TABLE

L.U. SWITCH TABLE:

THE L.U. SWITCH TABLE HAS ONE WORD PER ENTRY:

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
!  NEW L.U. - 1          !   !   !   !   !    ORIGINAL LU - 1   !
-----------------------------------------------------------------
```

SPECIFICATION:

```
$LUSW         HEADS THE TABLE
              = # ENTRIES IN THE TABLE
```

RESOURCE NUMBER TABLE:

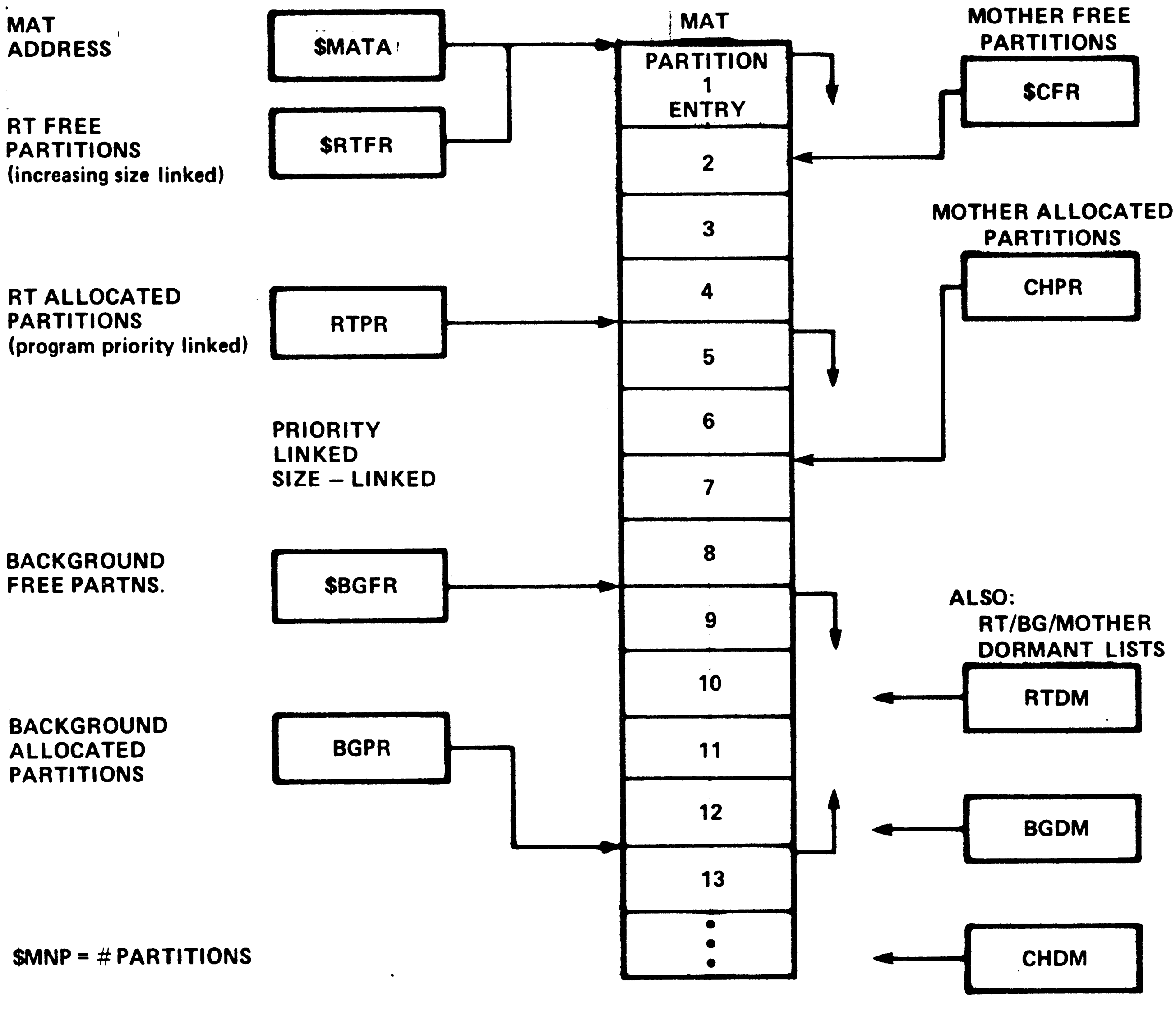
RESOURCE NUMBER TABLE HAS ONE WORD PER ENTRY:

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
-----------------------------------------------------------------
! OWNER'S ID NUMBER            ! LOCKER'S ID NUMBER            !
          OR                                OR
  377B  FOR GLOBAL ALLOCATE            377B FOR GLOBAL LOCK
          OR                                OR
   0    FOR AVAILABLE                    0  FOR UNLOCKED
```

SPECIFICATION:

```
$RNTB   HEADS THE TABLE AND
        = NUMBER OF ENTRIES
```

MEMORY ALLOCATION TABLE (MAT)



(NOTE: listheads are kept in Table Area 2 and DISPM)

- last entry points to start of allocated list
- priority linked

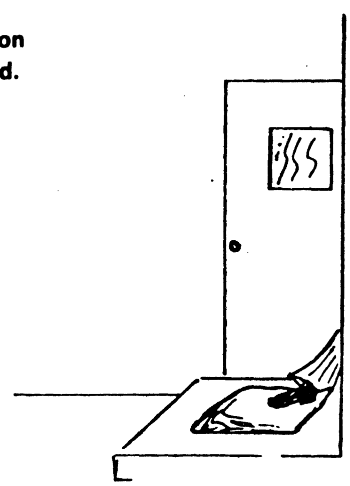
MAT ENTRY

WORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Linkage pointer to next entry (-1 if undefined partition)															
1	Priority of current resident program															
2	Current resident's ID-segment address															
3	M		D				Beginning page of partition									
4	R	C					Number of pages in partition (-1)									
5	RT													S		
6	Subpartition Link Word (SLW)															

D = Resident is dormant — save resources, serially reusable, or operator suspended
M = Mother partition
R = Partition is reserved
C = Partition is part of a chained mother partion
RT = REAL TIME PARTITION
S = Program's dispatching status
- 0 - Read in progress
- 1 - Program is resident
- 2 - SWAP out or segment load in progress
- 3 - SWAP out complete but program still resident
- 4 - Subpartition swap-out started for mother partition
- 5 - Subpartition swap-out completed. Mother cleared.

"NORMAL" sequence: 2,3,0,1

SLW
- = 0 - Partition not a subpartition
- = Next subpartition
- = Mother partition if partition is last subpartition



MEMORY PROTECT FENCE TABLE (MPFT)

$MPFT →

FENCE ADDRESS FOR:

DISC RESIDENT, NO COMMON (LARGE BG-TYPE 4)	Ø
MEMORY RESIDENT, NO COMMON	1
ANY PROGRAM USING RT COMMON	2
ANY PROGRAM USING BG COMMON	3
ANY PROGRAM USING SSGA	4
DISC RESIDENT, NO COMMON (RT/BG-TYPE 2 OR 3)	5

- EACH PROGRAM HAS A MPFT INDEX (MPFI) IN ID SEG. WORD 22
- CURRENT MEMORY PROTECT FENCE IS IN BP WORD FENCE (1775).

WORD 1 OF DMT ENTRY FOR EQT ENTRY:

	15	14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
1	SD	(RESERVED)	M
2	SD	(RESERVED)	M
3	SD	(RESERVED)	M
•			
•			
•			
N	SD	(RESERVED)	M

WORD 2 OF DMT ENTRY FOR EQT ENTRY:

	15	14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
1	MR	(RESERVED)	P
2	MR	(RESERVED)	P
•			
•			
•			
N	MR	(RESERVED)	P

WHERE:

SD = 0 IMPLIES DRIVER RESIDES IN A DRIVER PARTITION, AND
M = STARTING PAGE NUMBER OF PARTITION IN BITS 0-9

SD = 1 IMPLIES DRIVER RESIDES IN SYSTEM DRIVER AREA, AND
M = 0 IMPLIES DRIVER NOT DOING ITS OWN MAPPING
M = 1 IMPLIES DRIVER DOING ITS OWN MAPPING

MR = 1 IMPLIES THAT THE I/O REQUEST BUFFER IS LOCATED IN
A MEMORY RESIDENT PROGRAM.
(P VALUE NOT SIGNIFICANT – RESERVED FOR FUTURE USE)

MR = 0 IMPLIES THAT THE I/O REQUEST BUFFER IS NOT LOCATED
IN A MEMORY RESIDENT PROGRAM. BUFFER LOCATION IS
INDICATED BY THE VALUE OF P, AS FOLLOWS:
P = 0 IMPLIES BUFFER IS IN THE SYSTEM AREA
P NOT ZERO IMPLIES BUFFER IS LOCATED IN A DISC
RESIDENT PROGRAM. P IS THE PHYSICAL
PAGE NUMBER OF THE PROGRAM'S BASE PAGE

N = NUMBER OF EQT ENTRIES IN SYSTEM (BP 1651)
$DVMP = address of driver mapping table
$DVPT = logical start page of driver partition
$DLTH = # pages per driver partition

DRIVER MAPPING TABLE EXAMPLE

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	2714	2	2	
2	2715	30	24	
3	2716	32	26	
4	2717	36	30	
5	2720	30	24	
6	2721	2	2	
7	2722	30	24	
8	2723	2	2	
9	2724	34	28	
10	2725	34	28	
11	2726	34	28	
12	2727	2	2	
13	2730	34	28	
14	2731	30	24	
15	2732	100001	-32767	
16	2733	100001	-32767	
17	2734	100001	-32767	
18	2735	100001	-32767	
19	2736	100001	-32767	
20	2737	64	52	4
21	2740	0	0	
22	2741	0	0	
23	2742	0	0	
24	2743	0	0	
25	2744	0	0	
26	2745	0	0	
27	2746	0	0	
28	2747	0	0	
29	2750	0	0	
30	2751	0	0	
31	2752	0	0	
32	2753	0	0	
33	2754	0	0	
34	2755	0	0	
35	2756	0	0	
36	2757	0	0	
37	2760	0	0	
38	2761	0	0	

TRACK MAP TABLES

7900 DISC

		WORD
$TB31	NEG. # OF 64 WRD SECT./TRK	0
	FIRST TRACK, SUBCHANNEL 0	1
	FIRST TRACK, SUBCHANNEL 1	2
	FIRST TRACK, SUBCHANNEL 2	3
	FIRST TRACK, SUBCHANNEL 3	4
	FIRST TRACK, SUBCHANNEL 4	5
	FIRST TRACK, SUBCHANNEL 5	6
	FIRST TRACK, SUBCHANNEL 6	7
	FIRST TRACK, SUBCHANNEL 7	8
	# OF TRACKS, SUBCHANNEL 0	9
	# OF TRACKS, SUBCHANNEL 1	10
	# OF TRACKS, SUBCHANNEL 2	11
	# OF TRACKS, SUBCHANNEL 3	12
	# OF TRACKS, SUBCHANNEL 4	13
	# OF TRACKS, SUBCHANNEL 5	14
	# OF TRACKS, SUBCHANNEL 6	15
	# OF TRACKS, SUBCHANNEL 7	16

7905/7906/7920 DISC

		WORD
$TB32	NEG. # OF TOTAL SUBCHANNELS	1
	STARTING CYLINDER #	2
SUBCHANNEL 0	# SURFACES, HEAD, UNIT	3
	# TRACKS	4
⋮	⋮	⋮
	STARTING CYLINDER #	3n-1
SUBCHANNEL n	# SURFACES, HEAD, UNIT	3n
	# TRACKS	3n+1

WORD	LOCATION	VALUE(8)	VALUE(10)	VALUE(AS)
1	2000	177772	- 6	
2	2001	0	0	
3	2002	11000	4608	
4	2003	400	256	
5	2004	403	259	
6	2005	11000	4608	
7	2006	226	150	
8	2007	0	0	
9	2010	20000	8192	
10	2011	313	203	
11	2012	147	103	
12	2013	20000	8192	
13	2014	313	203	
14	2015	316	206	
15	2016	20000	8192	
16	2017	313	203	
17	2020	465	309	5
18	2021	20000	8192	
19	2022	313	203	

APPENDIX H

SE LEVEL II HOMEWORK

DAY (DUE)	ASSIGNMENT
Mon (Wed am)	1. Included in your material is a "User Program State Diagram". Indicate on the diagram one specific reason for a user program to make <u>each</u> legal state change. Types of reasons include: Operator Commands Exec Calls Environment Events
Mon (Fri)	2. Included in your material is a generation map listing. Indicate on the listing where each generator response is located in the final RTE system (i.e. which table, list, BP location, or global in operating system).
Tues (Thur)	3. Included in the course material is a RTE memory dump and two blank WHZAT printouts. Using the dump, fill in both WHZAT printouts. The partition list can be completed after Tuesday's lecture and the program state list after Wednesday (see Section 14 of the workbook for a sample WHZAT listing).
Wed (Fri)	4. Trace a "SS, PROGA" request from interrupt thru completion. See the "ON,XYZ" trace in Section 6 for an example. How does RTE implement an EXEC 7 request differently than a "SS" command?

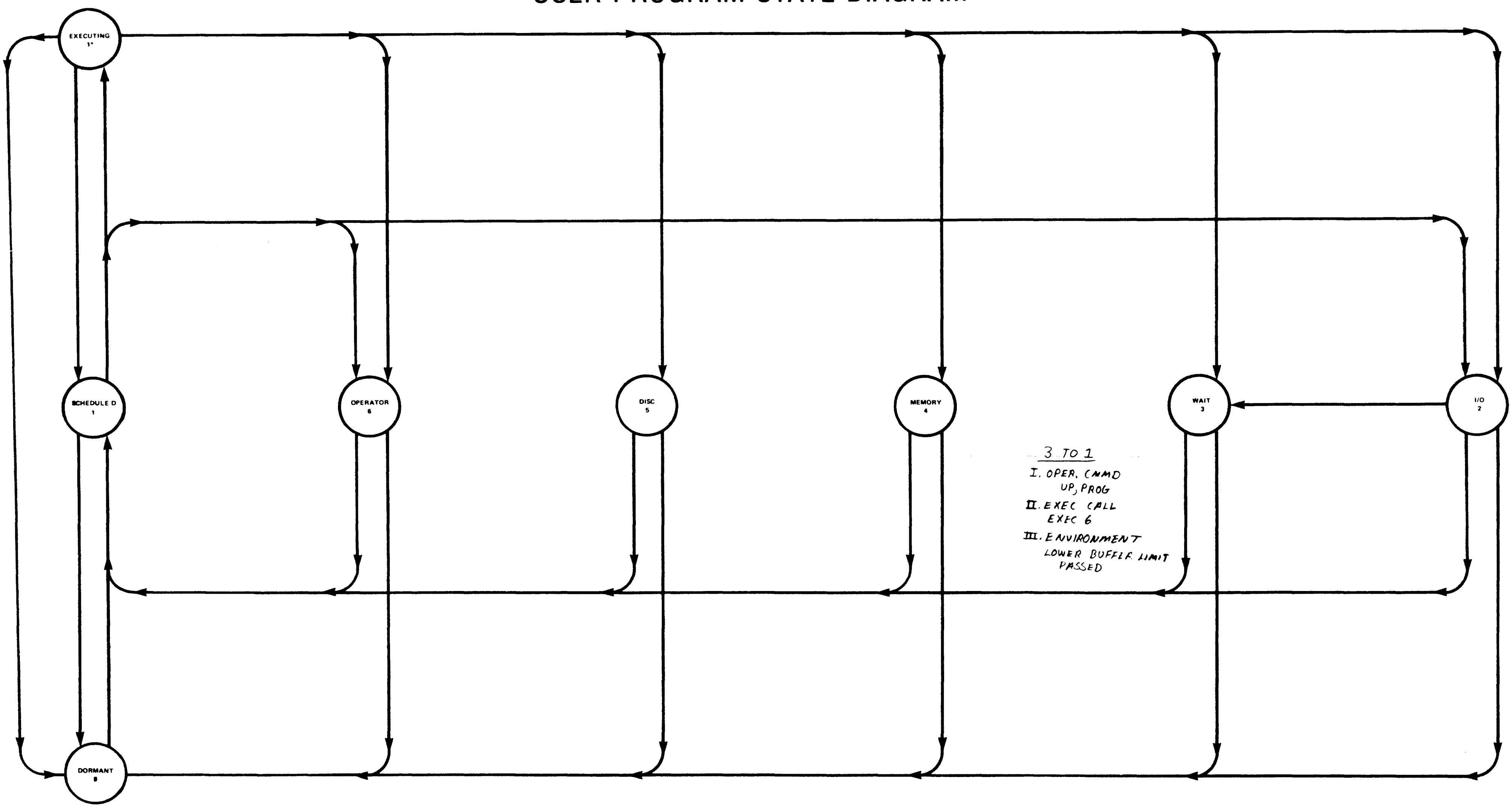
SE LEVEL II LABS
(5 REQUIRED)

DAY	ASSIGNMENT
Mon. (required)	1. Use CMM4 to find the following information about the training systems: a. TBG select code and system console EQT entry address. b. Starting address and number of EQT entries. c. Location of and number of class numbers in the class table ($CLAS). d. Number of programs in the schedule and general wait lists. e. Program name in ID segment #5. f. List the MR map. g. Memory address of $CIC, $LIST, and $XEQ. h. Value in trap cells 11 thru 15 (careful). i. EQT and subchannel of LU 26. j. ID segment address of currently executing program. k. Size of BG and RT common. l. Memory address of WHZAT and DVR32. m. Number of tracks on LU2.
Mon.	2. a. Use CMM4 to determine the number of pages in each section of physical memory thru the SAM extension. (See Section 2 for a copy of physical memory.) b. Program LABL2 sorts and calculates the average of 20 numbers. Compile it, load it and use only DBUGR to: 1) Correct the cause of the DMS error. 2) Correct the spelling of the second title to "SORTED ARRAY". 3) Set a breakpoint in the sorting loop in the subroutine SORT. Use variations of the "n/P", "T", and "n/T" commands in the load and examine the array "NUMB" as it is sorted. 4) Initialize "ISUM" to 0 instead of 999 by setting a breakpoint before it is initialized and modifing the A or B register.

Day Material Presented		ASSIGNMENT
Mon.	3.	a. Without a generation map, find all the RP microcode values generated into the training systems. (Use the utilities.) b. Find which loader ROM's (and their locations) are installed in the training systems. Use the front panel and "Loader ROM's" installation manual only.
Mon.	4.	Write a program to list all four DMS maps and the DMS status and violation registers. You will need to make your routine privileged. See page 18-16 of work book.
Tues.	5.	Write a program to search the MAT table and report the number of memory pages in the partition area.
Tues.	*6.	Write a program which will accept any EQT# and return the number of I/O requests queued on the EQT, the type of each request (user, buffered, class, or system), and the size of each request.
Wed.	*7.	Patch $CIC to keep a trace of the last 25 interrupt select codes (Hint" use a circular buffer to record the SC's.)
Wed.	8.	Patch PRMPT (on the disc) to change its prompt character from ">" to "?". Re-boot the system and verify the change. Restore PRMPT and boot the system when you finish.
Thur.	9.	Write a program to list the number and size of each block in the free SAM list.
Thur.	10.	Run program RENT. At each pause record the structure of the re-entrant list and TDB's linked to it.
Thur.	11.	Write a reentrant subroutine to return total, average, min., max., and medium of a 10 element array.

*Labs marked with an * are equivalent to 2 labs.

USER PROGRAM STATE DIAGRAM



DEMOLF T=00003 IS ON CR00002 USING 00153 BLKS R=0000

```
ECHO?
YES                                      * ECHO?

EST. # TRACKS IN OUTPUT FILE?
*****************************************************************************
*
*           DEMO      SYSTEM GENERATION
*
*                     3-21-78                                                 *
*
*****************************************************************************
40                                       * EST. # OF TRACKS IN OUTPUT FILE?

OUTPUT FILE NAME?
DEMOSY::14,                             * OUTPUT FILE NAME?

SYSTEM DISC?
7905                                     * SYSTEM DISC?

CONTROLLER SELECT CODE?
12                                       * CONTROLLER SELECT CODE?

# TRKS, FIRST CYL #, HEAD #, # SURFACES, UNIT, # SPARES FOR SUBCHNL:
  00?
* # TRKS, FIRST TRK ON SUBCHNL:
256,0,2,1,0,3                            * SUBCHANNEL 0
  01?
150,259,2,1,0,2                          * SUBCHANNEL 1
  02?
203,0,0,2,0,3                            * SUBCHANNEL 2
  03?
203,103,0,2,0,3                          * SUBCHANNEL 3
  04?
203,206,0,2,0,3                          * SUBCHANNEL 4
  05?
203,309,0,2,0,1                          * SUBCHANNEL 5
  06?
/E

SYSTEM SUBCHNL?
0                                        * SYSTEM SUBCHNL?

AUX DISC (YES OR NO OR # TRKS)?
NO                                       * AUX. DISC

TBG SELECT CODE?
11                                       * TBF SELECT CODE?

PRIV. INT. SELECT CODE?
0                                        * PRIV. INT. SELECT CODE?

MEM. RES. ACCESS TABLE AREA II?
YES                                      * MEM. RES. ACCESS TABLE AREA II?

RT MEMORY LOCK?
YES                                      * RT MEMORY LOCK?

BG MEMORY LOCK?
```

```
YES                                    * BG MEMORY LOCK?
SWAP DELAY?
50                                     * SWAP DELAY?
MEM SIZE?
128                                    * MEM SIZE?
BOOT FILE NAME?
0                                      * BOOT FILE NAME?
PROG INPUT PHASE:
-
****************************
-
*** PROGRAM INPUT PHASE ***
-
****************************
-
MAP MODULES
-
LINKS IN CURRENT
-
******************************
-
** STANDARD RTE-IV MODULES **
-
******************************
-
REL,%CR4SY::1904,                      * RTE-IV OPERATING SYSTEM
-
*************
-
** DRIVERS **
-
*************
-
REL,%DVR32::1904,                      * 7905 DISC
-
REL,%DVR12::1904,                      * LINE PRINTER(2767)
-
REL,%4DV05::1904,                      * 2645 CRT
-
REL,%DVR36::1904,                      * WCS DRIVER
-
REL,%DVR23::1904,                      * 7970 MAG. TAPE
-
REL,%DVA13::1904,                      * TV MONITOR
-
REL,%DVR07::1904,                      * MULTIPOINT
-
REL,%2DV37::1904,                      * HPIB
-
REL,%4DP43::1904,                      * POWER FAIL
-
REL,%DVA12::1904,                      * LP (2607)
-
*****************************
-
```

```
** SPECIAL SYSTEM MODULES **
-
******************************
-
REL,%AUTO7::1904,                  * DVR07 POWER FAIL ROUTINE
-
*******************
-
** USER PROGRAMS **
-
*******************
-
REL,%HPIB::1904,                   * HPIB DEVICE SUBROUTINE
-
REL,%4LDR::1904,                   * RELOCATING LOADER
-
REL,%GASP4::1904,                  * GASP
-
REL,%4SPOL::1904,                  * SPOOLING
-
REL,%BMPG::1904,                   * FMGR MODIFIED FOR RTE-IV
-
REL,%4MTM::1904,                   * MULTI-TERMINAL MONITOR
-
***************
-
** LIBRARIES **
-
***************
-
REL,%4SYLB::1904,                  * RTE-IV SYSTEM LIBRARY
-
REL,%BMLIB::1904,                  * RTE FILE MANAGEMENT LIBRARY
-
REL,%CLIB ::1904,                  * RTE COMPILER LIBRARY
-
REL,%FF4.N::1904,                  *
-
REL,%RLIB1::1904,                  *
-
REL,%RLIB2::1904,                  *
-
REL,%IMAG1::1904,                  * IMAGE LIBRARY
-
REL,%IMAG2::1904,
-
REL,%DBUGR::1904,                  * USER DEBUG
-
REL,%MPLIB::1904,                  * MULTIPOINT LIBRARY
-
REL,%BASLB::1904,                  * BASIC
-
REL,%MLIB::1904,,                  * BASIC NEW MESSAGE FILE
-
REL,%TRAP::1904,                   * SR0.T IN SSGA FOR BASIC
-
***************
-
** UTILITIES **
-
```

```
****************
-
REL,%4WHZT::1904,                    * WHZAT PROGRAM
-
REL,%CMM4::1904,                     * CMM4 PROGRAM
-
REL,%LUMAP,                          * INIT. LU'S FOR DEMO
-
REL,%KYDMP::1904,                    * KEY DUMP UTIL.
-
REL,%UTIL::1904,                     * SET UP SSGA FOR TVST+TVMEM
-
REL,%DSPMP::1904,                    * MP UTIL.
-
REL,%EXMP::1904,                     * MP UTIL.
-
REL,%LGTAT::1904,                    * TRACK ASSIGNMENT LISTING
-
DISPLAY UNDEFS
 UNDEFS
&6940
-
/E
 UNDEFS
&6940

PARAMETERS
-
*******************************
*** PARAMETER INPUT PHASE ***
*******************************
D.RTR,1,1
-
CMM4,1,90
-
WHZAT,1,1
-
AUTOR,19
-
PRMPT,1,10
-
RSPN%,1,10
-
LOADR,3,80
-
EXTND,17
-
SMP,18
-
SPOUT,18
-
JOB,18
-
GASP,19
-
.DBRN,30                             * SSGA FOR IMAGE
-
TRAP,30                              * SSGA FOR BASIC
-
TTYEV,17                             * FOR SSGA ACCESS
```

```
-
IDGET,17                                          *       "
-
/E

CHANGE ENTS?
-
*******************
** CHANGE ENTS? **
*******************
.MPY,RP,100200                                    * EAU MACRO'S
-
.DIV,RP,100400
-
.DLD,RP,104200
-
.DST,RP,104400
-
.FAD,RP,105000                                    * HFP MACRO'S
-
.FSB,RP,105020
-
.FMP,RP,105040
-
.FDV,RP,105060
-
IFIX,RP,105100
-
FLOAT,RP,105120
-
DBLE,RP,105201                                    * FFP MACRO'S
-
SNGL,RP,105202
-
.XMPY,RP,105203
-
.XDIV,RP,105204
-
.DFER,RP,105205
-
.XADD,RP,105213
-
.XSUB,RP,105214
-
.GOTO,RP,105221
-
..MAP,RP,105222
-
.ENTR,RP,105223
-
.ENTP,RP,105224
-
.PWR2,RP,105225
-
.FLUN,RP,105226
-
.SETP,RP,105227
-
.PACK,RP,105230
-
```

```
.XFER,RP,105220
-
.XPAK,RP,105206
-
XADD,RP,105207
-
XSUB,RP,105210
-
XMPY,RP,105211
-
XDIV,RP,105212
-
.XCOM,RP,105215
-
..DCM,RP,105216
-
DDINT,RP,105217
-
MVW.,RP,105777                      * 21MX EXTENDED INSTRUCTION SET
-
.MVW,RP,105777
-
.EMAP,RP,105257                     * 21XE EMA
-
.EMIO,RP,105240
-
MMAP,RP,105241
-
/E



TABLE AREA I


EQUIPMENT TABLE ENTRY

EQT 01?
******************************
*** TABLE GENERATION PHASE***
******************************
* TABLE AREA I               *
******************************
* EQUIPMENT TABLE ENTRY      *
******************************
12,DVR32,D                          * EQT 01 = 7905 DISC

EQT 02?
21,DVR05,T=32767,X=13             * EQT 02 = SYSTEM CONSOLE (2648)

EQT 03?
10,DVR36,D                          * EQT 03 = WCS

EQT 04?
20,DVR37,T=6000,X=25              * EQT 04 = HPIB

EQT 05?
23,DVR05,T=32767,X=13               * EQT 05 = AUX. 2645 OR 2648

EQT 06?
```

```
24,DVR12,B,T=32767                        * EQT 06 = LINE PRINTER(2767)

EQT 07?
25,DVA12,B,T=32767                        * EQT 07 = LP(2607)

EQT 08?
16,DVR23,D,T=32767                        * EQT 08 = MAG. TAPE

EQT 09?
22,DVR07,X=5                              * EQT 09 = MP LINE CONTROL

EQT 10?
77,DVR07,X=5                              * EQT 10 = MP TERMINAL #1

EQT 11?
77,DVR07,X=5                              * EQT 11 = MP TERMINAL #2

EQT 12?
15,DVA13,D,T=20                           * EQT 12 = TV

EQT 13?
77,DVR07,X=5                              * EQT 13 = MP TERMINAL #3

EQT 14?
26,DVR05,B,T=32767,X=13                   * EQT 14 = AUX. TERMINAL (2645 OR 2648)

EQT 15?
60,DVS43,S,M,X=18                           * EQT 15 = SPOOL

EQT 16?
61,DVS43,S,M,X=18                           * EQT 16 = SPOOL

EQT 17?
62,DVS43,S,M,X=18                           * EQT 17 = SPOOL

EQT 18?
63,DVS43,S,M,X=18                           * EQT 18 = SPOOL

EQT 19?
4,DVP43,M                                 * EQT 19 = POWER FAIL

EQT 20?
/E


DEVICE REFERENCE TABLE

 1 = EQT #?
*************************
* DEVICE REFERENCE TABLE *
*************************
2,0                                       * LU  1 = SYSTEM CONSOLE (2648)

 2 = EQT #?
1,0                                       * LU  2 = SYSTEM DISC (SUB. 0)

 3 = EQT #?
0                                         * LU  3 = BIT BUCKET

 4 = EQT #?
```

```
2,1                    * LU  4 = SYS CONSOLE LEFT CTU
 5 = EQT #?
2,2                    * LU  5 = RT. CTU
 6 = EQT #?
2,0                    * LU  6 = SYS CONSOLE(2648)
 7 = EQT #?
2,3                    * LU  7 = 2648 GRAPHICS
 8 = EQT #?
8,0                    * LU  8 = MAG. TAPE
 9 = EQT #?
9,0                    * LU  9 = MP LINE CONTROL
10 = EQT #?
10,0                   * LU 10 = MP TERM. #1
11 = EQT #?
11,0                   * LU 11 = MP TERM. #2
12 = EQT #?
12,1                   * LU 12 = TV MONITOR
13 = EQT #?
13,0                   * LU 13 = MP TERM. #3
14 = EQT #?
1,1                    * LU 14 = AUX. DISC (SUB. 1)
15 = EQT #?
0                      * LU 15 = BIT BUCKET
16 = EQT #?
5                      * LU 16 = AUX. TERM.
17 = EQT #?
5                      * LU 17 = LT. CTU
18 = EQT #?
5                      * LU 18 = RT. CTU
19 = EQT #?
5                      * LU 19 = GRAPHICS
20 = EQT #?
0                      * LU 20 = BIT BUCKET
21 = EQT #?
1,2                    * LU 21 = AUX. DISC
22 = EQT #?
4,0                    * LU 22 = HP IB
23 = EQT #?
4,1                    * LU 23 = HP IB DEVICE #1
24 = EQT #?
```

```
4,2                                   * LU 24 = HP IB DEVICE #2

25 = EQT #?
4,3                                   * LU 25 = HP IB DEVICE #3

26 = EQT #?
1,3                                   * LU 26 = AUX. DISC

27 = EQT #?
14,0                                  * LU 27 = EXTRA 2648

28 = EQT #?
14,1                                  * LU 28 = LT. CTU

29 = EQT #?
14,2                                  * LU 29 = RT. CTU

30 = EQT #?
14,3                                  * LU 30 = CRAPHICS

31 = EQT #?
1,4                                   * LU 31 = AUX DISC

32 = EQT #?
3,0                                   * LU 32 = 1K WCS,LOWER

33 = EQT #?
3,1                                   * LU 33 = 1K WCS,UPPER

34 = EQT #?
15,0                                  * LU 34 = SPOOL

35 = EQT #?
16,0                                  * LU 35 = SPOOL

36 = EQT #?
17,0                                  * LU 36 = SPOOL

37 = EQT #?
18,0                                  * LU 37 = SPOOL

38 = EQT #?
6,0                                   * LU 38 = LP(2767)

39 = EQT #?
7,0                                   * LU 39 = LP(2607)

40 = EQT #?
1,5                                   * LU 40 = AUX DISC

41 = EQT #?
19                                    * LU 41 = POWER FAIL

42 = EQT #?
/E


 INTERRUPT TABLE

 -
```

```
********************
* INTERRUPT TABLE *
********************
4,ENT,$POWR                              * POWER FAIL/AUTO RESTART
-
12,EQT,1                                 * 7905 DISC
-
13,EQT,12                                * TV
-
14,EQT,12                                * TV
-
15,EQT,12                                * TV
-
16,EQT,8                                 * MAG. TAPE
-
17,EQT,8                                 * MAG. TAPE
-
20,EQT,4                                 * HP IB
-
21,PRG,PRMPT                             * SYSTEM CONSOLE (2648)
-
22,PRG,PRMPT                             * MP LINE CONTROL
-
23,PRG,PRMPT                             * AUX. TERM. 2648
-
24,EQT,6                                 * LP 2767
-
25,EQT,7                                 * LP 2607
-
26,PRG,PRMPT                             * AUX. TERMINAL 2648
-
77,ABS,0                                 * MULTIPOINT TERMINALS
-
/E


TABLE AREA I MODULES

$$TB1(0099)03176 03320  92067-16014 REV.1805 780223


DRIVR PART 00002
 CHANGE DRIVR PART?
*********************************
*** SYSTEM BOUNDARIES PHASE ***
*********************************
0                                        * CHANGE DRIVER PARTITION?


DP  01!

DVR32(0099)04000 05525  92060-16031 REV 1805 780126

DVR12(0099)05553 06311   29028-60002  780103  REV 1805

DVR23(0099)06322 07165      92202-16001  REV. A

DVA13(0099)07166 07421  91200-16031 REV 1648 -- 761124
```

H 14

```
SUBSYSTEM GLOBAL AREA

SP.CL      )10000 10002  92067-16028 REV.1805 780317

.DBRN      )10003 10033  92063-12001 REV. 1805 770601

TRAP       )10034 11072   92101-16010 770208

UTIL       )11073 11106



RT COMMON  00000
 CHANGE RT COMMON ?
100                               * CHANGE RT COMMON?
RT COM ADD     11107


BG COMMON  00341
 CHANGE BG COMMON ?
0                                 * CHANGE BG COMMON?
BG COM ADD     11253
BG COMMON  00341


SYSTEM DRIVER AREA

DVP43(0099)12000 12634  92067-16004 REV.1805 771219

DVS43(0099)12653 15600  92067-16028 REV.1805 771110



TABLE AREA II

# OF I/O CLASSES?
32                                * # OF I/O CLASSES?

# OF LU MAPPINGS?
24                                * # OF LU MAPPINGS?

# OF RESOURCE NUMBERS?
32                                * # OF RESOURCE NUMBERS?

BUFFER LIMITS (LOW, HIGH)?
100,400                           * BUFFER LIMITS (LOW,HIGH)?

# OF BLANK ID SEGMENTS?
50                                * # OF BLANK ID SEGMENTS?

# OF BLANK SHORT ID SEGMENTS?
35                                * # OF BLANK SHORT ID SEGMENTS?

# OF BLANK ID EXTENSIONS?
15                                * # OF BLANK ID EXTENSIONS?

MAXIMUM # OF PARTITIONS?
15                                * MAXIMUM # OF PARTITIONS?
```

```
TABLE AREA II MODULES

$$TB2(0099)24317 24363  92067-16014 REV.1805 771107



SYSTEM

$CSY4(0099)24364 24363  92067-16014 REV.1805 780125

DISP4(0099)24444 32026  92067-16014 REV.1805 780317

RTIME(0099)32035 32641  92067-16014 REV.1805 780104

$ASC4(0099)32642 32734  92067-16014 REV.1805 780125

RTIO4(0099)33016 40336  92067-16014 REV.1805 780310

EXEC4(0099)40356 42734  92067-16014 REV.1805 780310

$TRN4(0099)42760 43132  92067-16014 REV.1805 780104

SCHD4(0099)43150 50327  92067-16014 REV.1805 780317

$ALC (0099)50337 50544  92067-16014 REV.1805 741120

OCMD4(0099)50545 51706  92067-16014 REV.1805 771102

PERR4(0099)51716 52656  92067-16014 REV.1805 780227

$BMON(0099)52657 52656  92002-12001 REV.1805 771116

$YSLB(0099)52657 52656  92067-16035 REV.1805 770714

$BALB(0099)52657 52656  92002-16006 REV.1805 771116

FF4.A(0099)52657 52656     24998-16002 REV.1805 780303

RLIB1(0099)52657 52656   24998-16001  REV.1805  771116

RLIB2(0099)52657 52656   24998-16001  REV.1805  771116

MPLIB(0099)52657 52656  91730-12001 REV 1805 780301

$CNFG(0099)52710 57546  92067-16014 REV.1805 770112


PARTITION DRIVERS

DP  02:

DVR05(0099)04056 06670      92001-16027  REV.1806  1-17-78

DVA12(0099)06763 07663   92001-16020  1806   780112

DP  03:

DVR36(0099)04014 06075  RTE DVR36 13197-16001 REV.A 751221

```

```
DP  04:

DVR07(0099)04070 06431  91730-16001 REV 1805 780307 &DV7D2

DP  05:

DVR37(0099)04065 06411   59310-16003 REV. 1805, 780306


MEMORY RESIDENT LIBRARY

  PRTN      26000 26112  92067-16035 REV.1805 771005
  TMVAL     26113 26202  92067-16035 REV.1805 770715
  IFBRK     26203 26232  92067-16035 REV.1805 770621
  PARSE     26233 26252  92067-16035 REV.1805 770714
  $PARS     26253 26473  92067-16035 REV.1805 770621
  CNUMD     26474 26513  92001-16035 REV.1805 770621
  CNUMO     26514 26533  92067-16035 REV.1805 770621
  $CVT3     26534 26621  92067-16035 REV.1805 770621
  IPUT      26622 26642  92002-16006   740801
  IABS      26643 26655   750701  24998-16001


MEMORY RESIDENTS

EXTND(0010)30002 30154  92067-16028 REV.1805 771115
  RMPAR     30155 30213  771116  24998-16001

D.RTR(0001)30246 32305  92002-16007 1805 780106
  P.PAS     32370 32416  92002-16006 740801

PRMPT(0010)32421 33036  92067-16003 REV.1805 780119
  TRMLU     33037 33132  92067-16035 REV.1805 771117
  IDGET     33133 33215  92067-16037 REV.1805 771227

RSPN$(0010)33220 33735  92067-16003 REV.1805 780119
  TRMLU     33742 34035  92067-16035 REV.1805 771117
  IDGET     34041 34123  92067-16037 REV.1805 771227

TTYEV(0002)34126 34135            29102-60013

WHZAT(0001)34154 36256  92067-16007 REV.1805 771219

CMM4 (0090)36310 47534
  REIO      47540 47644  92067-16035 REV.1805 780221
  CLRIO     47645 47653  750701  24998-16001
  IAND      47654 47663   750701  24998-16001
  IGET      47664 47672   750701  24998-16001
  IOR       47673 47702    750701  24998-16001
  RMPAR     47703 47741  771116  24998-16001
  IXGET     47742 50411
  DOIO      50414 51575
  DISC3     51611 52340
  DTRK      52346 52654



RT DISC RESIDENTS

SMP  (0030)26002 31360  92067-16028 REV.1805 771115
```

```
   RNRQ      31361 31622  92067-16035 REV.1805 780222
   $ALRN     31623 31740  92067-16035 REV.1805 770715
   PRTN      31744 32056  92067-16035 REV.1805 771005
   .DRCT     32063 32071  92067-16035 REV.1805 741120
   REIO      32072 32176  92067-16035 REV.1805 780221
   READF     32177 32735  92002-16006 770801
   POST      32736 32764  92002-16006  740801
   P.PAS     32765 33013  92002-16006 740801
   RW$UB     33014 33265  92002-16006 750422
   RWND$     33266 33410  92002-16006  771121
   R/W$      33411 33544  92002-16006  740801
   RMPAR     33545 33603  771116  24998-16001

 JOB   (0030)26002 27760  92067-16028 REV. 1805 760715
   RNRQ      27761 30222  92067-16035 REV.1805 780222
   LURQ      30224 30604  92067-16035 REV.1805 771013
   $ALRN     30605 30722  92067-16035 REV.1805 770715
   .DRCT     30723 30731  92067-16035 REV.1805 741120
   REIO      30732 31036  92067-16035 REV.1805 780221
   $PARS     31037 31257  92067-16035 REV.1805 770621
   OPEN      31260 31445  92002-16006 741205
   READF     31472 32230  92002-16006 770801
   CLOSE     32255 32373  92002-16006  771115
   POST      32374 32422  92002-16006  740801
   $OPEN     32423 32631  92002-16006   740801
   P.PAS     32632 32660  92002-16006 740801
   RW$UB     32661 33132  92002-16006 750422
   RWND$     33133 33255  92002-16006  771121
   R/W$      33256 33411  92002-16006  740801
   SPOPN     33412 33462  92002-16006  741025
   RMPAR     33463 33521  771116  24998-16001

 SPOUT(0011)26002 26755  92067-16028 REV.1805 780309
   LURQ      26756 27336  92067-16035 REV.1805 771013
   $ALRN     27337 27454  92067-16035 REV.1805 770715
   .DRCT     27455 27463  92067-16035 REV.1805 741120



 BG DISC RESIDENTS

 $CNFX(0099)26002 31460  92067-16006 REV.1805 780112
   $PARS     31461 31701  92067-16035 REV.1805 770621
   $CVT3     31702 31767  92067-16035 REV.1805 770621

 AUTOR(0001)26002 26361   91730-16009  REV.1805  780203
   FIXMP     26362 26444  91730-16008 REV 1805 771206

 LOADR(0080)26002 41350  92067-16002 REV.1805 780211
   LURQ      41351 41731  92067-16035 REV.1805 771013
   $ALRN     41736 42053  92067-16035 REV.1805 770715
   PRTN      42054 42166  92067-16035 REV.1805 771005
   REIO      42167 42273  92067-16035 REV.1805 780221
   IFBRK     42274 42323  92067-16035 REV.1805 770621
   $CVT3     42324 42411  92067-16035 REV.1805 770621
   LOGLU     42412 42461  92067-16035 REV.1805 771117
   CREAT     42462 42737  92002-16006  741022
   OPEN      42740 43125  92002-16006 741205
   READF     43126 43664  92002-16006 770801
   APOSN     43671 44032  92002-16006 750227
```

```
      LOCF        44041 44227   92002-16006 750416
      CLOSE       44230 44346   92002-16006  771115
      NAM..       44347 44443   92002-16006 740801
      $OPEN       44444 44652   92002-16006   740801
      P.PAS       44653 44701   92002-16006 740801
      RW$UB       44702 45153   92002-16006 750422
      RWND$       45154 45276   92002-16006  771121
      R/W$        45277 45432   92002-16006  740801
      NAMR        45433 45727    750701  24998-16001
      RMPAR       45730 45766   771116  24998-16001

    GASP (0080)26002 27415   92067-16028 REV.1805 780317
      G1CEX       27416 27527   92002-16001 760615
      ST.LU       27530 27705   92067-16028 780317
      G1ROT       27715 30070   92002-16001  760615
      G0QIP       30073 30360   92002-16001 760621
      RNRQ        30361 30622   92067-16035 REV.1805 780222
      $ALRN       30623 30740   92067-16035 REV.1805 770715
      .DRCT       30741 30747   92067-16035 REV.1805 741120
      REIO        30750 31054   92067-16035 REV.1805 780221
      KCVT        31055 31070   92001-16035 REV.1805 770621
      PARSE       31071 31110   92067-16035 REV.1805 770714
      $PARS       31111 31331   92067-16035 REV.1805 770621
      $CVT3       31332 31417   92067-16035 REV.1805 770621
      OPEN        31420 31605   92002-16006 741205
      READF       31623 32361   92002-16006 770801
      CLOSE       32407 32525   92002-16006  771115
      POST        32526 32554   92002-16006  740801
      $OPEN       32555 32763   92002-16006   740801
      P.PAS       32764 33012   92002-16006 740801
      RW$UB       33013 33264   92002-16006 750422
      RWND$       33265 33407   92002-16006  771121
      R/W$        33410 33543   92002-16006  740801
      RMPAR       33544 33602   771116  24998-16001

    GASP1(0099)33603 33615   92067-16028 REV.1805 760615
      G1COJ       33624 34210
      G1CCJ       34223 34637   92002-16001 760615
      G1CKS       34640 35454   92002-16001 760627
      G1CDS       35505 36532   92002-16001 760621
      G1STM       36536 36730   92002-16001  740807
      CNUMD       36731 36750   92001-16035 REV.1805 770621

    GASP2(0099)33603 33613   92067-16028 REV.1805 760615
      G1CSD       33624 34242   92002-16001 760622
      G1C??       34246 35056   92002-16001 741027
      G1CIN       35063 36425   92002-16001 760630
      G1CDA       36470 37044   92002-16001 760627
      CNUMD       37045 37064   92001-16035 REV.1805 770621
      CREAT       37065 37342   92002-16006  741022
      PURGE       37343 37441   92002-16006  740801
      NAM..       37442 37536   92002-16006 740801

    FMGR (0090)26002 26757   92002-16008 REV.1805 760627
      FM.CM       26760 30767   92002-16008 771208
      LURQ        31104 31464   92067-16035 REV.1805 771013
      $ALRN       31465 31602   92067-16035 REV.1805 770715
      .DRCT       31603 31611   92067-16035 REV.1805 741120
      IFBRK       31612 31641   92067-16035 REV.1805 770621
      IFTTY       31642 31715   92067-16035 REV.1805 771208
```

```
OPEN         31724 32111   92002-16006 741205
CLOSE        32124 32242   92002-16006  771115
$OPEN        32243 32451   92002-16006   740801
RWNDS        32452 32574   92002-16006  771121
R/WS         32575 32730   92002-16006  740801
RMPAR        32731 32767   771116  24998-16001

FMGR0(0099)32770 32775     92002-16008  740801
PK..         33006 34431
CR..         34527 35611   92002-16008 760616
COR.A        35612 35632   92067-16035 REV.1805 770621
READF        35647 36405   92002-16006 770801
REIO         36434 36540   92067-16035 REV.1805 780221
RWNDF        36541 36622   92002-16006  740801
NAM..        36623 36717   92002-16006 740801
P.PAS        36720 36746   92002-16006 740801
RWSUB        36747 37220   92002-16006 750422
LOCK.        37221 37270   92002-16006 771118
FM.UT        37271 40436   92002-16006 771118
CREA.        40502 40553
CREAT        40554 41031   92002-16006  741022

FMGR1(0099)32770 33116     92002-16008  760929
.PARS        33120 34403   92002-16008 765025
C.TAB        34472 34635   92002-16008 760720
CA..         34636 35057   92002-16008  760513
REA.C        35060 35132   92002-16008  770823
EE..         35133 35173   92002-16008  760512
TR..         35174 35425   92002-16008  760616
MR..         35426 35670   92002-16008 760621
SE..         35672 36056
IF..         36072 36307   92002-16008 760929
AB..         36310 36536   92002-16008 780221
DP..         36537 36604   92002-16008  760511
MESSS        36605 36744   92067-16035 REV.1805 771227
CNUMD        36745 36764   92001-16035 REV.1805 770621
$CVT3        36765 37052   92067-16035 REV.1805 770621
READF        37053 37611   92002-16006 770801
REIO         37612 37716   92067-16035 REV.1805 780221
POSNT        37720 40163   92002-16006 760702
P.PAS        40207 40235   92002-16006 740801
RWSUB        40236 40507   92002-16006 750422
WRLG.        40510 40657   92002-16006 760622
CK.SM        40660 41003   92002-16006 REV. 1805 771205

FMGR2(0099)32770 33000     92002-16008 760622
IN.IT        33004 34101   92002-16008 780106
IN..         34120 36030   92002-16008  771229
MC..         36043 36361   92002-16008  760511
RC..         36362 36547
PU..         36550 36772
PURGE        36773 37071   92002-16006  740801
NAM..        37072 37166   92002-16006 740801
J.PUT        37167 37213   92002-16006  740801
IPUT         37214 37234   92002-16006   740801
FID.         37235 37354
MSC.         37355 37411
LOCK.        37412 37461   92002-16006 771118
FM.UT        37462 40627   92002-16006 771118
.OPSY        40630 40632   771116  24998-16001
```

```
FMGR3(0099)32770 32775   92002-16008 760720
  DL..       33004 34301   92002-16008 771020
  F.SET      34361 34551   92002-16006 760719
  CS..       34552 35000   92002-16008 760318
  READF      35001 35537   92002-16006 770801
  REIO       35540 35644   92067-16035 REV.1805 780221
  LOCF       35654 36042   92002-16006 750416
  P.PAS      36060 36106   92002-16006 740801
  RWSUB      36107 36360   92002-16006 750422
  MSC.       36361 36415
  FM.UT      36416 37563   92002-16006 771118
  CK.ID      37564 37606   92002-16006 771205
  LULU.      37607 37677   92002-16006 760227

FMGR4(0099)32770 33001   92002-16008 760622
  ST.DU      33003 34254   92002-16008  760622
  CO..       34310 35012
  F.UTM      35013 35254   92002-16008 760514
  CREAT      35255 35532   92002-16006  741022
  READF      35554 36312   92002-16006 770801
  REIO       36333 36437   92067-16035 REV.1805 780221
  RWNDF      36440 36521   92002-16006  740801
  LOCF       36522 36710   92002-16006 750416
  NAM..      36711 37005   92002-16006 740801
  P.PAS      37006 37034   92002-16006 740801
  RWSUB      37035 37306   92002-16006 750422
  FM.UT      37307 40454   92002-16006 771118
  CREA.      40525 40576
  CK.SM      40577 40722   92002-16006 REV. 1805 771205

FMGR5(0099)32770 33002   92002-16008 760622
  RU..       33003 33641   92002-16008 761004
  RP..       33642 33751   92002-16008  761004
  SESSN      33754 34012   92002-16008 761005
  .RENM      34015 34141   92002-16008 761004
  .EXCP      34142 34206   92002-16008 761002
  IDDUP      34207 34551   92002-16008 770902
  IDRPL      34552 35277   92002-16008 780106
  IDRPD      35300 35517   92002-16008 771115
  OPMES      35520 35710   92002-16008 760513
  TL..       35711 35730
  MESSS      35735 36074   92067-16035 REV.1805 771227
  READF      36102 36640   92002-16006 770801
  REIO       36641 36745   92067-16035 REV.1805 780221
  NAM..      36746 37042   92002-16006 740801
  P.PAS      37043 37071   92002-16006 740801
  RWSUB      37072 37343   92002-16006 750422
  IO.A       37344 37433   92002-16006 780207
  CNT.       37434 37670   92002-16006 760520
  FCONT      37671 37766   92002-16006 751104
  BUMP.      37770 40026   92002-16006  741025
  SET.T      40033 40061   92002-16006 740801
  TL.        40062 40115   92002-16006 760322
  ST.TM      40116 40152   92002-16006 741223

FMGR6(0099)32770 33000   92002-16008    740801
  CN..       33001 33041
  JO..       33046 34050   92002-16008 760719
  EO..       34063 34666   92002-16008 770620
```

```
      OF..      34667 34762   92002-16008 740820
      LG..      34763 35010   92002-16008 760517
      RNRQ      35011 35252   92067-16035 REV.1805 780222
      KCVT      35253 35266   92001-16035 REV.1805 770621
      MESSS     35267 35426   92067-16035 REV.1805 771227
      $CVT3     35427 35514   92067-16035 REV.1805 770621
      NAMF      35515 35670   92002-16006  771115
      READF     35701 36437   92002-16006 770801
      REIO      36471 36575   92067-16035 REV.1805 780221
      POST      36576 36624   92002-16006  740801
      NAM..     36625 36721   92002-16006 740801
      P.PAS     36722 36750   92002-16006 740801
      RWSUB     36751 37222   92002-16006 750422
      SPOPN     37223 37273   92002-16006  741025
      SET.T     37274 37322   92002-16006 740801
      ST.TM     37323 37357   92002-16006 741223
      B.FLG     37360 37426   92002-16006 741118
      LULU.     37427 37517   92002-16006 760227
      RANGE     37520 37543   92002-16006 740801
      ONOFF     37566 40131   92002-16006 750128
      EX.TM     40133 40350   92002-16006 771115
      IPUT      40351 40371   92002-16006   740801
      LU.CL     40372 40440   92002-16006 760702
      AVAIL     40441 40533   92002-16006 741231

    FMGR7(0099)32770 32776   92002-16008 760702
      ??..      33000 35355   92002-16008 771111
      SY..      35356 35414   92002-16008 760520
      NX.JB     35417 36311   92002-16008 760702
      RNRQ      36364 36625   92067-16035 REV.1805 780222
      MESSS     36626 36765   92067-16035 REV.1805 771227
      READF     36766 37524   92002-16006 770801
      REIO      37525 37631   92067-16035 REV.1805 780221
      POST      37632 37660   92002-16006  740801
      P.PAS     37661 37707   92002-16006 740801
      RWSUB     37717 40170   92002-16006 750422
      SPOPN     40173 40243   92002-16006  741025
      B.FLG     40244 40312   92002-16006 741118
      LULU.     40313 40403   92002-16006 760227
      LU.CL     40404 40452   92002-16006 760702

    FMGR8(0099)32770 32776   92002-16008   740801
      SA..      32777 33751   92002-16008 760621
      SP..      33752 34721                            780221
      MS..      34753 35246
      PRTN      35247 35361   92067-16035 REV.1805 771005
      READF     35402 36140   92002-16006 770801
      REIO      36157 36263   92067-16035 REV.1805 780221
      RWNDF     36264 36345   92002-16006  740801
      LOCF      36346 36534   92002-16006 750416
      P.PAS     36535 36563   92002-16006 740801
      RWSUB     36564 37035   92002-16006 750422
      IPUT      37036 37056   92002-16006   740801
      CREA.     37057 37130
      CREAT     37131 37406   92002-16006  741022
      NAM..     37407 37503   92002-16006 740801
      CK.SM     37504 37627   92002-15006 REV. 1805 771205
      ID.A      37630 37717   92002-16008 780207
      WRISS     37720 37756   92002-16006 740801
      READ.     37757 40003   92002-16006  740801
```

```
   %WRIS      40006 40403  750701  24998-16001
   SREAD      40404 41046  771116  24998-16001

FMGR9(0099)32770 32776  92002-16008 760720
   LI..       33001 34503  92002-16008 760720
   CL..       34603 35064
   LU..       35071 36212  92002-16008 760702
   RNRQ       36253 36514  92067-16035 REV.1805 780222
   KCVT       36515 36530  92001-16035 REV.1805 770621
   $CVT3      36531 36616  92067-16035 REV.1805 770621
   READF      36617 37355  92002-16006 770801
   REIO       37356 37462  92067-16035 REV.1805 780221
   FSTAT      37463 37507  92002-16006  740801
   LOCF       37510 37676  92002-16006 750416
   POST       37677 37725  92002-16006  740801
   P.PAS      37726 37754  92002-16006 740801
   RW$UB      37757 40230  92002-16006 750422
   SPOPN      40232 40302  92002-16006  741025
   LULU.      40303 40373  92002-16006 760227
   RANGE      40374 40417  92002-16006 740801
   AVAIL      40420 40512  92002-16006 741231

LUMAP(0020)26002 27065
   FMTIO      27074 30512   24998-16002 REV.1805 780303
   REIO       30560 30664  92067-16035 REV.1805 780221
   FMT.E      30665 30665   24998-16002 REV.1805 780303
   FRMTR      30726 33527   24998-16002 REV.1805 780303
   CLRIO      33740 33746  750701  24998-16001
   RMPAR      33750 34006  771116  24998-16001
   PNAME      34010 34055   771121  24998-16001

KYDMP(0010)26002 27326
   KCVT       27327 27342  92001-16035 REV.1805 770621
   $CVT3      27343 27430  92067-16035 REV.1805 770621
   OPEN       27431 27616  92002-16006 741205
   READF      27634 30372  92002-16006 770801
   REIO       30422 30526  92067-16035 REV.1805 780221
   CLOSE      30527 30645  92002-16006  771115
   $OPEN      30646 31054  92002-16006   740801
   P.PAS      31055 31103  92002-16006 740801
   RW$UB      31104 31355  92002-16006 750422
   RWND$      31356 31500  92002-16006  771121
   R/W$       31501 31634  92002-16006  740801
   CLRIO      31635 31643  750701  24998-16001
   IAND       31644 31653   750701  24998-16001
   RMPAR      31654 31712  771116  24998-16001

USPMP(0099)26002 26437  91730-16003 REV 1805 780117
   FMTIO      26442 30060   24998-16002 REV.1805 780303
   REIO       30064 30170  92067-16035 REV.1805 780221
   FMT.E      30171 30171   24998-16002 REV.1805 780303
   FRMTR      30212 33013   24998-16002 REV.1805 780303
   CLRIO      33154 33162  750701  24998-16001
   RMPAR      33163 33221  771116  24998-16001
   PNAME      33222 33267   771121  24998-16001
   CNVSC      33270 33334  91730-16004 REV 1805 771219
   REPT       33365 35233

EXMP (0099)26002 32006  91730-16002 REV 1805 780117
   FMTIO      32007 33425   24998-16002 REV.1805 780303
```

```
  REIO       33426 33532  92067-16035 REV.1805 780221
  FMT.E      33533 33533   24998-16002 REV.1805 780303
  FRMTR      33560 36361   24998-16002 REV.1805 780303
  CLRIO      36401 36407  750701  24998-16001
  PAUSE      36410 36510  771122  24998-16001
  RMPAR      36511 36547  771116  24998-16001
  PAU.E      36550 36550  750701  24998-16001
  PNAME      36551 36616   771121  24998-16001

LGTAT(0099)26002 30050  92067-16008 REV.1805 780127
  $CVT3      30051 30136  92067-16035 REV.1805 770621




RT PARTITION REQMTS:
  SMP   04 PAGES
  JOB   04 PAGES
  SPOUT 02 PAGES

BG PARTITION REQMTS:
  $CNFX 03 PAGES
  AUTOR 02 PAGES
  LOADR 09 PAGES
  GASP  06 PAGES
  FMGR  07 PAGES
  LUMAP 05 PAGES
  KYDMP 03 PAGES
  DSPMP 05 PAGES
  EXMP  06 PAGES
  LGTAT 03 PAGES

MAXIMUM PROGRAM SIZE:
W/O COM 29 PAGES
W/  COM 28 PAGES
W/  TA2 22 PAGES


SYS AV MEM:  02944 WORDS

1ST PART PG  00044
 CHANGE 1ST PART PG ?
TR
 CHANGE 1ST PART PG ?
45

SYS AV MEM:  03968 WORDS

PAGES REMAINING: 00083

DEFINE PARTITIONS:

PART  01?
2,BG

PART  02?
5,BG

PART  03?
11,BG
```

```
PART 04?
65,BG

SUBPARTITIONS?
YES

PART 05?
7,S

PART 06?
7,S

PART 07?
22,S

PART 08?
29,S

PART 09?
/E

MODIFY PROGRAM PAGE REQUIREMENTS?
-
FMGR,7
-
LOADR,20
-
/E

ASSIGN PROGRAM PARTITIONS?
-
/E

SYSTEM STORED ON DISC
SYS SIZE: 35 TRKS, 036 SECS

RT4GN FINISHED

0000 ERRORS
```

PARTITION STATES

```
@@@
@@@ 0:11:43:730
@@@*************************************************************************
\@@PTN#  SIZE    PAGES   BG/RT PRGRM
@@@*************************************************************************
@@@
@@@
@@@
@@@
@@@
@@@
@@@ 7S     22     77-   98 BG     FMG16
@@@
@@@
@@@1
@@@11   <UNDEFINED>
@@@12   <UNDEFINED>
@@@13   <UNDEFINED>
@@@14   <UNDEFINED>
@@@15   <UNDEFINED>
@@@*************************************************************************
@@@ 0:11:43:780
@@@
```

NOTE: $MATA = 23500B
$MNP = 15.

PROGRAM STATES

```
@@@
@@@ 0:11:28:690
@@@*****************************************************************************
'@@PT SZ PRGRM,T ,PRIOR*DRMT*SCHD*I/O *WAIT*MEMY*DISC*OPER * NEXT TIME  *
@@@*****************************************************************************
@@@
@@@
@@@ 2  5 TVST *3 *00001  0 *********************************** 0:11:28:950
@@@
@@@
@@@
@@@*****************************************************************************
@@@DOWN LU'S,
@@@*****************************************************************************
@@@DOWN EQT'S
@@@*****************************************************************************
@@@ 0:11:28:730
@@@
```

```
LOCATIONS   1600 THROUGH    1677
033127 037624 035725 033016 032253 025467 024715 024352*6W? ! 6 4 +7) (
024325 014011 114016 014016 014015 003251 003263 003277*(
003232 012000 003214 003207 006057 006274 006120 006135*        /   P )
006067 006302 006303 006207 006255 006240 006307 006266* 7
006265 002000 003221 003313 003223 000000 000000 036306*             <
002023 000023 002762 000051 003104 000072 023717 016071*       ) D !'  9
002136 002137 002140 002141 002142 002143 002144 002145* ↑ ←
002146 002147 002150 000006 000015 002042 000003 051474*           "  S<
LOCATIONS   1700 THROUGH    1777
000003 051552 051540 000000 000000 000000 000000 000000*  S S
000000 016535 000000 017755 000000 000000 000000 016535*   )           )
016535 016536 016537 016540 016541 016542 016543 016544* ) ↑ ←
016545 016546 016547 016550 000000 000000 031017 000000*           2
000224 000057 000002 001445 000002 026000 011107 000144*   /    %  ,  G
052654 052654 011253 000525 052654 177400 000400 000140*U U    UU
000000 010422 001224 004646 000234 000000 000000 000000*
000000 002151 002152 002153 002154 030000 000000 052654*          0   U
LOCATIONS   2000 THROUGH    2077
177772 000000 011000 000400 000403 011000 000226 000000*
020000 000313 000147 020000 000313 000316 020000 000313*
000465 020000 000313 000000 005127 004214 100017 015120* 5     W    P
001101 036575 177700 000042 000105 136575 000100 000000* A=    " E   @
000000 000000 000000 004056 004145 010024 002400 000401*       .
003504 000021 007230 007252 005752 000401 002460 100000* D           0
000000 000000 004016 005106 100010 017000 000000 000000*       F
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS   2100 THROUGH    2177
000000 004065 004601 000023 017400 000000 000000 000000*   5
000000 000000 000000 000031 002475 164217 000000 017714*        =
004056 004145 010025 102402 000401 003431 000044 007062* .          $ 2
007172 004517 000401 002526 100000 101135 061240 005553*  O   V   )
006100 040016 105000 040202 142542 177666 177711 000000* @@   @
100047 177754 177703 100000 100001 000000 006763 007200* '
040025 005000 000000 000000 000000 000000 000000 000000*@
000000 000000 100000 000000 000000 006322 007137 100021*          ←
LOCATIONS   2200 THROUGH    2277
011401 000403 177776 000044 000000 177766 000006 000000*      $
130740 100000 000000 000000 004070 004637 000022 003400*        8
000000 000000 000000 000000 000000 000000 000005 002543*
000000 000000 000000 004070 004637 000077 003400 000000*      8   ?
000000 000000 000000 000000 000000 000005 002550 000000*
000000 000000 004070 004637 000077 003400 000000 000000*     8   ?
000000 000000 000000 000000 000005 002555 000000 000000*
000000 007166 007401 130120 005400 000002 033226 000036*      P    6
LOCATIONS   2300 THROUGH    2377
000000 000000 000000 000000 000000 177753 000000 000000*
004070 004637 000077 003400 000000 000000 000000 000000* 8   ?
000000 000000 000005 002562 000000 000000 000000 004056*             .
004145 040026 002400 000000 000000 000000 000000 000000*  @
000000 000015 002567 100000 000000 000000 012654 014266*
000060 021400 000000 000000 000000 000000 000000 000000* 0#
000022 002604 000000 000000 000000 012654 014266 000061*             1
021400 000000 000000 000000 000000 000000 000000 000022*#
LOCATIONS   2400 THROUGH    2477
002626 000000 000000 000000 012654 014266 000062 021400*           2#
000000 000000 000000 000000 000000 000000 000022 002650*
000000 000000 000000 012654 014266 000063 021400 000000*         3#
000000 000000 000000 000000 000000 000022 002672 000000*
000000 000000 012577 012472 000004 021400 000000 000000*     !  #
```

```
000000 000000 000000 000000 000000 000000 000000 000000*
000001 016351 000000 000000 000020 000000 000000 006216*
000000 000000 000000 006423 000000 000000 000000 000000*
LOCATIONS    2500 THROUGH     2577
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000001 016351*
000000 000000 000000 000000 000000 006216 000000 000000*
000000 006423 000002 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS    2600 THROUGH     2677
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS    2700 THROUGH     2777
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000002 000030 000032 000036*
000030 000002 000030 000002 000034 000034 000034 000002*
000034 000030 100001 100001 100001 100001 100001 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000002 000001 000000 004002 010002 000006*
014002 000110 000011 000012 000013 004014 000015 004001*    H
LOCATIONS    3000 THROUGH     3077
000000 000005 000005 000005 000005 000000 010001 000004*
004004 010004 014004 014001 000016 004016 010016 014016*
020001 000003 004003 000017 000020 000021 000022 000006*
000007 024001 000023 000000 000000 000000 000000 000000*
000000 000000 100000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
```

```
LOCATIONS  16000 THROUGH   16077
016030 016033 016036 016041 016044 016047 016052 016055*        ! $ ' * -
016060 016063 016066 000000 000000 000000 000000 000000* 0 3 6
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 016257 016314 016351 016406 016443 016500 016535*          # @ )
LOCATIONS  16100 THROUGH   16177
016572 016633 016674 016735 016776 017037 017100 017141*             @
017202 017243 017304 017345 017406 017447 017510 017551*           ' H
017612 017653 017714 017755 020016 020057 020120 020161*           / P
020222 020263 020324 020365 020426 020467 020530 020571*        ! !7!X!
020632 020673 020734 020775 021036 021077 021140 021201*! ! ! ! " "?" "
021242 021303 021344 021405 021446 021507 021550 021611*" " " # #&#G# #
021652 021713 021754 022015 022056 022117 022160 022221*# # # $ $.$0$ $
022262 022323 022364 022425 022466 022527 022570 022616*$ $ $ % %6%W% %
LOCATIONS  16200 THROUGH   16277
022627 022640 022651 022662 022673 022704 022715 022726*% % % % % % %
022737 022750 022761 022772 023003 023014 023025 023036*% % % % & & &
023047 023060 023071 023102 023113 023124 023135 023146*&'&0&9&B&K&T&]&
023157 023170 023201 023212 023223 023234 023245 023256*& & & & & & &
023267 023300 023311 023322 023333 023344 023355 023366*& & & & & & &
023377 023410 023421 023432 023443 023454 000000 000000*& ' ' ' '*',
000000 000000 000000 000000 000000 000012 030011 000000*           0
000000 016260 030016 042530 052116 042001 000000 000000*     0 EXTND
LOCATIONS  16300 THROUGH   16377
000000 025000 177574 000000 001000 030000 030214 000004*  *        0 0
000011 000000 000000 000000 017612 000000 000000 000000*
000000 000000 000001 030460 000000 002301 016315 031010*      10      2
042056 051124 051001 000000 000000 000000 025000 177574*D.RTR         *
000000 000200 030214 032417 000011 000013 000000 000000*     0 5
000000 017612 002003 000000 177767 000000 000000 000012*
032421 032737 002402 000000 032732 050122 046520 052001*5 5      5 PRMPT
000000 000000 000000 025000 177574 000000 000200 032417*      *       5
LOCATIONS  16400 THROUGH   16477
033216 000013 000016 000000 000000 000000 000000 000000*6
000000 000000 000000 000000 000012 033220 000000 000000*          6
016407 000000 051044 050116 022001 000000 000000 000000*    RSPNS
025000 177574 000000 000200 033216 034124 000016 000024**        6 8T
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000002 034126 000000 000000 016444 000000 052124*    8V      $  TT
054505 053001 000000 000000 000000 025000 177574 000000*YEV           *
001000 034124 034136 000024 000026 000000 000000 000000*  8T8↑
LOCATIONS  16500 THROUGH   16577
017612 000000 000000 000001 000000 000000 000001 034154*              8
000000 002400 016501 135336 053510 055101 052001 000000*     A  WHZAT
000000 000000 025000 177574 000000 000200 034136 036306*   *        8↑<
000026 000031 000000 000000 000000 017612 000054 103324*              ,
000021 000000 000000 000132 041015 050017 051466 050014*        ZB P S6P
100000 041515 046464 020001 000001 000000 000000 025000*  CMM4          *
177574 040017 000200 036306 052655 000031 000266 000000*  @   < U
000000 177777 000000 000000 000000 000000 000000 000000*
LOCATIONS  16600 THROUGH   16677
000036 026064 000000 000000 016573 000000 051515 050040*  ,4         SMP
020002 000000 000000 000000 025000 177574 000000 007000*        *
026000 033604 000002 000173 001616 000000 000000 000000*, 7
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000036 026002 000000 000000 016634 000000 045117*  . H-30     JO
```

```
041040 020002 000000 000000 000000 025000 177574 000000*B         *
007000 026000 033522 000002 000045 001700 000000 000000*  , 7R   %
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  16700 THROUGH   16777
000000 000000 000013 026107 000000 000000 016675 000000*      ,G
051520 047525 052002 000000 000000 000000 025000 177574*SPOUT      *
000000 003000 026000 027464 000002 000021 002020 000000*     , /4
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000143 026273 000000 000000 016736*     ,
000000 026054 026054 026003 000000 000000 000000 025000*  ,,,,,       *
177574 000000 005200 026000 031770 000002 000162 002040*      , 3
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  17000 THROUGH   17077
000000 000000 000000 000000 000001 026002 000000 000000*        ,
016777 000000 040525 052117 051003 000000 000000 000000*    AUTOR
025000 177574 000000 003000 026000 026445 000002 000004**      , -%
002102 000000 000000 000000 000000 000000 000000 000000* B
000000 000000 000000 000000 000000 000120 026104 000000*         P,D
000000 017040 000000 046117 040504 051003 000000 000000*      LOADR
000000 025000 177574 000000 047200 026000 045767 000002*  *     N , K
000651 002112 000000 000000 000000 000000 000000 000000*   J
LOCATIONS  17100 THROUGH   17177
000000 000000 000000 000000 000000 000000 000120 026661*          P-
000000 177777 017101 127064 043501 051520 020003 000000*      A 4GASP
000000 000000 025000 177574 000000 013005 026000 033603*    *    , 7
000002 000064 002422 000000 000000 037537 000000 000000*   4     ?←
000000 000000 016535 103577 000044 000000 000000 000132*    ]   $     Z
026222 033312 000000 177777 133276 043115 043522 020043*, 6      FMGR *
110003 000000 000000 025000 177574 000000 015202 026000*      *     ,
032770 000002 000045 002642 000000 000000 041047 000000*5    %      B'
LOCATIONS  17200 THROUGH   17277
000000 177777 017612 000601 163646 177572 000000 000000*
000024 026144 026245 000087 027271 126240 046125 046501*  , ,  .    LUMA
050003 000000 000000 000000 025000 177574 000000 007205*P         *
026000 033675 000002 000354 016400 000000 000000 000000*, 7
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000012 026002 000000 000000 017244 000000 045531*   ,         KY
042115 050003 000000 000000 000000 025000 177574 000000*DMP       *
005200 026000 031713 000002 000025 004246 000000 000000*  , 3
LOCATIONS  17300 THROUGH   17377
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000143 026012 000000 000000 017305 000000*      ,
042123 050115 050003 000000 000000 000000 025000 177574*DSPMP      *
000000 011200 026000 035237 000002 000027 004310 000000*    , :
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000143 031154 000000 000000 017346*      2
000000 042530 046520 020003 000000 000000 000000 025000*  EXMP       *
177574 000000 013200 026000 036617 000002 000253 004446*      , =     &
LOCATIONS  17400 THROUGH   17477
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000143 026002 000000 000000*        ,
017407 000000 046107 052101 052003 000000 000000 000000*     LGTAT
025000 177574 000000 005200 026000 030137 000002 000042**     , 0←   "
004622 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000143 026002 000000*        ,
000000 017450 000000 054122 042506 020203 000000 000000*    (  XREF
000000 025000 177574 000000 035200 026000 041173 000002*  *    ; , B
LOCATIONS  17500 THROUGH   17577
000436 026222 000000 000000 000000 000000 000000 000000* ,
000000 000000 000000 000000 000000 000000 000143 026046*           ,&
```

```
000000 002400 017511 145422 046511 041522 047603 000000*      I  MICRO
000000 000000 025000 177574 000000 017200 026000 042176*    *        , D
000002 000511 067474 000000 000000 000000 000000 000000*   I <
000000 017612 000006 033226 000035 000000 000000 000001*      6
026720 031333 026476 000000 131324 052126 051524 020303*- 2 ->    TVST
020000 000000 050000 007466 177575 000000 011001 026000*    P  6        ,
LOCATIONS  17600 THROUGH   17677
034320 000002 000151 063100 000000 000000 000000 000000*8      @
000000 000000 000000 000001 000000 000000 000000 000000*
077777 026003 026017 000000 017613 003116 046505 046440*  , ,       NMEM
020303 000001 000000 000000 025000 177574 000000 003000*        *
026000 026037 000002 000006 063072 000000 000000 000000*, ,      :
000000 000200 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  17700 THROUGH   17777
000000 000000 000200 000000 000000 004244 000000 000000*
000000 000000 000000 000000 000000 000401 103564 000044*              $
000000 000000 000132 026222 037706 037677 000047 137676*      Z, ? ?  '
043115 043461 033243 100002 000000 000000 025000 177574*FMG16       *
026003 015206 026000 032770 000002 000045 025204 000000*,     , 5    %*
000000 041047 000000 000000 177760 017141 000004 177776*  B'
000044 000000 000000 000132 026222 037745 037745 000010* $     Z, ? ?
137744 043115 043460 032643 100003 000000 000000 025000*  FMG05        *
LOCATIONS  20000 THROUGH   20077
177574 000400 015204 026000 032770 000002 000045 025204*       , 5    %*
000000 000000 041047 000000 000000 177777 017612 000000*     B'
000000 000000 000000 000000 000132 026007 000000 002402*         Z,
020017 126022 000000 000000 000000 000000 000000 000000*
025000 177574 000000 003207 026000 026761 000002 000011**        , -
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20100 THROUGH   20177
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20200 THROUGH   20277
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20300 THROUGH   20377
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
```

```
LOCATIONS  20400 THROUGH   20477
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20500 THROUGH   20577
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20600 THROUGH   20677
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  20700 THROUGH   20777
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21000 THROUGH   21077
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21100 THROUGH   21177
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21200 THROUGH   21277
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
```

```
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21300 THROUGH   21377
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21400 THROUGH   21477
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21500 THROUGH   21577
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21600 THROUGH   21677
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  21700 THROUGH   21777
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22000 THROUGH   22077
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22100 THROUGH   22177
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
```

```
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22200 THROUGH   22277
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22300 THROUGH   22377
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22400 THROUGH   22477
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22500 THROUGH   22577
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
LOCATIONS  22600 THROUGH   22677
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 000000 000000 000000 000000 000000 000000 000000*
000000 033603 043501 051520 030425 033603 036751 000064*   7 GASP1 7 =   4
000122 002504 033603 043501 051520 031025 033603 037537* R D7 GASP2 7 ?←
000064 000122 002600 032770 043115 043522 030025 032770* 4 R   5 FMGR0 5
041032 000045 000105 002714 032770 043115 043522 030425*B   % E   5 FMGR1
032770 041004 000045 000147 003040 032770 043115 043522*5 B   %     5 FMGR
LOCATIONS  22700 THROUGH   22777
031025 032770 040633 000045 000113 003124 032770 043115*2 5 A   % K T5 FM
043522 031425 032770 037700 000045 000104 003246 032770*GR3 5 ?   % D   5
043115 043522 032025 032770 040723 000045 000112 003320*FMGR4 5 A   % J
032770 043115 043522 032425 032770 040153 000045 000130*5 FMGR5 5 @   % X
003442 032770 043115 043522 033025 032770 040534 000045* "5 FMGR6 5 A\ %
000143 003516 032770 043115 043522 033425 032770 040453*    N5 FMGR7 5 A+
000045 000122 003636 032770 043115 043522 034025 032770* % R   5 FMGR8 5
041047 000045 000113 003716 032770 043115 043522 034425*B' % K   5 FMGR9
```

```
LOCATIONS  23000 THROUGH   23077
032770 040513 000045 000120 004042 044622 043064 027060*5 AK % P "I F4.0
020225 044621 050405 000613 000752 026666 045014 043064*  I Q     - J F4
027061 020225 044621 051044 000613 000761 026730 044622*.1  I RS    - I
043064 027062 020225 044621 045573 000613 000627 027040*F4.2  I K      .
044655 043064 027063 020225 044621 047016 000613 000641*I F4.3  I N
027054 044622 043064 027064 020225 044621 051162 000613*.,I F4.4  I R
000753 027102 044622 043064 027065 020225 044621 050507*  .BI F4.5  I QG
000613 000722 027212 040470 040523 046502 030225 040446*     . A8ASMB0 A&
LOCATIONS  23100 THROUGH   23177
042327 000415 000470 026026 040527 040523 046502 030625*D     8, AWASMB1
040446 042522 000415 000540 026052 040524 040523 046502*A&ER     ,*ATASMB
031225 040446 042555 000415 000512 026100 040510 040523*2 A&E    J,@AHAS
046502 031625 040446 041365 000415 000432 026126 040513*MB3 A&B      ,VAK
040523 046502 032225 040446 041774 000415 000430 026202*ASMB4 A&C      ,
032671 041101 051503 030625 032263 035750 000123 000357*5 BASC1 4 ;  S
034060 032367 041101 051503 031225 032254 037777 000123*804 BASC2 4 ?  S
000176 034124 032422 041101 051503 031625 032264 035030*  8T5 BASC3 4 :
LOCATIONS  23200 THROUGH   23277
000123 000240 034246 032551 041101 051503 032225 032266* S   8 5 BASC4 4
041664 000123 000527 034300 032475 041101 051503 032625*C  S W8 5=BASC5
032273 036630 000123 000313 034444 032434 041101 051503*4 =  S  9$5 BASC
033225 032273 036706 000123 000364 034514 032564 041101*6 4 =  S  9L5 BA
051503 033625 032306 036332 000123 000313 034630 032370*SC7 4 <  S  9 4
041101 051503 034225 032265 034163 000123 000205 034676*BASC8 4 8  S  9
000000 000000 000000 000020 000000 000000 000000 000000*
000000 000000 000000 000000 000020 000000 000000 000000*
LOCATIONS  23300 THROUGH   23377
000000 000000 000000 000000 000000 000020 000000 000000*
000000 000000 000000 000000 000000 000000 000020 000000*
000000 000000 000000 000000 000000 000000 000000 000020*
000000 000000 000000 000000 000000 000000 000000 000000*
000020 000000 000000 000000 000000 000000 000000 000000*
000000 000020 000000 000000 000000 000000 000000 000000*
000000 000000 000020 000000 000000 000000 000000 000000*
000000 000000 000000 000020 000000 000000 000000 000000*
LOCATIONS  23400 THROUGH   23477
000000 000000 000000 000000 000020 000000 000000 000000*
000000 000000 000000 000000 000000 000020 000000 000000*
000000 000000 000000 000000 000000 000000 000020 000000*
000000 000000 000000 000000 000000 000000 000000 000020*
000000 000000 000000 000000 000000 000000 000000 000000*
000020 000000 000000 000000 000000 000000 000000 000000*
000000 000020 000000 000000 000000 000000 000000 000000*
000000 000000 000020 000000 000000 000000 000000 000000*
LOCATIONS  23500 THROUGH   23577
023534 077777 017612 000055 000001 000001 000000 000000*'\        -
000001 017551 000057 000004 000001 000000 023552 000132*      /      '  Z
017141 000064 000012 000001 000000 000000 000000 000000*   4
100077 000100 000000 023534 023516 000132 017755 000077* ? @   '\'N Z   ?
000006 000001 023543 027275 000024 017202 020106 000006*     ' .     F
000001 023552 023507 000132 017714 000115 000025 000001*  ' 'G Z   M
023561 000000 000132 000000 000143 000034 000001 023525*'     Z        'U
177777 000000 000000 000000 000000 000000 000000 177777*
LOCATIONS  23600 THROUGH   23677
000000 000000 000000 000000 000000 000000 177777 000000*
000000 000000 000000 000000 000000 177777 000000 000000*
000000 000000 000000 000000 177777 000000 000000 000000*
000000 000000 000000 177777 000000 000000 000000 000000*
000000 000000 177777 000000 000000 000000 000000 000000*
```

```
000000 000040 000001 000002 000003 000004 040005 040006*           @ @
040007 040010 040011 040012 040041 000042 000043 000044*@ @ @ @ @! " # $
000045 000046 000047 000050 000051 000052 000053 140000* % & ' ( ) * +
```

READER COMMENT SHEET

Manual Name: ____

(Please Print)

Part Number: ____

We welcome your evaluation of this publication. Your comments and suggestions will help us improve our training materials. Please use additional pages if necessary.

Is this book technically accurate?

Did it meet your expectations?

Was it complete?

Is it easy to read and use?

Other comments?

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

____

Training Coordinator/Technical Marketing
Hewlett-Packard Co.
11000 Wolfe Road
Cupertino, California 95014

HEWLETT hp PACKARD

22999-90200

Printed in U.S.A.