60454300

CONTROL DATA
CORPORATION

NOS VERSION 1 INTERNAL MAINTENANCE SPECIFICATION

VOLUME 1 OF 3

CDC® COMPUTER SYSTEMS:
  CYBER 170 SERIES
  CYBER 70
    MODELS 71, 72, 73, 74
  6000 SERIES

SW-39

REVISION RECORD

REVISION	DESCRIPTION
A (06/26/78)	Manual released. Manual reflects NOS 1.3.
B (08/03/79)	Revised to update manual to NOS 1.4 and to make typographical and technical corrections. New features documented in this manual include: extended character set/print train support; expanded ECS status; on-line ECS diagnostic support; retry on time/SRU limit; IAF enhancements; deadstart from mass storage; CYBER 170 Model 176 support; extended TIM function; 885 Disk Storage Subsystem support; task initiated K.DUMP; TAF internal XJP trace; LIBTASK enhancements; TAF CYBER Record Manager support; and TAF/COBOL interface enhancements. This revision obsoletes all previous editions.

Publication No.
60454300

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual

PREFACE

The Network Operating System (NOS) was developed by Control Data Corporation to provide network capabilities for time-sharing and transaction processing, in addition to local and remote batch processing, on CONTROL DATA CYBER 170 Series Computer Systems; CDC CYBER 70 Series, Models 71, 72, 73, and 74 Computer Systems; and CDC 6000 Series Computer Systems.

AUDIENCE

This internal maintenance specification (IMS) provides the systems analyst with detailed internal documentation of NOS. Included are detailed descriptions of system routines and the system interfaces, tables, and flowcharts of these routines. Some user interfaces are mentioned, but these are fully described in other NOS manuals.

CONVENTIONS

Extended memory for the CYBER 170 Models 171, 172, 173, 174, 175, 720, 730, 750, and 760 is extended core storage (ECS). Extended memory for CYBER 170 Model 176 is large central memory (LCM) or large central memory extended (LCME). ECS and LCM/LCME are functionally equivalent, except as follows:

- LCM/LCME cannot link mainframes and does not have a distributive data path (DDP)capability.
- LCM/LCME transfer errors initiate an error exit, not a half exit. Refer to the COMPASS Reference Manual for complete information.

The Model 176 supports direct LCM/LCME transfer COMPASS instructions (octal codes 014 and 015). Refer to the COMPASS Reference Manual for complete information.

In this manual the acronym ECS refers to all forms of extended memory on the CYBER 170 Series. However, in the context of a multimainframe environment or DDP access, the Model 176 is excluded.

In this manual, the order of importance of headings is denoted as follows.

LEVEL 1 HEADINGS ARE FULL CAPS AND UNDERLINED

LEVEL 2 HEADINGS ARE FULL CAPS

Level 3 Headings are First-Capped and Underlined

Level 4 Headings are First-Capped

Conventions for central memory word formats are as follows:

- Cross-hatching indicates a field is not used by or is not applicable to a function processor. However, CDC reserves the right to assign these fields to system use in the future.
- Fields reserved for system use are so labeled.
- Fields labeled with mnemonics indicate a specific parameter must be inserted (generally described after the word format).
- Fields with numeric identifiers indicate the actual value that is used or returned for a particular function.

RELATED PUBLICATIONS

For further information concerning CYBER 170, CYBER 70, and 6000 Series Computer Systems, the NOS time-sharing systems, and the user interface for NOS, consult the following manuals.

Control Data Publication	Publication No.
CYBER 170 Computer Systems Reference Manual	60420000
CYBER 170 Computer Systems Models 720, 730, 750, and 760 Model 176 (Level B)	60456100
CYBER 70/Model 71 Computer System Reference Manual	60453300
CYBER 70/Model 72 Computer System Reference Manual	60347000
CYBER 70/Model 73 Computer System Reference Manual	60347200
CYBER 70/Model 74 Computer System Reference Manual	60347400
Modify Reference Manual	60450100
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Transaction Facility Version 1 Reference Manual	60455340
Network Products Transaction Facility Version 1 User's Guide	60455360
Network Products Transaction Facility Version 1 Data Manager Reference Manual	60455350

Control Data Publication	Publication No.
Network Products Transaction Facility Version 1 CYBER Record Manager Data Manager Reference Manual	60456710
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Access Method Version 1 Internal Maintenance Specification	60490110
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
NOS Version 1 Installation Handbook	60435700
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual Volume 1	60435400
NOS Version 1 Reference Manual Volume 2	60445300
NOS Version 1 System Maintenance Reference Manual	60455380
NOS Version 1 System Programmer's Instant	60449200
NOS Version 1 Time-Sharing User's Reference Manual	60435500
NOS Version 1 Export/Import Reference Manual	60436200
TAF/TS Version 1 Reference Manual	60453000
TAF/TS Version 1 User's Guide	60436500
TAF/TS Version 1 Data Manager Reference Manual	60453100
TAF/TS Version 1 CYBER Record Manager Data Manager Reference Manual	60456700
6400/6500/6600 Computer System Reference Manual	60100000

DISCLAIMER

CONTENTS

FIGURES

FIGURES (Continued)

FIGURES (Continued)

FIGURES (Continued)

FIGURES (Continued)

FIGURES (Continued)

FIGURES (Continued)

TABLES

TABLES (Continued)

The Network Operating System (NOS) is a group of programs and subprograms that monitors the input, compilation, assembly, loading, execution, and output of all jobs submitted to the computer. NOS accepts jobs in four ways: time-sharing, local batch, remote batch, and system console input. NOS controls CYBER 170 Series Computer Systems, CYBER 70 Series, Model 71, 72, 73, and 74 Computer Systems, and 6000 Series Computer Systems.

Efficient processing of user jobs is the prime objective of the operating system. This section describes the inherent hardware characteristics, the basic software elements, and how they work together to accomplish the prime objective. Figure 1-1 shows the NOS system equipment configuration.

HARDWARE OVERVIEW

NOS uses peripheral processors (PP) for system and input/output tasks and one or two central processor units (CPU) to execute user and system jobs. Central memory (CM) contains user programs; system software areas are located at the lower end of central memory. Extended core storage (ECS) may also be used by NOS.

CENTRAL PROCESSOR UNIT

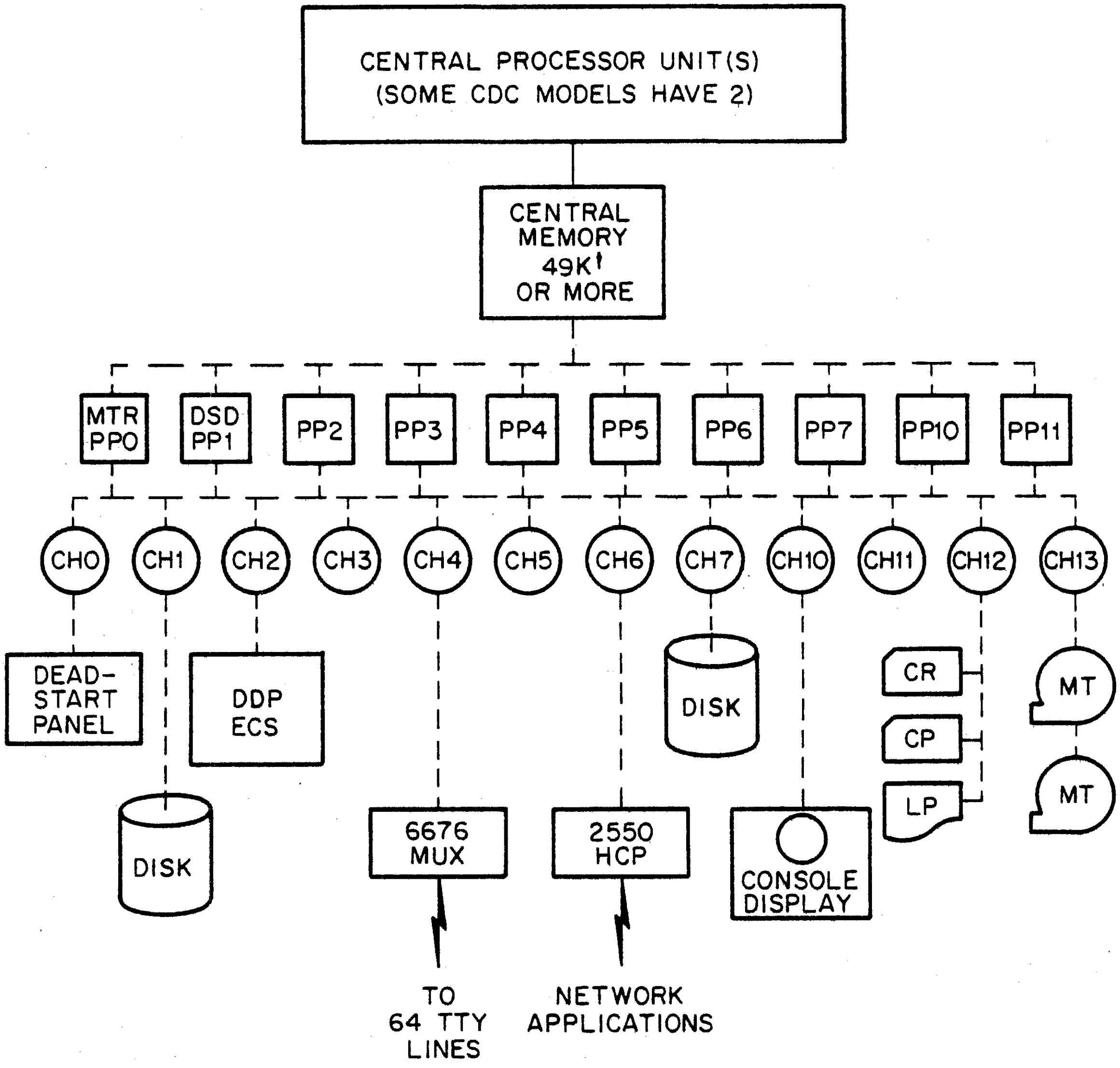
The CPU performs tasks of a computational nature; it has no input/output capability. It communicates with other system components through central memory. Under NOS, the CPU is used almost exclusively for program compilations, assemblies, and executions. The CPU makes system requests through a CPU request register located at the reference address plus one (RA+1) of the current program in execution. However, system work that can be done more efficiently in the CPU is processed there.

PERIPHERAL PROCESSORS

The system may have up to 20 peripheral processors. The peripheral processors (identified as PP0, PP1, ..., PPn) are identical and perform many tasks for requesting programs in central memory. Each PP consists of 4K, 12-bit, 1-byte words of memory.

A PP can control input/output, job scheduling, control statement interpreting, system housekeeping, and other tasks as required. Tasks are assigned one at a time to each PP by the CPU monitor (CPUMTR). When an assigned task is completed, the PP signals the system. CPUMTR waits for this signal before assigning another task to the PP.

Each PP is assigned a block of eight words in central memory resident through which communication with the system is conducted. This area is referred to as the PP communications area. Each block contains an input register, an output register, and a message buffer.

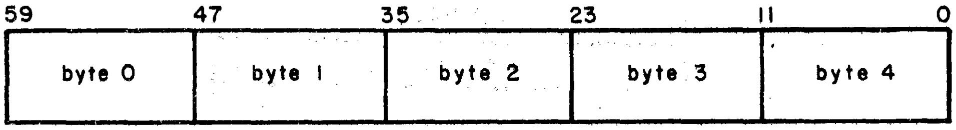


[†] Special consideration is needed for NOS to execute with 49K of central memory (refer to the NOS Installation Handbook).

Figure 1-1. System Equipment Configuration

CENTRAL MEMORY

Central memory words are 60 bits long; each is composed of five 12-bit bytes. Each 12-bit byte in a CM word is numbered 0 through 4, from the left, as follows.



One or more user programs may be in some state of execution concurrently under NOS. These programs are stored in central memory in an assigned user area called control points; a set of system components necessary for the operation of the system is also stored in central memory, forming central memory resident (CMR). Central memory is accessible by all PPs and CPU(s) and forms the communication link between all processor units in the computer system.

CMR contains system communication areas, system tables, CPU resident routines, the library directory, and information about each job currently in execution.

EXTENDED CORE STORAGE

Extended core storage (ECS) is a high-speed peripheral storage device. It is used by the multimainframe software for storage of common tables since ECS can be accessed by two or more mainframes. ECS is also used to retain system routines and compilers that are called frequently. It is often used by the system to move blocks of central memory. This is known as a storage move of control points and is described later. ECS may also be used for rolling jobs out of central memory, and user created files, and for direct access of large data arrays by using the read/write ECS instructions.

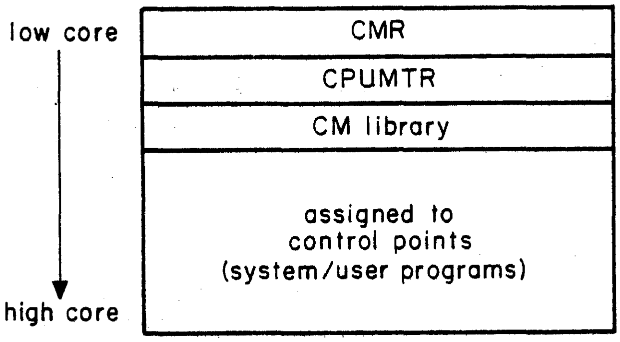
SOFTWARE OVERVIEW

Under NOS all processing of user jobs is controlled in central memory. NOS consists of PP programs, CPU programs, macro definitions, and symbol definitions. The entire system is contained on a magnetic tape file produced by the NOS utility Modify. Programs in the library file are in source language form. Installation options are provided to permit flexible selection of system features during the assembly and creation of an NOS deadstart (system initialization) medium. The most frequently used options are selected during deadstart.

A system monitor is in complete supervisory control of the hardware system. The system monitor is composed of PP routine MTR (PP monitor) which operates in PP0, and CPUMTR (CPU monitor) which is loaded as part of central memory resident (CMR).

CENTRAL MEMORY ORGANIZATION

The allocation of central memory is as follows.



Low core is allocated to the central memory resident portion of NOS and executable system programs. The remaining area is assigned to control points.

CONTROL POINTS

The system can control execution of several jobs at one time. When placed into CM before execution, each job is assigned a control point number. Jobs at control points are assigned to a processor for execution. Each control point area in CMR contains all the information necessary to process the assigned job.

Control Point Concepts

Blocks of central memory storage not allocated for system use are ordered by control point number and assigned to jobs. Each control point number has a corresponding table in CMR called the control point area. A control point is not a physical entity, but rather a concept used to facilitate bookkeeping. The control point number and the control point area, however, are physical quantities that do appear in the system.

Under NOS up to 23 (27 octal) control points are possible. In an installation with n control points for user jobs they are numbered from 1 to n. A job assigned to a control point is identified by its control point number; only one job can be assigned to a control point at any one time. Once a job is assigned to a control point, system resources such as central memory, ECS, channels, equipment, and processors may be assigned to the control point for use by the job.

The amount of CM/ECS words assigned to a single control point is contiguous and an integer multiple of 100B for CM and 1000B for ECS; storage for all control points is not necessarily contiguous. The central memory storage block assigned to the job at control point 2 is higher than the block for the job at control point 1, and storage for control point 3 is higher than that for control point 2, and so on.

In the Figure 1-1.1 no storage is assigned to control points 3 and 5; unassigned storage appears between assigned storage.

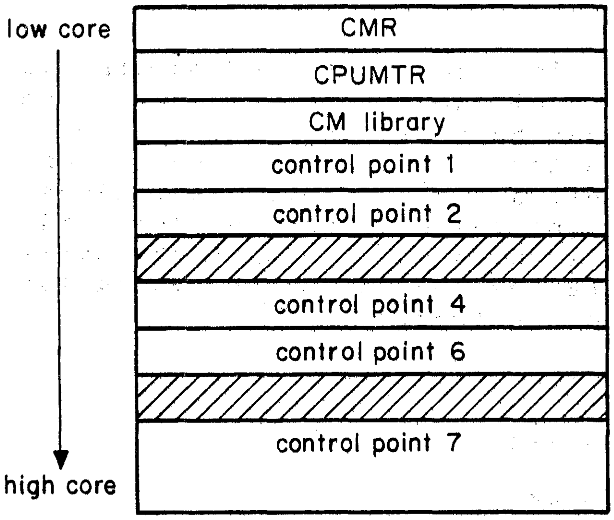


Figure 1-1.1. Central Memory Storage Layout Example

Subcontrol Points

Another feature of NOS is subcontrol points. Basically, the memory of a regular control point is divided into a number of distinct blocks. Various applications programs are loaded and executed in these blocks under the control of an executive program. The executive manages the subprograms and assigns the CPU according to priorities it establishes. The executive program and each subprogram is protected from other subprograms. This protection is accomplished by the CPU as explained in section 3. Currently, the transaction subsystem (TAF) uses this feature.

Special Control Points

In addition to the n control points defined for running jobs, there are two special control points used for system control: control point zero and control point n+1.

Control point zero is essentially CPU monitor (CPUMTR) which controls the memory of the entire machine. Also, some peripheral equipment can be assigned by control point zero to jobs at other control points and later returned to the system. Thus, the control point number associated with an equipment determines whether the system or the user has control. Similarly, logical files are associated with user jobs or the system via the control point number. Files belonging to the system (those assigned to control point zero) include:

- System dayfile
- Account dayfile
- Error log dayfile
- Jobs in the input queue
- Jobs in the rollout queue
- Jobs in the output queue

CPUMTR uses control point n+1 for certain monitor functions that might require a large amount of CPU time. For example, the delinking of tracks in a mass storage allocation table may require a significant amount of CPU time. Thus, this function is best done at control point n+1. While running at control point n+1, CPUMTR is in program mode, not monitor mode, and can be interrupted by PP exchange jumps (MXN). However, the CPU priority of control point n+1 is 100 octal, which is the highest available.

Job Rollout

During the course of execution, a job might not remain continuously at the same control point. It is possible for the job to be rolled out while it is only partially executed, thus making CM available for higher priority jobs. When a job is rolled out, it is not associated with a control point. When it is rolled back in, it is probably associated with a control point other than its previous control point.

During the time a job is rolled out, the only table in CMR that contains information about the job is the file name table entry (file type rollout). The system periodically updates the priorities of rolled out jobs and eventually reschedules the job to a control point.

Storage Moves

When a job begins or finishes processing, or as jobs are rolled in and out, CM storage must be reallocated and jobs must be moved. If a job at a control point requests additional storage, it may be necessary to move jobs to obtain the required storage.

A request for a reduced field length (FL or FLE) resets the FL/FLE size in the control point area; no storage move takes place, unless the field length reduction takes place at the last control point. A request for an increased field length, when unallocated storage is available and adjacent to the control point, results in resetting the FL/FLE size in the control point area; no storage move is required.

If it is necessary to take unallocated storage adjacent to other control points to satisfy a request for increased field length, control points above and below the requesting control point will be scanned. This scan locates the combination of unallocated storage blocks that will result in a move of the least amount of storage.

In figure 1-1.1, if control point 1 needs more storage, it will be necessary to move control point 2. If control point 6 needs storage, sufficient unallocated storage may be available to make a control point move unnecessary. If, however, control point 7 needs additional storage, control points 4, 6, and 7 may be moved to provide the storage. Added storage always extends the field length upward.

Storage moves are determined by MTR and are performed by CPUMTR. There are three possible methods used by CPUMTR:

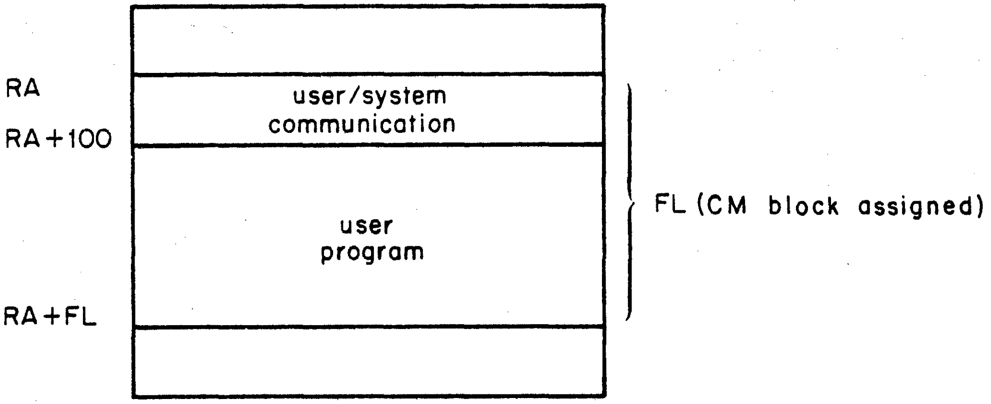
- Use compare/move unit (CMU) if available

- Use ECS block transfers if ECS is available

- Use CPU if previously mentioned hardware is unavailable

Job Field Length

When a user program is assigned a control point, the system allocates a certain amount of CM to the control point. This storage is contiguous in memory and is a multiple of 100 octal words. The block of CM assigned is defined by a starting address called the reference address (RA) and a word count field length (FL).



The user program is loaded at location RA+100, with the first 100 octal words (RA through RA+77) reserved for system communication. Once loaded, a user program cannot access memory beyond its boundaries of RA and RA+FL. The CPU uses the RA to convert addresses to absolute. If the program attempts to read or write beyond its boundaries, the CPU detects the error and aborts the job. Since the user program cannot access memory outside its FL, any area reserved for system communication must be within the FL of the job. Thus, the first 100 octal locations of each job's FL are reserved for this purpose (refer to section 2).

PROGRAM/SYSTEM COMMUNICATION

All communication with the system is performed by entering a system request in location RA+1 of the field length. A user program may communicate with the system as described in the following examples.

- The CPU does not peform input/output. Therefore the user program sends I/O requests to the system. This is most often a request for the PP program CIO.

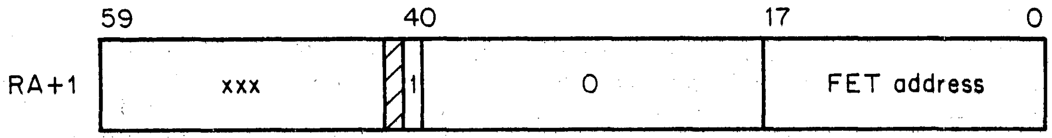
- When a user program terminates, it must advise the system that it may process the next control statement.

If a CPU program wishes to call a PP program it places the PP program name and arguments in RA+1. If autorecall is desired, bit 40 is set. If the central exchange jump (CEJ) instruction is available, the program should use it immediately after placing a call in RA+1. This causes CPUMTR to begin execution immediately. If CPUMTR determines that the RA+1 call should be assigned to a PP, CPUMTR writes the RA+1 word into the PP input register in CMR. The name and any parameters in bits 35 through 0 appear in the input register exactly as they did in RA+1. Parameters are passed from a CPU program to a PP program through this parameter field. The format for the PP communication area is shown in section 2.

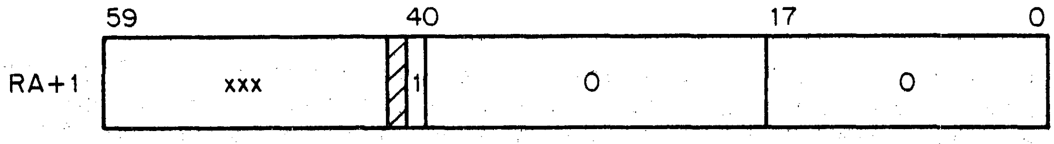
For example, if the PP program CIO is called, CIO finds the relative address of the file environment table (FET) to be used in the operation by reading its input register. It can find the RA of the control point field length by reading the control point number from its input register, computing the address of the control point area, and reading the value of RA from the control point area. By adding the RA to the relative FET address, CIO obtains the absolute address of the start of the FET. CIO then reads the parameters for the I/O operation from the FET.

MTR continually scans RA+1, in the event that the user's program does not use the central exchange jump, or the instruction is not available (CEJ/MEJ disabled). When an RA+1 call is found, MTR initiates CPUMTR.

The following illustrates an RA+1 call with the FET address specified.



A system-forced autorecall without the FET address is as follows.



Program Recall

The recall program status is provided to enable efficient use of the central processor and to capitalize on the multiprogramming capability of NOS. Often, a CPU program must wait for an I/O operation to be completed before more computation can be performed. To eliminate the CPU time wasted if the CPU program were placed in a loop to await I/O completion, a CPU program requests the control point be put into recall status until a later time; the CPU may be assigned to execute a program at some other control point. If there is nothing to do, the CPU executes an idle loop in CPUMTR.

Recall may be automatic or periodic. Autorecall should be used when a program requests I/O or other system action and cannot proceed until the request is completed. NOS does not return control until the specific request has been satisfied. Periodic recall can be used when the program is waiting for any one of several requests to be completed. The program will be activated periodically so that it can determine which request has been satisfied and whether or not it can proceed.

Periodic Recall

To enter periodic recall, a CPU program puts the characters RCL left-justified into RA+1. On encountering the RCL request, the system assigns the CPU to some other control point. After a certain interval of time has elapsed, the control point is restarted and the CPU is again assigned to execute the program at the control point.

Automatic Recall

If a CPU program makes a request in RA+1 and bit 40 of RA+1 is set to 1, the control point will be put into automatic recall after the request has been initiated. Again, the CPU is assigned to another control point as in periodic recall. In this case, however, the program in recall will be restarted by CPUMTR after the PP has dropped or issued the RCPM functions. The completion bit in the FET is never statused. The only criterion for CPU startup is the RCPM or PP drop (DPPM).

Recall and autorecall are most often used while waiting for CIO to process an I/O request. However, any time a PP program is called from RA+1, with bit 40 of RA+1 set to 1, the control point will be put into autorecall.

If bit 40 is set, bits 17 through 0 of RA+1 must contain the address of a word in the program's field length called a reply word. When the PP has completed its function, it will set the completion bit (low-order bit) in the reply word, and drop or issue an RCPM. The completion has no basic significance to NOS.

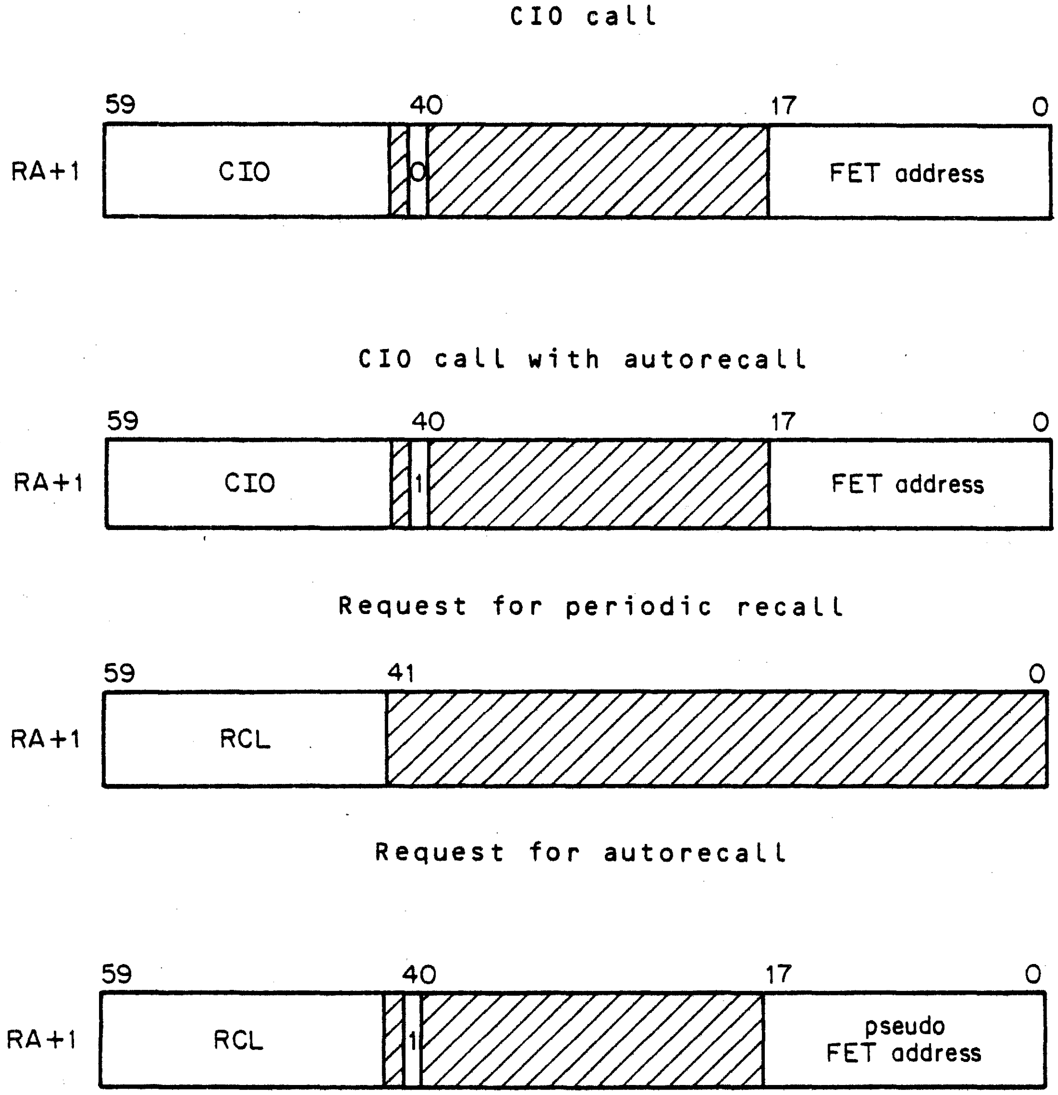
For a call to CIO, the reply word is the first word of a FET. For other programs the reply word need not be part of a FET.

A CPU program can put itself into autorecall without calling a PP program by putting RCL left-justified in RA+1 and setting bit 40 of RA+1 to 1. Bits 17 through 0 of RA+1 must contain the address of a reply word. A program which has already initiated one or more I/O operations might go into autorecall in this way, using the first word of the FET associated with one of the I/O operations as the reply word. Figure 1-2 shows the formats of RA+1 for: a normal CIO call; a request for periodic autorecall; a CIO call with autorecall bit set; and an RCL call with autorecall bit set. For periodic recall, a user must issue a normal CIO call followed by an RCL request. For autorecall, only one request is required.

Any CPU program making a call to a PP program using autorecall needs to be restarted by the PP program unless the PP program intends to drop before the CPU program is started up. Just setting the completion bit in the pseudoFET word is not enough to get the CPU program restarted. In addition, the PP routine must issue the monitor function RCPM (request CPU) to get the CPU program restarted. Unless a CPU program has queue priority greater than MXPS 7760B), all calls to PP programs, with the exception of CIO, are forced into auto-recall by CPUMTR.

Autorecall initiated by the RECALL macro is treated as follows. CPUMTR checks the completion bit and if set takes the CPU out of autorecall. If not set, CPUMTR leaves the recall request (RCLP) in RA+1 and exits. This request is detected later by MTR, and CPUMTR is called.

Normally, CPU programs use autorecall for convenience, but only one request involving autorecall can be processed at one time. For example, to initiate I/O action on several files at once, a user must employ the periodic recall technique. All requests are issued without recall (using a separate FET for each request) and then periodic recall is begun. Each time the CPU program is restarted by the system, it can check all the files for completion and go back into periodic recall if any are still incomplete.


CIO call

CIO call with autorecall

Request for periodic recall

Request for autorecall

Figure 1-2. RA+1 CIO and Request Calls

Periodic recall may be used also when a CPU program can initiate an I/O request and then perform some computation. In some cases, the I/O is completed before the computation; in others, the computation is done first. The user enters recall only when the computation is done, and then only if the I/O is still in process.

Periodic recall should also be used, if possible, to continue processing while only part of the data buffer has been read or written by the I/O driver.

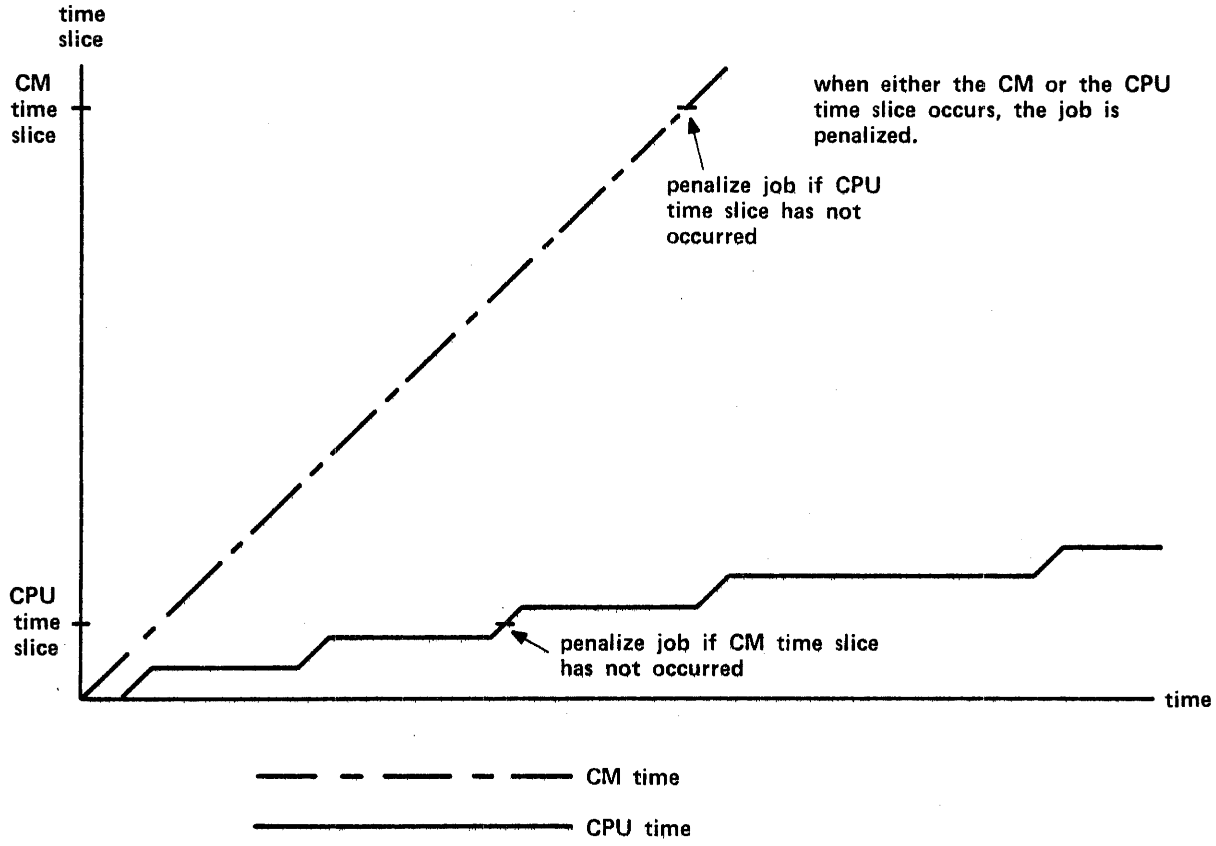
The definitions in tables 1-1 and 1-2 are used extensively in NOS. A graph of CPU and CM time slice (figure 1-3) is provided to illustrate the relationships between these two concepts.

TABLE 1-1. SYSTEM RESOURCE TIMES

Item	Description
Queue priority	The priority that governs entry to a control point from the INPUT or ROLLOUT queue and also governs disposition to a printer.
CPU priority	The priority that governs which candidate for the CPU will access the CPU.
CPU time slot	The time period when the CPU is shifted from one candidate to another.
CPU time slice	The total time period that a control point can use the CPU without being penalized.
CM time slice	The total time period a job can reside at a control point without being penalized.
Penalized means that the queue priority in the control point area is reduced to the lower queue priority (LQP) for the origin type specified.	

TABLE 1-2. JOB ORIGINS

Source	Origin Type
SYOT	System
BCOT	Local batch
EIOT	Remote batch
TXOT	Time-sharing
MTOT	Multi-terminal



CM time increases linearly with time as long as the job is at a control point without respect to the use of the CPU.

CPU time increases as a step function with a linear relation only while the job is actually using the CPU.

Figure 1-3. Graph of CM Time Slice and CPU Time Slice

CENTRAL MEMORY AND TABLES

Central memory resident (CMR) is the low end of central memory. It is reserved by NOS and provides the major coordinating area for system operation. CMR contains pointers, tables, CPU monitor (CPUMTR), libraries, and library directories.

The length of CMR is dependent upon several factors, including the number of peripheral processors, the number of control points, the number of mass storage devices, and others. This secton gives an overview of the layout of CMR giving the relative positions of the various parts of CMR, in addition to other system defined tables, symbols, and codes. The CMR part details:

- Central memory layout
- Pointers and constants
- Control point area
- PP communication area
- Dayfile buffer pointers
- Central memory tables
- System sector format
- Rollout file

The following descriptions are also provided:

- Job communication area
- Exchange package area
- Error flags
- File types
- Equipment codes
- Multimainframe tables
- PP memory layout

CENTRAL MEMORY RESIDENT

CENTRAL MEMORY LAYOUT

000 : 077	system pointers and control words
100 : 111	channel status table
112 : 122	status/control registers
123 • • 126	miscellaneous pointers and data
127 • • 141	reserved
142 • • 177	channel release table
200	control point areas
(n+1)*200	system control point
(n+2)*200	PP communication area (pointer in word 002, byte 4)

dayfile buffer pointers (pointer in word 003, byte 0)
equipment status table (EST) (pointer in word 005, byte 0)
file name/file status table (pointer in word 004, byte 0)
FNT interlock table (pointer in word 004, byte 1)
CDC CYBER 176 exchange package area
mass storage allocation area
mass storage tables (MST)
job control area
dayfile buffers
dayfile dump buffer
ECS/PP buffer
CPUMTR
resident peripheral library (RPL)
resident central library (RCL)
peripheral library directory (PLD)
central library directory (CLD)
system user library directory (LBD)

POINTERS AND CONSTANTS

	59	47	35	29	23	17	11	5 0	
000	zeros								
001	fwa resident PP library		number of PPUs		†1		memory size/100		RPLP,PPUL, CPUL,MFLL
002	fwa PP library directory		▨		number of ctrl pts		PP comm area adr		PLDP,NCPL, PPCP
003	dayfile pntr fwa	fwa dayfile dump buffer			†2		no. excess dayfiles		DFPP
004	fwa FNT	lwa+1 FNT	▨		fwa job control area				FNTP,JBCP
005	fwa EST	lwa+1 EST	lwa+1 ms equipment		fwa ECS/PP buffer				ESTP
006	fwa mass storage allocation		fwa user library directory				▨		LBDP, MSAP
007	fwa CPU library directory		fwa COS format CPU lib directory				▨	†3	CLDP
010 ⋮ 017	installation area								INOL,INSL ⋮ IN7L
020	▨						CMR size /100		CMRL
021	system name					▨		†4	
022	▨			job sequence number counter				▨	JSNL
023	▨	avail ECS 1000B blocks	▨				available mem/100B		ACML,AECL
024	job scheduler	CPU recall	PP/auto recall		job activity		job switch		MSCL
025	†5	ECS first user track	user 1000B word ECS blks		ECS RA/1000B for CPO		ECS FL/1000B for CPO		ECRL
026	▨			julian date (yyddd)					JDAL
027	▨		packed date (yr-1970,mo,da,hr,mn,sc)						PDTL
030	time of day (Δhh. mm. ss.)								TIML
031	date (Δyy/mm/dd.)								DTEL
032 ⋮ 035	system title line								
036 037	system version name								
040	▨						scheduler cy. intvl.		JSCL
041	←†6 ▨	1CK recall time			1SP recall time				

Ref	Bit No.	Description
† 1	23-20	Unused.
	19-18	CDC CYBER 176 CPU type: 0 = Not a CDC CYBER 176. 1 = CDC CYBER 176 Model A. 2 = CDC CYBER 176 Model B. 3 = CDC CYBER 176 Model C.
	17	Set if 2x PPs are selected.
	16	Set if machine type is CDC CYBER 170.
	15	Set if CMU is present.
	14	Set if CEJ/MEJ option is available.
	13	Set if CPU0 has an instruction stack.
	12	Set if CPU1 is present.
† 2	23-12	Nonzero if dayfile dump is disabled.
† 3	5-0	ACCFAM FL/100.
† 4	5-3	LIBDECK number.
	2-0	Recovery mode.
† 5	59-48	Reserved.
† 6	59	Scheduler active flag.

	59	47	35	23 17	11 0	
042	†1					IPRL
043	†2					SSTL
044	TELEX/IAF	EXPORT/ IMPORT	BATCHIO	MAGNET	TAF	SSCL
045	STIMULATOR	NETWORK INTER PROC	RBF	CDCS	MCS	
046	MASS STORAGE CONTROL	TRANSACTION STIMULATOR	reserved			
047	reserved					
050	reserved				IR addr next PPU	PPAL
051	idle time					
052 053 054	load code for MS error processors					MSEL
055 056	reserved					
057	ctrl point for move	internal to MTR				CMCL
060	←†3 ▨	▨	CPO ctrl pt assig	CPO exchange address		ACPL
061	←†4 ▨	▨	CP1 ctrl pt assig	CP1 exchange address		
062	▨	▨	▨	address of PPO exchange package		PXPP
063	first word of PP exchange package					
064 065	reserved					
	zeros					ZERL
067 : 075	reserved					
076	reserved			CPUMTR exchange address for MTR		MTRL
077	EQ CPSL			PS 0		CPSL

Ref	Bit No.	Description
†1	59-54	Index for CPU1 multiplier.
	53-48	Index for CPU0 multiplier.
	47-36	Secondary rollout sector threshold.
	35	Keypunch mode (0=O26, 1=O29).
	34-25	Unused.
	24	System character set mode (0=63, 1=64 character set).
	23-12	Assumed conversion mode (2=ASCII/USASI, 3=EBCDIC).
	11-6	Assumed 9-track tape density (3=800, 4=1600, 5=6250).
	5	Assumed tape type (7-track=0, 9-track=1).
	4-0	Assumed 7-track density (1=200, 2=556, 3=800).
†2	59-54	Reserved for CDC use.
	53	Disable user ECS.
	52	Disable PF validation.
	51-50	Disable MS validation.
	49	Ignore USER statement.
	48	Disable account verification.
	47	Disable BATCHIO.
	46	Disable TELEX/IAF.
	45	Disable EI200.
	44	Disable MAGNET.
	43	Disable TAF/TS.
	42	Disable removable device checking.
	41	Disable queue protect.
	40	Disable secondary user statements.
	39	Disable SCP facility.
	38	Disable TAF.
	37	Disable NAM.
	36	Disable RBF.
	35	Disable subcontrol points.
	34	Disable MCS.
	33	Disable CDCS.
	32-15	Reserved for CDC use.
	14	ENGINEERING switch.
	13	Console initial lock status.
	12	DEBUG switch.
	11-0	Reserved for installation use (local).
†3	59	Set if CPU0 is off.
†4	59	Set if CPU1 is off.

	59	47	35	23 17	11 0	
100	CH0	CH1	CH2	CH3	CH4	CTIL †1
101	CH5	CH6	CH7	CH10	CH11	
102	CH12	CH13	CH14	CH15	CH16	
103	CH17 (unused)	CH20	CH21	CH22	CH23	
104	CH24	CH25	CH26	CH27	CH30	
105	CH31	CH32	CH33	CH34 (unused)	CH35 (unused)	
106	seconds		milliseconds			RTCL
107	reserved					
110	†2					PFNL
111	†3		▨			
112	†4					SCRL
113	4	3	2	1	0	S16L †5
114	9	8	7	6	5	
115	14	13	12	11	10	
116	▨			16	15	
117	4	3	2	1	0	S36L †6
120	9	8	7	6	5	
121	14	13	12	11	10	
122	▨			16	15	
123	MID	†7			machine index	MMFL
124	reserved					
125	reserved					
126	reserved			flag register		EFRL
127	†8					INWL
130	reserved		MXN time	worst case MTR cycle time	current MTR cycle time	SD0L
131	count of ECS moves		count of CM moves			SD1L
132	rollout count		count of sectors rolled			SD2L
133	reserved	user commits + time slice with output		count of time slices		SD3L
134	reserved		jobs in recall due to PP priority exchanges			SD4L
135 ● ● ● 162	reserved					
163 ● ● ● 177	DSD - 1DS communication area					

<table>
<tr><th>Ref</th><th>Bit No.</th><th>Description</th></tr>
<tr><td>† 1</td><td>--</td><td>Channel status table; one byte per channel, each with the following bit descriptions.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>11</td><td>Set if channel requested.</td></tr>
<tr><td>10-7</td><td>PP number of requesting PP.</td></tr>
<tr><td>6</td><td>Set if channel not available.</td></tr>
<tr><td>5-0</td><td>PP assigned.</td></tr>
</table></td></tr>
<tr><td>† 2</td><td>59-56</td><td>Reserved.</td></tr>
<tr><td></td><td>55</td><td>Total PF system interlock.</td></tr>
<tr><td></td><td>54</td><td>Request total PF system interlock.</td></tr>
<tr><td></td><td>53-48</td><td>PF activity count.</td></tr>
<tr><td></td><td>47-18</td><td>Reserved.</td></tr>
<tr><td></td><td>17-12</td><td>Default family equipment number.</td></tr>
<tr><td></td><td>11-6</td><td>Alternate family count.</td></tr>
<tr><td></td><td>5-1</td><td>Reserved.</td></tr>
<tr><td></td><td>0</td><td>Word interlock.</td></tr>
<tr><td>† 3</td><td>59-48</td><td>Seconds left until label check.</td></tr>
<tr><td></td><td>47-36</td><td>Seconds left until devices checkpointed.</td></tr>
<tr><td>† 4</td><td>59</td><td>Set to inhibit MTR from calling 1MB for S/C register error processing.</td></tr>
<tr><td></td><td>58</td><td>Set if error processing ignored at deadstart.</td></tr>
<tr><td></td><td>57</td><td>Set to allow MTR to accept DSRM function for emergency step from 1MB, and to prevent DSD from allowing UNSTEP command to be entered.</td></tr>
<tr><td></td><td>56</td><td>Set to indicate MTR has set step mode on request from 1MB (emergency step).</td></tr>
<tr><td></td><td>55-36</td><td>Unused.</td></tr>
<tr><td></td><td>35-24</td><td>Real-time clock from RTCL, in seconds/1000_8, at which the last threshold count or time interval was exceeded for single SECDED errors.</td></tr>
<tr><td></td><td>23-12</td><td>SECDED count.</td></tr>
<tr><td></td><td>11-0</td><td>Threshold count.</td></tr>
</table>

Ref	Bit No.	Description
†5	--	The channel 16 S/C register contents, words 0 through 16 (bits 0-203).
†6	--	The channel 36 S/C register contents, words 0 through 16 (bits 0-203).
†7	47-42	Reserved.
	41-36	Equipment number of link device.
	35	Set if this machine has DATI recovery interlock.
	34-30	Unused.
	29-24	Count of devices with initialize pending that have not been checkpointed.
	23-20	Machines active.
	19-16	Machines down.
	15-12	Machine mask.
†8	59-15	Unused.
	14	Disable priority evaluation.
	13	Disable job scheduler.
	12	Disable autoroll.
	11-2	Unused.
	1	Fatal mainframe error flag.
	0	System control point (SCP) subsystem abort interlock.

CONTROL POINT AREA

	59	47	41	35	29	23	17	11	5 0	
000 • • • 017	exchange package area									
020	†1	error flags		activity count		RA/100B		FL/100B		STSW
021	job name						job orgn	operator equipment		JNMW, OAEW
022	CPU priority	queue priority		†2		▨	CPUs allowable			JCIW
023	CM residence time limit			†3	CPU time slice limit					TSCW
024	time entered X status									CPCW
025	†4	reserved				ECS RA/1000B		ECS FL/1000B		ECSW, CPIW
026	PP recall register									RLPW
027	†5							snse swchs	▨	SNSW
030 • • • 034	message 1 area									MS1W
035 036 037	message 2 area									MS2W
040 • • • 047	installation area									INOW • • • IN7W
050	†6		SRU accumulator (micro units*10)							ACTW, SRUW
051	**CP accumulator**									CPTW
052	MS accumulator		MT accumulator			PF accumulator				IOAW
053	M13=M1*M3		M14=M1*M4			▨	adder accumulator			MP1W, ADAW
054	M1* 1000		M12=M1*M2			reserved				ACTWE, MP2W
055	←†7 CPM (SRU=SRU + CPM*CP)				IOM (SRU=SRU + IOM*IO)					MP3W
056	SRU account block limit		computed SRU job step limit							STLW
057	reserved	SRU job step limit			SRU at beginning of job step					SRJW
060	reserved	CP time job step limit			CP time at beginning of job step					CPJW

Ref	Bit No.	Description
† 1	59	CPU W status.
	58	CPU X status.
	57	CPU auto recall (I status).
	56	CPU subcontrol point active status.
	55-54	Unused.
	53	Job advancement flag.
	52-48	Number of PPs assigned to job.
† 2	35-33	CPU status for rollout.
	32-25	Unused.
	24	Set if rollout is requested.
† 3	35	Set if CPU time slice is active.
	34-30	Queue control (0=input, 1=rollout).
† 4	59-51	Job control flags (reserved).
	50	Return private user files.
	49	Set privacy ID on new files.
	48	Preserve ECS over job steps.
	47	FNT interlock.
† 5	59	Reserved.
	58	O26/O29 punch mode.
	57	Set if OVERRIDE required to drop job.
	56-36	Unused.
	35-24	Reserved for installation use.
	23-15	Reserved.
	14	Subsystem idledown flag.
	13	NOGO flag.
	12	PPU pause flag.
† 6	Limit flags:	
	59	Time validation limit.
	58	Time limit.
	57	SRU validation limit.
	56	SRU limit.
	55	Control statement limit.
	54-48	Reserved.
	Overflow flags:	
	47	MS accumulator.
	46	MT accumulator.
	45	PF accumulator.
	44	AD accumulator.
	43-42	Reserved.
† 7	59	Disable SRU accumulation if set.

	59	53	47	35	29	23	17	11	0	
061	†1									FPFW
062	†2					rollin FL		FL increase request		FLCW
063	†3					rollin ECS FL		ESC FL increase req		ELCW
064	†4									SSCW
065	TXOT	list of files address		TTY interrupt address †5			output pointer			TXSW,TIOW, TIAW, LOFW
066	auxiliary pack name						†6			PFCW
067	user number †9						†7 user index			UIDW
070	†8			†11 terminal input pointer			error exit †10 return address			EECW,TINW
071	input FST		primary FST	▨	event descriptor			rollout time		TFSW,TERW
072	†12		control statement count			next statement index		limit index		CSPW
073	†13	eq num	first track	current track		current sector		half sector flag		CSSW
074	job sequence number			control statement address (TCS)			demand file random index			RFCW
075	reserved		†14							ALMW
076	reserved		dayfile msg count	control stmt count		†15	mass storage PRU count			ACLW
077	each bit has a special meaning									AACW
100	buffer 0 length		buffer 0 address		buffer 1 length		buffer 1 address			ICAW
101	special entry point word †16									SEPW
102	system processor call word †17									SPCW
103	EFG	R1G		CCL data			reserved			JCDW
104	EF	R3		R2			R1			JCRW
105	†18	input buffer address		right screen buffer address			left screen buffer address			DBAW
106 107 110	loader control words †19									LB1W LB2W LB3W
111	▨		†20				FWA of dump			PPDW
112	reserved							†21		SSOW
113	computed CP job step limit									CPLW
114 • • • 127	reserved									
130 • • • 177	control statement buffer									CSBW

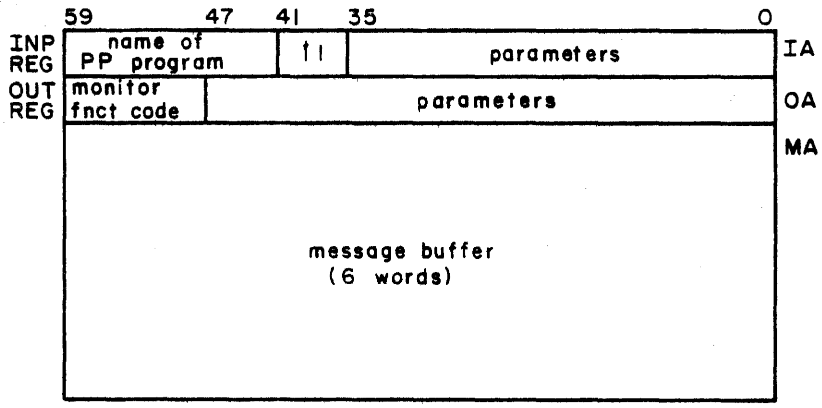
Ref	Bit No.	Description
†1	59	Set when first charge processed.
	58	Set if second entry in level-3 block.
	57-48	Reserved.
	47-36	SRU validation limit.
	35-24	FNT ordinal of PROFILE file.
	23-12	Track of level-3 block.
	11-0	Sector of level-3 block.
†2	59-48	Maximum field length (MFL) for current job step.
	47-36	Initial running field length; always less than or equal to MFL (value of zero indicates system field length control).
	35-24	Maximum field length for entire job; MAX FL is upper bound on MFL.
†3	59-48	Maximum ECS field length (MFL) for current job step.
	47-36	Initial running ECS field length; always less than or equal to MFL (value of zero indicates system ECS field length control).
	35-24	Maximum ECS field length for entire job; MAX FL is upper bound on MFL.
†4	59-48	Rollout indicators (one bit per subsystem) indicating the user job is a candidate for normal rollout.
	47-0	Connection indicators (four bits per subsystem) representing particular subsystem the user job is communicating with.
†5	35-17	Previous error flag value if bit 58 set in word EECW indicating extended RPV mode.
†6	17-12	Family EST ordinal.
	11-0	Indexes into tables of limits.
	11-9	Limit for size of direct access files.
	8-6	Limit for number of permanent files.
	5-3	Limit for cumulative size of indirect access files.
	2-0	Limit for size of indirect access files.
†7	17	Set if charge statement is required.

Ref	Bit No.	Description
†8	59	No exit flag.
	58	Extended RPV mode.
	57	Interrupt handler in progress flag (extended RPV mode only).
	56	Set if one-time error previously entered (extended RPV mode only).
	55-48	Unused.
	47	For nonextended RPV mode, set if bits 46-36 are error flag instead of reprieve error option.
	46-36	Error flag or reprieve error option for nonextended RPV mode.
	47-36	Mask bits for extended RPV mode.
†9	17	Job reprieved.
†10	17-0	RPV parameter block address (extended RPV mode only).
†11	30	Valid event descriptor present.
†12	59-54	Job class.
	53-48	Reserved.
	47	Set if EOR is on control statement file.
†13	59	Set if information is for INPUT file.
	58	Skip to EXIT flag.
	57-54	Unused.

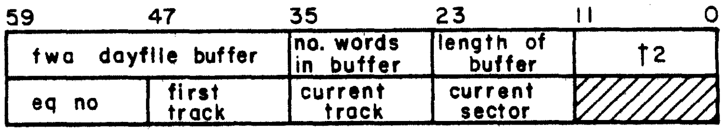
Ref	Bit No.	Description
†14	47-45	Magnetic tapes.
	44-42	Removable packs.
	41-39	Deferred batch jobs.
	38-36	Local files.
	35-30	Time limit.
	29-24	SRU limit.
	23-18	Field length.
	17-12	ECS field length.
	11-6	Lines printed.
	5-0	Cards punched.
†15	23-18	Disposed output count.
†16	59	Set indicates presence of entry points.
	58-54	Reserved.
	53	Set if ARG= entry point present.
	52	Set if DMP= entry point present.
	51	Set if SDM= entry point present.
	50	Set if SSJ= entry point present.
	49	Set if VAL= entry point present.
	48	Set if SSM= entry point present.
	47-36	Reserved.
	35	Restart flag.
	34	Reserved.
	33	Suppress DMP= if control statement call.
	32	Create DM* file only flag.
	31	Dump FNTs with control point area.
	30	Leave DM* file unlocked.
	29-18	DMP= FL/100 (if field is 0, dump entire FL).
	17-0	SSJ= parameter block address.
†17	For input:	
	59-42	Entry point if RA+1 request, 770000B if control statement call.
	41	Special program request active (1AJ only).
	40	Clear RA+1 upon completion.
	39	If set, parameter list is in bits 35-0; if clear, address of parameter list is in bits 17-0.
	38	Does not start CPU at completion of control statement call (1AJ only).
	37	DMP= initiation in progress.
	36	Unused.
	35-0	Refer to description of bit 39.
	For output:	
	59-36	Unused.
	35-24	Status return.
	23-0	Unused.

Ref	Bit No.	Description
†18	59	Disable dumps.
	58-56	Unused.
	55	ECS common memory manager flag.
	54	CM common memory manager flag.
†19	LB1W:	
	59	Use default map options if not set.
	58	Reserved.
	57	Local map option X.
	56	Local map option E.
	55	Local map option B.
	54	Local map option S.
	53	Reduce flag.
	52-36	Reserved.
	35-24	CDC CYBER Interactive Debug control byte.
	23-0	Global library set indicators (6-bit fields): 00 End of library set. 01-76 LBD ordinal of system library. 77 User library; logical file name of first user library in LB3W; logical file name of second user library in LB2W.
	LB2W, LB3W:	
	59-0	Either logical file name of second (LB2W) or first (LB3W) user library, or a collection of 6-bit global library set indicators.
†20	47-36	ECS FL of program making DMP= call.
	35-24	Field length of program making DMP= call.
	23-18	Dump word count.
†21	12	Swap out (SF.SWPO) in progress.
	11-0	Subsystem outstanding connection count.

PP COMMUNICATION AREA



DAYFILE BUFFER POINTERS

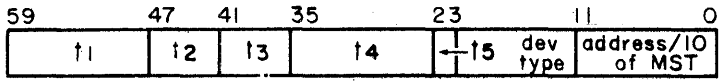


Ref	Bit No.	Description
†1	41	Set if called with auto recall.
	40-36	Control point assignment.
†2	11-0	Interlock byte (0 = no dump in progress, 1 = dump in progress).

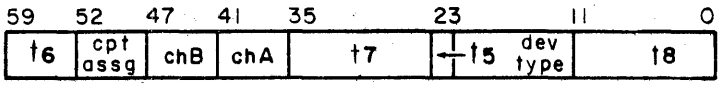
CENTRAL MEMORY TABLES

Equipment Status Table (EST) Formats

Mass Storage Device



Nonmass Storage Device (3000 Type Equipment)



Ref	Bit No.	Description
†1	59	Set to indicate mass storage device.
	58	Set if device has copy of system.
	57	Set if shared device.
	56	Set if removable device.
	55	Set if 844/885 disk type equipment.
	54	Set if device is not currently available for access.
	53	Set if equipment is down.
	52-48	Reserved.
†2	47	Channel down bit.
	46-42	Alternate channel.
†3	41	Channel down bit.
	40-36	Primary channel.
†4	For 844/885 disk type equipment:	
	35-24	Zero.
	For other equipment types:	
	35-33	Physical equipment number.
	32-30	Zero.
	29-27	Device selection for connect code.
	26-24	First physical unit for device.
†5	23	ON/OFF flag (set if access not allowed).

Ref	Bit No.	Description
† 6	59	Unused.
	58	Allocatable device.
	57-56	Unused.
	55	Set if 580 PFC printer.
	54	Set if V carriage control processed.
	53	Set if equipment is down.
† 7	For unit record equipment:	
	35-24	Forms code.
	For other equipment:	
	35-30	Channel D.
	29-24	Channel C.
† 8	For magnetic tape equipment:	
	11-9	Equipment number.
	8-4	Flags: 01 GCR (1600/6250) tape unit. 02 Disable block-ID (66x only). 04 Reserved. 10 67x tape unit. 20 66x tape unit.
	3-0	Unit number.
	For other equipment types:	
	11-9	Controller number.
	8-6	Print train (if applicable).
	5-0	Unit number.
	For unit record equipment:	
	5-0	ID number.

Equipment Codes

Code	Description
CP	Card punch (3446/3644-415).
CR	Card reader (3447/3649-405).
DE	Extended core storage.†
DI-n	Disk storage subsystem (7x54-844-21).
DJ-n	Disk storage subsystem (7x5x-844-4x/44).
DK-n	Disk storage subsystem (7154-844-21).
DL-n	Disk storage subsystem (715x-844-4x).
DM-n	Disk storage subsystem (7155-885).
DP	Distributive data path to ECS.
DQ-n	Full-track disk storage subsystem (7155-885).
DS	Display console.
LP	Line printer.
LR	Line printer (580-12).
LS	Line printer (580-16).
LT	Line printer (580-20).
MS	Mass storage device.
MT	Magnetic tape drive (7-track).
NE	Null equipment.
NP	255x Host Communications Processor.
NT	Magnetic tape drive (9-track).
ST	Remote batch multiplexer (6676 or 2550-100).
TT	Time-sharing multiplexer (6676, 6671, or 2550-100).

---

† ECS subequipment values exist in associated MST. The values are in word DILL (byte 3) and further define the type of ECS equipment.

File Name/File Status Table (FNT/FST) Entry

File in Input Queue

59	53	47	35	23	17	11	5 0
job name					job org	type INFT	←†1
id code	eq no	first track	binary card sequence no	field length		queue priority	

File in Print Queue

59	53	47	35	17	11	5 0
job name				job org	type PRFT	←†1
†2	eq no	first track	†3		queue priority	

File in Punch Queue

59	53	47	35	17	11	5 0
job name				job org	type PHFT	←†1
†2	eq no	first track	†3		queue priority	

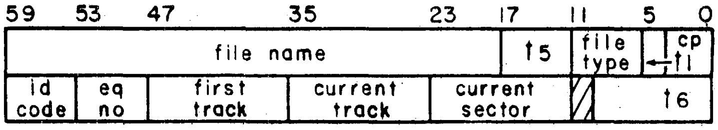
File in Rollout Queue

59	53	47	35	23	17	11	5 0
job name					job org	type ROFT	←†4
id code	eq no	first track	ECS FL/1000B	field length		queue priority	

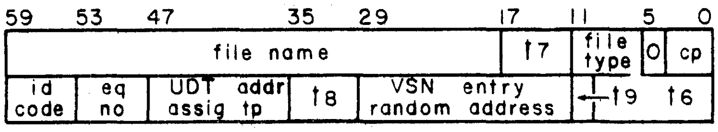
File in Timed/Event Rollout Queue

59	53	47	35	23	17	11	5 0
job name					job org	type TEFT	←†4
event des	eq no	first track	event descriptor	field length		rollout time pd	

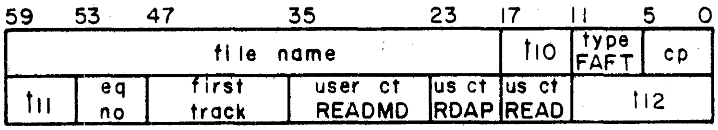
Mass Storage Files
Not in Input, Print, Punch, or Rollout Queue



Magnetic Tape Files



Fast Attach Permanent Files



Ref	Bit No.	Description
†1	5	Set if system sector contains control information.
†2	59-57	Device selection field.
	56-54	External characteristics.
†3	35-33	Forms code.
	32-12	Terminal identification (TID).
†4	5	Set if user job has subsystem connection (either long term connection or wait response).
†5	17	Unused.
	16	Set if extend-only file.
	15	Set if alter-only file.
	14	Set if execute-only file.
	13	Unused.
	12	Write lockout.
†6	10	Unused.
	9	Indicates the track interlock status of LIFT files (mass storage only).
	8	Set if file is opened.
	7	Set if file is written since last open.
	6	Set if file is written on.
	5-4	Unused.
	3-2	Read status (0 = incomplete read, 1 = EOR, 2 = EOF, 3 = EOI).
	1	Set if last operation write.
	0	Clear if busy status.

Ref	Bit No.	Description
†7	17-14	Unused.
	13	Set if opened.
	12	Write lockout.
†8	35-32	Data format: 0 I 1 SI 2 F 3 S 4 L
	31-30	Reserved.
†9	11	Set if labeled tape.
†10	17	Unused.
	16	Set if modify.
	15	Set if append.
	14	Set if execute.
	13	Set if write.
	12	Set if read.
†11	59-54	Fast attach entry index in ECS (if globally fast attach), 0 if local fast attach file.
†12	11-9	Write attach mode (7 = write, 3 = modify, 1 = append).
	8-1	Unused.
	0	Clear if busy status.

File Types

Files in Queues

Type	Value	Description
INFT	0	Input.
ROFT	1	Rollout.
PRFT	2	Print.
PHFT	3	Punch.
TEFT	4	Timed/event rollout.

Special Queue Files

Type	Value	Description
S1FT	5	Special file type 1.
S2FT	6	Special file type 2.
S3FT	7	Special file type 3.

Other Files

Type	Value	Description
LIFT	10	Library.
PTFT	11	Primary terminal.
PMFT	12	Direct access permanent file.
FAFT	13	Fast attach file.
SYFT	14	System.
LOFT	15	Local.

Job Origin Codes

Type	Value	Description
SYOT	0	System.
BCOT	1	Local batch.
EIOT	2	Remote batch.
TXOT	3	Time-sharing.
MTOT	4	Multiterminal.

Mass Storage Allocation (MSA) Area

	59	47 0
000	last temp eq	temporary devices†
001	last input eq	input file devices†
002	last output eq	output file devices†
003	last rollout eq	rollout file devices†
004	last dayfile eq	user dayfile devices†
005	last primary eq	primary file devices†
006	last local eq	local file devices†
007	last LGO eq	LGO file devices†
008	last secondary rollout eq	secondary rollout file devices†

---

†Bit 47-eq is set for each equipment with the allocation type selected.

Mass Storage Table (MST)

	59 51	47 40	35	23	17	11	5 0	
000	†1	▨	TRT length	†2		no. avail. tracks		TDGL
001	†3	user ECS first track	file count	IQFT track		†4		ACGL
002	ECS address of MST/TRT		ECS MST/TRT update cnt				†5	SDGL
003	1st track IAF	label track	permits track	no. catalog tracks		DAT track		ALGL
004	family or pack name				DN	▨	†6	PFGL
005	user number for private pack				†7			PUGL
006	†8		driver name	0		sector limit		MDGL
007	▨							R1GL
010	installation area (global)							ISGL
011	▨							I2GL
012	activity count	unit interlocks	current position	MTR internal		ECS error #		DALL
013	†9							DILL
014	DAYFILE track	ACCOUNT track	ERRLOG track	system table track		†10		DULL
015	†11			user count		†12		STLL
016	†13							DDLL
017	installation area							ISLL

Ref	Bit No.	Description
†1	59-48	Number of tracks on device.
†2	23	NOS format MST.
	22-12	First available track word pointer.
†3	59	CTI present.
	58	System deadstart file present.
	57-52	Reserved.
	51-48	Global interlock (machine mask).
†4	11	Redefinition requested flag.
	10-7	Redefinition reply bits (machine masks).
	6	Set if sector of local areas is present.
	5	Unload (all machines).
	4	Device error idle status: 0 No error. 1 Error detected on device.
	3-0	Permanent file utility active (machine mask).

Ref	Bit No.	Description
† 5	5-4	Reserved.
	3-0	Interlock (machine mask).
† 6	5-3	Relative unit in multiunit device.
	2-0	Number of units in multiunit device.
† 7	17	Catalog track contiguous with label track.
	16	Catalog track overflow (O).
	15-8	Secondary device mask.
	7-0	Device mask.
† 8	59	Removable (R).
	58	Auxiliary permanent file device (X).
	57	Sixteen-word PFC device.
	56	Device last checkpointed on MMF system (in label section only).
	55-48	DAT entry index.
	47	Half track status (1=half, 0=full)
	46	Release reservation when channel released.
	45	Reserved.
	44-36	Single-unit sector limit.
† 9	59-48	Mass storage allocation flags.
	47	715x controller present on second channel.
	46-42	Second channel in CMRDECK in definition of EQ.
	41	715x controller present on first channel.
	40-36	First channel in CMRDECK in definition of EQ.
	35-24	Unused.
	23-22	Reserved.
	21	Maintenance mode set (ECS).
	20-18	Memory type: 0 No CPU. 1 ECS I. 2 ECS II. 3 LCME. 4-7 Reserved.

Ref	Bit No.	Description
	17-15	CPU type: 0 No CPU path. 1 ECS. 2 LCME. 3-7 Reserved.
	14-12	PP path type: 0 No DDP. 1 DC145 parity enhanced DDP. 2 DC135 DDP. 3-7 Reserved.
	11-6	Unused.
	5-0	Algorithm index for 844/885 disk monitor function.
† 10	11	Family idle down status.
	10-0	Family activity count.
† 11	59	Format pack (844/885 disk equipment).
	58	Half/full track initial requeues.
	57	Initialize permanent files (I).
	56	Initialize IQFT (I).
	55	Initialize DAYFILE (I).
	54	Initialize ACCOUNT (I).
	53	Initialize ERRLOG (I).
	52	Initialization (HT/FT) (I).
	51	Unloaded in this machine (L).
	50	Checkpoint requested (C).
	49	TEMP (T).
	48	Alternate system device (A).
	47-42	Reserved.
	41-36	Error status.
	35-24	A 2-character machine identification.
† 12	11-6	Multiple equipment link.
	5-3	Original number of units.
	2	Device in use.
	1	Local utility interlock.
	0	Local area interlock.
† 13	59	Redefinition in progress (drive reserved).
	58	Null equipment indicator.
	57-54	Reserved.
	53-48	Number of units minus 1.
	47-0	Unit list, ordered right to left, 6 bits per unit.

Track Reservation Table (TRT)

Word Format

Ref	Bit No.	Description
†1	11-8	Each bit set indicates corresponding byte (0 through 3) is first track of a preserved file.
	7-4	Track interlock bits.
	3-0	Track reservation bits.

Track Link Byte (Format 1)

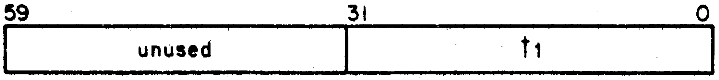
Bit	Contents
11	Set.
10-0	Next track in track chain.

Track Link Byte (Format 2)

Bit	Contents
11	Clear.
10-0	End of chain (EOI sector in file).

Machine Recovery Table (MRT)

Word Format



Ref	Bit No.	Description
†1	31-0	Each bit represents one logical track (bits 10-5 of the logical track number denote the word number in the MRT and bits 4-0 are the bit numbers within the word).

The meaning of the MRT bit depends upon the state of the track interlock bit in the TRT.

Track Interlock Bit	MRT Bit	Description
0	0	Track is not interlocked or it is local to another machine.
0	1	First track of a file is local to this machine.
1	0	Track is interlocked by another machine.
1	1	Track is interlocked by this machine.

Job Control Area (JCB)

One for each origin type and job class

59	47	35	23	11 0	
in. queue priority	lower bound	upper bound	priority age intvl	cur. intvl count	INQT
in. queue priority	lower bound	upper bound	priority age intvl	cur. intvl count	ROQT
in. queue priority	lower bound	upper bound	priority age intvl	cur. intvl count	OTQT
init. CPU priority	CPU time slice	CM time slice	▨	▨	SVJT
max jobs or users	max FL any job	max FL all jobs	max ECS FL any job	max ECS FL all jobs	
†1	reserved				PFCT
reserved					ETB
▨	▨	▨	▨	▨	

Ref	Bit No.	Description
†1	59-48	Index into tables of limits.
	59-57	Index a table of limits for size of each direct access file.
	56-54	Index a table of limits for number of permanent files.
	53-51	Index a table of limits for cumulative size of indirect access files.
	50-48	Index a table of limits for size of each indirect access file.

Libraries/Directories

Resident CPU Library (RCL)

Type OVL

59	17 0
program name	length (links to next program)

Type ABS

59		17 0
1	▨	length (links to next program)

Resident PPU Library (RPL)

59	41	35	23	11 0
package name	▨	load address	▨	length (links)

PPU Library Directory (PLD)

CM Resident

59	41	35	23	11 0
package name	1	RPL address	length	load address

Non-CM Resident

59	41	35	23	11 0
package name	↑1	track	sector	load address

CPU Library Directory (CLD)

Type OVL

59	47	23	17	11	5 0
program name			↑2		▨
▨	↑3	track		sector	

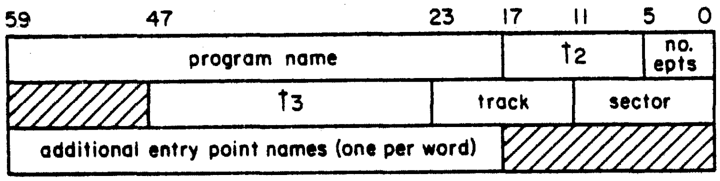
Type ABS

59	47	23	17	11	5 0
name of first entry point			↑2		no. epts
↑4	↑3	track		sector	
additional entry point names (one per word)			▨		

Type PROC

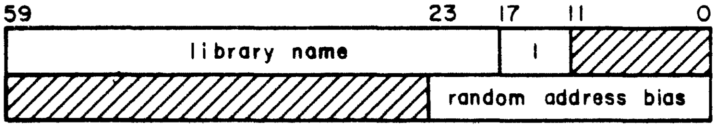
59	23	17 0
procedure name		↑5 ▨
▨	random address bias	

Type REL



User Library Directory (LBD)

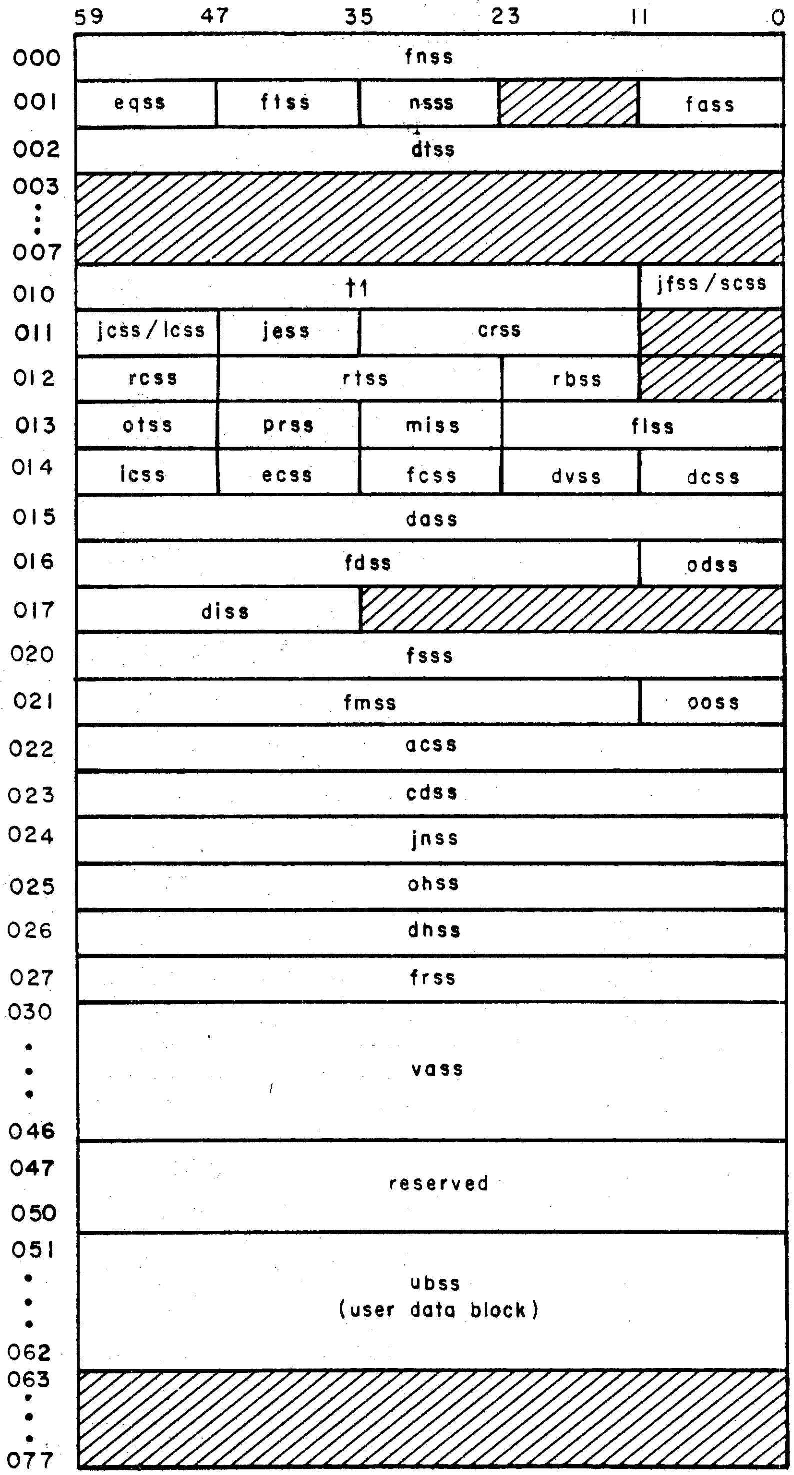
Type ULIB



Ref	Bit No.	Description
† 1	41-36	Alternate device or system device equipment number.
† 2	17-15	Unused.
	14	Relocatable record flag.
	13	NOS/BE record flag.
	12	Unused.
	11-6	Alternate device equipment number.
†3	47-24	If program is CM resident, field contains the absolute address in RCL. If program is assigned to alternate system device, field has mass storage address of copy on system device.
†4	59-48	FL required (use of bits 59 and 58 indicate MFL= entry point).
†5	17	Set if CCL procedure.

SYSTEM SECTOR FORMAT

Standard Format



† 1 For print/punch files, pfss (bits 47-36), rass (bits 35-12); for input files, jsss (bits 59-36), bits 35-24 unused, jtss (bits 23-12).

† 2 For input files, bits 59-18 are defined as terminal name (tnss).

The following apply to all system sectors.

fnss	FNT entry.
eqss	Equipment number.
ftss	First track.
nsss	Next sector.
fass	Address of FST entry.
dtss	Last modification date and time (packed format).

The following apply to input files only.

jsss	Job sequence number.
jtss	Job time limit.
jfss	Job flags.
jcss	Job statement CM field length.
jess	Job statement ECS field length.
crss	Cards read.
tnss	Terminal name.

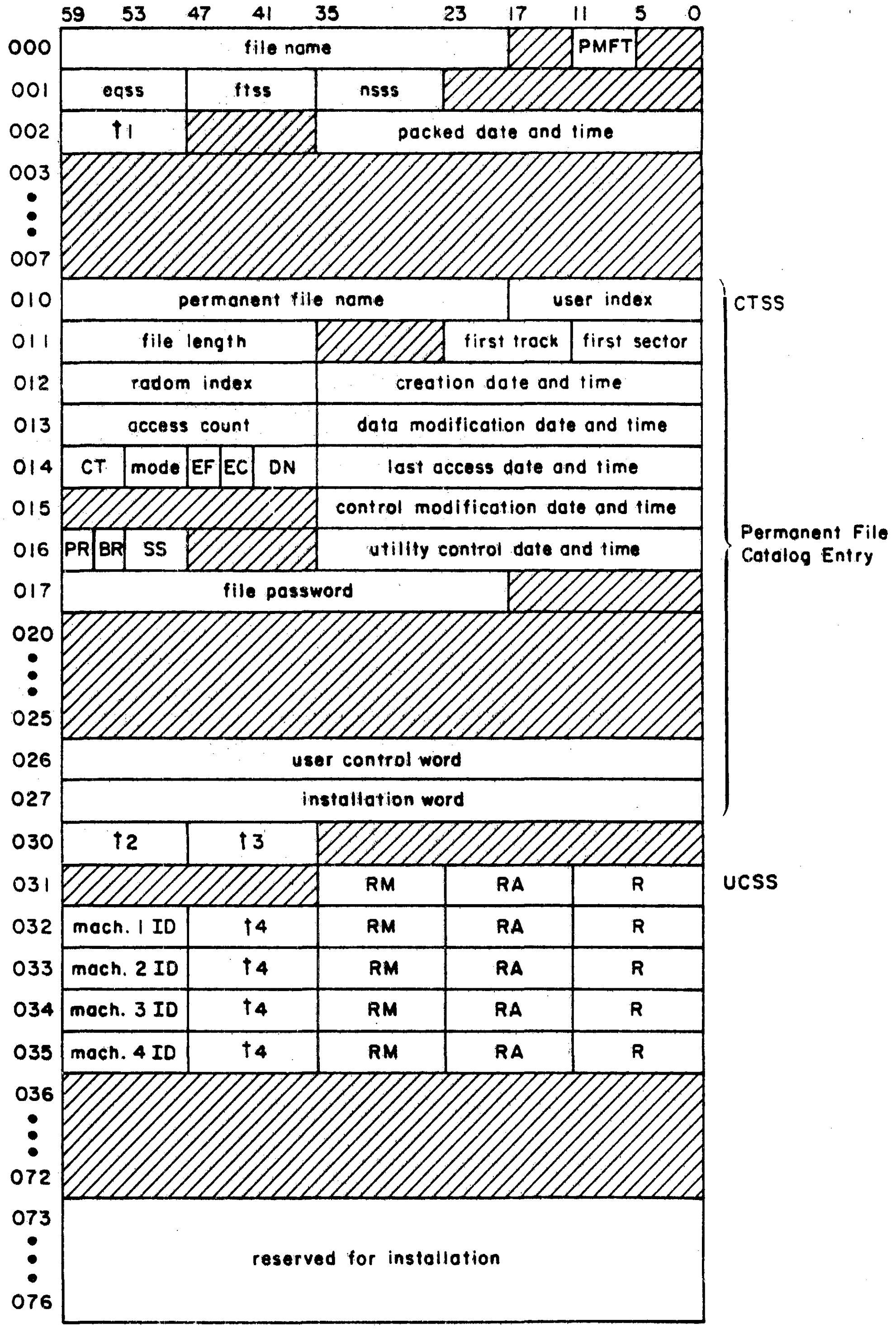
The following apply to print/punch files only.

pfss	Punch format.
rass	Random address of dayfile.
scss	Spacing code for 580 PFC support.
lcss	Lines or statement limit index.
rcss	Repeat count.
rtss	Random index.
rbss	Requeue number.

The following apply to all queued files.

otss	Origin type.
prss	Priority.
miss	Machine ID.
flss	File size (sectors/10_8).
icss	Internal characteristics.
ecss	External characteristics.
fcss	Forms code.
dvss	Device code.
dcss	NOS/BE device code.
dass	Destination user number.
fdss	Destination family name.
odss	Family ordinal of destination (future).
diss	Destination terminal identification (TID).
fsss	FST entry.
fmss	Family name of creator.
ooss	Family ordinal of creator (future).
acss	User number of creator.
cdss	Queued file creation date and time.
jnss	Job statement name.
ohss	Origination host name (future).
dhss	Destination host name (future).
frss	File routing control.
vass	Account file validation block.
ubss	User block.

Direct Access File System Sector Format



eqss	Equipment number.	
ftss	First track.	
ucss	Current user counts:	
	RM	READMD users.
	RA	READAP users.
	R	READ users.

Ref	Bit No.	Description
† 1	59-49	Zero.
	48	Set if enhanced EOI sector present.
† 2	59-54	Reserved.
	53	File has been purged.
	52	File can be shortened (W mode).
	51	File can be rewritten (W or M mode).
	50	Zero.
	49	File can be extended (W, M, or A mode).
	48	Zero.
† 3	47-36	Fast attach (40xx); upper bit set indicates file is in fast attach mode and lower 6 bits (41-36) contain index into ECS tables if file is global fast attach.
† 4	47-37	Zero.
	36	Local write flag (file attached in W, M, or A mode).

ECS Direct Access Chain

<table>
<tr><th></th><th>59</th><th>47</th><th>35</th><th>23</th><th>17</th><th>11</th><th>5 0</th></tr>
<tr><td>000</td><td colspan="4">** UECS.</td><td>▨</td><td>LIFT</td><td>▨</td></tr>
<tr><td>001</td><td>eqss</td><td>ftss</td><td colspan="5">▨</td></tr>
<tr><td>002</td><td colspan="2">▨</td><td colspan="5">dtss</td></tr>
<tr><td>003</td><td colspan="7">▨</td></tr>
<tr><td>004</td><td>mid1</td><td>ft1</td><td>ln1</td><td colspan="2">ra1</td><td colspan="2">lt1</td></tr>
<tr><td>005</td><td>mid2</td><td>ft2</td><td>ln2</td><td colspan="2">ra2</td><td colspan="2">lt2</td></tr>
<tr><td>006</td><td>mid3</td><td>ft3</td><td>ln3</td><td colspan="2">ra3</td><td colspan="2">lt3</td></tr>
<tr><td>007</td><td>mid4</td><td>ft4</td><td>ln4</td><td colspan="2">ra4</td><td colspan="2">lt4</td></tr>
</table>

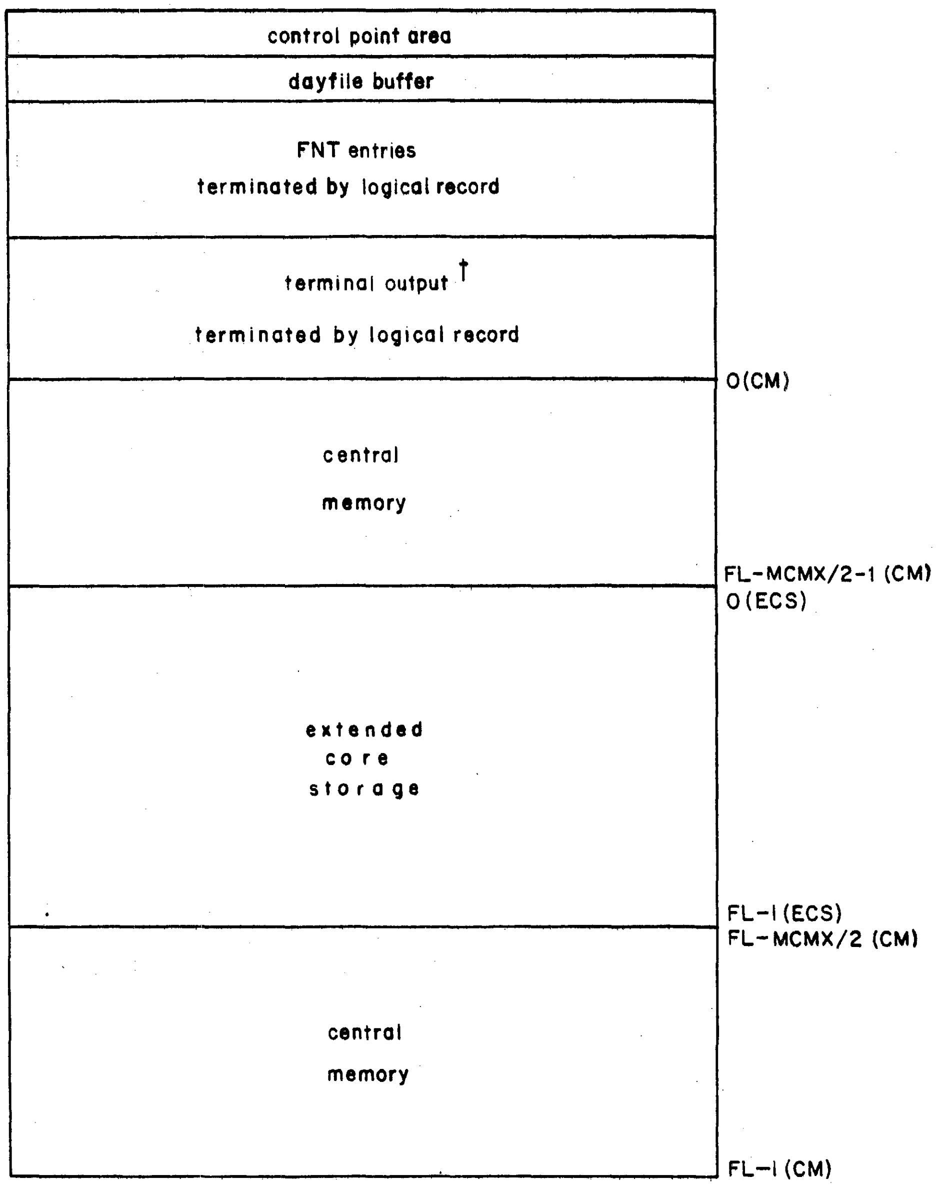
eqss	Equipment number.
ftss	First track.
dtss	Last modification date and time (packed format).
mid	Machine ID.
ft	First track of subchain.
ln	Length of ECS block.
ra	RAE of ECS block.
lt	Last track of subchain.

ROLLOUT FILE

System Sector

000	007	▨
010	011	dayfile buffer pointer
012	013	input file FNT entry
014	027	list of equipment assigned to job (terminated by zero word)
030	037	SSJ= parameter block
040	057	terminal table contents at last rollout
060	077	terminal table contents for recovery

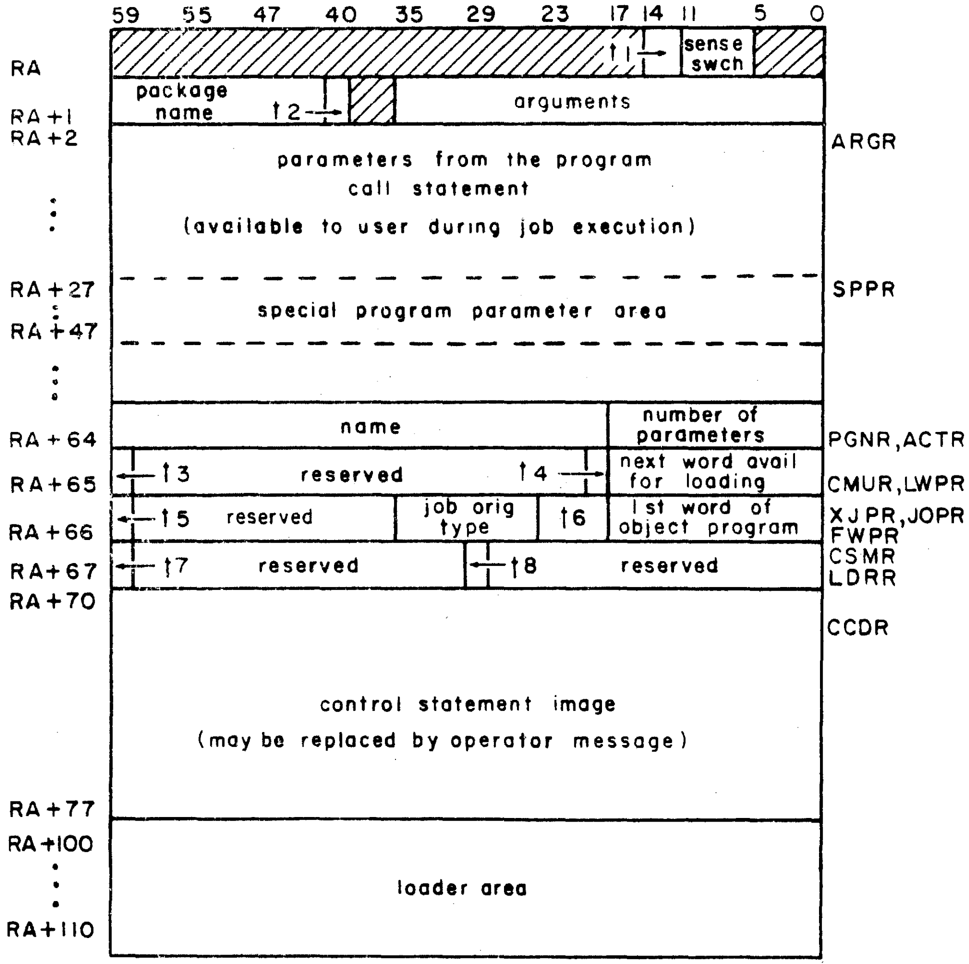
File Format



---

† This part of the rollout file is used only for TXOT jobs.

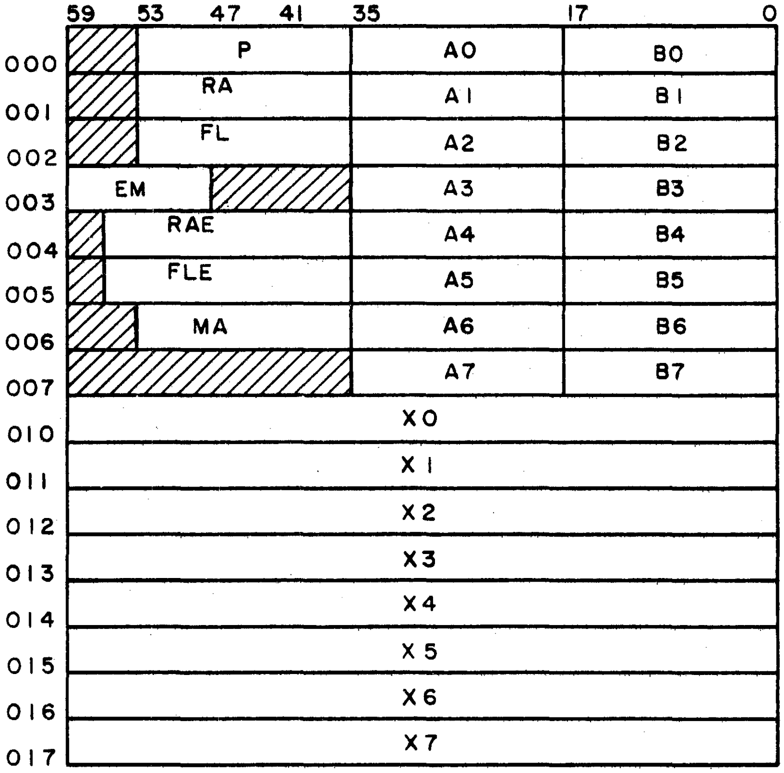
JOB COMMUNICATION AREA



Ref	Bit No.	Description
†1	14	CFO bit if console forced operator command is allowed.
	13	Subsystem idledown flag.
	12	Pause flag.
†2	40	Auto recall.
†3	59	Set if compare/move unit (CMU) is present.
†4	18	Set if load from system library.
†5	59	Set if CEJ/MEJ option is available.
†6	23-20	Reserved.
	19	Set if program called from DIS.
	18	RSS bit.
†7	59	Set indicates system is in 64-character set mode.
†8	29	Set if load has completed.

EXCHANGE PACKAGE AREA

Exchange package area for CDC CYBER 170 Series, Models 171, 172, 173, 174, 175, 720, 730, 750, and 760; CDC CYBER 70 Series, Models 71, 72, 73, and 74; and CDC 6000 Series Computer Systems.



Exchange package area for CDC CYBER 170 Series, Model 176 Computer Systems.

	59	53	35	17 0
000	▨	P	A0	B0
001	▨	RA	A1	B1
002	▨	FL	A2	B2
003	▨	PSD	A3	B3
004	▨	RAE	A4	B4
005	▨	FLE	A5	B5
006	▨	NEA (MA)	A6	B6
007	▨	EEA	A7	B7
010	X0			
011	X1			
012	X2			
013	X3			
014	X4			
015	X5			
016	X6			
017	X7			

The exchange package area fields apply to all NOS computer systems unless otherwise noted.

Field	Description
P	Program address.
Ai	Address registers.
Bi	Increment registers.
RA	Reference address for central memory.
FL	Field length for central memory.
EM†	Exit modes. An exit mode is selected by setting the appropriate bit and disabled by clearing the appropriate bit.

Bit	Description
59	CM data error.††
58	CMC input error.††
57	ECS flag register operation parity error.††
56-53	Not used.
52-51	Hardware error exit status bits.†††
50	Indefinite operand.
49	Operand out of range.
48	Address out of range.

PSD††††	Program status designator (PSD) register.

Bit	Description
53	Exit mode flag.
52	Monitor mode flag.
51	Step mode flag.
50	Indefinite mode flag.
49	Overflow mode flag.
48	Underflow mode flag.
47	LCME (ECS) error condition.
46	CM error condition.
45	LCME block range condition.
44	CM block range condition.
43	LCME direct range condition.
42	CM direct range condition.
41	Program range condition.
40	Not used.
39	Step condition.
38	Indefinite condition.
37	Overflow condition.
36	Underflow condition.

---

† Does not apply to CDC CYBER 170 Series, Model 176.

†† CDC CYBER 170 Series, Models 171, 172, 173, 174, 175, 720, 730, 750, and 760 only.

††† CDC CYBER 70 Series, Model 74 only.

†††† CDC CYBER 170 Series, Model 176 only.

Field	Description
RAE	Reference address for ECS.
FLE	Field length for ECS.
MA	Monitor address.
NEA†	Normal exit address.
EEA†	Error exit address.
Xi	Operand registers.

ERROR FLAGS

Error flag	Mnemonic	Description
1	ARET	Arithmetic error.
2	PSET	Program stop.
3	PPET	PP abort.
4	CPET	CPU abort.
5	PCET	PP call error.
6	TLET	Time limit.
7	FLET	File limit.
10B	TKET	Track limit.
11B	SRET	SRU limit.
12B	FSET	Forced error.
13B	ODET	Operator drop.
14B	RRET	Operator rerun.
15B	OKET	Operator kill.
16B	SSET	Subsystem abort.
17B	ECET	ECS parity error.
20B	PEET	CPU parity error.
21B	SYET	System abort.
22B	ORET	Override error condition.

---

† CDC CYBER 170 Series, Model 176 only.

MASS STORAGE LABEL FORMAT

DEVICE LABEL TRACK FORMAT

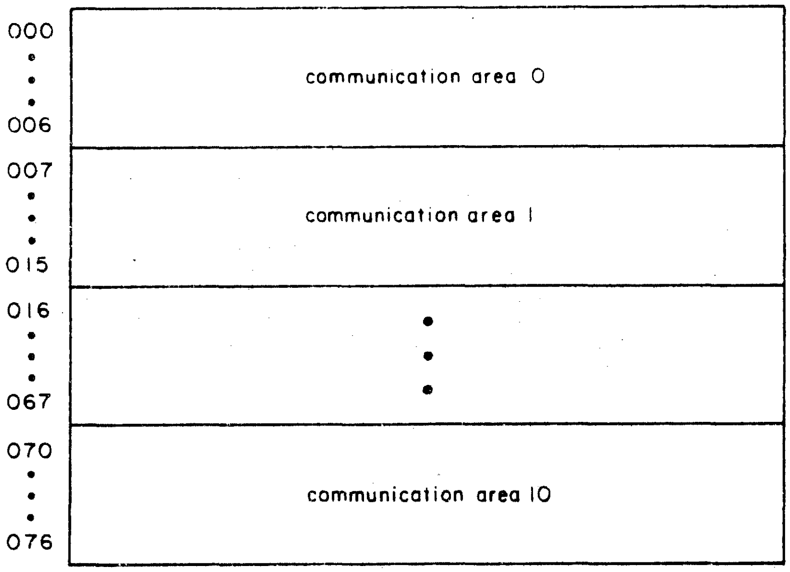
000	label sector
001 • • • 012	track reservation table
013	sector of local information (2-word entries)
014	device information sector
015	intermachine communication area (ECS label track only)
016	MMF environment tables (ECS label track only)
017	CPUMTR storage move area for ECS (ECS label track only)

DEVICE LABEL SECTOR FORMAT

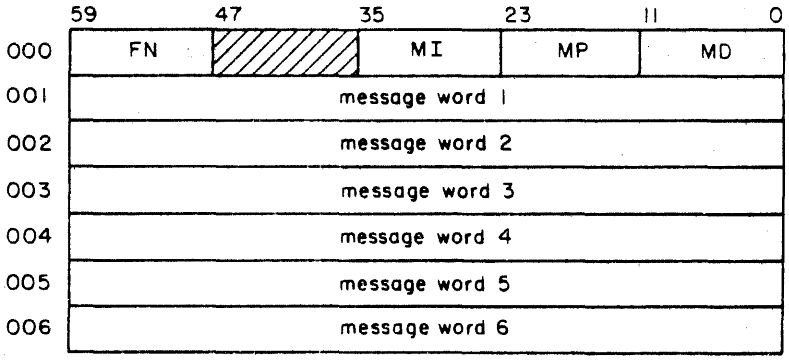
000 001 002	reserved		
003	label level	equipment type	reserved
004 005 006 007	reserved		
010 • • • 027	NOS MST		
030 • • • 077	unused		

MULTIMAINFRAME TABLES

INTERMACHINE COMMUNICATION AREA



Each communication area has the following format.



FN	Intermachine function number.
MI	Machine initiating request.
MP	Machines to process request.
MD	Machines done processing request.

MMF ENVIRONMENT TABLES

Sector 16_8 of the ECS label track is defined as follows:

	59	47	11 0
000	MMFL for mainframe 1		
001	MMFL for mainframe 2		
002	MMFL for mainframe 3		
003	MMFL for mainframe 4		
004	multi-mainframe 1 system time		
005	multi-mainframe 2 system time		
006	multi-mainframe 3 system time		
007	multi-mainframe 4 system time		
010	next DAT track	▨	DAT count
011	▨	▨	FAT count
012 • • • 033	One word per flag register bit. Each word contains the MMFL word of the machine which currently has the corresponding flag register interlock.		
034	machine 1 requests		
035	machine 2 requests		
036	machine 3 requests		
037	machine 4 requests		
040	machine 1 requests		
041	machine 2 requests		
042	machine 3 requests		
043	machine 4 requests		
044 • • • 067	unused		
070 • • • 077	installation area		

MMF - DAT TRACK CHAIN (ECS)

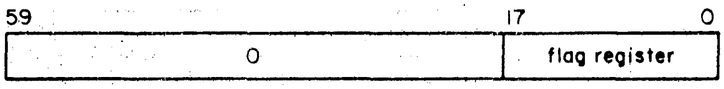
Track N

0000 ⋮ 0777	device access table (DAT)
1000	fast attach table (FAT)

Track M (same format for each device)

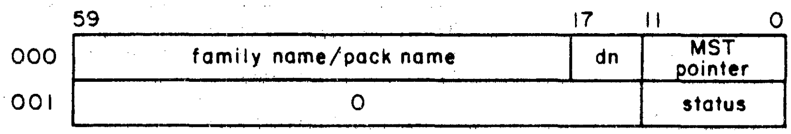
0000 ⋮ 0011	MST for shared device (global area)
0012 ⋮ 0017	local area for machine index 1
0020 ⋮ 0025	local area for machine index 2
0026 ⋮ 0033	local area for machine index 3
0034 ⋮ 0041	local area for machine index 4
0042 ⋮ 0077	unused
0100 ⋮ 1077	TRT for device
1100 ⋮ 1177	MRT1 (machine recovery table)
1200 ⋮ 1277	MRT2
1300 ⋮ 1377	MRT3
1400 ⋮ 1477	MRT4

MMF - ECS FLAG REGISTER FORMAT



Bit Set	Name	Description
17-12	---	Reserved.
11	COMI	CPUMTR intermachine communication request present.
10	CIRI	CPUMTR interlock recovery.
9	FATI, PFNI	FAT and PFNL interlock.
8	IFRI	Intermachine function request interlock.
7	BTRI	Block transfer in progress.
6	PRSI	Deadstart ECS preset in progress.
5	DATI	Device access table interlock.
4	TRTI	TRT interlock; machine. specified by bits 3-0 is requesting a TRT interlock.
3-0	---	Machine mask indicating which machine has TRT interlock bit set.

DEVICE ACCESS TABLE (DAT) ENTRY



dn	Device number.
MST pointer	If zero, device is not shared.
status	Bits 11-5 are reserved, bit 4 is set if recovery is in progress, and bits 3-0 are machine mask of machines accessing device.

FAST ATTACH TABLE (FAT) ENTRY - GLOBAL

	59	47	35	23	17	11 0
000	fast attach file name				▨	
001	▨	first trk	RM	RA	R	▨
002	mach. 1 ID	▨	RM	RA	R	▨
003	mach. 2 ID	▨	RM	RA	R	▨
004	mach. 3 ID	▨	RM	RA	R	▨
005	mach. 4 ID	▨	RM	RA	R	▨
006	family name				dn	▨
007	0					

RM	READMD users.
RA	READAP users.
R	Read/write users.
dn	Device number.

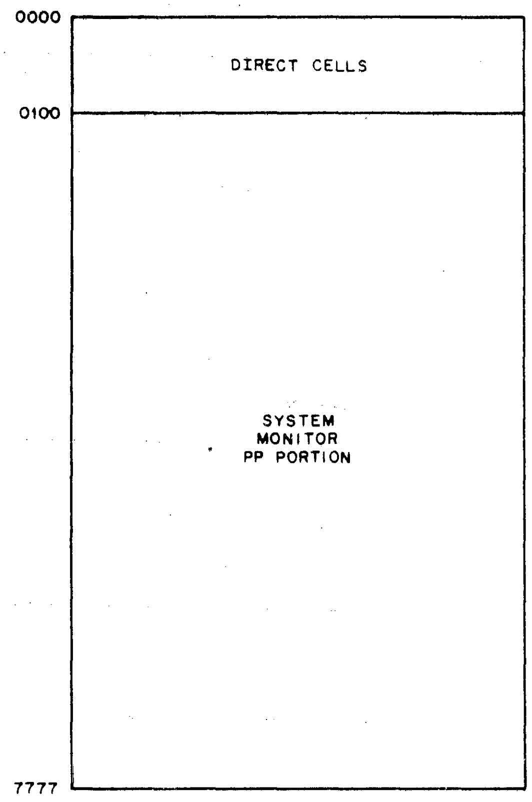
PFNL ENTRY FORMAT - GLOBAL

000	0
001	PFNL (global)
002	PFNL for mainframe 1
003	PFNL for mainframe 2
004	PFNL for mainframe 3
005	PFNL for mainframe 4
006	0
007	0

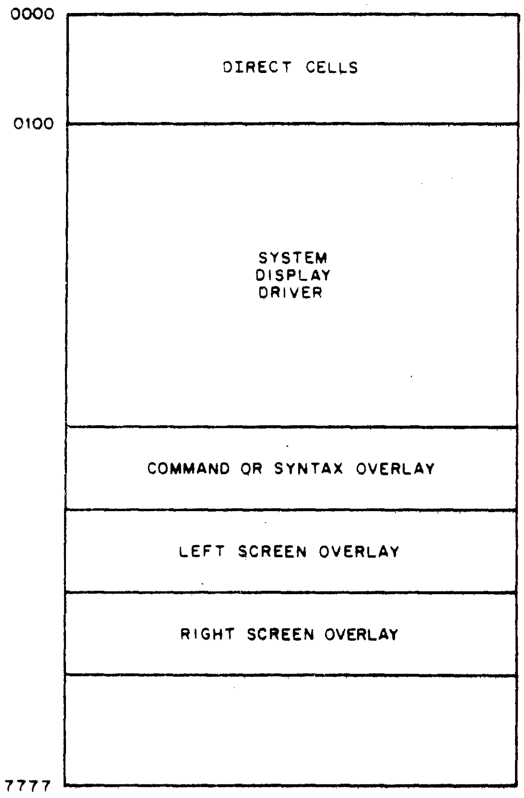
The first entry of the FAT is an 8-word entry of PFNL words in the preceding format.

PPU MEMORY LAYOUT

PP0 - SYSTEM MONITOR (PPU PORTION)

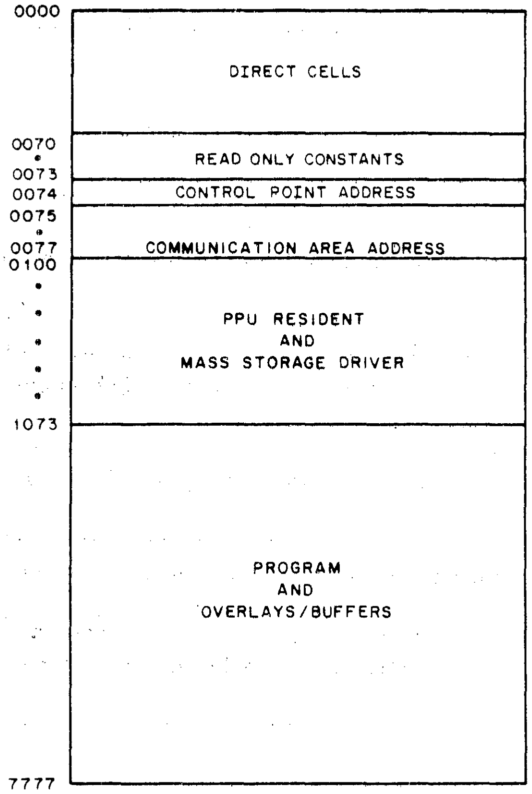


PP1 - SYSTEM DISPLAY DRIVER (DSD)



POOL PROCESSORS

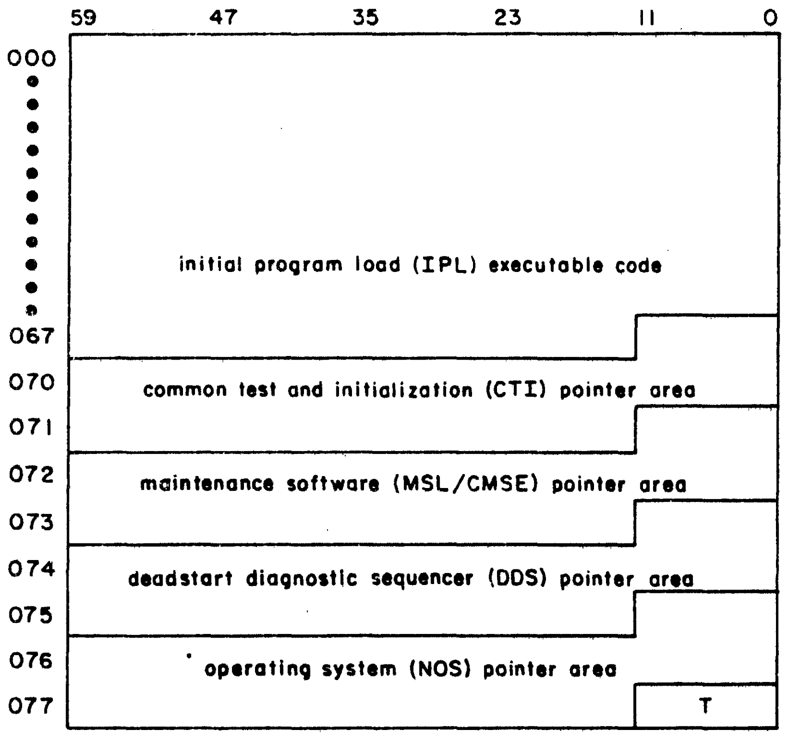
(PP2 through PP11 on 10 PP machines; PP2 through PP11 and PP20 through PP31 on 20 PP machines.)†



---

† PP numbers are in octal notation.

DISK DEADSTART SECTOR FORMAT



T = IPL transfer address −1 (7420_8)

CPU AND PP MONITORS

NOS utilizes two monitors: CPUMTR (central processor monitor) which controls CPU monitor mode execution and CPU scheduling; and MTR (peripheral processor monitor) which is in general control of the system and operates in PP0.

These two monitors work together, yet independently to allow the system to run smoothly and effectively.

Figure 3-1 is an overview of system interaction showing both monitors as a controlling entity. PPs communicate with the CPU and vice versa through MTR by means of input registers (IR), output registers (OR), and RA+1 calls.

Figure 3-2 shows the interaction between this monitor concept and PP resident using the PP IR and OR.

Figure 3-3 shows the monitor interaction between the CPU, PP, and each monitor using the exchange jump feature. With the central exchange jump/monitor exchange jump (CEJ/MEJ) option, the CPU program can either wait for MTR to call CPUMTR by finding RA+1 nonzero, or the CPU program can directly call CPUMTR. PP routines may either wait for MTR to call CPUMTR by finding the OR nonzero or call CPUMTR directly. Without the CEJ/MEJ option, CPU routines and PP routines must wait for MTR to call CPUMTR for them.

Figure 3-4 shows the entry points for CPUMTR, while tables 3-1, 3-2, and 3-3 show the monitor functions processed by CPUMTR.

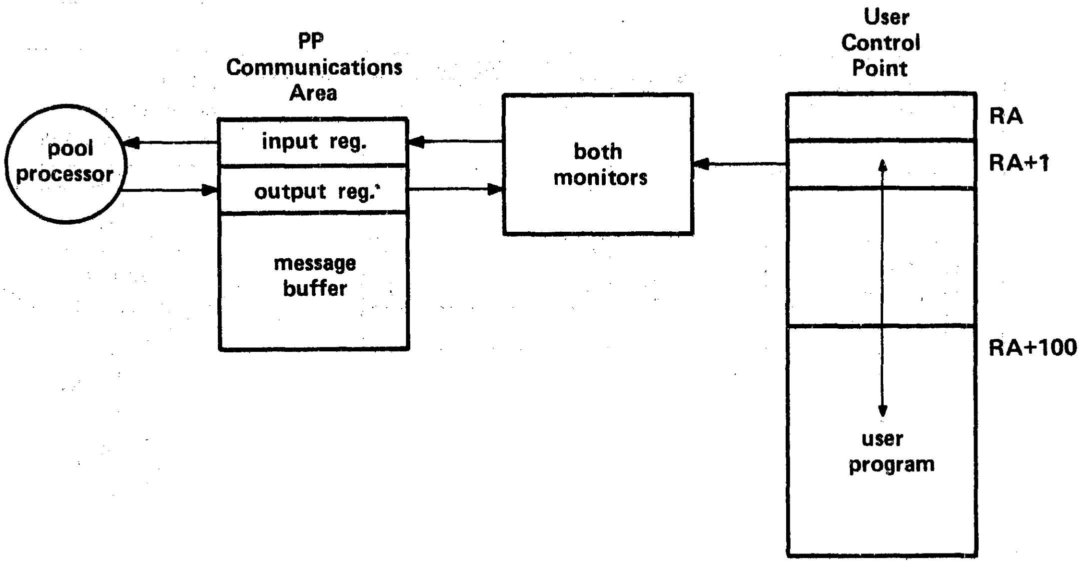


Figure 3-1. System Interaction

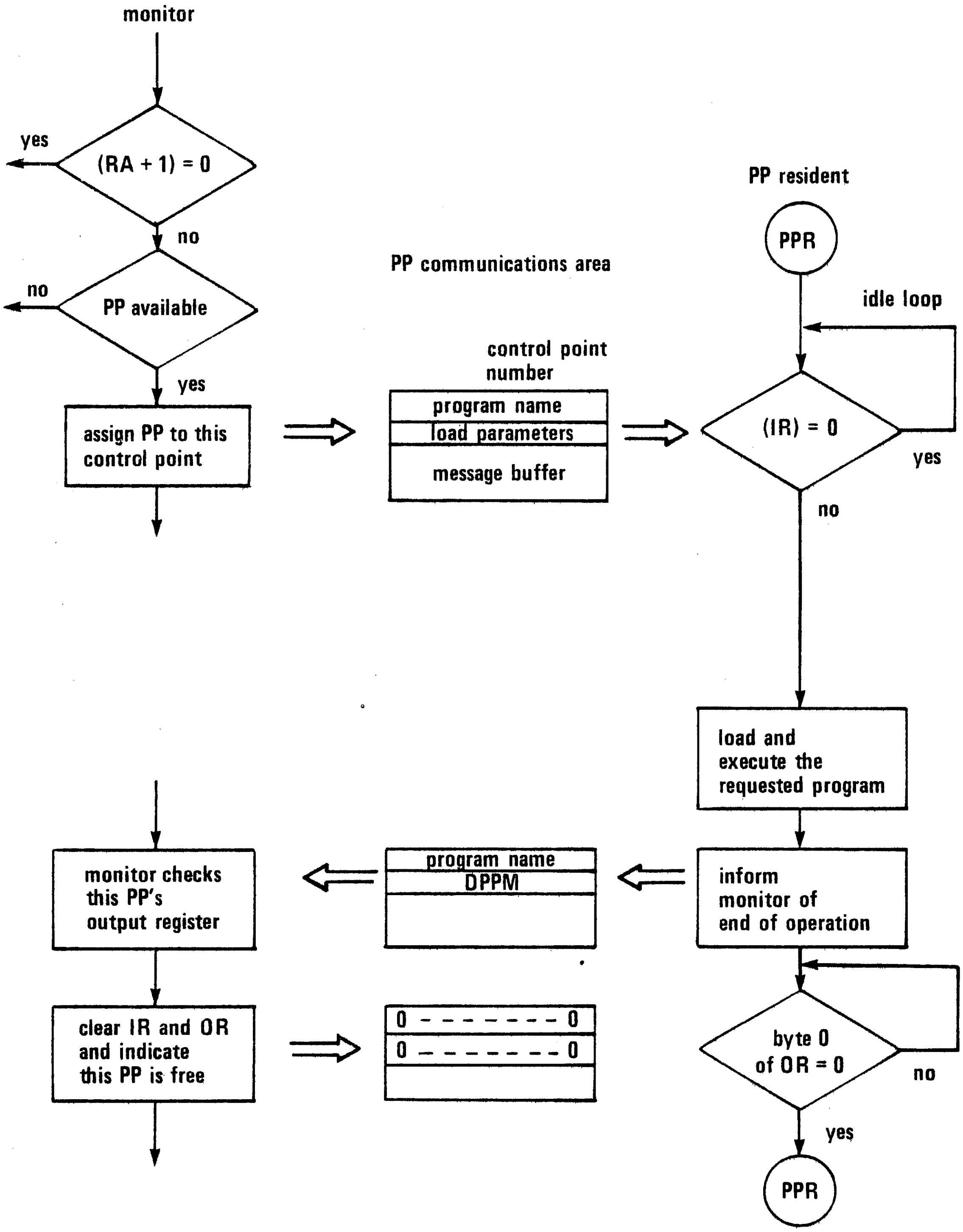


Figure 3-2. System Interaction

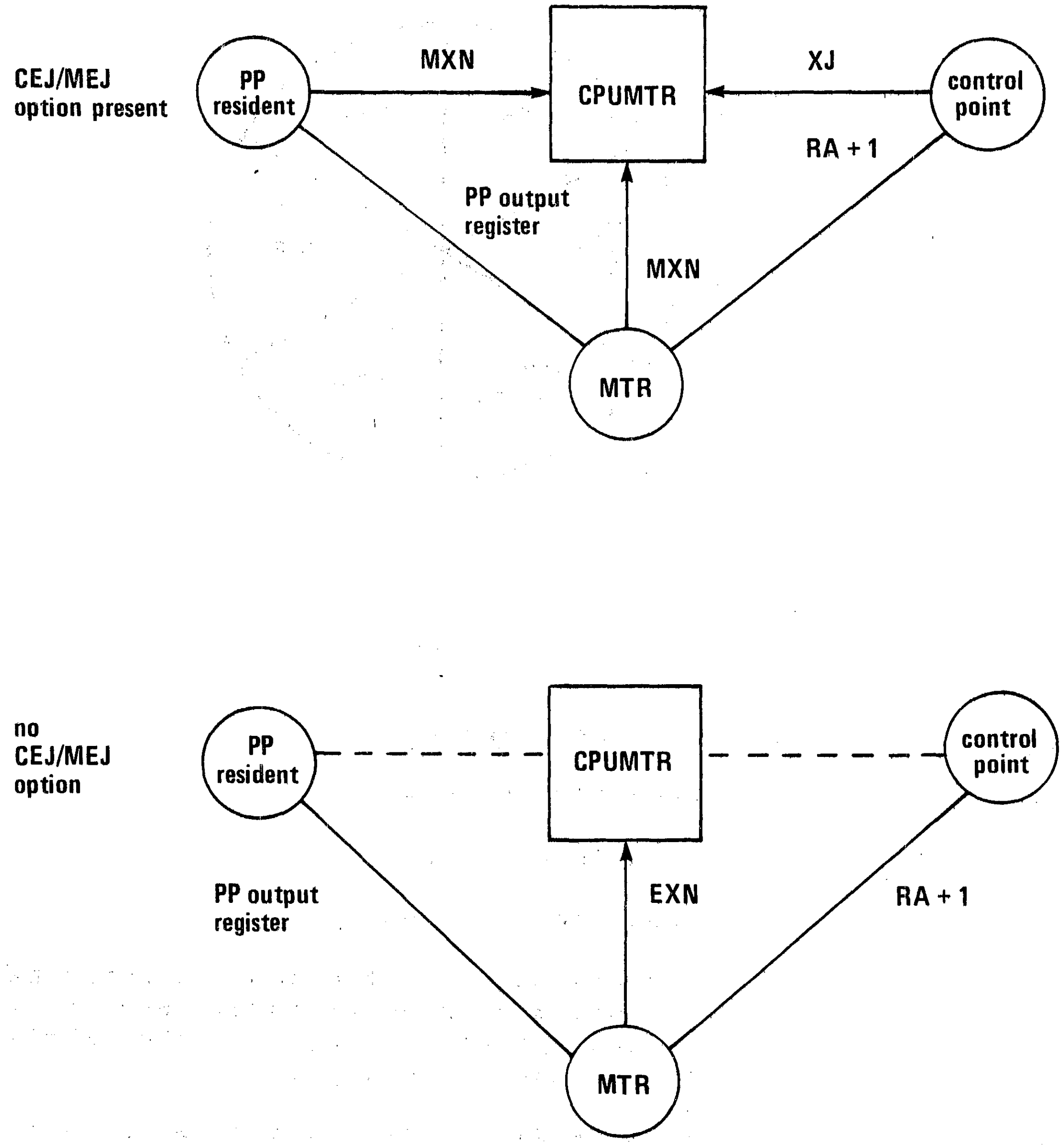
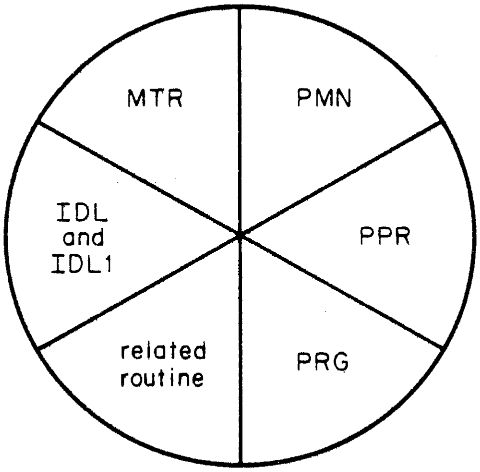


Figure 3-3. Monitors Interaction



Entry Name	Description
MTR	From CPU program
PMN	From PP monitor
PPR	From pool PP program
PRG	Address where system control point begins execution in program mode. When system control point exchanges to the CPUMTR, CPUMTR begins execution at MTR
IDL IDL1	From CPUMTR. These are idle loops for CP0 and CP1 respectively

Figure 3-4. CPUMTR Entry Points from Exchange Packages

All system interaction is effected using the exchange jump instructions.

The executable code of CPUMTR begins at the end of the dayfile dump buffer.

Functions processed by MTR for pool PPs enter CPUMTR at PPR (the value determines that the function is intended for MTR).

TABLE 3-1. VALUES OF MTR FUNCTIONS

Name	Value	Description
--	1	Unassigned
--	2	Unassigned
CCHM	3	Check channel
DCHM	4	Drop channel
DEQM	5	Drop equipment
DFMM	6	Issue dayfile message
--	7	Unassigned
SEQM	10	Set equipment parameters
PRLM	11	Pause for storage relocation
RCHM	12	Reserve channel
REMM	13	Request exit mode
REQM	14	Request equipment
ROCM	15	Rollout control point
RPRM	16	Request priority
RJSM	17	Request job sequence number
--	20	Unassigned
RSTM	21	Request storage
--	22	Unassigned
DSRM	23	DSD requests
ECXM	24	ECS transfer
TGPM	25	IAF/TELEX get pot
TSEM	26	IAF/TELEX request
DEPM	27	Disk error processor
DRCM	30	Driver recall CPU
SCPM	31	Select CPU(s) allowable for job execution
EATM	32	Enter access system event table
DSWM	33	Driver seek wait
--	34-35	Unassigned

Functions processed by CPUMTR, enter CPUMTR at PPR.

TABLE 3-2. VALUES OF CPUMTR FUNCTIONS

Name	Value	Description
ABTM	36	Abort control point
CCAM	37	Change CP assignment
CEFM	40	Change error flag
DCPM	41	Drop CPU
SFIM	42	Set FNT interlock
DTKM	43	Drop tracks
DPPM	44	Drop PP
ECSM	45	ECS transfer
RCLM	46	Recall CPU
RCPM	47	Request CPU
RDCM	50	Request data conversion
IAUM	51	Interlock and update fields in CMR/ECS
ACTM	52	Accounting functions
RPPM	53	Request PP
RSJM	54	Request job scheduler
RTCM	55	Reserve track chain
SFBM	56	Set file busy
STBM	57	Set track bit
UADM	60	Update accounting and drop PP
SPLM	61	Search peripheral library
JACM	62	Job advancement control
DLKM	63	Delink track chain
TDAM	64	Transfer data
TIOM	65	Tape I/O processor
RLMM	66	Request time or SRU limit
LCEM	67	Load central program
CSTM	70	Clear storage
CKSM	71	Checksum specified area
LDAM	72	Load disk address
VMSM	73	Validate mass storage
PIOM	74	PP IO via the CPU
--	75	Unassigned
MXFM	76	Maximum number of functions

Functions issued by MTR (only) and processed by CPUMTR enter CPUMTR at PMN.

TABLE 3-3. MTR FUNCTIONS PROCESSED BY CPUMTR IN MONITOR MODE

Name	Value	Description
ARTF	1	Update running time
IARF	2	Initiate auto recall
EPRF	3	Enter program mode request
MRAF	4	Modify RA
MFLF	5	Modify FL
SCSF	6	Reset CPU I status
SMSF	7	Set monitor step
CMSF	10	Clear monitor step
ROLF	11	Set rollout required
ACSF	12	Advance CPU switch
PCXF	13	Process alternate CPU exchange
ARMF	14	Advance running time MMF mode
MREF	15	Modify ECS RA
MFEF	16	Modify ECS FL

MTR functions processed by CPUMTR in program mode enter CPUMTR at MNR (table 3-4). Table 3-5 lists RA+1 requests processed by CPUMTR.

TABLE 3-4. MTR-CPUMTR PROGRAM MODE REQUESTS

Name	Value	Description
MSTF	0	Storage move
PDMF	1	Process down machine
PMRF	2	Process intermachine function request
MECF	3	Move ECS storage

TABLE 3-5. RA+1 REQUESTS PROCESSED BY CPUMTR

Name	Description
ABT	Abort control point
CPM	Resident CPM functions:
	16 Read error exit
	24 Read job control word
	25 Write job control word
	32 Return user number
	33 Read FL control word
	37 Read TELEX subsystem
	43 Read special entry point word
	45 Read first loader control word
	50 Read machine ID word
	55 Read ECS FL control word
	61 Read list of files pointer
	62 Set list of files pointer
END	Terminate current CPU program
LDR	Request overlay load
LDV	Request loader action
LOD	Request autoload of relocatable File
MEM	Request memory
MSG	Send message to system
PFL	Set (P) and change field length
RCL	Place program on recall
RFL	Request field length
RSB	Read subsystem program block *2
SIC	Send intercontrol point block to subsystem *1
SPC	Process special PP requests *3
TIM	Request system time
XJP	Initiate subcontrol point *4
XJR	Process exchange jump request

*1 Honored for jobs with QP less than MXPS, SSJ= or access bit (CSTP) set

*2 Honored for jobs with QP greater than MXPS or SSJ=

*3 Honored for jobs with QP greater than MXPS

*4 Allowed only when subcontrol points are enabled (SUBCP block is loaded)

MTR FUNCTIONS

The following paragraphs describe the MTR functions. The format for the calls are contained in the NOS Systems Instant and the external documentation of MTR and CPUMTR using the control statement DOCMENT.

CCHM (3) - CHECK CHANNEL

This function allows a PP to have a channel checked for availability. If the channel is free, it is assigned; if not, the channel requested bit (bit 1) in the CST is set. Control is returned to the PP immediately (compare with RCHM).

DCHM (4) - DROP CHANNEL

Sets assignment for this channel in the CST bits 10-7 to zero. It is used to release the channel reserved with RCHM or CCHM. This function is used by the PPR routine DCH. This also does a release unit reserve function when the device is MS and the R option is set for a dual access controller. Refer to the CMRDECK mass storage EST entry in the NOS Installation Handbook.

DEQM (5) - DROP EQUIPMENT

This function releases the equipment by setting bits 52-47 of the EST entry to zero. It is used to release equipment reserved with the AEQM or REQM.

DFMM (6) - PROCESS DAYFILE MESSAGE

This function allows a PP to send a dayfile message to any of the system or control point dayfiles. Used by the PPR routine DFM.

SEQM (10) - SET EQUIPMENT PARAMETERS

Depending upon subfunction code, this function performs one of the following.

0	ON equipment (set bit 23 of EST)
1	OFF equipment (clear bit 23 of EST)
2	Set channels for access in EST
3	Set equipment mnemonic in EST
4	Set byte 0 of EST
5	Set byte 1 of EST
6	Set byte 2 of EST
7	Set byte 3 of EST
10	Set byte 4 of EST

PRLM (11) - PAUSE FOR STORAGE RELOCATION

Any PP which determines that its control point has a storage move request pending (CMCL word 57 byte 0) must issue this function. MTR will not move the control point until all PP activity for that control point has recognized the requested move via PRLM, DSWM, or DFMM. This function is used by the PPR routine PRL.

RCHM (12) - REQUEST CHANNEL

This function sets the CST bits 10-7 to the control point number, thereby assigning the channel for the up to four channels available. The RCHM will not return control to the PP until the channel can be reserved. Compare with the CCHM which returns control whether the channel can be assigned or not.

REMM (13) - REQUEST EXIT MODE

This function sets the exit mode in the exchange package to the specified 12 bits.

REQM (14) - REQUEST EQUIPMENT

This function allows the PP to request an equipment. Control is returned whether the equipment is available or not.

ROCM (15) - ROLLOUT CONTROL POINT

This function sets the rollout requested bit (bit 24 in word JCIW of the control point area). A PP routine cannot force a job to rollout immediately; it must request rollout action. CPUMTR determines when the job may be rolled out and 1AJ is then called.

RPRM (16) - REQUEST PRIORITY

This function sets the CPU or queue priority in the control point area (word JCIW).

RJSM (17) - REQUEST JOB SEQUENCE NUMBER

This function returns the current job sequence number from central memory word JSNL, and increases it by one.

RSTM (21) - REQUEST STORAGE

This function allows a PP routine to change the FL/FLE at a control point. The request is the amount of FL desired at the control point. If the request is for the same amount of FL or less than that already assigned, then the request is honored immediately (unless for the last control point). If the request is for an increase, storage moves may be necessary. Control is returned immediately in any case. If a PP wishes to reduce FL it should make this request. If it wishes to increase FL it should use the common routine COMPRSI to make increase storage requests.

NOTE

> The control point may be moved while this function is pending.

DSRM (23) - DSD REQUESTS

This function is only accepted from DSD; any other PP will be hung. When the operator types in STEP, UNSTEP, DATE, or TIME, DSD issues this function. STEP mode forces MTR to accept only one function at a time under direction of DSD. MTR steps CPUMTR and controls the processing of those functions (refer to SMSF). DSD can specify whether to step the system or only one control point. MTR reissues all CPUMTR functions that were stepped when an unstep is issued from DSD. The subfunction to set emergency step is also allowed from 1MB.

ECXM (24) - ECS TRANSFER

This function is used to transfer data between ECS and CM. The transfer is between a relative address in CM to/from a relative address in ECS. The function also allows the specification of an alternate response address. This allows the calling PP to overlap other monitor functions with this function.

TGPM (25) - IAF/TELEX GET POT

This is used to get a pot chain from IAF/TELEX. It is useful because the PP does not need to interrupt or start up IAF/TELEX for the request.

TSEM (26) - PROCESS IAF/TELEX REQUEST

Used to request various procedures from IAF/TELEX.

DEPM (27) - DISK ERROR PROCESSOR

Used for mass storage error processing.

DRCM (30) - DRIVER RECALL CPU

Used to issue an RCLM if the CPU is in periodic recall status. This function allows the PP to request MTR to determine the CPU status and issue an RCLM rather than do it itself. This request does not require an exchange jump; therefore the PP needs only to place the request in its OR and does not need to wait for it to be processed. This is critical for mass storage or tape drivers, that could lose a revolution or tape speed if it needed to wait for a CPUMTR request. However, the routine must wait for OR to clear before again issuing this function. Thus, mass storage drivers must wait for OR to clear.

SCPM (31) - SELECT CPUS ALLOWABLE FOR JOB EXECUTION

Sets byte 4 of the JCIW word of the control point area to zero for any CPU, one for CPU 0 only, and two for CPU 1 only. A selection of CPU 1 is ignored if user ECS is assigned.

EATM (32) - ENTER/ACCESS SYSTEM EVENT TABLE

Enter or read events to or from system event table.

CPUMTR FUNCTIONS

ABTM (36) - ABORT CONTROL POINT

Abort the control point to which this PP is assigned. Sets PPET error flag and performs a DPPM.

CCAM (37) - CHANGE CONTROL POINT ASSIGNMENT

Used to change the control point assignment for this PP. It reduces the PP count in the control point at STSW bits 52-48 in the old control point assignment, and increases it by one for the new control point assignment.

CEFM (40) - CHANGE ERROR FLAG

Replaces bits 47-36 in STSW word of the control point area. It is used to set or clear the error flag.

DCPM (41) - DROP CPU

If control point is in W status it is placed in zero status. Since there is PP activity the control point will not be advanced.

SFIM (42) - SET FNT INTERLOCK

Sets or clears an interlock bit for a particular FNT entry. The interlock bit for each FNT entry is kept in the FNT interlock table which is appended to the FNT. The interlock on an individual FNT entry should be held for the shortest time possible to avoid performance degradation.

This technique is used in the following circumstances.

- Bringing an input file into execution
- Performing a job advance
- Rolling in or rolling out a job
- Terminating a job
- Altering the FNT or system sector of a queued file
- Moving a file from one queue to another
- Assigning a queue file to a control point

DTKM (43) - DROP TRACKS

This is executed in program mode and is used to drop trailing tracks from a track chain.

DPPM (44) - DROP PP

This is the last function issued before a PP jumps to its idle loop. It signifies that this PP routine is done and the PP is available for other assignments.

ECSM (45) - ECS TRANSFER

Used to get from 1 to 100B words transferred from ECS to/from absolute or relative CM. Also used to set/clear flag register and read display information for DSD/DIS.

RCLM (46) - RECALL CPU

Used to change the control point status from periodic recall to CPU candidate; that is, X status to W status.

RCPM (47) - REQUEST CPU

Used to start the CPU for this control point and set the control point status to W. This function is also used by a PP program called with autorecall to bring the CPU back into execution to its control point.

RDCM (50) - REQUEST DATA CONVERSION

Used to convert 30-bit integer to FORTRAN F10.3 display code format.

IAUM (51) - INTERLOCK AND UPDATE

Used to interlock and update fields in CMR or ECS.

ACTM (52) - ACCOUNTING FUNCTIONS

Performs the following accounting functions.

- Begin account block
- Compute SRU multipliers
- Accounting block change
- Compute and convert elapsed SRUs
- Compute accumulators
- Increment accumulator

RPPM (53) - REQUEST PP

Used to start a PP routine in some other PP. The response indicates whether the PP was assigned or none available. A PP can read PPAL and determine in advance if a PP is available. This saves time and overhead.

RSJM (54) - REQUEST JOB SCHEDULER

This function is used to interlock scheduler calls, so that only one copy of 1SJ is running at one time in the system.

RTCM (55) - REQUEST TRACK CHAIN

This is executed in program mode. Allows the PP routine to request a specified number of sectors and reserve the proper track chain. When no equipmet is specified one is selected based upon the allocation parameter in the call.

SFBM (56) - SET FILE BUSY

Used to interlock the FNT/FST entry for a specific file. A PP issues this function to reserve the file and when done releases the file by setting bit 0 of the FST to one. SFBM sets bit 0 of the FST to zero. This function is used to interlock any word in CM, such as PFNL, or any word in the MST. If SFBM is issued for an FNT/FST, the file name word must also be provided to check that another PP has not dropped the file just after the PP issuing SFBM found it. In both the FST and the FET, the file is busy when bit 0 is clear.

STBM (57) - SET TRACK BIT

This is executed in monitor mode unless the system control point is active; then it is done in program mode. Used to set the w, d, or i bits in the TRT.

UADM (60) - UPDATE ACCOUNTING AND DROP

SPLM (61) - SEARCH PERIPHERAL LIBRARY

Used to search PLD for a PP routine.

JACM (62) - JOB ADVANCEMENT CONTROL

Options 1, 2, 3, and 4 are used to set or clear the job advancement flag at a control point with implied DPPM if desired. PP routines should not call 1AJ directly for job advancement. CPUMTR will decide when a job needs to be advanced and call 1AJ to the job. 1AJ then decides if the control point needs advancement or rollout.

DLKM (63) - DELINK TRACKS

This is executed in program mode. DLKM is used to drop intervening tracks on an existing file chain and relink the file chain properly. An example is PFM delinking an indirect access file chain in response to a user issuing a PURGE on a file which is long enough to completely cover several tracks. PFM attempts to keep the indirect access file chain to a minimum size when possible.

TDAM (64) - TRANSFER DATA BETWEEN MESSAGE BUFFER, JOB

Allows a PP to transfer up to 6 words from/to the message buffer to/from a job. The address to transfer to/from is a relative address. The transfer must be to/from a subsystem. It alleviates the problem of a PP finding the subsystem and deciding if it is ready for reception of data. This is equivalent to the SIC/RSB facility except no intercontrol point communication area is necessary.

TIOM (65) - TAPE I/O PROCESSOR

This function updates the tape accounting information; that is, the number of blocks transferred in MTUW word 53 of the control point area. Exit from this function is to CCAM to change the PP assignment to MAGNET's control point. If the completion code is nonzero, the specified UDT word is cleared, the FET is set complete, and the tape activity count is decremented in STSW word, byte 2. Routine 1MT uses this function when it completes a read/write request on a tape. Since the UDT and the FET must be changed, and they are at two different control points, this function prevents any problem by keeping the control point and MAGNET from interfering with each other. UDT must be cleared before the FET is set complete or an I/O sequence error could occur.

RTLM (66) - REQUEST CPU TIME LIMIT

Used to change the CPU time limit in CTLW word, bytes 2, 3, and 4 in the control point area. The time limit exceeded flag in ACTW word, byte 0 is cleared.

LCEM (67) - LOAD CENTRAL PROGRAM

This is executed in program mode. Used to load an ECS or CM resident routine into the control point field length.

CSTM (70) - CLEAR STORAGE

Used to clear a specified amount of CM or ECS. When clearing an FNT/FST entry, CSTM can also be used to set the control point area FNT interlock (ECSW word, bit 47).

CKSM (71) - CHECKSUM SPECIFIED AREA

Checksum area from FWA to LWA+1 and compare to checksum in message buffer (MB).

LDAM (72) - LOAD DISK ADDRESS

Used to convert from logical to physical addresses for 844 equipments.

VMSM (73) - VALIDATE MASS STORAGE

This function validates a mass storage device's MST and TRT by checking track reservation and preserved file track count against the count in the MST. Also, critical track chains are validated.

PIOM (74) - PP IO VIA CPU

This function is used by the 6DE driver to transfer data to/from ECS via the PP buffers immediately preceding CPUMTR.

MXFM (76) - MAXIMUM FUNCTION NUMBER

This is used by a PP when it desires to hang itself for some reason it considers catastrophic. CPUMTR will see that is is out of range and will hang the PP. Whenever a PP issues this function it should allow the analyst to clear the PP's output register and complete its operation gracefully.

A PP is hung when one of the monitors determines that a function is illegal. For example, function out of range, or RCHM on some nonexistent channel. If CPUMTR hangs a PP the message PP HUNG is displayed at the system control point.

If MTR hangs a PP the message is HUNG PP.

In any case the packed date and time of the hang is placed in MB+5.

MTR FUNCTIONS TO CPUMTR

These are special functions and the request is transmitted via the X0 register instead of MTR's output register.

(0) - RA REQUEST

This function tells CPUMTR that some control point has an RA+1 request. This is used for systems where the XJ is not available or the user's program is not doing an XJ. Upon entry X0 is zero.

ARTF (1) - ADVANCE RUNNING TIMES

Update running times. Updates RTCL in CMR and ACTW in the control point area and sets time limit exceeded flag if time limit has been exceeded. It also checks for P equal to 0 and program stop. CPUMTR checks the active control point and the instruction P points to.

	59	17 0
entry (X0)	0	ARTF

IARF (2) - INITIATE AUTORECALL

MTR while in the routine PPL (process PP recalls) checks RA+1 of a control point in autorecall and if RA+1 is set with autorecall requested, it reissues the PP request.

If a PP routine who is called with autorecall finds that it cannot process the request it was called for at this time, it can copy its IR back to RA+1 if the control point is in R status. When MTR goes through its PPL routine it will find the request and have CPUMTR reissue it to a PP.

	59	35	23	17 0
entry (X0)	0	cpa fwa	0	IARF

EPRF (3) - ENTER PROGRAM MODE REQUEST

	59	35	23	17 0
entry (X0)	0	pr	0	MSTF

pr	Program mode request number as defined in COMSMTR

MRAF (4) - MODIFY RA

CPUMTR changes RA in STSW and the entry point by the specified amount.

	59	47	35	17 0
entry (X0)	in/100	0	cpa fwa	MRAF

in	Value to change RA

MFLF (5) - MODIFY FL

CPUMTR changes FL in STSW and the entry point by the specified amount.

	59	47	35	17	0
entry (X0)	in/100	0	cpa fwa	MFLF	

SCSF (6) - SET (RESTORE) CPU STATUS

CPUMTR places the specified status in the STSW word. This is used when MTR issues the DCPM function. The status is returned to MTR to be restored after the control point is storage moved. When MTR is ready to restart the CPU it issues this function restoring the former status.

Functions EPRF, MRAF, and SCSF may all be used when a control point needs to have its FL changed via the RSTM function. If MTR has to move the control point it issues the DCPM and saves the status, then issues the EPRF for the move. If no storage move is required, then the MRAF is used.

Finally, it issues SCSF to restore the former status. When a control point is going to be moved, the only criterion for that move is no PP activity, so the control point could be in any status when MTR is ready to make the move, and after the move, the proper status must be restored.

	59	47	35	17	0
entry (X0)	status	0	cpa fwa	SCSF	

SMSF (7) - SET MONITOR STEP

This allows CPUMTR to disable its automatic processing of monitor functions and to wait for MTR to indicate which function to process. SMSF and CMSF are used to set and clear the system STEP mode. Refer to DSRM.

	59	17	0
entry (X0)	0	SMSF	

CMSF (10) - CLEAR MONITOR STEP

Reenables automatic processing of monitor functions.

	59	17 0
entry (XO)	0	CMSF

ROLF (11) - SET ROLLOUT FLAG AND CHECK JOB ADVANCE

This dual timing is used to set the rollout flag and check for job advancement.

	59	35	17 0
entry (XO)	0	cpa fwa	ROLF

ACSM (12) - ADVANCE CPU JOB SWITCH

Used to change the control point assignment of the CPU. It is used in the MTR routine JSW to process CPU job switching. This involves exchanging the CPU from one control point to another (slot time exceeded processing).

	59	35	17 0
entry (XO)	0	cpa fwa	ACSM

PCXF (13) - PROCESS CPU EXCHANGE REQUEST

If CPUMTR is executing in one CPU and needs to be in the other CPU it will inform MTR via the CX words and XJ. MTR then issues this request to the other CPU. This is done in the AVC advance clock routine, which is the one section of MTR that must execute at least every 4 milliseconds. For example, consider function ABTM. PPR cannot distinguish which CPU its control point is in, so it starts CPUMTR up in CP0. If the control point to be aborted is in CP1, then CPUMTR must get itself into CP1 in order to get the control point out of CP1.

MTR processes pool PP OR requests as follows.

ne.

If the CEJ/MEJ is available or is disabled, MTR checks all OR requests. If a request is for CPUMTR, MTR jumps to its CPR routi CPR exchanges in CPUMTR for that PP.

If the CEJ/MEJ is available, MTR ignores any CPUMTR request, since the PP must issue its own MXN; that is, CP0 cannot stop CP1, so the PCXF alternate exchange request is made.

	59	17 0
entry (X0)	0	PCXF

ARMF(14) - ADVANCE RUNNING TIME AND MMF PROCESSING

This function is called once every second by MTR to:

- Status flag register bits
- Write real-time clock to ECS
- Read other mainframe clocks in ECS (every two seconds)

	59	35	23	17 0
entry (X0)	0	s	0	ARMF

MREF (15) - MODIFY ECS RA

CPUMTR changes the ECS RA in ECSW and the exchange package by the amount specified.

	59	47	35	23	17 0
entry (X0)	in/1000	0	CPA FWA	0	MREF

MFEF (16) - MODIFY ECS FL

CPUMTR changes the ECS FL in ECSW and the exchange package by the amount specified.

	59	47	35	23	17 0
entry (X0)	in/1000	0	CPA FWA	0	MFEF

CPUMTR STRUCTURE

During deadstart, CPUMTR is loaded into CMR with the appropriate blocks for a particular environment. For instance, if ECS is available, the block of code pertaining to ECS is loaded; if multimainframe has been selected, the associated MMF code is loaded. Since unnecessary blocks of code are not loaded, the size of CMR is optimally maintained. Optional blocks of code which might be loaded include the following.

Block	Purpose
CMU	Move storage with compare/move unit (single CPU system, only)
OCMU	Move storage with registers (for non-CMU machines)
CMUMTR	Monitor mode CMU move
OCMUMTR	Monitor mode move storage with registers
CP176	Code to process CYBER 170 Model 176 hardware
DCP	Dual CPU operations
MMF	Multimainframe processing routines
OMMF	Processing routines without MMF
SCP	System control point facility
SUBCP	Subcontrol point processing
UEC	User ECS routines
VMS	Validate mass storage
ECS	ECS processing routines
ECSBUF	ECS buffer space
MMFBUF	MMF buffer space
EXPACS	Exchange packages
CEJ	Central exchange enabled
XP176	Exchange packages for the CYBER 170 Model 176
OCEF	CEJ disabled
PRESET	Preset CPUMTR (overlaid by PPU exchange packages)

CPUMTR has the following structure.

- MTR main program. Entry point from CPU program.
- Utility subroutines
- CPR - CPU program request processing. Requests are passed through RA+1 (refer to table 3-5).
- PMN - MTR request processor (refer to MTR Functions to CPUMTR).

- PPR - PPU request processor (functions listed later in this section).

- Program mode subroutines.

- MNR - Monitor request processor. Program mode processors not initiated by PP functions.

- Tables:

TPMN	PPU monitor requests
TPPR	PPU request table

MTR STRUCTURE

MTR is loaded into PP0 at deadstart time and remains there for the duration of system execution.

MTR performs the following functions:

- Processes certain PP requests
- Allocates central memory and user ECS
- Maintains the real-time clock
- Checks (RA+1) of active CPU programs for system requests
- Checks OR of each pool PP
- Checks the SCR (CYBER 170) or ILR (CYBER 70 or 6000) for errors which require 1MB processing.

STARTING MTR AT DEADSTART TIME

MTR is loaded in PP0. The first location of the code is:

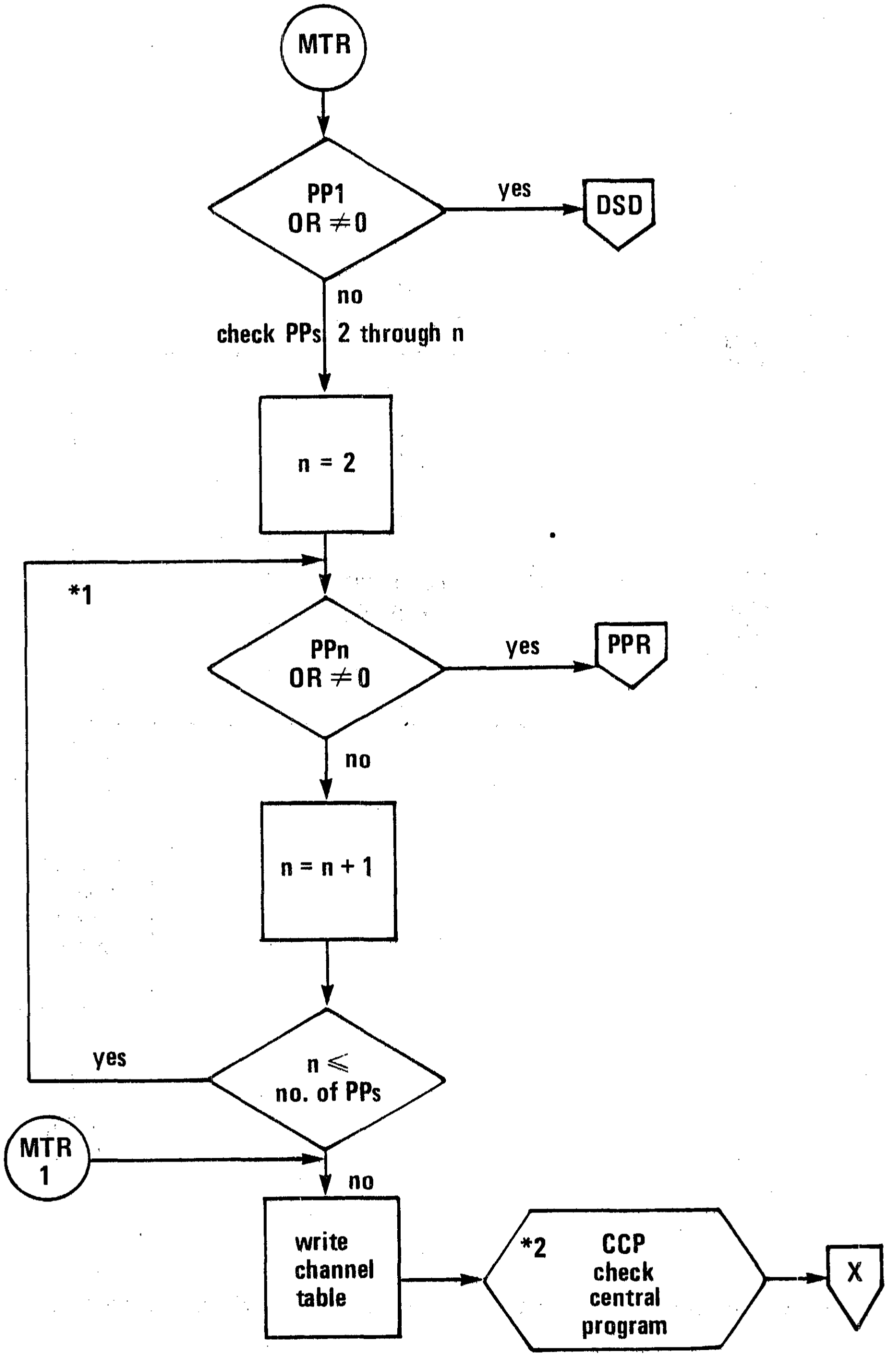
```
    T0   CON   PRS-1
```

This forces the constant PRS-1 to fall into T0. At the end of the load, (P) is set to (T0)+1 which will be (P)=PRS, the MTR preset routine. PRS presets all tables and constants.

PRS overlays itself with tables and buffers.

CPUMTR/MTR FLOWCHARTS

Figures 3-5 through 3-22 flowchart the main routines used by MTR and CPUMTR.



*1 This simulated loop is a DUP statement in MTR code.

*2 When MTR releases a channel, it sets a flag. At this time, the reservation byte in the channel table in CMR is cleared.

Figure 3-5. Main Loop for MTR

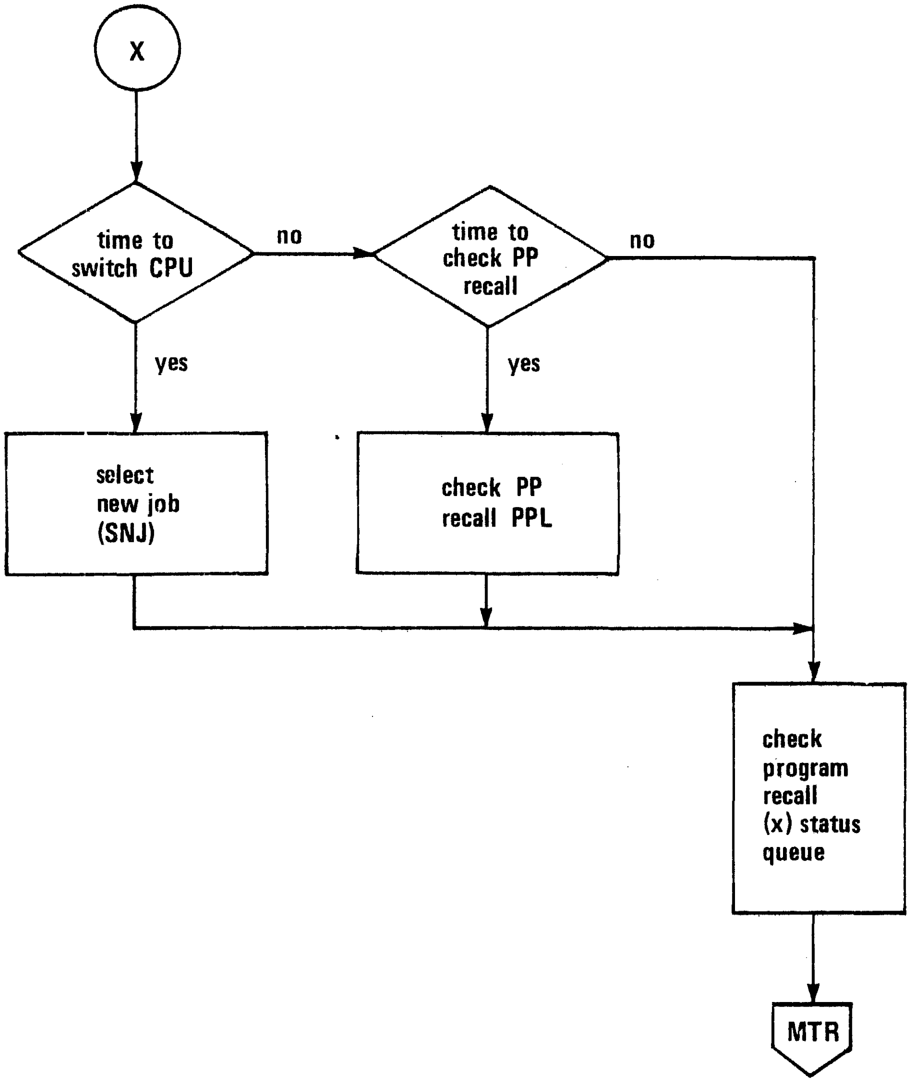


Figure 3-6. Process Time Dependent Scanners

REAL-TIME CLOCK

The real-time clock starts with power on and runs continuously. It may be read by any peripheral processor with an input to A (70) instruction from channel 14B. This channel is separate from the data channels.

The clock period is 4096 (10000B) micro seconds. It is a 12-bit register that is advanced each microsecond from 0 through 7777B. When it reaches 7777B, it starts over at 0. It must, therefore, be read at least every 4.096 milliseconds for accurate timing.

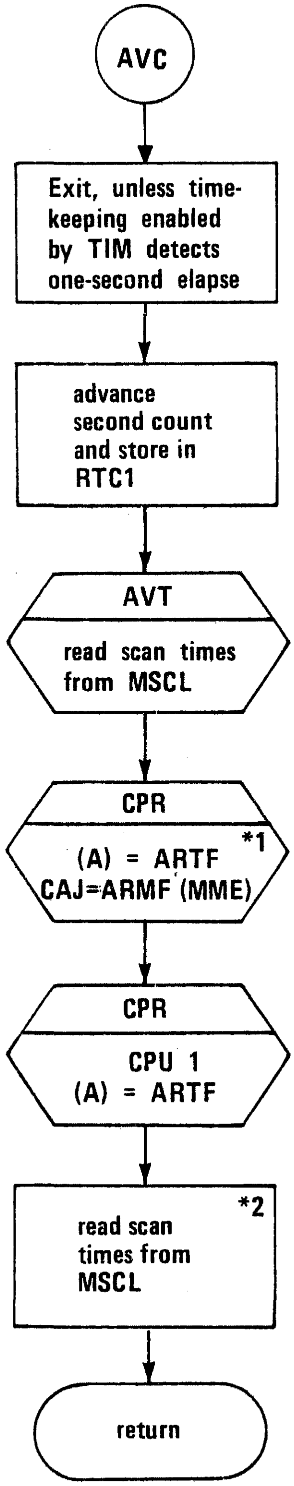
TIME KEEPING

MTR controls all time-keeping activities with routines TIM, AVC, and AVT.

Routine TIM reads the real-time clock and updates RTCL (the central memory real-time clock). This routine must be entered at least once every millisecond. When one second has elapsed, the calls to AVT, ARTF, or ARMF are enabled in routine AVC.

Routine AVC has no time-keeping activity until one second has elapsed. The calls to AVT, ARMF, or ARTF are then enabled by TIM.

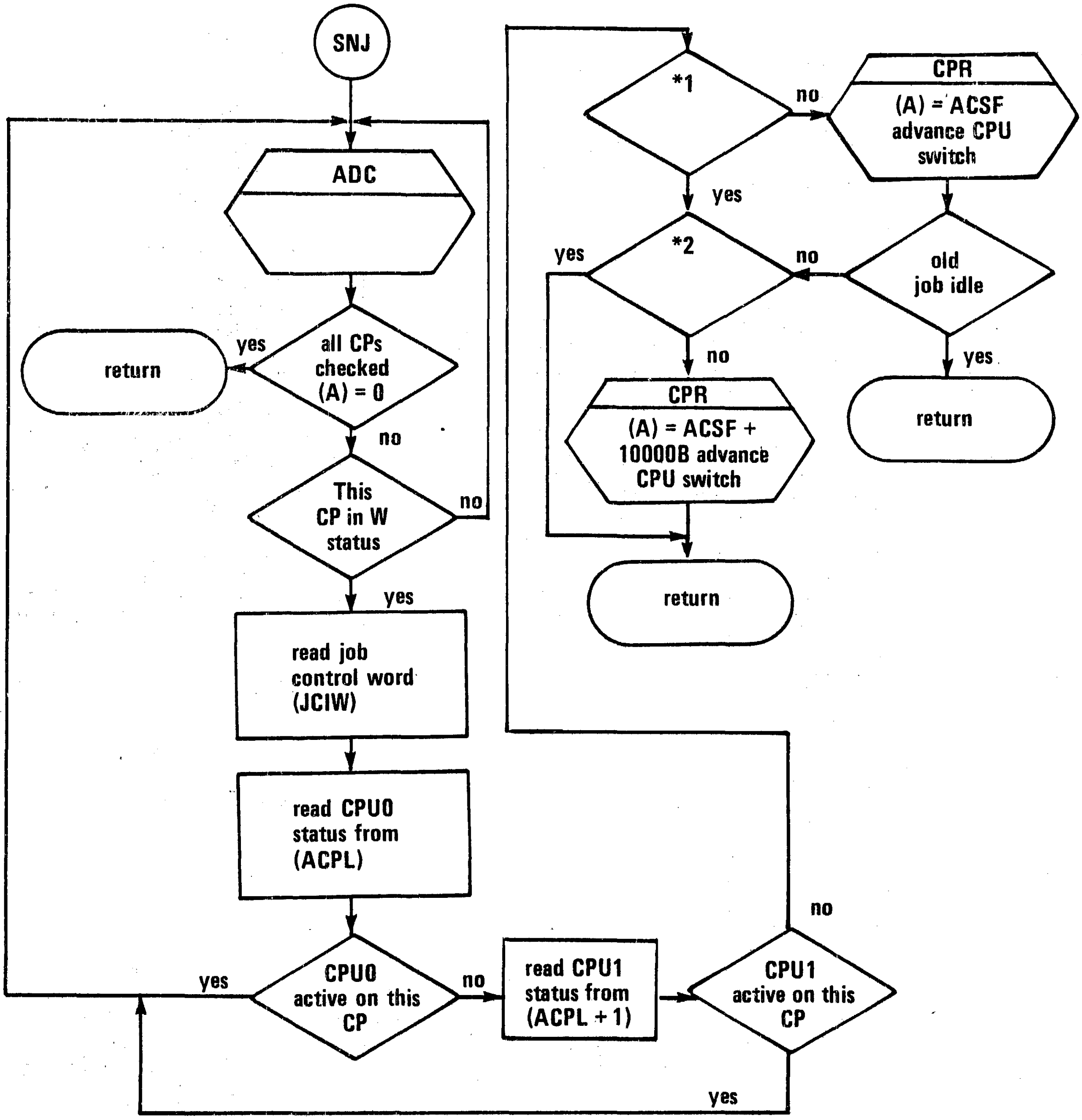
Routine AVT advances the time of day and date in words JDAL, PDTL, TIML, and DTEL in CMR.



*1 Advance CPU 0 time. Accumulated control point time for active control point at CPU 0.

*2 MSCL can be dynamically set from the console. ART reads it every second.

Figure 3-7. AVC Advance Running Times



*1 CPU 0 active job CPU priority greater than this control point priority.

*2 CPU 1 active job CPU priority greater than this control point priority.

Figure 3-8. JSW - Process CPU Job Switching (CPU Slot Time)

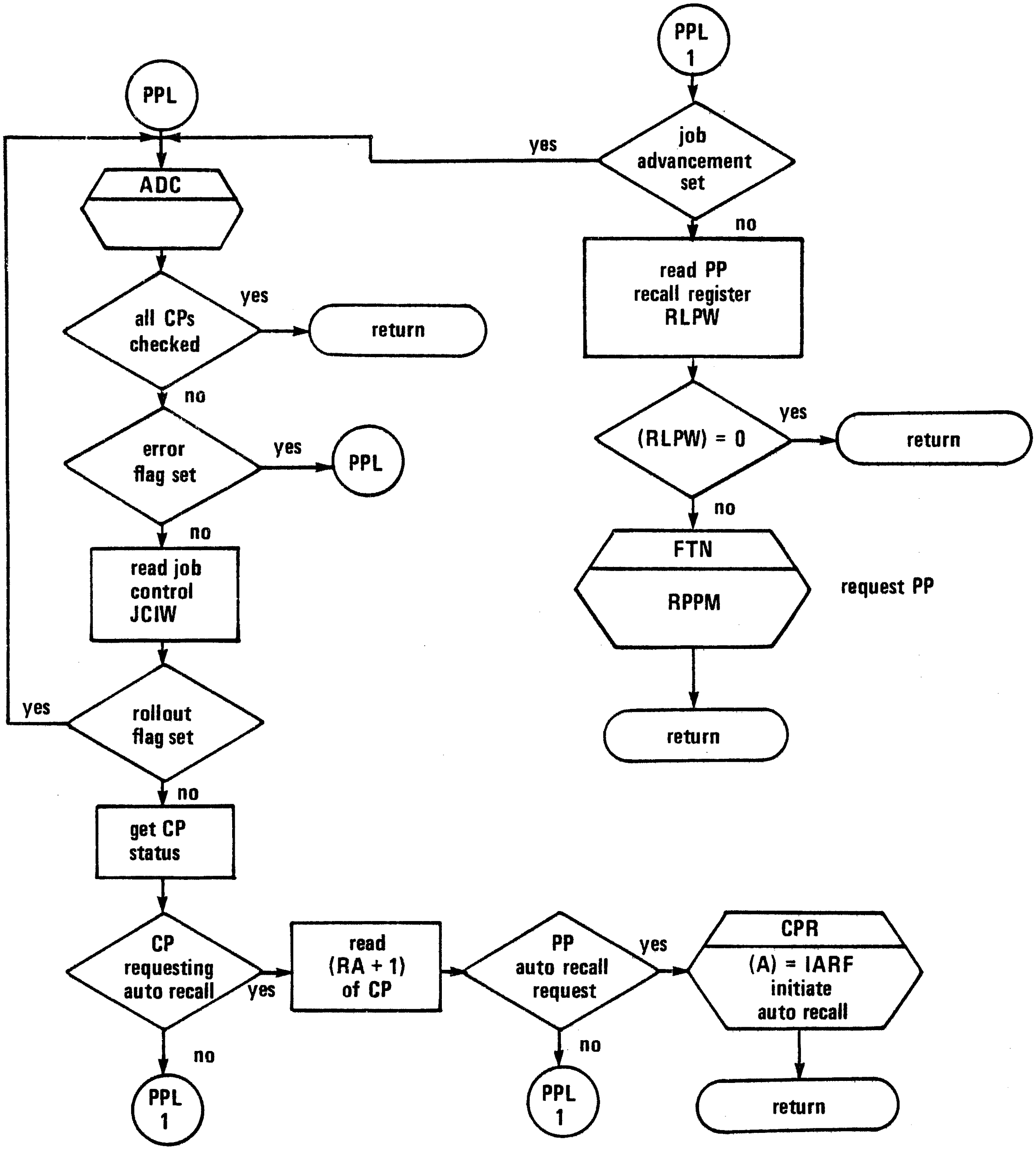
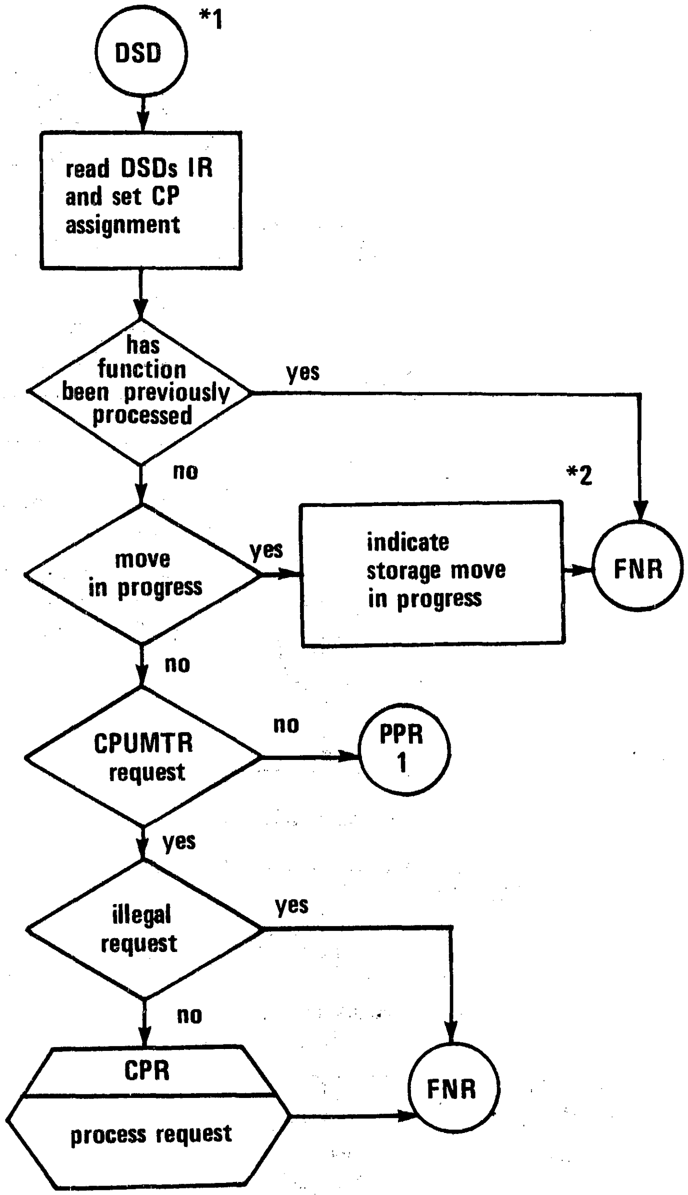


Figure 3-9. PPL - Process PP Recalls

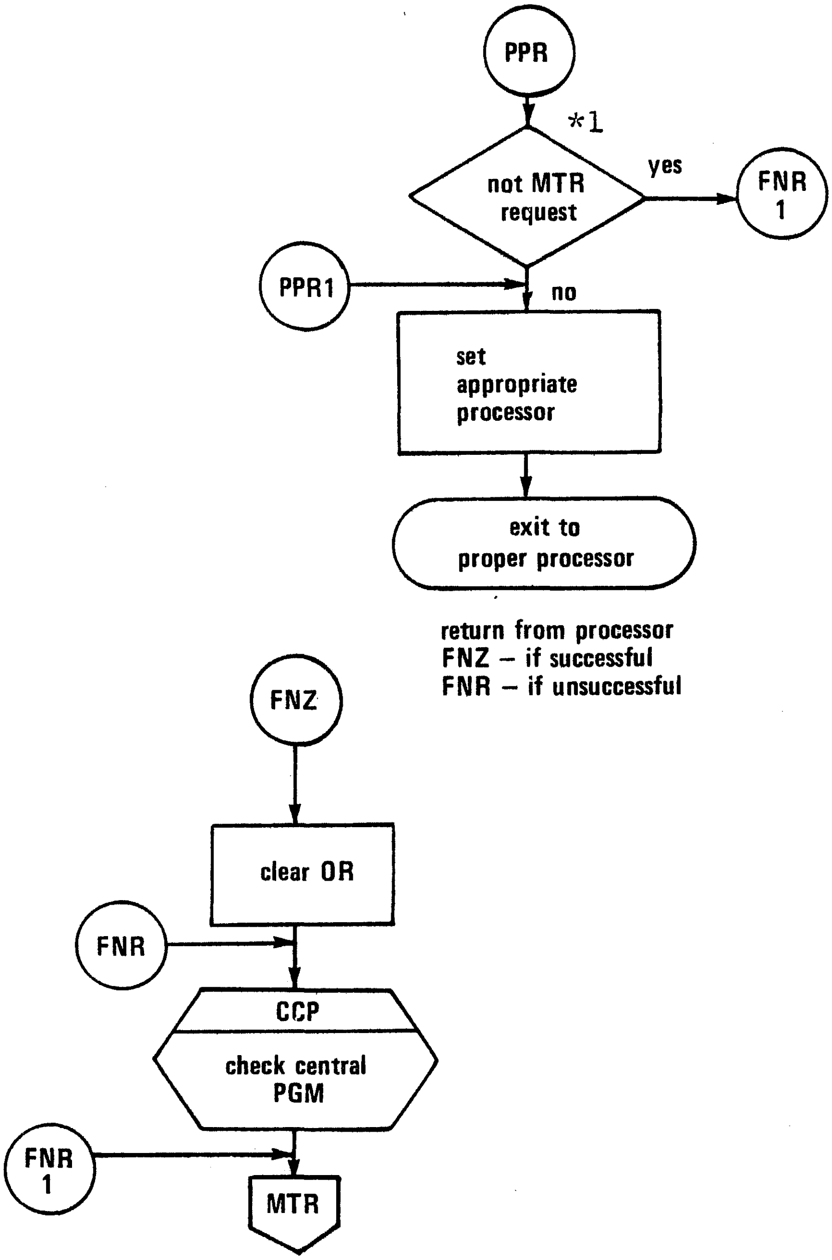
PP function requests are made to MTR by placing the function code in byte 0 of the PP's OR. When the request is complete, MTR clears byte 0 of the OR.



*1 When DSD wants to do an action for a control point (such as n.XXX), it temporarily attaches itself to that control point by placing the control point number in its IR; it then makes the request.

*2 If this control point is moving, the status must be set.

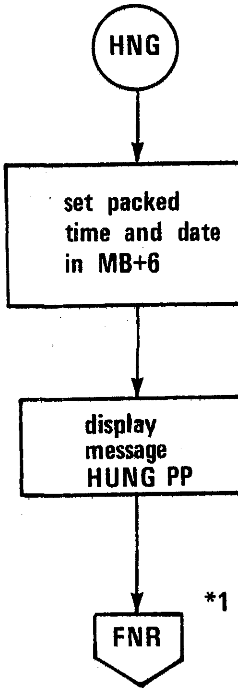
Figure 3-10. DSD PP Function Request



*1 If request illegal then effectively hang PP since OR is never cleared, this will not display PP hung at system PP.

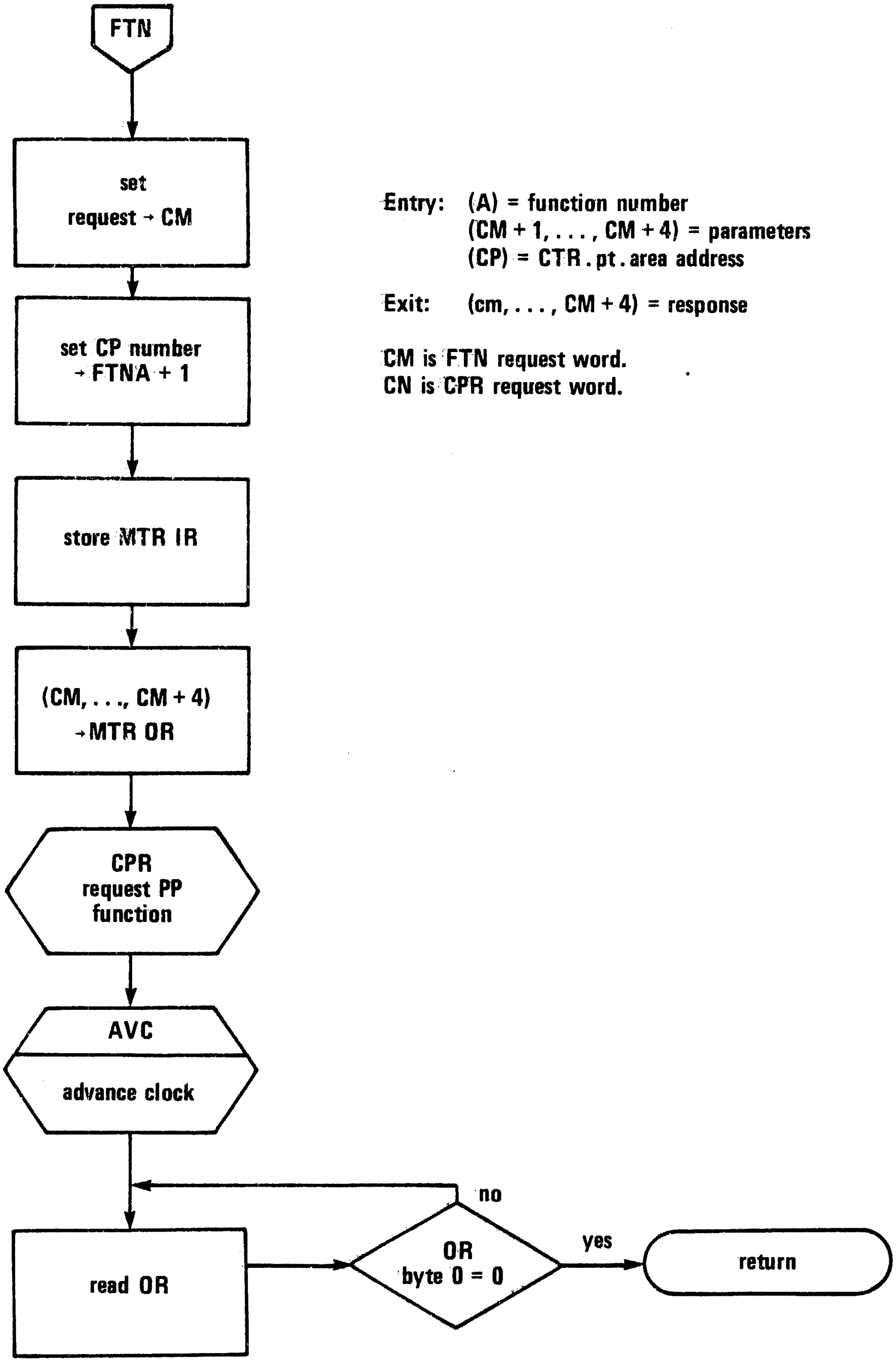
Figure 3-10. DSD PP Function Request (Continued)

If any of the functions requested desire an illegal operation (for example, DCHM drop channel wishes to drop a channel which does not exist) then it will jump to this routine.



*1 Do not clear OR and thereby hang this PP.

Figure 3-11. HNG - Hang PP and Display Message

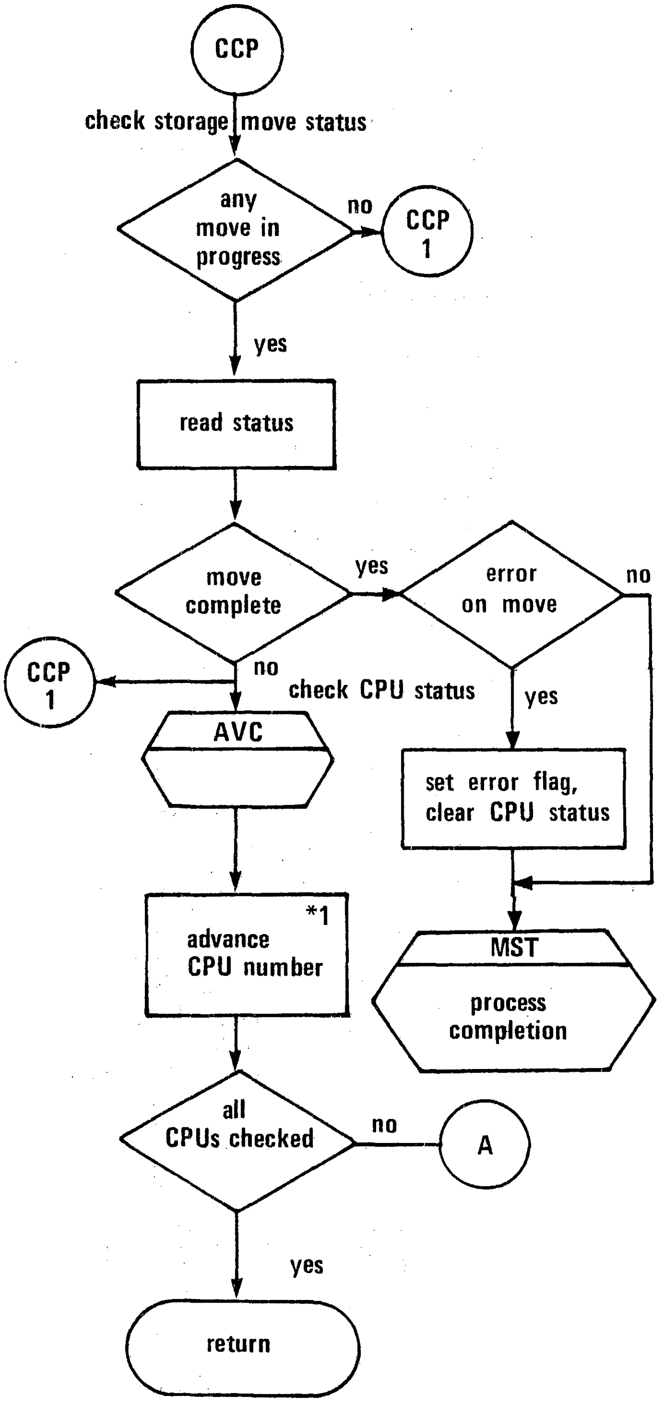


Entry: (A) = function number
(CM + 1, . . . , CM + 4) = parameters
(CP) = CTR . pt . area address

Exit: (cm, . . . , CM + 4) = response

CM is FTN request word.
CN is CPR request word.

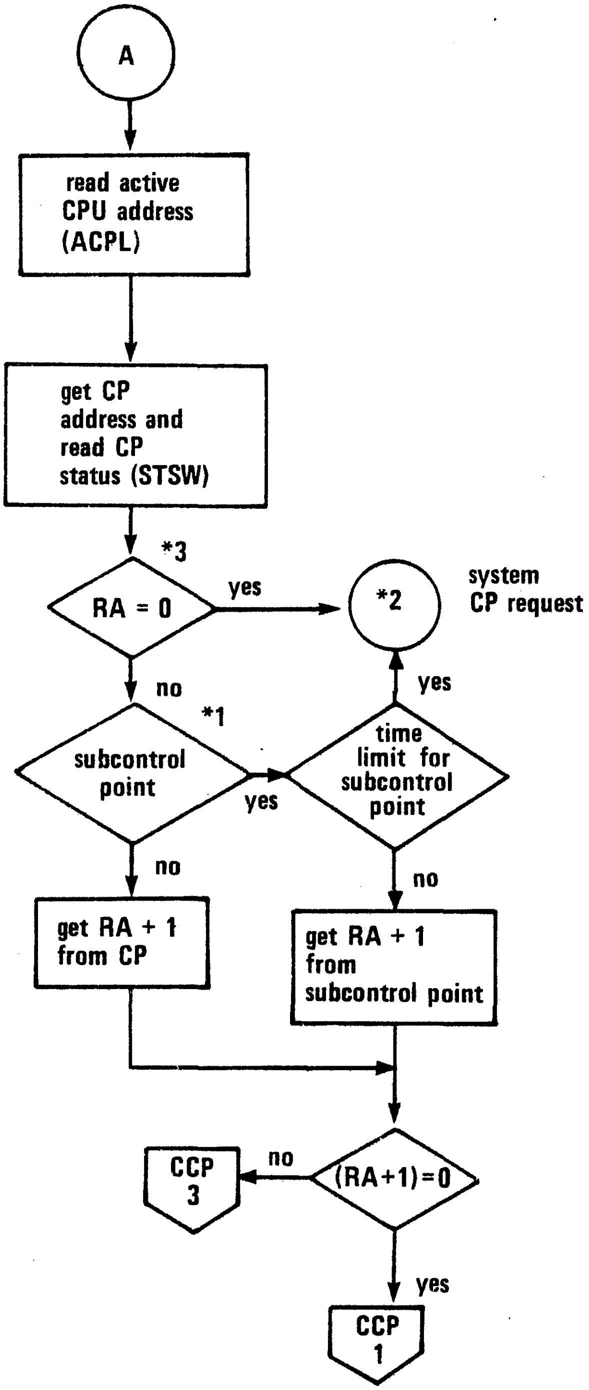
Figure 3-12. FTN - Process Monitor Function



The contents of the RA+1 for the control point at this CPU are checked and CPUMTR is requsted to process the request. The program address (P) is checked and if = 0, CPUMTR is requested.

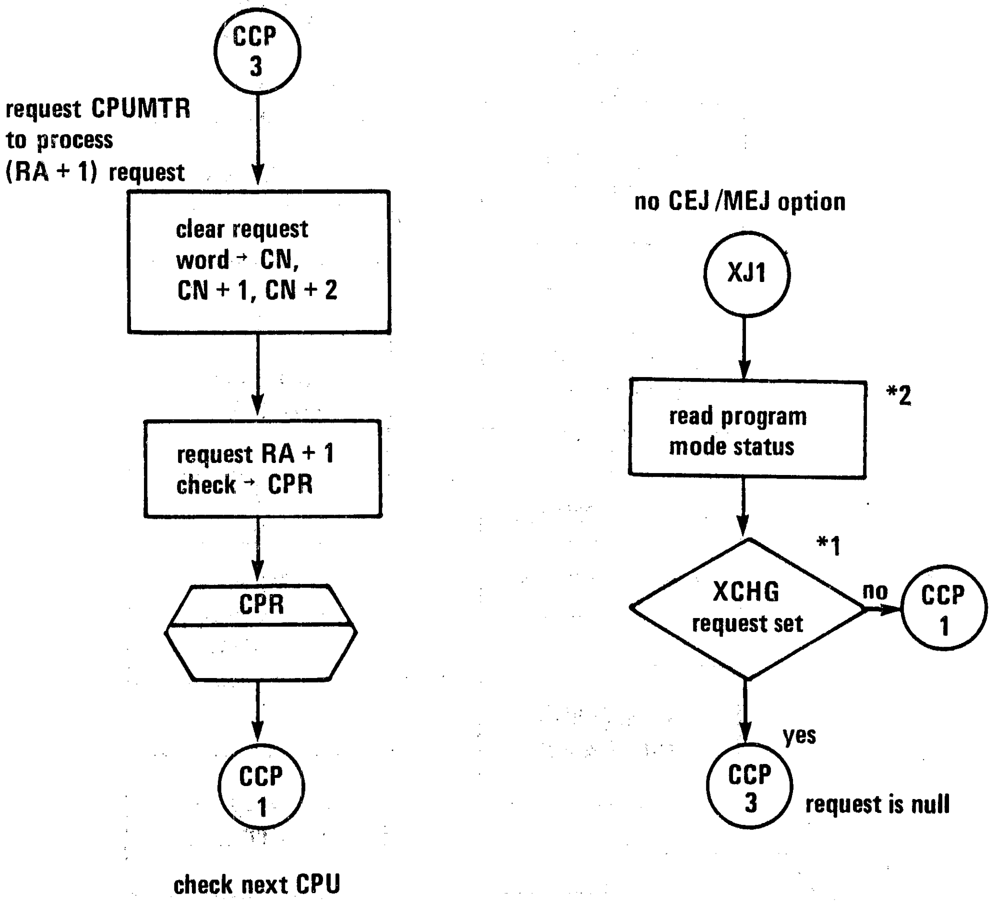
*1 Check active control point in CPU 0; then CPU 1 gets control point number in CPU 0.

Figure 3-13. CCP - Check Central Program



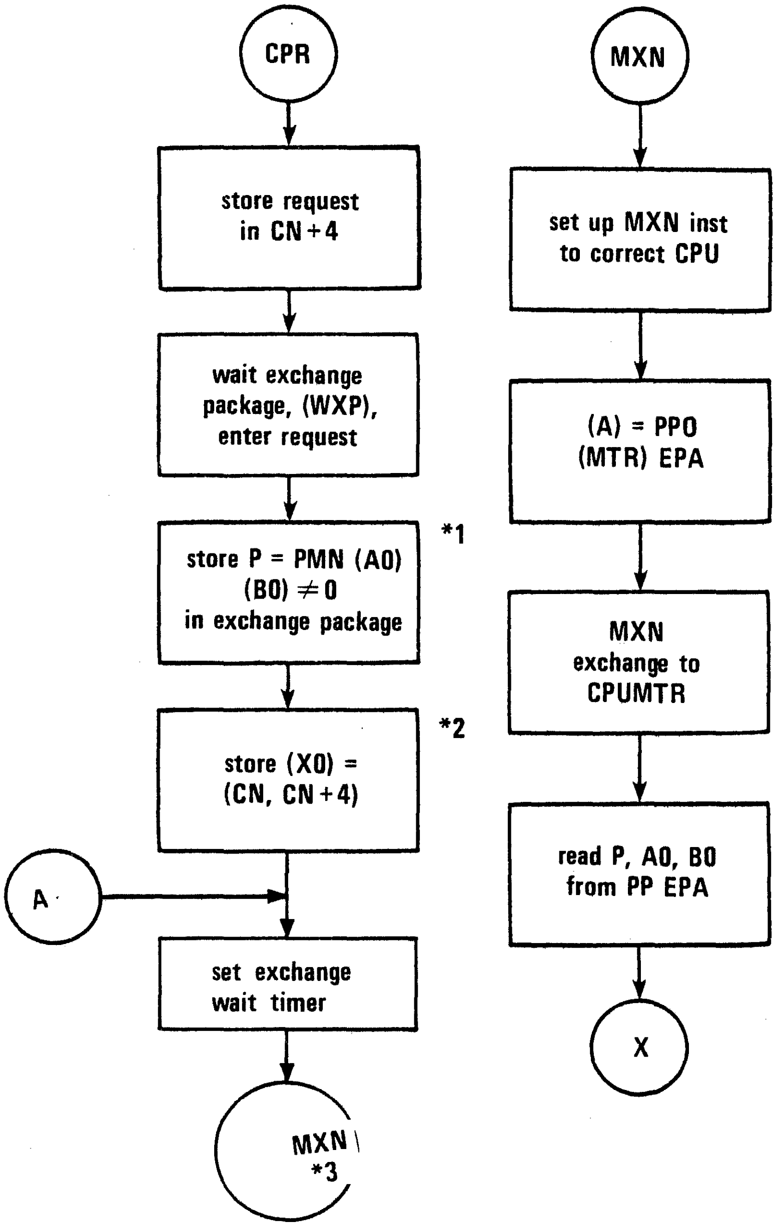
*1 A user control point is running; that is, this is not CPUMTR.
*2 If CEJ/MEJ available, go to CCP1; if not, go to XJ1.
*3 If (RA) = 0, this is CPUMTR and is ignored.

Figure 3-13. CCP - Check Central Program (Continued)



*1 No if (PX)=0; yes if (PX) nonzero. PX is defined in CPUMTR.
*2 Use PR defined in CPUMTR.

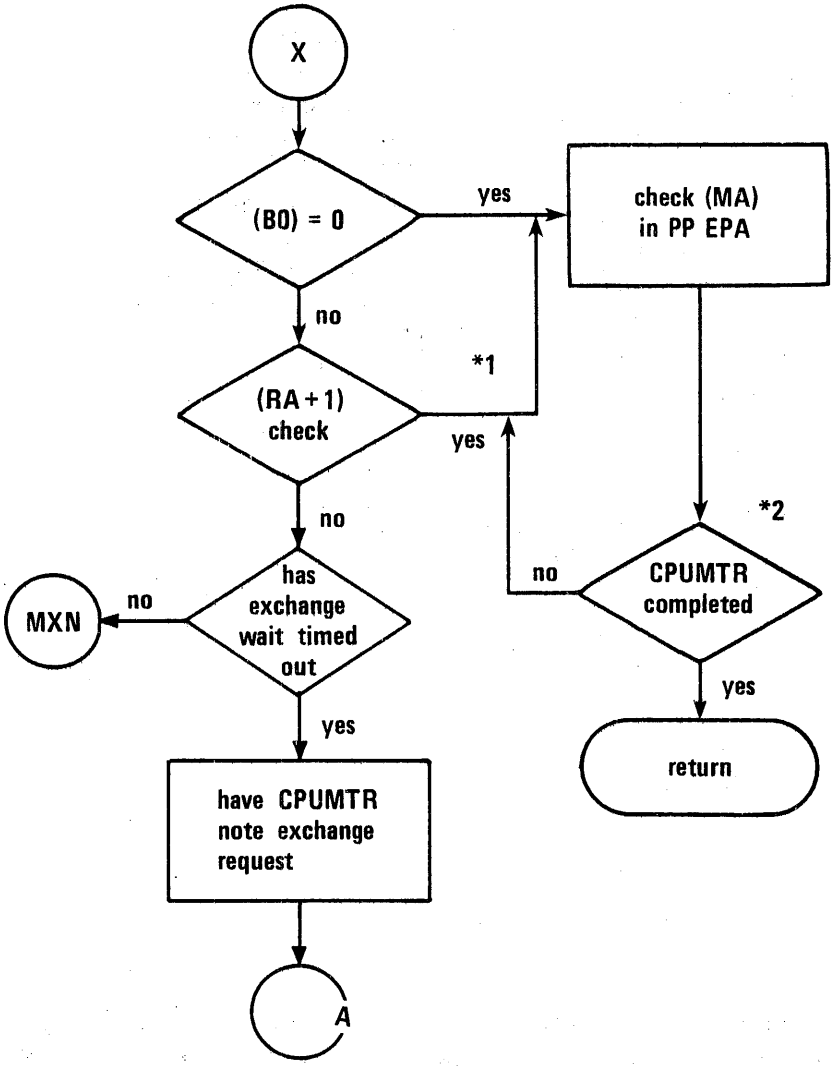
Figure 3-13. CCP - Check Central Program (Continued)



*1 This request will be processed by CPUMTR at PMN.
*2 PMN expects the request in X0.
*3 If CEJ/MEJ option available, use code on this page.

Figure 3-14. CPR-CPUMTR Request Processor

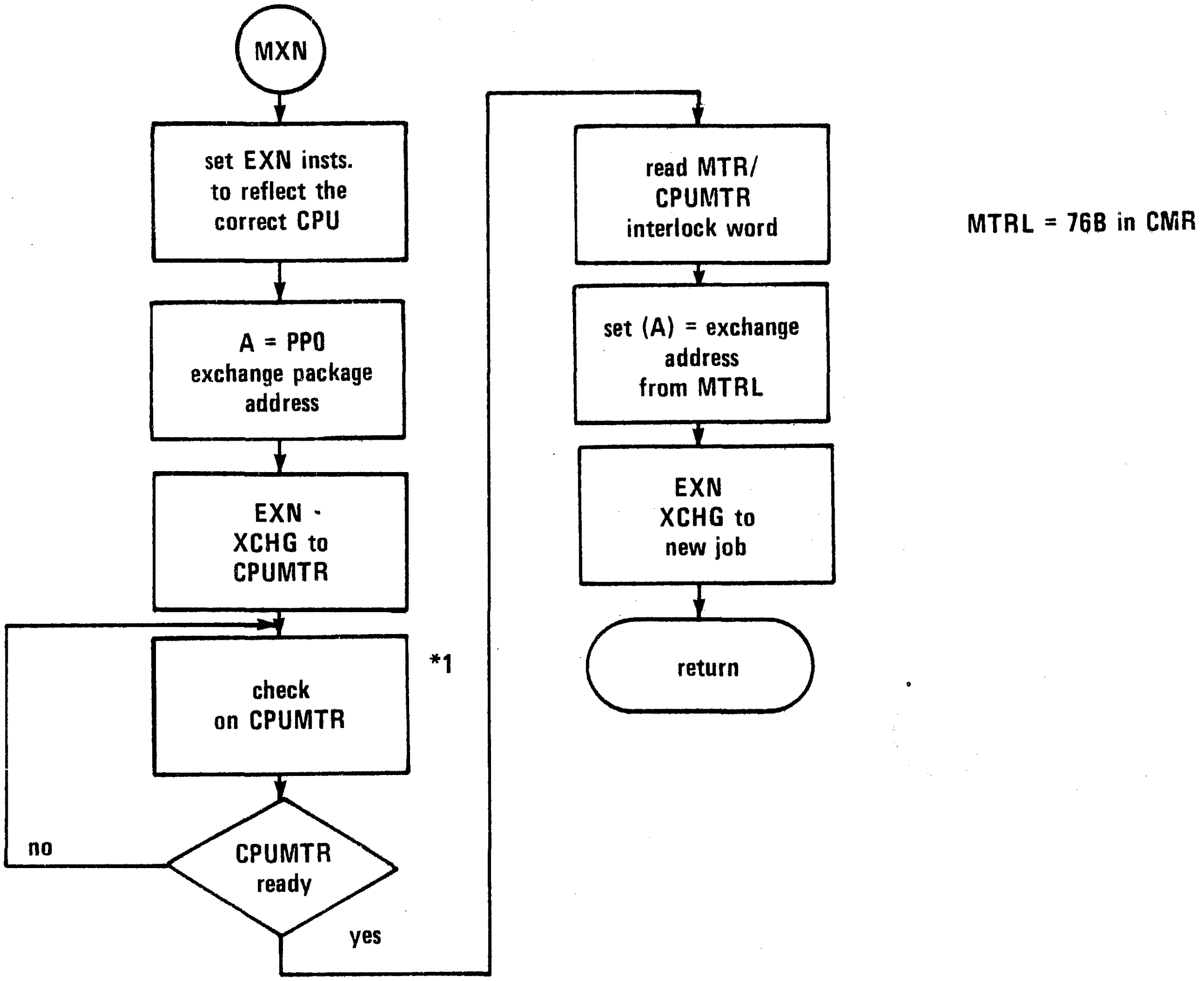
Entry: (A) bits 0-11 = request
12-17 = CPU number
(CN, . . ., CN+2) = parameters



*1 Was this an RA+1 check. If no and exchange occurred, CPUMTR is now running and it will automatically process this request. If not, reissue the exchange.

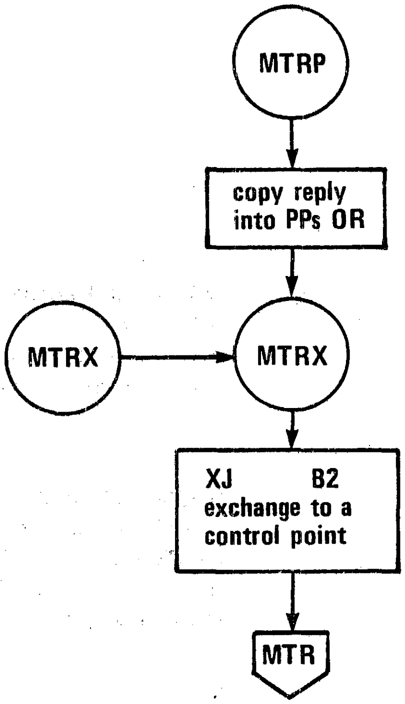
*2 There is a delay loop.

Figure 3-14. CPR-CPUMTR Request Processor (Continued)



*1 When CPUMTR has completed, it places the exchange address of the control point to be started in MTRL and jumps to a one-word idle loop at CPSL = 77B in CMR, which is a zero word; that is, a PS. MTR is doing an RPN 0 and waiting for (P) = CPSL.

Figure 3-15. XCHG - The CPU with CEJ/MEJ Not Available

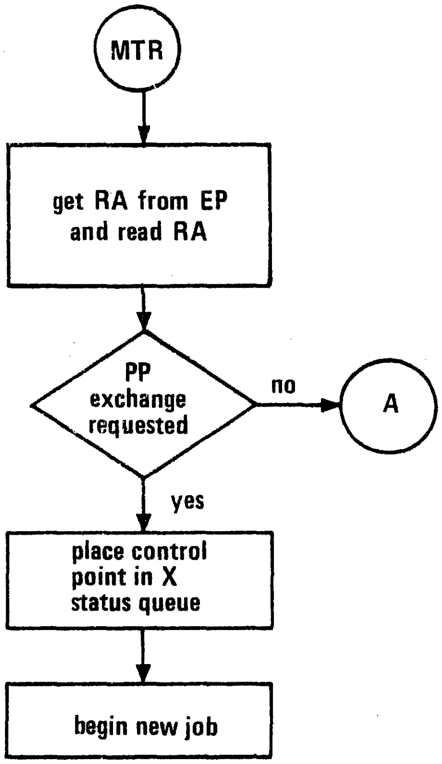


MTRP entry: (X7) = reply word to be set into calling PPs output register

MTRX entry: (B2) = address of exchange package for control point to be exchanged into CPU

exit: (P) = MTR

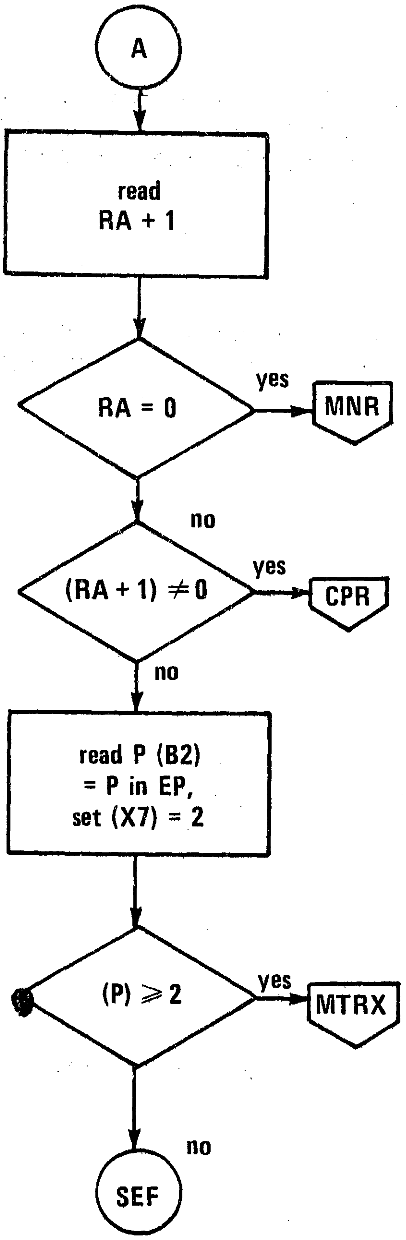
Figure 3-16. CPUMTR Return Points



Entry: (A0) = CPU number (0 or 1)
(B1) = 1
(B2) = address of caller's EP
(B7) = control point area address

If CPn exchanged itself, then (B2) = (B7) and EP will be in CPA. If CPn was exchanged by MTR or some other pool PP, then (B2) = the address of the PP EPA which performed the exchange and (B7) = CPA.

Figure 3-17. MTR - Exchange Entry From A CPU Program



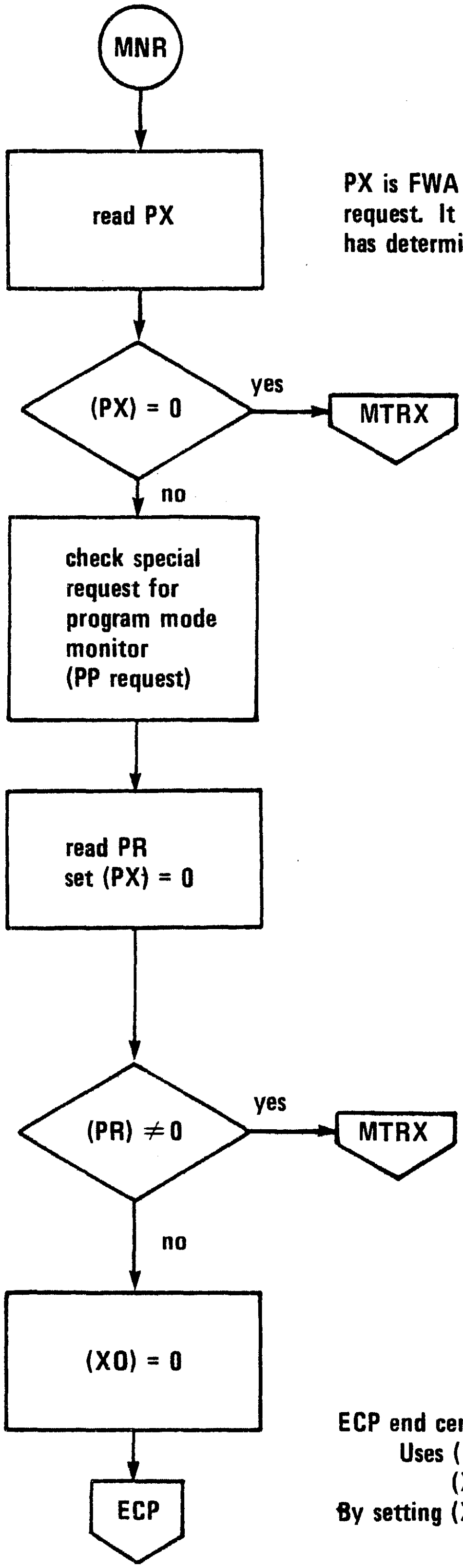
Check for monitor request. Is this exchange a CPUMTR EP or a CP EP.

Process the RA + 1 request.

Exchange CP back in. CP wanted a short pause.

Set error flag CPU detected on ARITH error. Uses (B7) = CPA, (X7) = error code. SEF will abort the CP program on ARITH error.

Figure 3-17. MTR - Exchange Entry From A CPU Program (Continued)



PX is FWA of CPUMTR code and is the program mode exit request. It is set when program mode portion of CPUMTR has determined that this CP job is complete.

This is to determine if a CP or if the system CP (CPUMTR program mode) was interrupted.

PR is a pointer to a stack of requests for program mode execution (that is, system CP queue).

Let system CP continue running.

ECP end central program.
Uses (B7) = control point area address
(X0) = status bits
By setting (X0) = 0 we set CP status = 0 or not active.

Figure 3-18. CHECK - For System CP Request

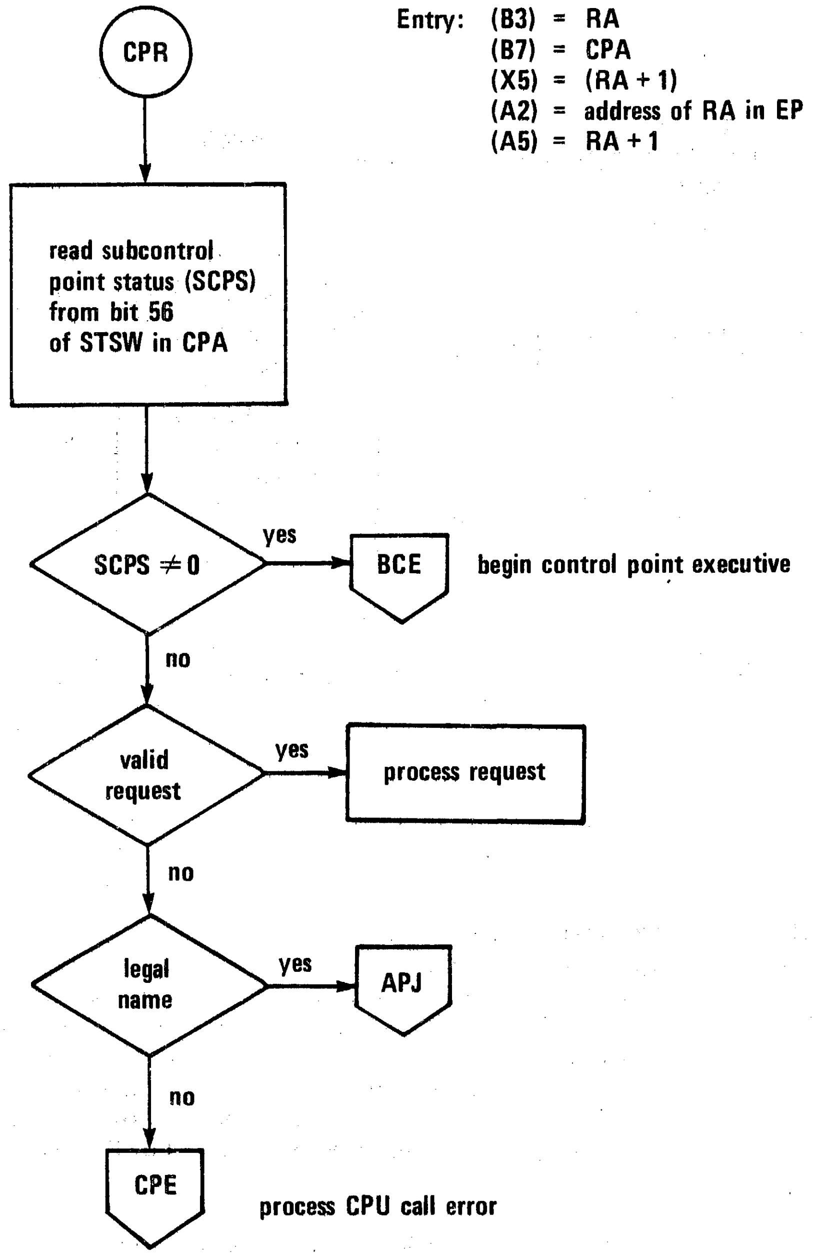
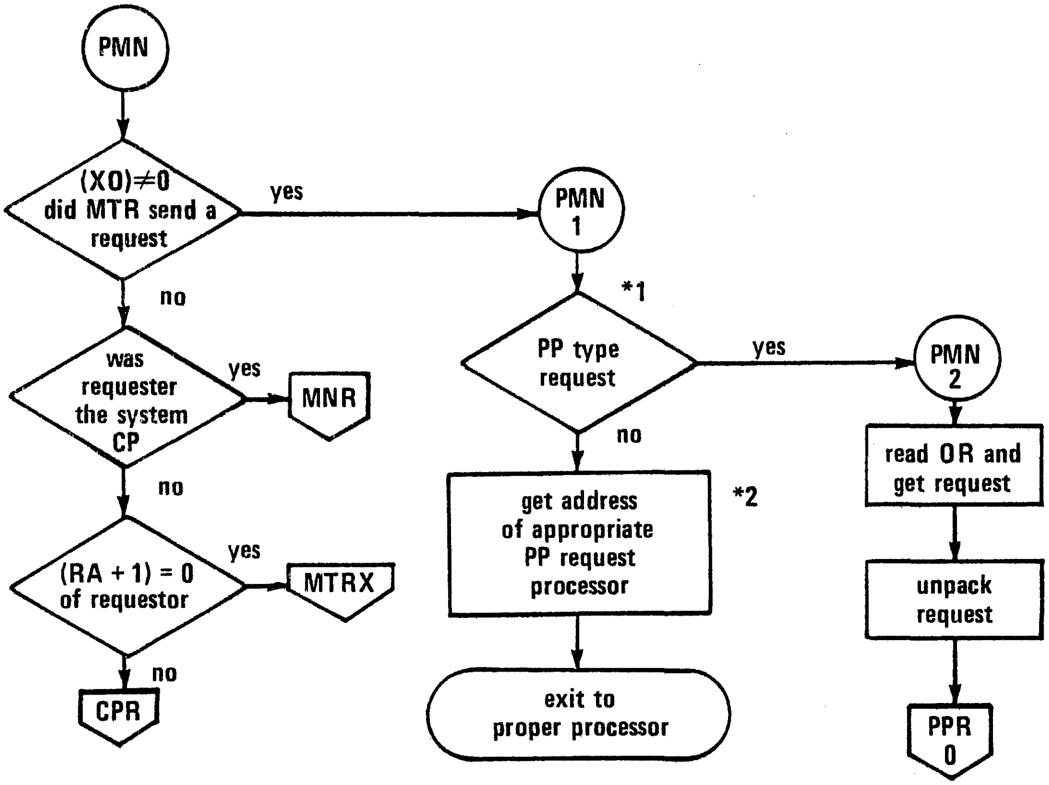

Entry: (B3) = RA
(B7) = CPA
(X5) = (RA + 1)
(A2) = address of RA in EP
(A5) = RA + 1

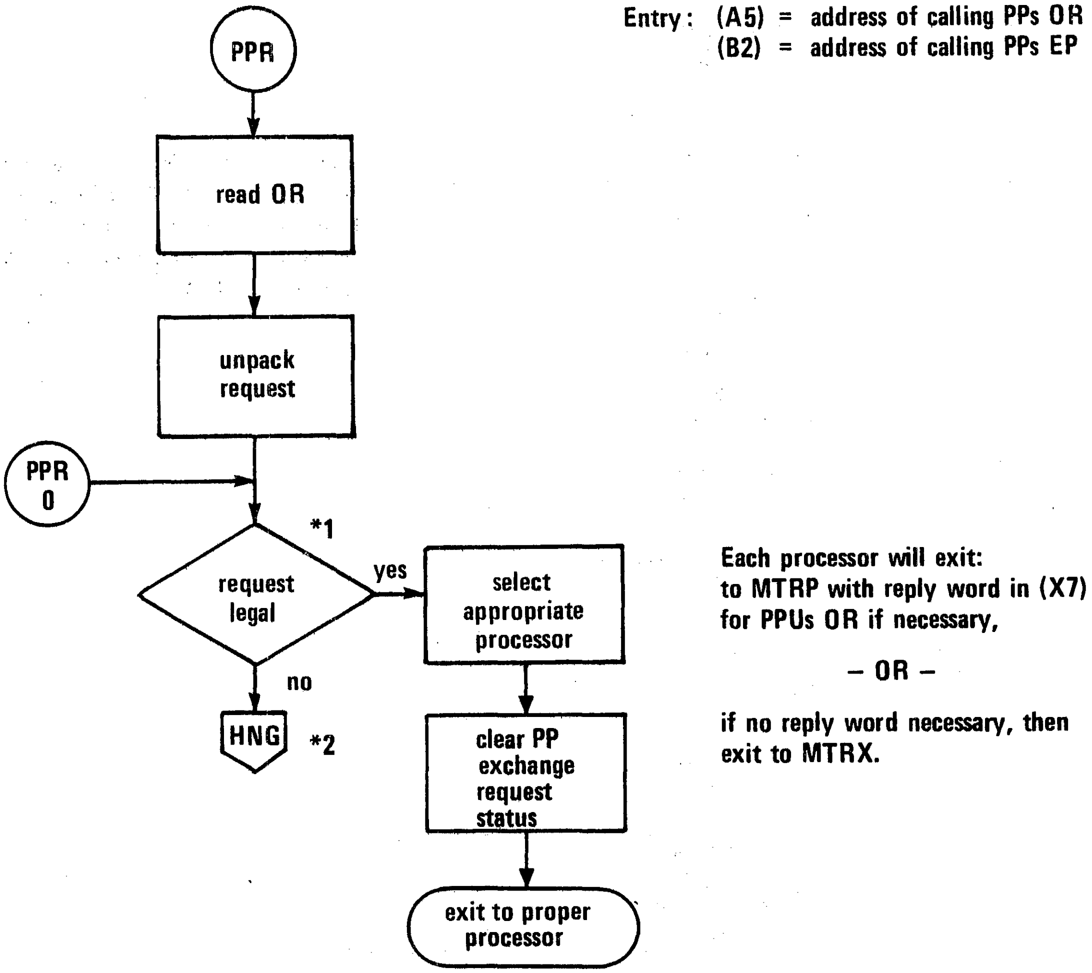
Figure 3-19. Process - RA+1 Requests



*1 MXPF is maximum number a MTR request can be, so test is (X0) - MXPF > 0, then go to PMN2.

*2 Those processors which require program mode CPUMTR will exit via EPR. EPR will check to see if the system control point was interrupted for this request and if so, will exit to MTRX. If control point n was interrupted, then it will exit to BCP1, which will place this now deactivated control point into W status, and then exit to MTRX.

Figure 3-20. PMN - Exchange Entry From MTR



Entry: (A5) = address of calling PPs OR
(B2) = address of calling PPs EP

Each processor will exit:
to MTRP with reply word in (X7) for PPUs OR if necessary,

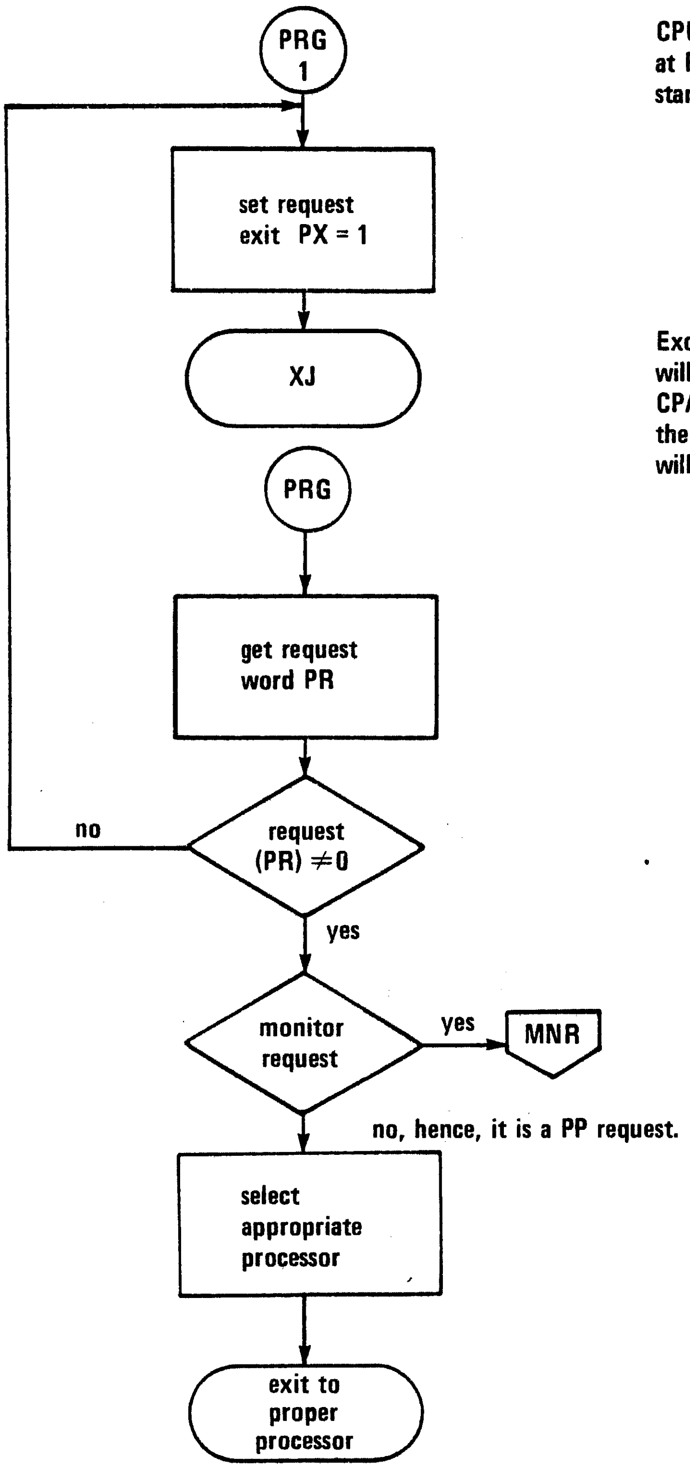
– OR –

if no reply word necessary, then exit to MTRX.

If the processor requires program mode CPUMTR, the macro PPR will generate a queue entry and set up the exchange package, then jumps to PRG, sees no request, and jumps to PRG1.

*1 Check to see if request (which is a number) is larger than the maximum.
*2 Hang PP by not clearing OR, and display message PP HUNG at system control point.

Figure 3-21. PPR - Exchange Entry for Pool PPs



CPUMTR starts the program mode portion at PRG in program mode. This is the standard exit for program mode CPUMTR.

Exchange to CPUMTR in monitor mode. This will force (P) = PRG in EP in the system CP CPA, so that the next time CPUMTR starts up the system CP, execution in program mode will begin at PRG.

Figure 3-22. PRG - Exchange Entry for System CP (Program Mode CPUMTR)

IDL, IDL1 - CPU0 AND CPU1 IDLE LOOPS

The exchange package for IDL is loaded at the end of CPUMTR

IDL1 and its exchange package are located in the dual CPU block.

```
-----------------------------------
|(P)  = 2                         |
|(RA) = location of IDL in CPUMTR |
|(FL) = 5                         |
|(MA) = location of this EP       |
|(EM) = 7007                      |
|                                 |
|all other register = 0           |
-----------------------------------
```

```
-----------------------------------
|(P)  = 2                         |
|(RA) = location of IDL1 in CPUMTR|
|       DCP block                 |
|(FL) = 5                         |
|(MA) = location of this EP       |
|(EM) = 7007                      |
|                                 |
|all other register = 0           |
-----------------------------------
```

```
     Program IDL                       Program IDL1

0000 IDL   CON  0    (RA) for idle     IDL1  CON  0
                     routines

0001       CON  0    (RA+1=0) for            CON  0
                     idle routine

                     never any
                     requests

0002       EQ   2    jump to itself          EQ   2
```

Program IDL and IDL1 runs until a PP or MTR interrupts them and exchanges CPUMTR into the CPU. If CPUMTR finds no other jobs to run, it exchanges IDL or IDL1 back into the CPU.

CPUMTR SEGMENTATION

A significant amount of code is required in CPUMTR to support a multimainframe environment which is not needed by sites not utilizing this feature. Since CPUMTR resides in central memory, it is desirable to provide a mechanism whereby code associated with a particular feature (in this case multimainframe) may be optionally loaded or discarded at system deadstart time.

CPUMTR accomodates blocks of code that may be optionally loaded. These blocks of code are placed into labeled common by USE cards. Blocks come in two types. One type always requires the presence of an associated block and one of the two blocks will always be loaded. The other type of block has no associated block and with either be loaded or discarded by CPUMLD. For example, if OMMF is the name of an associated block which is loaded when MMF processing is desired, then OMMF is loaded in its place if MMF processing is not desired. The convention therefore is to place a zero in front of the block name for the option-not-present block. Any given feature may have as many blocks associated with it as is necessary with any number of them being loaded.

A CPU program, CPUMLD, loads the desired CPUMTR blocks. CPUMLD is a simple relocating loader which reads in and loads the segments required to utilize any optional feature selected during the pre-deadstart process. This covers the case of wanting one set of code for environment A and another set for environment B. STL loads CPUMLD and CPUMLD issues requests to STL to read in CPUMTR from the deadstart tape.

EXCHANGE JUMPS

An installation may make use of the optional hardware instructions MXN (monitor exchange) and XJ (exchange jump) or EXN (exchange). NOS requires either the combination of MXN/XJ or EXN.

Exchange jumps use an exchange package (refer to section 2).

CENTRAL PROCESSOR MONITOR

System functions are normally handled by the monitor located in a peripheral processor. The CYBER 170/70 computer systems are equipped with certain hardware capabilities to effectively implement monitor activities in the central processor. Since the central processor can reference extended core storage directly for service routines, programs, and data, a central processor monitor program to handle these and other functions is faster and more efficient than a monitor residing in a peripheral processor.

The hardware elements of the CYBER 170/70 system which provide the essential capabilities for implementing a central processor monitor are described in the following paragraphs.

Monitor Address Register (MA)

Contained in the exchange jump package (bits 53 through 36 of word 6) is an 18-bit monitor address. Just as other central processor operational registers are loaded during an exchange operation, so is the monitor address register loaded with the 18-bit monitor address. This monitor address is the starting address of the exchange package for an ensuing central exchange jump instruction (except when the monitor flag bit is set).

Monitor Flag Bit

The central processor has, in the central memory control section, a monitor flag bit. A master clear (deadstart) clears the monitor flag bit. Any action thereafter on this bit is via the monitor exchange or the central exchange jump instructions. (There is no instruction with which to sample the status of this bit directly and/or independently of these instructions.)

Mode	Flag Bit	CPU
Monitor mode	1	Not interruptable
Program mode	0	Interruptable

Central and Monitor Exchange Jump Instructions

With the CEJ/MEJ option two instructions exist for central processor monitor implementation. The first, XJ, is executable by the central processor and the second, MXN, is executable by the peripheral processors. These instructions are as detailed in the COMPASS Reference Manual.

The XJ instruction unconditionally exchange jumps the central processor, regardless of the state of the monitor flag bit. The instruction action differs, however, depending on whether the monitor flag is set or clear. Operation is as follows:

- Monitor flag bit clear

  The starting address for the exchange is taken from the 18-bit monitor address register. This starting address is an absolute address. During the exchange, the monitor flag bit is set (MF=1)

- Monitor flag bit set

  The starting address for the exchange is the 18-bit result formed by adding K to the content of register Bj. This starting address is an absolute address. During the exchange, the monitor flag is cleared.

The MXN instruction, typically used to initiate central processor monitor activity, is a conditional exchange jump to the central processor. If the monitor flag bit is set, this instruction acts as a pass instruction. The starting address for this exchange is the 18-bit address held in the peripheral processor A register. (The peripheral processor program must have loaded A with an appropriate address prior to executing this instruction.) This starting address is an absolute address.

In an installation without the MXN/XJ instruction set, the EXN is the only exchange instruction available. It is a PP initiated exchange jump which occurs independently of the mode of the CPU and has no effect on the CPU mode. MTR is the only PP program that may perform an EXN; it must simulate the MXN for all PPs in the system and simulate XJ for the central processor. When MTR detects a request for CPUMTR in a PP output register, it will EXN to the exchange package for the pool PP which desires the exchange jump.

NOTE

PP memory instruction layout is the same as MXN.

Programming Notes

Any exchange to the exchange package loads the contents of word 6 into the monitor address register (other operational registers are similarly loaded). Thus, any ensuing XJ instruction using the contents of the monitor address register as a starting address uses those contents as loaded.

The exchange packages for entering the central processor monitor should usually have the reference address (RA) equal to 000000 and the field length equal to central memory size.

Since the monitor flag bit cannot be directly sampled, a program cannot directly determine its state; hence, success in performing a peripheral processor monitor exchange cannot readily be predicted. Further, program control always is given to the next instruction, whether or not the exchange is honored.

Table 3-6 summarizes the operational differences between the normal exchange jump instruction EXN and the monitor and central exchange jump, MXN and XJ.

TABLE 3-6. EXCHANGE INSTRUCTION DIFFERENCE

			Operational Differences	
	Instruction	Conditional/ Unconditional	Effect on Monitor Flag Bit	FWA of Exchange Package in CM
No CEJ/ MEJ	EXN 260 (normal peripheral processor exchange jump)	Unconditional	No effect on flag	Peripheral processor A register
	MXN 261 (peripheral processor monitor exchange jump)	Conditional (occurs only if monitor flag bit is clear; passes if flag is set)	Sets flag	Peripheral processor A register
With CEJ/ MEJ	XJ 013 (central exchange jump) with monitor flag bit clear	Unconditional	Sets flag	Central processor monitor address register
	XJ K+(Bj) 013 (central exchange jump) with monitor flag bit set	Unconditional	Clears flag	Address formed by K+(Bj)

To determine whether the MXN took place:

1. Set B0 (bits 0-17 of word 0) in the exchange package to 7777.
2. Initiate the monitor exchange (261).
3. Read B0 from the exchange package in central memory. If the monitor exchange was honored, B0 in the exchange package will equal 000000. If the instruction passed, this location still holds 7777.

Different exchange packages should be used for central processor exchanges and peripheral processor exchanges. This aids software determination of which of two jumps (central or monitor exchange) was executed when both were initiated at approximately the same time.

Simultaneous exchange requests are resolved in favor of the central processor.

The state of the monitor flag bit has no effect on the operation of the normal PP exchange jump (260); nor has this instruction any effect on the flag.

In addition, there may be CPUMTR requests which require more CPU time than it is feasible for CPUMTR to use in monitor mode and still ensure smooth system flow. For these requests, such as DTKM (drop tracks), CPUMTR will queue them at the system control point and exchange jump to this control point. The system control point operates in program mode and is treated as any other user program. If the system control point is interrupted with another long request, the request is placed in the system control point queue and the system control point is restarted. The system control point can be interrupted by any MXN from a PP. However, because its CPU priority is the highest in the system (100), it will always get the CPU back immediately. No other control point will get the CPU if the system control points wants it.

Table 3-7 shows the correspondence between a control point, control point address, and the exchange package MA for a system configured to have 17B control points.

Table 3-8 shows all the system exchange packages and the entry points into CPUMTR.

A control point will always have (MA) equal to its exchange package address. Additional exchange packages are provided for the two idle routines, subcontrol points, disabled central exchange, return package, disabled central exchange program, and a simulated exchange exit to monitor mode. These packages are generated at the end of the CPUMTR code. PP0, MTR's exchange package, is not contiguous with the other PP exchange packages.

FLOW OF EXCHANGES

The flow of exchanges are illustrated and explained in Figures 3-25 through 3-28. The four types of exchanges are:

- Pool PP
- MTR
- Control point program
- System control point

TABLE 3-7. CONTROL POINT/EXCHANGE PACKAGE CORRESPONDENCE

Control Point	Address	Exchange Package MA
1	200	200
2	400	400
3	600	600
4	1000	1000
5	1200	1200
6	1400	1400
7	1600	1600
10	2000	2000
11	2200	2200
12	2400	2400
13	2600	2600
14	3000	3000
15	3200	3200
16	3400	3400
17	3600	3600
20 (System)	4000	4000

TABLE 3-8. SYSTEM EXCHANGE PACKAGES

	PPUs*2	PPU Monitor	Control Point n+1	Control Points 1 thru n	Subcontrol Points and Idle Programs
Graphic Representation	PXP PPU (PP2) Exchange Package ● ● ● PPU(PPn) Exchange Package	MXP PPU Monitor Exchange Package (PP0)	SXP System Control Point n+1 Exchange Package	200B Control Point 1 Exchange Package ● ● ● n*200B Control Point n Exchange Package	SCX Sub CP EP1 SCX1 Sub CP EP2 IXP IDLE CPU0 IXP1 IDLE CPU1
Significant Content	P=PPR MA=zero B2=address of PPi EP (PXP+(i-2)*21B)	P=PMN MA=zero B2=MXP	P=PRG MA=System Control Point Area Address =SXP	P=CP Prog P address MA=This Control Point Area Address =addr. of CPi XJPKG [i*200B]	Sub CP P= MTR MA=SCX SCX1 B2=SCX, SCX1 IDLE P= idle Loop addr. (IDL,IDL1) MA=IXP, IXP1
Size, Numbers and Location	21 words per pkg. Up to 18 pkgs.These start at end of CPUMTR code*	20 words for this pkg.This is at the end of CPUMTR	First 20 of system control point area in CMR	First 20 words of each control point area in CMR	20 words for each package.
Symbolic address	CPUMTR address PXP	CPUMTR address MXP	CPUMTR address SXP	200B 400B . . . n*200B	CPUMTR address SCX and IXP SCX1 IXP1

* The 21B words spaces the packages so that no bank conflicts will arise when PPs access them.

In figure 3-23, assume the CPU is active with control point n and monitor flag zero. If monitor flag is equal to 1, then the exchange does not take place. PPn builds a CPUMTR exchange package in its exchange package area.

Note

> CPUMTR will exit to MTRX by executing an XJ B2. MTR follows MTRX; therefore, after the exchange (P)=MTR in the CPUMTR and exchange package in the PPn exchange package area.

Figure 3-24 is the same as the pool PP request except that (P)=PMN and (X0) equals the request in the MTR exchange package area.

In figure 3-25, control point n is running in the CPU (monitor flag zero), the monitor address is the address of control point n, and the control point address equals the exchange package first word address.

Figure 3-26, is the system control point program mode.

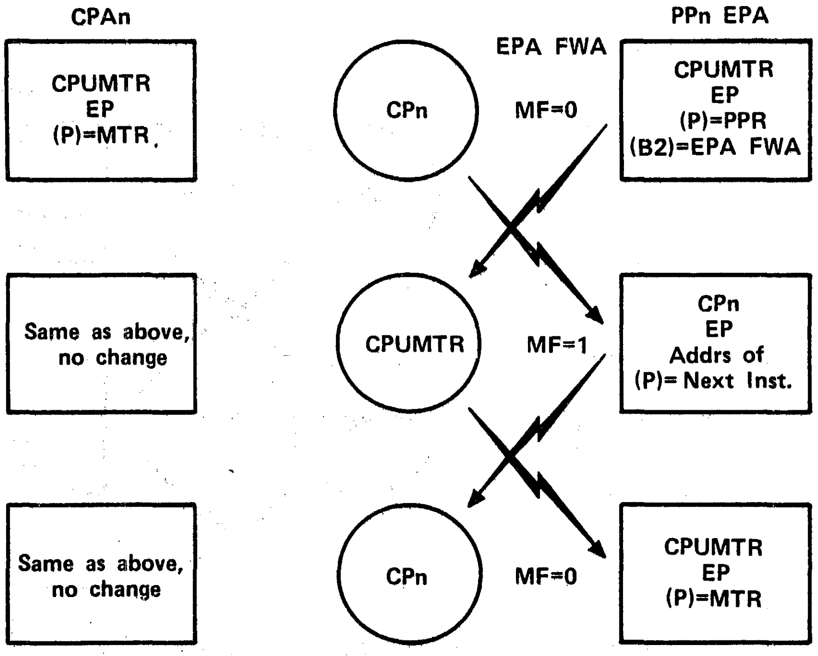
Note

> The system control point can be interrupted by a PP program. In this case the PPn exchange package area contains the system control point exchange package of which (P) equals the address of the next instruction to execute (not PRG).

Table 3-9 illustrates the relationships of the monitors, pool PPs, and control points. .

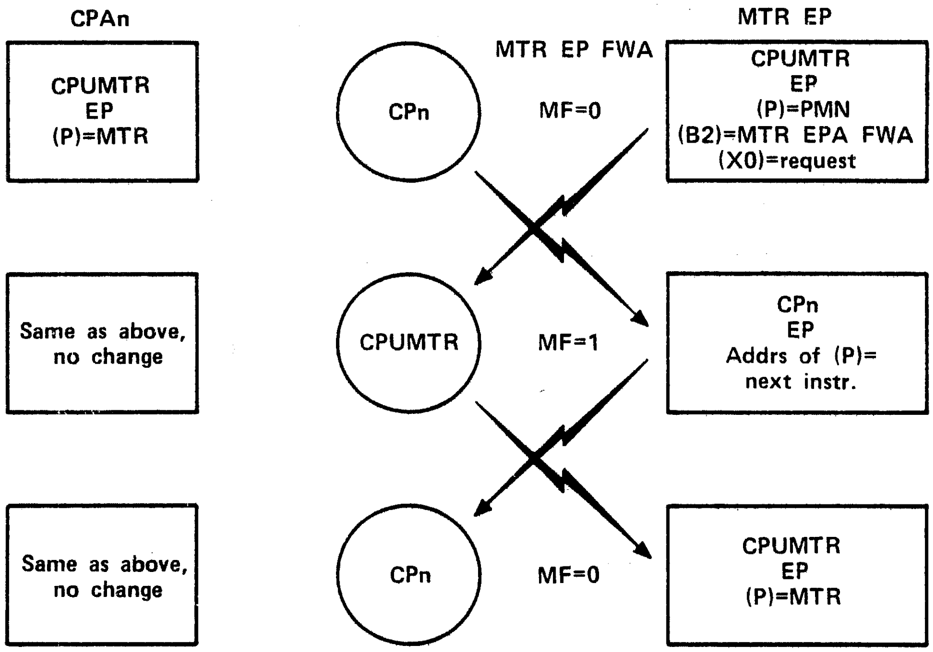
TABLE 3-9. MONITOR, POOL PP, CONTROL POINT RELATIONSHIPS

Type of Exchange	Initiated by	Action	Request to	Reason for Request	Location of Request	Final Disposition
Control Point	Control Point Program	Request	CPUMTR	Needs help	RA+1	CPUMTR/PP
System Control Point	Program Mode CPUMTR	Request	CPUMTR	Needs action from CPUMTR	PX	CPUMTR/PP
Pool PP and MTR	Pool PPs/ MTR	Request	CPUMTR	Needs help or interlock function 35-71.	OR	CPUMTR/PP
Pool PP and MTR	Pool PPs	Request	MTR	Needs help or interlock function 1-34.	OR	MTR
MTR	MTR	Special Request	CPUMTR	Needs help	X0 in EP	CPUMTR



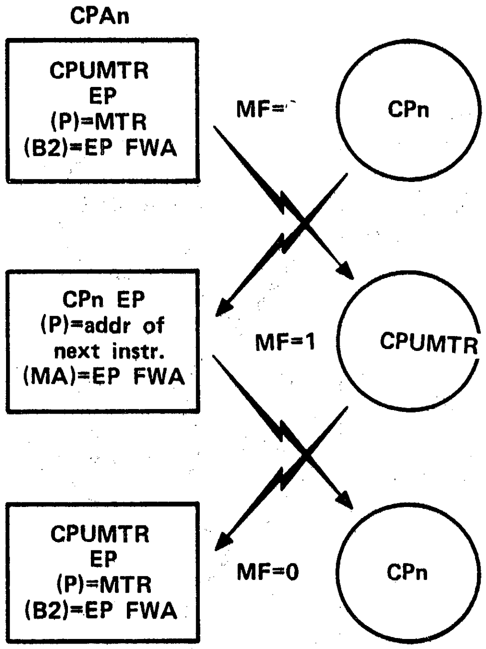
1. PP sets word zero of exchange package. (P)=PPR, (B0)≠0 (B2)=EP address for the PP issuing MXN.
2. CPUMTR starts executing at PPR. When complete, it issues XJ B2.
3. (P)=MTR since this location follows MTRX in CPUMTR. The next time this PP calls CPUMTR, it will reset (P)=PPR.

Figure 3-23. Pool PP Request



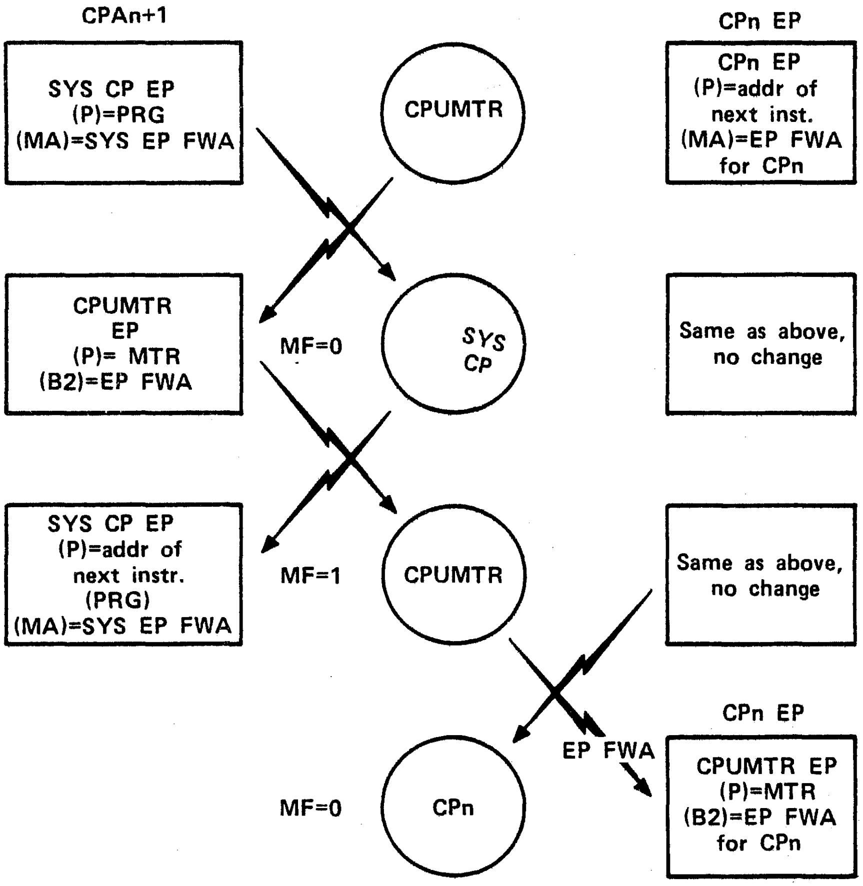
1. MTR sets up P, B0, B2. The request is stored in (X0) and the MTR issues MXN.
2. CPUMTR starts executing at PMN and exits at MTRX.
3. Same as pool PP.

Figure 3-24. PP MTR



1. Control point n places the request in RA+1, and will either XJ(MA) (where MA is the hardware register in the CPU), or wait for MTR to notice the request.
2. CPUMTR processes the RA+1 request and (unless recall is requested) reactivates control point n by XJ B2. (Note: (B2)=EP FWA)
3. CPUMTR exits at MTRX which sets (P)=MTR in control point n control point area.

Figure 3-25. Program Request

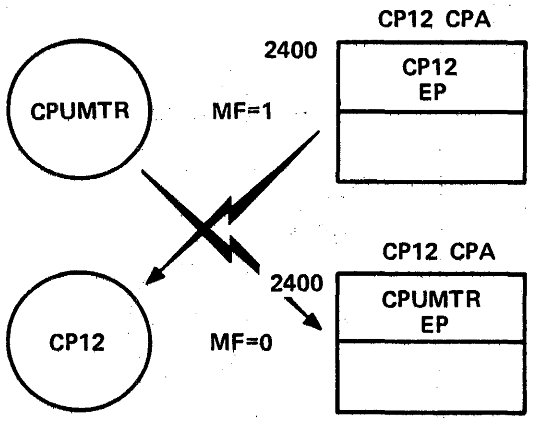


1. CPUMTR will add this request to the system control point queue. It then exits to MTRX (which is an XJ), thereby setting (P)=MTR in the exchange package.
2. When the system control point has exhausted its queue, it will XJ (MA) back to CPUMTR.
3. System control point has finished and exchanges to CPUMTR.
4. CPUMTR will now start the highest priority control point n.

Figure 3-26. System CP Program Mode

A probable sequence of system interaction is illustrated and explained in figures 3-27 through 3-37.

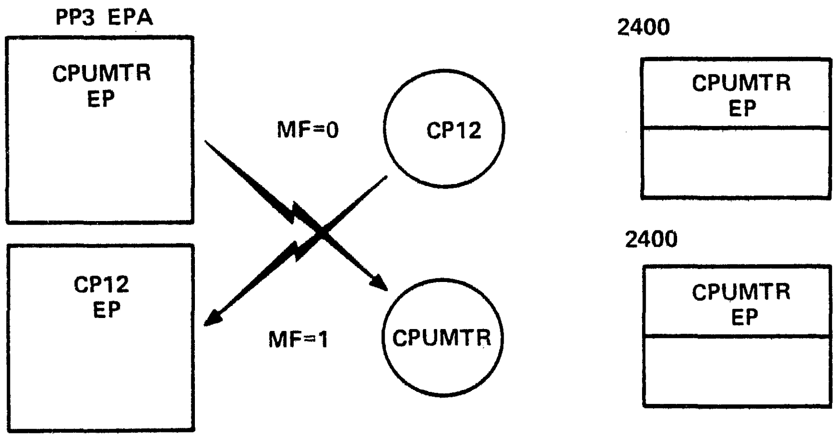
In figure 3-27, assume CPUMTR is running in MM, and it decides to activate control point 12; that is, give the CPU to control point 12.



PTX CPUMTR EP in CPU
(MA)=2400
CPUMTR issues
XJ B2 (B2=2400)
ATX CP12 EP in CPU
CPUMTR EP in CP12 EPA

Figure 3-27. CPUMTR Running in MM Activates CP12

Figure 3-28 assumes that PP3 asks CPUMTR to perform a function. PP3 must build a CPUMTR exchange package in exchange package area PP3. Note that RA is 0, FL equals machine field length, and P equals PPR, the FWA of CPUMTR PP function processor. PP3 issues an MXN. Since MF is 0, this exchange will occur.



PTX CP12 EP in CPU
PP3 issues MXN
ATX CPUMTR has CPU

Figure 3-28. PP3 Requesting Function from CPUMTR

In Figure 3-29, CPUMTR processes the PP request and then determines from CPU priorities that control point 14 should be activated.

Note

> Control point area 14 may exist from a previous XJ by MIR or may have been built due to a request by the scheduler or the advancement routines. Since control point 12 will not be activated, it is necessary for CPUMTR to move control point 12 from the PP3 exchange package area to the control point 12 exchange package area before issuing XJ=3000.

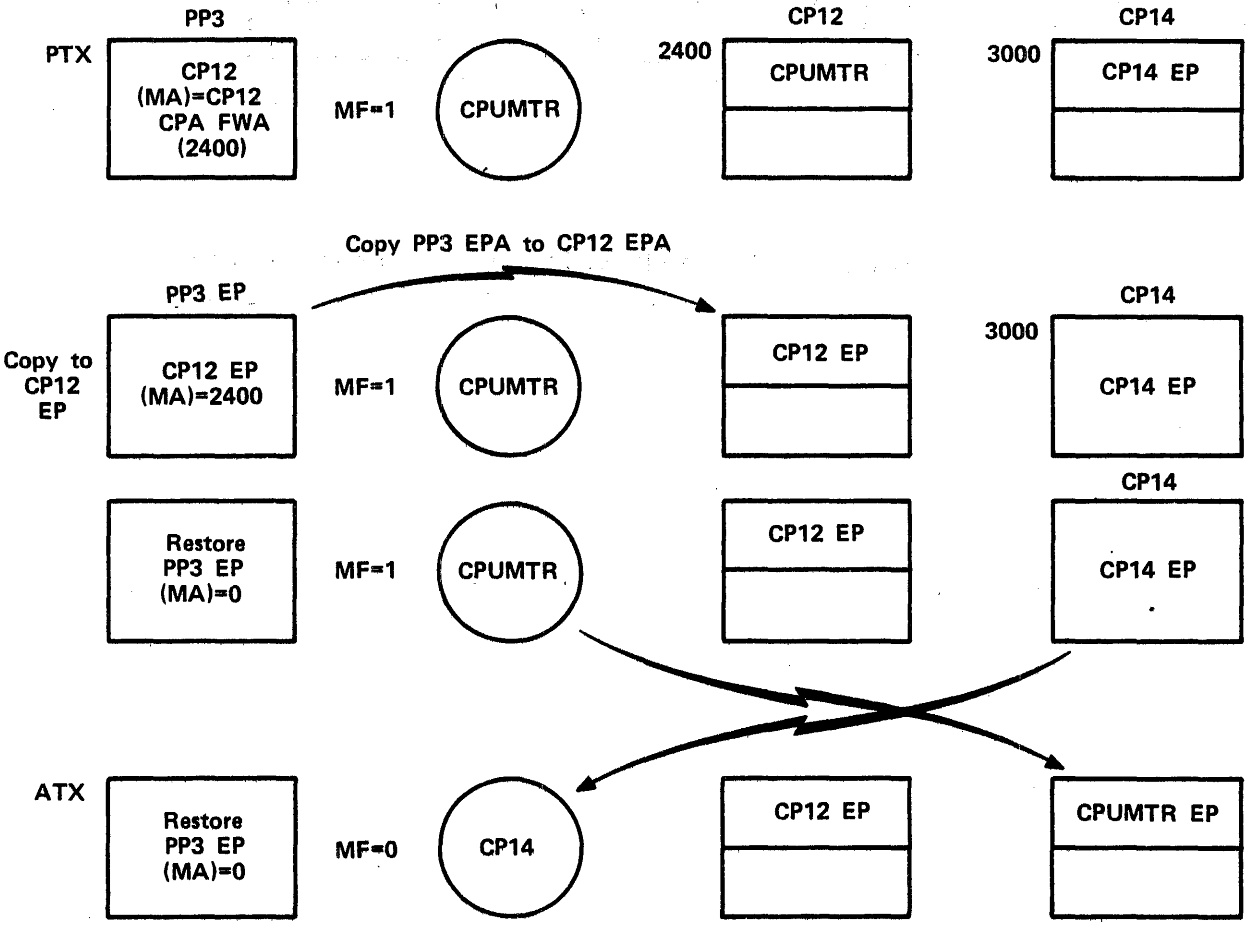


Figure 3-29. CPUMTR Processing PP Request Activates Control Point 14

In figure 3-30, MTR decides to switch control points (that is, stop control point 14 and start control point 10) and issues an ACSF (switch job request) to the CPUMTR. MTR must build a CPUMTR exchange package in its exchange package area and issue MXN.

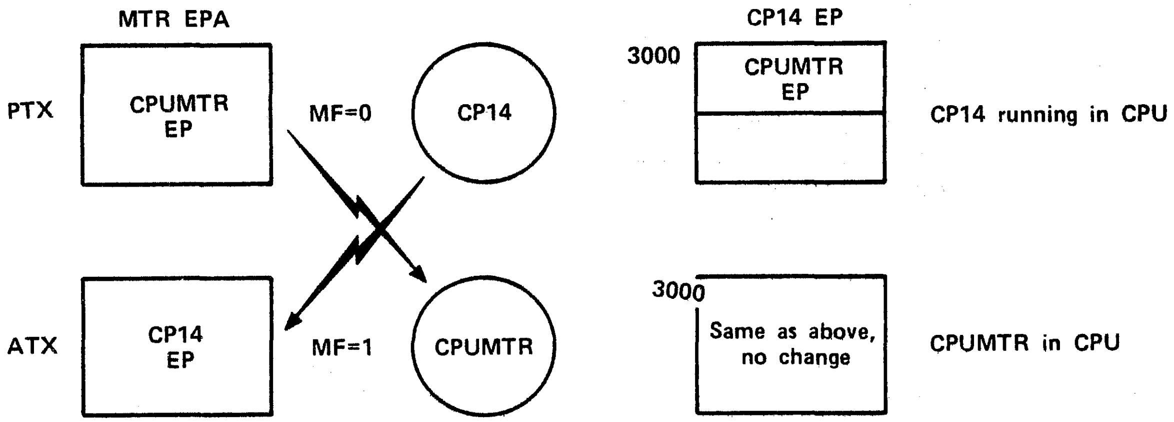


Figure 3-30. MTR Switches Control Points

In figure 3-31 CPUMTR activates control point 10. MTR decides which control point to start, and CPUMTR starts it.

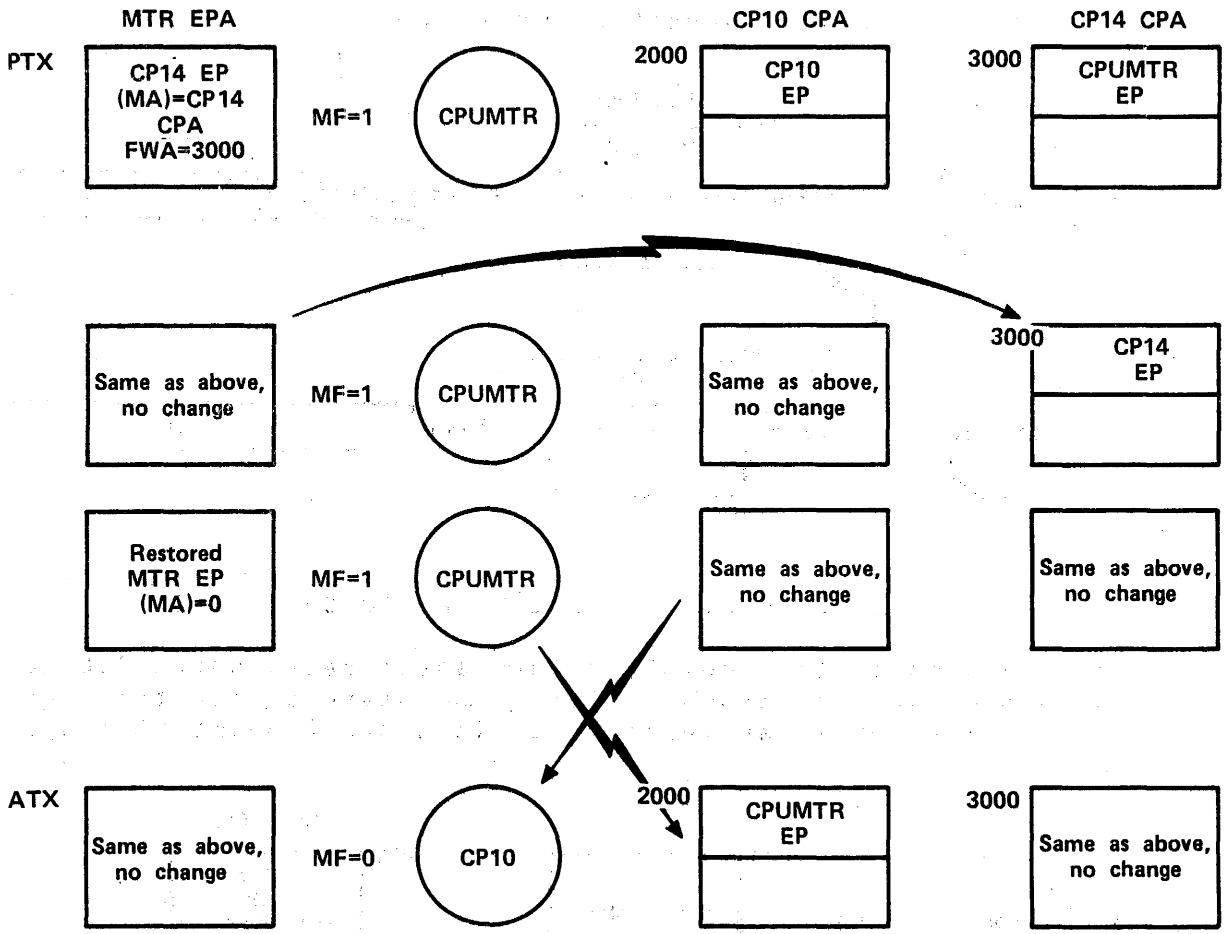


Figure 3-31. CPUMTR Activates Control Point 10

In figure 3-32 control point 10 needs to call CIO. It places the call in RA+1 and issues an XJ. Since the monitor flag is zero, the exchange will store the CPU exchange package value in location (MA). Now, whenever CPUMTR builds the control point 10 exchange package, it sets (MA)=2000 and (P)=MTR, the FWA of CPUMTR control point request processor.

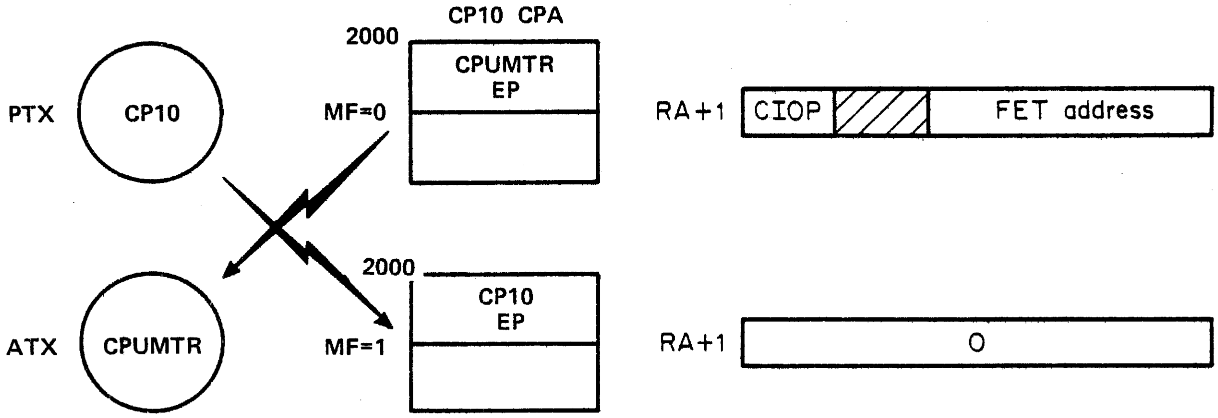


Figure 3-32. Control Point 10 Calls CIO

CPUMTR places control point 10 into autorecall, calls CIO to a pool processor (for example, PP6), and searches for the highest CPU priority job to activate which is control point 16 (figure 3-33).

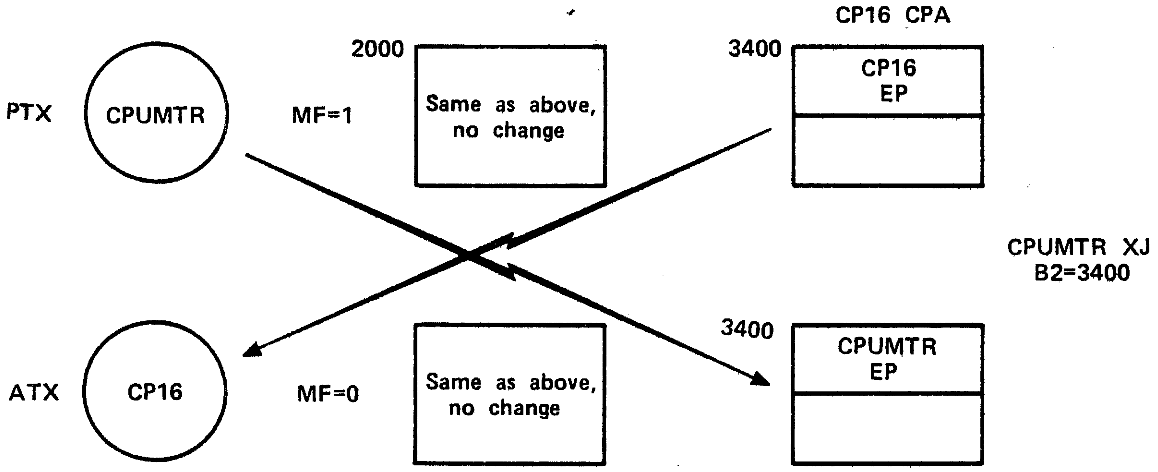


Figure 3-33. CPUMTR Calls CIO, Activates Control Point 16

CIO runs to completion, sets the status of its operation to complete, and prepares to drop. In order to drop, CIO will MXN to monitor with a DPPM (drop PP request). Refer to figure 3-34.

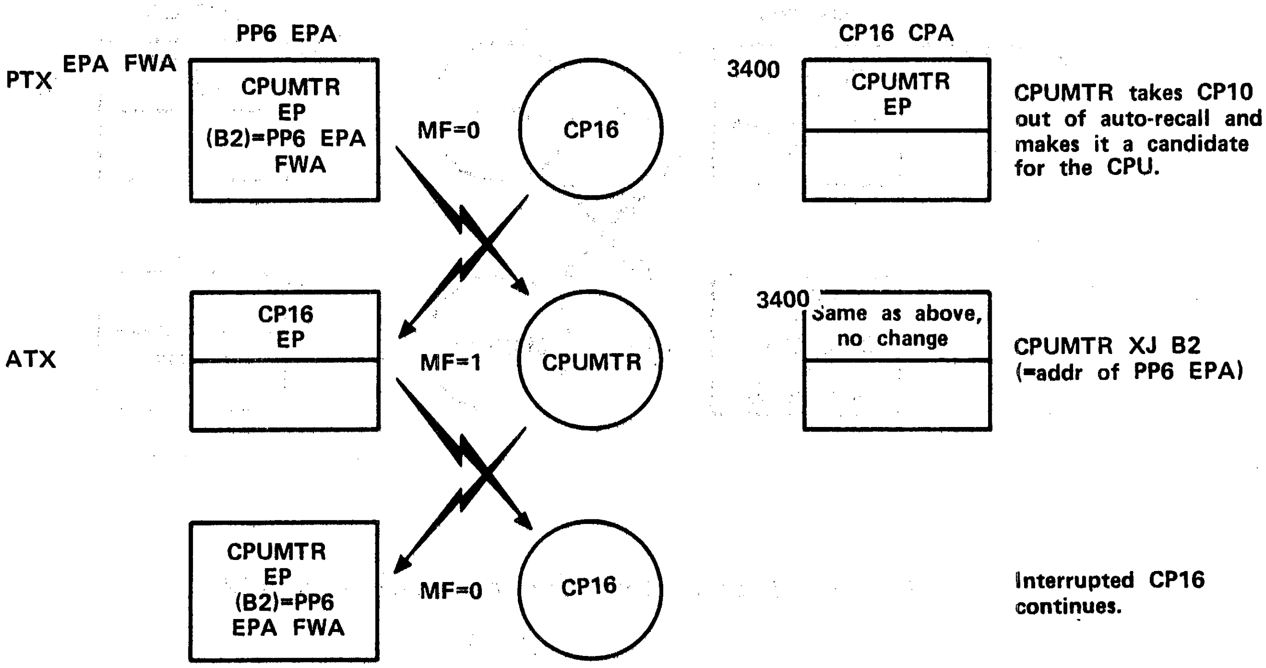


Figure 3-34. CIO Runs to Completion and MXNs to Monitor

In figure 3-35 PP4 issues a DTKM (drop track function) via an MXN.

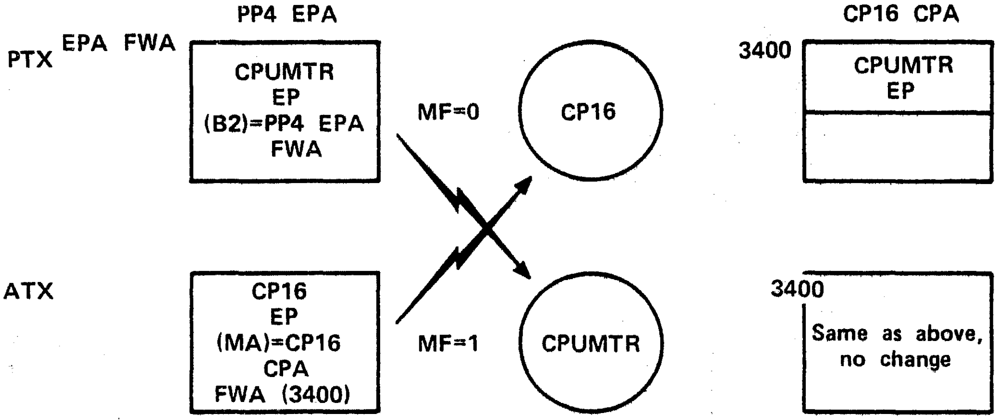


Figure 3-35. PP4 Issues DTKM Via MXN

Now PP4 idles on its OR until monitor satisfies its request. DTKM is a request which takes too long a CPU time-slice; therefore, it is processed by CPUMTR in program mode via the system control point. The system control point is treated as any other control point except that it has the highest priority. CPUMTR begins processing this request by queuing the request and executing XJ B2=4000, thereby activating the system control point. If the system control point is interrupted, CPU/MTR processes the interrupting request.

If it is a request which is also processed by the system control point, CPUMTR queue, this request and reactivates the system control point. In this way, all these types of requests are handled in a first come, first served order.

Before the exchange can occur, however, CPUMTR must copy the control point 16 exchange package from PP4 exchange package area as shown in figure 3-36.

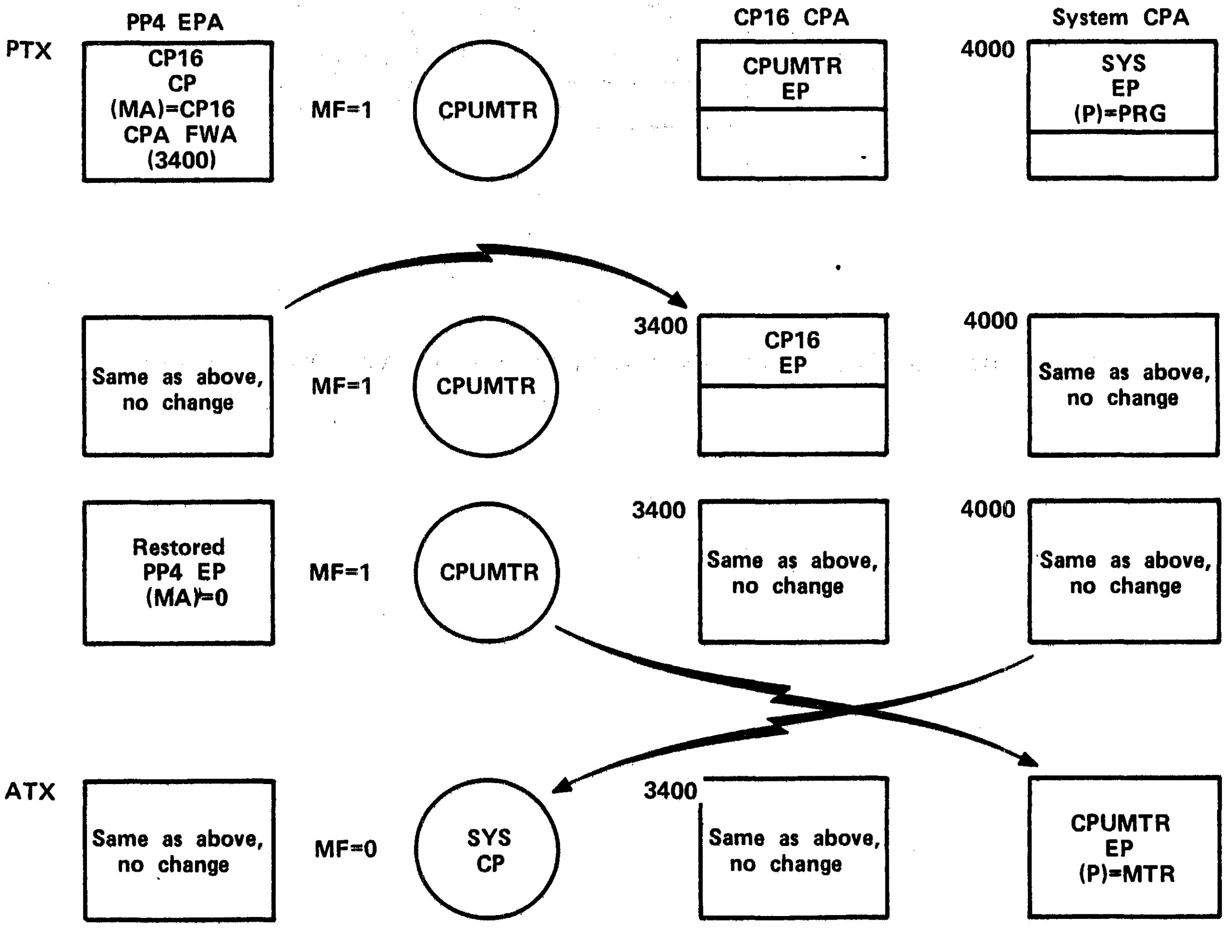


Figure 3-36. System Control Point Processing

When the system control point completes all the requests in its queue, it will XJ (MA) to the CPUMTR.

For system control point (MA)=4000, CPUMTR sets (P)=MTR in the CPUMTR exchange package at system control point area. When the system control point exchanges, CPUMTR begins at MTR. However, the system control point begins executing at PRG (figure 3-37).

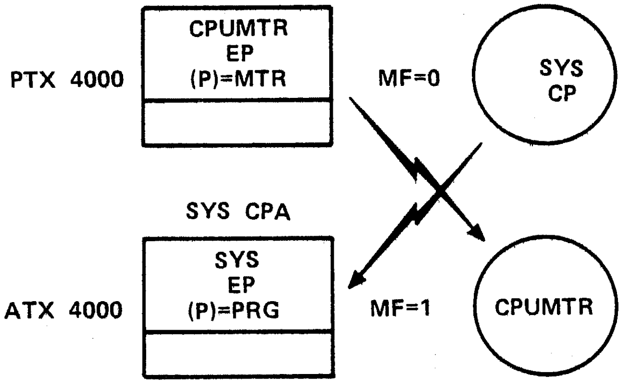


Figure 3-37. System Control Point XJ (MA) to CPUMTR

SUBCONTROL POINTS (SCP)

Subcontrol points are divisions of a central memory control point. A user can set up a control point to contain two or more programs; one of these is designated as the executive, and monitors the other program(s) or subcontrol points.

The executive controls its subcontrol points in much the same manner that the system monitor controls the control points. When a control point makes a system request or exceeds its time limit or makes an error, control is given back to the system monitor. Similarly, when a subcontrol point makes a system request or exceeds its time limit or makes a CPU error, control is given back to the executive. The executive sets up each subcontrol point so that, within the field length of the control point, each subcontrol point has its own RA and field length and cannot go outside its boundaries. The executive is thus protected from access by the subcontrol points, whereas the executive's RA and FL define the full control point so the executive can watch over and control all subcontrol points within the field length.

The subcontrol point concept depends on the executive program's handling of the subcontrol points. This involves starting, stopping, error processing, and other functions similar to those of the system monitor.

Just as the system monitor keeps track of each control point through its exchange package, the executive can control the subcontrol points through their exchange packages.

It is the responsibility of the executive to set up an exchange package for each subcontrol point; each exchange package must have the appropriate RA, FL, and so on, for the subcontrol point. These exchange packages must be set up somewhere within the executive's field length, but probably not within the field length of the subcontrol point. To start execution of a subcontrol point, the executive uses an XJP RA+1 request indicating the address of the exchange package area of the subcontrol point to be activated. When CPUMTR picks up the request, it terminates the executive and activates the subcontrol point described in the exchange package area indicated on the XJP request. CPUMTR also sets a flag in the control point area showing that at this control point a subcontrol point is now active. Once activated a subcontrol point runs until:

1. It makes a CPU error

2. It exceeds its time limit

3. It makes an RA+1 request

Under any of these three conditions, control is given back to the executive.

The executive can thus monitor error processing for the subcontrol points. Errors can be noted and examined without termination of the control point. Upon returning control to the executive, certain information is set up in the X registers:

(X2) = msec before this subcontrol point began
(X6) = error flag (12 bits) and RA of this subcontrol point
(X7) = msec used by this subcontrol point

One of the parameters on the XJP request is the time for the subcontrol point. When this time limit is passed, control goes back to the executive.

When a subcontrol point makes an RA+1 request, control is returned to the executive; the executive can then decide whether to:

1. Ignore the request

2. Handle the request itself

3. Pass the request on to CPUMTR using RA+1 of the control point (executive)

Subcontrol points can be set up by any CPU programmer using any programming language; some features are only usable by COMPASS programs. The structure of the executive is flexible within the limits we have discussed so far. As an example, consider the transaction subsystem using subcontrol points.

TRANSACTION EXECUTIVE

The transaction executive is designed to let many different users use one system; each user needs transaction processing. Users can set up their own programs for transaction processing and all transactions can be handled through the transaction executive.

The transaction executive uses subcontrol points so that it can maintain complete control over each task to be performed. Within its field length is needed a protected area for the executive; the remaining field length can be used by up to 31 subcontrol points. The tasks to be performed require different programs that do not need to be in memory simultaneously; rather than using traditional overlays which have no protected area for the executive, each task or transaction program can be set up as a subcontrol point which can be activated as necessary by the executive.

Transaction programs can be written in any programming language. In order to make the programs more useful, the first 100 words of each program should be allocated for communication between subcontrol points; this can be done by using labeled common which is always at the beginning of the field length, for example:

```
     (FTN)  COMMON  /CCOMMON/  A(100)

     (COMPASS)  USE   /CCOMMON/

                BSS   100

     (COBOL)  COMMON  STORAGE  SECTION.

              77 A  OCCURS  100  TIMES.
```

NOTE

> RA+0 through RA+100 is normally not easily available to higher level languages, therefore the technique of labeled common allows, an easy method of access to RA+101 through RA+201.

The user programs should be compiled and then loaded to create a (0,0) overlay from each transaction program.

Each transaction to be processed must give enough information to indicate the proper transaction program to be brought in for processing. This information could include:

1. User's name (code)

2. Type of transaction

3. Data to be used in the transaction

The executive then brings in the appropriate transaction program into its field length and sets up the program as a subcontrol point. Since the user program is an absolute (0,0) overlay the loader cannot be used to load it*, so the executive has to use a CIO function to bring in the program. The executive also has to set up an exchange package for the subcontrol point and put any necessary information into the 100 word communication area in the subcontrol point's field length. If the transaction requires another program to complete the task, a request must be made to the executive to bring in the other program. The executive always checks to see if the program is available in memory already and brings in a copy if necessary; then the executive copies the appropriate data from the communications block of the calling subcontrol point to the communications block of the called subcontrol point.

---

* LDR always gives control directly to the (0,0) overlay after loading; this does not allow the executive to start the subcontrol point.

The transaction executive's job sets up the field length in the most efficient way. The field length must contain:

- The executive's code
- Tables
- Subcontrol points
- Exchange package areas for each subcontrol point

The field length could be set up as shown in figure 3-38.

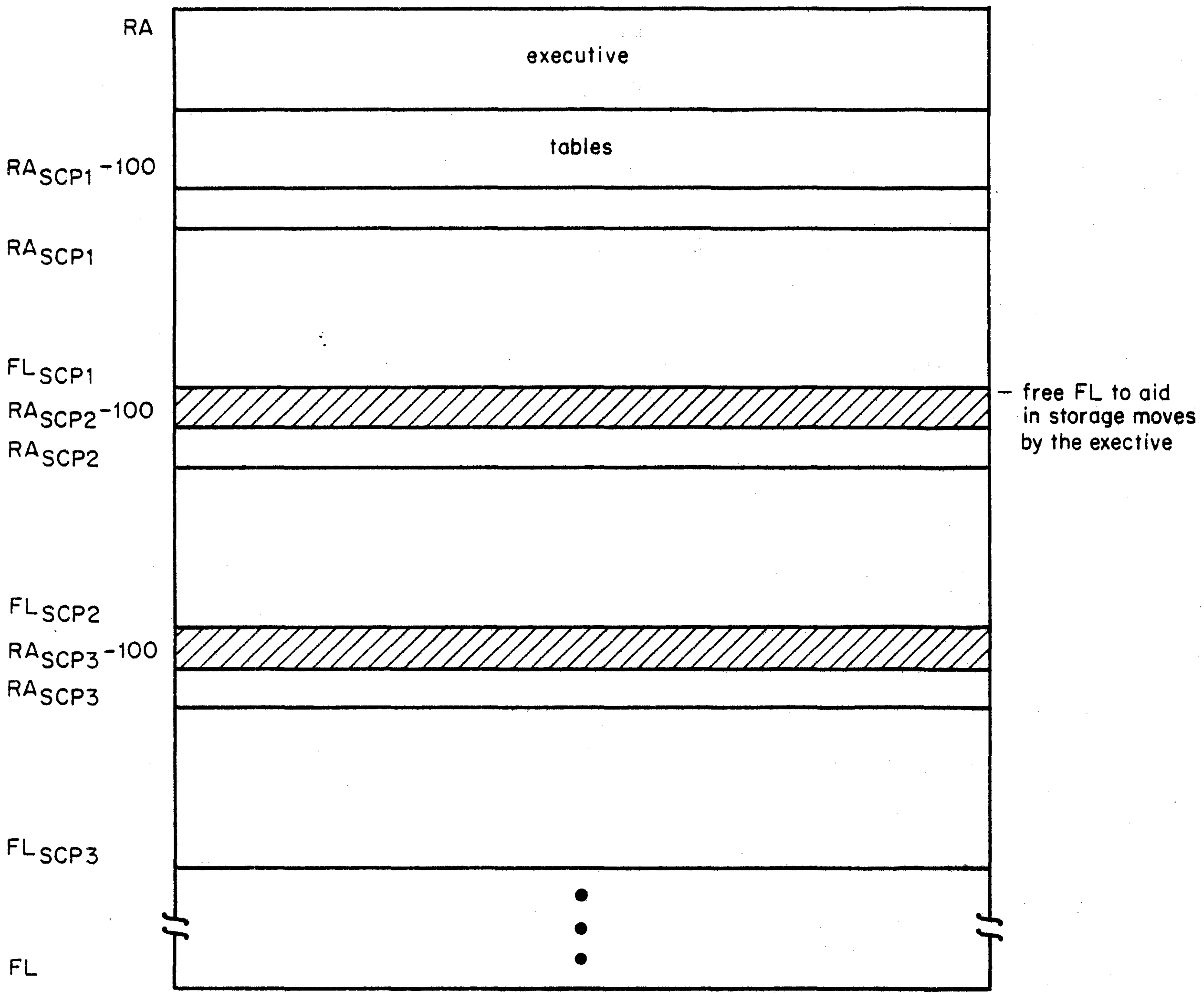


Figure 3-38. Subcontrol Point Field Length

The area RA scp -100 through RA scp can be used for the exchange package area for the subcontrol point. The executive can fill in this area as it reads in the program; it gets P from the 50 table of the (0,0) overlay binary, it can set up values for the registers for COMPASS programs, it sets up RA and FL depending on where the program was read into memory and how many words were read in.

The executive always checks through its tables to see if the program is already at a subcontrol point; if it is already at a subcontrol point, the executive checks to see if it is a reusable program if the program is not in memory or not reusable, the executive will read in another copy of it. The executive looks for the next available place inmemory to put the program and brings it in using READR (READSKP) and updates its tables. The executive must set up the exchange package area. When CPUMTR picks up the request it exchanges in the subcontrol point and sets the flag in the control point area to indicate that there is a subcontrol point at the control point.

Transaction Subcontrol Points

Transaction subcontrol points are all (0,0) absolute overlays. These programs are loaded by the executive using a CIO function. The executive also sets up an exchange package for each subcontrol point so that each subcontrol point can use only memory within its own RA through RA+FL-1.

The transaction executive has set up one subcontrol point (ITASK) which decides which other program needs to be brought in to handle a transaction. ITASK can look at the transaction code from the user and find the name of the program to do the task. Since ITASK is a subcontrol point itself and cannot go outside its own field length, ITASK must ask the executive to activate the appropriate transaction program at a subcontrol point.

When a subcontrol point needs assistance from the executive, it puts a request in its own RA+1; this causes an exchange back to the executive. The executive looks at the request and can:

1. Ignore the request
2. Process the request itself
3. Pass the request on to CPUMTR

After the request has been handled, the executive can give control back to the subcontrol point if its is appropriate.

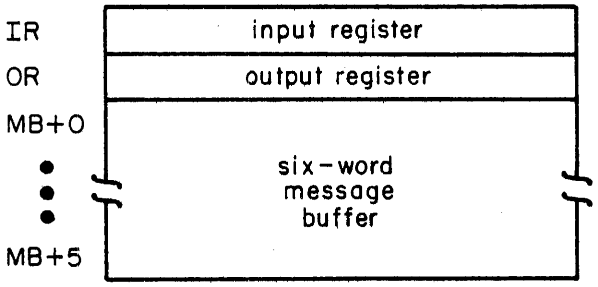
An example of a request would be a subcontrol point requiring the loading of another subcontrol point to complete a task. When the first subcontrol point puts the request in its RA+1, the executive is exchanged in; the executive brings in a copy of the program if necessary and copies the communications block from the calling program to the called program. The RA+1 of the subcontrol point is within the FL of the executive who can read the request.

PPR/SYSTEM INTERACTION

Each PP functions independently of the CPU and other PPs. To enable the PP to communicate with and work for the system, PPR provides the necessary links between the PPs and the CPs. PPR serves as a PP idle program, the loader of PP programs, and a source of commonly used subroutines for other programs and routines. PPR is loaded into pool PPs at deadstart time by STL and is never changed. A dedicated PP program such as 1TD (the multiplexer driver) overlays PPR and restores it via 1RP prior to dropping back to pool PP status. MTR (PP monitor) and DSD (display driver) are two other dedicated PP programs which do not contain a copy of PPR.

Initially, PPs can be loaded only at deadstart time by transferring data across their respective channels (refer to section 26). This method of loading PP routines during normal system operation is unacceptable because other peripheral equipment may be on the channels. The alternative is to have each PP execute an idle loop which checks the status of a word in CM. This is accomplished through the PP communication area in CMR. There is one entry for each available PP, and each entry is 10B words in length (refer to section 2).

The first word of each entry is the input register (IR), the second word is the output register (OR), and the remaining six words are used as a message buffer. A sample entry is as follows.



The CM addresses for each PP input register, output register, and message buffer are stored in direct cells named IA, OA, and MA in each PP. These are 12-bit absolute CM addresses and, therefore, the PP communication area must reside below address 10000B.

Figure 4-1 illustrates the interaction between CP monitor (CPUMTR) and a pool PP to activate a PP program. CPUMTR checks for an available PP and places the PP routine name (three characters) and arguments (36 bits) in the pool PP input register. The pool PP is cycling through an idle loop waiting for its IR to become nonzero. When the IR is nonzero, PPR calls subroutine PLL (peripheral library loader) to load the requested routine. If the requested routine is not found, the SCOPE function processor (SFP) is loaded. If the requested routine is found, execution of that routine begins after calling the pause routine (PRL). As the routine executes, it can communicate with the system by monitor requests utilizing the FTN (process monitor function) subroutine in PPR. FTN places the monitor request in the PP OR. Monitor responds to the request and completes it by setting byte 0 of OR equal to 0. When the PP routine terminates, it informs monitor of this condition via a monitor function DPPM and jumping to the idle loop in PPR.

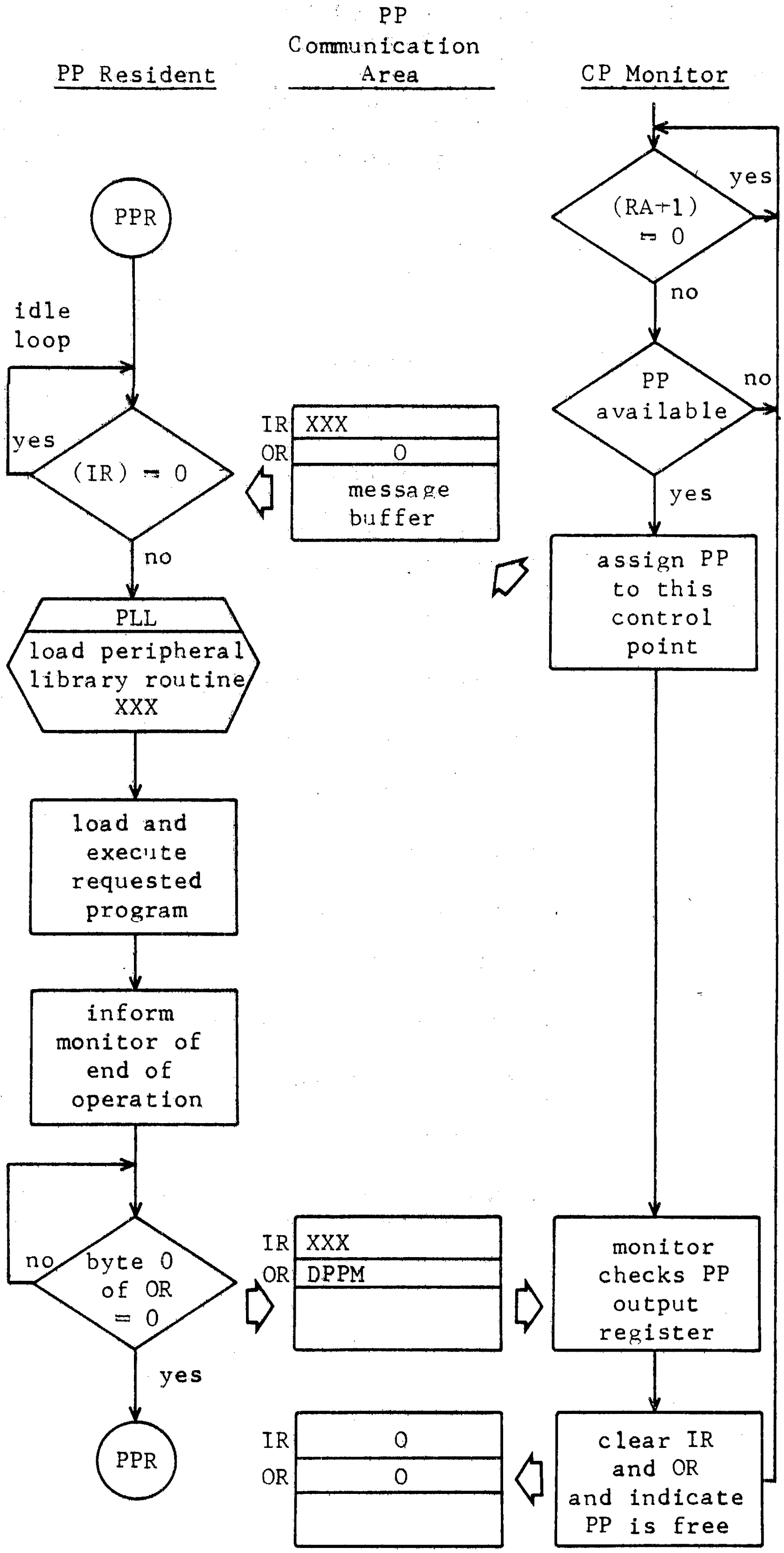


Figure 4-1. System Interaction - PPR

Table 4-1 represents a pool PP memory map. The address to the left is the first word address of the functional area. Direct cells are memory locations 0 through 77B. The mass storage buffer is normally located at address BFMS for those routines requiring mass storage I/O. The first executable instruction begins at PPFW (1100B) with a one CM-word library table entry preceding it at 1073B. Mass storage drivers are loaded at MSFW.

TABLE 4-1. POOL PP MEMORY MAP

FWA (Octal)	Routine/Function	Name
0000 - 0077	Direct cells	
0100 - 1100	PP resident routines and mass storage driver area	
	Idle loop	PPR
	Peripheral library loader	PLL
	Load MS error processor	LEP
	Process monitor function	FTN
	Pause for relocation	PRL
	Reserve channel	RCH
	Release channel	DCH
	Send dayfile message	DFM
	Execute routine	EXR
	Set mass storage	SMS
	Mass storage driver designator	MSD

TABLE 4-1. POOL PP MEMORY MAP (CONTINUED)

Routine/Function	Name
FWA of mass storage drivers	MSFW
Read sector	RAS
Write sector	WDS
End mass storage operation	EMS
Library entry of current PP routine	-
First word of PP routine	PPFW
Mass storage buffer (502B words)	BFMS
MS error processor	EPFW
Last word of PP	7777

PPR SUBROUTINE DESCRIPTIONS

Whenever a pool PP is waiting to be assigned, it executes the idle loop, PPR. This routine reads the input register in CM every 128 microseconds (for both 1X and 2X modes). That is, if byte 0 of IR is zero, the PP delays 128 microseconds before reading IR again. If IR is nonzero, then the name of the requested PP program is in IR, and that routine is loaded and executed.

In order to load a PP program or overlay, subroutine PLL is used. This routine requests monitor to search the PLD for the requested routine (monitor function SPLM). If the overlay is found, it is loaded; if it is not found, overlay SFP is loaded. If SFP does not recognize the PP overlay named in IR, the error message xxx NOT IN PP LIBRARY is issued and the control point is aborted. The PP then reenters the idle loop.

Subroutine LEP is used to load the mass storage error processing overlays from CM.

Subroutine FTN is called to issue monitor requests. The function is stored in the output register. If this is a CPUMTR request, FTN executes a monitor exchange instruction (MXN). If not a CPUMTR function, FTN waits for the completion of the function. Completion is indicated by byte 0 of OR being set to zero by monitor. FTN then returns control to the calling routine.

If a PP is assigned and executing at a control point, that control point cannot be moved by monitor. To enable a storage move, the PP must pause by using subroutine PRL. If a move takes place, CM addresses being used by the PP routine will have to be adjusted because RA has changed. Do not use PRL with nondedicated channels reserved.

Subroutines RCH and DCH issue monitor functions RCHM and DCHM to reserve and release a channel or pseudochannel.

When a PP issues a dayfile message, subroutine DFM is used. The appropriate dayfile is selected and the message is passed in 40-character blocks through the PP message buffers. Again, do not use DFM with nondedicated channels reserved.

For a PP program to load an overlay, subroutine EXR is used. Do not use EXR with nondedicated channels reserved.

Subroutine SMS is called to load the proper mass storage driver into PPR. SMS must be called prior to a request for positioning or I/O (POS, RDS, and WDS).

NOS PP NAMING CONVENTIONS

The following PP naming convention is used by NOS.

xxx	Three alphabetical characters, used for RA+1 callable overlays (for example, CIO).
0xx	Zero level overlay, also known as location-free routine (for example, 0AV).
1xx	Reserved for system programs.
2xx	Reserved for system programs.
3xx	Reserved for system programs.
4xx	Reserved for system programs.
5xx	Reserved for diagnostic programs.
6yy	Mass storage driver (for example, 6DI); callable by SMS in PPR.
7yy	MS error processor (for example, 7DI); called by LEP in PPR.
8xx	Unused
9xx	Syntax and display type overlays used by DSD, DIS, 1TD, and 1LS.

In the preceding list, x refers to any alphabetic character and yy is a mass storage driver mnemonic (DE, DI, or DP).

NOTE

User programs can call a PP routine only if its name begins with an alphabetic character. Routine names beginning with a numeral character are callable by the system, other PP routines, subsystems, or special system jobs.

ERROR MESSAGES

All error messages from PPR are issued by the routine SFP. The dayfile messages are as follows.

Message	Description
xxx NOT IN PP LIB.	PP package xxx was not found in the PP library directory.
xxx NOT IN PP LIB. - CALLED BY yyy.	PP overlay/program xxx was not found in the PP library directory and was called by package yyy.
SFP/xxx PARAMETER ERROR.	Parameter address outside FL.
SFP/xxx ILLEGAL ORIGIN CODE.	Function illegal for user's job origin.
SFP CALL ERROR.	SFP not loaded by default.

DIRECT CELLS

Table 4-2 shows the direct location assignments available for PP routines. Cells ON, HN, TH, TR, IA, OA, and MA must not be changed by the PP routine. All others may be used. However, remember that T0 is used to hold the P register for the CRM, CWM, IAM, and OAM instructions and, therefore, is subject to change.

ROUTINE RESIDENCE

All PP routines reside in either the resident peripheral library (RPL) in central memory or on mass storage, and are pointed to by the peripheral library directory (PLD). System performance can be affected by the residence of frequently used routines. Further, the following routines must reside in CM in the RPL: 1MB, 1MC, 1DD, SFP, 0DF, 7SE, 7EP, and all the mass storage drivers and error processors. Other routines recommended to be in the RPL are contained in the default LIBDECK released with NOS.

1DD AND 1RP

Two routines associated with PPR are 1DD and 1RP. Routine 1DD is called by DFM when a dayfile buffer is full and requires flushing to the disk. Routine 1RP is called by a PP routine to restore that PP's copy of PPR. For instance, 1TD (the multiplexer driver) calls 1RP to restore PPR when TELEX is dropped. This is done by passing a copy of PPR from another PP through the message buffer.

TABLE 4-2. DIRECT LOCATION ASSIGNMENTS

Symbol Name	Location (Octal)	Description
T0	0	Temporary storage
T1	1	
T2	2	
T3	3	
T4	4	
T5	5	
T6	6	
T7	7	
CM	10	CM word buffer (five locations)
LA	15	Package load address
	Set by PP resident before entry to program	
IR	50	Input register (five locations)
RA	55	Reference address/100
FL	56	Field length/100
	Read-only constants	
ON	70	Constant 1
HN	71	Constant 100B
TH	72	Constant 1000B
TR	73	Constant 3
	Set by PP resident before entry to program	
CP	74	Control point address
	Read-only constants	
IA	75	Input register address
OA	76	Output register address
MA	77	Message buffer address

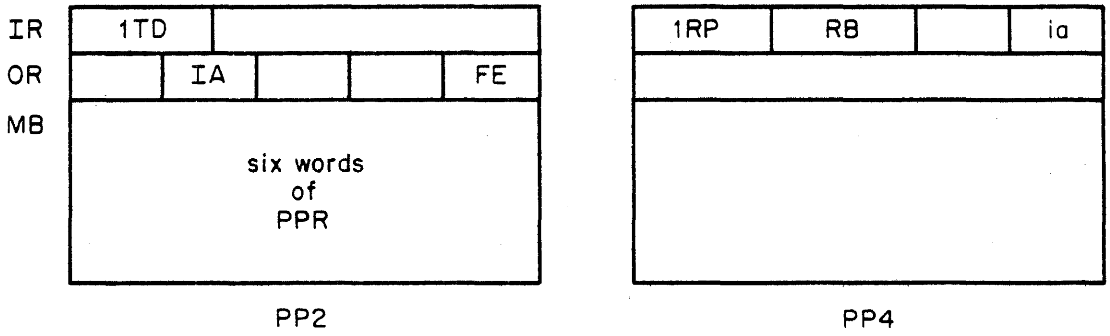
In figure 4-2, 1TD is running in PP2 and needs to restore PPR prior to dropping. Routine 1TD requests 1RP via RPPM. CPUMTR assigns an available pool PP (say PP4) to execute 1RP. Next CPUMTR informs PP2 which PP was assigned. Now that both PPs acknowledge each other, PP4 can pass its copy of PPR to PP2. This is done in six-word blocks using PP2's message buffer. Completion is marked by a short (less than six words) transfer.

7SE

Routine 7SE is called by routine PLL in PPR when an error occurs loading a routine from mass storage. If the routine was on an alternate system device, the library entry is changed to point to the routine on the system device. Routine 7SE then returns to PLL to retry the load from a system device.

7EP

Routine 7EP is called after a DEPM monitor function to further process the disk error. Routine 7EP issues the dayfile messages, processes unrecovered errors and return and retry operations. Routine 7EP is also called after recovered disk errors to issue a message to the error log indicating the recovery status.

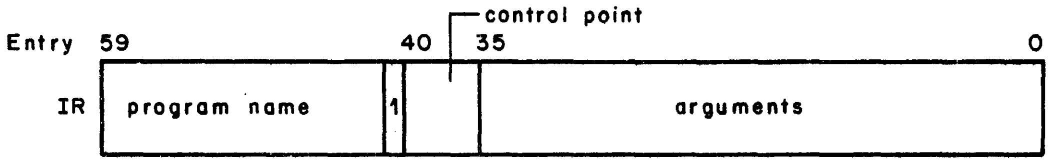


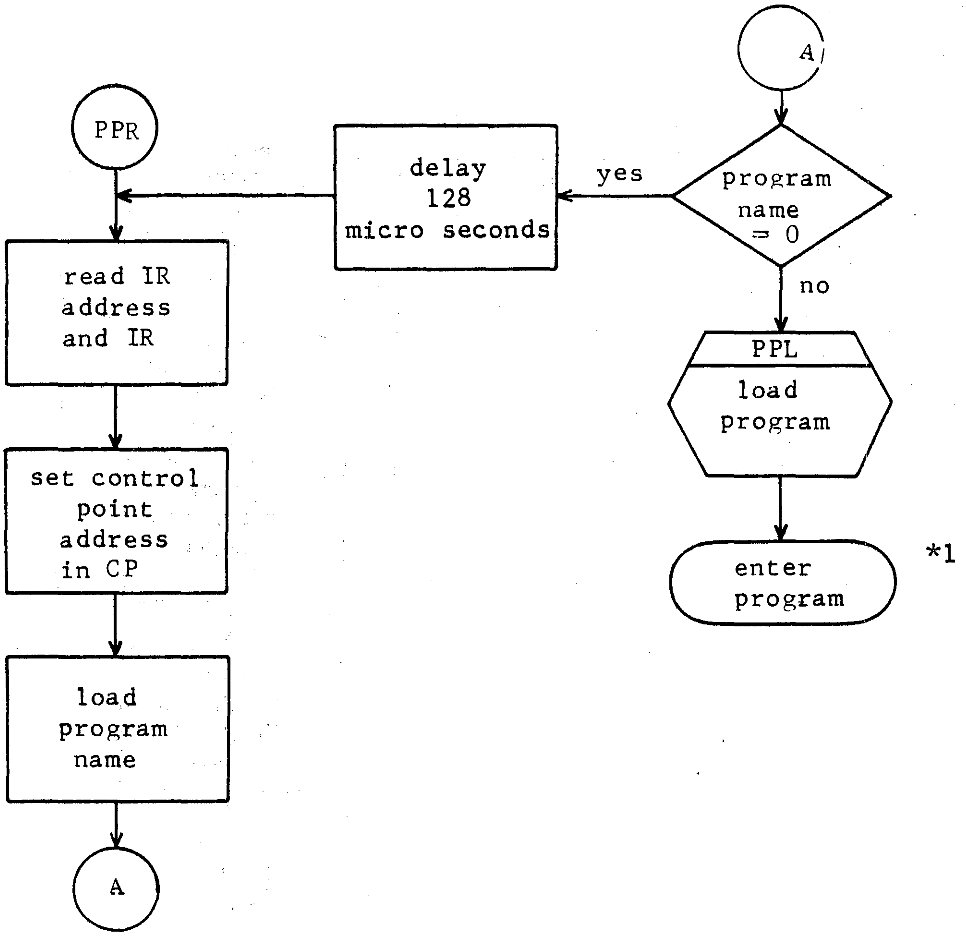
IA	Input register address for 1RP (byte 1).
ia	Input register address for 1TD (byte 4).
FE	Full/empty flag (1TD sets FE=0 to indicate empty buffer and 1RP sets FE=1 to indicate full buffer).
RB	Ready byte (byte 2). When 1RP is ready to transmit, byte 2 of 1RP's IR is set to 7777B. 1RP then waits for RB=0 before the next transmit. If this does not take place within 1 second, 1RP exits, thus aborting the load.

Figure 4-2. 1RP - Restore PPR

PP RESIDENT FLOWCHARTS

Figures 4-3 through 4-9 illustrate the PP resident routines.





*1 LJM 5,LA
LA contains the program load address. The first 5 words of the program are loader information.

Figure 4-3. PP Resident (PPR)

Entry (A) = program name
(LA) = load address for zero level overlay

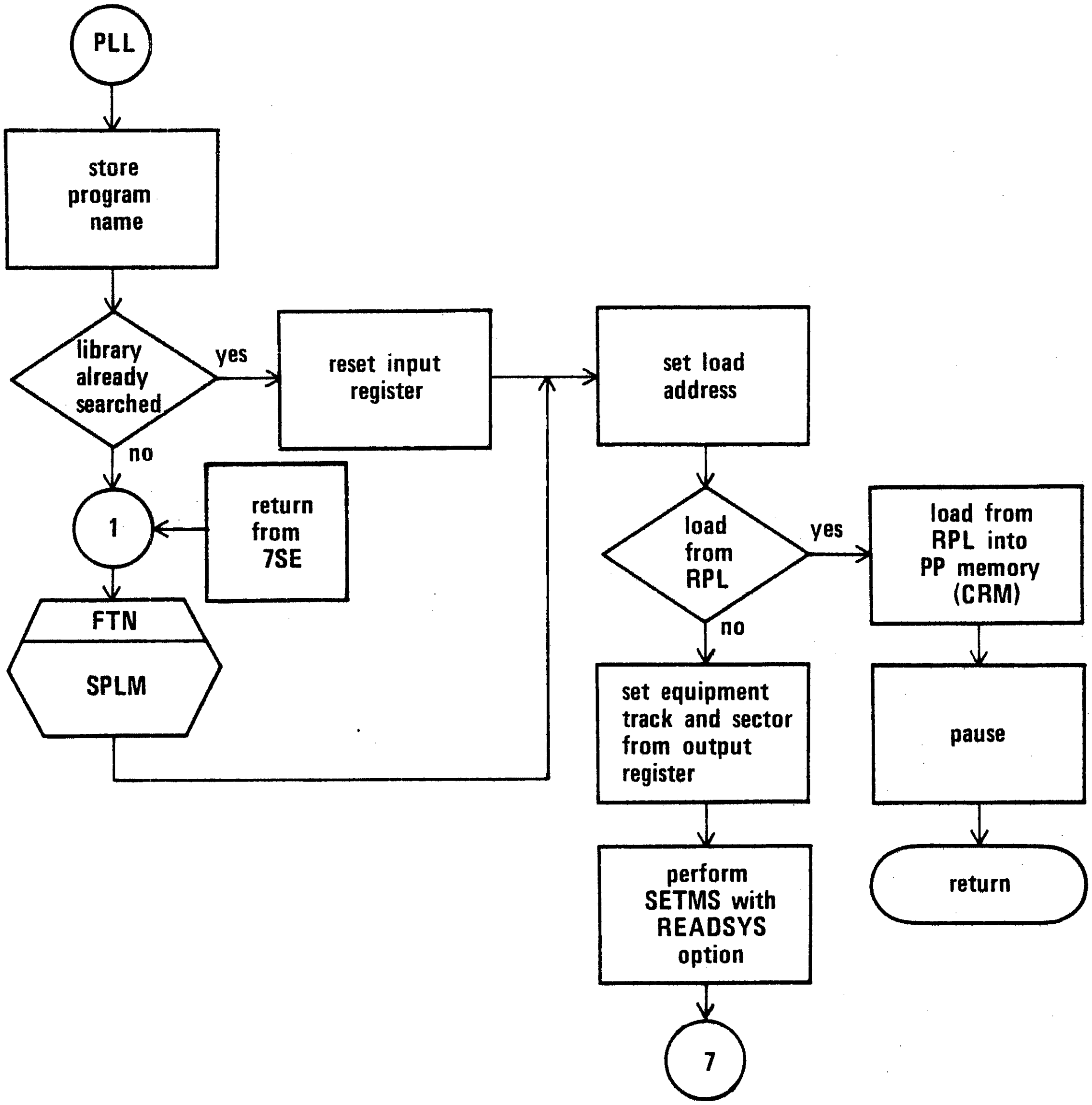


Figure 4-4. Peripheral Library Loader (PU)

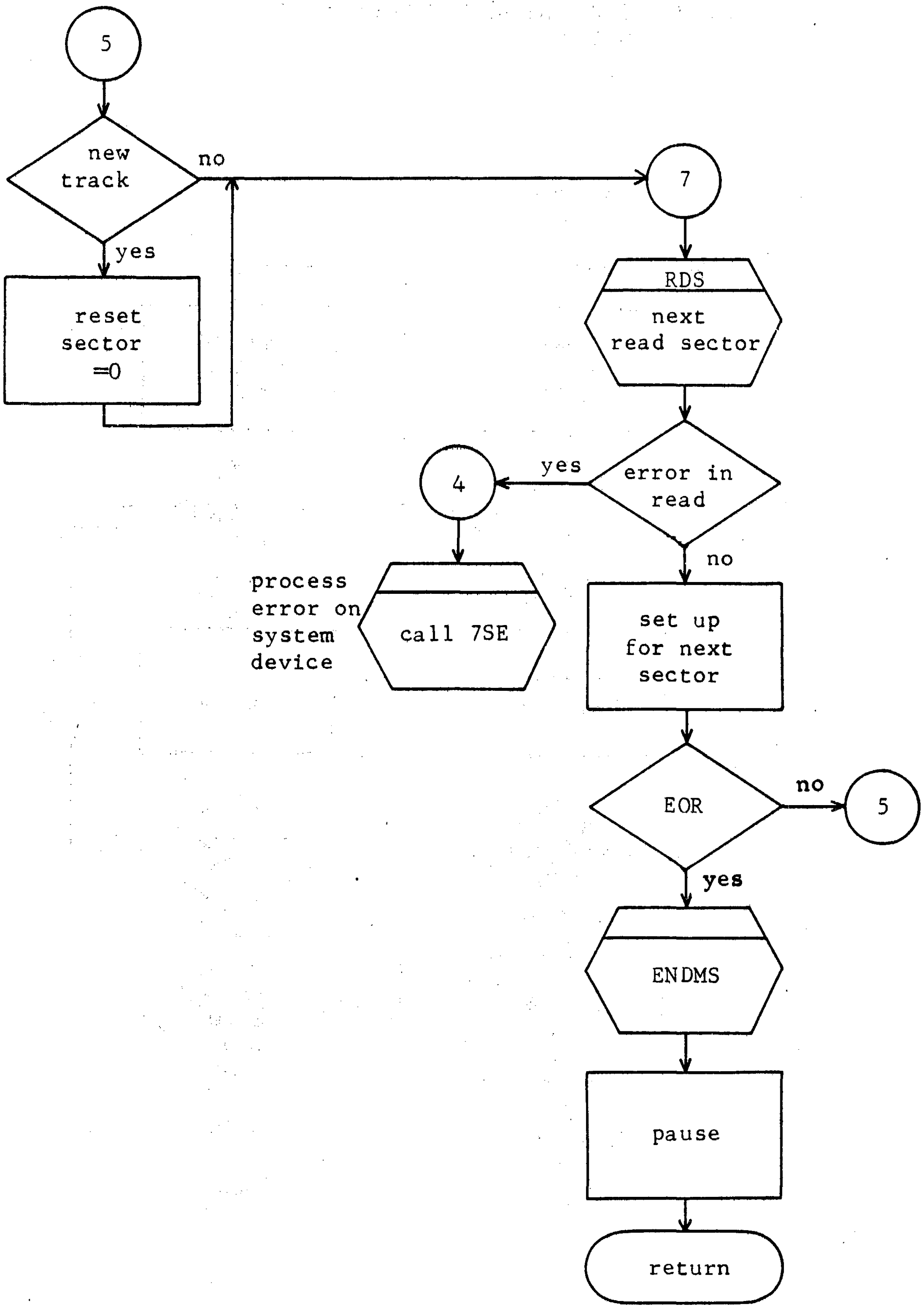


Figure 4-4. Peripheral Library Loader (PU) (Continued)

```
Entry (A) = MTR function
      (CM+1 through CM+4) = parameters
      (CP)= Control point number

Exit  (CM through CM+4) = OR
      (A) = 0
```

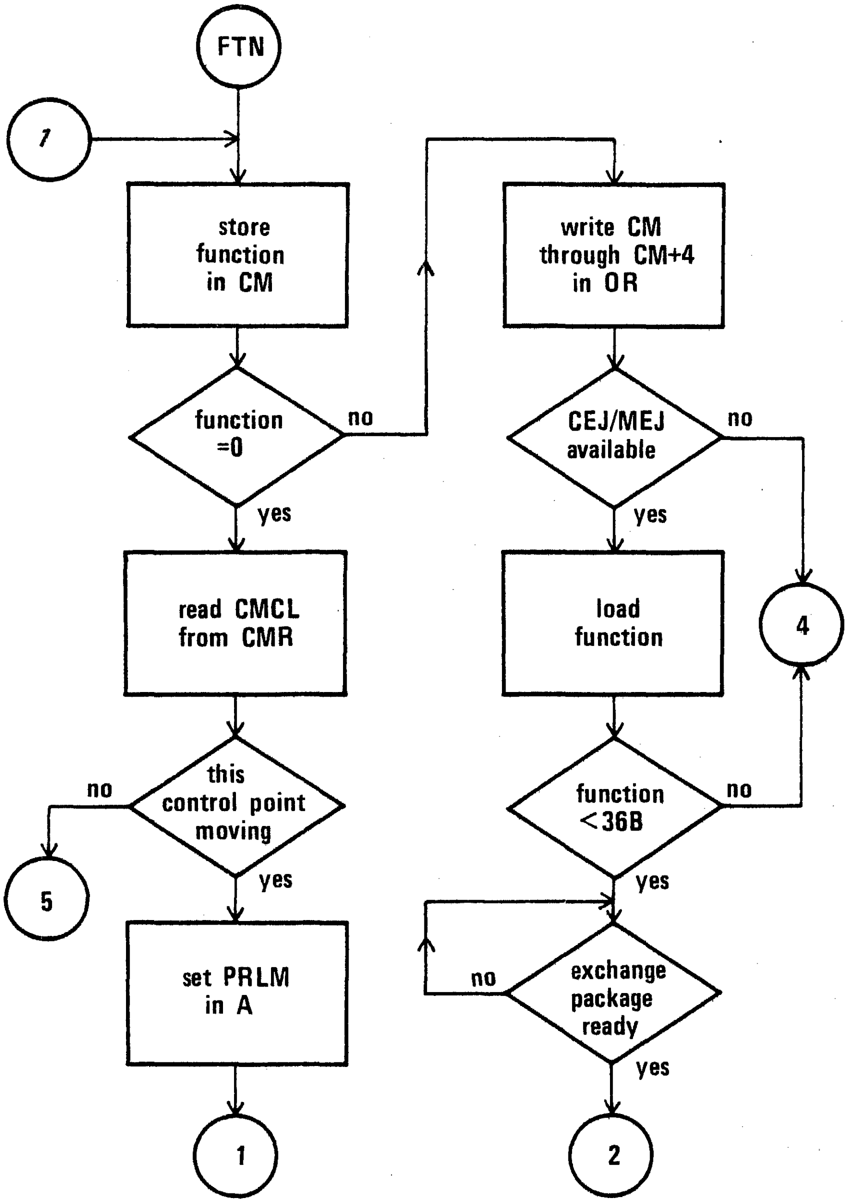
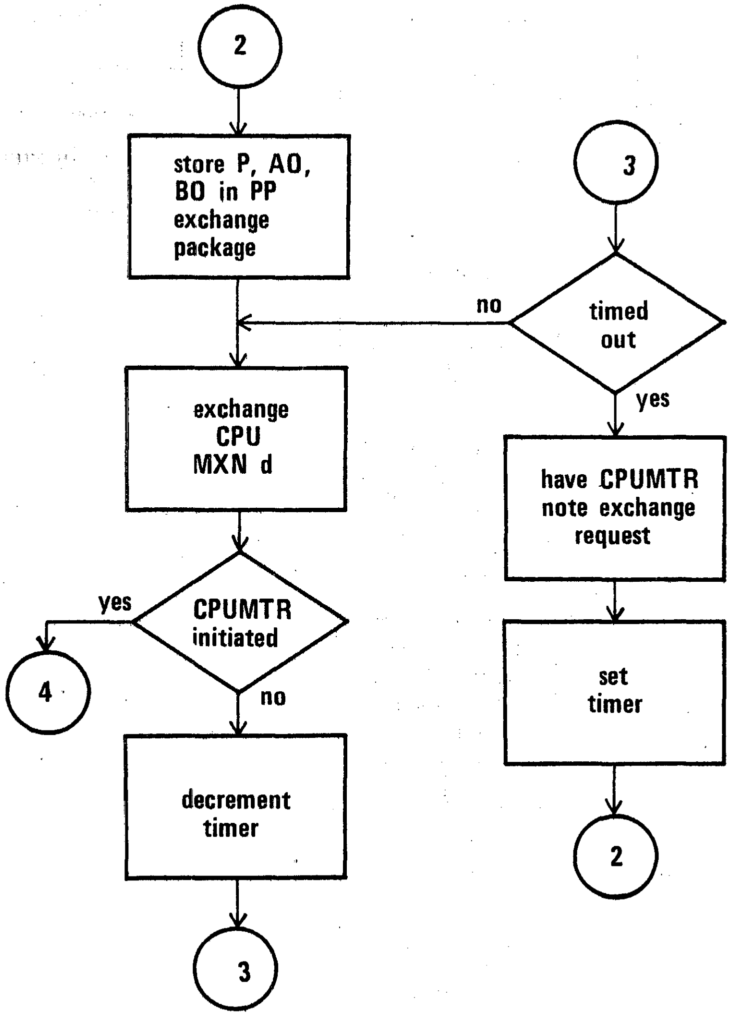


Figure 4-5. Process Monitor Function (FTN)



Note: (P), (A0) and (B0) are from PXPP+1 in CPUMTR

Figure 4-5. Process Monitor Function (FTN) (Continued)

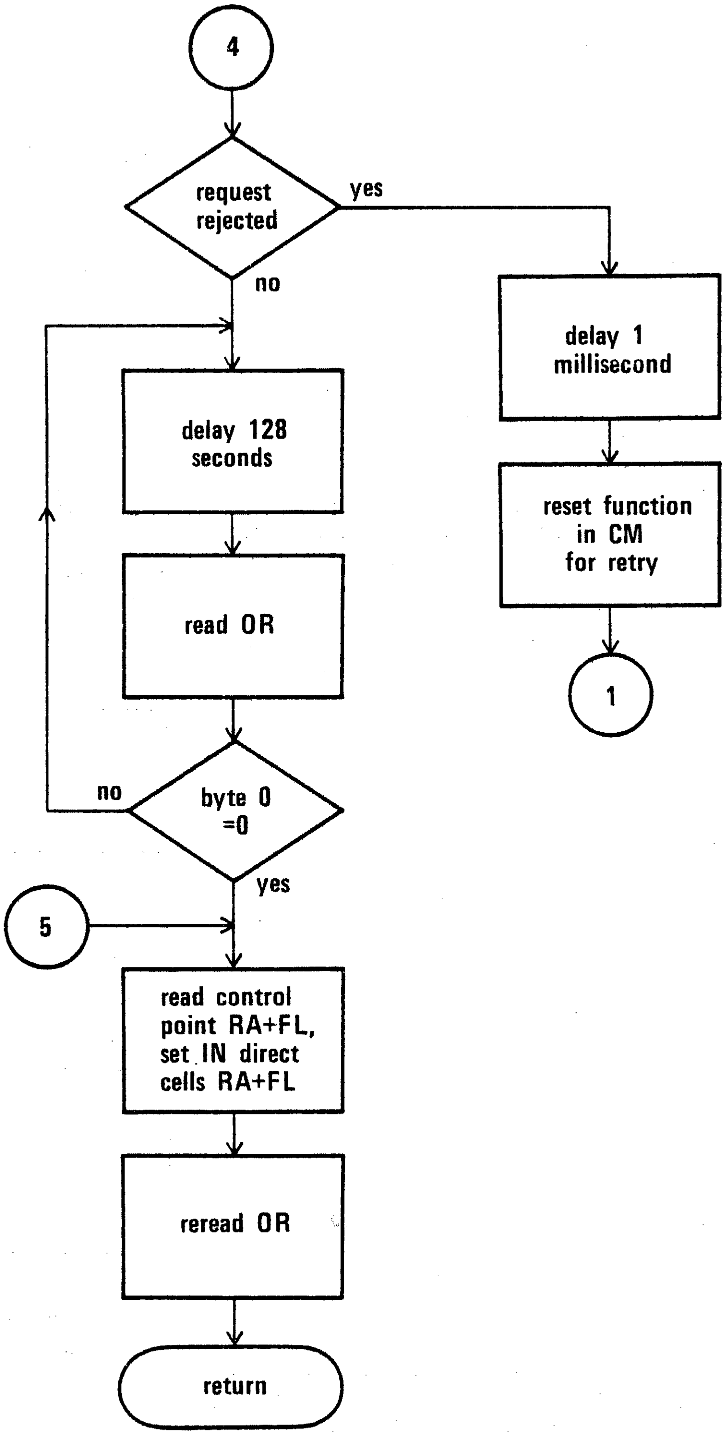
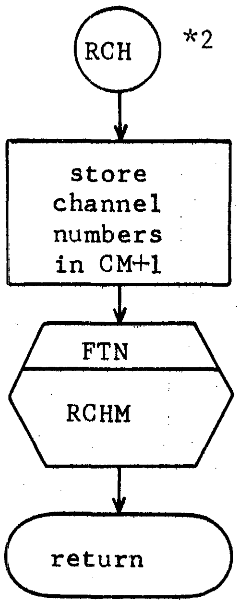


Figure 4-5. Process Monitor Function (FTN) (Continued)

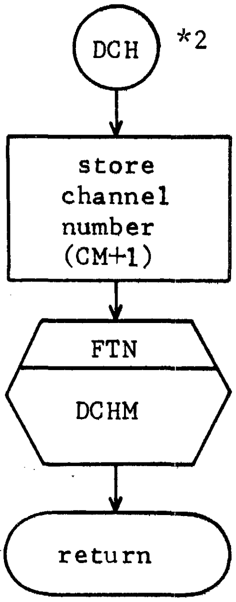
Entry (A) = 1 or 2 channel numbers *2
(CM+2) = additional channel numbers
(if more than 2 needed)

Exit (CM+1) = assigned channel



Reserve channel function

Entry (A) = channel number



Release channel function

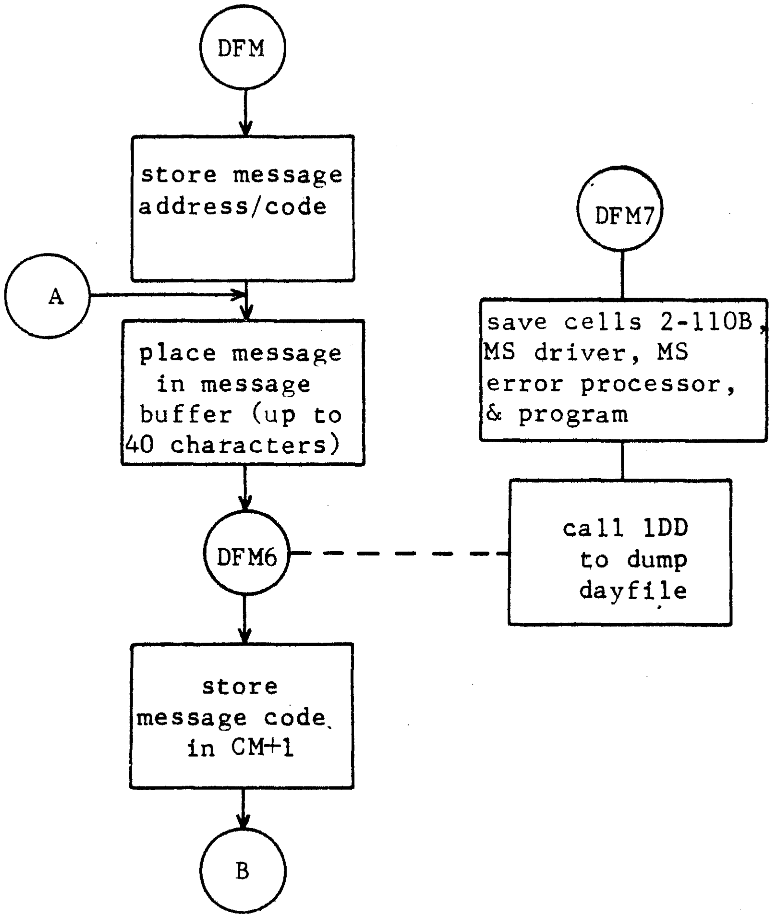
NOTE
Storage move
may occur while
this function
is pending.

*1 RCHM will assign one of the channels requested if it can. (A) and (CM+2) are used for optional channels.

*2 This entry point will not be supported in future versions of NOS.

Figure 4-6. Reserve Channel (RCH)

```
Entry     (A) = FWA of message (0-11)
                message code (12-17)
```



*1 Dayfile message function

Figure 4-7. Send Dayfile Message (DFM)

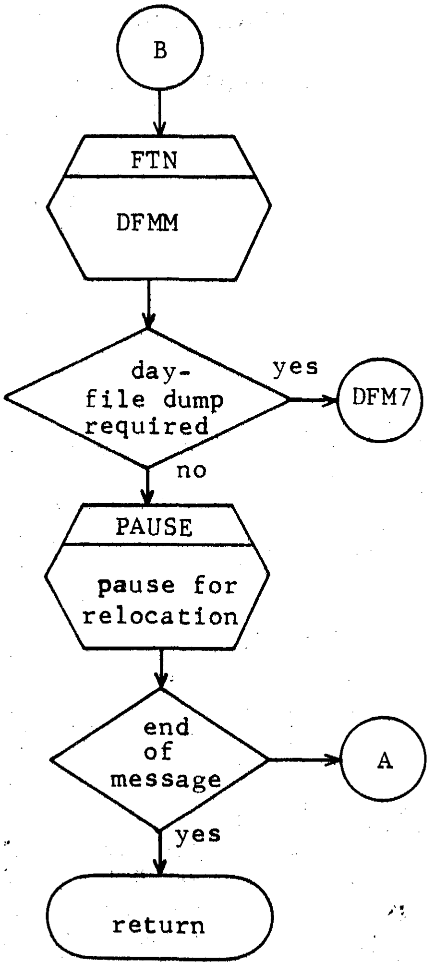


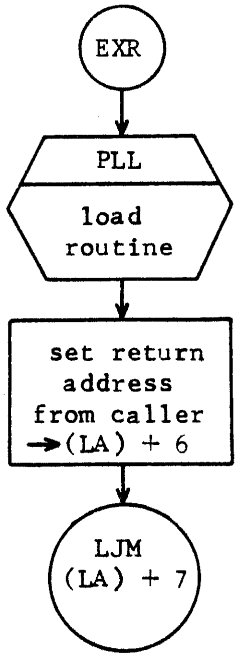
Figure 4-7. Send Dayfile Message (DFM) (Continued)

Entry (A) = Routine name
(LA) = Load address for location-free routines

Exit Exit to called routine via simulated return jump from caller

Example: Call overlay 2XY

(A) = 2XY
(LA) = load address
RJM EXR



then core from (LA) to (LA) + 7 is

(LA) + 0	2X
1	Y-
2	load address
3	0
4	length
5	0100 LJM
6	return address from caller of EXR
7	1st executable statement address

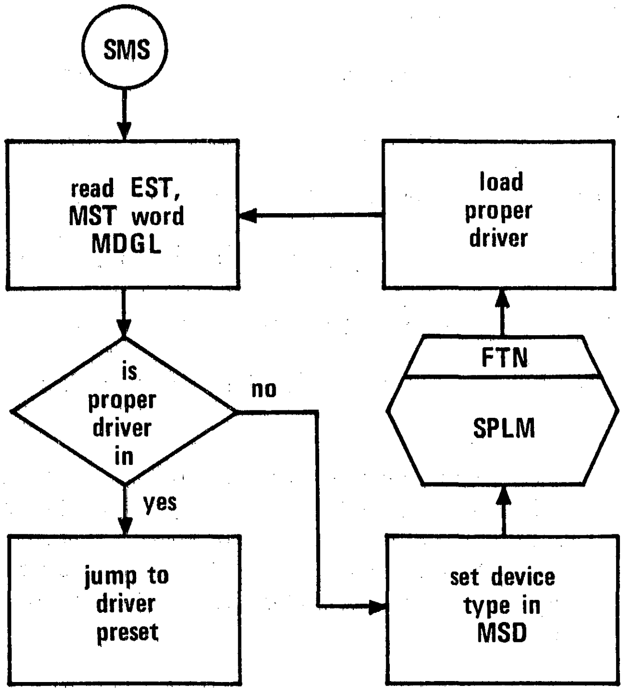
program 2XY at completion does a RETURN, which is a LJM (LA) + 5, which will LJM (return address from caller).

Figure 4-8. Execute Routine (EXR)

Entry (T5) = Est ordinal (refer to section 2 for description of EST entry)

The address of the initialize routines for all drivers begins at MSFW. These routines set the appropriate preset information for that equipment.

Exit (CM+1) through (CM+4) = EST entry bytes 1 to 4
Driver loaded if necessary
Driver initialized



*1 ESTS = FWA of EST
*2 SMS has stored the driver name in MSD when that driver was loaded, so that it can compare new driver requirement against the loaded driver.

Figure 4-9. Set Mass Storage (SMS)

DAYFILE MESSAGE OPTIONS

A normal dayfile message is sent to the master dayfile, control point dayfile, and control point message area. The job name is defined in the control point area. Following are the dayfile message options.

Option	Description
(00000)	Normal message
NMSN (10000)	Normal message with no message at control point
JNMN (20000)	Message to master dayfile only, with job name
CPON (30000)	Message to control point dayfile only
ACFN (40000)	Message to account dayfile only
AJNN (50000)	Message to account dayfile with job name
ERLN (60000)	Message to error log only
EJNN (70000)	Message to error log only with job name
FLIN (400000)	Flush and interlock dayfile

The FLIN option flushes the dayfile buffer and leaves the dayfile pointers interlocked. It is used in conjunction with any of the preceding dayfile options. If the message is issued to more than one dayfile, each is flushed and left interlocked. FLIN is used by SFM to terminate an active account, error log, or system dayfile.

MASS STORAGE DRIVER RESIDENT AREA

Mass storage drivers are overlays loaded by PP resident in an area between PP resident and the first word address of PP programs. Mass storage drivers are coded such that the entry points remain constant between all drivers.

Parameters passed to the driver are:

(T4) = channel
(T5) = equipment number
(T6) = track
(T7) = sector

The rules are:

- Name is the character 6 followed by the equipment mnemonic.
- Origin is MSFW.
- First word is the address of the driver initialization routine. This entry is used by SMS to cause initialization of the driver. Exit from initialization is to SMSX. SMS enters the initialization routine with CM to CM+4 = EST parameters, SLM-4 to SLM = MDGL word of MST.
- The entries for read, write, and position originated at the appropriate symbolic names (RDS, WDS, EMS). These entries are entered via return jump.
- The driver must not use any direct locations except T1, T2, CM to CM+4.
- The driver and its associated error processor must reside in RPL.

All drivers use the following three entry points.

RDS Read sector

Entry driver initialized (SMS called)

(T4) = channel (if driver previously called)
(T5) = equipment
(T6) = track
(T7) = sector
(A) = FWA of data buffer (502 word buffer needed)

Exit (A) = -0, if unrecoverable error

WDS   Write sector

Entry driver initialized (SMS called)

(T4) = channel (if driver previously called)
(T5) = equipment
(T6) = track
(T7) = sector
(A) = FWA of data buffer (502 word buffer needed) + WCSF for WLSF
(WDSE) = Write error processing buffer address (502 word buffer)

Exit (A) = -0, if unrecoverable error

(A) = -1, if recovered error on previous sector; current sector data and linkage bytes must be regenerated and reissued

EMS   End mass storage operation

Entry (T4) = channel, if RDS/WDS previously called
(T5) = equipment

All drivers begin at location MSFW.

Use of mass storage drivers is described in detail in section 7. Refer to table 4-3 for a list of symbols used with mass storage drivers.

TABLE 4-3. SYMBOLS USED WITH MASS STORAGE DRIVERS

Symbol	Value	Description
MSD		Mass storage driver identification
MSFW		FWA of mass storage drivers
RDS	MSFW+1	Read sector
WDS	MSFW+4	Write sector
EMS	MSFW+7	End mass storage
	Other mass storage processing constants	
BFMS		Sector buffer address
FSMS		First data sector of file
	System sector addresses	
FNSS	BFMS+2	FNT entry (five bytes)
EQSS	BFMS+2+5	Equipment number
FTSS	BFMS+2+6	First track
FASS	BFMS+2+11	Address of FST entry
DTSS	BFMS+2+12	Packed time/date

Whenever a PP program desires to read or write mass storage, the program always executes a SETMS macro with the appropriate option selected. A flowchart of SMS is illustrated in figure 4-9.

All jobs which flow through the system are processed from start to finish by PP routines 1SJ, 1AJ, 1CJ, 1RO, 1RI, and (in the case of time-sharing origin jobs) 1TA. Flow is controlled by the queue priorities and CPU priorities, in association with time and equipment limits. Depending on the resources needed by the job, all action is initiated, controlled, and eventually error- or end-processed by these routines.

All jobs are one of the following origin types.

Origin Type	Value	Description
SYOT	0	System origin includes all jobs entered by the operator at the system console, such as DIS, FST, MY1, and so on.
BCOT	1	Local batch origin jobs are entered from all local batch devices.
EIOT	2	Remote batch origin jobs are entered from the remote low speed batch terminals.
TXOT	3	All jobs entered via the IAF executive (IAFEX) or time-sharing executive (TELEX) are TXOT origin types.
MTOT	4	Multi-terminal origin includes jobs which do one specific task for many terminals while only being scheduled into the system once.

Figure 5-1 illustrates the general system flow for jobs.

GENERAL JOB PROCESSING

The priorities are controlled dynamically at the operators console and updated by routine 1SP. The job control (JCB) area in CMR contains the current values of these priorities for the system. Each job can be further restricted by the VALIDUs file, PROFILa file, or job statement parameters, but no job can be less restricted than the JCB. Routine 1SP also updates queue priorities in the input and rollout queues, checks central memory time slices, periodically calls 1CK to checkpoint all mass storage devices and CMS to initialize or recover mass storage devices online, issues dayfile messages for mass storage drivers that are unable to do so, and calls 0AU to process the accounting accumulator.

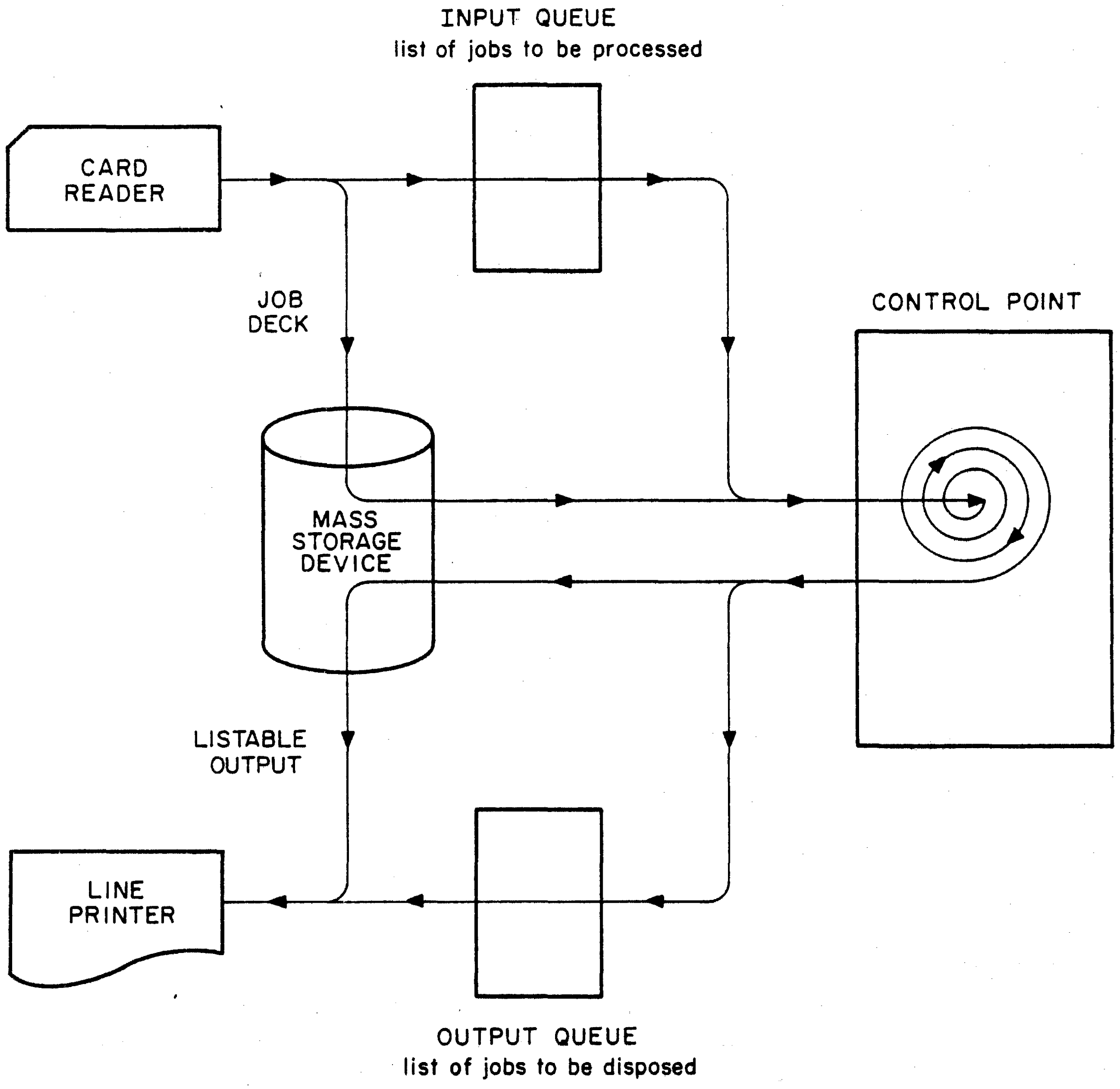
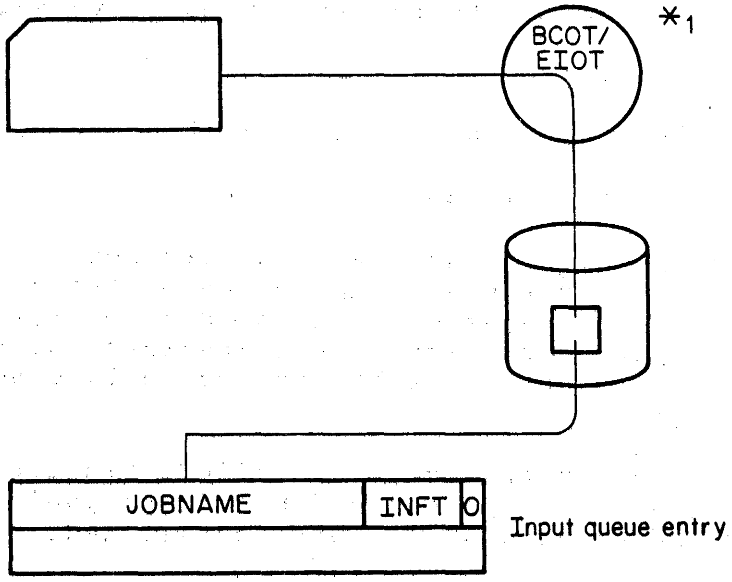


Figure 5-1. General System Flow

Jobs enter the system at the initial (original) queue priority for their origin type (figure 5-2). As they wait in the input queue, they are aged. The queue priority is increased until it reaches the upper bound priority, at which point the priority cannot be raised. At any time, the scheduler, 1SJ, may determine that this job is the best candidate (best job) for a control point by an algorithm that takes into account queue priority and resources desired (FL, etc.). It then attempts to schedule or assign it to a control point.



*1 TXOT/MTOT are started by IAFEX or TELEX and SYOT is initiated by DSD.

Figure 5-2. Read Card Reader

The job selection proceeds in the following order.

1. The highest priority job that will fit in unassigned or rolling memory with the service constraints FL/FLE (individual job field length) and AM (maximum amount of memory available) for the candidate's origin type.

2. If candidates of equal priority are found, the job selected is the one residing on the mass storage device with the least amount of activity. The amount of disk activity includes no free channel, channel being requested, and first unit reserved.

3. If the mass storage activity is also equal, the job with the largest field length is selected.

4. If no job is selected, but one was rejected due to service constraints, it may be scheduled if no jobs have to be rolled out. If this is done, its priority is set to the lower bound priority (LP). This prevents resources from being idle during periods of low activity.

When 1SJ assigns the best job to a control point, it gets the required FL, rolling out other jobs if necessary. It selects a control point according to the following criteria.

1. Exact fit

2. Smallest hole that is larger than needed

3. Largest hole if none is big enough

If no control points are available or are not in the process of rolling out, the first control point encountered with a lower priority than the candidate is selected to be rolled out. If all control points have higher priority than the candidate or control points are not available or are rolling out, no control point is selected.

Once a control point has been identified, its queue priority is set to the upper bound priority (UP) of the job's origin type and its CPU and CM time slices are initialized.

If the job is being scheduled from the input queue, 1AJ is called to begin the job; if the job is being scheduled from the rollout queue, 1RI is called to roll in the job (figure 5-3).

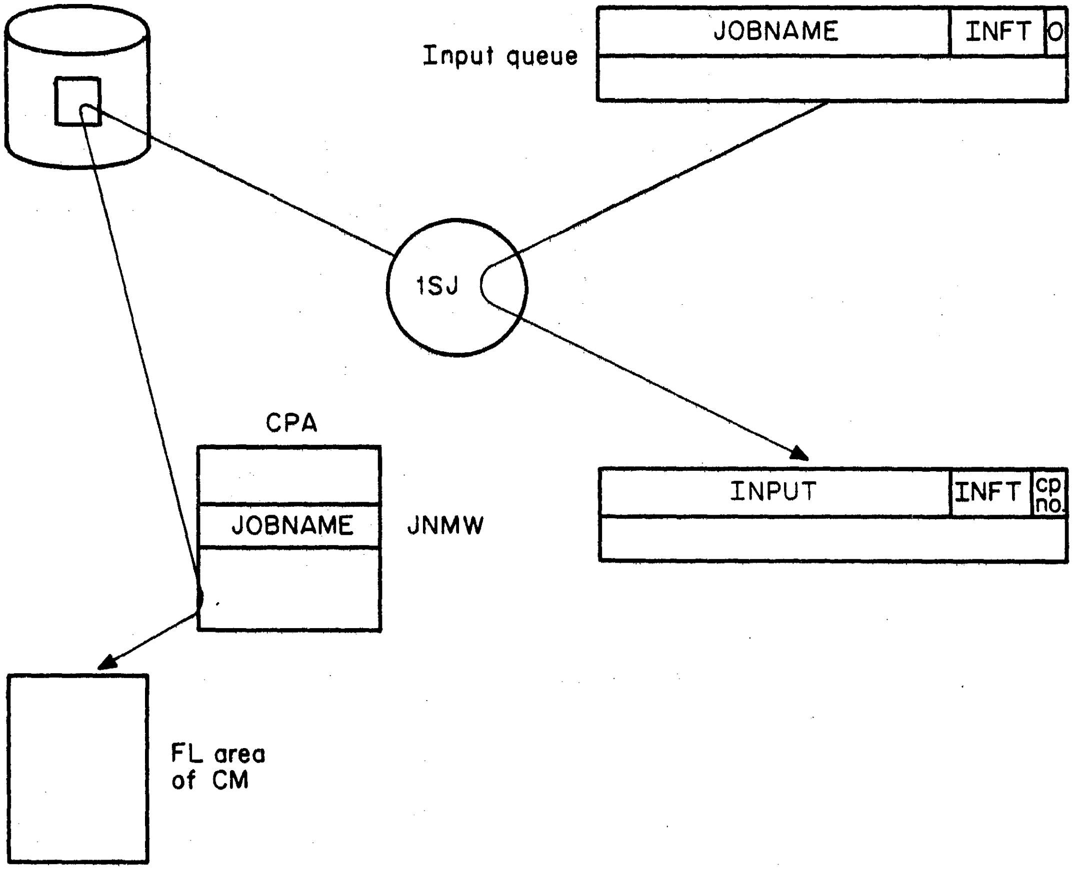


Figure 5-3. 1SJ Prepares a CP for the Job

The job advancement routine, 1AJ, knows it has been called by the scheduler and will call overlay 3AA (figure 5-4) to start this job up. The job can at any time create local files, and if the name is OUTPUT, PUNCH, PUNCHB, or P8 it is treated special at job completion time (figure 5-5).

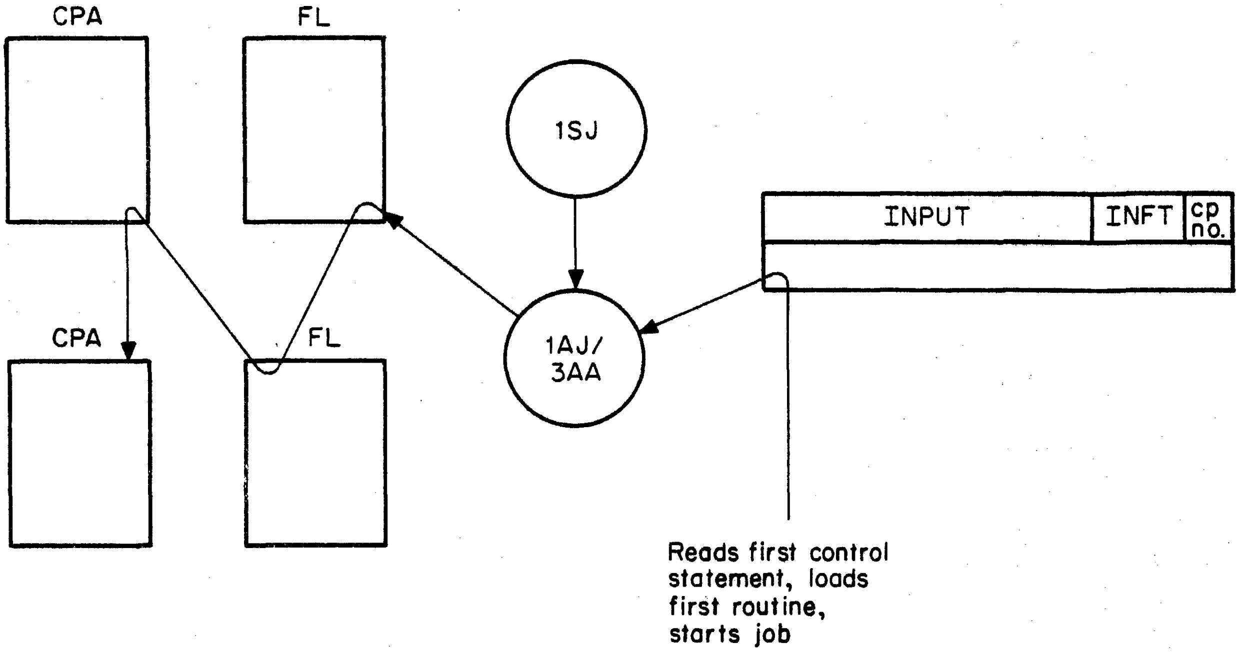


Figure 5-4. 1AJ Starts the Job

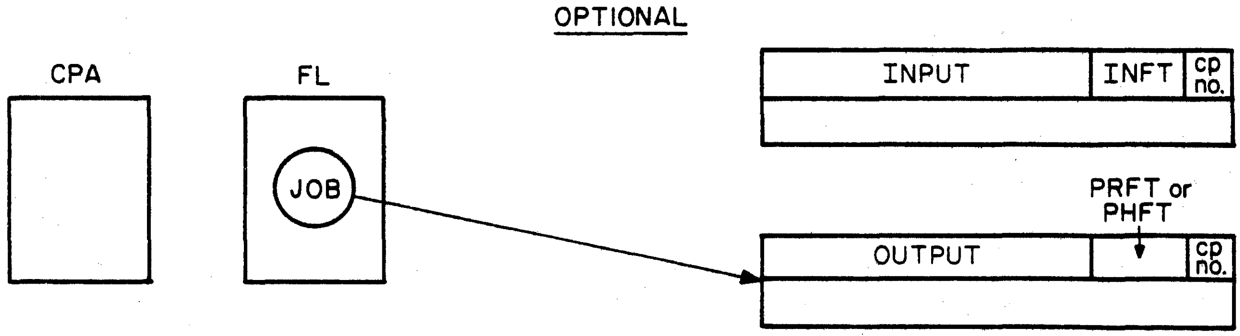


Figure 5-5. Job Creates Local File

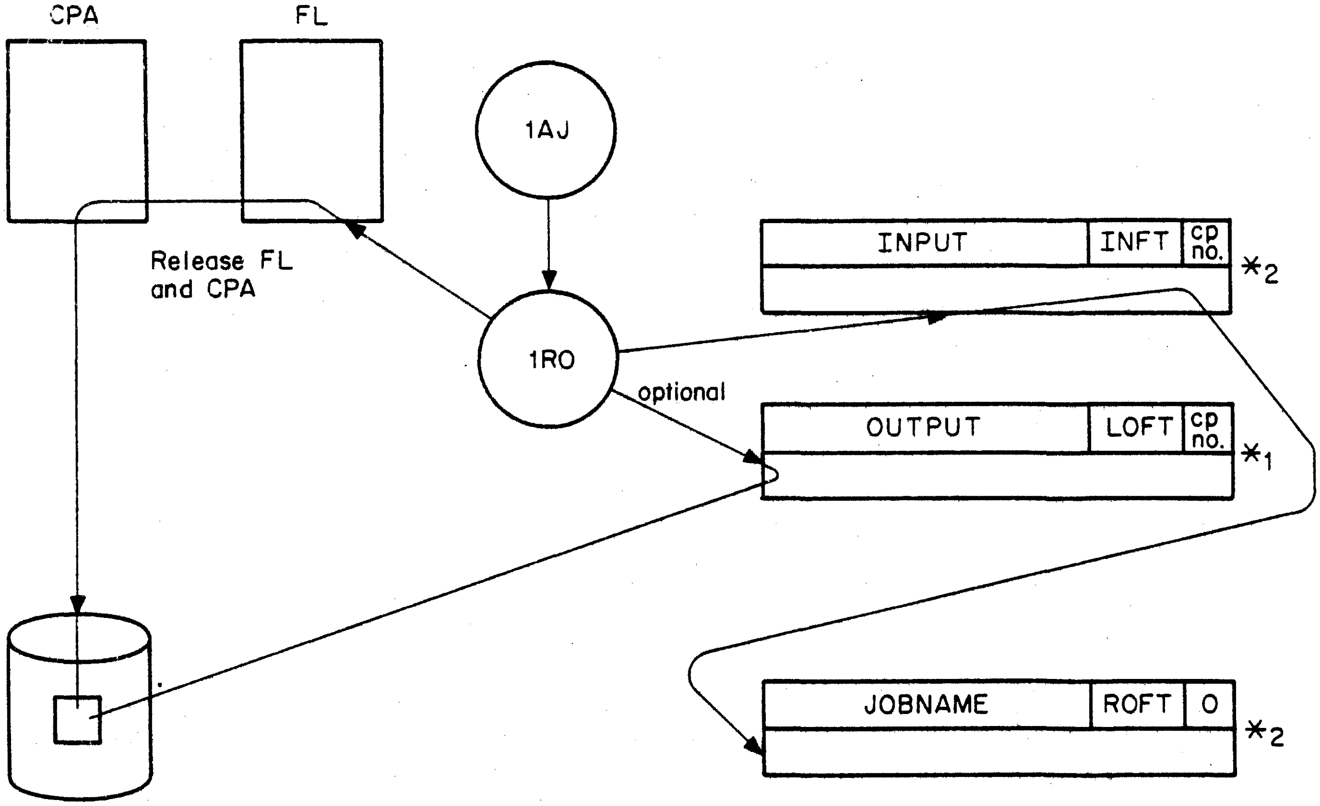
As the job progresses, CPUMTR and MTR periodically check all the jobs running at control points and call 1AJ if no activity is detected (W, X, and I status zero). If the error flag is set, 1AJ processes the error. If the error is nonfatal, 1AJ advances to the next control statement. If the error is fatal but an EXIT statement exists, 1AJ advances to the statement following EXIT. CPUMTR and MTR also monitor the CPU time slice, and if the job exceeds its time slice, its queue priority is dropped to the lowest queue priority (lp) of that origin type. This does not mean that the job loses its control point. If 1SJ finds a best job in the input or rollout queues, then low priority jobs are candidates for rollout. Also, 1SP monitors all the contol points, and if it detects that the CPU time slice is exceeded before either monitor does, it lowers the queue priority to LP. An interlock is provided in bit 35 (CPU time slice active bit) of TSCW in the control point area so its queue priority is only dropped once.

Routine 1RO may be called by 1AJ, 1SJ, DIS, and other routines (figure 5-6). It dumps the job according to the rollout file format, sets W, X, and I status to zero, requests the control point be made available, and releases all FL, nonallocatable equipment (tapes are not released, but the control point number in the EST is set to 37B), and all files assigned to this control point. The job is then placed into the rollout queue with whatever queue priority the job had when rollout was initiated. If 1RO is called as part of special entry point processing by 1AJ, the rollout file is called DM* and left assigned to this control point. Then 1RO releases everything else except the input and control statement file, and calls 1AJ to advance the job. In this way FNT space is not wasted while a job is rolled out.

Routine 1RI reads the rollout file and reestablishes all the files, equipment, and so on, to allow the job to continue (figure 5-7). It sets W, X, and I status to its former values. The control point is now a candidate for the CPU. A job always gets a fresh time slice when it is rolled in.

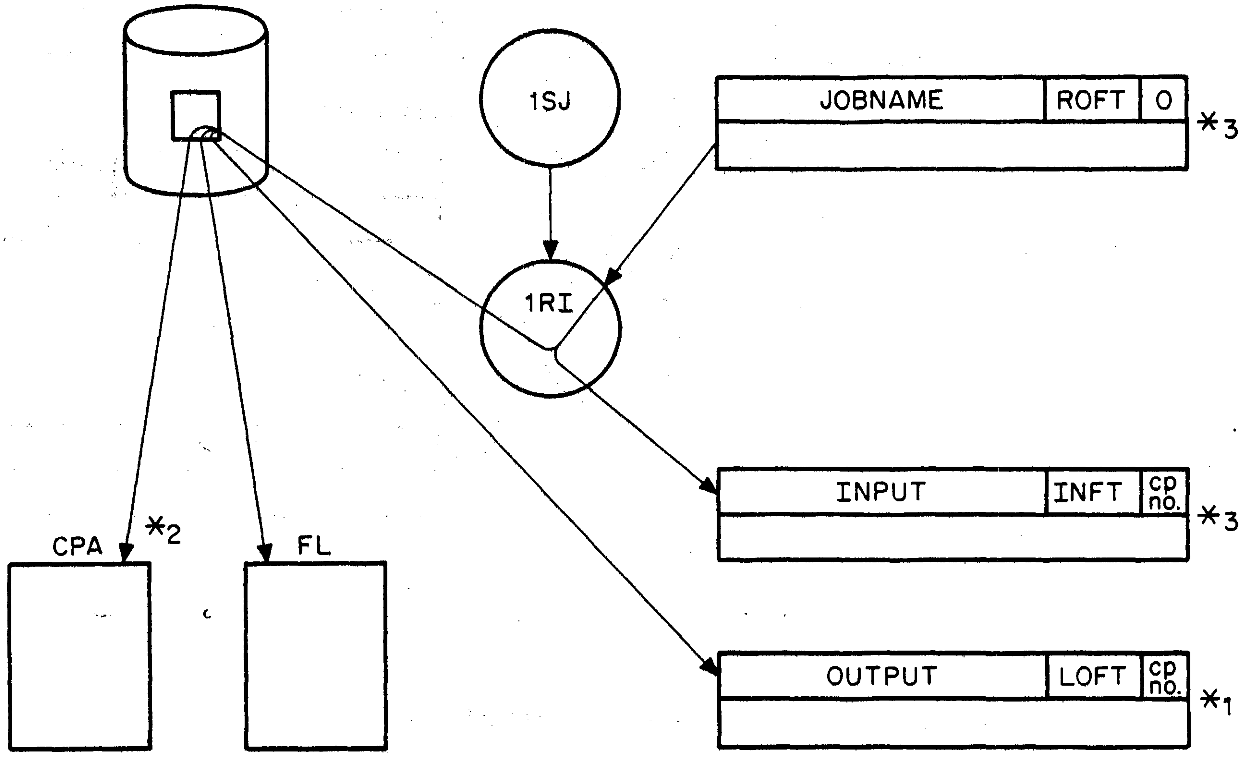
When 1AJ detects an end-of-job card stream, a fatal error with no recovery, an illegal control statement, or some other fatal condition, it calls 1CJ to complete the job. If any of the job flow routines ever detect an origin type which is not defined (type not SYOT, BCOT, EIOT, TXOT, or MTOT), it calls 1CJ immediately to end the job. This is protective coding.

Routine 1CJ locates the local file OUTPUT assigned to this job, if it exists (figure 5-8). It then appends the job dayfile to the end, writes an EOI, and moves the file to the output queue by setting the control point field to zero and setting the queue priority to the output queue entry priority (OP) for the origin type.



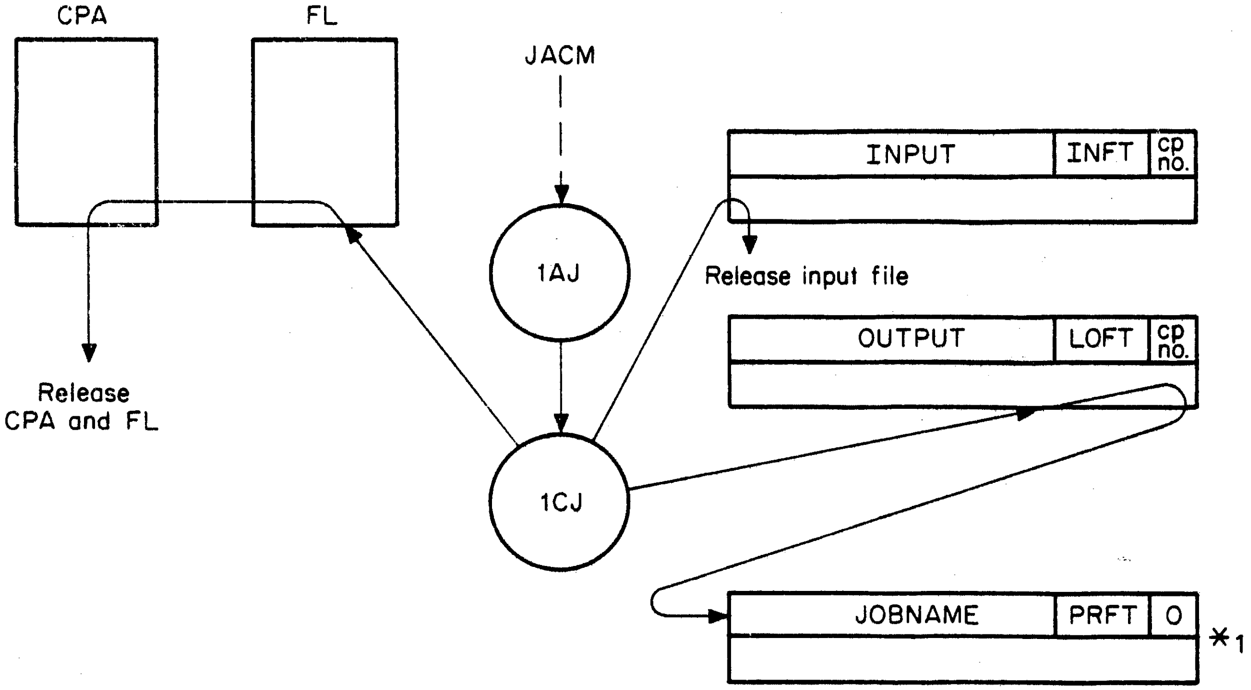
*1 And any other local files
*2 This is the same FNT entry

Figure 5-6. Job Is Rolled Out



*1 And any other local files
*2 Not necessarily same control point area and field length as figure 5-6
*3 This is the same FNT/FST entry

Figure 5-7. Job Is Rolled In (From Rollout)



*1 Same FNT/FST entry as local OUTPUT file.

Change OUTPUT file name to JOBNAME and file type from LOFT to PRFT. Append dayfile onto end of OUTPUT file.

1CJ also returns all files associated with this job except OUTPUT type files.

Figure 5-8. Job Completes

JOB FLOW

This section provides an overview of priority aging, rollout, scheduling, queues, and control statements. The details for the routines that do the actual processing (1AJ, 1RI, 1RO, 1SJ, 1SP, 1CJ) are presented in another section.

PRIORITY AGING

A job of a particular job origin type waiting in the input, rollout, or output queue is aged if its current priority falls between the lower priority and the upper priority limits.

A job is aged by the scheduler in conjunction with the job control area parameters in CMR. The job control area word is illustrated in section 2.

For each cycle of the priority increment routine (1SP), the counter (byte 4 of JCB) is incremented by one. This continues until the counter is greater than or equal to the age increment (byte 3 of JCB). At that time, the job queue priority is aged in the FST entry by one. Refer to the NOS Installation Handbook for the IPRDECK entries used to establish the JCB values for each job origin type, and the NOS Operator's Guide for the DSD commands to dynamically alter them.

QUEUES

The queues (input, output, rollout, for example) are FNT/FST entries in the FNT/FST table area of CMR. When a routine checks a queue, it searches the FNTs for entries with the appropriate file type which are not assigned to a control point.

When a job is moved from the input or rollout queues to a control point, the file name field of the FNT word contains INPUT instead of JOBNAME. The control point assignment field is set to the control point number and the queue priority is set accordingly (input or rollout UP).

When a job is sent to the rollout queue, the FNT name contains JOBNAME instead of INPUT. The file type is set to rollout (ROFT), the control point assignment field is set to zero, and the queue priority is set to whatever the control point area held at rollout time.

When a job completes, the special FNT name OUTPUT, if one exists, is changed to JOBNAME. The file type is changed from local (LOFT) to output (PRFT), the control point assignment field is set to zero, and the queue priority is set accordingly (output OP). This is also done for special files named PUNCH, PUNCHB, or P8 with the exception that their file type is changed to punch (PHFT).

ROLLOUT SCHEDULING

When a job is scheduled for rollout, the rollout-request flag, bit 24 in word JCIW of the control point area, is set and 1RO may or may not be called. When 1RO is called (by ROCM) it sets the rollout-in-progress flag, bit 27 in JCIW. When 1RO has rolled the job out, it resets these bits to zero. Also, if 1RO was called by a special entry point routine, 1RO sets these flags to zero. A special entry point job can also be scheduled to be rolled out. In this case, when 1RO is called it is a regular rollout, not a response to a special entry point job. Many copies of 1RO and 1RI can be run simultaneously.

SCHEDULER

Only one copy of 1SJ may run at any one time, and it can only be called by the monitor function RSJM. RSJM checks the scheduler active flag in JSCL+1 (bit 59) and if the bit is set, the scheduler is already active. If the bit is not set, monitor places a call to 1SJ in the next available PPU.

Any time the status of the system changes, 1SJ should assess the status and modify system flow as needed. The scheduler selects candidates as described earlier. It continues to select candidates until mass storage activity reaches a given limit or until no more candidates are found. In a normal job mixture, all jobs are eventually scheduled and any minor delay in the scheduling of one particular job is inconsequential to the total throughput of the system.

Figure 5-9 illustrates a typical queue priority scheme.

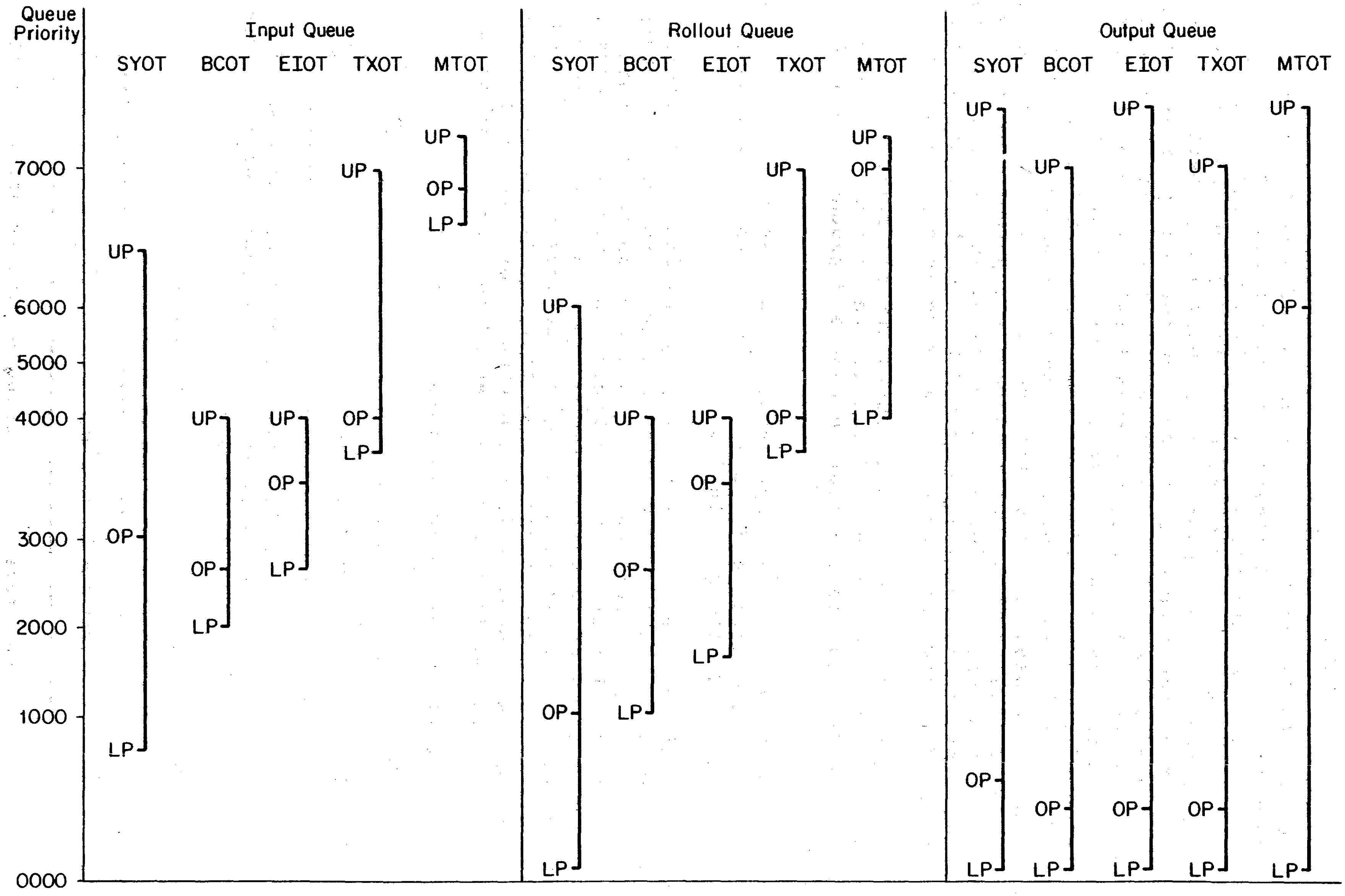


Figure 5-9. Typical Queue Priority Scheme

CONTROL STATEMENTS

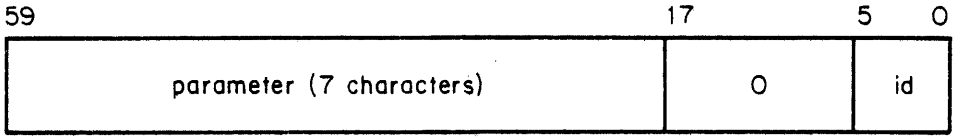
An overlay in 1AJ called TCS can be called directly from a CPU routine or by 1AJ. TCS (translate control statement) cracks a control statement and tests it for validity. Each control statement is a call to the system to load a routine whose entry point is the statement name (such as MODIFY and COPYBR). TCS disassembles the arguments, if any, on the control statement and makes them available to the routine. Then a search is made to locate the routine. First, the FNTs locally assigned to this control point are scanned, then the central library directory (CLD), and then the resident central library (RCL). If the routine is found in any of these, the first occurrence of the routine is loaded, the arguments are sent to it, and it begins executing. Thus, a programmer can define a program or routine local to his control point which may exist in the system already. If the control statement is preceded by a dollar sign ($MODIFY or $COPYBF, for example), the local FNT scan is bypassed.

If the entry point name is not found, the peripheral library directory (PLD) is scanned. If found, the routine is loaded into a PP (set IR equal to the routine name and argument) and TCS terminates.

If no match is found, an appropriate error message is issued to the dayfile and error procedures are initiated by setting the error flags and returning to 1AJ.

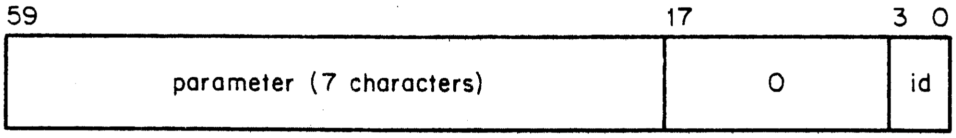
Before a CPU program is given control, TCS places the control statement image which called this overlay into central memory locations RA+70 through RA+77. Also, the control statement which was cracked by TCS and parameters are placed in locations RA+2 through RA+62 terminated by a zero word. If the control statement is preceded by a slash, the parameters are cracked in operating system format; otherwise they are cracked in product set format. All compiler (FTN and COBOL, for example) binaries expect control statements to be cracked in product set format.

- Operating system format (6-bit ID code):



id	0 for all separators except = and /, and in those cases the character is placed in the 6 bits.

- Product set format (4-bit ID code):



parameter	String of characters up to the separator

id Separator equivalence:

id	Separator
0	Continuation (for literals)
1	
2	=
3	/
4	(
5	+
6	-
7	Space
10	;
17	Termination) or .

For example, the control statement

`MODIFY(I,P=0,N=FILE,A,NR,X,CL)`

would be passed as follows: PGNR = RA + 64B = MODIFY 11B

```
              Operating System                     Product Set

                   42          12   6                   42          14   4
        ----------------------------------    ----------------------------------
RA+2    | I                | 0 |         |    | I                | 0 | 1       |
        |--------------------------------|    |--------------------------------|
   3    | P                | 0 | =       |    | P                | 0 | 2       |
        |--------------------------------|    |--------------------------------|
   4    | 0                | 0 |         |    | 0                | 0 | 1       |
        |--------------------------------|    |--------------------------------|
   5    | N                | 0 | =       |    | N                | 0 | 2       |
        |--------------------------------|    |--------------------------------|
   6    | FILE             | 0 |         |    | FILE             | 0 | 1       |
        |--------------------------------|    |--------------------------------|
   7    | A                | 0 |         |    | A                | 0 | 1       |
        |--------------------------------|    |--------------------------------|
   8    | NR               | 0 |         |    | NR               | 0 | 1       |
        |--------------------------------|    |--------------------------------|
   9    | X                | 0 |         |    | X                | 0 | 1       |
        |--------------------------------|    |--------------------------------|
  10    | CL               | 0 |         |    | CL               | 0 |17       |
        |--------------------------------|    |--------------------------------|
  11    | Binary Zeros                   |    | Binary Zeros                   |
        ----------------------------------    ----------------------------------
```

6-bit code is display character when used and binary zeros when blank.

Full word of zeros terminates control statement.

4-bit code is binary number.

One word of zeros preceded by other than a code 17 implies another control statement.

The flow chart in figure 5-10 shows the flow of control statement processing. Routine 1AJ processes CTIME, RTIME, and STIME directly.

Local absolute files with multiple entry points cannot be loaded. However, local relocatable files with multiple entry points can be loaded.

The type of automatic parameter cracking depends upon whether the load is from a system or local file. If a system load, the default is operating system format unless *SC is specified in LIBDECK. If a local load, default is product set format unless a slash (/) precedes the control statement.

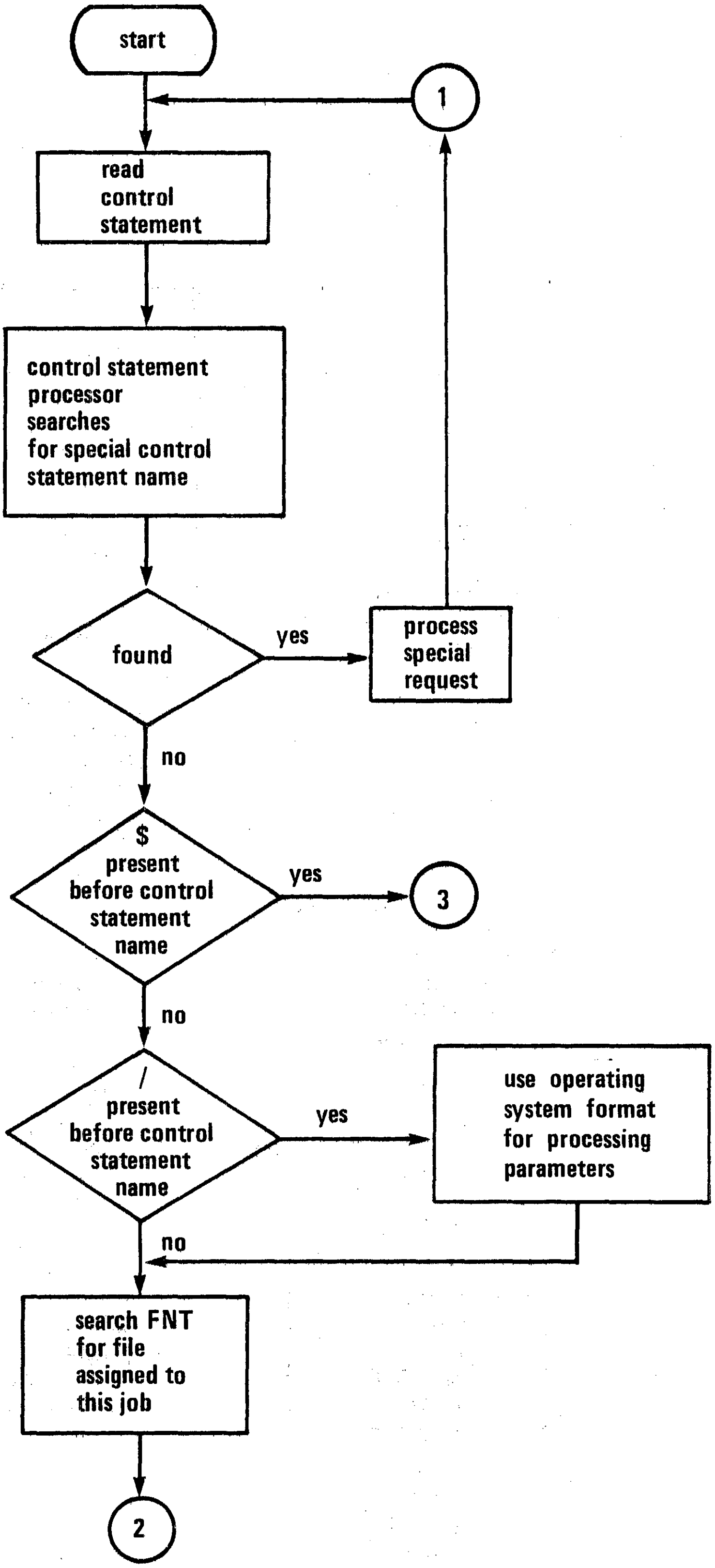


Figure 5-10. Control Statement Processing

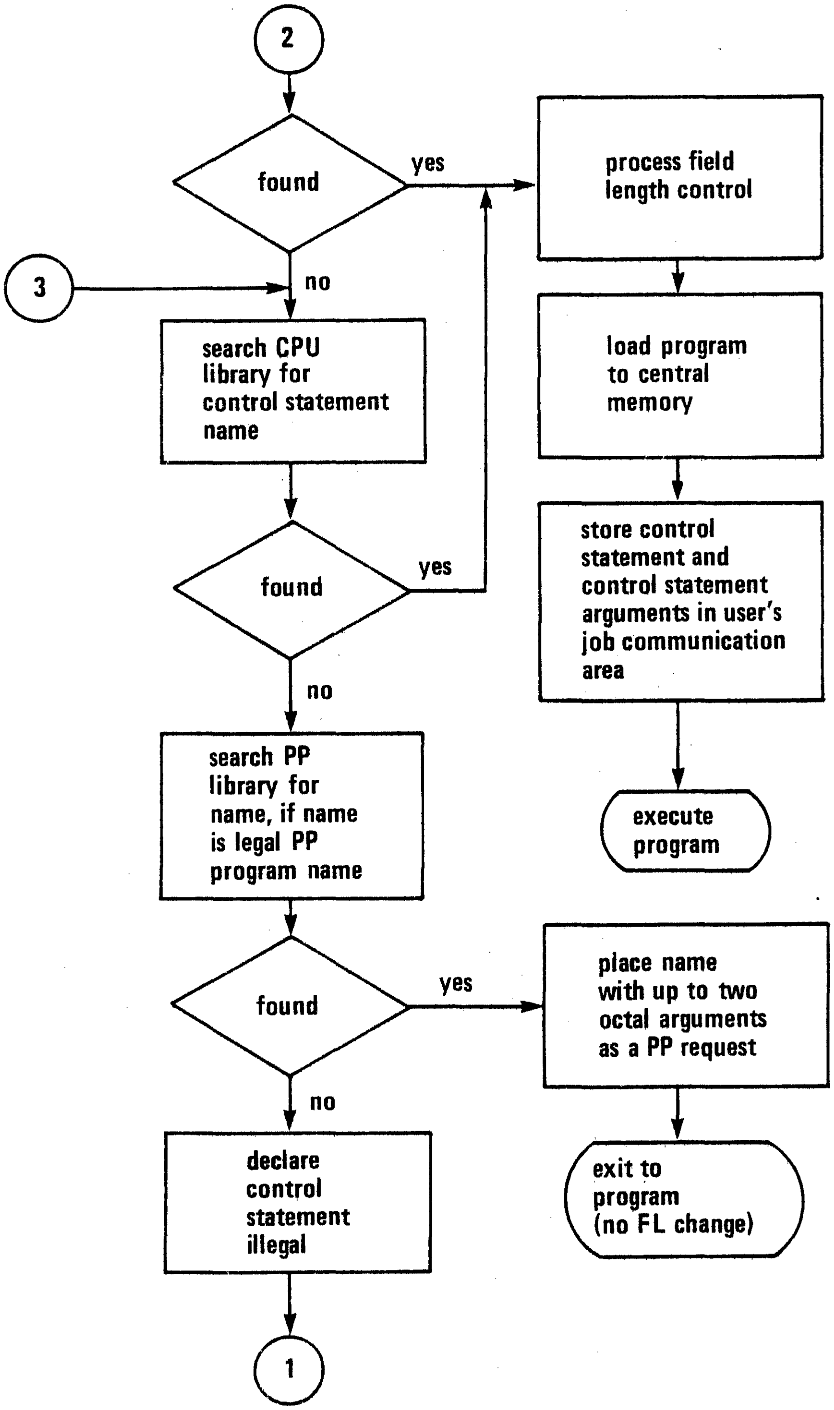


Figure 5-10. Control Statement Processing (Continued)

SPECIAL FILE INPUT*

When the user returns the file INPUT, file INPUT* is set up to point to the input file, but the user cannot access it.

When a procedure file call is encountered, the procedure file is expanded on file INPUT*.

When a procedure file from the system is encountered, a dummy call is generated to the CPU routine CONTROL or BEGIN (if a CCL procedure) and the expanded file is pointed to by INPUT*.

When any combination of the preceding occurs, INPUT* is used to link up the several files.

NOTE

> The file INPUT* may not explicitly exist for precedure file calls. Thus there is no FNT/FST entry, but INPUT* is pointed to by CSPW in the control point area (bit 59 in word CSSW).

TIMED/EVENT ROLLOUT PROCESSING

When a CPU program goes into timed/event rollout, it uses the ROLLOUT macro and specifies an event and/or a time. Routine 1RO is called to roll the job out and create an FNT/FST with file type TEFT (refer to section 2).

When 1SP is called by 1SJ it checks each entry in the TEFT queue and if the rollout time period has expired it changes the entry to a regular ROFT entry. If the time period has not expired, 1SP uses the EATM monitor function to read the event table from MTR's field length. It compares the events with this 18-bit event descriptor and if there is a match 1SP changes the entry to a regular ROFT entry (refer to section 2).

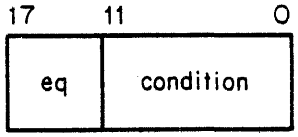
EESET Macro

Only PP programs may access the event table via the EATM MTR request. Therefore, the macro EESET allows a previously set event to be matched by an event set by a CPU program. The format of the EESET macro is as follows.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	EESET	event

event	18-bit event descriptor

The event is an 18-bit value that has the following format.



eq	EST ordinal of equipment on which the system is waiting for condition to occur.
condition	Variable event condition.

EESET calls CPM to enter an event descriptor into the event table. A job must have SYOT origin to use the EESET macro.

The only PP routines currently using the EATM function are the following.

- CPM for EESET enter event.
- IMS and MSM to specify when a removable pack has been initialized or recovered (for missing pack name event).
- ORP to specify when a write mode permanent file is no longer busy and to specify when a removable pack has been returned and has no more users (for overcommitment event).
- OFA to specify when a write mode fast attach file is not busy.
- 1DS to specify when the operator has supplied a VSN.
- 1MT to specify when a VSN has been mounted (for missing VSN event) or when a tape unit has been returned (for overcommitment event).

DSD and DIS Commands

In all DSD file displays the timed/event rollout files are displayed as TEFT file types. In addition, the Q display has all TEFT rollout files flagged by **.

The DSD command, ROLLIN,xx. may be used to roll in a TEFT job.

For a job at control point n, the DSD command n.ROLLOUT,xxxx. will roll the job out for xxxx seconds.

The following command to roll a job out for a time period may also be used under DIS.

```
ROLLOUT,xxxx.
```

Description of Timed/Event Rollout

The timed/event rollout feature allows jobs to access system resources as they become available. Through use of the ROLLOUT macro, the user may request to be rolled out until an event occurs or time period expires. If the desired event does not occur within the specified time period, the job is scheduled to roll in for further processing anyway.

To determine when a specified event has occurred, a system event table is maintained in MTR's memory. System programs can make entries to this table to indicate occurrence of events. Routine 1SP compares the requested event with the system events recorded in this table to determine if any matches have occurred. If a match occurs, 1SP initiates roll in. If no one is waiting for the system events they are cleared from the table.

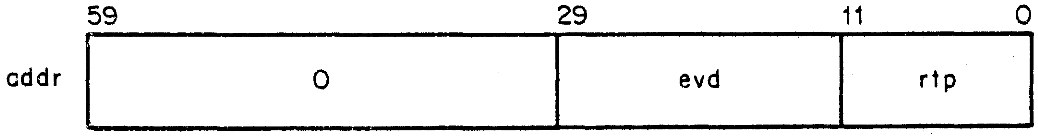
ROLLOUT Macro

The format of the ROLLOUT macro is as follows.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	ROLLOUT	addr

addr	Optional address containing further parameters

If addr is not specified, the job rolls out until the operator initiates rollin. If addr is specified, the job is rolled out for the specified time and event description. The format of addr is as follows.



rtp	Rollout time period in job scheduler delay intervals (0≤rtp≤7777B). If rtp = 0 the job rolls out for a time determined by the system to insure that the job will roll in if the event for which it is waiting for is lost or never occurs.
evd	Event descriptor

If evd is nonzero, the event descriptor and rollout time period, rtp, are placed in the control point area (TERW). When the job rolls out it waits for the occurrence of the event in evd or the specified time period (rtp) to elapse before becoming eligible for roll in.

If evd is 0, the event is taken from the control point area if bit 30 in TERW is set to indicate a valid event descriptor and only the rollout time period is taken from addr. This option allows the user to roll out waiting for events that the system specifies.

If evd equals 7700xxB, then extended timed rollout is made. (Assume the job scheduler delay is 1 second.) Since the maximum time rtp can specify is approximately 1 hour and 8 minutes, the extended time rollout allows the user to roll out for any length of time. This is a strict time rollout with no event dependency. The job rolls out for (4096*xx+rtp) seconds.

The ROLLOUT macro calls CPM to read the rollout time and event from the users field length and store it into control point area address TERW. CPM then does a ROCM and control is returned to the user. The user then can execute until the rollout bit is detected by MTR who initiates 1AJ, who calls 1RO. In order to insure the rollout, the user must issue a PP request, since CPUMTR will not honor a PP request for a control point scheduled for rollout. CPUMTR places the control point in I autorollout status with an outstanding RA+1 request. The simplest method is to build a dummy FET and issue the RETURN macro. This issues an RA+1 request to CIO.

MTR detects that this control point is in I status and is scheduled for rollout and calls 1AJ, who calls 1RO.

Routine 1RO rolls the job out and then checks control point area address TERW. If it is zero, this is a regular rollout. If it is nonzero, then 1RO builds a TEFT type FNT and places the event and time limit from UPCW into the FST. Routine 1RO then clears TERW.

When the job rolls in, MTR finds the control point in I status, and an RA+1 request. MTR calls CPUMTR with a zero request and CPUMTR then honors the RA+1 request. In the case of the RETURN dummy, CIO treats it as a null operation (file does not exist) and terminates. Then the control point can continue.

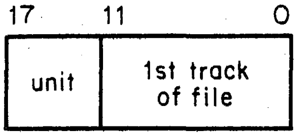
Example 1

An attempted attach results in file busy status.

Assume error processing is set. Upon restarting the job, use of the ROLLOUT macro with evd equals 0 rolls the job out for the time specified by rtp, waiting for the event (file ready to be accessed) to occur. Routine ORP enters this event in the system event table when the file becomes not busy. PFM stored the descriptor for this event in the control point area (TERW) when it found the file busy but it did not set the rollout flag, allowing the user to choose whether to rollout immediately, or to process some other function first.

If error processing is not set the job is automatically rolled out, waiting for the file to be ready to be accessed. When the job rolls back in, the ATTACH request is retried.

The event for example 1 is as follows.



When a user attempts to access files that are interlocked, the system automatically sets the error flag and terminates the job step until the file becomes available (unless the user is doing his own error processing).

The user may bypass this automatic job step abort, by specifying the NA option on the ATTACH control statement, so that the job step is not aborted if the file is busy.

The user calling PFM via the macros provided, can avoid job step abort by specifying error processing. If error processing is specified, the system returns control to the user with error status reflecting file busy.

Example 2

Suppose that before JOB1 continues processing that it wants JOB2 (a system origin type job) to execute a certain function. Assume JOB1 uses the rollout macro with evd = 1300 and rtp = 600. The rollout flag will be set for JOB1 to rollout for 600 seconds or until event 1300 takes place. Before the 600 seconds has elapsed, suppose JOB2 makes the macro call EESET 1300, entering the event 1300 in the system event table. JOB1 will then be scheduled for rollin to resume processing. If 600 seconds elapse because event 1300 has not occurred (or the event was cleared from the table before JOB1 rolled out), JOB1 will be scheduled for rollin.

In any case, JOB1 does not know if it was rolled in because of time or event occurrence. Hence, it is necessary for JOB2 to do something; for example, write a code word on a permanent file which JOB1 can check to see if the event occurred.

This job dependency can be accomplished by JOB2 attaching a direct access file in write mode and then JOB1 doing the same. JOB1 will wait as in example 1 for JOB2 to release the file. However, if JOB1 gets the file first, it must release the file for JOB2 and then attempt to attach it again. In order to use EESET effectively, an installation must change CPM to accept other origin types that issue EESET. This solution may cause the filling of the event stack. So, a change to CPM warrants careful consideration by the installation to limiting the number of EESET requests per origin type.

Example 3

A user requests a magnetic tape with a specified volume serial number (VSN) or a removable pack with a specified pack name. RESEX, the resource executive which is called to allocate magnetic tape and removable pack resources, will effect a timed/event rollout if it does not find the specified tape or pack mounted. The event used is the sum of the bytes in the VSN or pack name, truncated to 12 bits. The equipment portion of the event descriptor is 76B, which is equivalent to the timed/event EST entry.

When 1MT reads the VSN from the tape or IMS and MSM initialize or recover a removable pack, the matching event is entered into the event table in MTR via the EATM function. Routine 1SP then detects a match and has the job scheduled for rollin.

FNT INTERLOCKING AND SCHEDULING

A transition state is defined to be the state in which a job may be in the process of rolling in or rolling out. The concept of the individual FNT interlock provides better protection for jobs and files that are in a transition state than was previously provided by the technique of disabling job scheduling. The following paragraphs describe the various FNT interlock mechanisms, how they are used to protect jobs and files that are in the transition state, and the impact they have on scheduling.

INDIVIDUAL FNT INTERLOCK

Interlocking an individual FNT entry is accomplished through the monitor function SFIM (set FNT interlock). This function sets or clears an interlock bit for a particular FNT entry. The interlock bit for each FNT entry is kept in the FNT interlock table which is appended to the FNT. The interlock on an individual FNT entry should be held for the shortest time possible to avoid performance degradation.

This technique is used in the following circumstances:

- Bringing an input file into execution.
- Performing a job advance.
- Rolling in or rolling out a job.
- Terminating a job.
- Altering the FNT or system sector of a queued file.
- Moving a file from one queue to another.
- Assigning a queue file to a control point.

The format of the SFIM monitor function is described in the NOS Systems Programmer's Instant.

GLOBAL FNT INTERLOCK

The FNT may be globally interlocked by the reservation of the FNT pseudo-channel (FNCT). The use of this mechanism is to avoid conflicts which may occur when more than one system routine attempts to update the FST entry of a queued file. The global interlock is only used when the contents of queued file FSTs are to be altered. This interlock should be used with caution as the priority evaluation scheme is disabled by it.

In cases where the individual FNT interlock (SFIM) and global interlock (FNCT) are both required, the SFIM interlock should be obtained first and then the FNCT channel reserved. This order must be maintained to avoid a deadlock situation.

An example of where the FNCT interlock is used is the DSD command ENQP. Routine 1SP is periodically called to do queue priority evaluation and updates the priority field in queued file FSTs. DSD updates the priority field in a queued file FST in performing the ENQP command. If both DSD and 1SP tried to update the same queued file FST, a conflict would occur. DSD performs the following sequence to avoid the possibility of making a conflicting ENQP entry. First, the desired FNT is interlocked via the SFIM mechanism. Then the entire FNT is interlocked by reserving the FNCT pseudo-channel. After DSD updates the appropriate information in the FST, the pseudo-channel is released, clearing the FNCT interlock, and the individual FNT interlock is cleared using a SFIM monitor function.

FNT ENTRY INTERLOCK

The FNT is also globally interlocked by those system routines making new FNT entries by the reservation of the FNT entry pseudo-channel (FECT). This mechanism guarantees that a system routine may determine where within the FNT to write the FNT/FST entry without being disturbed by another system routine making FNT/FST entries.

An example of the use of the FECT interlock mechanism is found in routine OBF. Routine OBF obtains the FNT entry interlock by reserving the FECT pseudo-channel. The FNT is then scanned for an empty position. Routine OBF writes the FNT/FST entry at this location and then releases the pseudo-channel, clearing the FECT interlock.

JOB ADVANCEMENT

The individual FNT interlock must not be set on the job's input file in order for the job to be advanced. The job advancement process automatically sets the FNT interlock on the job's input file to indicate that it is in a transition state. Thus, the FNT interlock is always set for a job if the job advancement flag (bit 53 in control point area word STSW) is set. (The converse of this is not true; that is, the presence of the FNT interlock does not imply that the job advance is set for the job.) The issuance of the JACM (job advancement control) monitor function by the system routines involved in the advancement process (1AJ, 1RO, and 1CJ) clears the FNT interlock when the job advance flag is cleared. To facilitate the setting and clearing of the individual FNT interlock during job advancement, all jobs have an input file whose FST address is contained in control point area word TFSW bits 59 through 48. The job advancement process, including the JACM function, sets or clears the individual FNT interlock for the FNT/FST entry pointed to by TFSW.

TRANSITION STATE SCHEDULING

For system routines to properly control transition state activity it is necessary to set the FNT interlock on the queued file or input/rollout file being manipulated before any transition activity may take place.

The following example shows how the individual FNT interlock is used during the rollin and rollout transition states. In following the example, remember that the same FNT position is occupied by the job's rollout (when the job is rolled out) and the job's input file (when the job is rolled in).

In the case of rolling in a user job, the scheduler (1SJ) selects the job and then sets the FNT interlock on the rollout file before assigning it to a control point. During the rollin process, 1RI replaces the FNT/FST entry for the rollout file with that for the job's input file and sets this FST address into TFSW. When 1RI requests the job to be advanced, the FNT interlock is cleared.

In the case of rolling out a user job, the rollout request (ROCM) issued by the scheduler causes the job advance flag to be checked. If the FNT interlock (on the input file) is already set, the job advancement is requeued for reissuing. If the conditions for job advancement are met and the FNT interlock is not set, both the FNT interlock on the input file (as determined through the TFSW entry) and the job advance flag are set, and 1AJ is called. Then 1AJ calls 1RO. Routine 1RO writes the rollout file FNT/FST entry at the address specified by TFSW and issues a JACM function to clear the job advance flag, the FNT interlock (which is now on the rollout file), and selected control point area words including TFSW.

With the individual FNT interlock structure, system routines are able to identify when transition states are completed by the successful issuance of their own FNT interlock request. This in turn prohibits a transition state from occurring while they perform their specified function on that job or queued file.

SPECIAL PROCESSING

This section overviews the processing of subsystems, special entry point jobs, and special RA+1 requests.

SUBSYSTEMS

A subsystem is a special type of job with many privileges not granted to user jobs within the system. Some of the characteristics of a subsystem are:

- Cannot be rolled out except in system checkpoint situations.
- Can make use of the intercontrol point communication and special RA+1 requests (SIC and RSB) for receiving and sending data buffers.
- Can get a CPU priority above user jobs.
- Need not be restricted by JCB or VALIDUs; however it must have a user index set in UIDW, in order to access permanent files.

- May elect to run at a specific control point.

- Has an implicit special entry point (SSJ=) status.

- Can request the CPUMTR to load a PP routine whose name begins with a numeric (RA+1 call SPC). (Any PP request from a normal job must be for a PP routine whose name begins with a letter. Any other PP call aborts the CPU program.)

In order for a job to qualify as a subsystem, it must satisfy each of the following requirements.

- Have a queue priority greater than LSSS (defined in NOSTEXT) and have a byte for it in the SSCL words in CMR.

- Have an entry defined in 1DS so that it can be called from a DSD command.

- Have a unique queue priority, since it interacts with the system based on its queue priority and not on its user index, name, or control point number.

The current subsystems and their queue priorities are described as follows.

Subsystem	Symbol	Queue Priority
Deadstart Sequencing	DSPS	7777
Time-sharing (TELEX or IAF)	TXPS	7776
Remote Batch (EI200)	EIPS	7775
Unit Record (BATCHIO)	BIPS	7774
Magnetic Tapes (MAGNET)	MTPS	7773
Transaction (TAF/TS,TAF/NAM)	TRPS	7772
Time-sharing Stimulation (STIMULA)	STPS	7771
Network Interface Processor (NIP)	NMPS	7770
Remote Batch Facility (RBF)	RBPS	7767
CYBER Data Management Control System (CDCS)	CDPS	7766
Message Control System (MCS)	MCPS	7765
Mass Storage Control (MSM)	MSPS	7764

Subsystem Startup

A subsystem has a PP program that initializes the subsystem. For example, TELEX has 1TD; MAGNET, 1MT; EI200, 1LS; and so forth. In many cases, the PP program is also the driver for the subsystem in addition to performing its initialization. As an example in this discussion, the initialization of the remote batch facility (RBF) subsystem is used. The PP routine 1SI performs the control point initialization for RBF (as well as several other subsystems).

The jobs for subsystem initialization are entered into the input queue by 1DS functions 32 and 33. Function 33 is used when the subsystem is activated by default when an AUTO. is done; function 32 is used when the subsystem is activated by entering the DSD command for the individual subsystem. Routine 1DS maintains a table of parameters from which the FNT/FST input queue entries for the subsystems are built. An entry in this table has the following format.

byte 0	byte 1	byte 2	byte 3	byte 4
qp	pp	cp	sm	sb

qp	Subsystem queue priority as described previously
pp	Name of the PP processor that performs the subsystem control point initialization
cp	Relative control point number required by the subsystem
sm	Mask bit setting (12 bits) that corresponds to the subsystem enabled/disabled bit for the subsystem in SSTL
sb	Byte in SSTL to which sm applies

Subsystems have a requirement to reside at a given control point in order to minimize the system overhead used by the subsystem (for example, never storage moved). The number 1 for cp indicates that control point 1 is required; 2, control point 2 required; and so on. If cp is greater than 40B, the required control point is determined as the system control point minus 1 minus the complement of cp. Thus, the value 77B indicates that the last control point is required; 76, last control point minus 1 is required; and so on. If a rollable job is at the control point, it is rolled out so that the subsystem may have the control point it requires.

The following octal values are referenced through the symbol IASD.

Subsystem	qp	pp	cp	sm	sb
TELEX/IAF	TXPS	1TD	1	2000	1
EI200	EIPS	1LS	77	1000	1
BATCHIO	BIPS	1IO	76	4000	1
MAGNET	MTPS	1MT	75	0400	1
TAF/TS	TRPS	1TP	2	0200	1
TAF/NAM	TRPS	1SI	2	0004	1
NIP	NMPS	1SI	74	0002	1
RBF	RBPS	1SI	73	0001	1
STIMULA*	STPS	1TS	77	0000	0
Mass Storage*	MSPS	CMS	74	0100	1
CDCS	CDPS	1SI	71	1000	2
MCS	MCPS	1SI	72	2000	2
Deadstart*	DSPS	SET	1	0000	0

When 1DS is called to issue the subsystem initialization jobs through an AUTO. or an individual subsystem DSD command, such as TELEX. or n.RBFffff., it builds an FNT/FST entry using data from this table. The FNT/FST produced as the result of an n.RBFffff. command would have the following format.

---

*Not initiated via AUTO.

59	53	47	41	35	17	11	5 0
1	S (pp)	I	cp	sn	SYOT (jot)	INFT (ft)	0
0	77	ffff			fl	RBPS (qp)	

pp	Controlling routine
cp	Control point required
sn	Job sequence number
jot	Job origin type
ft	File type
ffff	Procedure file sequence number
fl	Field length
qp	Queue priority

Eventually, 1SJ is initiated and if no other jobs are found of a higher priority (that is, other subsystems), it selects this job as the best candidate for scheduling. It then calls its 3SA overlay to schedule this candidate as a special subsystem since its queue priority is greater than LSSS. The FNT/FST entry and the three subsystem control words SSCL, SSCL+1, and SSCL+2 are read. If the byte in the SSCL word for this subsystem is nonzero, then the subsystem is already active and so all interlocks are cleared and the PP is dropped. If the subsystem control byte is zero, then the required control point must be assigned for this job. If the requested control point is occupied by a lower priority job, the job is rolled out so that the control point can be used by the subsystem. If the job at the control point is of a greater priority than the subsystem, the subsystem uses the next available control point.

When the control point becomes available, it is assigned to the subsystem. The control point number is entered in byte 4 of the FNT entry and in the subsystem control word. Protective coding prevents a subsystem from requesting a control point which is not defined in the system. The control point area is then built with all limit values set to unlimited/infinite.

The control statement pointer (CSPW) is set to indicate an EOR on the input file. Default family information is set into PFCW and the family count incremented. The subsystem's queue priority and a CPU priority of MRPS-2 are set in JCIW. The procedure file sequence number is set in CSBW for use by 1SI. The exit mode 7007 is set in the control point exchange package. The scheduler active bit is cleared from JSCL+1, the FNT interlock cleared, and the job name written into this PP's input register with an exit to PPR so that the PP program to initialize the subsystem is loaded.

Once the PP program is loaded into this PP, it initializes the control point field length, and so on, to fit the requirements of the subsystem, set up a control statement stream or procedure file call (for TAF, NIP, RBF), and call 1AJ to process the control statement stream which brings the subsystem into execution.

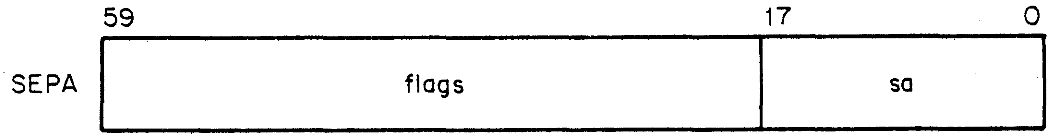
SPECIAL ENTRY POINTS

Many system operations can be performed more efficiently by a CPU routine rather than a PP routine. However, normal CPU routines are restricted by the system from accessing system information. To allow CPU routines to perform restricted system operations, special entry points are used. That is, a CPU routine using special entry points can access restricted system information such as CMR. All special entry points are three characters in length followed by an equal (=) sign.

The special entry points available are the following.

Special Entry Point	Description
ARG=	Suppress arguments processing (RA+2 through RA+63)
DMP=	Dump (save) previous job before load
RFL=	Automatic FL specification for load
MFL=	Minimum FL specification for load
SDM=	Suppress control statement dayfile message
SSM=	Secure system memory
SSJ=	Special system job specifications
VAL=	Define job as a validation processor

A CPU routine with any of the preceding special entry points defined is handled specially by SYSEDIT. That is, SYSEDIT appends an extra word (SEPA) to the CLD entry for this routine. This word is a condensed version of the special entry points defined in the routine and are used by 1AJ when the routine is loaded. The format of SEPA is as follows.



flags Each bit set indicates the following.

Bit	Description
59	Indicates special entry point table entry
58-54	Zero
53	ARG= entry point present
52	DMP= entry point present
51	SDM= entry point present
50	SSJ= entry point present
49	VAL= entry point present
48	SSM= entry point present
47-36	Zero
35	Restart rollin
34	Zero

Bit	Description
33	Suppress DMP= on control statement call
32	Only create DM* with nothing on it
31	Dump FNT entries, control point area and field length, to file DM*
30	Create file DM* as an unlocked file
29-18	0, for dump of full FL; nonzero for dump of FL* 100B of FL

sa	SSJ= parameter block address

All normal ABS entry point names in the CLD will have bit 59 of SEPA equal to 0.

Routine 1AJ detects the SEPA word and processes the load accordingly. System routines that are called via special entry points include CHKPT, CPMEM, and RESEX. These routines can be called from a PP or via an RA+1 request summarized as follows.

RA+1 Request	PP Request Processor	CPU Request Processor	Description
CKP	SFP	CHKPT	Checkpoint request
DMP	SFP	CPMEM	Dump FL
REQ	SFP	RESEX	REQUEST macro call
LFM/PFM	LFM/PFM	RESEX	Tape/pack request

These CPU routines can be called by an RA+1 request or by another PP routine. (When RA+1 is used to make the call, autorecall is designated.) The routine names CKP, DMP, and REQ must not be in the PP library since these calls are processed by SFP. In order for the PP request processor (SFP, LFM, or PFM) to call the CPU routine, the entry point name (which is the same as the RA+1 request) is placed in SPCW in the control point area. The PP request processor can perform the following.

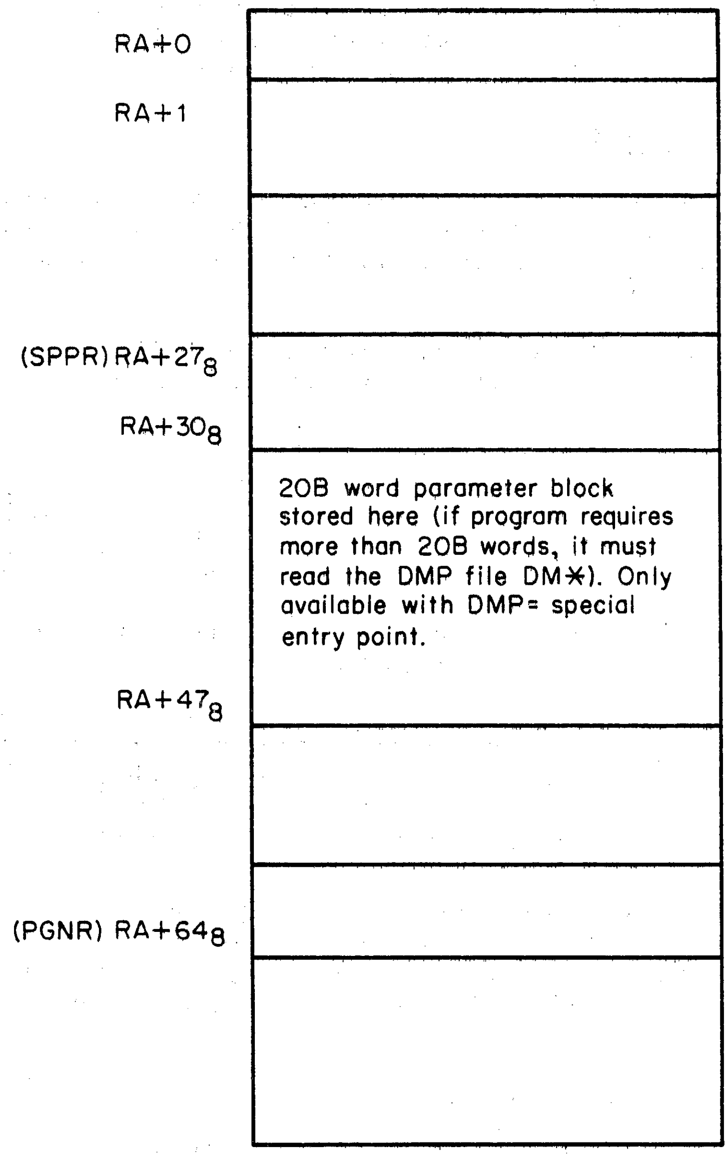
- Set any completion or status bits in the requesting jobs FL.
- Set bits 38, 39, and 40 of SPCW as desired.
- Write its own PP input register image in RA+1 so that this PP routine is called upon completion of the CPU routine.
- Set rollout flag (ROCM function).

Routine 1AJ picks up SPCW and loads the appropriate CPU routine for the specified entry point name. The upper six bits of SPCW are used as an interlock to prevent more than one call at a time from being processed. This means that one routine using special entry points cannot call another such routine. The upper six bits of SPCW are equal to 77B if such a routine is active. The CPU request processor contains entry points for the system function desired. For instance, RESEX has entry point names REQ, LFM, and PFM. When the PP request processor has completed setting up SPCW, it drops and 1AJ continues the processing. Routine 1AJ rolls out the calling CPU routine (filename is DM*) if a DMP= entry point exists for the CPU request processor routine to be loaded. If a parameter block address has been specified in the SPCW word, 1AJ picks up the parameter list and stores it in RA+30B through RA+47B. The SPCW word is stored in location RA+27B (defined by symbol SPPR), as shown in figure 5-11. (This is available only if DMP= has been specified.) Later, this parameter list will be available to the CPU request processor. Now the CPU routine is loaded and processing begins at the appropriate entry point. Prior to normal termination, the CP request processor can set a return status in RA+27B (SPPR). This status is later stored in bits 35 through 24 of SPCW by 1RI.

When 1AJ detects that the CPU request processor has completed, it calls 1RI to perform the following.

- Store the return status in SPCW.
- Retrieve the parameter block from RA+30B through RA+47B.
- Reload the control area and job's FL from the DM* file, if it exists.
- Store the updated parameter block back into the job's FL.
- Clear SPCW word.

Routine 1AJ now restarts the original calling program where it left off.



20B word parameter block stored here (if program requires more than 20B words, it must read the DMP file DM*). Only available with DMP= special entry point.

- Contents of SPPR stored here when loading CPU processor. CPU processor may put a status in byte 2 for return to the PP calling program.

- Cleared to indicate call was initiated by a PP-request processor (otherwise nonzero indicates normal control statement initiation).

Figure 5-11. Field Length of Loaded CPU Request Processor

ARG= Special Entry Point

ARG= is used by a job wishing to do its own control statement argument processing. If present, arguments are not passed to RA+2, but the entire control statement image, including statement label and other options ($,/), is placed in RA+70.

DMP= Special Entry Point

A program using the DMP= entry point should set up bits 35 through 18 in SEPA with a PP routine (in the case of the control statement or macro DMP it is done automatically) as previously described.

The DM* file is the rollout file. The only difference is in the FNT. If it were a rollout file, then the FNT would be as follows.

59	17	11	5 0
job name	job org	type ROFT	cp =0

However, as a DM* file the FNT would be as follows and the file remains attached to this control point.

59	17	11	5 0
DM*	job org	type LOFT	cp no.

DM* is not a legal file name and a CPU user cannot create a file whose name contains special characters. However, a CPU routine may read or write such a file if it already exists. Hence, 1RO must be asked to create the DM* file if a special entry point job needs to use the file.

The flow of a DMP= request is as follows.

- 1AJ finds this control point idle. That is, W = X = R = 0 or DIS calls 1AJ directly.
- 1AJ calls 1RO, which creates a rollout file as specified in bits 35 through 18 of SEPA. The file will be named DM* and left attached to the control point as a local file.
- 1AJ then loads the CPU program containing the entry point name specified in SPCW.
- The CPU processor completes normally (END or ABT).
- 1AJ is called to advance the job; it detects that a DMP= has just completed and calls 1RI to restore the control point FL and control point area from the DM* file.
- 1AJ advances the job or restarts the previous job.

Figures 5-12 through 5-14 illustrate the DMP= processing while figures 5-15 through 5-21 illustrate the flow charts for this procedure, using DMP as a example.

RFL= Special Entry Point

When a program with RFL= is loaded from the system, the program's field length is set to the value of RFL= (rounded to the next higher 100B).

MFL= Special Entry Point

Same as RFL= except nothing is changed if the RFL (as set by the last RFL control statement or by the last SETRFL macro call) is greater than the MFL= value (if present RFL> MFL=, then use present RFL value).

SDM= Special Entry Point

For programs with SDM= entry points, no dayfile message is generated on the control statement call. The program should issue its own messages. Using ACCFAM as an example, the password on a USER statement should not appear in the dayfile. When USER,ABCUSER,PASSWRD. is issued, ACCFAM using an SDM= entry point can strip off the password and issue USER, ABCUSER,. to the dayfile.

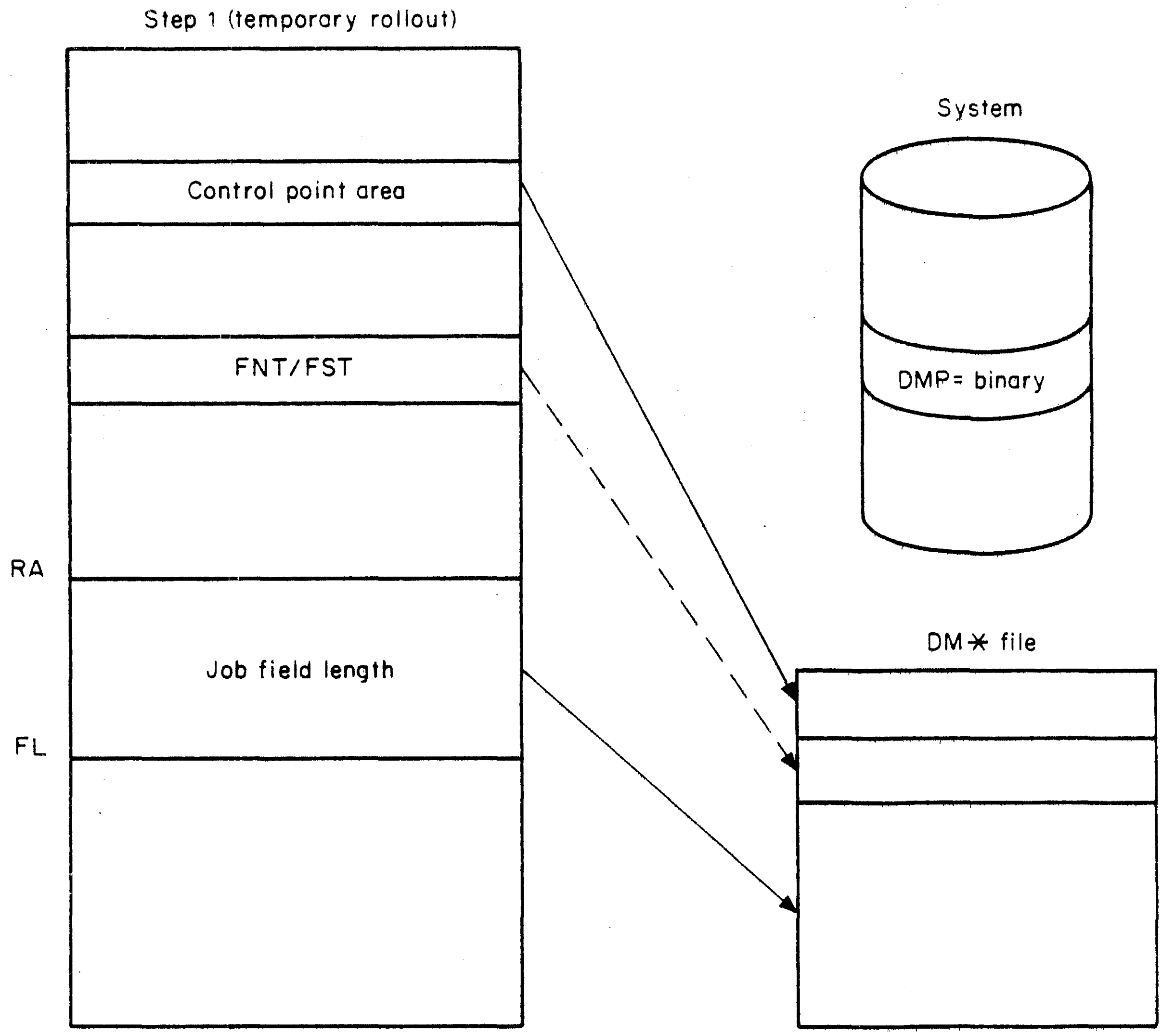


Figure 5-12. DMP= Processing (1AJ Calls 1RO)

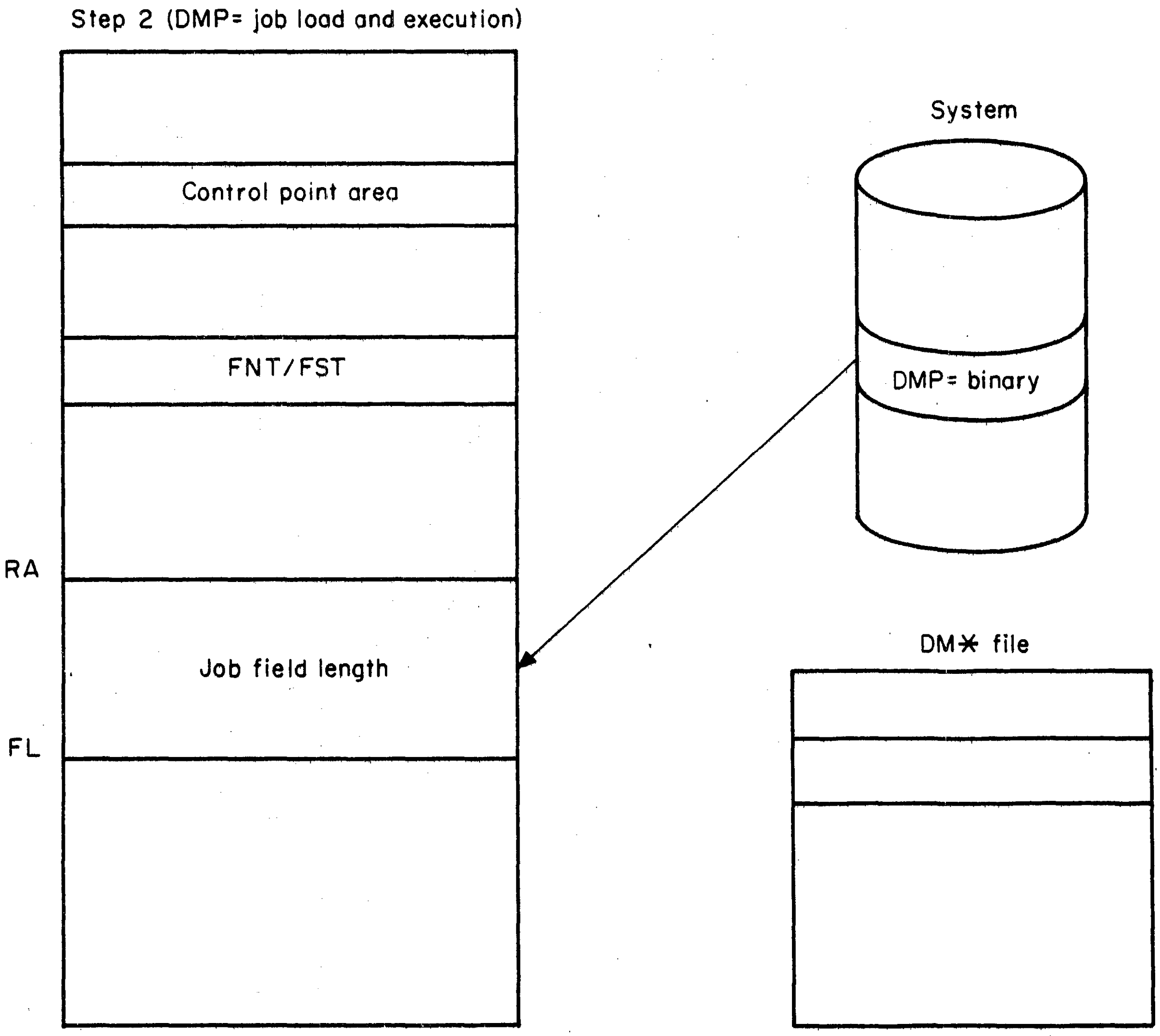


Figure 5-13. 1AJ Calls LDR to Load DMP= Program

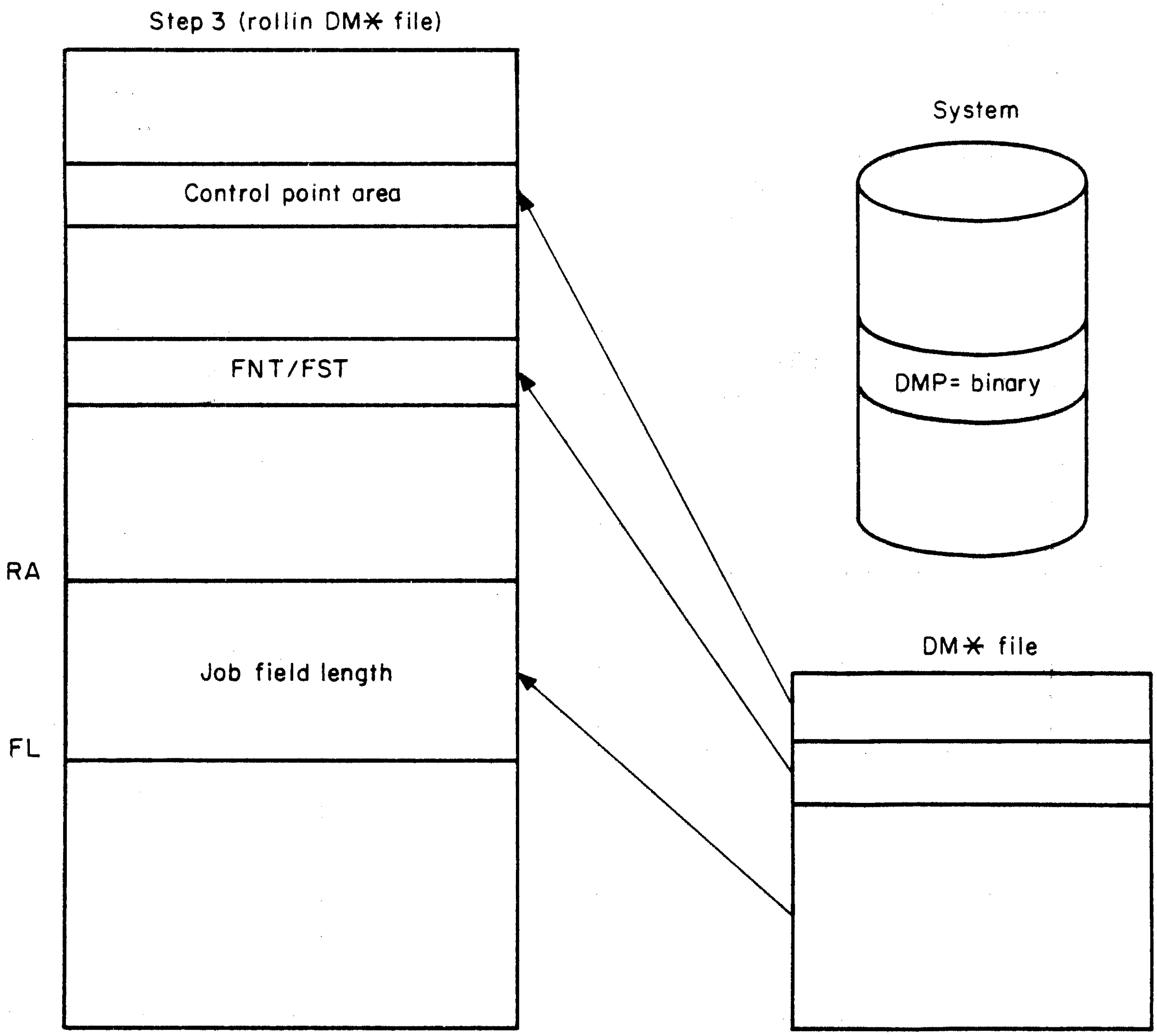


Figure 5-14. 1AJ Calls 1RI to Restore the Job

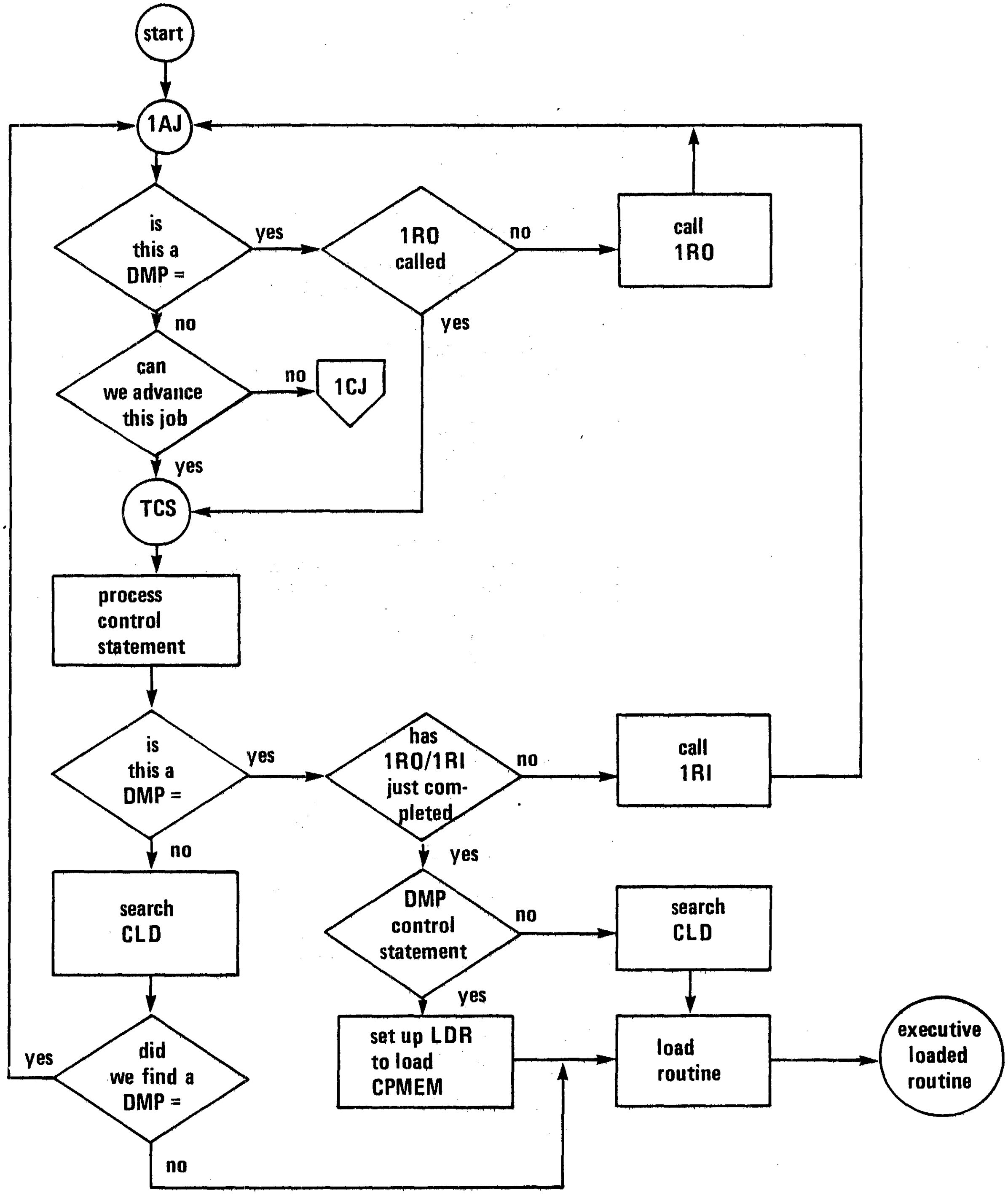


Figure 5-15. General Flow

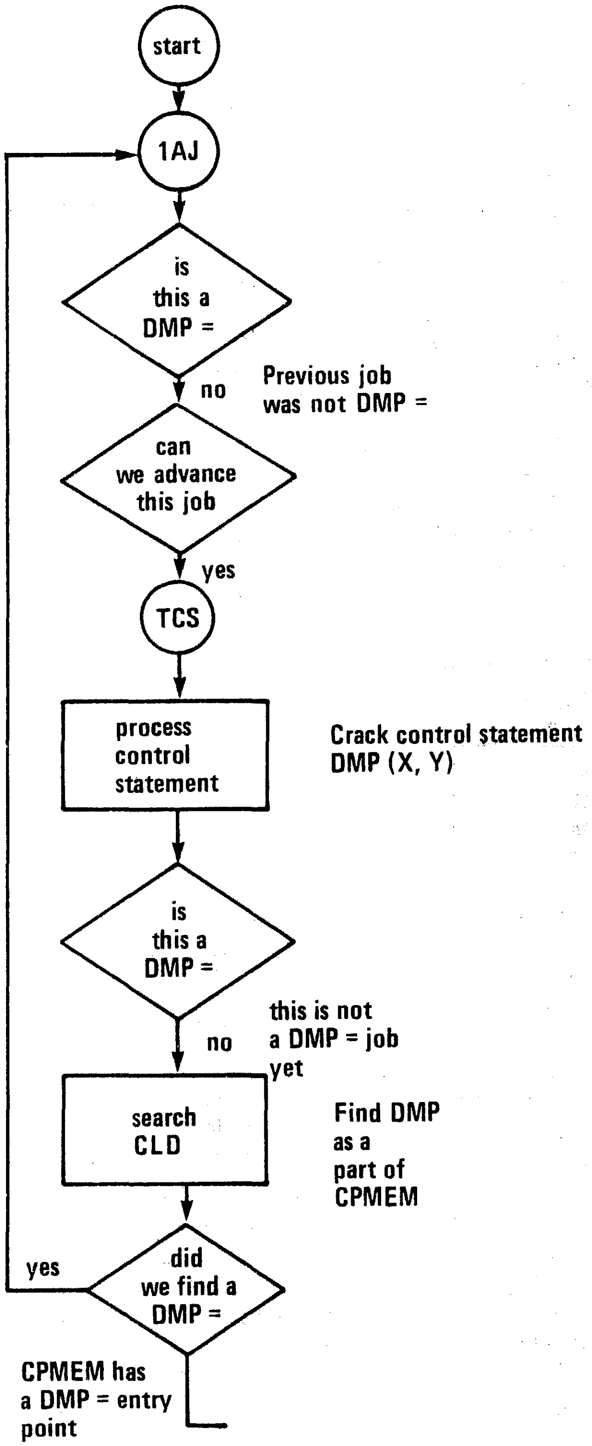


Figure 5-16. Pass 1 (Job Flow Has Come to a DMP Control Statement)

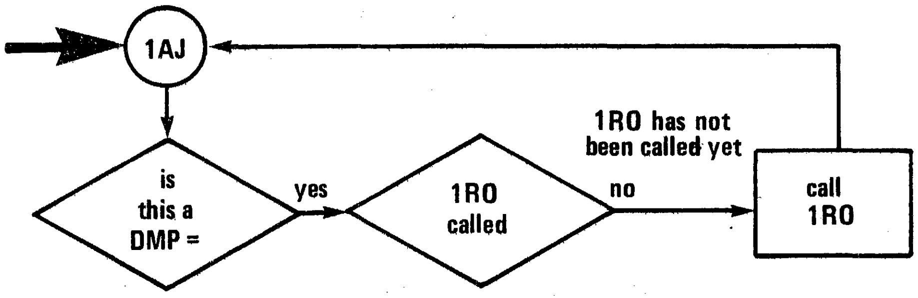


Figure 5-17. Pass 2

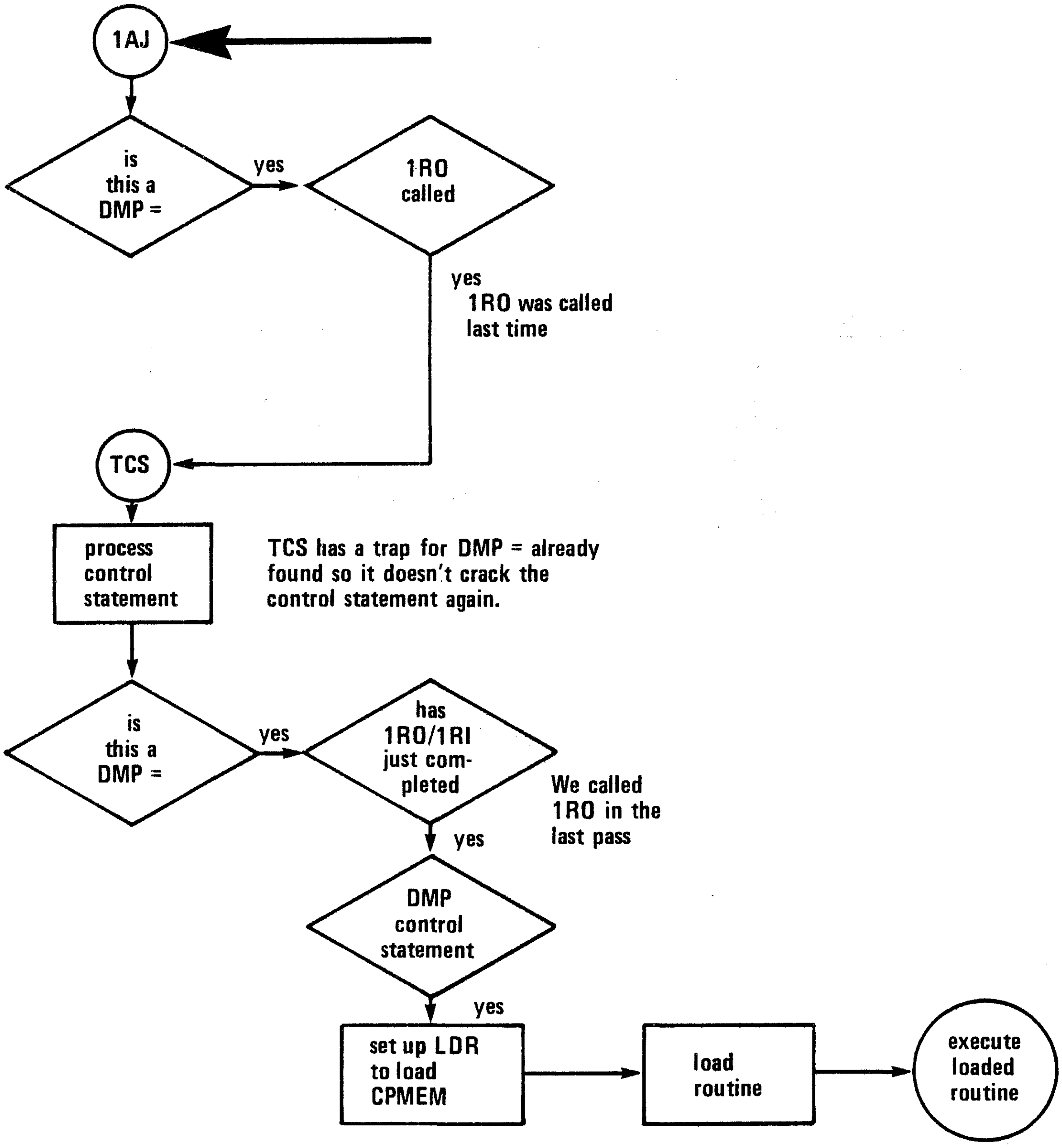


Figure 5-18. Pass 3

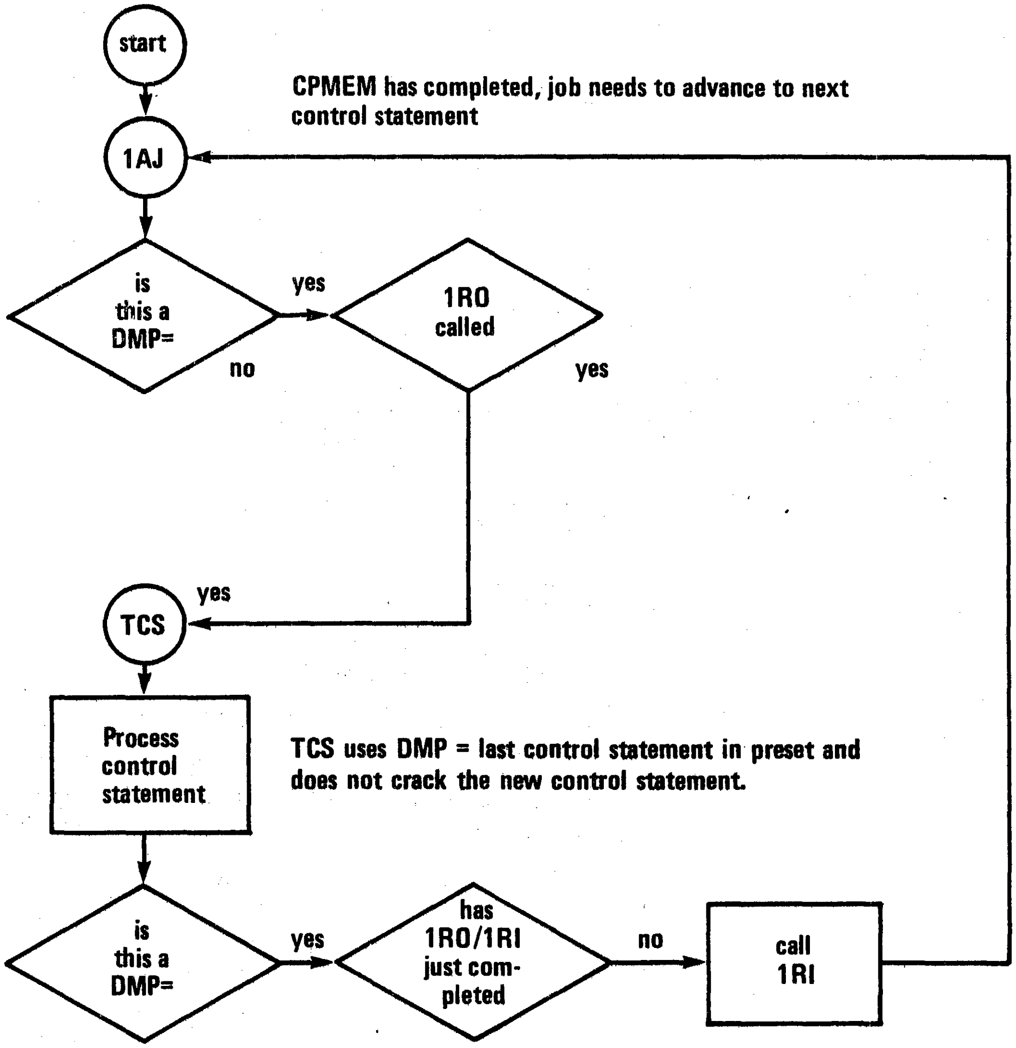
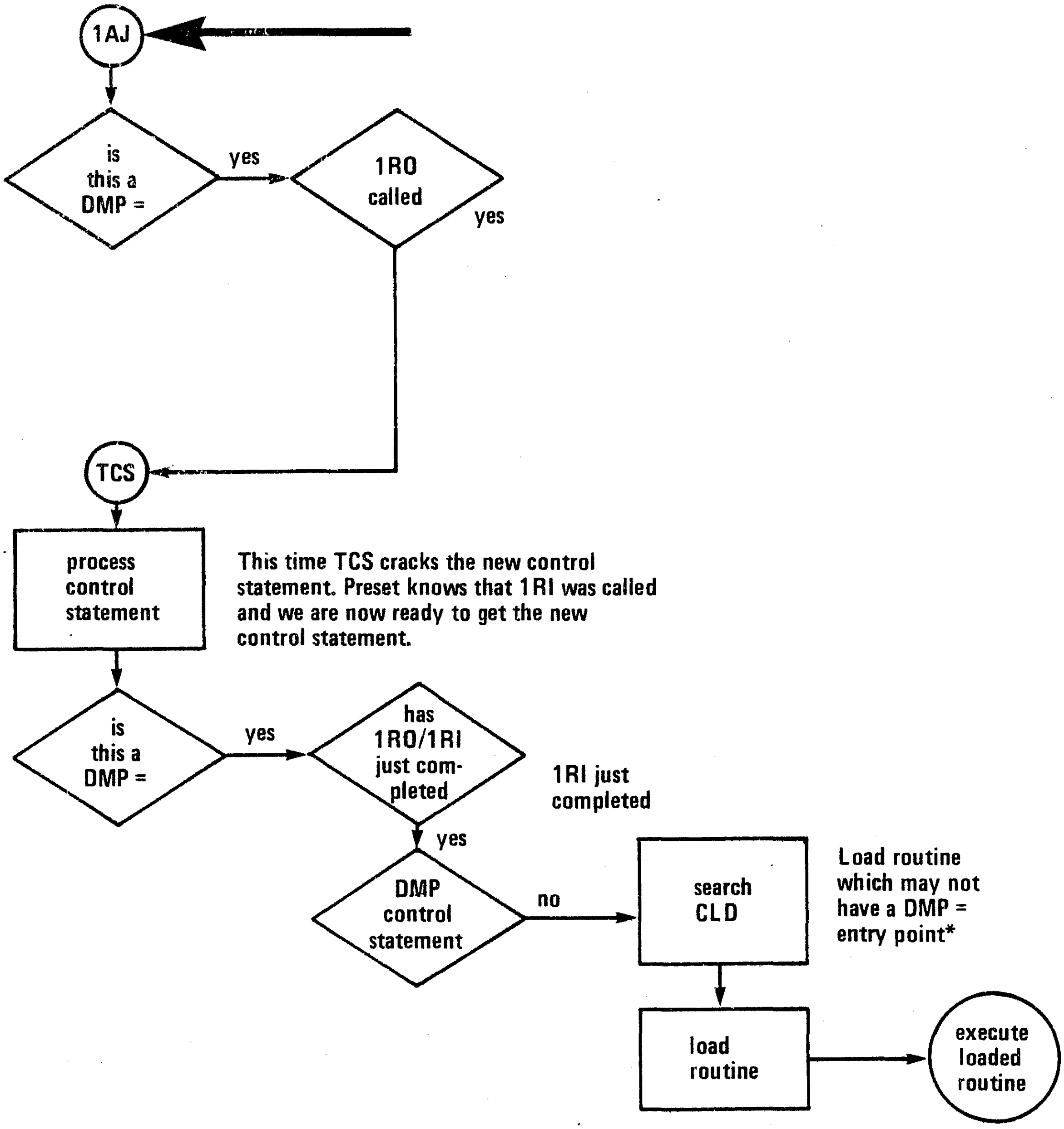


Figure 5-19. Pass 4



---

*1 A special entry point job cannot initiate another special entry point job

Figure 5-20. Pass 5

SSJ= Special Entry Point

Programs with SSJ= entry points are defined as special system jobs. The address specified by the SSJ= entry point, determines the start of a parameter area where the user accounting control words from the control point area are temporarily stored to allow the special system job access beyond the user's validation. When the special system job completes (or aborts) the user's validation parameters are retrieved from the parameter areas within the special system job's field length and restored to the control point area. All local files created by the special system job (ID=SSID=74) are returned before normal control statement processing is resumed. Whenever an SSJ= job creates a file, the FST ID field is set to SSID (74B). In this way, 1AJ can ensure that any files attached to this control point during SSJ= processing are released prior to returning control to the normal user.

The common deck COMSSSJ is provided to supply the calling program with special system job parameter equivalences.

An RFL= entry point must precede the SSJ= entry point to allow SYSEDIT to verify that the parameter area fits within the special system jobs field length. If this condition is not satisfied, the SSJ= entry point is considered a normal entry point for the program and no special processing will be done for it. The only acceptable order is:

```
ENTRY RFL=
ENTRY SSJ=
```

The first word of the parameter area (SPPS) is used to set the control point area values. If it is zero, the current values are retained. Limits for these values are:

$0 \le$ CPU priority $\le$ 70B
$0 \le$ queue priority $\le$ MXPS+1
$0 \le$ time limit $\le$ 77777B

Any other values are ignored. Thus, it can be ensured that a task does not get a time limit error, that a task has a higher CPU priority than a normal job, and so on. Values are reset when the task terminates.

The SSJ= parameter block format is as follows.

	59	47	23	17	11	0
SPPS	0	time limit	CPU priority		queue priority	
UIDS	user number			user index		
ALMS	exact copy of control point area word ALMW					
ACLS	exact copy of control point area word ACLW					
AACS	exact copy of control point area word AACW					

The entire SSJ= block is swapped with the control point area values unless word 0 is zero. If word 0 is zero, then just store the user's control point area in the 5-word block. In any case, when the SSJ= completes, the 5-word block is restored into the user's control point area. Thus the SSJ= program can and does place any values it sets in this block into the control point area.

That is the way that ACCFAM sets up the user verification area in the control point area, and the way that CHARGE clears the VAL= flag (bit 17) in UIDW. Also, the swap allows the SSJ= program to specify UI = 377777B for accessing validation, accounting, and resource files. If the SSJ= user defines SSJ= as 0, then the swap does not occur, and all files created by the SSJ= user do not get ID = 74B. The files remain for the caller, but the job gets SSJ= privileges (SIC, RSB, and so on).

VAL= Special Entry Point

When validation is enabled, the system aborts any job of nonsystem (SYOT) origin which attempts to load and execute as the first control statement, any routine which does not have a VAL= entry point. This is the method employed to check validation. The first two or three statements of a job stream must be job, USER, and CHARGE (if needed). USER causes the loading of ACCFAM, and CHARGE causes the loading of CHARGE, both of which contain VAL= entry points. The system allows these routines to run, and assuming that they do not abort the job, they enter this job stream into the system. Once they are done, the VAL= system checking is no longer done for this job. If a user did not have a USER statement as the second statement, it forces a load of a routine without a VAL= entry point, and the job is aborted by the system.

SSM= Special Entry Point

The SSM= entry point causes the secure system memory status to be set in the control point area. The setting of the secure system memory bit (bit 59 in DBAW) prevents the dumping of any portion of the job's field length.

SPECIAL RA+1 REQUESTS

The following RA+1 requests can be used only by a subsystem.

- SIC
- RSB
- SPC

SIC and RSB can also be used by SSJ= or queue priority greater than MXPS type jobs. SPC is used to call special PP routines. SIC and RSB are used for intercontrol point communications.

Special PP Calls

A normal CPU routine may request only PP routines whose name begins with a letter. This is a protective feature to keep normal jobs from accessing certain system PP routines. By convention, any PP routine which should be available to a user, and is coded in such a way as to keep from destroying the system if called by an improper request, has a letter as the first character of its name. Other restricted PP routines have a number as the first character their names.

The SPC request allows a CPU routine to call a special PP routine (such as IAF or TELEX calling 1TA). The SPC request is as follows.

	59	41	17	0
RA+1	SPC	0	addr	

addr	First word address of a list of names of the PP routines desired and their arguments. The list is terminated by a zero word.

In a SPC request, the following conditions apply.

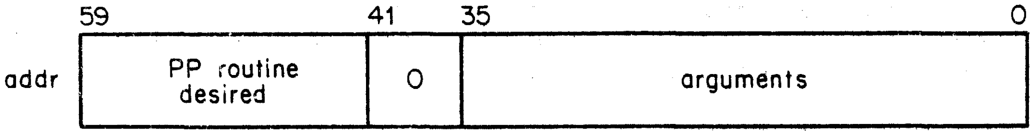
- Autorecall is not honored.

- If the addr word is cleared, the request has been honored and the PP routine started.

- If the addr word is unchanged when the CPU regains control, the PP routine was not started (possible PP saturation, for example.

- The call is honored only for jobs whose queue priority is greater than MXPS. All other job steps are aborted.

The format of location addr is as follows.



Intercontrol Point Communication

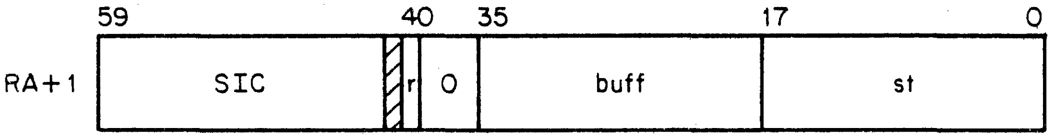
The control point concept allows each control point to run independently of any other control points in the system. In addition each control point is protected from any other control point destroying any part of its field length. In some cases, however, it is necessary for one control point to communicate with another, as in TELEX to TAF/TS, and RESEX to MAGNET.

A subsystem or any program with SSJ= or a queue priority greater than MXPS wishing to communicate with some other control point (maybe another subsystem) by sending information, can set up a communication block using ICAW in the control point area and transfer it to a designated control point. Also, it may receive a block of data from some other control point (which may also be another subsystem).

The control of the transfer is based on the subsystem's queue priority (which is why they must be unique). The buffers are defined in ICAW. The SIC and RSB RA+1 requests are used for this communication.

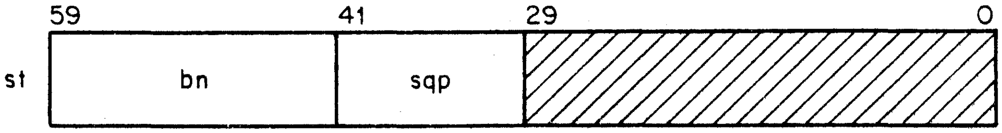
SIC Request

The SIC request is used to send an intercontrol point data block from a control point program to the specified subsystem. The format of the request is as follows.



r	1 if autorecall is desired (bit 40)
buff	First word address of the buffer to be transferred to the subsystem
st	Address of status word for the transfer

The format of location st is as follows.



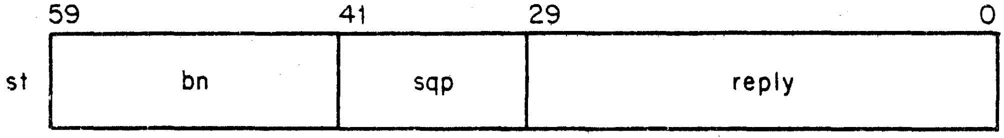
bn	Buffer number of subsystem to transfer to
sqp	Destination subsystem queue priority

A block starting at buff will be moved to the indicated subsystem. The block length is specified in bits 17 through 0 of the first word of the block (buff), which includes this header. The block length must be less than 101B (to force CPUMTR in MTR mode; this operation must be very fast).

NOTE

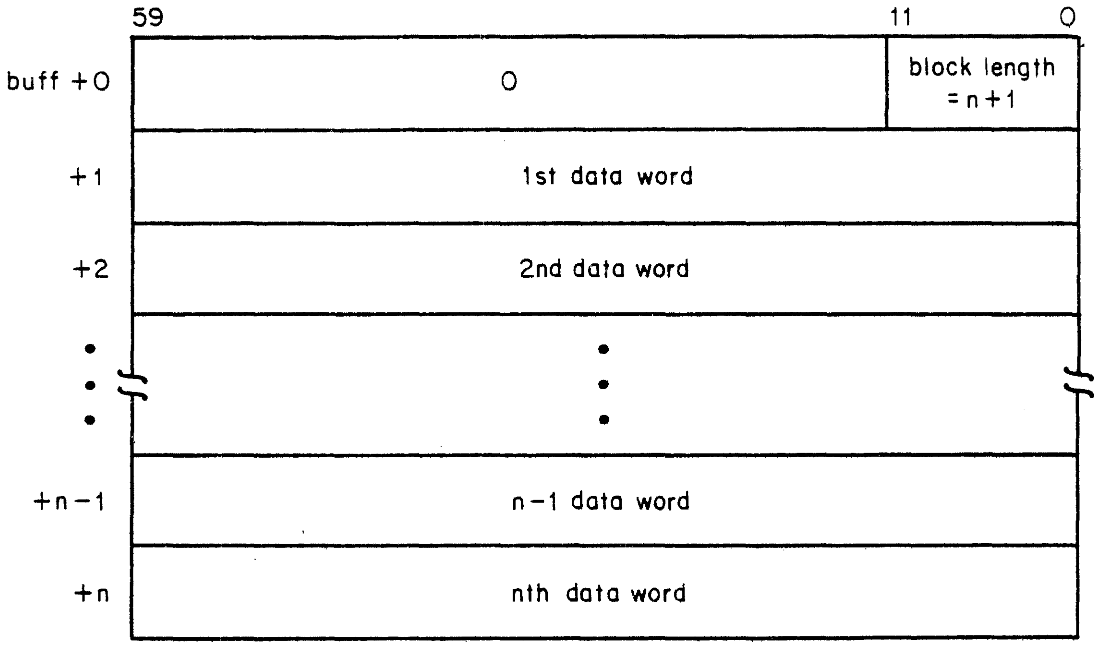
> The request is honored only from jobs with queue priority greater than or equal to MXPS (subsystem status), or an SSJ= entry point defined, or with access bit CSTP (user may access special transaction functions) turned on. If these conditions are not met, the call is treated as a call for a PP routine.

After the request is processed, location st has the following format.



bn	Unchanged	
sqp	Unchanged	
reply	1	If transfer completed successfully
	3	If designation subsystem is not present in the system
	5*	If subsystem buffer is full, subsystem being moved, or subsystem job is advancing
	7	If block length as specified in the first word is larger than that permitted by the subsystem
	11	If destination buffer is undefined by the subsystem

The format of the buffer block to be transferred is as follows.



NOTE

n is less than or equal to 100B so entire block length is 101B.

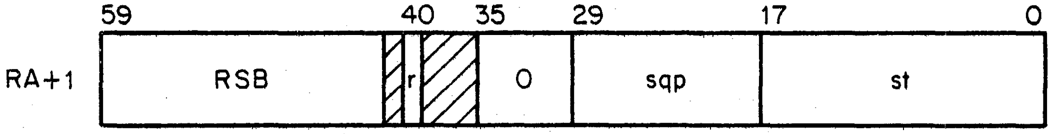
---

*If autorecall is specified, the control point remains in recall until condition 5 ends. The subsystem may indicate whether its buffer is full by setting the first word in the buffer nonzero. That is, if the first word of the buffer in the subsystem is nonzero it cannot receive data; if it is zero, it is ready to receive data.

RSB Request

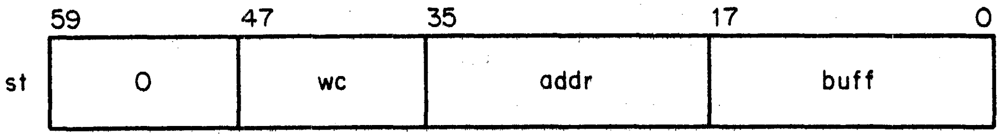
The RSB request is used to send an intercontrol point block from a subsystem to the calling control point; if no subsystem is specified, from absolute CM. The calling routine must have an SSJ= entry point defined.

The format for this call is as follows.



r	1 if autorecall desired (bit 40)
sqp	Subsystem queue priority (or control point to read). If zero, then block is read from absolute memory or relative to caller's control point area.
st	Address of status for the read.

The format of location st is as follows.



wc	Number of words to read.
addr	Address to read from CM or buffer address relative to the subsystem.
buff	Address of buffer to receive data in this control point's field length. When sqp = 0, the contents of buff determines whether the read is from absolute CM or relative to the caller's control point area.

If buff is less than 0, the read is from absolute CM and addr in the st word is the absolute address in CM to begin the read.

If buff is zero or greater, the read is relative to the caller's control point area, and buff contains a list of addresses located within the control point area which are to be read. The list ends at wc or a zero list entry. The contents of the control point area address read is stored in the buff location which contains that address.

Location buff is a flag denoting a read from absolute memory or relative to the control point area in the case where sqp is 0. The calling program must have an SSJ= entry point.

After the request is processed, the format of location st is as follows.

	59	47	35	17 0
st	reply	wc	addr	buff

reply	4000B Transfer completed successfully 2000B Subsystem not present
wc	Unchanged
addr	Unchanged
buff	Unchanged

If sqp is nonzero, the buffer is filled. If sqp is zero and buff is less than zero, buff is filled from absolute memory as specified in the addr field. If sqp is zero, and buff is greater than or equal to zero (control point area read), then an example of this format is as follows.

	59 0
buff+0	+1
+1	STSW
+2	STSW-17B
• • •	• • •
+WC-2	MS1W
+WC-1	APJW

In the preceding example, buff+1 contains the job status word from the control point area; buff+2 contains the second word of the exchange package area (from the exchange package area); buff+wc-2 is the first message buffer area; and buff+wc-1 is the program number area.

NOTE

> The buffer's length is wc words. It is not possible to get the first word of the exchange package area since the address would be 0 relative to the control point area and any 0 word ends the list. It would be necessary to know the absolute address of the control point area to get the first word of the control point area.

The above is an example and is not intended to imply that only the control point area shown can be read.

System job flow is controlled by routines 1SJ, 1SP, 1AJ, 1CJ, 1RO, and 1RI.

JOB SCHEDULER - 1SJ

The job scheduler (1SJ) scans the FNT/FST entries looking for files of type input (INFT) or type rollout (ROFT). It builds tables which it uses to determine which of the jobs in the input or rollout queue based on priority are to be assigned to a control point and started (Table 6-1). Routine 1SJ rolls out any jobs which have a lower priority and attempts to start the best job. If 1SJ cannot find a best job to start or cannot get enough resources for the best job, it drops.

The next time 1SJ is called, the best job may not be the same one picked the last time. A best job is only guaranteed a startup if the resources necessary are available at the time the job is being prepared.

Routine 1SJ works with the current system status. Whenever many jobs make changes, these changes affect 1SJ only while it is executing. The JSCL and JSCL+1 words ensure that only one 1SJ can run at any time in the system. The scheduler cycles itself until no jobs remain to be scheduled or a certain mass storage activity threshold is reached. This ensures that the system is not constantly scheduling jobs in and out and thereby wasting computer resources.

The scheduler selects the candidate by using the subroutine Search For Job (SFJ). The selection is done on the following basis.

1. The highest priority job that will fit in unassigned or rolling memory within the service constraints FL (maximum individual job field length), FLE (maximum ECS field length), and AM (maximum memory allowed) for the candidate's job origin type.

2. If candidates of equal priority are found, the job selected is the one residing on the mass storage device with the least amount of activity. The amount of disk activity is determined by the following factors: channel busy; channel being requested; and unit reservation.

3. If the mass storage activity is also equal, the job with the largest field length is selected.

4. If no job is selected, but one was rejected due to service constraints, it may be scheduled if no jobs have to be rolled out. If this is done, the job's priority will be set to its origin type's lower bound. This prevents resources from sitting idle during periods of low activity.

TABLE 6-1. 1SJ TABLES

Location	Description	Bits	Description
TACP	Active control points. One-word entry terminated by zero entry. Sorted in descending priority.	11	Rollout in process
		10	Rollout requested (used in subroutine CFL only)
		9-5	Zero
		4-0	Control point number
TRST	Table of rollout status. One-word entry indexed by control point number.	11	Rollout in process
		10	Rollout requested (used in CFL only)
		9-0	Zero
TJFL	Job field length. One-word entry indexed by control point number.	11-0	Field length assigned at control point
TJEC	Job ECS field length. One-word entry indexed by control point number.	11-0	ECS field length assigned at control point
TJJP	Job priority. One-word entry indexed by control point number.	11-0	Priority of job
TJOT	Job origin type. One-word entry indexed by control point number. Set only if job active.	11-0	Origin type of job
TMFO	Table of total available field length for all jobs of an origin type. One-word entry indexed by origin type.	11-0	Field length available
TMEO	Table of total ECS available field length for all jobs of an origin type. One-word entry indexed by origin type.	11-0	ECS field length available

TABLE 6-1. 1SJ TABLES (CONTINUED)

Location	Description	Bits	Description
TAFO	Table of assigned field length by origin type. One-word entry indexed by origin type.	11-0	Field length assigned.
TAEO	Table of assigned ECS field length by origin type. One-word entry indexed by origin type.	11-0	ECS field length assigned
TMJO	Table of maximum field length per job by origin type. One-word entry indexed by origin type.	11-0	Maximum FL allowable for a job
TMXO	Table of maximum ECS field length per job by origin type. One-word entry indexed by origin type.	11-0	Maximum ECS field length allowable for a job

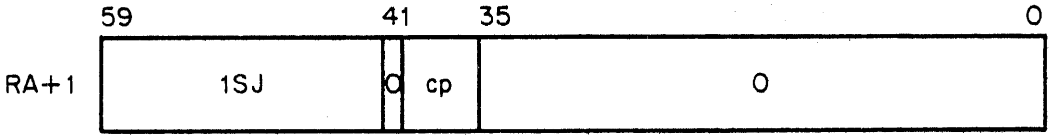
TABLE 6-1. 1SJ TABLES (CONTINUED)

Location	Description	Bits	Description
DACT	Device activity count table. One-word entry indexed by equipment number.	11-0	Device activity as found in byte 0 of MST word DALL

The scheduler is requested periodically or on a demand basis through the RSJM monitor function (refer to section 3). CPUMTR determines if the scheduler is active (bit 59 set in JSCL+1) and if so, takes no action. If the scheduler is not active, 1SJ is called unless the scheduling delay in JSCL has expired. In this case 1SP is called. Routine 1SP calls 1SJ into its PP when it has finished its tasks.

The RSJM function is issued when jobs are placed into the input or rollout queues (by QFM, 1RD, or 1TA), when a job is started (by 1AJ), and by certain routines when it is desirable to begin scheduling activities after they have completed (1CK, 1DS, 1MB, 1SP, and 3SA).

The call to 1SJ has the following format:



cp   Control point number

A flowchart of the main loop of 1SJ (SCJ), is shown in figure 6-1. The main subroutines of 1SJ are described in the following paragraphs.

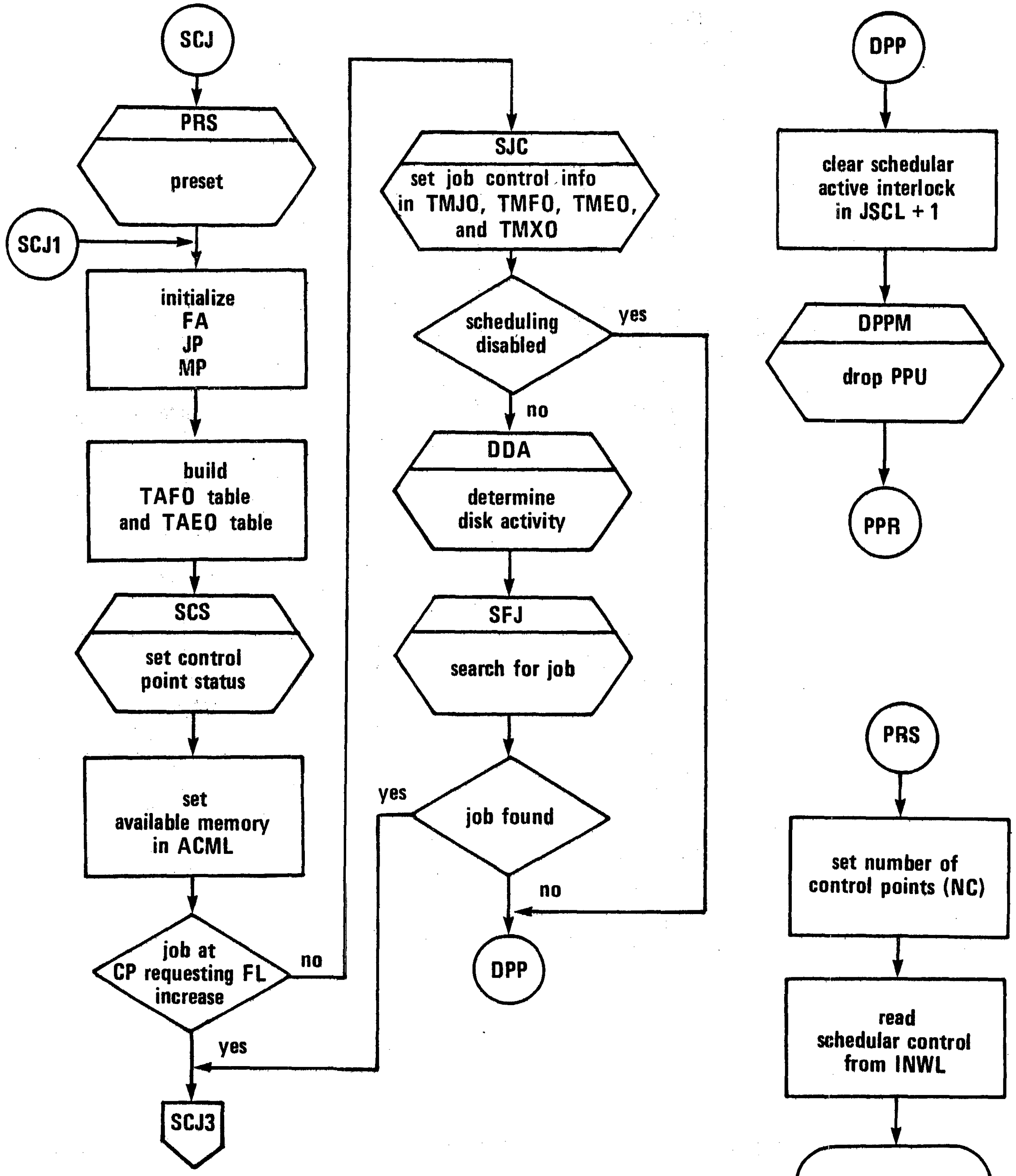


Figure 6-1. 1SJ Main Loop SCJ

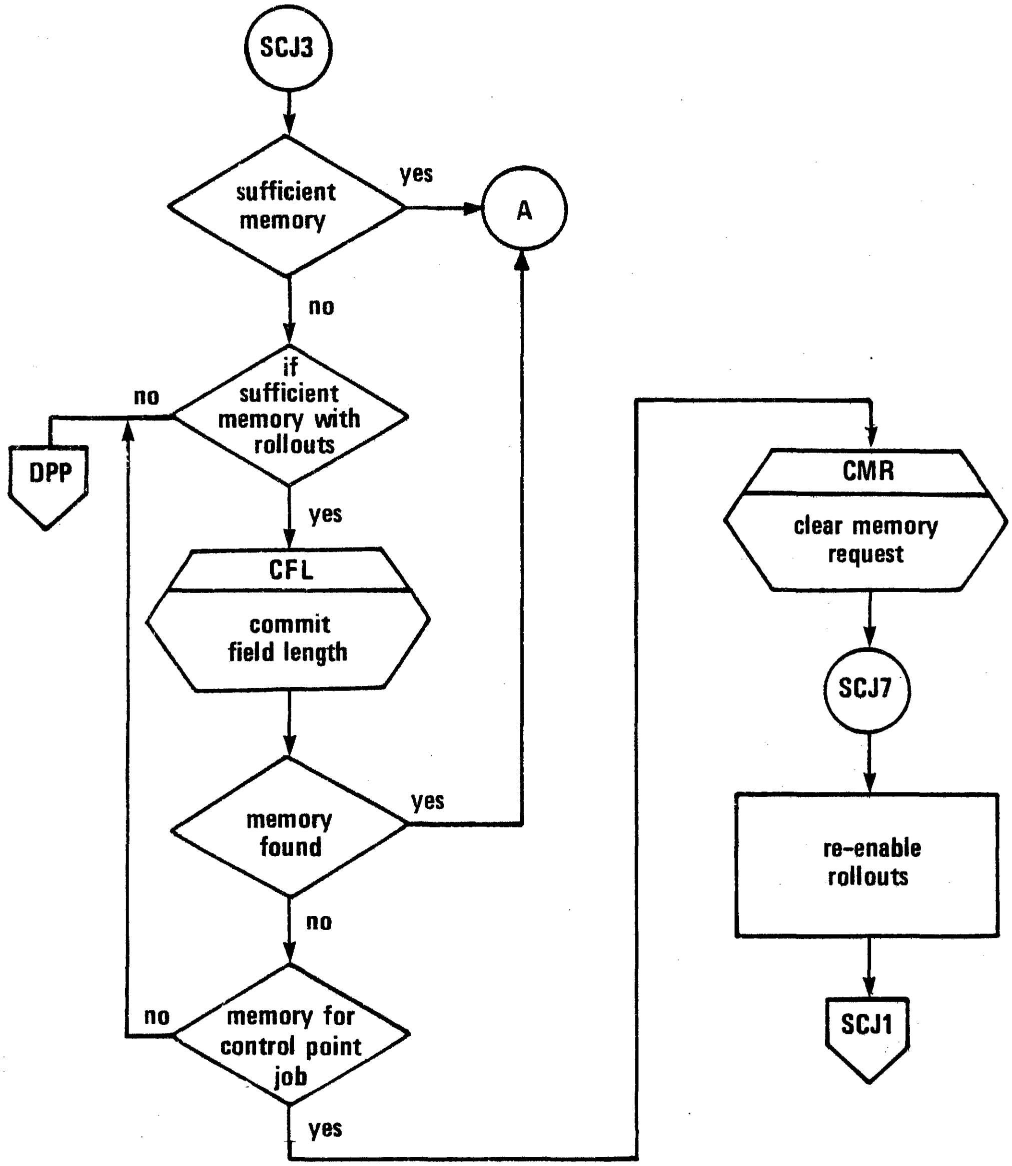
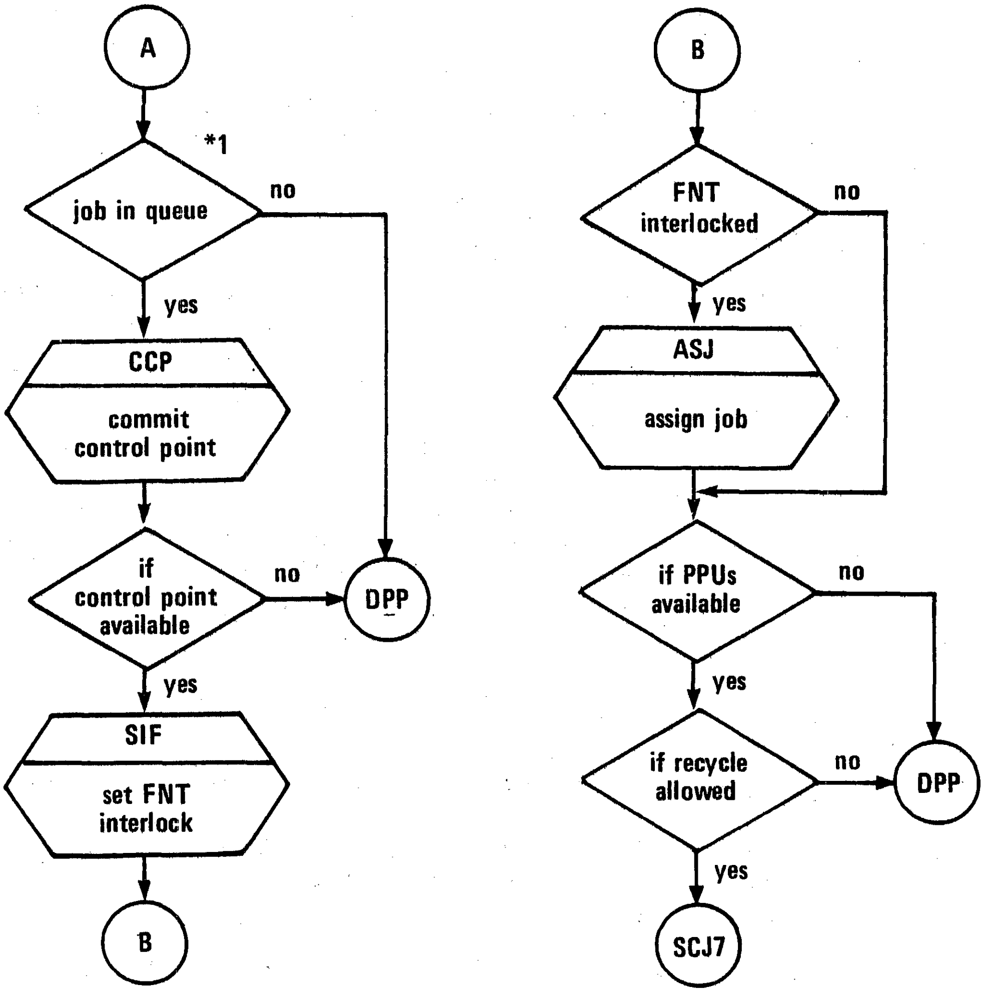


Figure 6-1. 1SJ Main Loop SCJ (Continued)



*1 The job in queue condition tells 1SJ if it is trying to schedule a job to a control point or attempting to increase a running job's field length

Figure 6-1. 1SJ Main Loop SCJ (Continued)

SET CONTROL POINT STATUS (SCS)

SCS builds the TACP, TJFL, TJEC, TRST, TJOT, TJPR, TAFO, and TAEO tables from information contained in the control point area. It initializes direct cells AC (available control points), AM (available CM), AE (available ECS), RM (rollout CM), RE (rollout ECS), JC (control point with field length request), JF or JE (amount of CM or ECS JC requires), and JP (queue priority of JC).

SET JOB CONTROL (SJC)

SJC builds the TMJO TMXO, TMEC, and TMFO tables from the job control area.

DETERMINE DISK ACTIVITY (DDA)

DDA builds the DACT table. DACT is the device activity count as found in byte 0 of MST word DALL.

SEARCH FOR JOB (SFJ)

SFJ chooses the best candidate for scheduling. If on the first pass in SFJ no candidate was selected and if a job had been rejected because of service constraints, the TMJO, TMFO, TMXO and TMEC tables are set with unlimited values, rollout disallowed, and a second pass through SFJ made. SFJ is flowcharted as figure 6-2.

COMMIT FIELD LENGTH (CFL)

CFL selects which jobs need to be rolled out in order to obtain the required amount of field length. All jobs necessary to be rolled will have a ROCM set for their control point. Jobs of the same origin type will be rolled before jobs of different job origins, if possible.

COMMIT CONTROL POINT (CCP)

CCP selects the control point for the job. If no control points are available and none are currently being rolled, a control point with a lower priority is selected to be rolled out and a ROCM isued on that control point. If control points are available, the control point selected is determined as follows (consider the control point's field length to include the field length of all unoccupied control points following it).

1. Exact fit
2. Smallest hole that is larger than needed
3. Largest hole if none is big enough

This selection process minimizes the amount of storage movement necessary to give the control point the required field length.

ASSIGN JOB (ASJ)

ASJ requests the storage for the job, initializes the JNMW and TFSW control point words, sets queue priority and time slices in JCIW and TSCW, and calls 1AJ or 1RI to process the job. Routine 1AJ is called if the job is scheduled from the input queue and 1RI is called if the jos is scheduled from the rollout queue. If a PP is available, a RPPM call is made for 1AJ or 1RI. If a PP is not available or one is not assigned, the scheduler active bit (bit 59 in JSCL+1) is cleared and this PP is used for the 1AJ and 1RI processing.

SCHEDULE SPECIAL SUBSYSTEM (SSS)

SSS is contained in overlay 3SA and is used to schedule jobs whose queue priority is larger than LSSS. The FNT/FST entry and the three subsystem control words SSCL, SSCL+1, and SSCL+2 are read. If the byte in the SSCL word for this subsystem is nonzero, then the subsystem is already active and all interlocks will be cleared and the PP dropped. If the subsystem control byte is zero, then the required control point must be assigned for this job. If the requested control point is occupied by a lower priority job, the job will be rolled so that the control point can be used by the subsystem. If the job at the control point is of larger priority than the subsystem, the subsystem will use the next available control point. When the control point becomes available, it is assigned to the subsystem. The control point number is entered in byte 4 of the FNT entry and in the subsystem control word. Protective coding prevents a subsystem from requesting a control point which is not defined in the system. The control point area is then built with all limit values set to unlimited or infinite. The control statement pointer (CSPW) is set to indicate an EOR on the input file. Default family information is set into PFCW and the family count incremented. The subsystem's queue priority and a CPU priority of MRPS-2 are set in JCIW. The procedure file sequence number is set in CSBW for use by 1SI. The exit mode 7007 is set in the control point exchange package. The scheduler active bit is cleared from JSCL+1, the FNT interlock cleared, and the job name written into this PP's input register with an exit to PPR so that the PP program to initialize the subsystem will be loaded. Once the PP program is loaded into this PP, it initializes the control point field length, and so on, to fit the requirements of the subsystem, sets up a control statement stream or procedure file call (for TAF, NIP, RBF), and calls 1AJ to process the stream which brings the subsystem into execution.

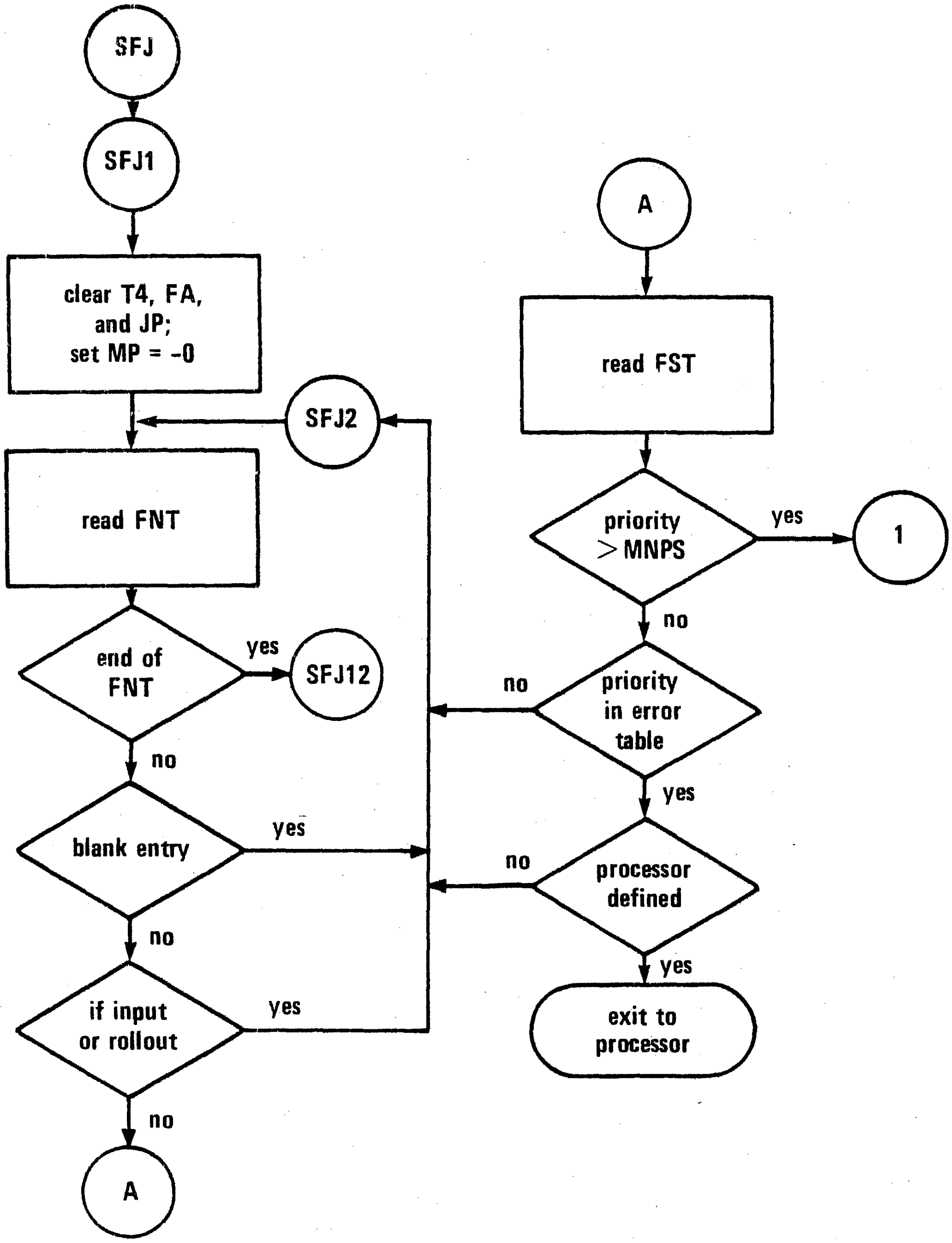


Figure 6-2. SFJ - Search For Job

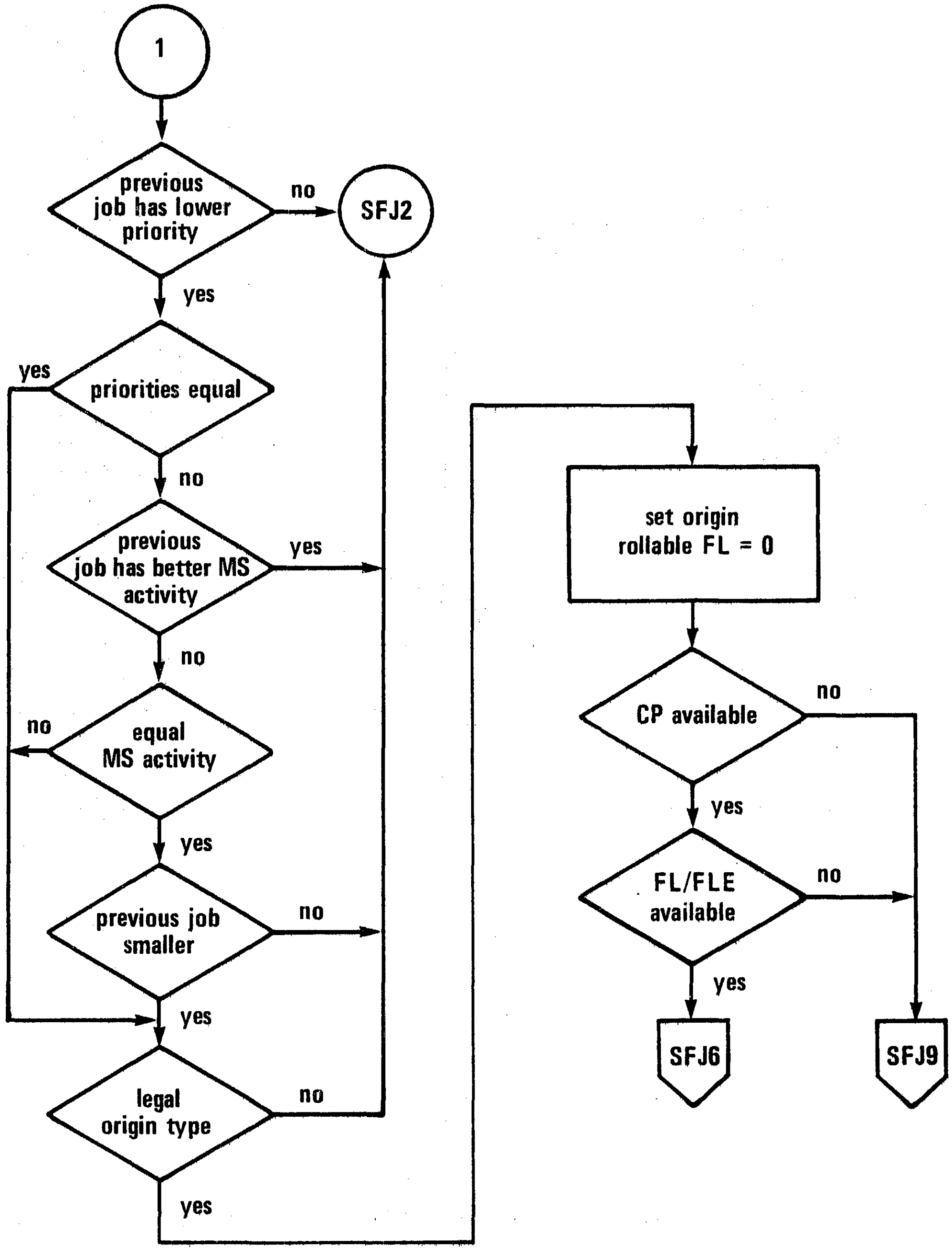
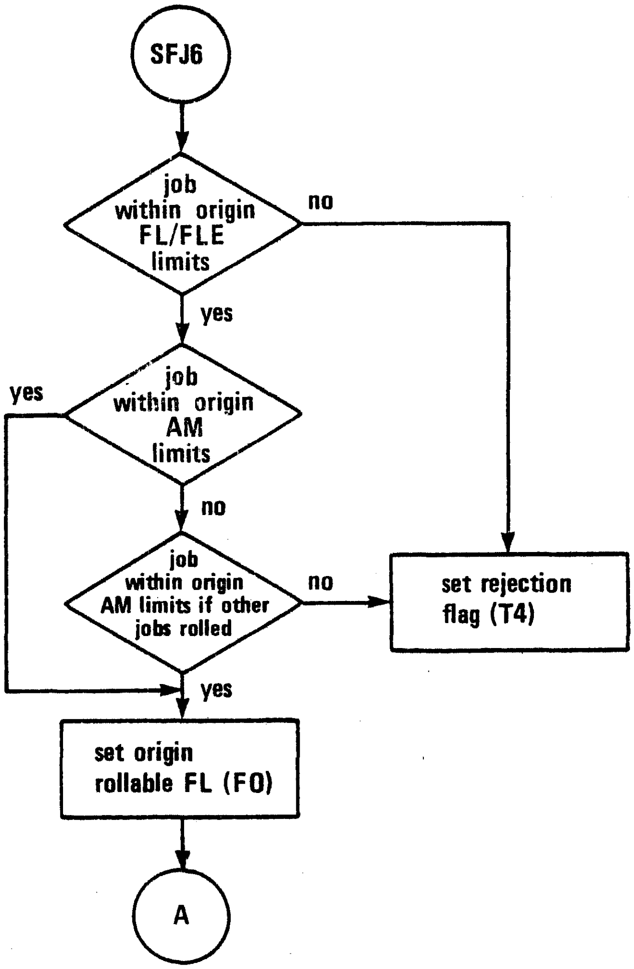


Figure 6-2. SFJ - Search For Job (Continued)



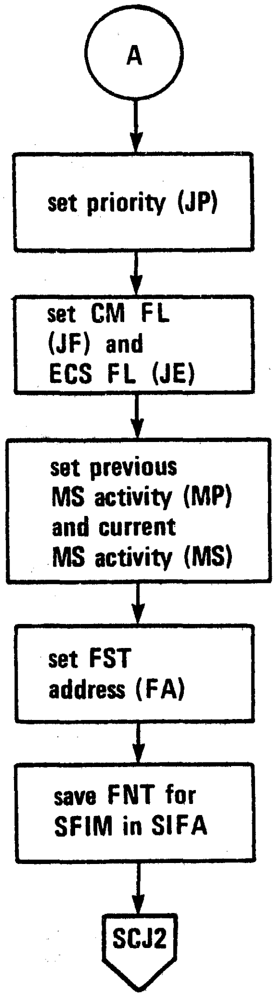


Figure 6-2. SFJ - Search For Job (Continued)

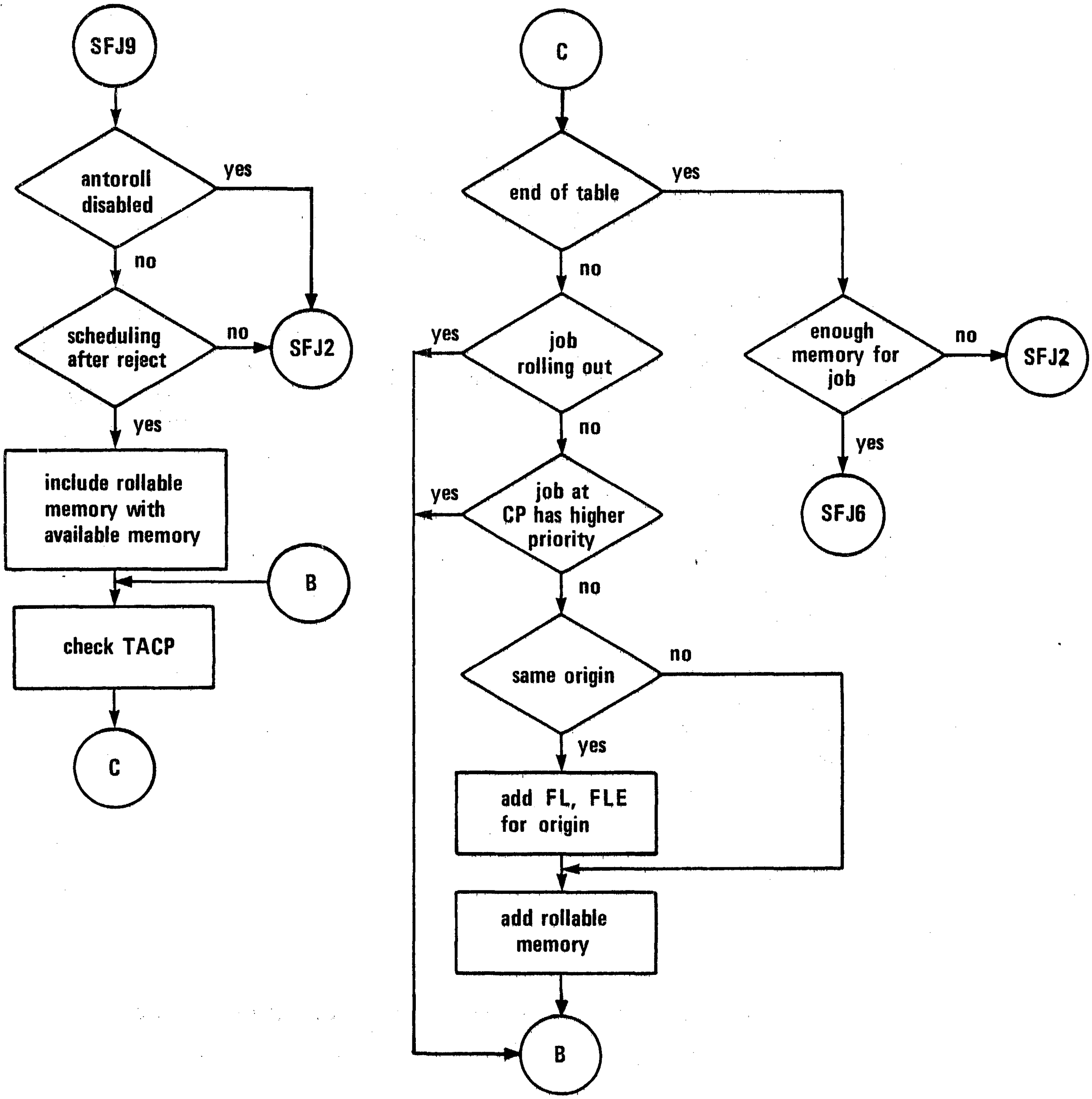


Figure 6-2. SFJ - Search For job (Continued)

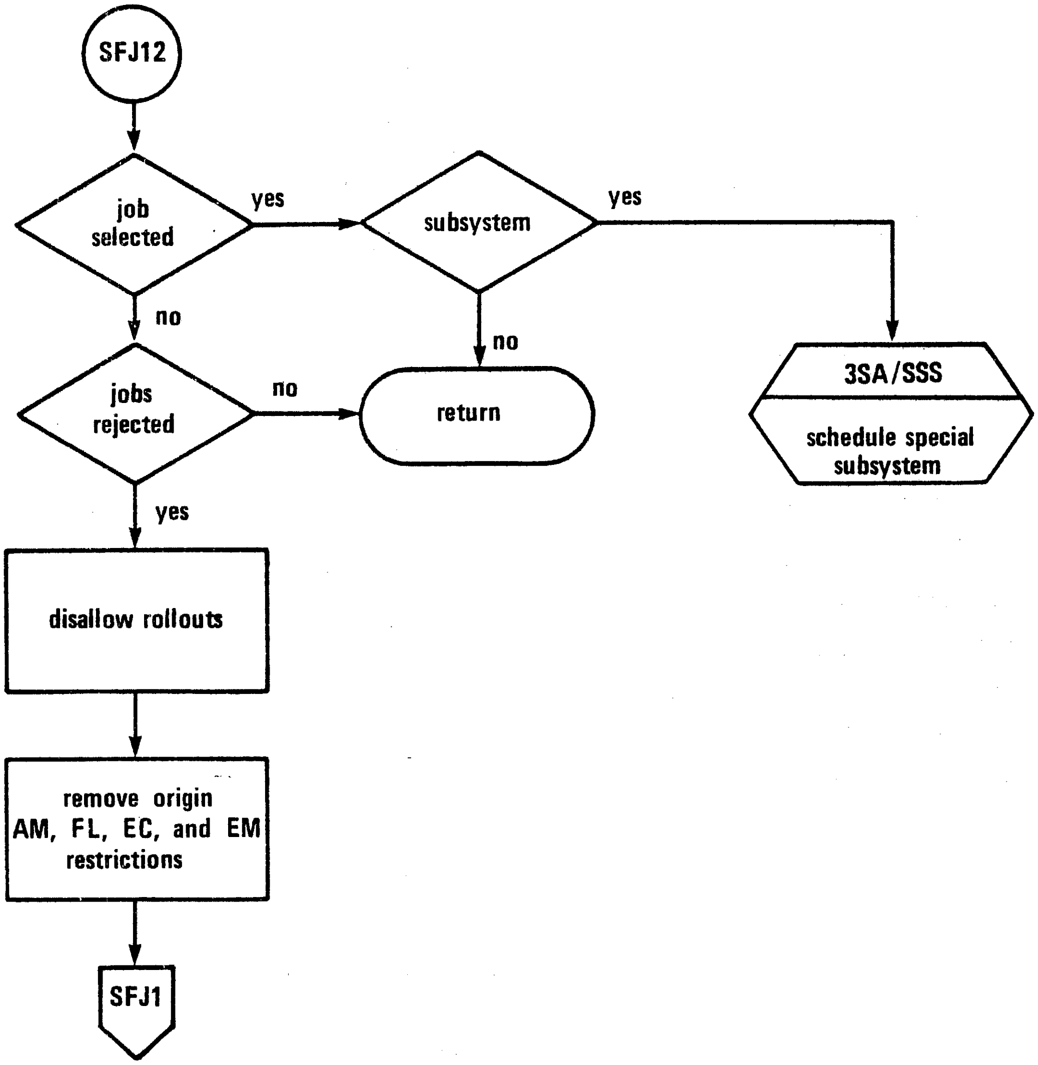


Figure 6-2. SFJ - Search For Job (Continued)

PRIORITY EVALUATOR - 1SP

Routine 1SP is called periodically by CPUMTR to perform the following functions.

- Evaluate priorities of files in various queues.
- Check central memory time slices for jobs at control points. If a time slice has expired, its priority is set to the lower bound for the job origin type and if the job is of time-sharing origin (TXOT) and output is available, it is rolled out.
- Check for device checkpoint requests and call 1CK if any are found.
- Check for device initialization requests and call CMS if any are found.
- All timed/event rollout jobs are made eligible for scheduling when the desired event has occurred or if their time has expired.
- Check for accumulator overflow and call routine OAU to update the PROFILa file accordingly.

A flowchart of the main routine of 1SP is shown in figure 6-3.

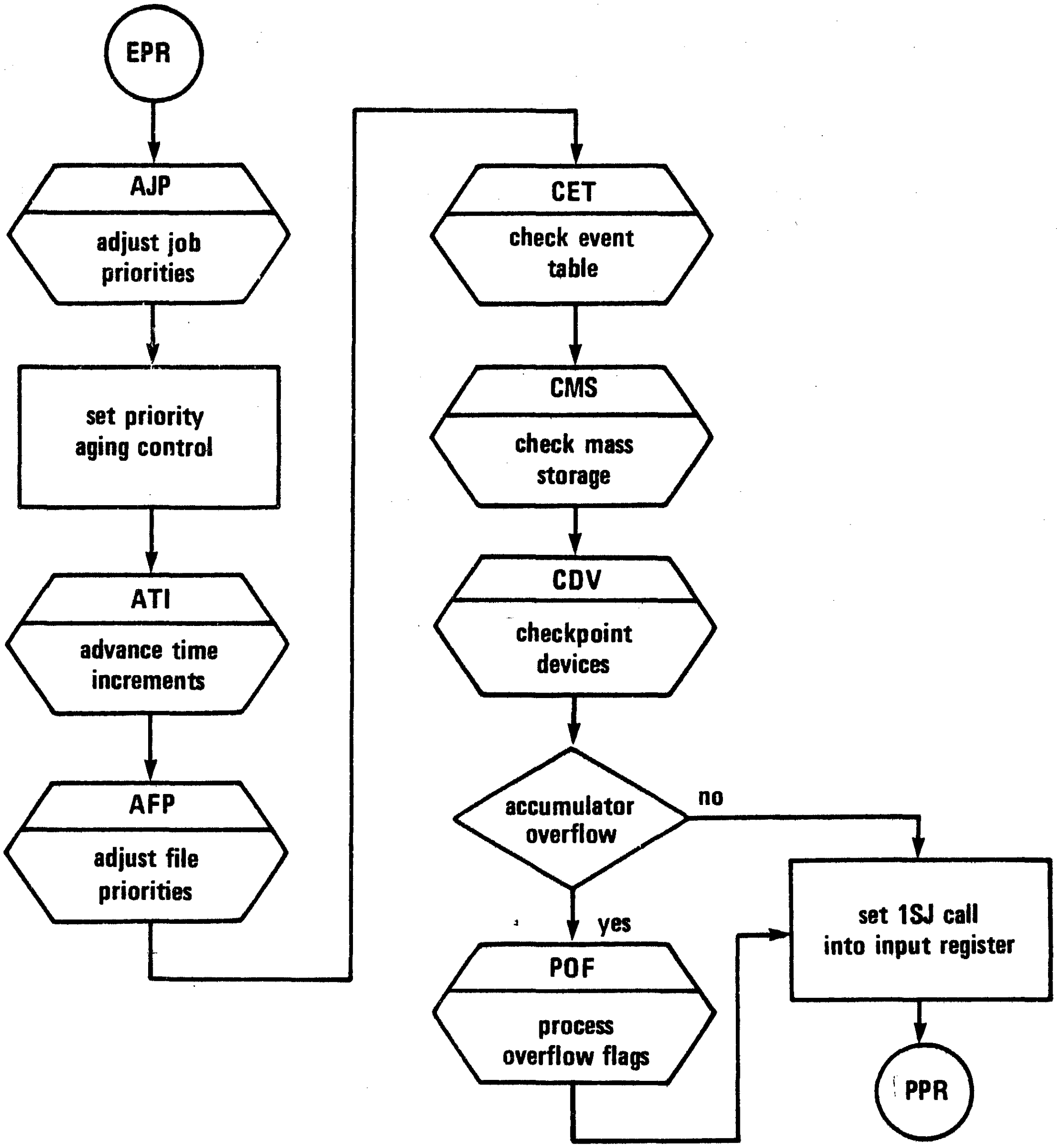


Figure 6-3. 1SP - Main Program

The following paragraphs describe several 1SP subroutines.

ADJUST JOB PRIORITIES (AJP)

AJP checks for wait response and swapout allowable indicators located in word SSCW of the control point area. If any wait response indicator is set without a corresponding swapout allowable indicator, the job receives a 2*CMSL times its specified CM slice to inhibit swapout. If wait response indicators are left set but the corresponding swapout allowable indicator is also set, the job will be considered a candidate for swapout. AJP also checks CM and CPU time slices and adjusts the job's priority if either of these has been exceeded.

ADVANCE TIME INCREMENTS (ATI)

ATI advances the increment interval associated with the IN service parameter for each queue type within the job origins.

ADJUST FILE PRIORITIES (AFP)

AFP ages queue files if priority aging is enabled. If the time increment was reset by ATI for the queue type for the origin type, the queue priority of the file being processed is advanced by one. In addition, if the time specified for a timed/event rollout file (TEFT) has expired, the file is converted to a rollout file (ROFT) and is given the upper bound rollout queue priority for its origin. When converting from TEFT to ROFT, the ECS field length is reset in FST byte 2 from the rollout file system sector.

CHECK EVENT TABLE (CET)

CET matches events from the systems event table with events specified in TEFT entries. AFP places those TEFTs waiting for events into a table for CET to read, thus requiring only one complete scan of the FNT to complete. If events match, the file is converted to a rollout file and given the upper bound rollout queue priority for its origin type.

CHECK MASS STORAGE (CMS)

CMS determines if a call to the mass storage subsystem is necessary. The following criteria are used.

- When its delay (maintained in PFNL+1) has expired and removable packs are enabled
- When CMS is required to diagnose mass storage error conditions
- When initializations are pending on a mass storage device

The activation of CMS is made by making a 1DS function 32 call to initiate the mass storage subsystem.

If it is necessary to call 1DS, the scheduler active bit (bit 59 in JSCL+1) is cleared, the scheduler is requested via an RSJM function, and the 1DS request is written into this PP's input register and PPR is entered.

CHECK IF CHECKPOINT NEEDED (CDV)

CDV checks the 1CK recall time in JSCL+1 and calls 1CK if it is time to issue a checkpoint request and checkpoint requests were detected by CMS. The call to 1CK is entered into this PP's input register and PPR is entered after clearing the scheduler active bit and requesting the scheduler via an RSJM function.

PROCESS OVERFLOW FLAGS (POF)

POF detects accumulator overflow at a control point and if overflow exists calls overlay OAU to update the PROFILa file accordingly.

ADVANCE JOB STATUS - 1AJ

Routine 1AJ advances the status of an active job. This action may be caused by one of the following occurrences.

- The job scheduler (1SJ) wants to start a new job just scheduled to a control point
- Monitor has sensed no activity at a control point (W and X bits clear)
- DIS or other similar programs wish to process an error flag or a control statement

The format of the 1AJ call is as follows.

	59	41		35	23 0
RA+1	1AJ	0	cp	fn	params

cp	Control point number
fn	Function number 0 through 5
params	Parameters depending upon the function number

For function number 0, TCS can be substituted for a 1AJ call. For function numbers 4 and 5, the call must be made to TCS rather than 1AJ.

The parameters for each function number are as follows.

fn	Bits	Description
0	23-12	Equal to 1 if set by 1AJ during DMP= processing in case of recall
	11-0	Equal to 1 if set by 1RO upon completion of DMP= processing; set to 2 by DIS for SSJ= and DMP= processing
1	23-12	Zero
	11-0	Address of input file from 1SJ
2	23-3	Zero
	2	Set indicates control statement in MS1W (from DIS)
	1	Set indicates return error message to MS2W with no error flag on invalid control statement (from DIS)
	0	Set indicates read statement and stop prior to execute (RSS indicator)
3	23-0	Zero (from other PP programs)
4	23-18	Subfunction number for reading control statement. 0 Advance pointers 1 Read only if not a local file load, do not advance pointers 2 Set bit 17 in argument count if local file load; do not advance pointers 4x If parameters to be cracked in product set format
	17-0	Address to read/write control statement from/to
5	23-18	Zero
	17-0	Address from which to read control statement (for control statement read and execute)

The programs called by 1AJ are as follows.

Program	Description
1CJ	Complete job
1RI	DMP= rollin
1RO	Rollout job, normal rollout and DMP= rollout
CIO	Complete special files on errors
DMP	Exchange package dump (for certain error flags)
OAU	Update PROFILa file
ODF	Drop file

The common direct location assignments are:

Name	Value	Description
AB	20-24	Assembly buffer
CN	25-31	CM word buffer
FS	32-36	FST entry
EP	37	Entry point pointer
SP	40-44	Statement pointer
OT	45	Job origin type
EF	46	Error flag
RO	47	Rollout flag
FA	57	Address of FST entry
CW	60-64	Library control word
RF	65	Reprieve error flag
SC	67	System control point (SCP) activity

In general, 1AJ is called by MTR, 1SJ, or DIS. However, in the case of special entry point programs 1RO will call 1AJ back after rolling a job out to DM* and setting up a control point for the special entry point routine. A special entry point job can be rolled out, and when it is rolled back in, 1RI calls 1AJ to advance it.

Interaction between 1AJ, 1SJ, MTR, 1RI, and 1RO is illustrated in figure 6-4.

1AJ uses the following overlays.

Overlay	Description
3AA	Begin job
3AB	Process error flag
TCS	Translate control statement
LDR	Load central program
3AC	Search peripheral library
3AD	Search for overlay
3AE	Load copy routines
3AF	Special entry point processor
3AG	Termination processing
3AH	Return special user files

The PP memory layout is shown in figure 6-5.

Figure 6-6 contains the flowchart of the main routine of 1AJ.

NOTE

Control point area words used by 1AJ are described in section 2.

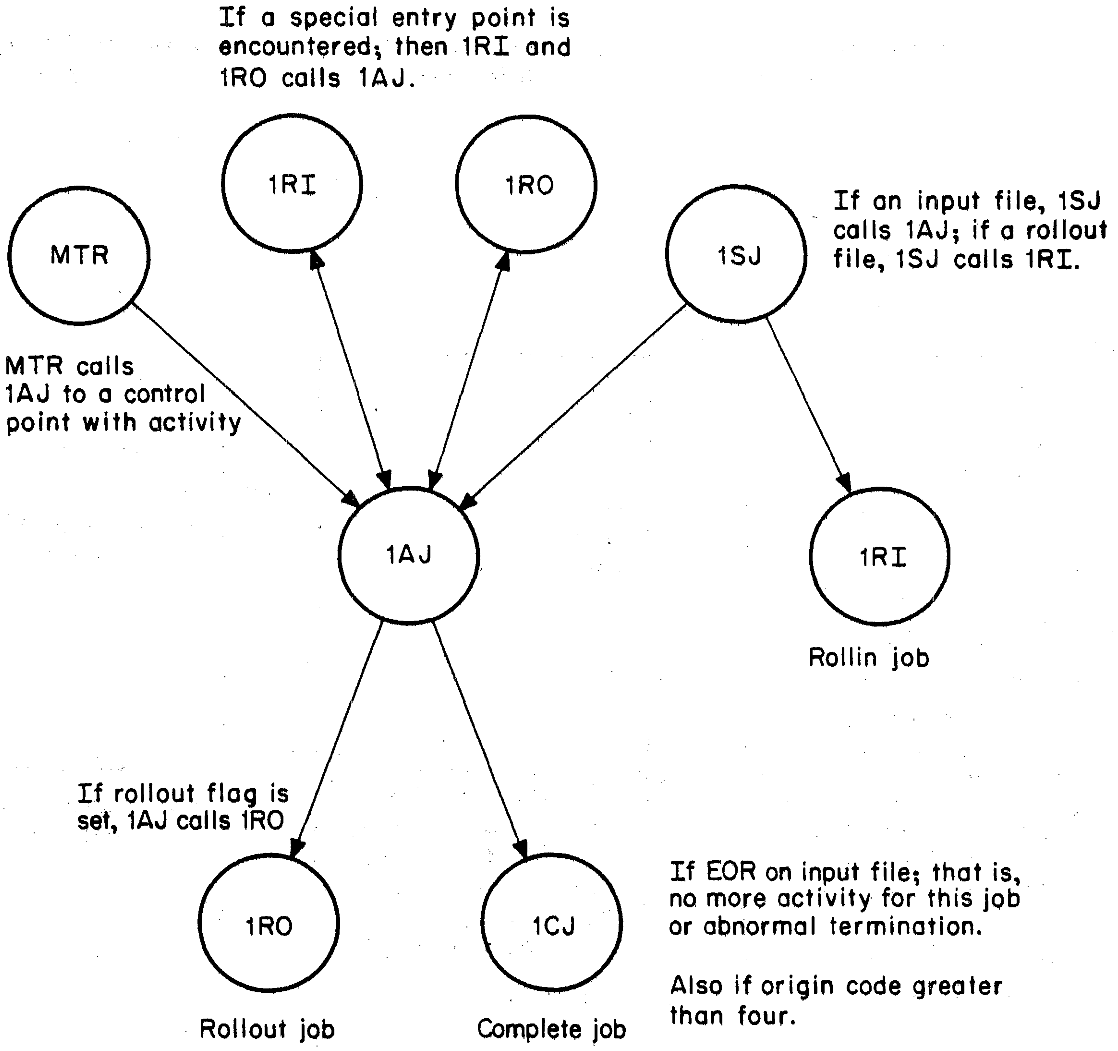


Figure 6-4. 1AJ Interaction

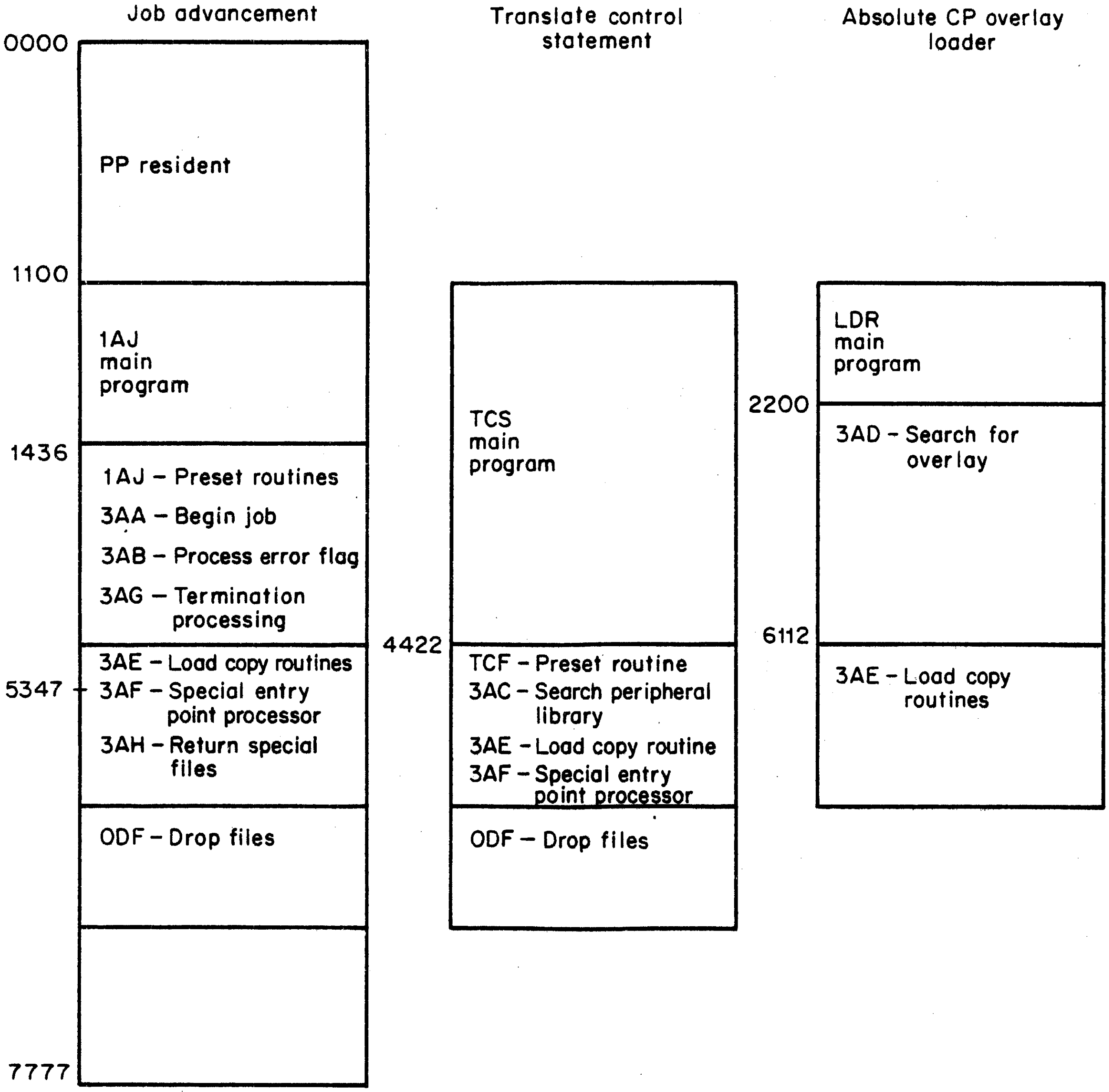
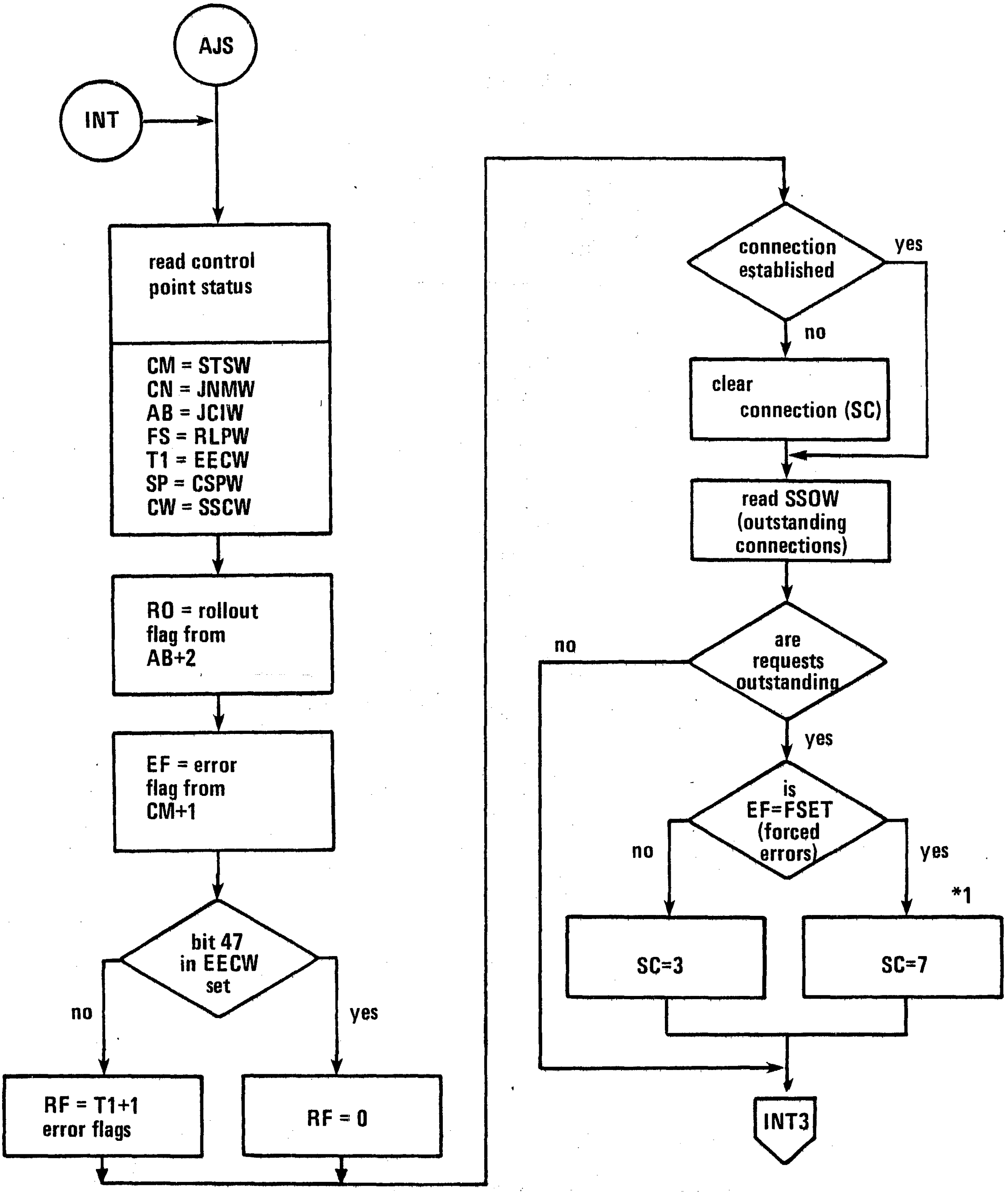
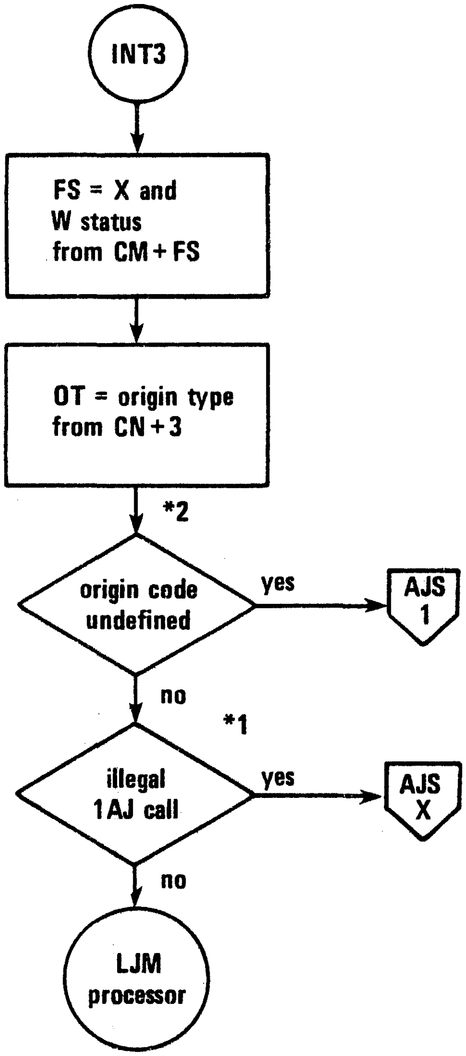


Figure 6-5. 1AJ Major Overlay Memory Layout



*1 Read one CM word into 5 PP words

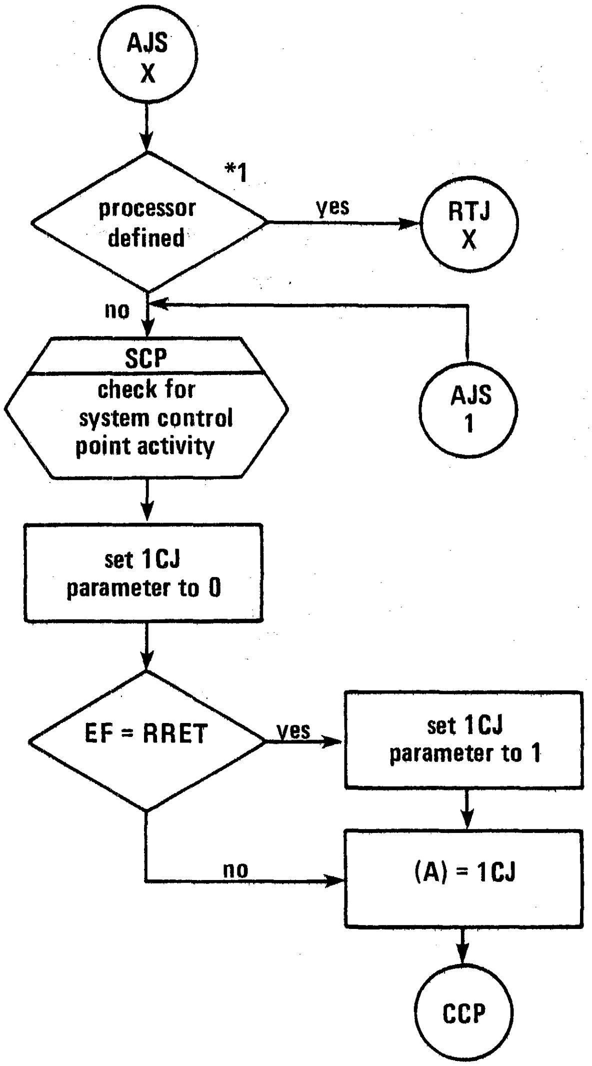
Figure 6-6. 1AJ - Advance Job (Continued)



*1 Is function number from IR+2 $\geq$ 4 (functions 4 and 5 are TCS functions)

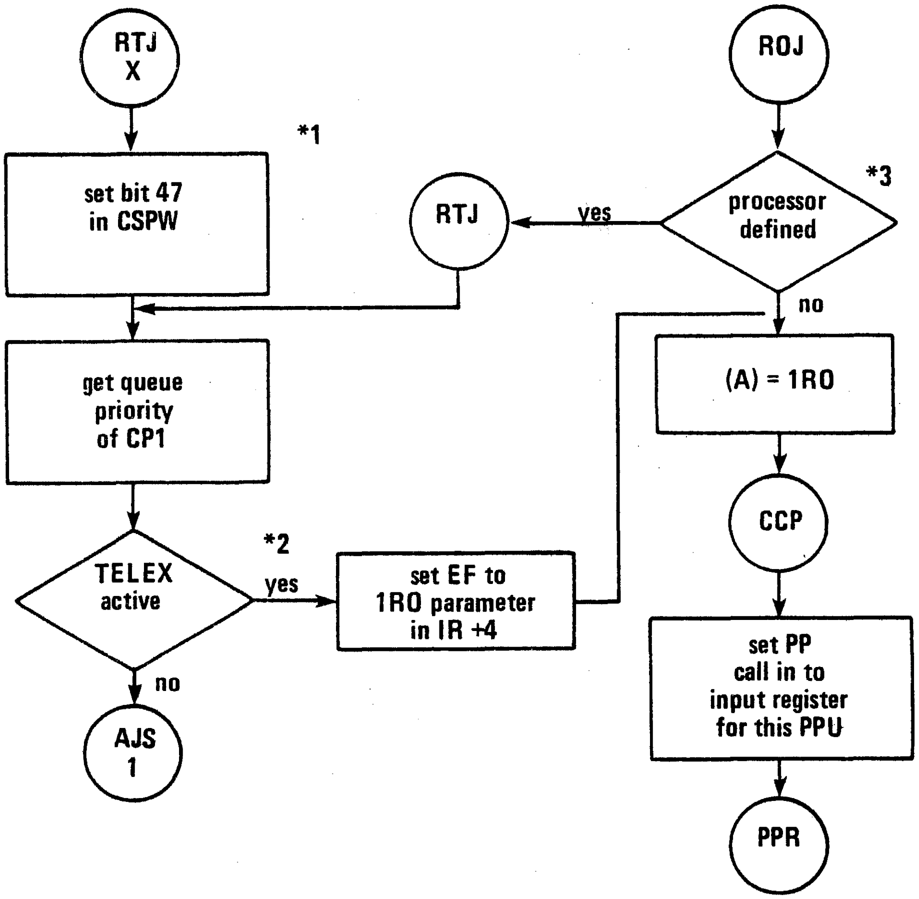
*2 Protective code. If an origin code > 4 is not trapped, the processors will malfunction and the system could crash.

Figure 6-6. 1AJ - Advance Job (Continued)



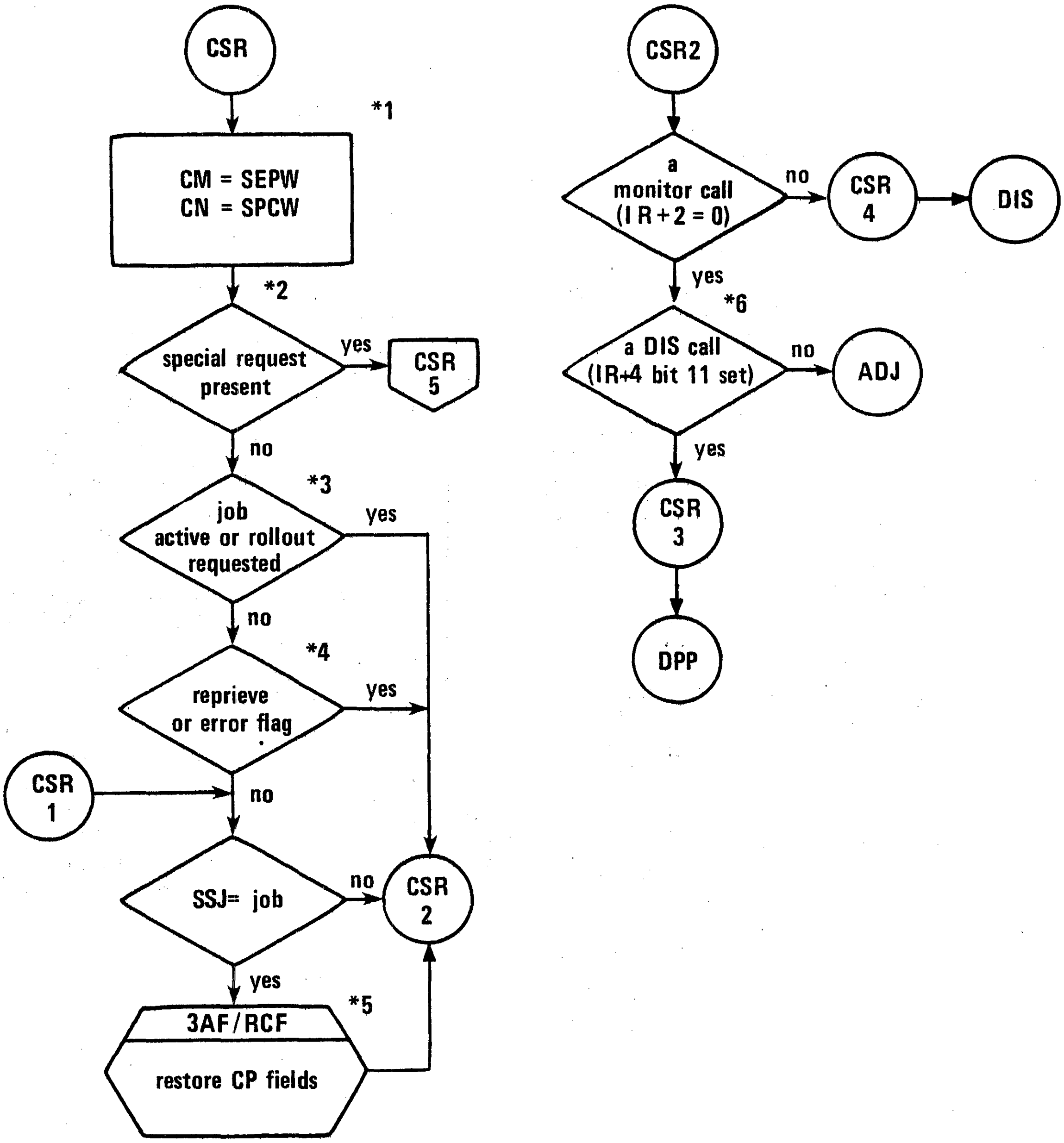
*1 Only TXOT and MTOT have a processor (RTJX); no processor exists for the other origin types.

Figure 6-6. 1AJ - Advance Job (Continued)



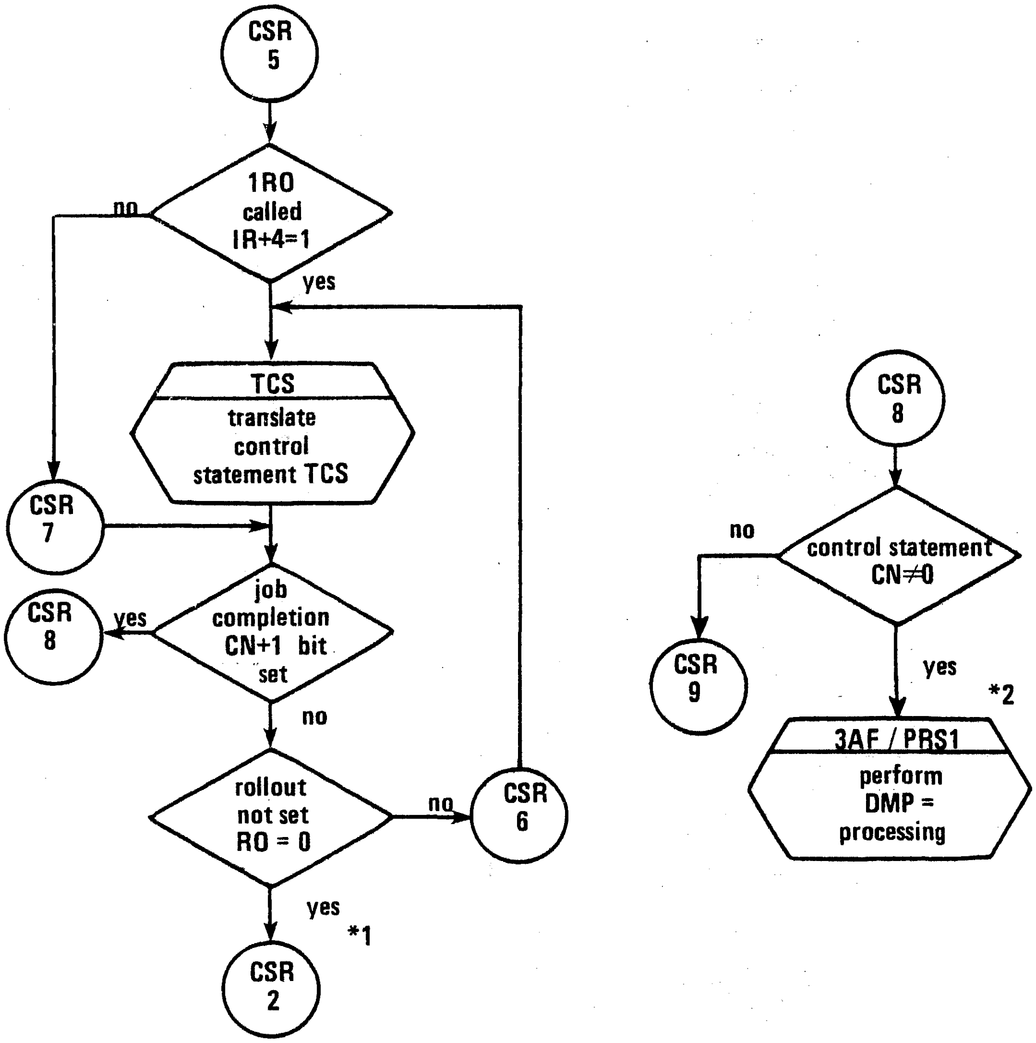
*1 Ensure empty control statement buffer by indicating EOR.
*2 Is queue priority of job at control point 1 equal to TXPS?
*3 Only TXOT and MTOT have a processor (RTJ); no processor exists for the other origin types.

Figure 6-6. 1AJ - Advance Job (Continued)



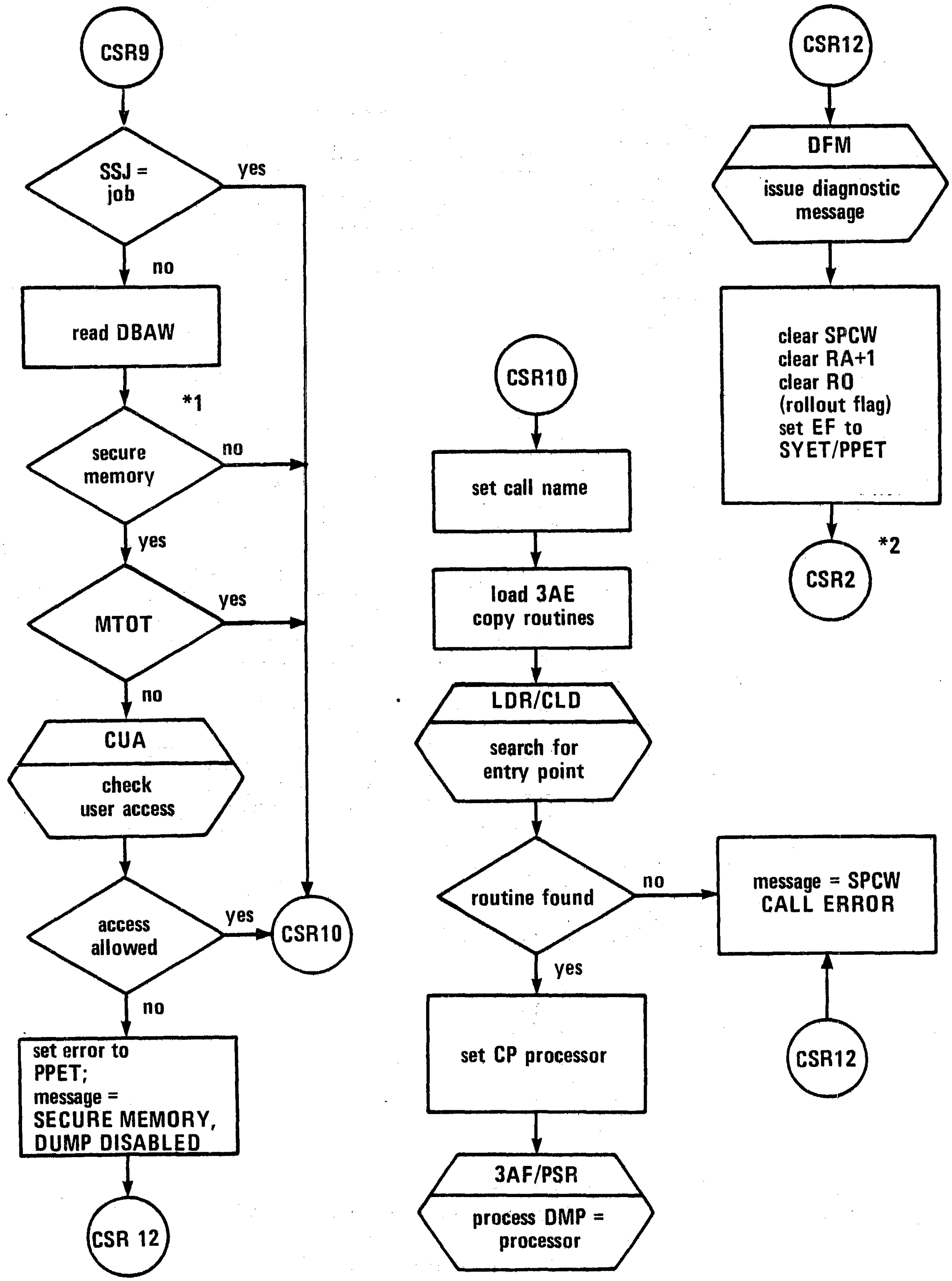
*1 Read 1 CM word into 5 PP words.
*2 Is CN=CP entry point name ≠ 0?
*3 Is RO+FS rollout flag + W + X status?
*4 Is RF or EF not = zero?
*5 See description of overlay 3AF.
*6 Function 0 call with n=2.

Figure 6-6. 1AJ - Advance Job (Continued)



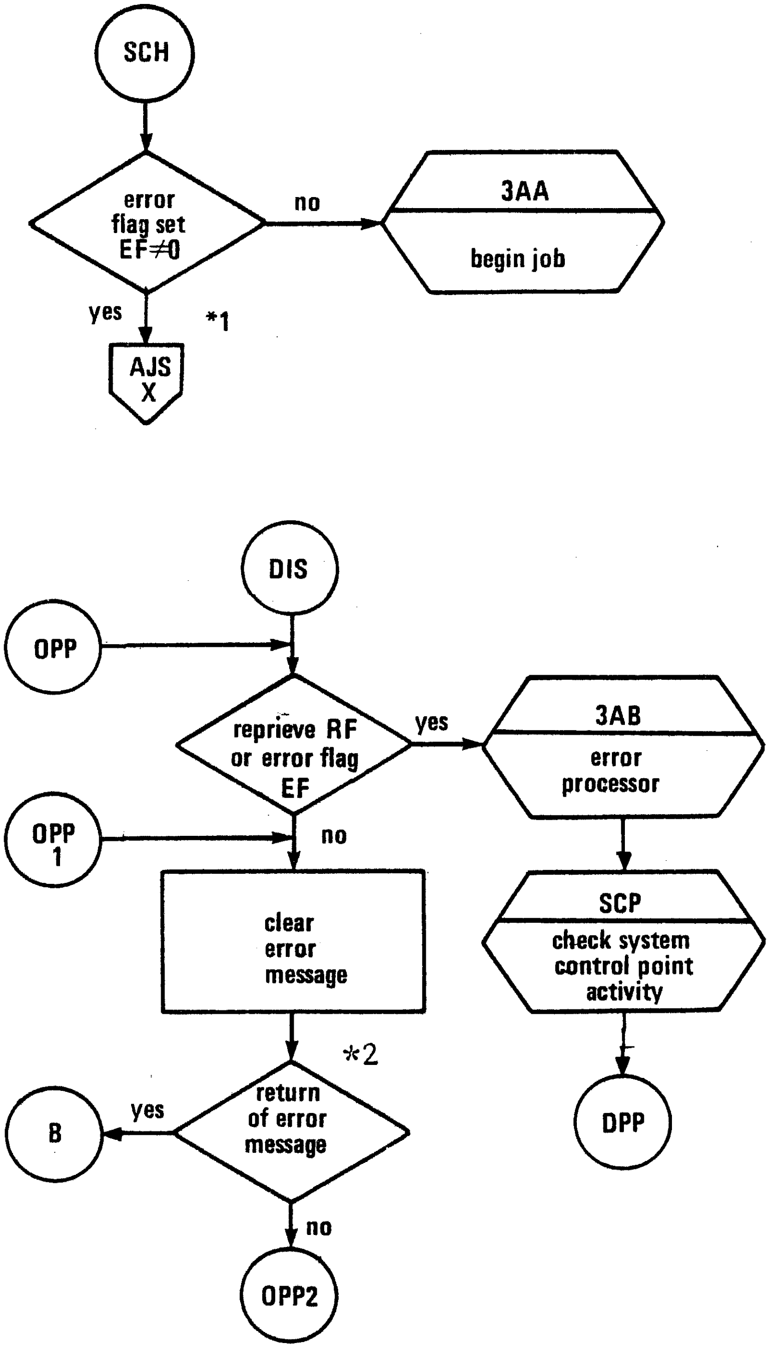
*1 This path forces job to be rolled out and 1AJ to drop.
*2 3AF exits via a call to 1RO and drops from PP.

Figure 6-6. 1AJ - Advance Job (Continued)



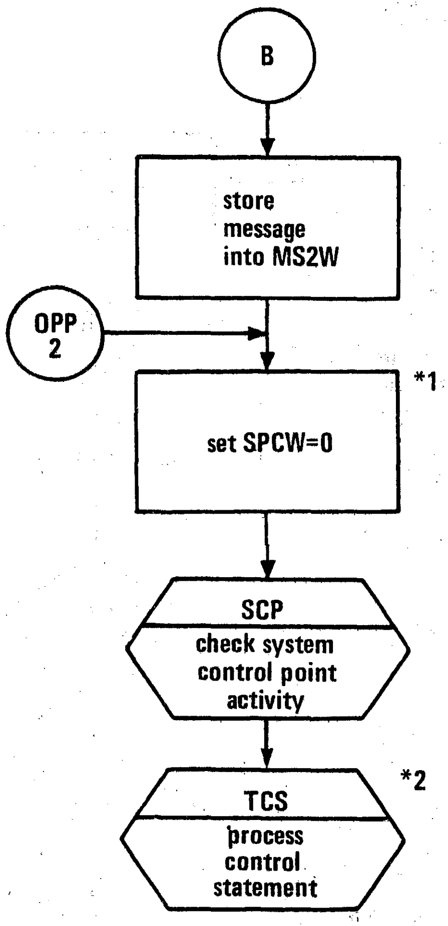
*1 Bit 59 of DBAW set.
*2 1AJ drops and this control point aborts.

Figure 6-6. 1AJ - Advance Job (Continued)



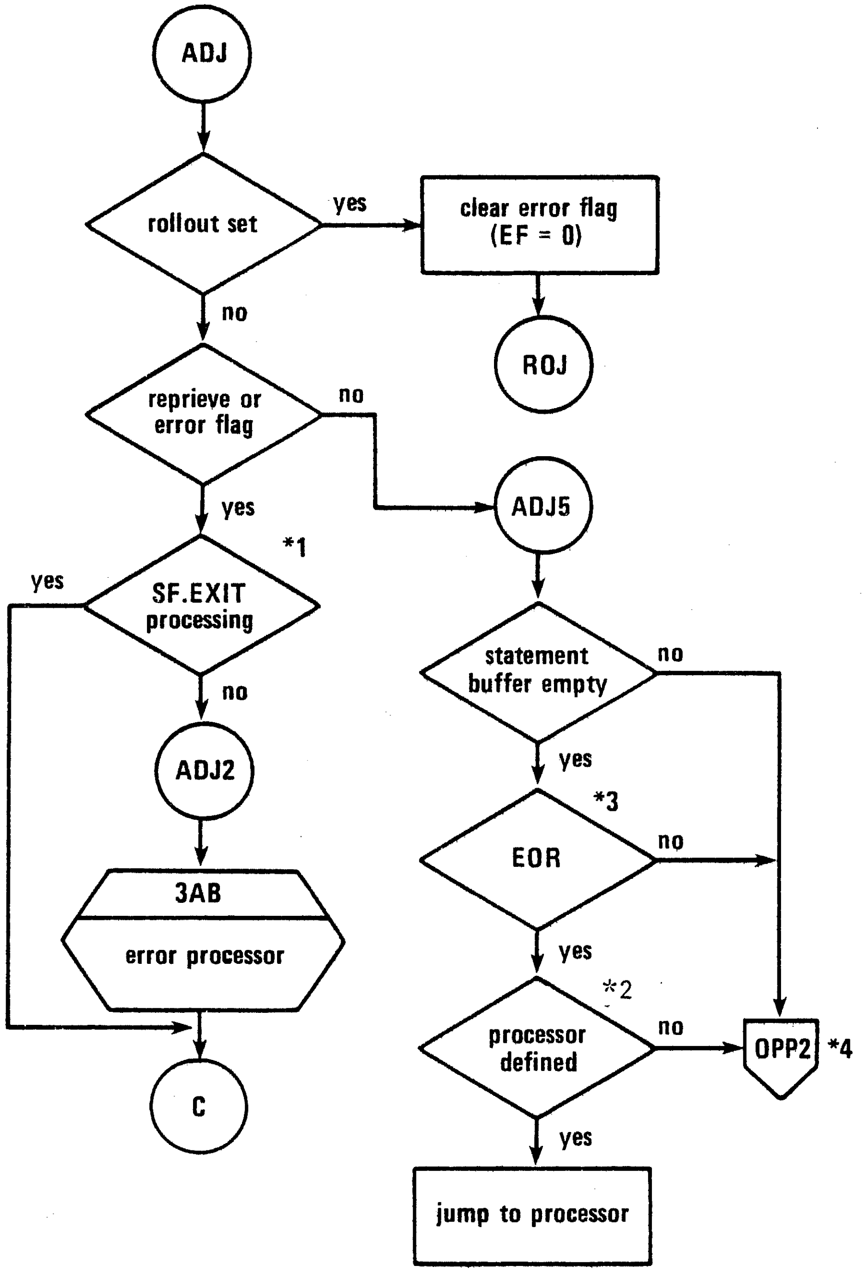
*1 Exit to 1CJ if error flag set.
*4 Bit 1 of IR+4 not set indicates no return of message.

Figure 6-6. 1AJ - Advance Job (Continued)



*1 Turn off any special processor commands.
*2 Read next control statement and advance the job. If illegal control statements abort.

Figure 6-6. 1AJ - Advance Job (Continued)

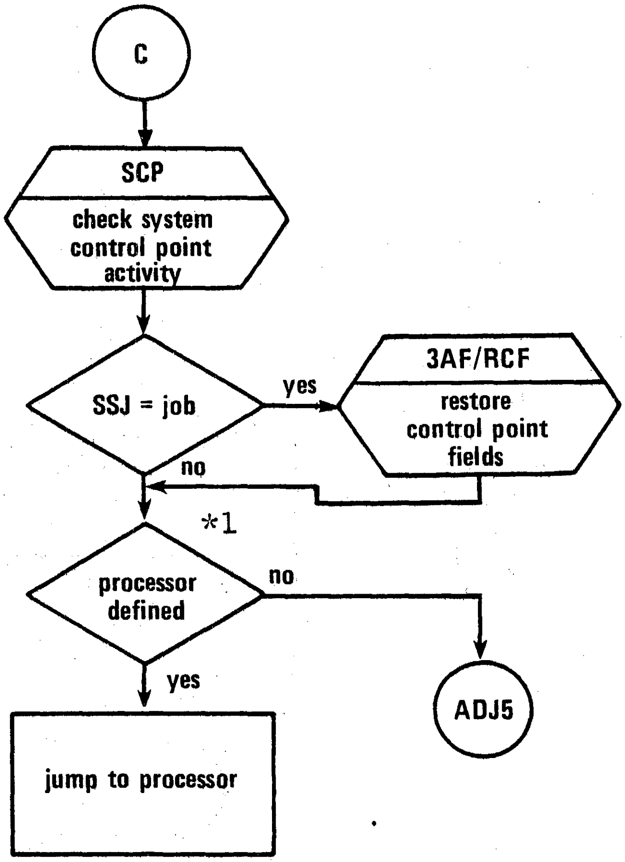


*1 (SC) bit 2 set.
*2 Processors are defined as follows.

SYOT	AJSX
BCOT	AJSX
EIOT	AJSX
TXOT	RTJ
MTOT	RTJ

*3 Is this the end of control statements; then terminate
*4 Calls TCS

Figure 6-6. 1AJ - Advance Job (Continued)



*1 Only TXOT and MTOT have processor (RTJ) defined; no processor exist for other job origins.

Figure 6-6. 1AJ - Advance Job (Continued)

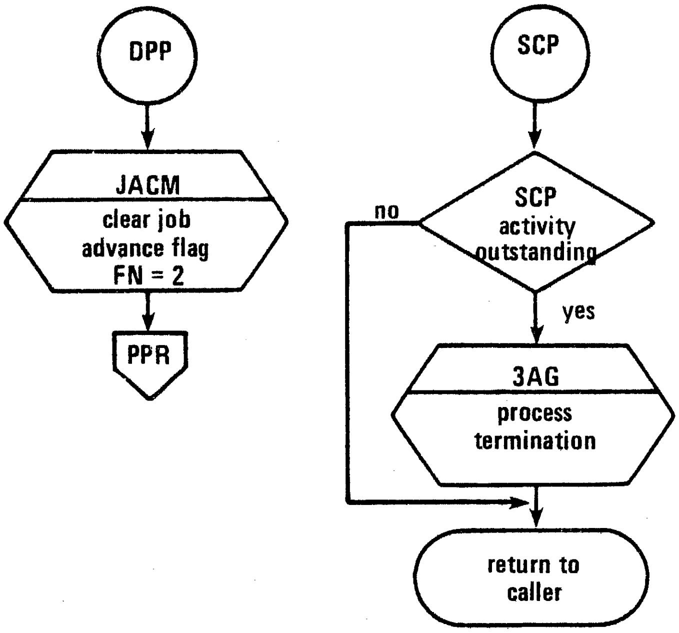


Figure 6-6. 1AJ - Advance Job (Continued)

The following paragraphs describe 1AJ subroutines.

BEGIN JOB (3AA)

Routine 3AA initiates job processing at a control point. The dayfile messages issued by 3AA are the following.

```
JOB CARD ERROR.
BINARY CARD xxxx SEQUENCE ERROR.
JOB IN NORERUN STATE ON RECOVERY.
```

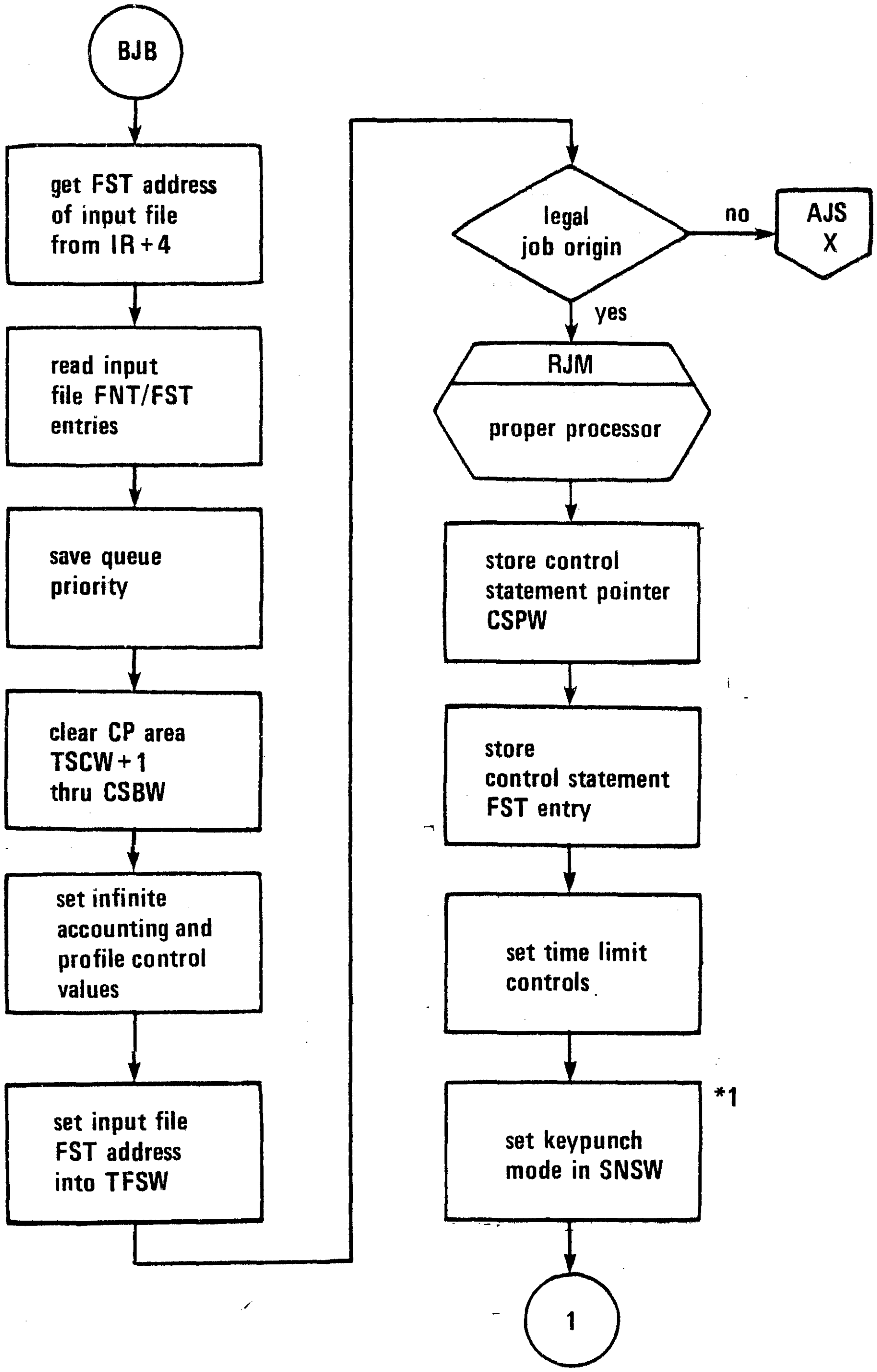
The direct location assignments are defined as follows.

Name	Value	Description
PP	60	Pot pointer
TN	61	Terminal number
PA	62	Pot address (2 words)
TT	64	Terminal table address (2 words)
TA	66	TELEX reference address

The table of processors for 3AA is as follows.

Origin	Processor
SYOT	BBC
BCOT	BBC
EIOT	BBC
MTOT	BMT

A flowchart of 3AA is shown in figure 6-7.



*1 Keypunch mode is passed to 1AJ in the system sector of the input file

Figure 6-7. 3AA - Begin Job

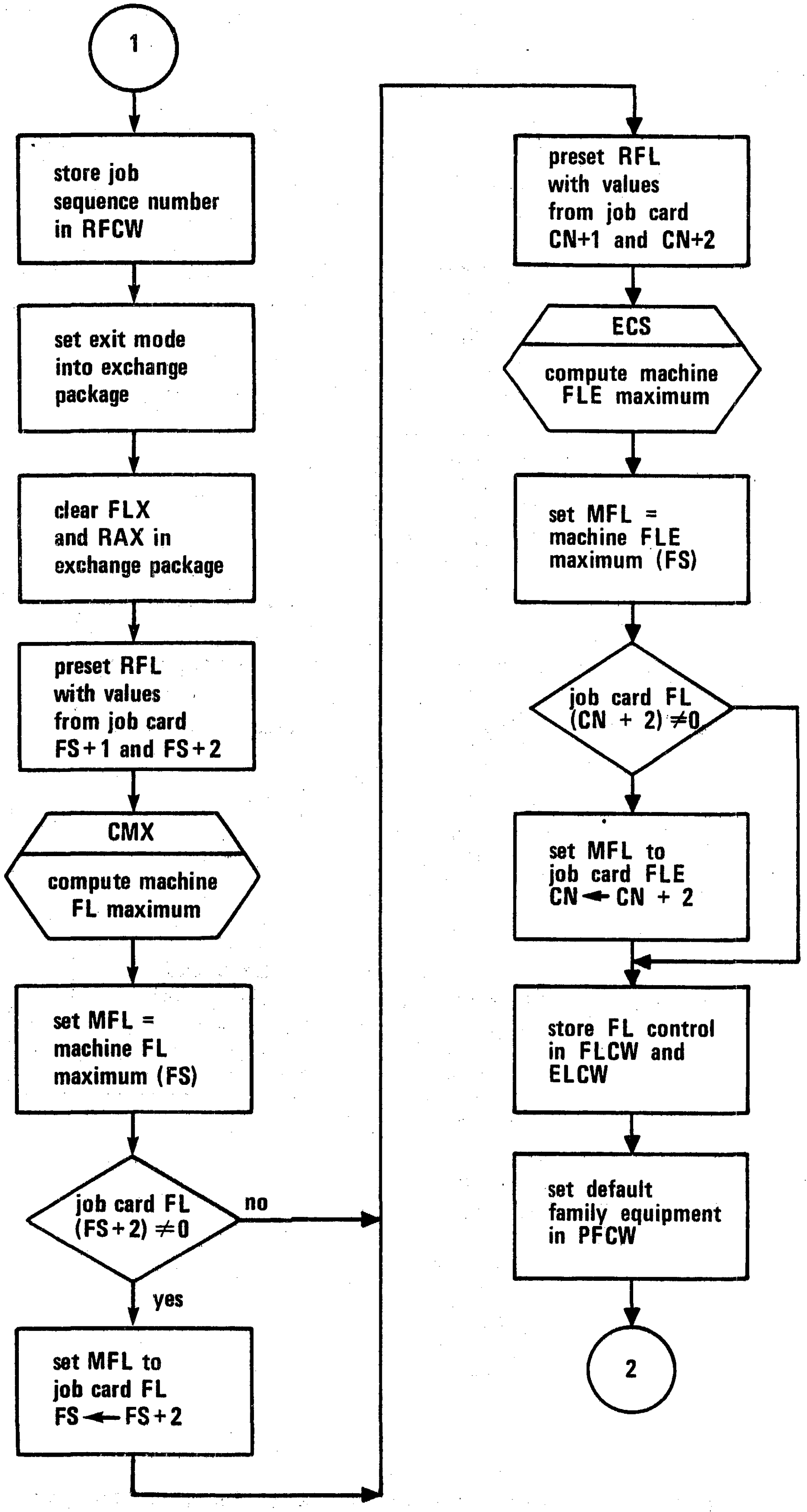
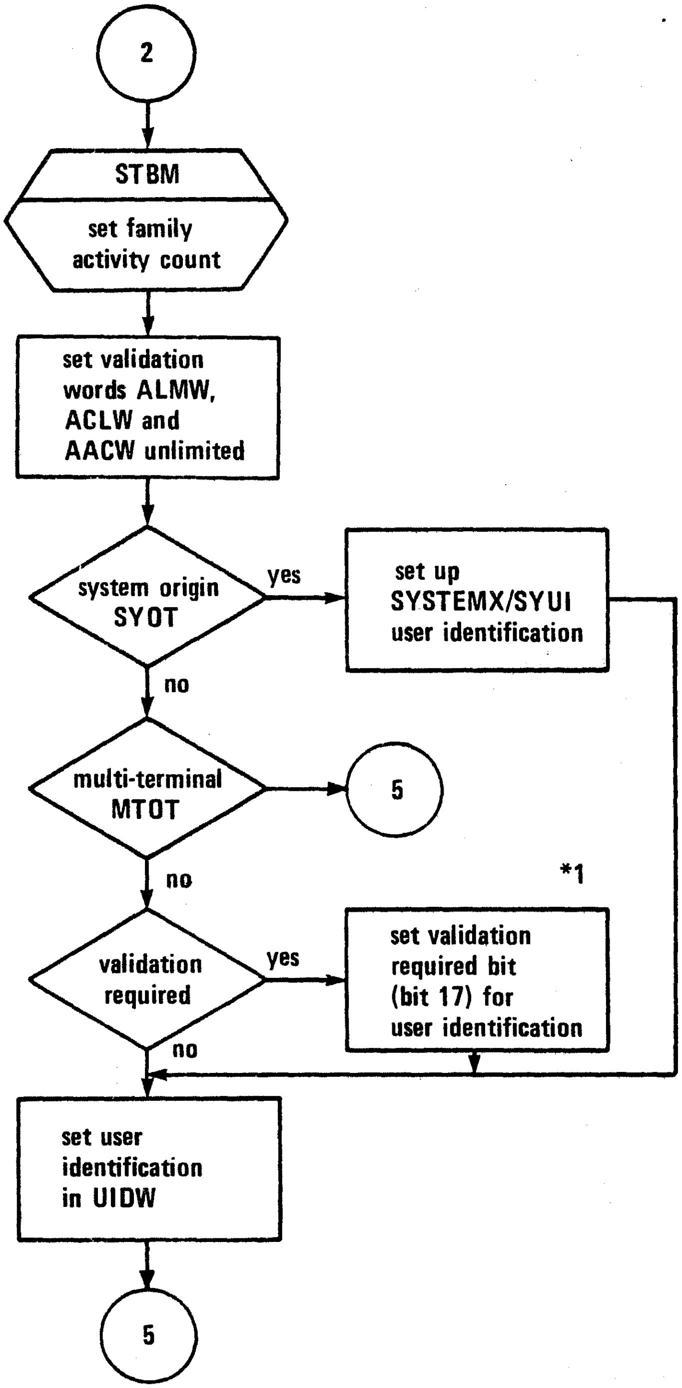
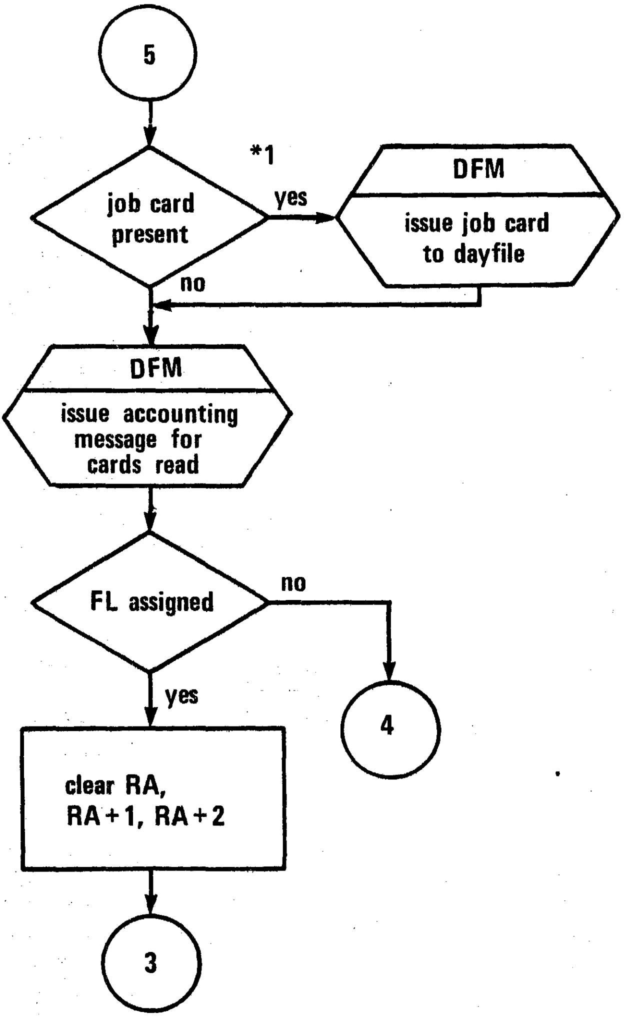


Figure 6-7. 3AA - Begin Job (continued)



*1 Validation required bit from SSTL is set as bit of UIDW

Figure 6-7. 3AA - Begin Job (Continued)



*1 Job card not present if MTOT.

Figure 6-7. 3AA - Begin Job (Continued)

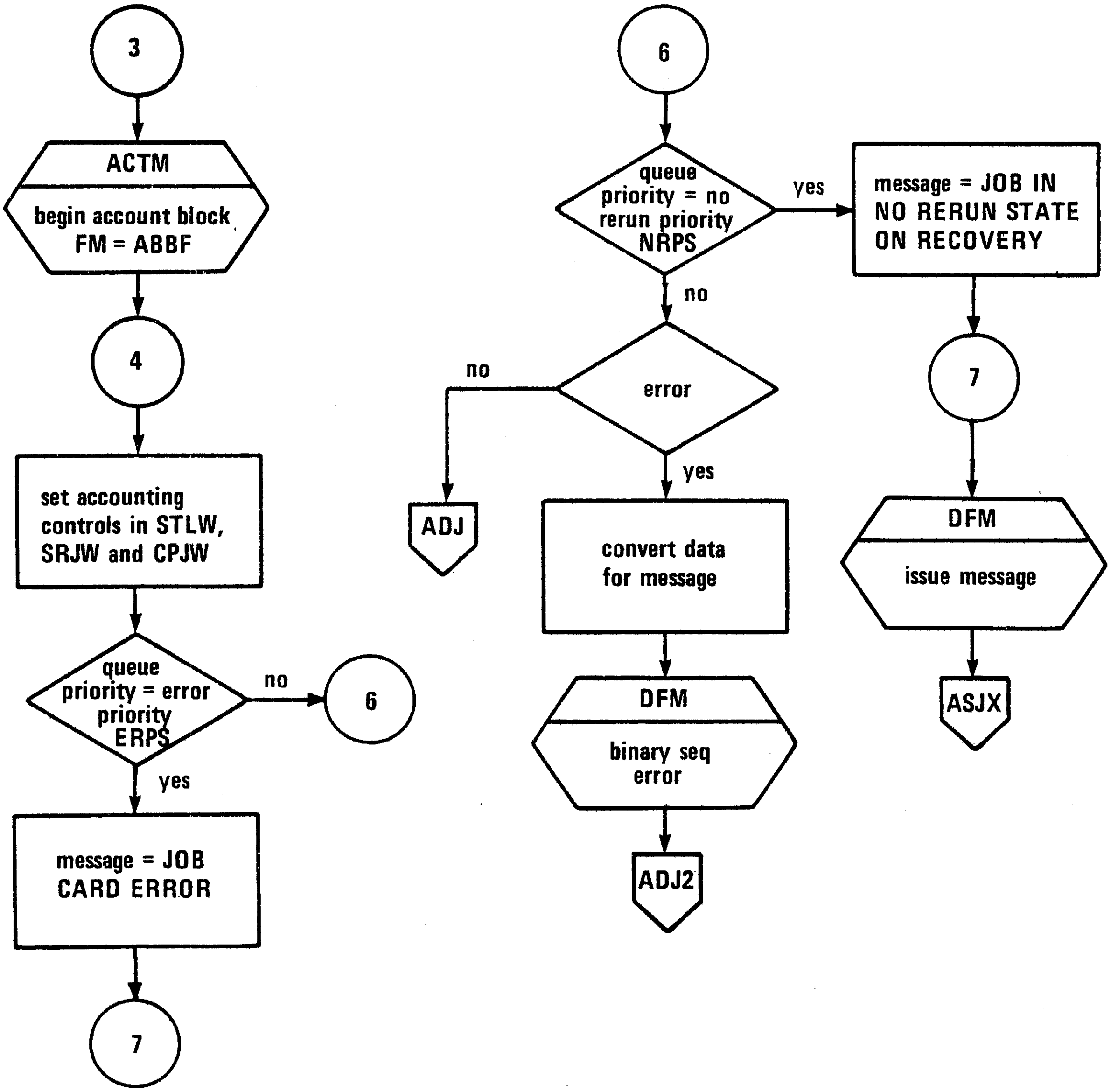
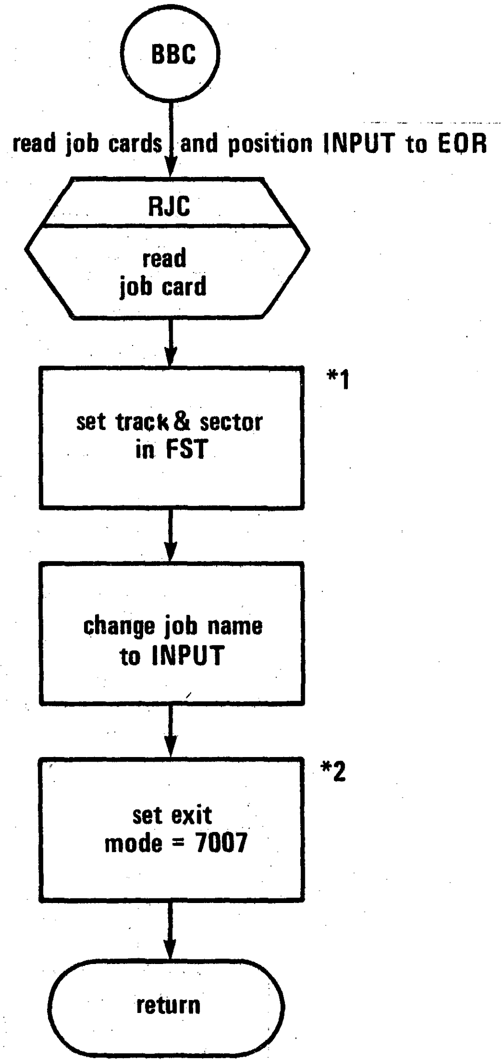
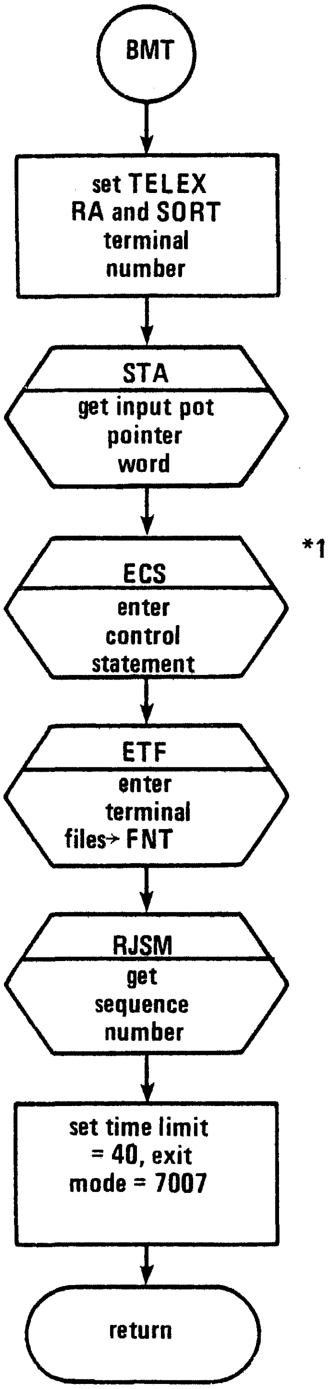


Figure 6-7. 3AA - Begin Job (Continued)



*1 The FNT/FST entry is described in section 2
*2 For use in the exchange package

Figure 6-7. 3AA - Begin Job (Continued)



*1 Read statement from TELEX pot and set up control statement

Figure 6-7. 3AA - Begin Job (Continued)

PROCESS ERROR FLAG (3AB)

Routine 3AB processes error flags by sending an error message to the dayfile. In the case of an arithmetic error, a call is made to DMP to dump the exchange package area.

When these operations are complete, the control statement buffer is searched for the control statement EXIT. If this statement is found, 3AB returns to 1AJ to continue statement processing. If an EXIT is not found, control returns to 1AJ to complete the job processing.

The dayfile messages are as follows.

Message	Description
TIME LIMIT.	The monitor has detected that the time limit for the job has expired.
CPU ERROR EXIT xx AT yyyyyy.	The monitor has detected CPU error exit condition at xx address yyyyyy.
PP CALL ERROR.	The monitor has detected an error in a CPU request for PP action.
OPERATOR DROP.	The operator has dropped the job.
PROGRAM STOP AT xxxxxx.	The monitor detected a program stop instruction at address xxxxxx.
SUBSYSTEM ABORTED.	A subsystem has aborted and all user jobs connected to this subsystem will have this message sent to their dayfiles and the SSET error flag set.
JOB STEP LIMIT.	The job step SRU limit has expired.
ACCOUNT BLOCK LIMIT.	The SRU limit for the account block has expired.
MONITOR CALL ERROR.	An illegal RA+1 call has been issued.
SYSTEM ABORT.	The job has been aborted with an SYET error type.
OPERATOR KILL.	The operator has killed the job. (Same as an operator drop except no error processing is done.)
SECURE MEMORY, DUMP DISABLED.	3AB attempts to produce an exchange package dump, but program has secure memory status.

SPECIAL REQUEST PROCESSING ERROR.	3AB attempts to produce an exchange package dump, but a program is a special call processor (SPCW set).
REPRIEVE IMPOSSIBLE - BAD CHECKSUM.	The checksum does not match checksum taken when reprieve control set up.
JOB REPRIEVED.	Job is reprieved after an error. A second message is issued to describe the conditions under which the job was reprieved.

The table of processors for 3AB is as follows.

Origin	Processor
SYOT	EBC
BCOT	EBC
EIOT	EBC
TXOT	EBC
MTOT	EBC

Overlay 3AB is flowcharted in figure 6-8.

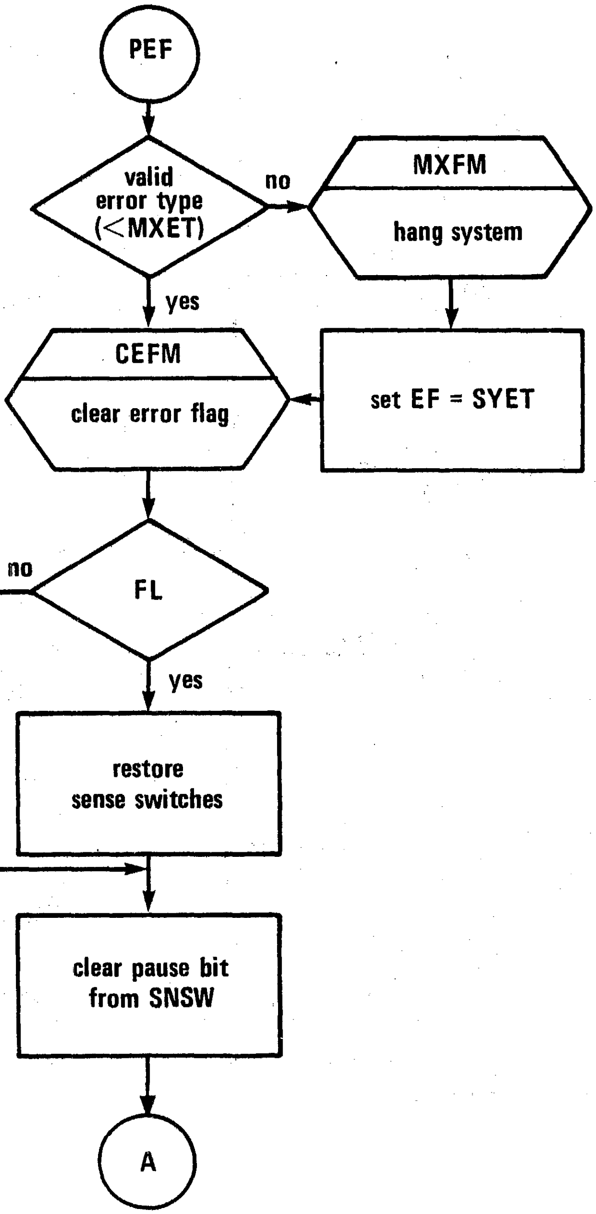


Figure 6-8. 3AB - Process Error Flag

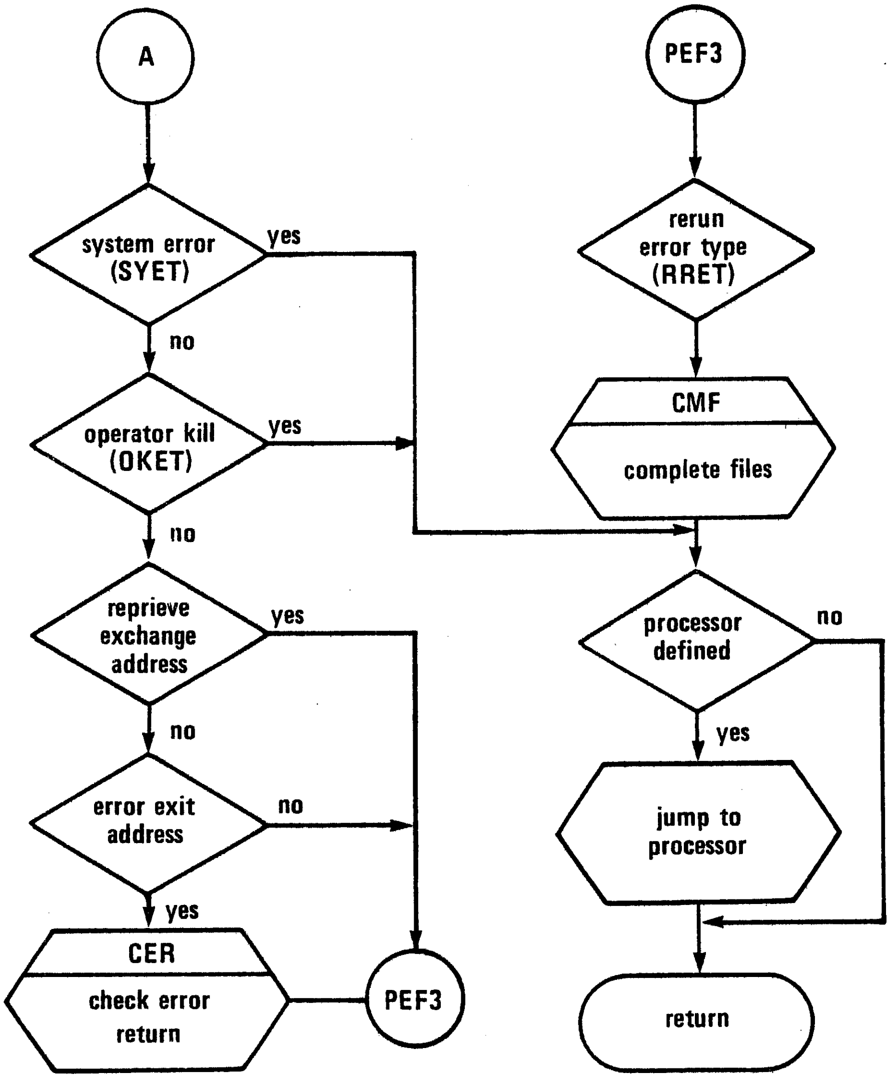
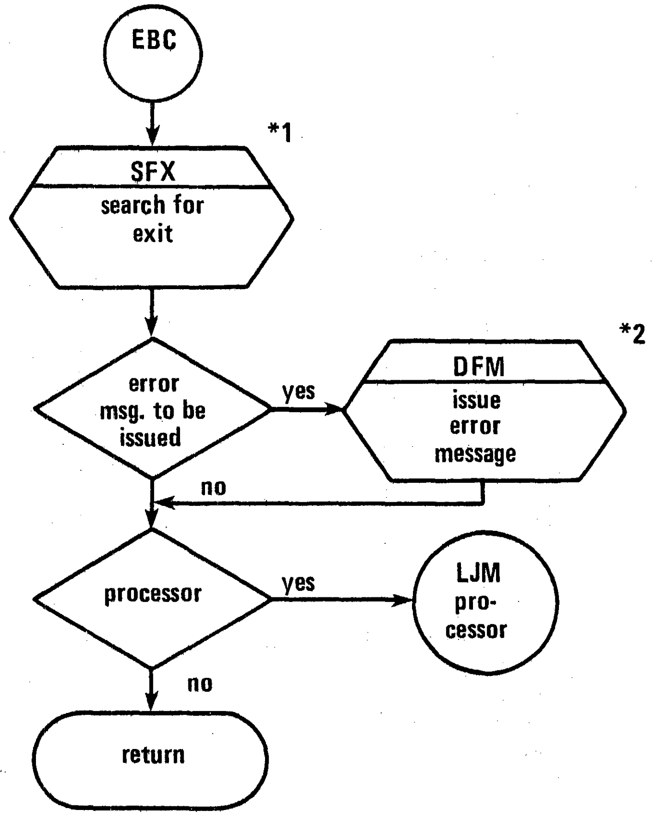
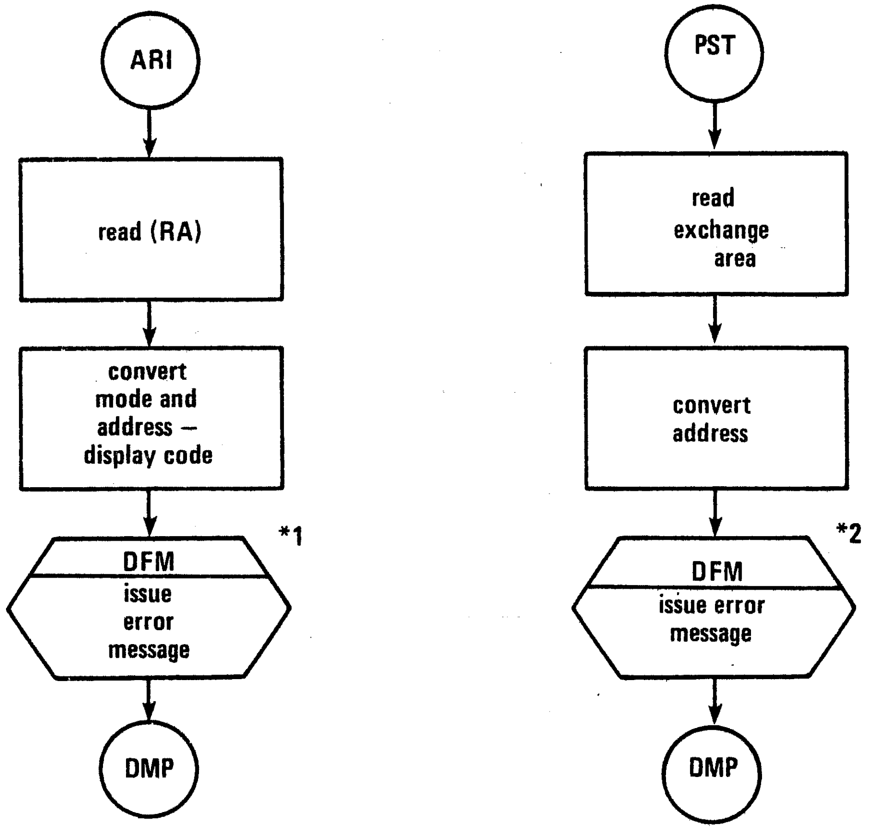


Figure 6-8. 3AB - Process Error Flag (Continued)



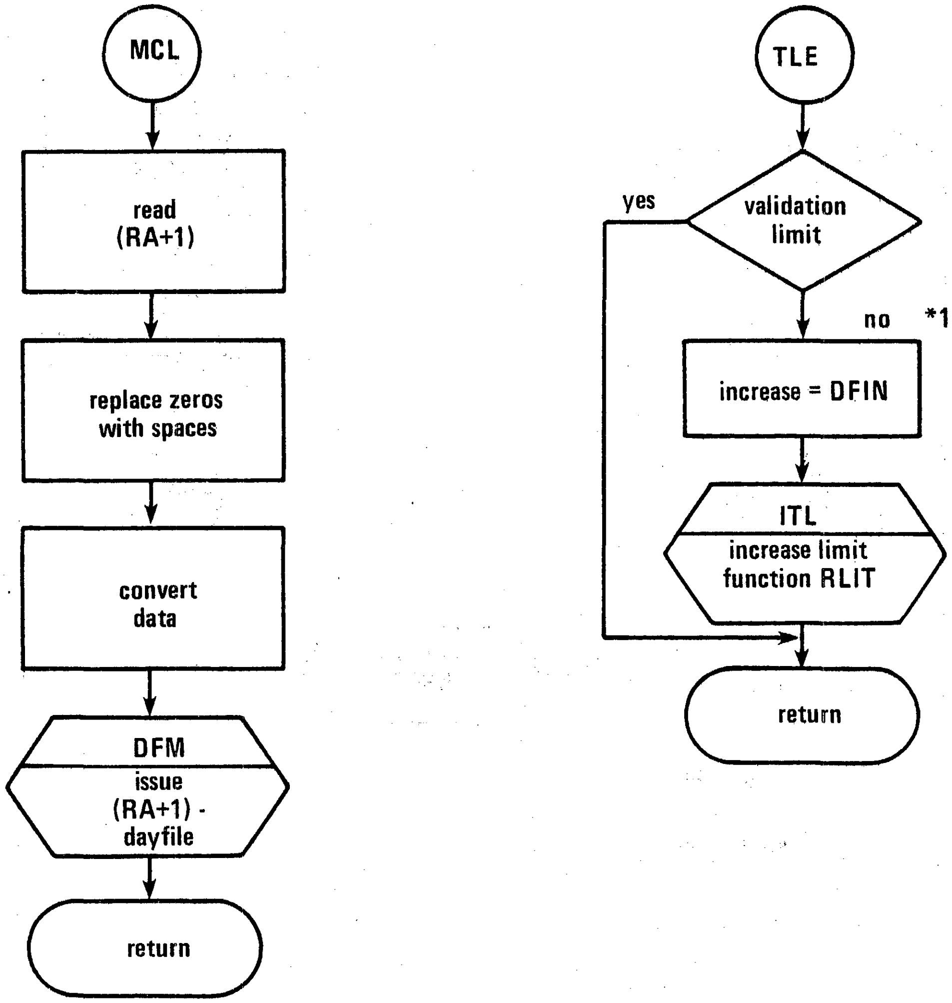
*1 Look for exit statement.
*2 Refer to the EREXIT macro, section 6, volume 2, of the NOS Reference Manual for a description of error flags.

Figure 6-8. 3AB - Process Error Flag (Continued)



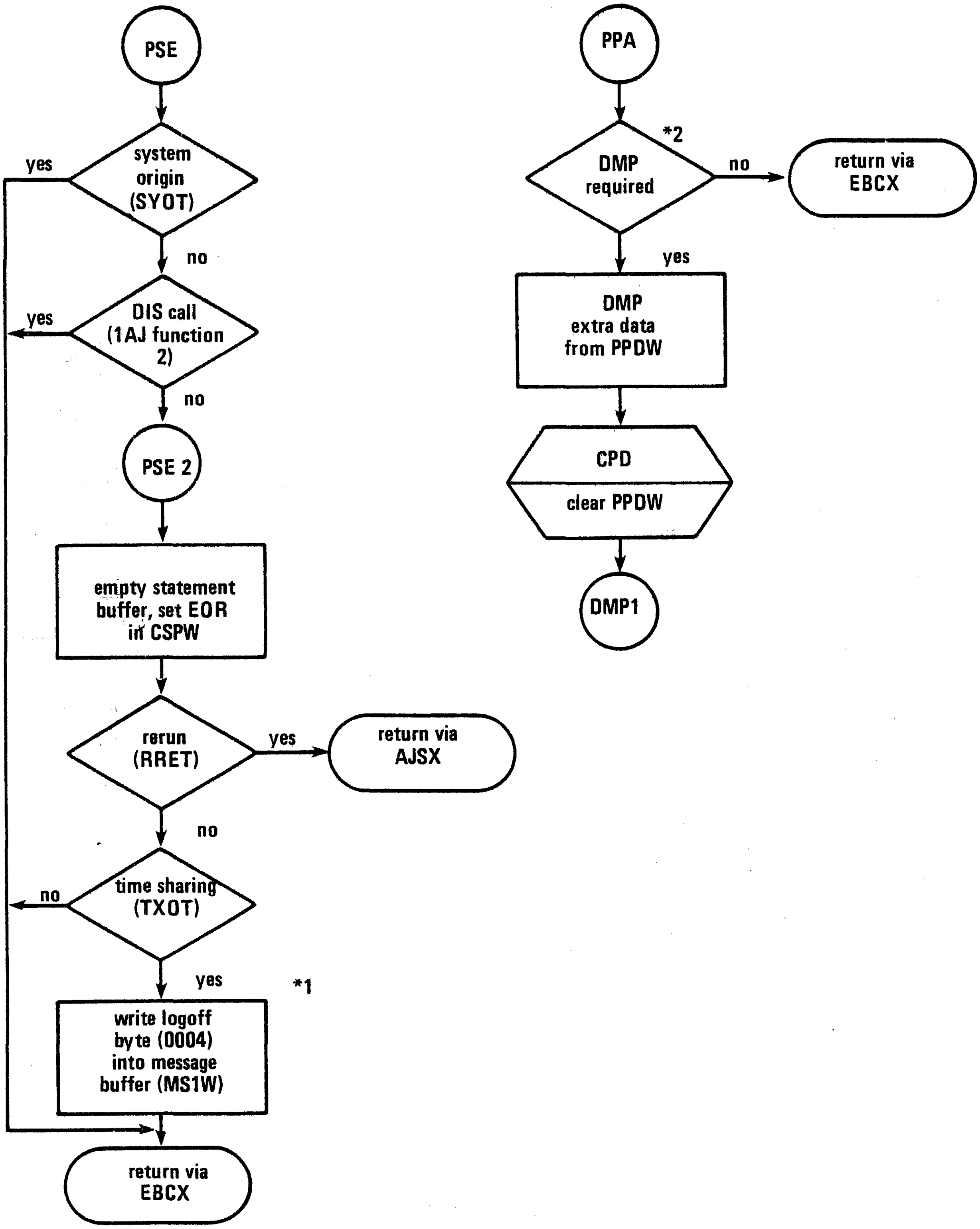
*1 CPU ERROR EXIT (mode) AT (address).
*2 PROGRAM STOP AT (address).

Figure 6-8. 3AB - Process Error Flag (Continued)



*1 Let user finish error processing if possible.

Figure 6-8. 3AB - Process Error Flag (Continued)



*1 Time sharing processings sends contents of message buffer to terminal. Since message buffer has log off byte in it, terminal will be logged off.

*2 Control point area PPDW contains the address of the control point area to dump and number of words to dump.

Figure 6-8. Process Error Flag (Continued)

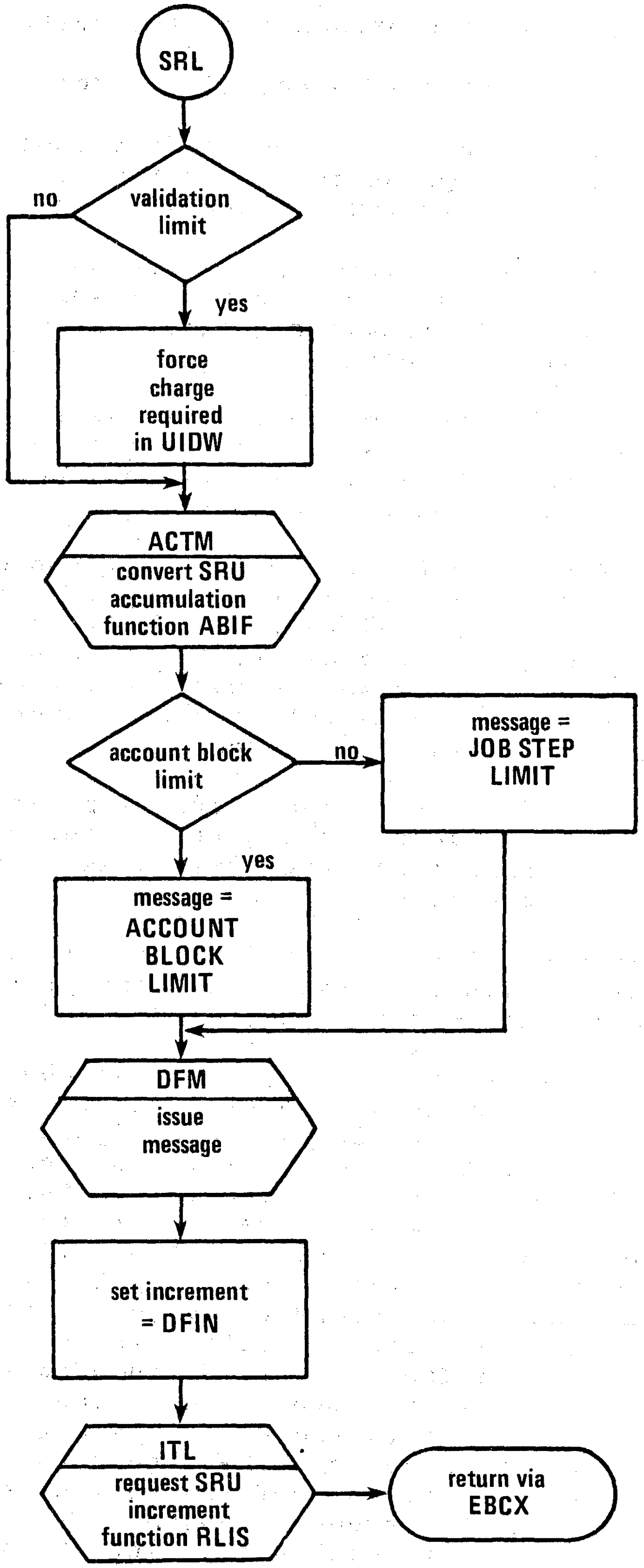


Figure 6-8. Process Error Flag (Continued)

TRANSLATE CONTROL STATEMENT (TCS)

TCS translates control statements in the following manner.

1. Reads statement from one of the following.

    - Control statement in the control point area
    - Message buffer for DIS type programs
    - Central memory location for an executing program

2. Programs loaded from the system have their parameters processed with operating system separator equivalences, unless a *SC SYSEDIT directive was used when entering the program into the system.

    Local file program loads have their parameters processed with product set separator equivalences, as do all programs with *SC specifications, unless a slash (/) is prefixed to the program name.

    For NOS equivalences, delete all embedded spaces, up to the termination character (a period or right parenthesis). Any characters not in the standard FORTRAN set (for example, > ≤ ;) are not allowed in the statement. They may be used in a comment. Arguments are processed such that the separator character is the lower six bits of the argument.

    For product set equivalences, separator characters are +-/=,($. Blanks are treated as separators. All special characters are treated as 4-bit codes in the lower six bits of the argument.

3. Searches a list of special control statement names for a match with the statement being processed. These special names are CTIME, RTIME, and STIME.

4. Extracts the first seven or less characters from the statement up to a separator character and searches the file name table for a file assigned to the control point with this name. If found, the field length is restored if it is different from the amount set by the last RFL statement or macro. If the running or nominal field length is zero, a system defined field length is used as the initial field length. If such a file is found on a mass storage device and is in absolute format, the loader is called to load and execute it. If the file does not reside on mass storage, the job is aborted. If the file is in relocatable format, control is transferred to the CDC CYBER Loader to load and execute the program. The arguments for the program call are extracted from the control statement and stored in the argument region of the job communication area, (RA+2 through RA+n). The CPU is requested to begin execution of the program.

5. Searches the central library (CLD) for a program with the name on the control statement. If such a program is found and contains an RFL= or MFL= special entry point, the field length is set accordingly. Otherwise the field length is set as described in step 4. The requested program is loaded and executed with arguments stored as described previously.

6. If the statement name is a three-character name, the first of which is alphabetical searches the PP library (PLD) for a program of this name. If found, places this name with up to two octal arguments as a PP program request and exits to the program. No change is made in the job field length. This type of request is valid from system origin only or if the caller has system origin privileges and the system is in DEBUG mode.

7. If none of the preceding steps are successful, the statement is declared illegal and the job is aborted.

All control statements, with the exception of CTIME, RTIME, STIME and *comment statements, cause some routine to be loaded or the job to abort.

The following messages are issued by TCS.

Message	Description
CONTROL STATEMENT LIMIT.	Control statements exceed control statement validation limit.
BUFFER ARG. ERROR.	CM address in call is not within the job's field length.
TCS ILLEGAL REQUEST.	TCS called with an illegal request.
IMPROPER VALIDATION.	A validation program (with VAL=) is required.
FORMAT ERROR ON CONTROL CARD.	An error has been detected in the format of the control statement.
SECURE MEMORY, DUMP DISABLED.	A DMP= processor is called following a job step that requires secure memory.
TOO MANY ARGUMENTS.	The number of arguments on the control statement exceeds the amount allowed.
FL TOO SHORT FOR PROGRAM.	54 table MINFL is larger than FL.

FL BEYOND MFL.	Request FL exceeds MFL.
ILLEGAL CONTROL CARD.	The control statement could not be identified by TCS.

A flowchart of TCS is illustrated in figure 6-9.

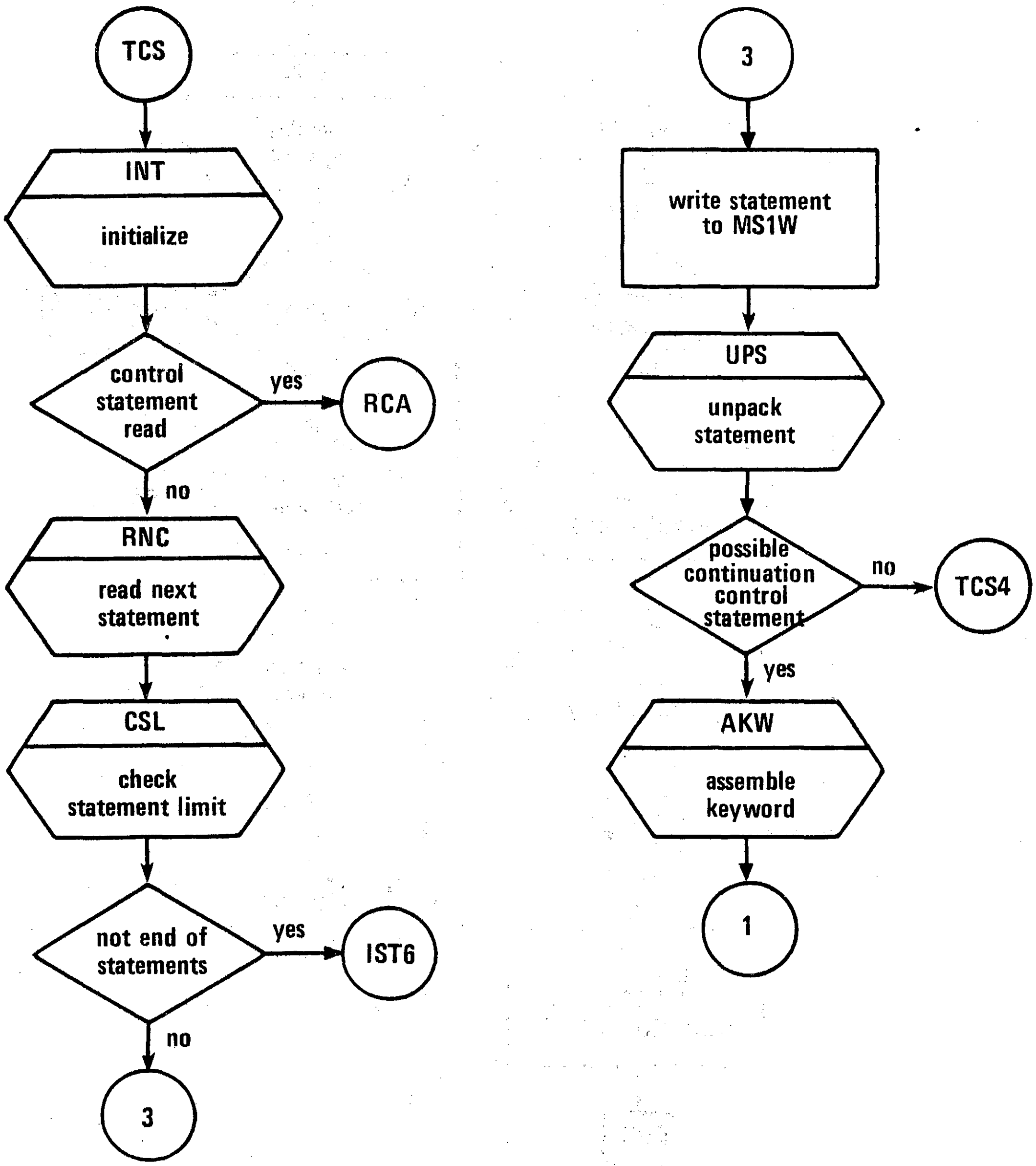


Figure 6-9. TCS - Main Routine

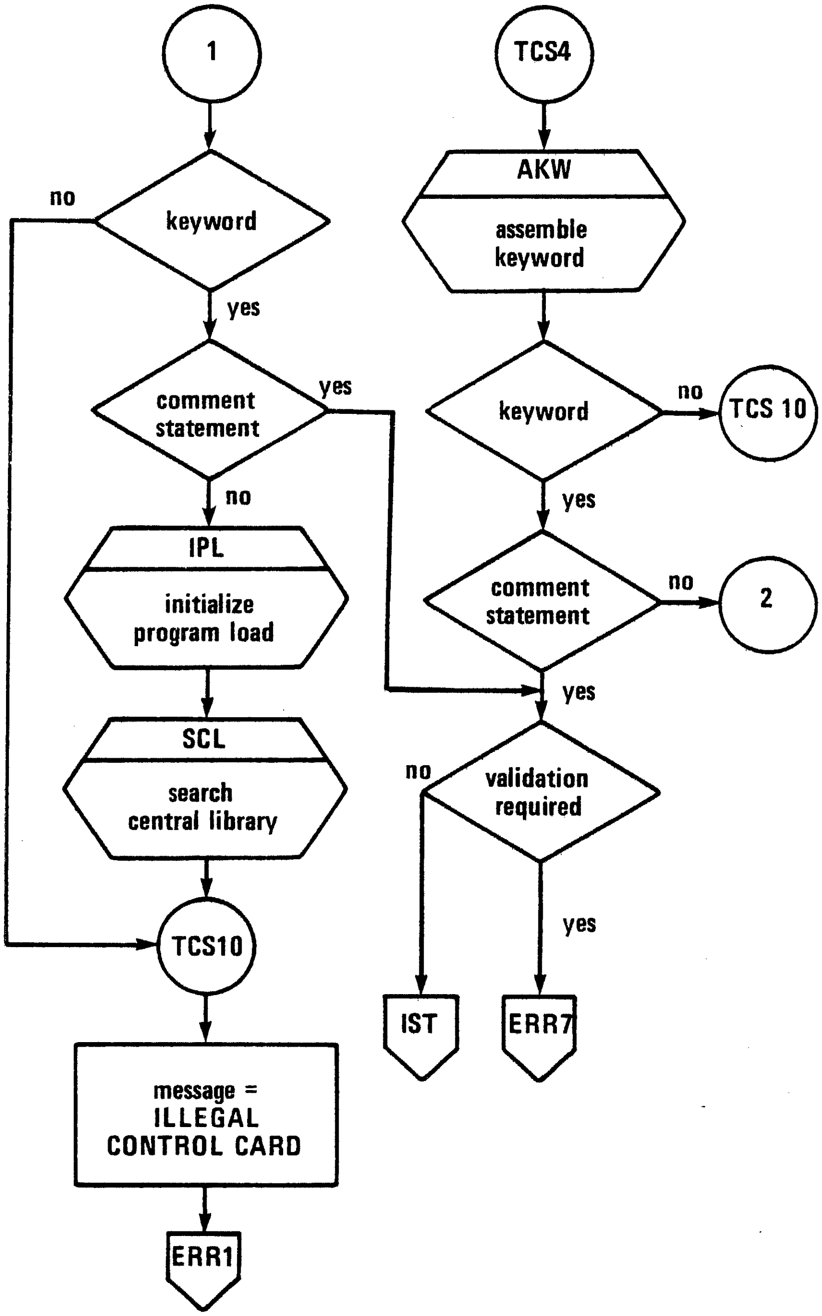


Figure 6-9. TCS - Main Routine (Continued)

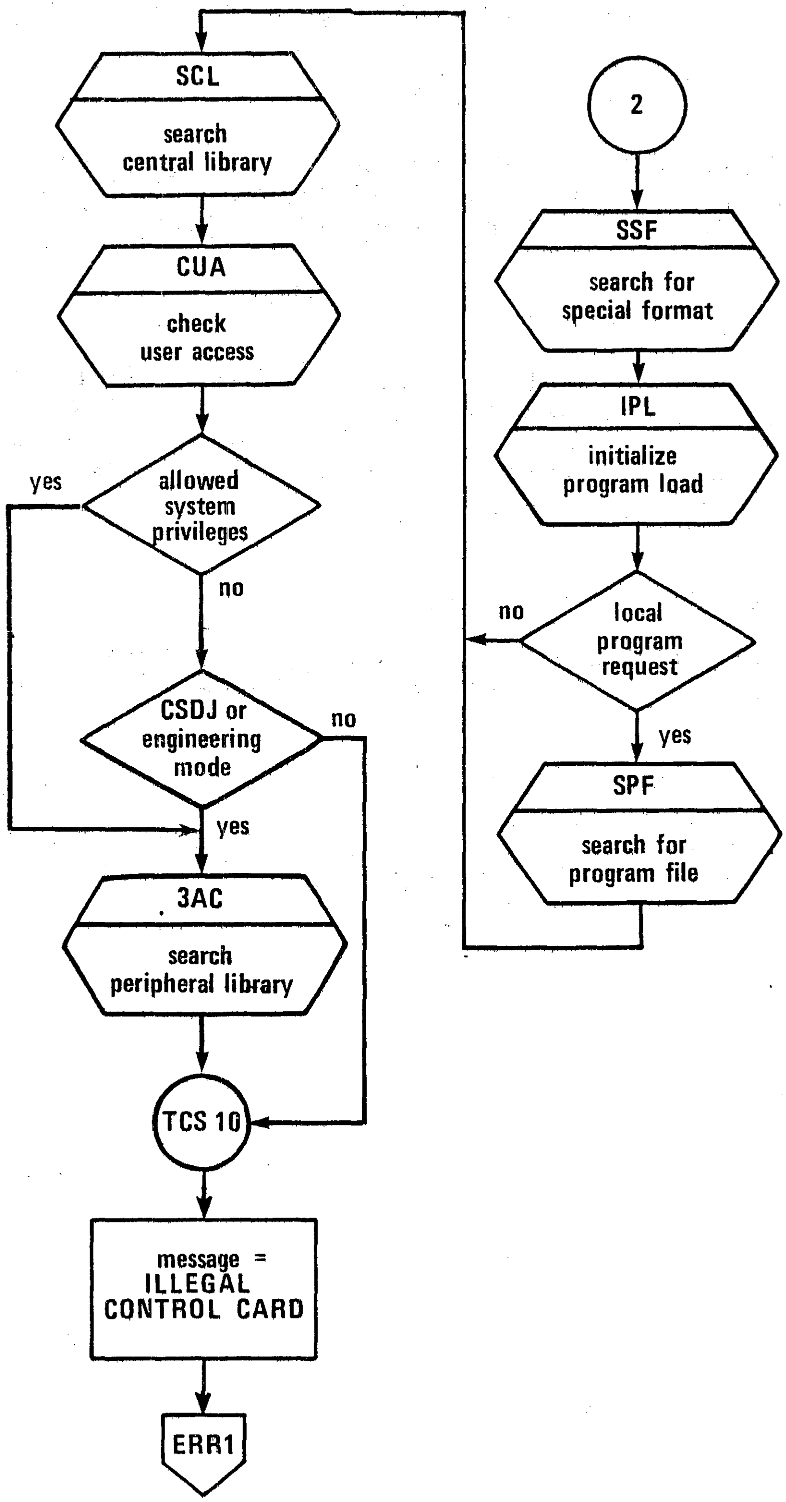


Figure 6-9. TCS - Main Routine (Continued)

The following paragraphs describe the major portions of TCS.

ISSUE STATEMENT TO DAYFILE (IST)

IST issues the control statement and error messages, if any, to the dayfile, updates the control statement pointers in CSPW and advances the job. IST is flowcharted in figure 6-10.

SEARCH FOR SPECIAL FORMAT (SSF)

SSF processes the control statements CTIME, RTIME, and STIME and issues the CPU time (control point area word CPTW), real time (word RTCL) or SRU accumulation (word SRUW) to the dayfile. This is done in 1AJ rather than by a CPU program to eliminate any system overhead in these values.

SEARCH FOR PROGRAM FILE (SPF)

SPF determines if the program requested is local to the control point. SPF exits to subroutine SSF if the file is present.

SEARCH CENTRAL LIBRARY (SCL)

SCL searches the CLD and RCL in an attempt to find the desired program and causes it to be brought to the control point. SCL is flowcharted in figure 6-11.

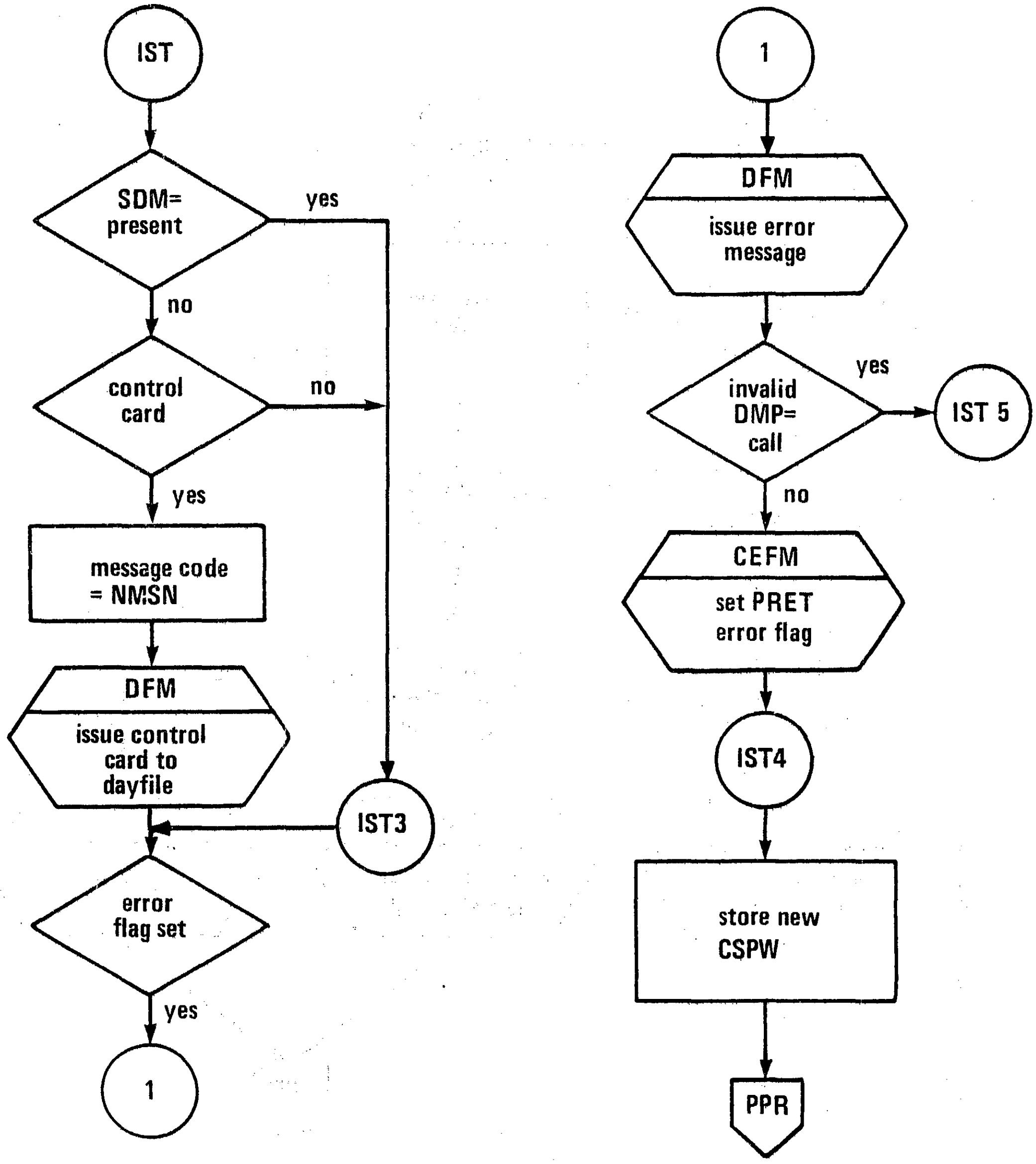


Figure 6-10. IST - Issue Statement

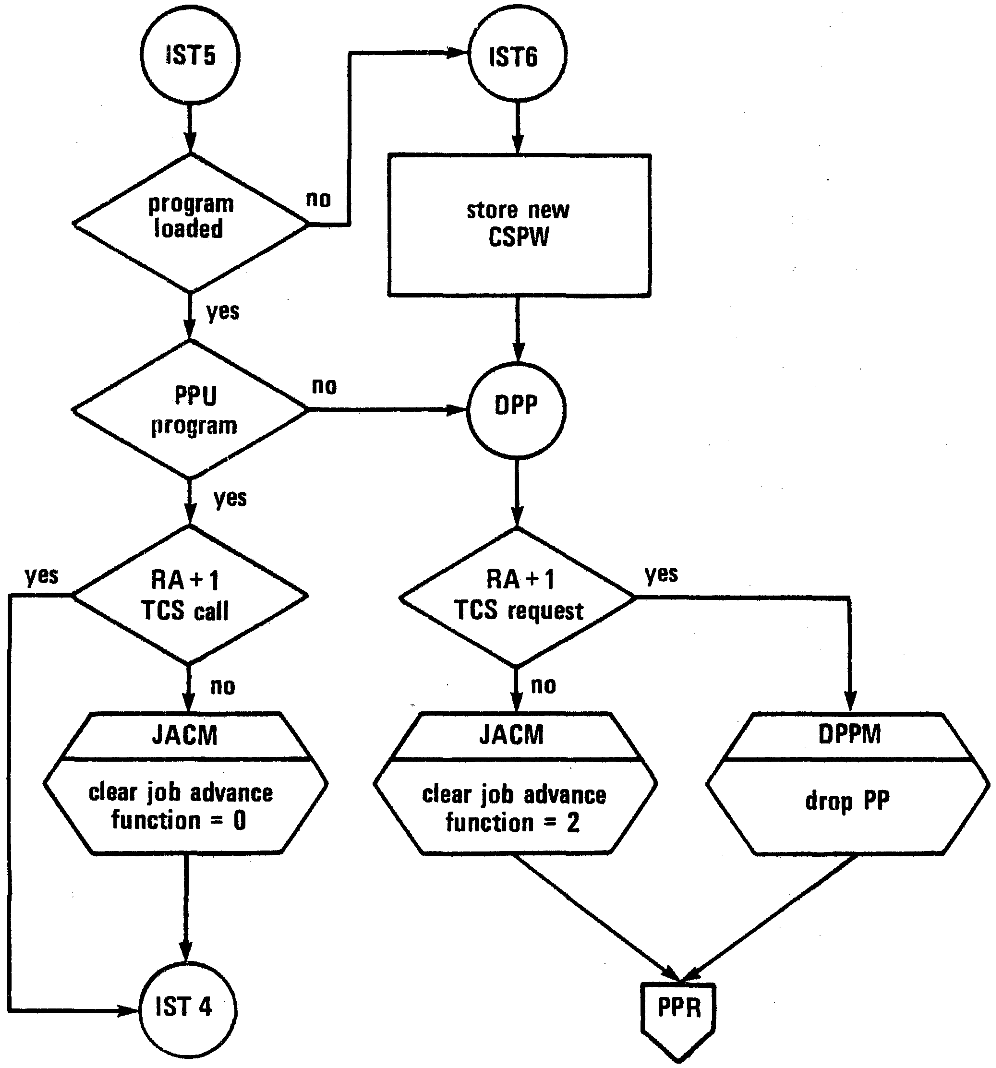
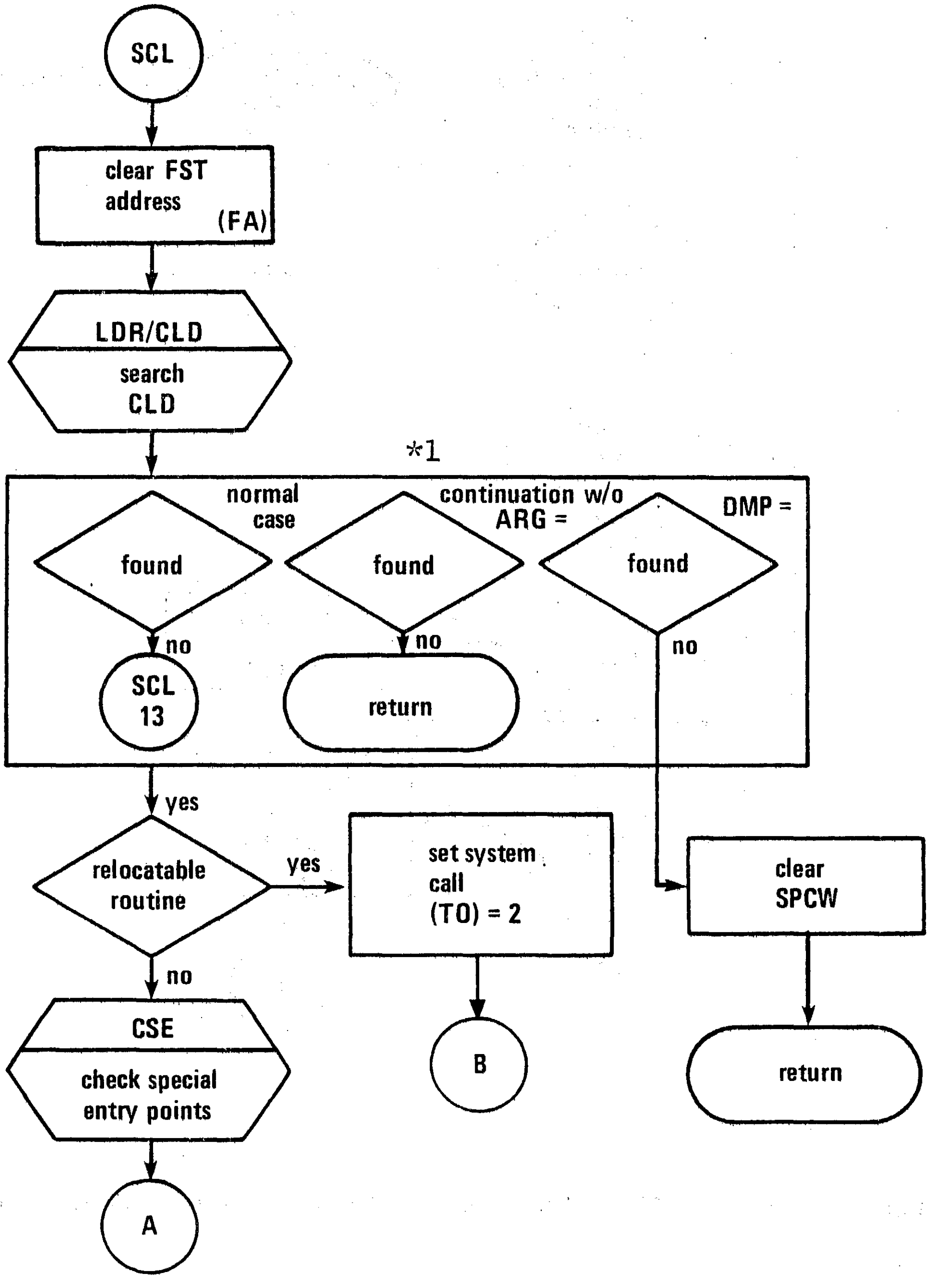


Figure 6-10. IST - Issue Statement (Continued)



*1 Test modified for given cases

Figure 6-11. SCL - Search Central Library

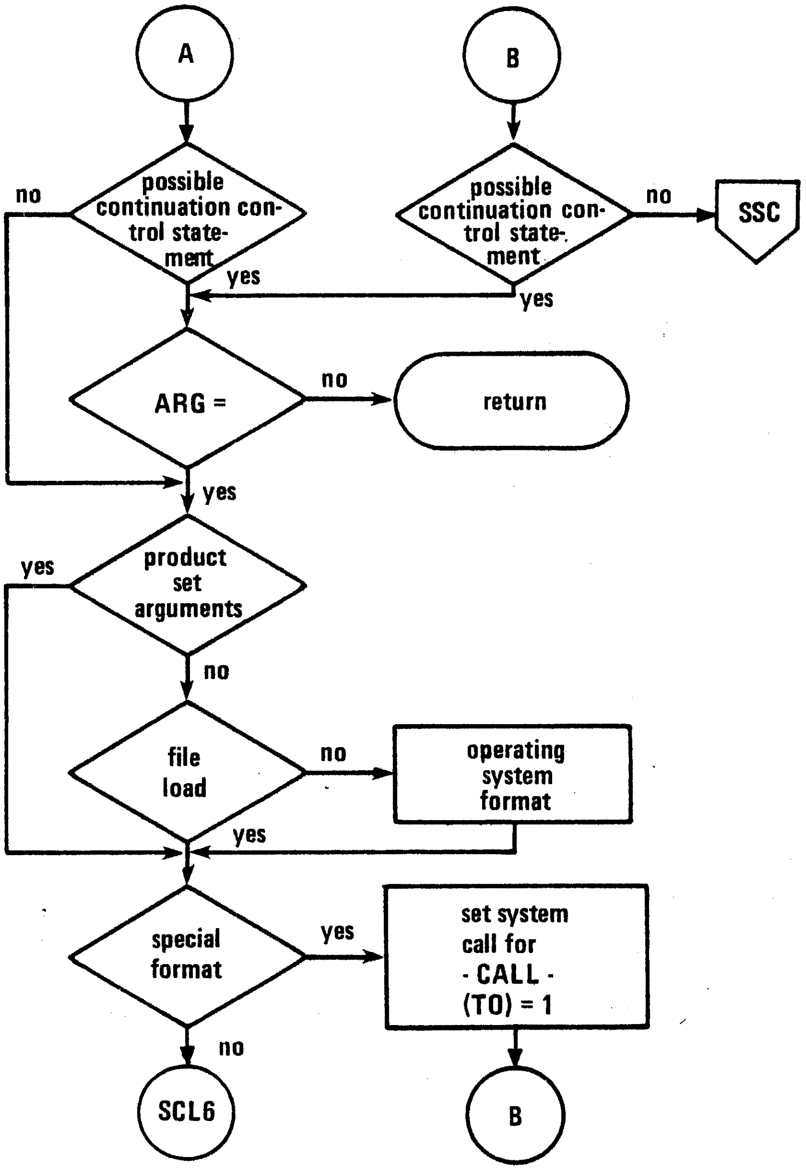
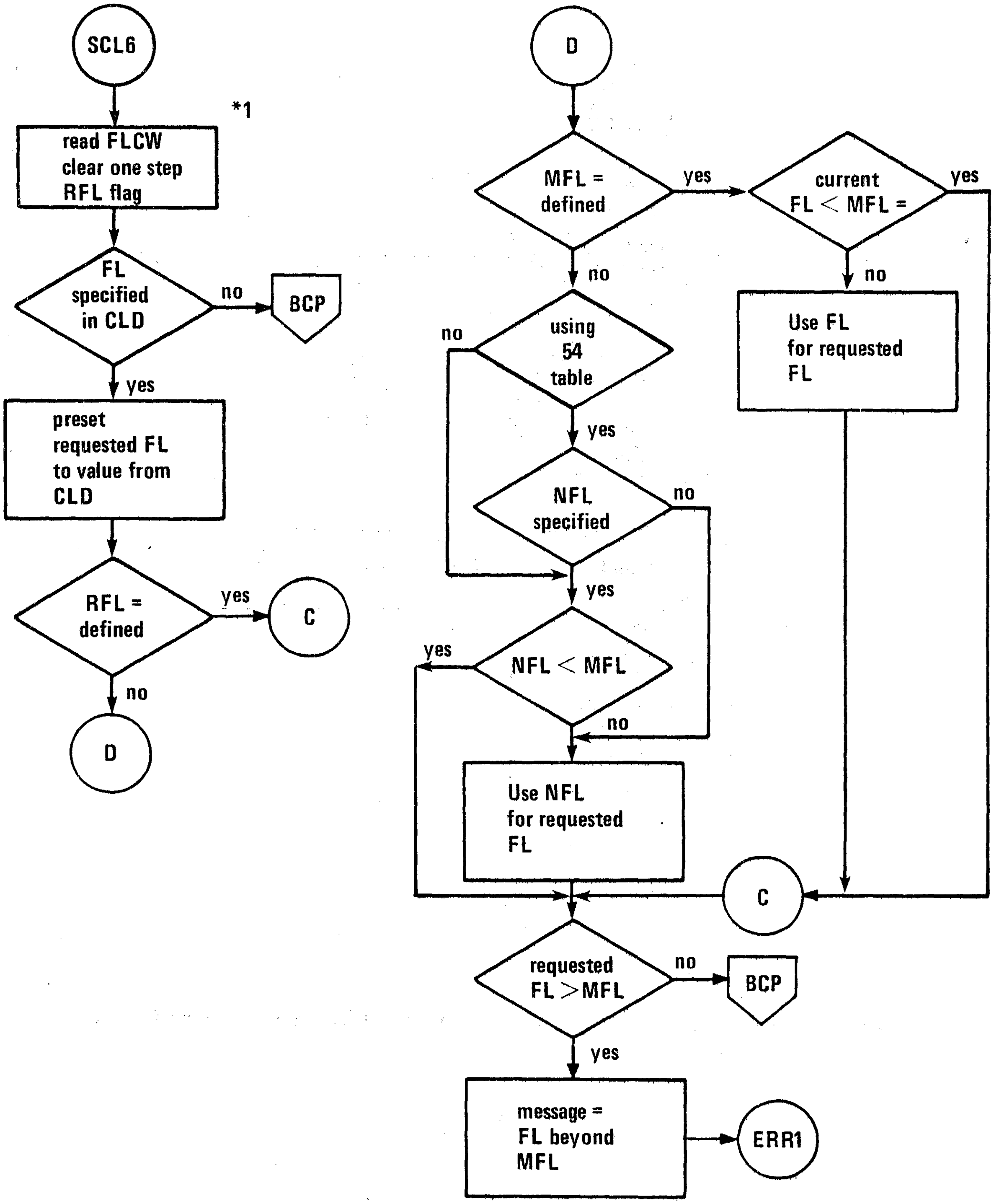


Figure 6-11. SCL - Search Central Library (Continued)



*1 Definitions
MFL = byte 0 of FLCW
NFL = byte 1 of FLCW
FL = current FL, byte 4 of STSW

Figure 6-11. SCL - Search Central Library (Continued)

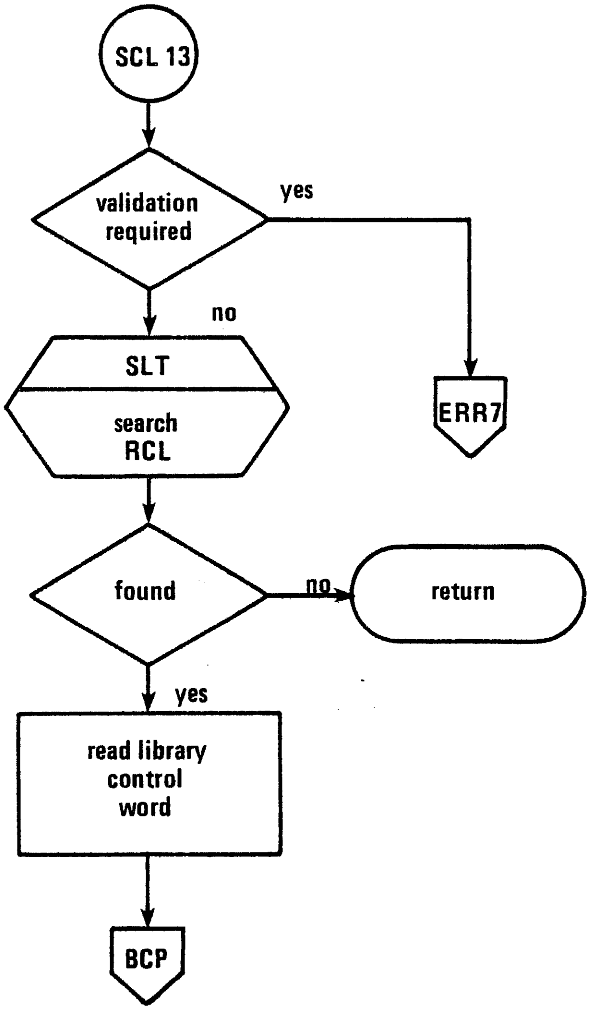


Figure 6-11. SCL - Search Central Library (Continued)

BEGIN CENTRAL PROGRAM (BCP)

BCP obtains storage for the program, initializes the job communication area with the cracked arguments (RA+2 through RA+62B), the number of arguments (RA+ACTR), the control statement (RA+CCDR), the program name (RA+PGNR), the exchange package, and sets FLCW for this job step. The program is loaded into the field length and the job step begun with the RLMM monitor function. BCP is flowcharted in figure 6-12.

ASSEMBLE KEYWORD (AKW)

AKW extracts the program name from the control statement. Appropriate initializations are done for / (use NOS arguments), $ (load from system rather than local file), and * (comment) first characters. Job control language tags are ignored.

ENTER ARGUMENTS (ARG)

ARG processes the arguments on the control statement and sets them in RA+ARGR (RA+2) through RA+62B. The arguments are terminated by a word of binary zeros. Arguments are cracked in operating system format or product set format as directed by the characteristics of the load. If more than 60B arguments are present, the job is aborted with the diagnostic TOO MANY ARGUMENTS. No argument processing is done if the program has an ARG= entry point.

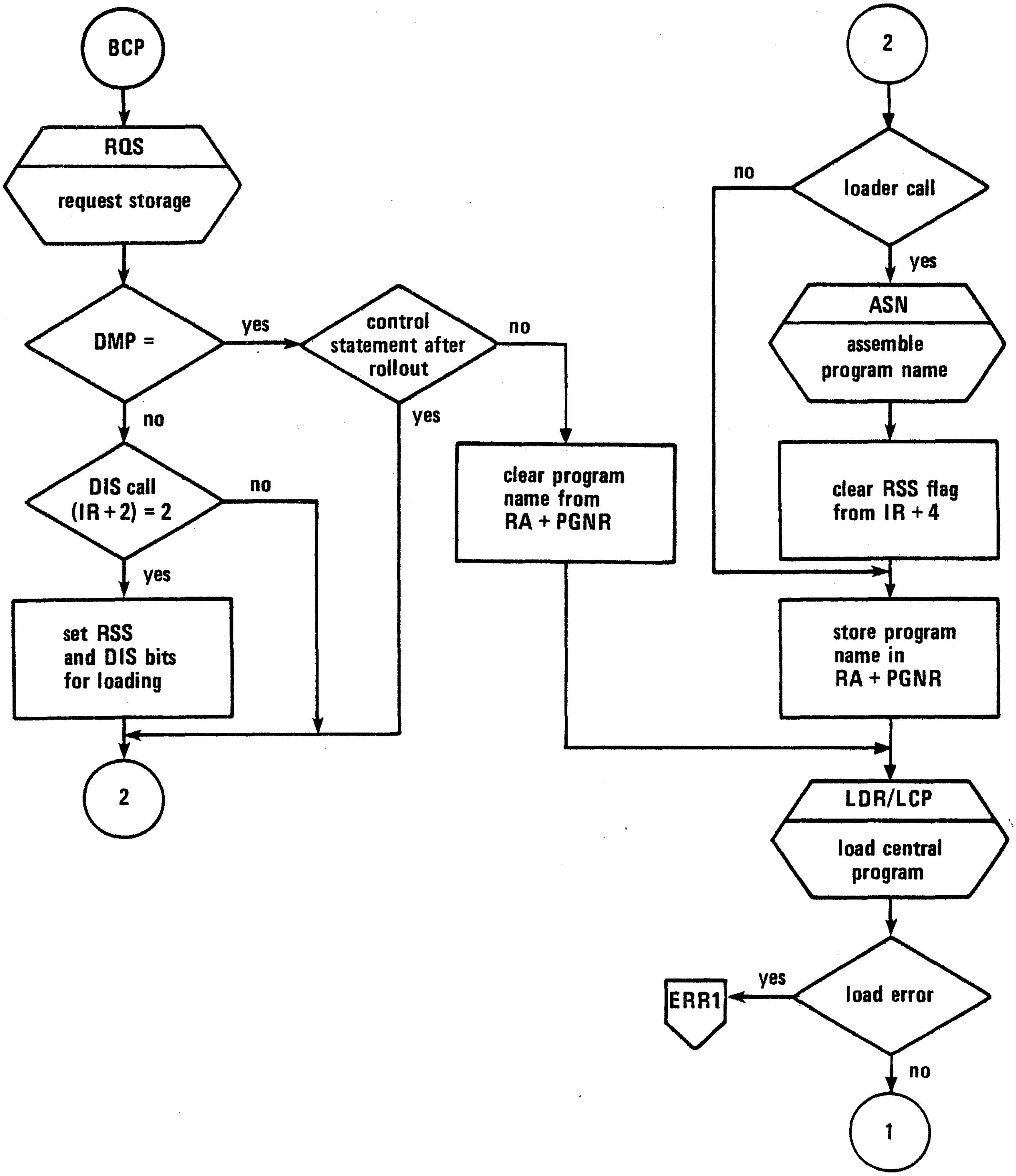


Figure 6-12. BCP - Begin Central Program

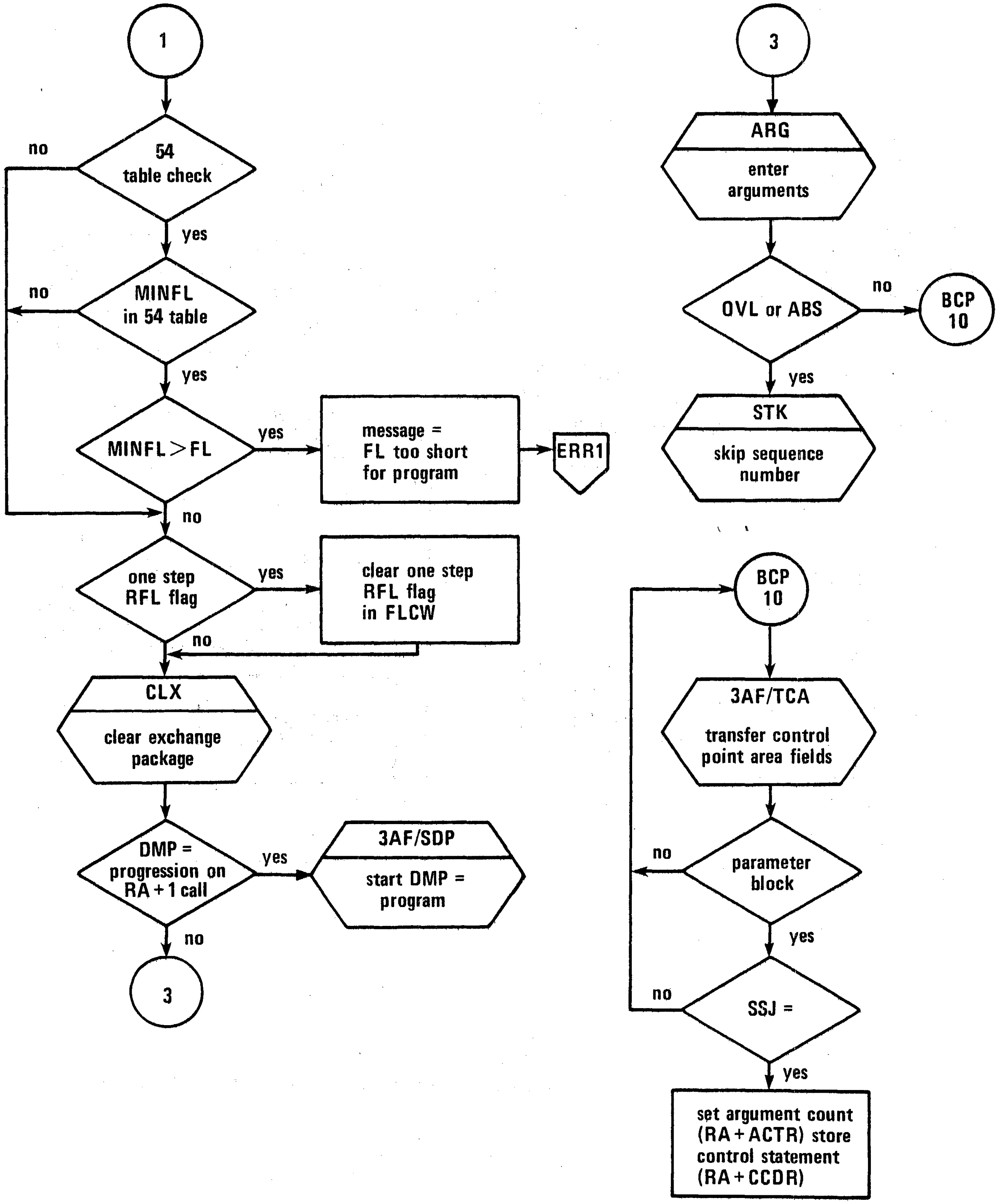


Figure 6-12. BCP - Begin Central Program (Continued)

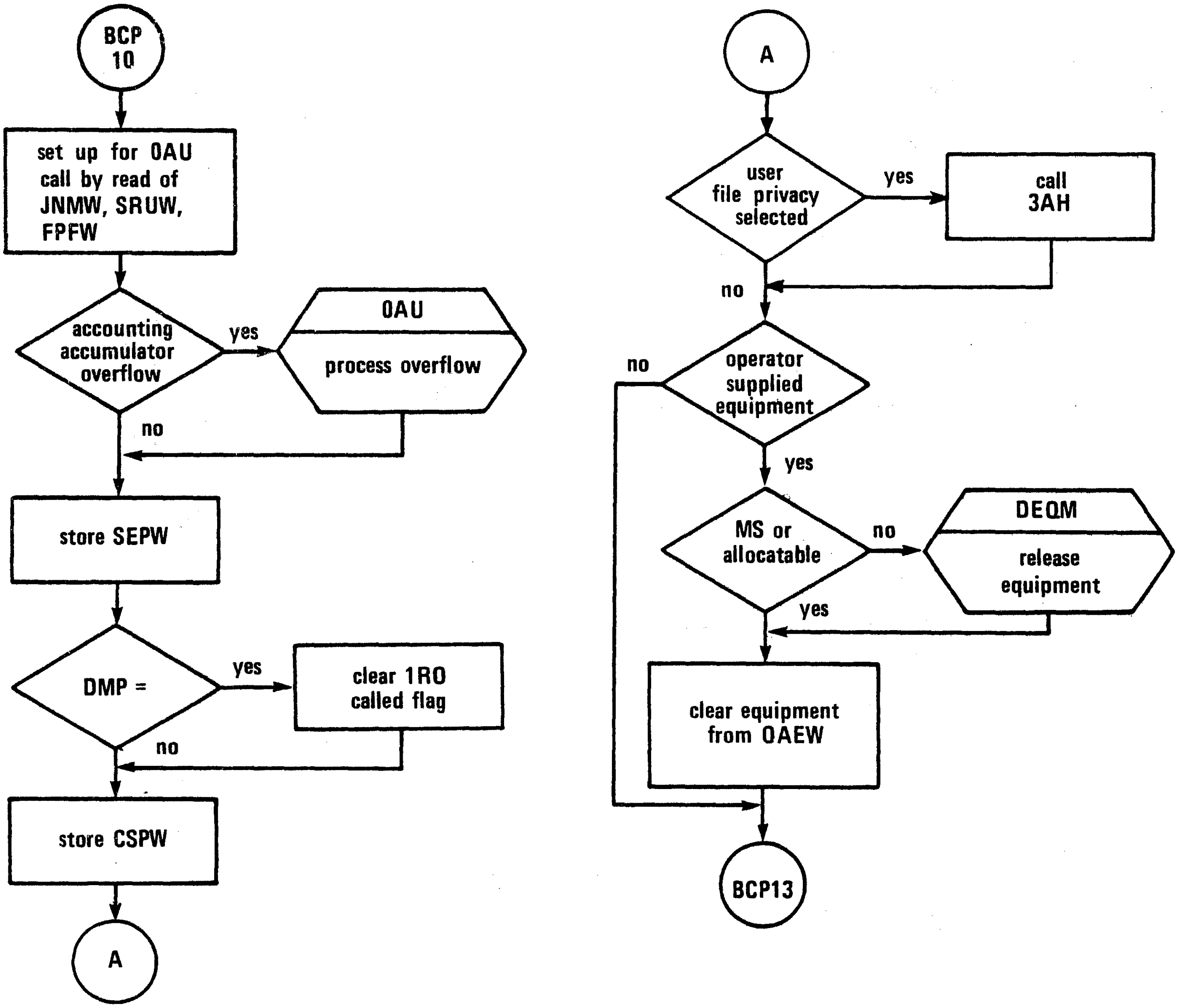


Figure 6-12. BCP - Begin Central Program (Continued)

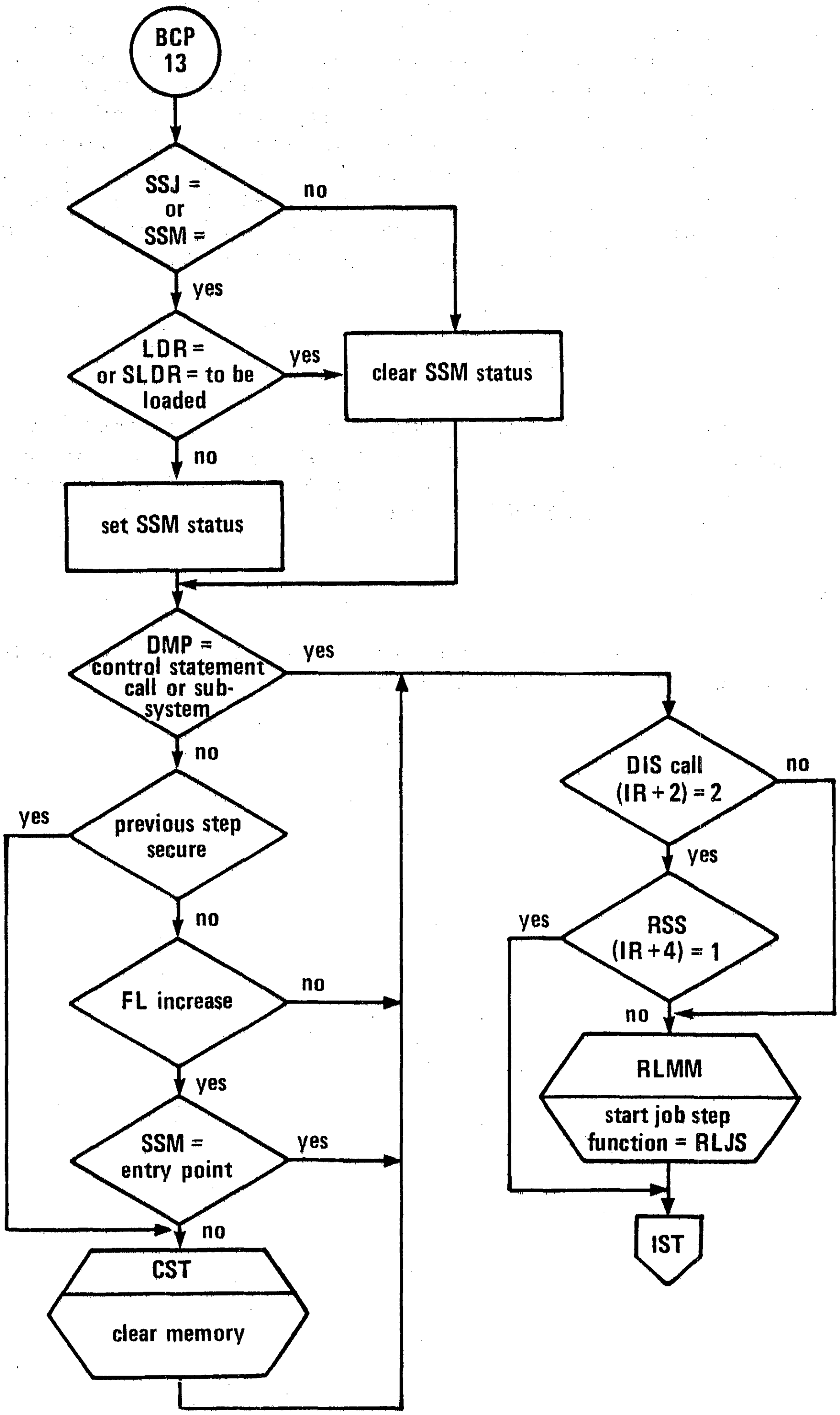


Figure 6-12. BCP - Begin Central Program (Continued)

CHECK FOR SPECIAL ENTRY POINTS (CSE)

CSE enforces the validation required condition (a VAL= entry point must be present if bit 17 is set in control point area word UIDW) and checks for DMP= control statements. If a DMP= control statement (a program with a DMP= entry point being called by a control statement) is encountered, the input register is written into control point area word SEPW, SPCW is set with the upper 18 bits of the input register, the DMP= and no RA+1 clear flags are set, and TERW has the lower 24 bits set with the lower 24 bits of the entry point word (DMP= control information). Routine 1AJ is recalled to process the SPCW request.

CHECK VALID DMP= CALL (CVD)

CVD enforces the secure memory protection required by the previous job step. If the current job step cannot follow the previous step because of secure memory, the diagnostic SECURE MEMORY, DUMP DISABLED is issued.

PROCESS ERROR (ERR)

ERR processes error conditions detected by TCS subroutines. ERR is flowcharted in figure 6-13.

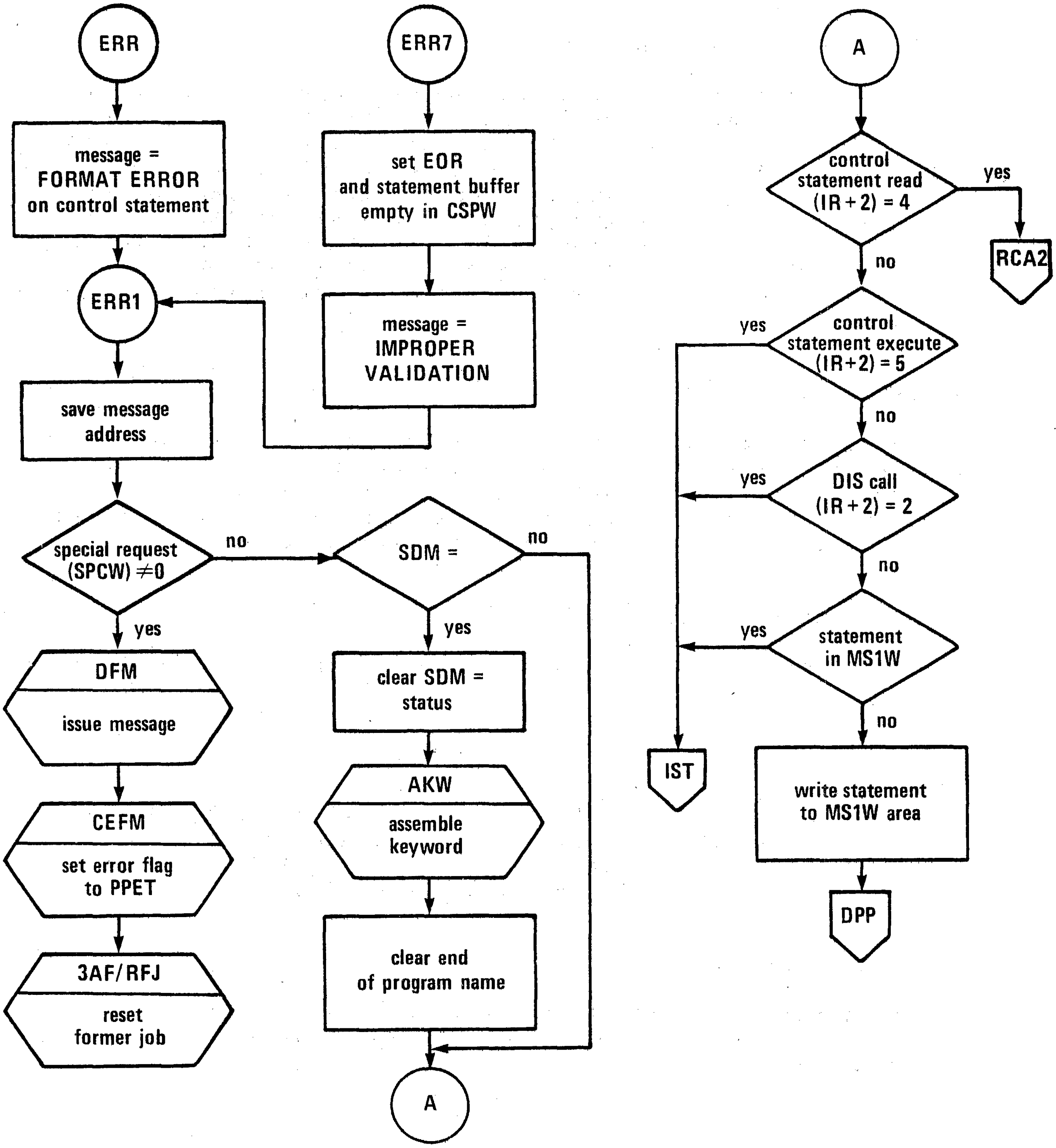


Figure 6-13. ERR - Error Processor

INTERROGATE ONE CHARACTER (IOC)

IOC determines the status of a single character, indicating whether it is a valid separator according to the argument format (operating system or product set) being used.

INITIALIZE PROGRAM LOAD (IPL)

IPL initializes the control point area, loads the copy routines (3AE), and presets the requested field length. The event descriptor is cleared from TERW, pause bit cleared from SNSW, reprieve conditions cleared from EECW, terminal interrupt address cleared from TINW, and console message cleared from MS2W. The requested field length preset is the RFL or the nominal field length (FLCW byte 1) unless it is zero, in which case the minimum of the default field length (SYSDEF=50K) and MFL (FLCW byte 0) is used. The terminal interrupt bit is cleared for TXOT jobs.

REQUEST STORAGE (RQS)

RQS obtains the necessary field length for the program loading. If the field length is not available but pending, the input register is written in RLPW to recall the request. If not available and not pending, the control point is requested to be rolled out (ROCM) with the request recalled through RLPW.

SEARCH LIBRARY TABLE (SLT)

SLT searches a given library for a match on the request program name.

SET SYSTEM CALL (SSC)

SSC makes a library search (SCL) for routines LDR=, CALL, or SLDR=. If the desired routine is not found, an MXFM is done to hang the system and isolate the failure condition. If this hang occurs, it is not likely that any processing can occur.

SKIP TO KEYWORD (STK)

STK reformats the control statement, if ARG= has not been selected, excluding job control statement numbers and special first characters.

TRANSLATE SCOPE PARAMETER (TSS)

TSS equivalences separators for the product set argument format. The equivalences used are as follows.

Value	Character
1	,
2	=
3	/
4	(
5	+
6	-
7	blank
10	;

INITIALIZE DIRECT CELLS (INT)

INT initialize the PP for processing of a control statement request. INT is flowcharted as figure 6-14.

The following subroutines may be overlayed by other 1AJ overlays called by TCS.

ADVANCE TO EXIT STATEMENT (ATX)

ATX checks for the exit flag set in CSSW (bit 58) and if set clears it and searches for an EXIT control statement. If found, it is issued to the dayfile and processing continues by exiting to IST.

CHECK STATEMENT LIMIT (CSL)

CSL decrements the control statement count in control point area word ACLW using monitor function UADM. If the limit has been reached, CSPW is set empty and at EOR and the diagnostic CONTROL STATEMENT LIMIT is issued. CSL also causes a charge increment (IMCS) to be added to the mass storage accumulator in IOAW.

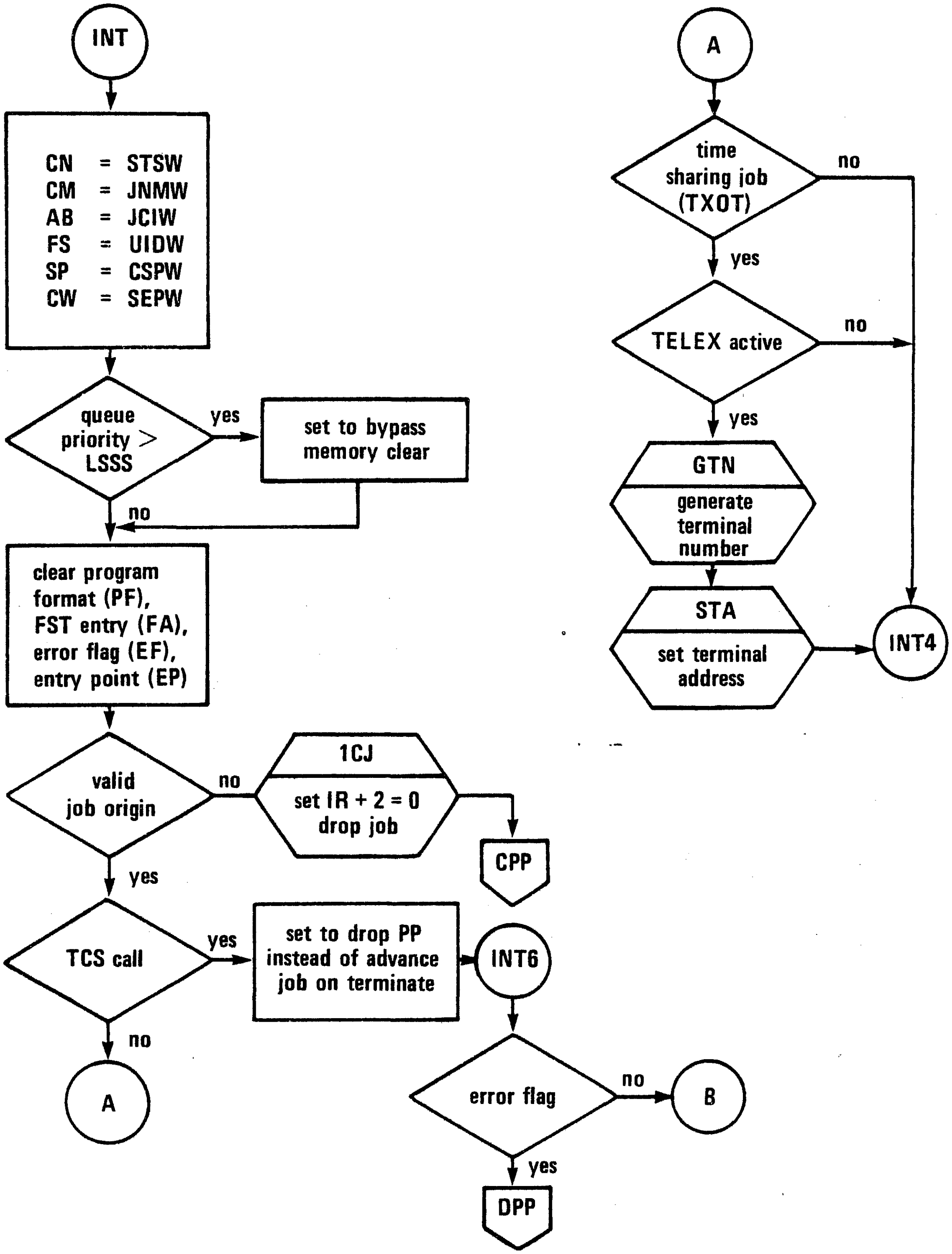


Figure 6-14. INT - Initialize Direct Cells

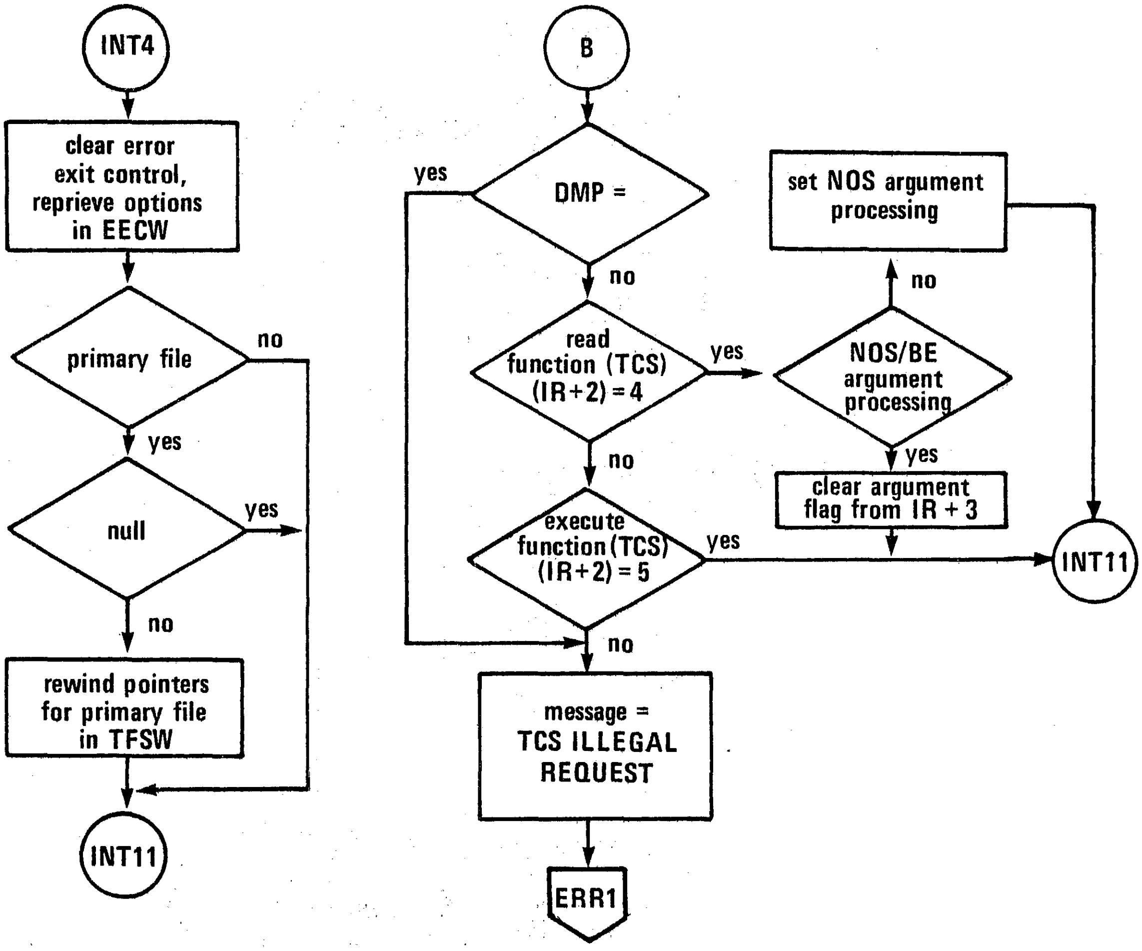


Figure 6-14. INT - Initialize Direct Cells (Continued)

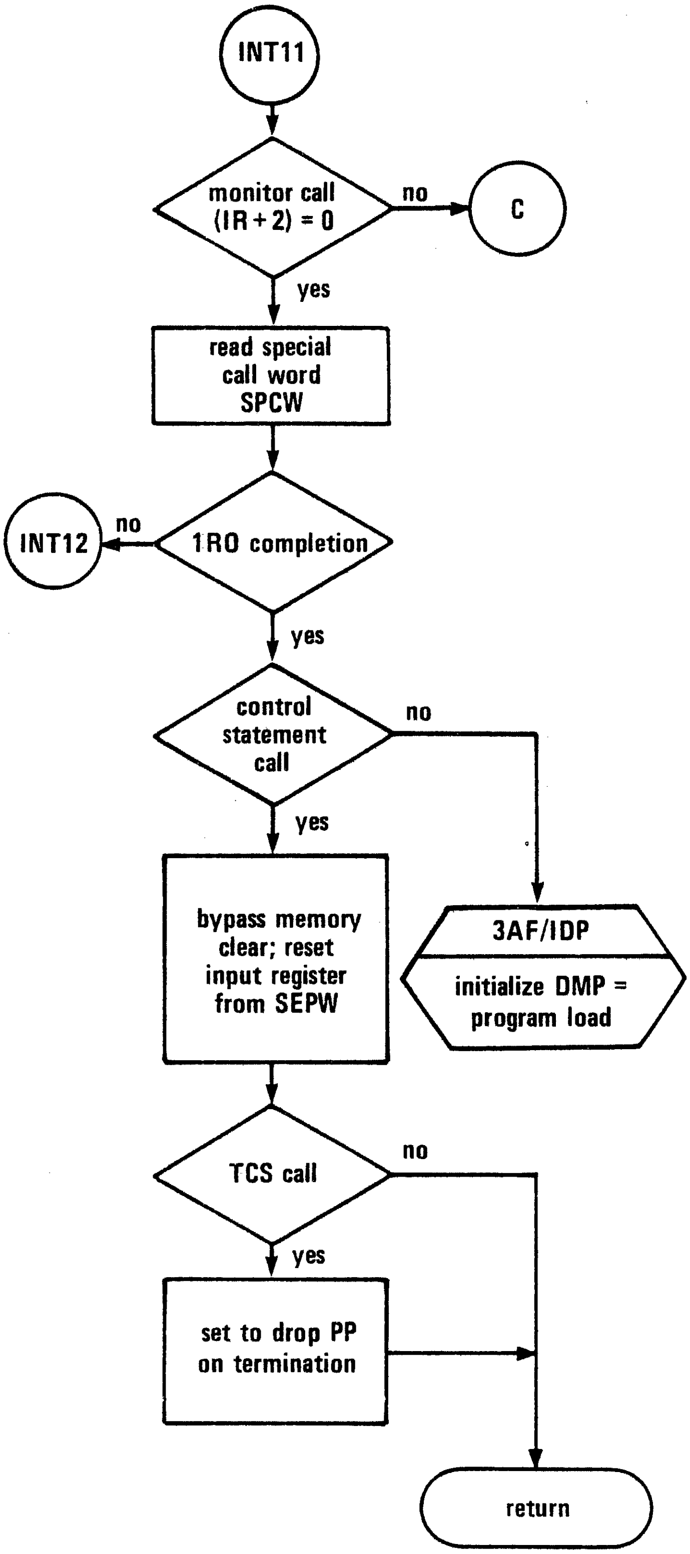


Figure 6-14. INT - Initialize Direct Cells (Continued)

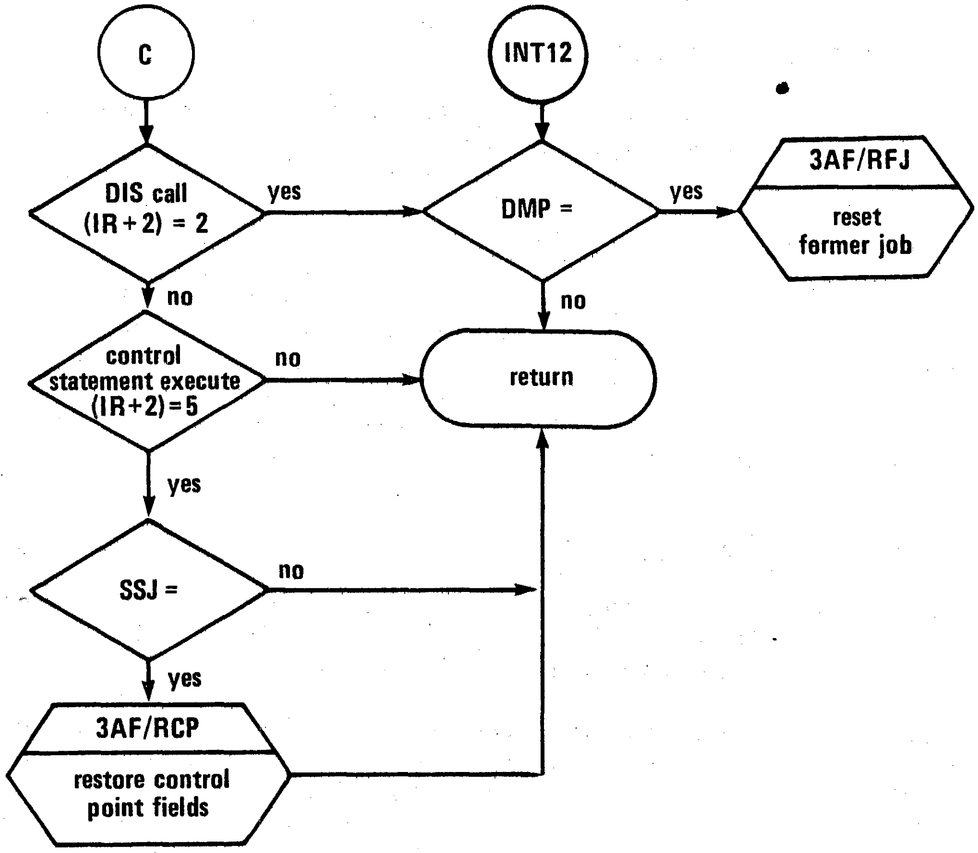


Figure 6-14. INT - Initialize Direct Cells (Continued)

READ CONTROL STATEMENT TO ADDRESS (RCA)

RCA passes the next control statement to the requesting program at a specified address. Arguments are processed (ARG) and the program name set in RA+PGNR. If read with advance is specified, CSPW is updated to indicate that this control statement was processed and the statement is issued to the dayfile (unless SDM= is present).

READ NEXT CONTROL STATEMENT (RNC)

RNC reads the next control statement whether from central memory, MS1W, or the control statement buffer.

SEARCH PERIPHERAL LIBRARY - 3AC

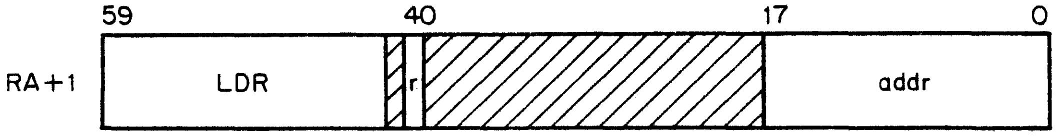
Overlay 3AC is called to search for the program name in the peripheral library. If the routine is found, the routine name and up to two 18-bit octal arguments are written to the PPI's input register and 3AC exits to IST. If the routine is not found, 3AC returns to its caller. Routine 3AC is called only from the TCS main program.

LOAD CENTRAL PROGRAM - LDR

LDR loads absolute overlays in response to CPU program requests. LDR consists of 1AJ overlays LDR, 3AD, and 3AE. TCS uses these overlays, or parts of them, to get routines loaded.

LDR is called with the parameters shown below. If the overlay is being loaded from the system, the central library (CLD) is searched for the overlay. The message OVERLAY NOT FOUND IN LIBRARY is issued if it is not found and the job step is aborted. If the load is not from the system library, overlay 3AD is called to find the overlay or entry point. Once found, subroutine LCP (of the 3AE copy routines) is called to load the program. LDR sets the entry point address into the P register and drops the PP, thus causing execution to begin at that address. The setting of the P address and transfer of control is an option in the LDR request.

The format of the LDR call is as follows.



r	Autorecall if desired
addr	Address of request

Refer to volume 2, section 11, of the NOS Reference Manual for a complete description of LDR requests.

SEARCH FOR OVERLAY - 3AD

Routine 3AD performs an end around search of the overlay (local) file for an overlay of the requested name and level. If the file is not positioned at the beginning of a logical record, random data could possibly be interpreted as a valid overlay header.

The following messages are issued by 3AD.

Message	Description
OVERLAY FILE NOT FOUND.	Requested file is not available.
I/O SEQUENCE ERROR.	Requested file is already busy.
OVERLAY FILE EMPTY.	No data appears in requested file.
OVERLAY NOT FOUND.	Requested overlay is not on file.
FILE NOT ON MASS STORAGE.	The requested file does not reside on mass storage.
ENTRY POINT NOT FOUND.	Requested entry point is not on file.

LOAD COPY ROUTINES - 3AE

Routine 3AE contains subroutines that are used to load central programs into the control point's field length. The individual subroutines are described in the following paragraphs.

LOAD CENTRAL PROGRAM (LDC)

If the load is from a local file, LDC exits to CMS (copy MS resident program). If the load is from the system, the library control word is interrogated to determine where the routine resides. If the routine resides on an alternate residency (ASR) device, control is transferred to subroutine CCM. Control is returned if the ASR device is not ECS or DDP or if a SYSEDIT is active. When control is returned or ASR is not available, LCP requests a system device using monitor function (RSYM). If the program is in a format loadable by 3AE (OVL, ABS, or COS), control is given to CMS; otherwise, LCP returns to its caller with the address of a diagnostic message to issue.

COPY MS RESIDENT PROGRAM (CMS)

CMS reads the program from mass storage into the control point's field length. The job is charged for the load by incrementing the mass storage accumulator in IOAW by the number of sectors transferred and charge IMLL using the UADM monitor function.

SET LOAD PARAMETERS (SLP)

SLP is called at the end of program loading by CMS and CCM to clear error mode and status bits from the exchange package, clear selected areas of the job communication area, and set up communication area words LWPR (last word address of program), FWPR (first word address of program), and LDRR (loader status). The status bits indicating CMU availability, CEJ/MEJ availability and character set mode are set in LWPR, FWPR and LDRR respectively. Parameters for the memory clearing done by TCS are also set by SLP. SLP exits to the caller of LCP.

LOAD CM/AD (ECS) RESIDENT PROGRAMS (CCM)

CCM loads system routines that reside in central memory (as directed by *CM SYSEDIT directives) or on ECS used as an alternate residency device (*AD SYSEDIT directives). The address to load the routine in the control point field length and the ECS position and CM address are passed as parameters on a LCEM monitor function. CPUMTR does the transferring of the program from ECS or CM to the control point. When the loading has completed, control is transferred to SLP.

MASS STORAGE READ ERROR PROCESSOR (MSR)

MSR is entered if a mass storage error is encountered or if a bad ECS load address is encountered. An attempt is made to find an alternate source of the program so that it may be loaded correctly from that device. If none is found, the diagnostic OVERLAY LOST is issued and the job is aborted. If the overlay is being loaded from an alternate device, the attempt to determine a new source to load from will be made with the ASR check disabled If the local was from a local file the diagnostic UNRECOVERED MASS STORAGE ERROR is issued and the job is aborted.

SET PROGRAM FORMAT (SPF)

SPF returns the program format OVL, ABS, or COS and the MINFL from the 54 table (if any).

CHECK PROGRAM FORMAT (CPF)

CPF reads the program file to determine its format. CPF calls SPF. The diagnostic UNIDENTIFIED PROGRAM FORMAT is issued and the job aborted if the program format is not OVL, ABS, or COS.

CHECK SYSEDIT ACTIVITY (CSA)

CSA reads the RPL pointer (low core word RPLP) to determine if a SYSEDIT is active. If SYSEDIT is active, loading from CM or ASR is prohibited.

Dayfile message issued from 3AE include the following.

Message	Description
UNRECOVERED MASS STORAGE ERROR.	An unrecoverable read error has occurred on a load from a local file.
FL TOO SHORT FOR PROGRAM.	Program length is larger than FL.
FLE TOO SHORT FOR LOAD.	ECS block exceeds FLE.
ILLEGAL LOAD ADDRESS.	Load address is less than 2.
UNIDENTIFIED PROGRAM FORMAT.	The file requested to be loaded was not in a recognized format.
OVERLAY LOST.	No alternative path exists to load system routine after an unrecovered write error.

SPECIAL ENTRY POINT PROCESSING - 3AF

Routine 3AF contains subroutines for processing DMP= and SSJ= entry points.

RESTORE CONTROL POINT FIELDS (RCF)

RCF restores the UIDW, ALMW, ACLW, and AACW control point area words, sets the CPU and queue priorities, and drops files with special system IDs (SSID) after a job step which used an SSJ= entry point has completed or aborted. If the SSJ= did not have a block address, only the special ID files are dropped if that option was selected. The SEPW word is cleared in all calls to RCF.

INITIALIZE DMP= LOAD ON RA+1 CALL (IDP)

IDP moves the 20B-word parameter block, if any, from the calling program's field length to the control statement buffer for moving to the DMP= processor when it is loaded. If the calling program is also an SSJ= program, the SSJ block, if any, is moved to the control statement buffer for passing to the DMP= processor if it is an SSJ= processor.

PROCESS SPECIAL PROCESSOR REQUEST (PSR)

PSR formats the call to 1RO to perform the dumping of the calling program's field length on a DMP= call. The dump active and 1RO called flags are set in SPCW, the DMP= parameter set in TERW, and the caller's field length set in PPDW. The DMP= function code (1) is set in IR+2, the DMP= parameter in IR+3 through IR+4, and 1RO is loaded into this PPU.

RESET FORMER JOB (RFJ)

RFJ formats the call to 1RI to reload the calling programs field length after a DMP= processor has completed or aborted. The SPPR parameter block is moved from the DMP= processor's communication area to the control statement buffer for restoring into the caller's field length when it is reloaded by 1RI. The DMP= function code (1) is set in IR+2 and 1RI is loaded into this PP.

START-UP DMP= JOB (SDP)

SDP transfers the parameter block from the control statement buffer to RA+SPPR in the communication area. If SSJ= values are also being passed, they are moved from the control statement buffer to the SSJ= block in the DMP= processor's field length by subroutine TCA. The CPU is requested by an RCPM function and the PP is dropped, thus transferring control to the DMP= processor.

SET PRIORITIES (SPR)

SPR sets CPU and queue priorities as well as the time limit associated with SSJ= processing.

TRANSFER CONTROL POINT AREA FIELDS (TCA)

TCA transfers control point area values to the SSJ= block in the program's field length, if an SSJ= block address has been specified. The CPU priority, queue priority and job step time limit, UIDW, ALMW, ACLW, and AACW are passed to the CPU program. The limit controls (ALMW, ACLW, and AACW) are then set to unlimited in the control point area and UIDW is set for the system user number and index. If any time limit, CPU priority, or queue priority were specified in the SSJ= block, they are set for the control point by a call to SPR.

TERMINATION PROCESSING - 3AG

Routine 3AG is called to terminate processing when the program has connections to the system control point (SCP) facility.

SEND RESPONSE TO SUBSYSTEM (SRS)

SRS reads the subsystem control word (SSCW) to examine the wait response and long term connection indicators for each subsystem. If any indicators are set for a particular subsystem, a message is sent to that subsystem informing it of the user end/abort.

If a subsystem aborts, all user jobs connected to the subsystem will have the error flag SSET (subsystem aborted) set. A system message is issued to the subsystems to which the user connected notifying them of the user abort.

CHECK SUBSYSTEM CONNECTION (CSC)

CSC determines whether a job is in the queue (type ROFT or TEFT) with connections set (bit 5 of FNT set), or a job is at a control point with connections set (SSCW word has appropriate connection indicators set).

CALCULATE SUBSYSTEM INDEX POSITION (CSP)

CSP uses the subsystem index to determine the position within the subsystem control word of the long term connection and wait response indicators (3 groups of indicators per byte).

END USER JOBS (EUJ)

EUJ ends all user jobs connected to a particular subsystem. The FNT is searched for all jobs with connections to this subsystem, as determined by CSC, and writes the subsystem index in the system sector and sets the job's priority to SSPS (subsystem aborted priority).

Routine 3AG issues the following messages for display from MS2W of the subsystem control point.

Message	Description
UCP ABORT.	User control point end/abort.
SUBSYSTEM BUSY.	Subsystem is unable to receive reply.
TERMINATION PROCESSING	Indicates subsystem end/abort processing.

USER FILE PRIVACY PROCESSING - 3AH

Routine 3AH returns all files associated with the job except those with user file privacy id set (UPID).

COMPLETE JOB - 1CJ

Routine 1CJ performs all the following job termination procedures.

- Release storage.
- Release assigned equipment.
- Release any common files used by the job.
- Drop any scratch files used by the job.
- Release all output files to output queue.
- Record in the account and control point dayfiles the accumulated system resource usage for the current account block. These resources consist of application units, permanent file usage, magnetic tape usage, mass storage usage, CPU time, and system resource units (SRU).
- Copy the control point dayfile to the end of the print file.
- Update resource files.
- Clear the control point for usage by the next job.

The following accounting messages are issued to both the user's dayfile and the account dayfile.

Message	Description
UEAD, XXXXX.XXXKUNS.	Application units (kilo-units)
UEPF, XXXXX.XXXKUNS.	Permanent file usage (kilo-units)
UEMT, XXXXX.XXXKUNS.	Magnetic tape usage (kilo-units)
UEMS, XXXXX.XXXKUNS.	Mass storage usage (kilo-units)
UECP, XXXXX.XXXSECS.	Accumulated CPU time (seconds)
UESR, XXXXX.XXXUNTS.	Accumulated SRUs (units)

The following messages are issued only to the account dayfile.

Message	Description
AEUN, usernum.	Job terminated and input file requeued (RERUN)
AUSR, XXXXX.XXXUNTS.	Accumulated SRUs (units) not updated into project profile file (PROFILA); this message indicates that 1CJ was unsuccessful in making its OAU call to update PROFILA.
AUSR, 219902.325UNTS	SRU overflow detected

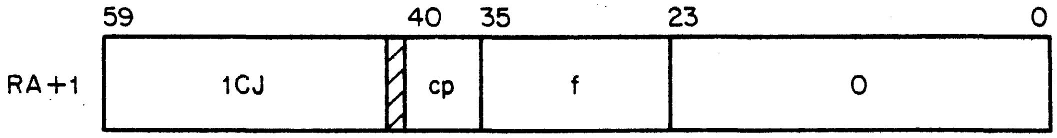
The following messages are issued to the system dayfile and user's dayfile:

Message	Description
JOB RERUN.	Named job is in RERUN
1CJ ARGUMENT ERROR.	Incorrect parameter in call

The following message is issued only to the ERRLOG dayfile:

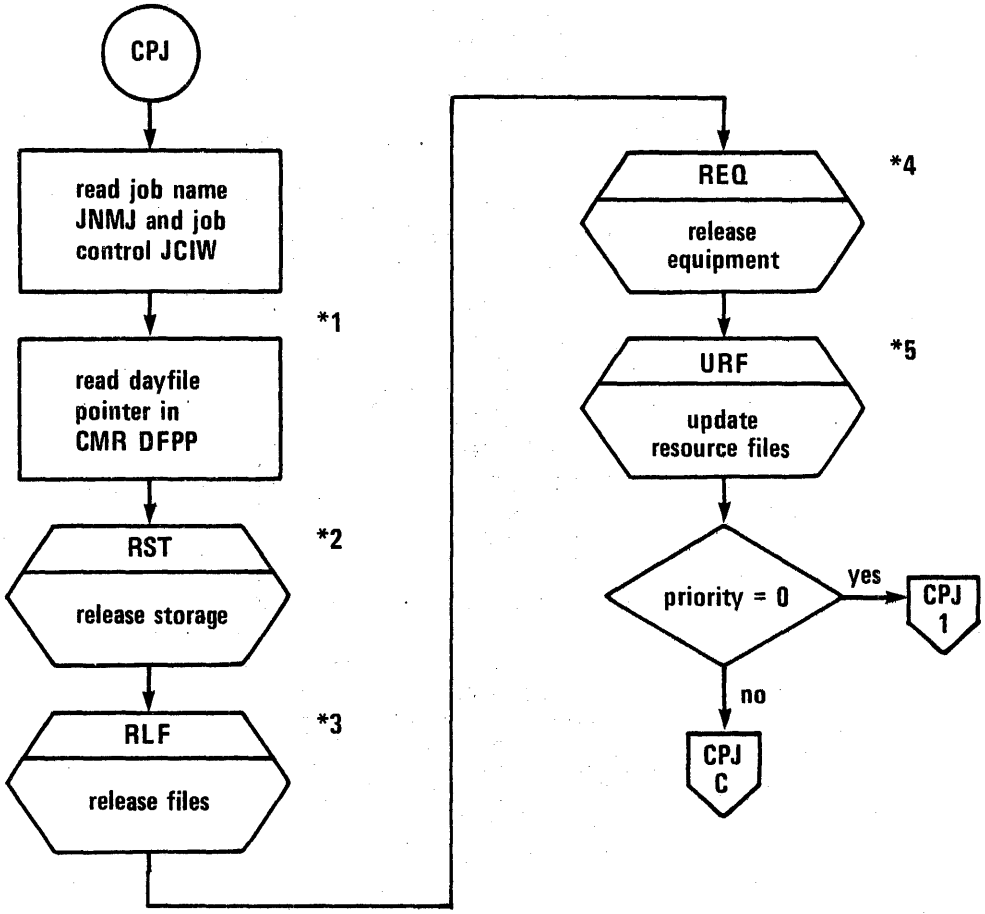
Message	Description
EQxx, OUTPUT LOST.	Write errors occurred adding dayfile to output file or dumping output buffers

The call to 1CJ has the following format:



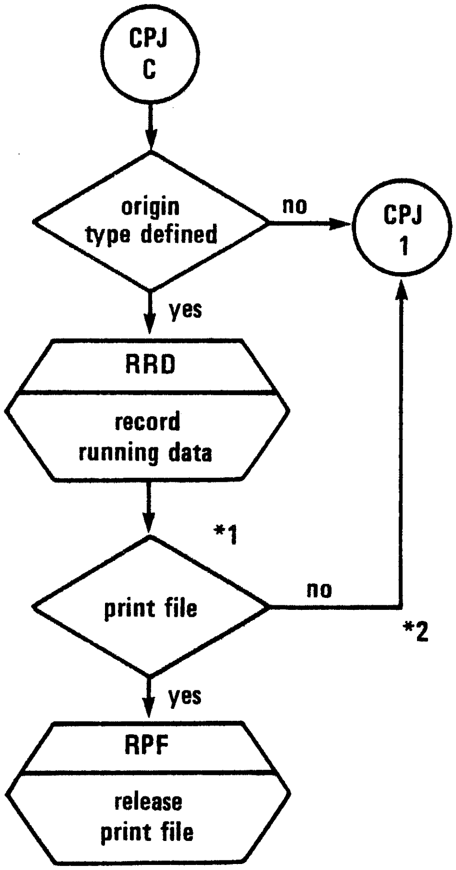
cp Control point
f Function:
  0 Normal completion
  1 Rerun job

Routine 1CJ is flowcharted in figure 6-15.



- *1 Used at CPJ1.
- *2 Release all memory for this control point. RST issues RSTM request for zero words of memory.
- *3 Close and clear all FNT/FST and drop all unused tracks for this control point (file OUTPUT will be checked later and if exists taken care of in RPF). Punch files are disposed to the output queue.
- *4 Release all equipment assigned to this control point.
- *5 Use ORF to clear the entry in RSXDid for this control point.

Figure 6-15 1CJ - Complete Job



*1 Issue UEAD, UEMT, UEMS, UEPF, UECP, and AESR accounting messages.

*2 Set print file name to job name, set type to PRFT, append dayfile to end of file, and release file from this control point.

Figure 6-15 1CJ - Complete Job (Continued)

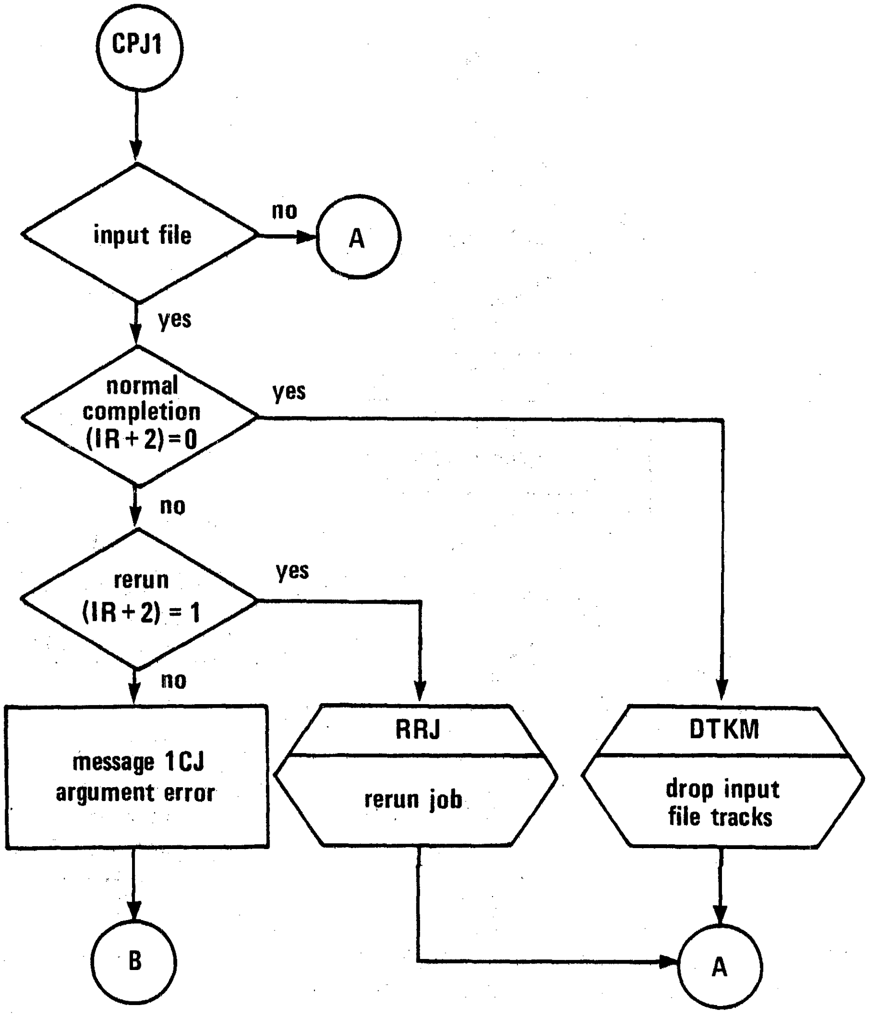


Figure 6-15 1CJ- Complete Job (Continued)

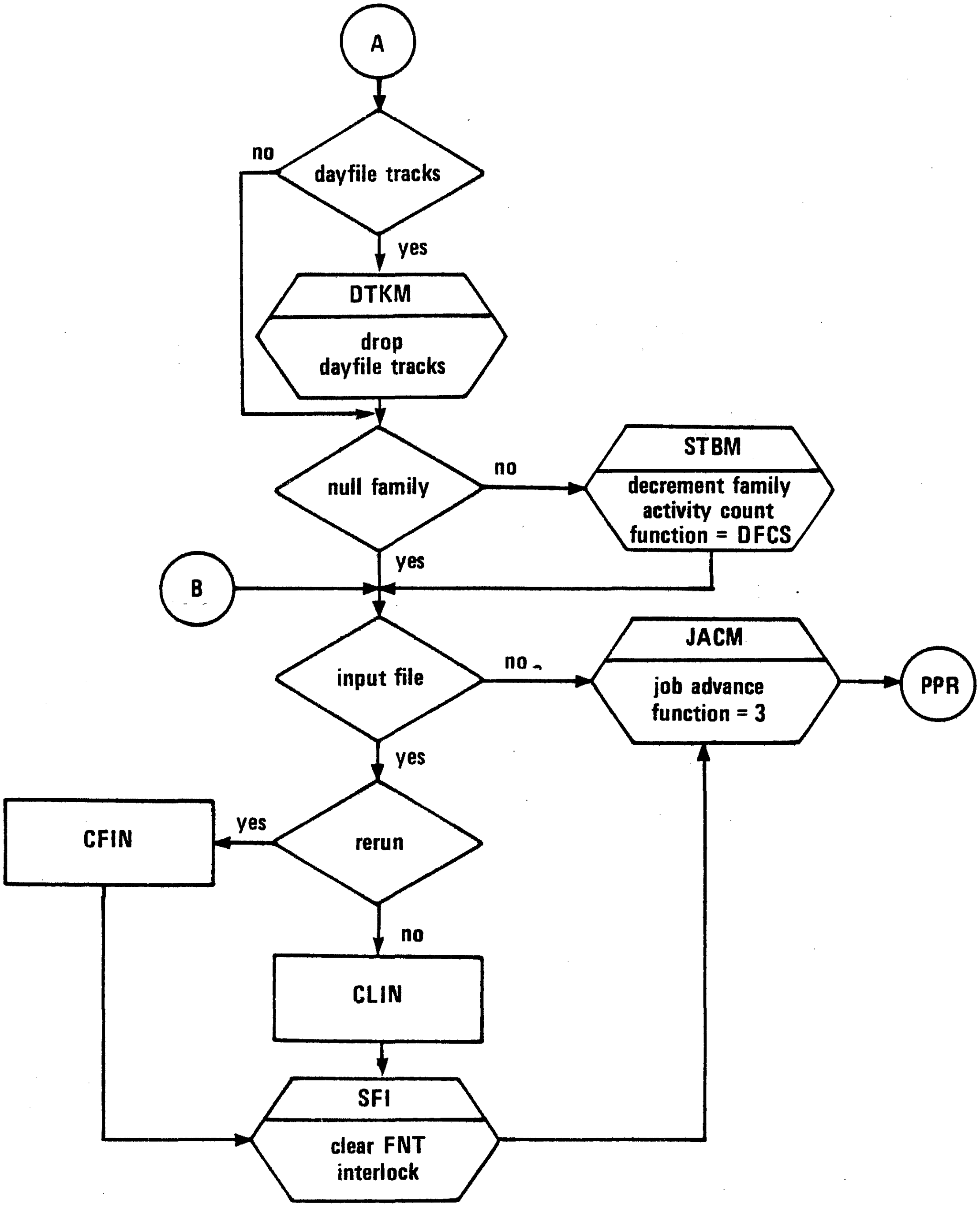
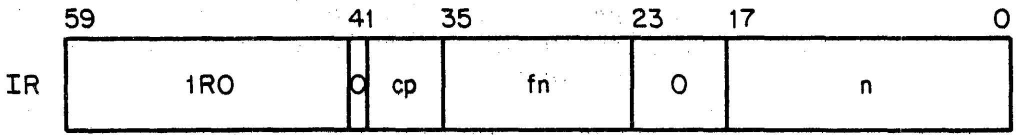


Figure 6-15 1CJ- Complete Job (Continued)

JOB ROLLOUT ROUTINE - 1RO

Routine 1RO performs job rollout in response to a calling program (such as the job scheduler) or a dump field length function from 1AJ.

The 1RO call is as follows:



cp	Control point number
fn	0 Rollout 1 Selective rollout to file DM* according to DMP= parameter
n	Error flag for TXOT job (function 0). For DMP= parameter (function 1) each bit defined as follows if set

Bits	Description
17	Checkpoint
16-15	Unused
14	Create DM* file only
13	Dump FNT entries to file DM*
12	Create DM* as an unlocked file
11-0	If 0, dump control point area and entire field length; if nonzero, dump control point area and FL*100B.

Routine 1RO uses the OBF, begin file, routine. Its direct location assignments are as follows:

Name	Value	Description
FS	20-24	FST entry (5 locations)
NT	25	Next track pointer
FW	26	FNT word count or central memory index
SC	27	Sector count terminal output
CN	30-34	CM word buffer (5 locations)
TW	35	Constant 2
DP	36	Dayfile pointer address
OT	37	Origin type
FN	40-44	FNT entry (5 locations)
TN	45	Terminal number
TT	46-47	Terminal table address (2 locations)
FA	57	Address of FST entry
ZR	60-64	CM zero word (5 locations)
TA	65	TELEX RA
OP	66-67	Output pointer (2 locations)

Routine 1RO is flowcharted in Figure 6-16

COMMON DECK COMSJRO

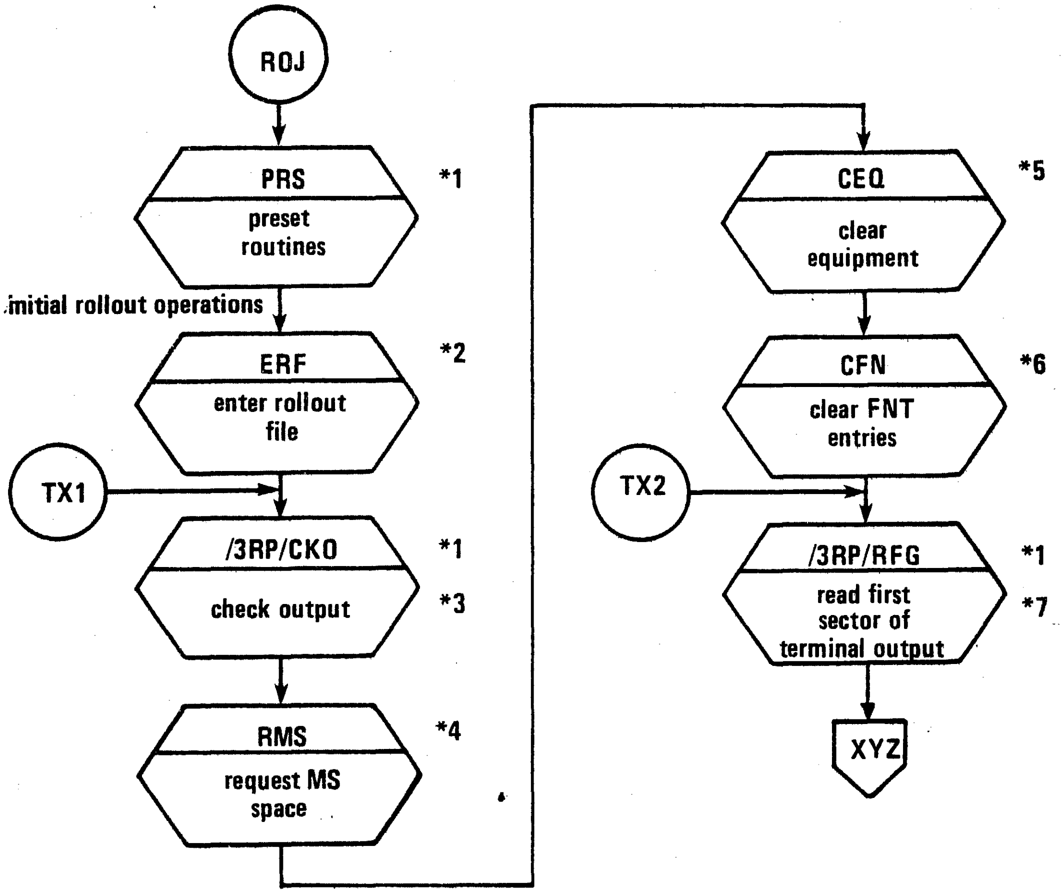
Common deck COMSJRO defines the sector format of the rollout file. The symbols are used by the rollout/rollin process.

Symbol	Description
CPAI	Control point area. The control point area is two sectors in length, and is the exact image of the control point area in central memory at the time of rollout.
DFBI	Dayfile buffer. The dayfile buffer area is one sector in length, and is an exact image of the job dayfile buffer in central memory.
FNTI	File name table. The file name table area is n sectors in length, terminated by a short sector (logical record). The FNT entries are stored as two-word (FNT/FST) entries in this area.
TOPI	Terminal output. The terminal output area is n sectors in length, terminated by a short sector (logical record). This is the only part of the rollout file that is unique for TXOT jobs.
JFLI	Job field length. The job field length area is n sectors in length, terminated by the EOI sector, and is an exact image of the job FL in central memory. If ECS is present, the FL is divided into two parts. The first part is the field length from RA to RA + (MCMX/2) - 1. The second part, which follows the ECS section, is the field length from RA + MCMX/2 to RA + FL - 1. The value MCMX defines the minimum CM field length when ECS is present.
JECI	Job ECS field length. The job ECS field length area is n sectors in length and is an exact image of the job ECS FL.

ROLLOUT FILE SYSTEM SECTOR

Common deck COMSSSE defines locations within the system sector that are unique for rollout files. These locations, located in the file dependent data area (DDSS), are:

Relative Location to BFMS (6776)	Symbol	Definition
52	DBSS	Dayfile buffer pointers (two 60-bit words)
64	INSS	Input file FNT and FST (two 60-bit words)
76	AESS	Assigned equipment list (1 byte per entry; terminated by a zero byte)
172	SJSS	SSJ= flag (nonzero if SSJ= job)
173	---	Reserved (2 bytes)
175	FQSS	Family EST ordinal
176	ERSS	Rollout ECS FL/1000B
177	SPSS	SSJ= job parameter block (five 60-bit words)
230	SWSS	System control point data (SF.SWPI); upper 6 bits contain priority, lower 18 bits are completion address
232	CLSS	Job class
233	---	Reserved (2 bytes)
235	SRSS	SRU information (60 bits); 12 bits reserved, 6 bits flags, 6 bits zero, 18 bits for time increment, and 18 bits for SRU increment
241	TLSS	Terminal table at last rollout (20 60-bit words)
361	TRSS	Terminal table for recovery (20 60-bit words)



- *1 Disable all jumps associated with TXOT origin jobs if this is a non-TXOT job.
- *2 Enter rollout file into FNT/FST. If DMP= call, then file name is DM* and control point will not be dropped.
- *3 Check for terminal input and output.
- *4 Request tracks for rollout file.
- *5 Release all equipment assigned to this control point.
- *6 Clear all FNT entries associated with this control point except the rollout file.
- *7 Prepares terminal output file for IAF/TELEX.

Figure 6-16 1RO-Rollout Job

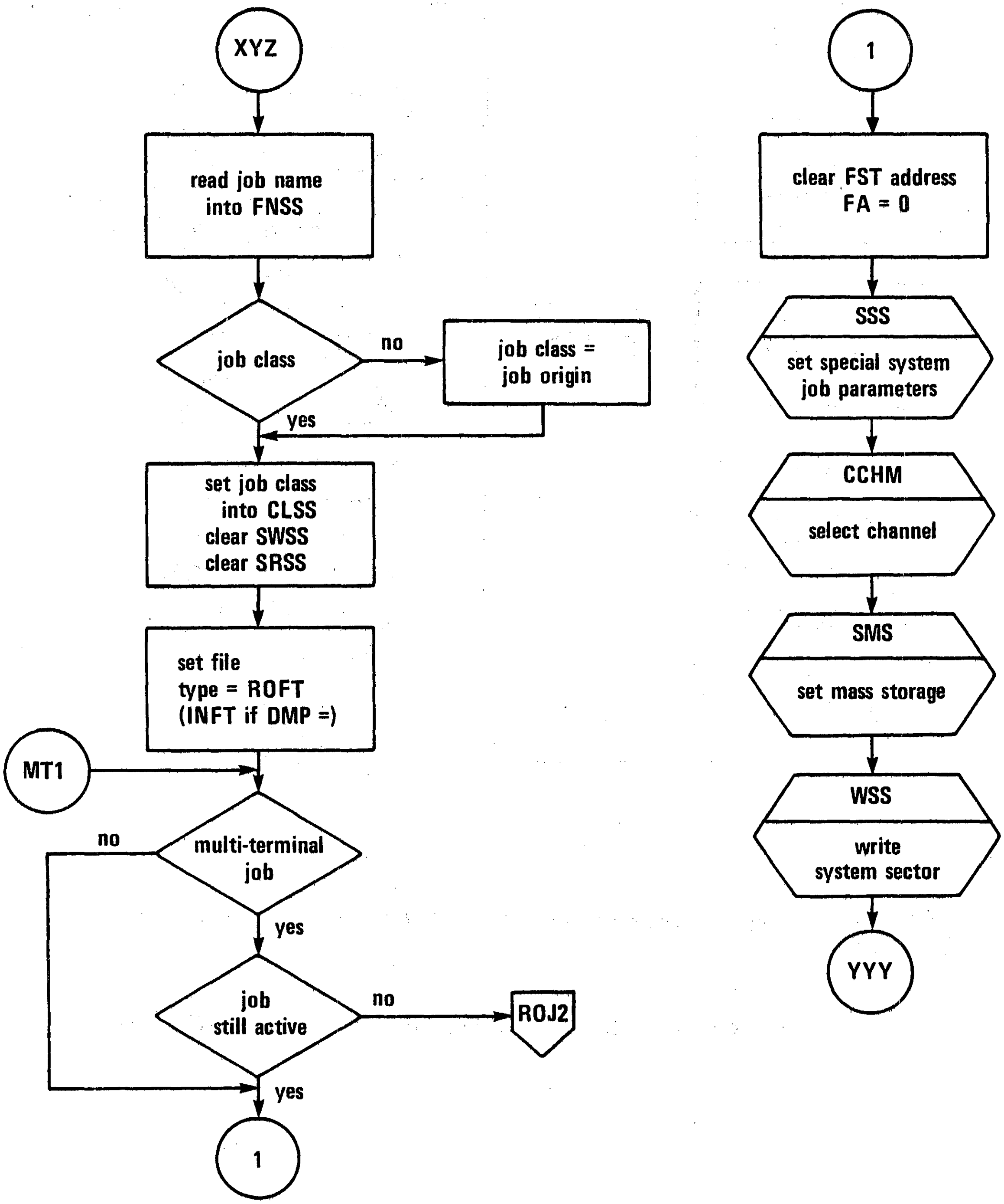
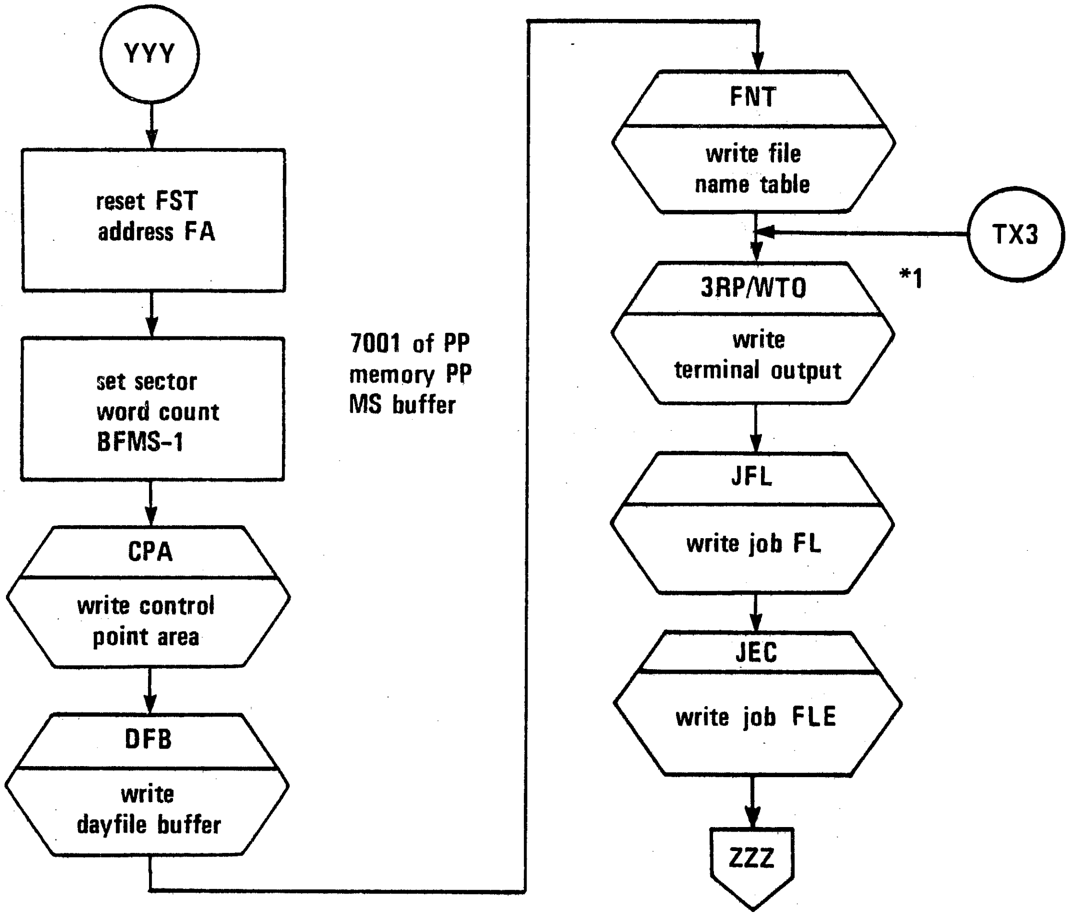


Figure 6-16 1RO-Rollout Job (Continued)



*1 Disable all jumps associated with TXOT origin jobs if this is a non-TXOT job.

Figure 6-16 1RO - Rollout Job (Continued)

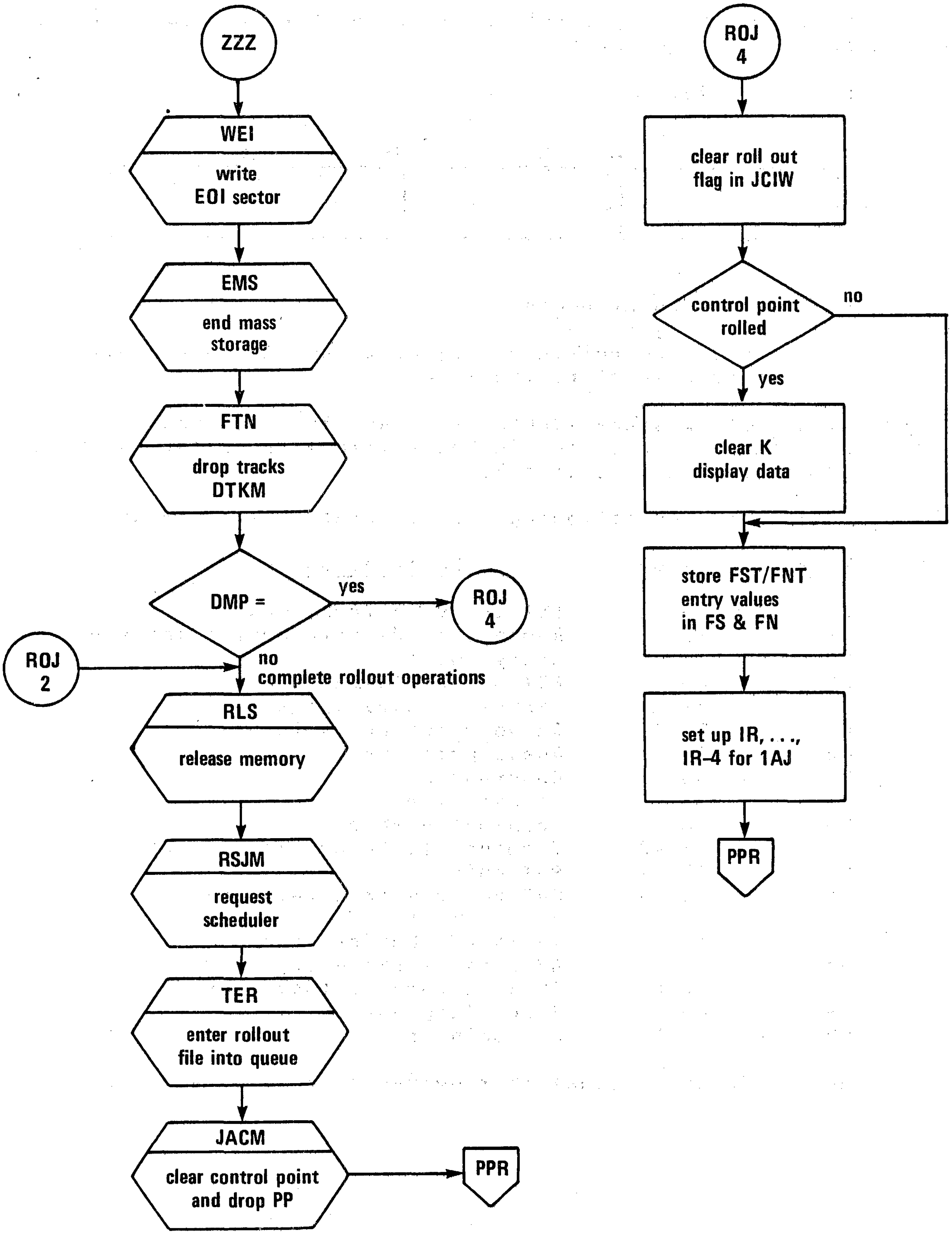


Figure 6-16 1RO - Rollout Job (Continued)

JOB ROLLIN - 1RI

Routine 1RI performs job rollin response to a calling program, such as the job scheduler.

Its call is as follows:

	59	41		35	23	11 0
IR	1RI	0	cp	fn	0	fa

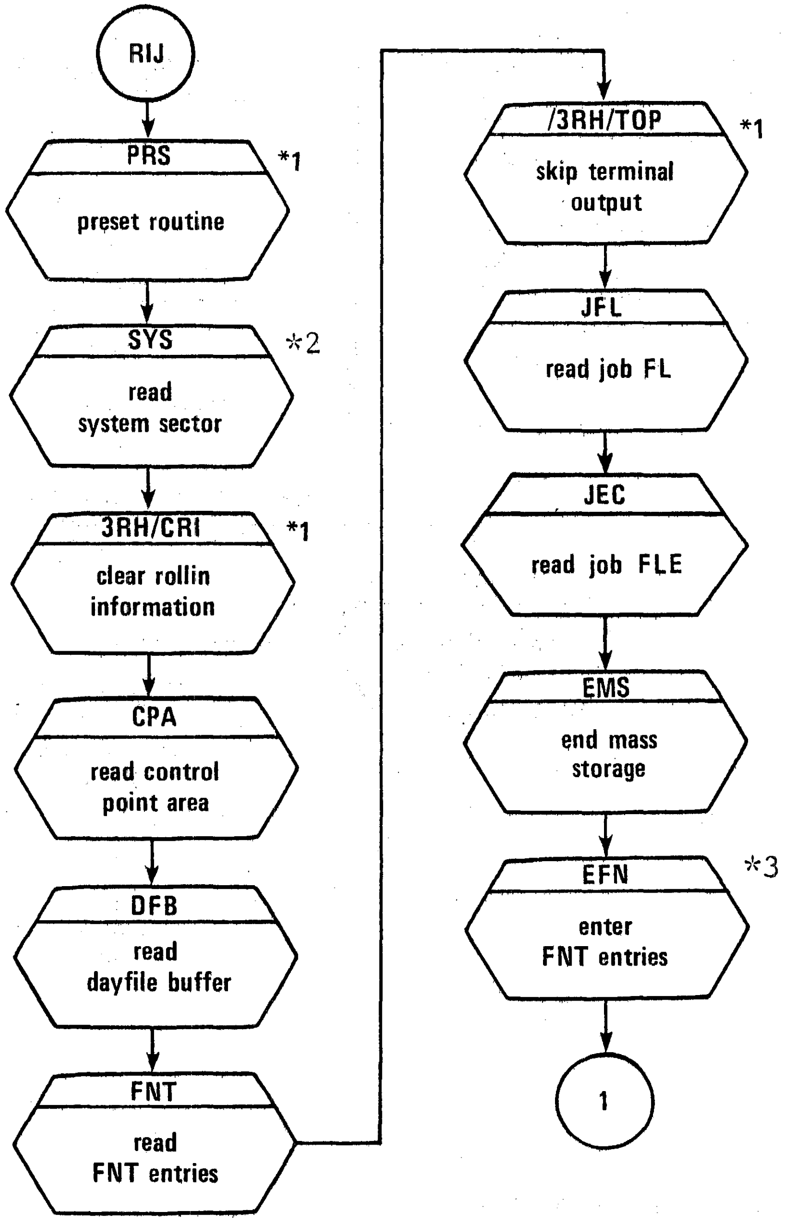
cp	Control point number
fn	0 Rollin job.
	1 Selective rollin according to special entry point
fa	FST address of rollin file

The 1RI dayfile message ROLLIN FILE BAD signifies that an illegal format was detected in the rollin file (refer to Common Deck COMSJRO, in this section).

Routine 1RI has the following direct location assignments.

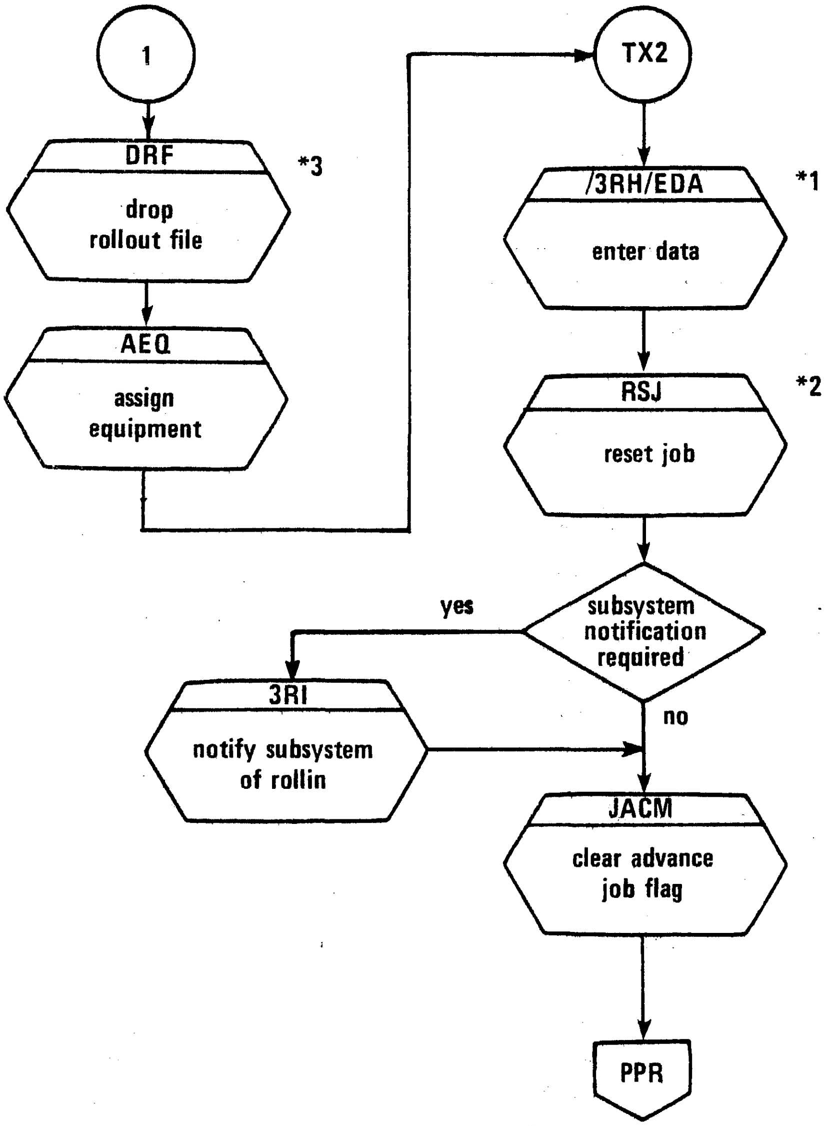
Name	Value	Description
FS	20-24	FST entry (5 locations)
DP	25	Address of dayfile buffer pointer
EP	25	Entry point
FI	26	FNT buffer index
CI	27	Central memory index
CN	30-34	CM word buffer (5 locations)
PR	35	Queue priority
TW	36	Constant 2
OT	37	Origin type
TN	40	Terminal Number
TT	41-42	Terminal table address (2 locations)
PP	43	POT pointer
PA	44-45	POT address (2 locations)
TA	46	RA of TELEX
TI	47	TELEX FNT buffer index
FA	57	Address of FST entry
ZR	60-64	CM zero word (5 locations)
EF	65	Error flag hold
OU	16-17	Out pointer

Routine 1RI (main routine) is flowcharted in figure 6-17.



*1 Disable all these if non-TXOT job.
*2 Mass storage set and positioned in preset.
*3 Disabled if no FNTS.

Figure 6-17 1RI - Rollin Job



*1 Disable if non-TXOT job.

*2 Set up control point area, put job in W status (RCPM) request.

*3 Drop FNT entry for this rollout file and drop all tracks.

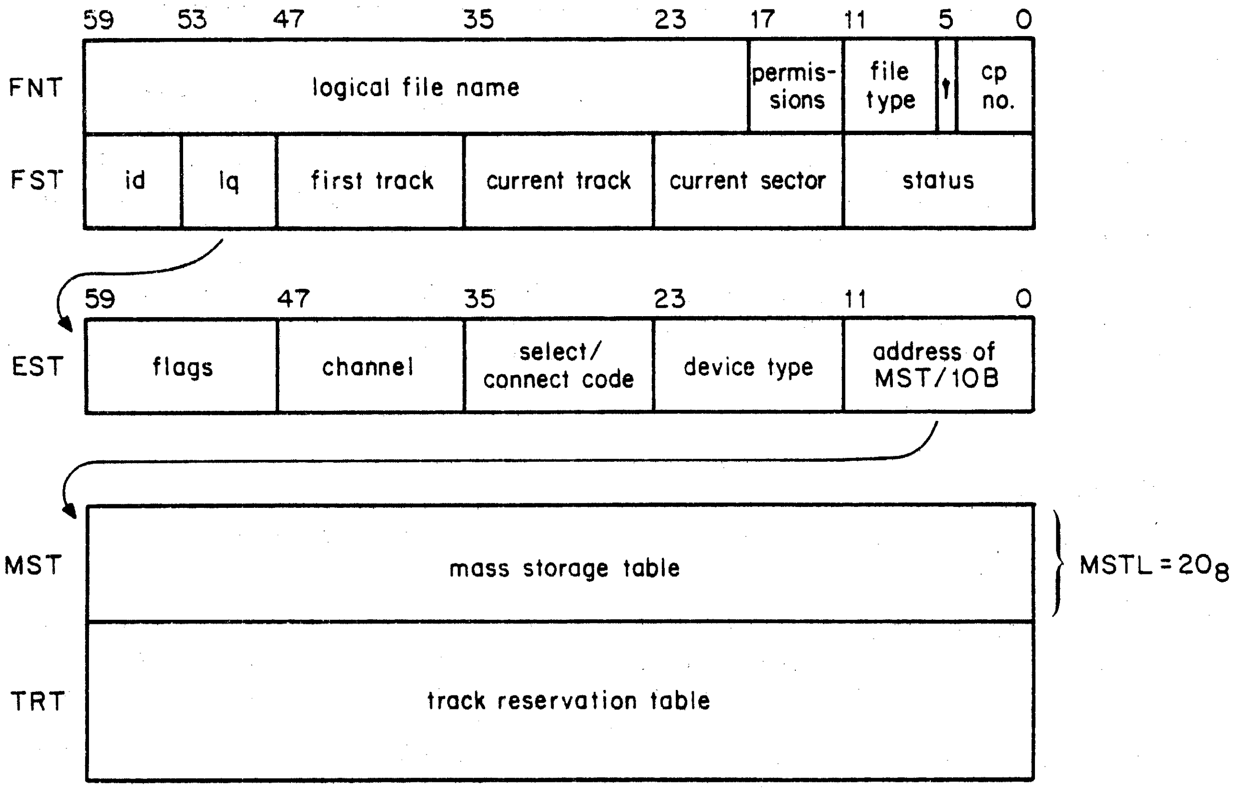
Figure 6-17 1RI - Rollin Job

All active files residing on rotating mass storage (RMS) are described by a file environment table (FET), a file name table (FNT), and a file status table (FST). The FET is supplied by the user and resides within the job field length. The FET is described in section 9. The FNT and FST are supplied by the system and are used by system routines to coordinate user requests for I/O and file positioning. Three other mass storage tables are involved with controlling I/O. These are the equipment status table (EST), the mass storage table (MST), and the track reservation table (TRT).

TABLE LINKAGE

The linkage between these tables is simple and reduces system overhead to a minimum.

The FNT and FST are two one-word entries in a single CM table. The FNT word (first word) contains the file name and a control point number which enables the system to associate an I/O request in a user FET with an FNT/FST word pair. The association of a user's FET with an FNT/FST word pair is obtained by comparing the file name in the FET and the control point number of the requesting job with the corresponding fields of each FNT entry until a match is found. The FST word (second word) contains file status, equipment number, and track linkage and position. The equipment number is used as an index into the EST which contains one-word entries describing the mass storage device type, the channels through which the device may be accessed, and a pointer to the MST for the device. The inter-relationship of these tables is as follows.



Each MST is located in CM on a 10B-word boundary so that the upper 12 significant bits of a 15-bit address can be stored in byte 4 of the EST as an MST pointer. The MST is a fixed-length table (20B words) which contains a complete description of the logical characteristics of a mass storage device. The MST is followed in memory by a variable-length TRT which is used to maintain allocation of the mass storage device.

TABLE CONTENT

Space for the FNT, FST, EST, MSTs, and TRTs is allocated in CM at deadstart time. Pertinent information from the CMRDECK is transferred into the EST and MST at this time. As mass storage devices are recovered (activated) at deadstart time by RMS or on-line by CMS, pertinent information is extracted from the mass storage device label and placed in the MST and TRT. As files are created, changed, or released on a device, the MST and TRT are updated to reflect logical device status. An FNT/FST entry is created for a file when a user request is processed for a nonexistent file (for example, CIO open or I/O request). As operations are performed on the file, the FNT/FST entries are updated to reflect current status of the file. The FNT/FST entry is cleared when a file is returned. The detailed content of the FNT, FST, EST, MST, and TRT are described in section 2.

---

† This bit is used to indicate special information in the system sector.

MASS STORAGE ALLOCATION

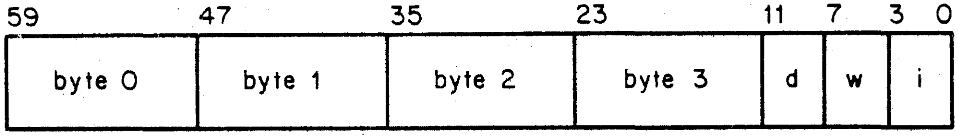
The system allocates mass storage space to a file in increments of a unit called a track. Tracks are further divided into units called sectors or physical record units (PRUs). A sector is the smallest unit of mass storage space that can be read or written at a time. The number of tracks per mass storage device and the number of sectors per track are device-dependent and are shown in table 7-1.

TABLE 7-1. TRT LENGTHS (OCTAL)

Device	Mnemonic	Length in CM Words	Track Count	Logical Sectors/Track	Sector Buffer Size (Bytes)
ECS	DE	Dependent on ECS length	Dependent on ECS length	20	502
844-21	DI	630	3140	153	502
844-4x	DJ	632	3150	343	502
844-21	DK	630	3140	160	502
844-4x	DL	632	3150	343	502
885	DM,DQ	645	3222	~~640~~ 1200	502
DDP/ECS	DP	Dependent on ECS length	Dependent on ECS length	20	502

Allocation of a mass storage device is controlled by a TRT. The TRT provides a track linkage byte and three flag bits for each track of a mass storage device. The track linkage bytes are used by the system to form a linked list of tracks as they are assigned to a file. The upper bit of each track linkage byte is a track linkage flag which indicates whether the remainder of the linkage byte represents a link to the next track for a file or whether it indicates a sector number within the track that is the last sector of the file. The last sector for a file is known as the end-of-information (EOI). The three flag bits for a track indicate whether it is free or allocated, whether it is interlocked or not, and whether it is the first track of a preserved file or not.

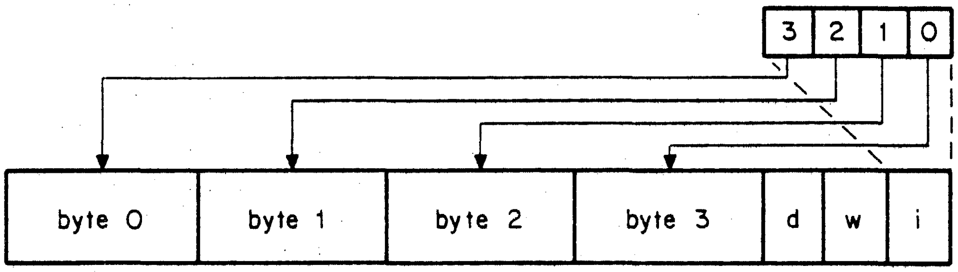
The TRT for a device contains the number of words as described in table 7-1 with each word containing linkage and control information pertaining to four tracks. Bytes 0, 1, 2, and 3 of each word of the TRT contain the track linkage byte pertaining to four tracks. Byte 4 contains the three flag bits for each of the four tracks represented by bytes 0 through 3. The following shows the format of each TRT word.



byte n	Track link byte
d	A bit is set corresponding to bytes 0 through 3 to identify the first track of a preserved file chain
w	A bit set establishes an interlock for a track
i	A bit set indicates track reservation

When set, the upper bit of each track linkage byte indicates linkage to the next track in a chain; when clear, it indicates which sector of the track is the EOI. For each of the three 4-bit flag fields contained in byte 4 (d, w, and i), the bits from left to right correspond to the same tracks represented by bytes 0 through 3, respectively. The track link byte format is as follows.

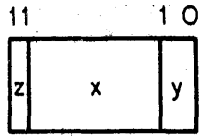
The following illustrates the correspondence between the control bit and the track linkage byte.



The d and w fields map in the same manner as the i field.

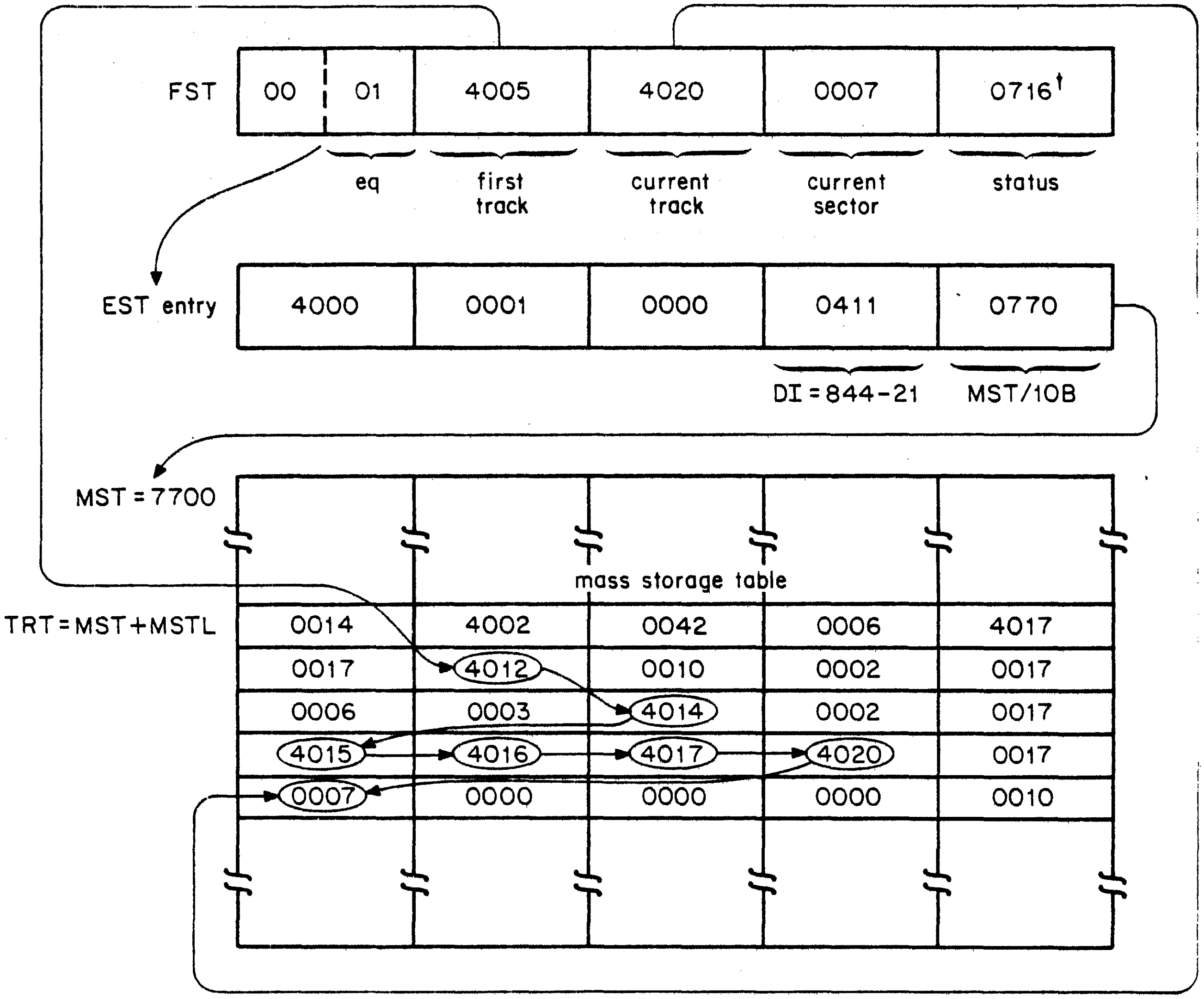
FILE LINKAGE

The first and current track fields in the FST and the track link bytes of the TRT contain a number that can be broken down to determine the word within the TRT and the byte within that word that is used to represent the track number. The general link byte format is as follows.



z	1 for next link in chain in bits 10 through 0, and 0 for EOI sector number in bits 10 through 0. (Always 1 for first and current track of FST.)
x	TRT word relative to word 0 of this TRT
y	Byte within word x.

The following is an example showing file linkage from FST to EST to MST. Notice that the file occupies space on tracks 5, 12, 14, 15, 16, 17 and 20. The EOI is sector 7 of track 20. The EST entry shows that the device is an 844-21 (device type byte equals 0411B, display code for DI) so that the MST/TRT length is 650 octal words. Also, the FST entry shows that the file is currently positioned at EOI. TRT linkage can also go backward (for example, 4012 points to 4002 which points to 4007). Tracks are linked and delinked by CPUMTR in response to PP requests.

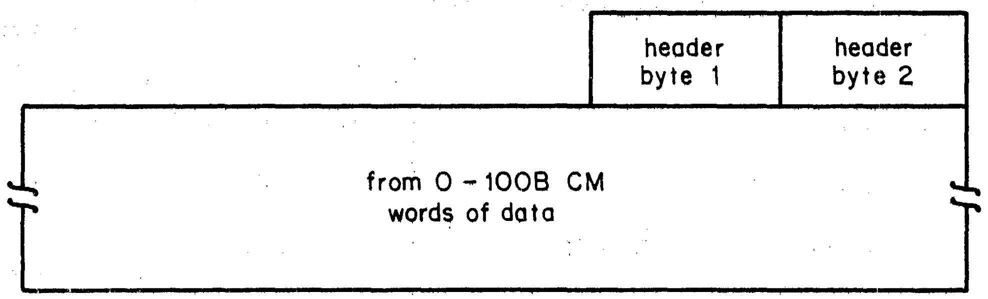


---

† FST status (refer to Section 2)

DISK SECTOR

Every sector, from a user standpoint, contains up to 64 (100B) CM words. However, the system always prefixes the sector with two header bytes (24 bits). These two header bytes contain file linkage and other information. The general format of a disk sector is as follows.



There are five types of sectors known to the system and labeled via the header bytes. These are:

- End-of-record (EOR) sector
- End-of-file (EOF) sector
- End-of-information (EOI) sector
- System sector
- Full sector

Header byte 2 contains a word count of the number of CM words within the sector as written by the user. The word count (WC) is in the range 0 to 100B. If the word count equals 100B, the sector is full. If the word count is less than 100B, the sector is called a short PRU and indicates an EOR. Table 7-2 shows the relationship between the sector types and the contents of the header bytes.

TABLE 7-2. SECTOR HEADER BYTE CONTENTS

Sector Type	Header Byte 1	Header Byte 2	Comment
EOR	Next Sector/Track	0 ≤ WC < 100B	May contain data
EOF	0	Next Sector/Track	No data
EOI	0	0	No data
System	3777B	77B	System data only
Full	Next Sector/Track	WC=100B	Full sector

In table 7-2, a full sector differs from an EOR sector by WC=100B rather than WC<100B as for the EOR sector.

To differentiate between a link to another track rather than to the next sector in header byte 1, the upper bit (bit 11) is set.

The PP common decks that read/write mass storage perform the reading and writing of the header bytes. Also, CIO reads/writes the header bytes for disk I/O.

In table 7-2, the system sector for a file is indicated by special header byte values. This is done to prevent accidental reading through the system sector itself. The system sector is always sector 0 of the first track of a file.

Examples of the various sector types are shown in figure 7-1. The device is assumed to be an 844-21; therefore, the sector count is from 0 to 152B. Two situations not shown in figure 7-1 are an EOR and an EOF as the last sector on a track which link to the next track.

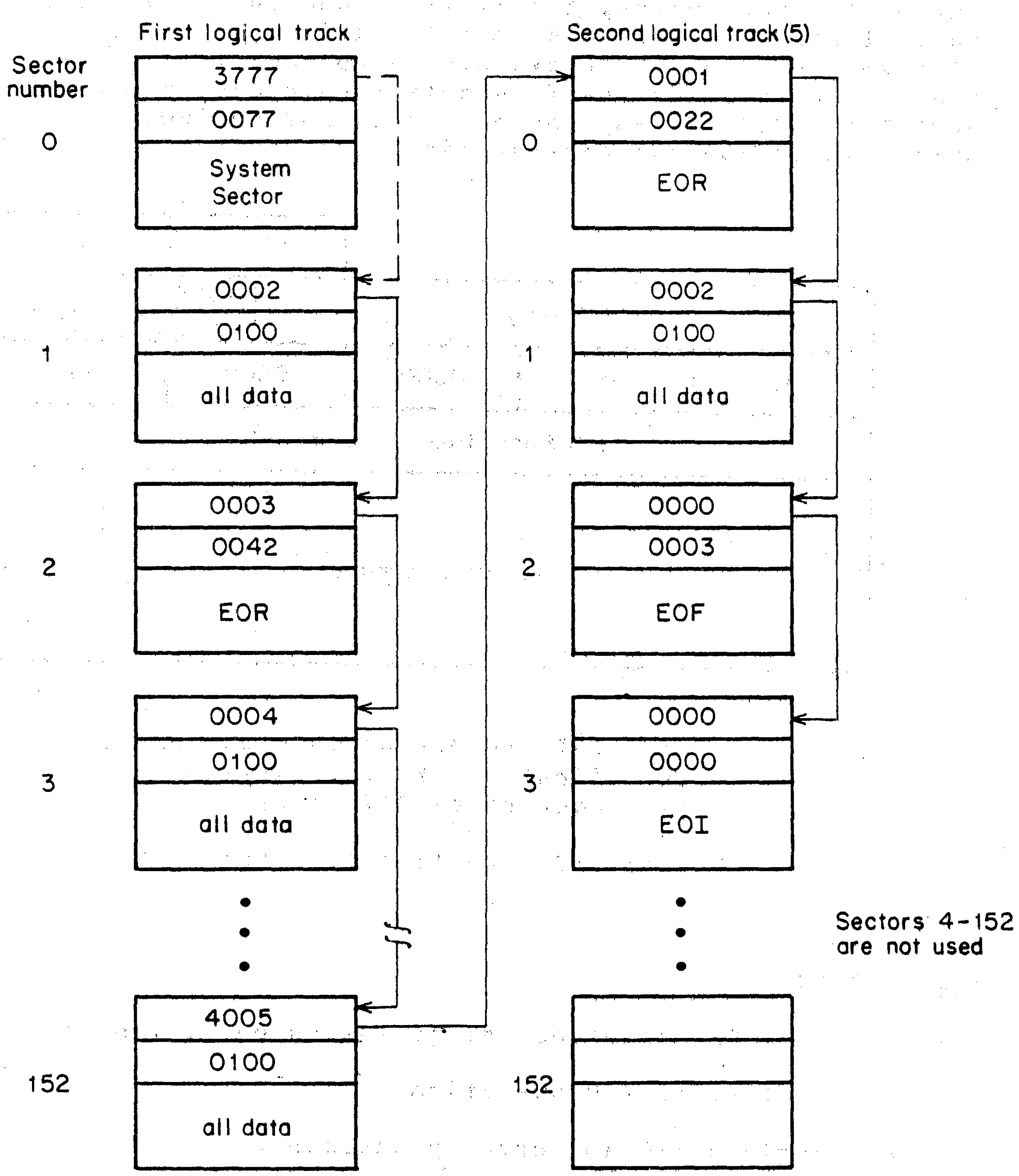
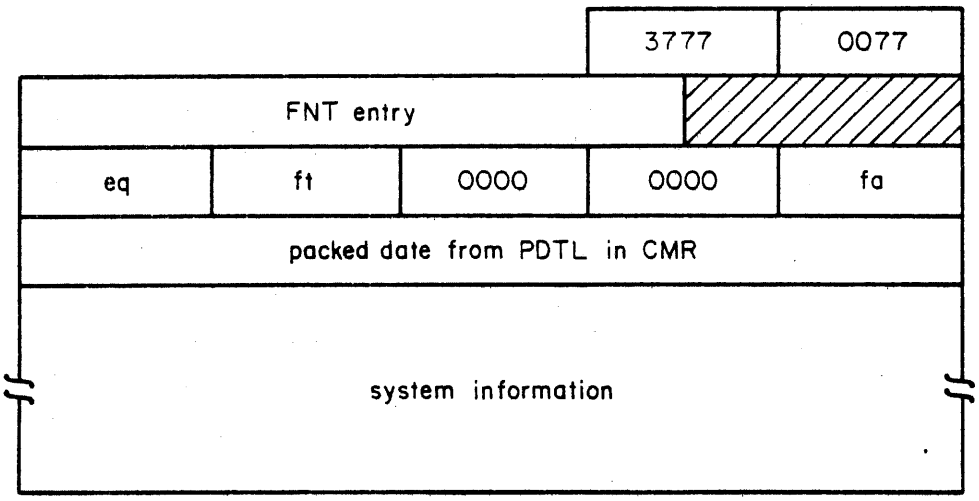


Figure 7-1. RMS File Structure

SYSTEM SECTOR

The system sector is the first sector of a mass storage file and contains system information. PP routines (for example, CIO, 1TA, 1RO) that write mass storage files begin by writing a system sector. The system sector is generally written by PP common deck COMPWSS. Although the calling routine stores various system information in the system sector, COMPWSS stores the control (header) bytes, the FNT/FST, and the data according to the following format. System information varies with different routines. For example, a rollout file system sector (Figure 7-2) includes dayfile buffer pointers, a copy of the input file FNT/FST, any operator assigned equipment, and terminal table information for time-sharing jobs.



eq	EST ordinal of this equipment
ft	first track of this file
fa	address of FST entry

Figure 7-2. Rollout File System Sector

DISK I/O FROM PPs

Disk I/O from PPs involves the following basic functions.

1. Initialize I/O operation.
2. Perform I/O and error processing.
3. End mass storage operation.

The following code illustrates these functions.

```
                .
                .
                .
     1.     STD     T5         SET EQUIPMENT
            SETMS   READ       INITIALIZE DRIVER
                .
                .
                .
            STD     T6         SET TRACK AND SECTOR
            STD     T7
     2.     LDC     BUF        READ SECTOR
            RJM     RDS
            MJN     ERR        IF ERROR

     3.     ENDMS              END MASS STOAGE OPERATION
```

INITIALIZE I/O OPERATION VIA SETMS MACRO

The first step of any I/O operation is to initialize the driver. This function is performed by first setting the equipment and then executing the SETMS macro. The SETMS macro performs the following functions.

1. Insures correct driver is loaded.
2. Passes the operation to the driver. Possible operations are read, write, and read of system file.
3. Executes driver preset and clears driver cells DRSW, RDCT, STSA, ERXA, and WDSE.
4. Selects error processing options and writes error processing buffer.

The format of the SETMS macro is as follows.

LOCATION	OPERATION	VARIABLE SUBFIELDS
	SETMS	op,(ep1,ep2,...,epn),wb

op I/O Operation Being Performed:

op	Meaning
READ	RDS will be called
WRITE	WDS will be called
READSYS	Read of system file.

NOTE

T5 must be set to any system device equipment upon entry to SETMS. This is so the correct driver can be loaded for that equipment type. The actual system equipment used will be set at the time the driver requests the channel.

epn Error processing options.

Error processing is selected by default if this field is not set. If this field is used COMSMSP must be present because it contains the bit definitions for the error options.

epi	Meaning
NE	No processing of errors by caller. If an unrecovered error occurs the job will be aborted and the PP will be dropped.
SM	Suppress error log messages. No error log messages will be issued for errors occurring while this option is selected.
RR	Return on reserve errors. If a CR (controller reserve) or an RS (drive reserve) error is encountered, control will return to the calling program without any retries.
NR	Return on not ready. If a NR (not ready) error is encountered, control will return to the calling program without any retries.
AR	Return on all errors. Whenever any error occurs control is returned to the caller without any retries.

wb Address of a 502B byte buffer than can be used during write error processing to retry errors encountered on the previous sector.

I/O OPERATION AND ERROR PROCESSING

The second functional area of an I/O operation is the I/O itself. I/O is done by the driver routines RDS (read sector) and WDS (write sector). The entry/exit conditions for these routines are as follows.

Entry to RDS:

(T4)= Channel if driver previously called. T4 is set by the driver upon initial entry when it reserves a channel.

(T5)= Equipment.

(T6)= Track.

(T7)= Sector.

(A) = Address of 502B byte buffer of data to be written to disk.

Exit: (A) = <0 if error and error processing is not deselected.

Entry to WDS:

(T4)= Channel if driver previously called. T4 is set by the driver upon initial entry when it reserves a channel.

(T5)= Equipment.

(T6)= Track.

(T7)= Sector.

(A) = Buffer address + consecutive sector indicator WCSF is added to the buffer address if a consecutive sector will follow the present sector.

WLSF is added to the buffer address if this is the last of a consecutive string of sectors being written.

See Example 1 for an illustration of the use of WCSF and WLSF.

Exit: (A) = -0 if unrecovered error.

(A) = -1 if a recovered error has occurred on the previous sector. The -1 indicates that the data buffer for the current sector has been destroyed and that the current sector must be reissued.

See Example 2 for the use of the reissue sector.

Example 1:

Write consecutive/last sector flag usage

```
                 .
                 .
                 .
         STD     T5
         SETMS   WRITE,,EBUF
                 .
                 .
*        WRITE A FILE OF EOI, BUF IS CLEARED WITH ZEROS.

         STD     T6
         LDN     0
         STD     T7

A        LDC     BUF+WCSF  CONSECUTIVE SECTOR FOLLOWS THIS SECTOR
         RJM     WDS
         MJN     ERR       IF ERROR
         AOD     T7
         LMM     SLM
         NJN     B         IF NOT END OF TRACK
         STD     T7        RESET SECTOR
         LDI     NT        SET NEXT TRACK
         STD     T6
         AOD     NT

B        SOD     SC
         NJN     A         IF NOT LAST SECTOR
         LDC     BUF+WLSF  WRITE LAST SECTOR
         RJM     WDS
         MJN     ERR       IF ERROR
         ENDMS             END MASS STORAGE OPERATION
```

NOTE

It is not necessary to tell the driver when writing the last sector of a track.

Example 2:

Reissue of current sector. Whenever the following conditions are met the PP program should be set up to reissue the current sector.

- More than one sector is being written consecutively. That is, WCSF being used.
- A write error processing buffer is not defined.
- Error processing is not deselected.

```
           .
           .
           .
           STD     T5             NO ERROR PROCESSING BUFFER BUT CALLER
           SETMS   WRITE          WISHES TO PROCESS ERRORS.
           .
           .
           .
           STD     T6
           .
           STD     T7
           .
           .
           LDC     BUF1+WCSF      WRITE SECTOR 1
           RJM     WDS
           MJN     ERR            IF ERROR - REISSUE NOT POSSIBLE ON FIRST
           AOD     T7

  A        LDC     BUF2+WCSF      WRITE SECTOR 2
           RJM     WDS
           PJN     B              IF NO ERROR
           ADN     1
           ZJN     A              IF REISSUE OF SECTOR 2
           UJN     ERR            PROCESS ERROR

  B        AOD     T7

  C        LDC     BUF3+WLSF      WRITE SECTOR 3 (LAST SECTOR)
           RJM     WDS
           PJN     D              IF NO ERROR
           ADN     1
           ZJN     C              IF REISSUE OF SECTOR 3
           UJN     ERR            PROCESS ERROR

  D        ENDMS                  END MASS STORAGE OPERATION
```

END MASS STORAGE OPERATION

ENDMS releases all resources assigned because of an I/O operation. For the 6DI driver this means the drive and controller are released via the operation complete function after which the channel and software unit interlock is released.

For the 6DE driver it means the PP buffer interlock is released since no channel is applicable to this driver.

ENDMS has the following entry conditions.

Entry: (T4)= Channel as set by the driver. If no driver call was executed since the initial SETMS, then T4 is ignored.

(T5)= Equipment.

Exit: None.

GENERAL PROGRAMMING CONSIDERATIONS

Storage Move

Any time the driver is entered (via RDS, WDS, ENDMS) the control point may be storage moved. Care must be taken if the value of RA is stored in any instructions.

Random I/O

Whenever the track (T6) is changed and the I/O operation is not starting at the beginning of the track (sector 0) a SETMS macro must be executed to have the driver note the random operation and reposition. The ENDMS will also force a reposition but should only be used if it is desirable to release the I/O resources.

Switching Equipments

It is imperative that an ENDMS is done for the equipment switching from and a SETMS is done for the equipment being switched to.

SETMS, ENDMS Sequences Allowed

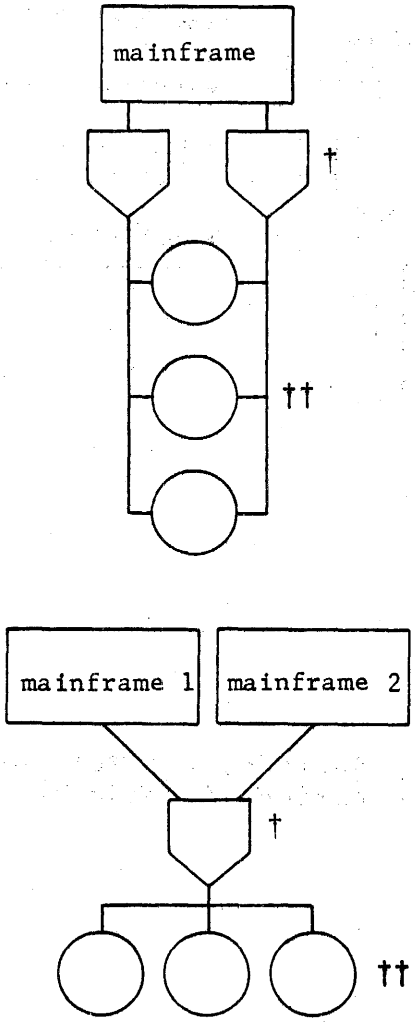
Given that a SETMS begins an I/O sequence on a device and the ENDMS ends the sequence, virtually any combination of SETMS, ENDMS, and I/O can be done in between. This is contingent upon following the proper rules for random I/O and specifying the operation via SETMS.

The following flow chart should illustrate the more common sequences, all of which are legal.

1. SETMS	1. SETMS	1. SETMS
2. No I/O	2. I/O	2. I/O
3. ENDMS	3. SETMS	3. ENDMS
	4. I/O	4. I/O
	5. ENDMS	5. ENDMS

DUAL, SHARED, AND MULTIPLE ACCESS

Dual access means that a disk storage unit (drive) has an access to it from two controllers on different channels from the same mainframe. Shared access means that a disk storage unit has an access to it from a controller that can be accessed by more than one mainframe. If a drive has both dual and shared access to it, it is called multiple access. This latter configuration is the one most commonly found in multimainframe environments. These configurations are displayed in figure 7-3.



DUAL ACCESS

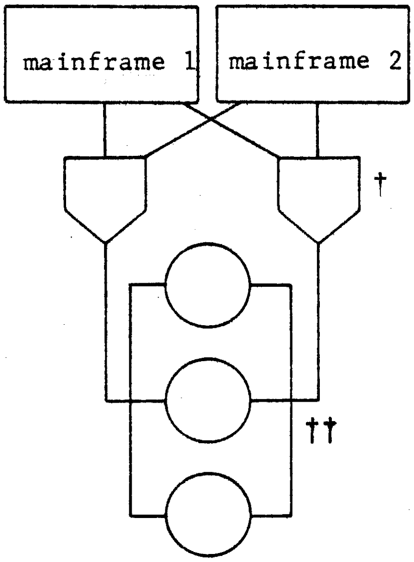
- Dual-access drives
- One drive access to each controller
- Each controller connected to same mainframe and has its own channel

SHARED ACCESS

- One controller access to drives
- Controller access by two mainframes

t Controllers
tt Drives

Figure 7-3. Dual-, Shared-, and Multiple-Access Configurations



MULTIPLE ACCESS

- Dual-access drives
- One drive access to each controller
- Controller access by two mainframes

† Controllers
†† Drives

Figure 7-3. Dual-, Shared-, and Multiple-Access Configurations (Continued)

SEEK OVERLAP - 6DI DRIVER

The 6DI driver performs a seek operation to inform the disk controller of an address to position to in preparation for the next data transfer. Once the seek is initiated by the controller, the disk drive can complete the positioning without further direction from the controller. This allows the controller to perform read and write operations or to initiate positioning on other drives that may be accessed through the controller. This overlapping of head positioning with reading, writing, or initiation of head movement is called seek overlap.

The overlapping of seek operations is managed by the driver seek wait monitor function (DSWM).

MMF OPERATION OF SEEK OVERLAP

There are two basic differences in driver operation between MMF and non-MMF configurations.

First, in an MMF environment it is possible to get a drive reserved status back from the seek operation. In non-MMF systems this status should never be seen because of the software interlock on the unit. The drive reserved status is handled similarly to the non-MMF drive busy. The controller is released via the operation complete and the channel is released via DSWM. A time-out scheme is employed for both drive and controller reserve conditions and any time the condition persists for 5 seconds an error indication is returned to the driver. The error must persist through 64 retries before being considered unrecoverable.

The second difference is that in an MMF environment the drive cannot be released during the seek operation. If it was, the other machine could reseek to a different position and thrashing would result. Thus, in an MMF environment the I/O operation must be done on the same channel as the seek.

NON-MMF OPERATION OF SEEK OVERLAP

After the seek function is issued the only status that should be received is either drive busy or an error status. If drive busy is received, the drive and controller are released via the operation complete function. This releasing of the drive allows the driver to come back and do I/O to that unit from the other channel on a dual channel configuration. The channel is then released to the system via the DSWM monitor function. The drive is protected from other requests for the unit by the software unit reserve in MST word DILL. This interlock is gained at the time a channel is initially assigned and is not released until an ENDMS is encountered.

The channel is now free for other I/O requests. Every time through the MTR loop a check is made for a free channel to assign to the seeking driver. If one is free it is assigned and the driver will reseek and check again for on-cylinder status. This sequence continues until an error or on-cylinder is detected.

FLOWCHARTS FROM 6DI DRIVER

A core map is found in figure 7-4. Flowcharts from the 6DI driver are shown in figures 7-5 through 7-11. PRS (preset) is entered from SMS while the other three routines are entered via return jumps to EMD, RDS, and WDS.

All disk drivers are originated at location MSFW for loading into PP resident. The first location (556) contains the entry point to the preset subroutine within the driver. This is used by SMS when it has been determined that the correct driver is loaded. Following this are the three entry points:

557 RDS - Read sector

562 WDS - Write sector

565 EMS - End mass storage operation

The symbols WDS and RDS are defined in PPCOM and are the same for all drivers. The following functions from the 6DI mass storage driver are flowcharted.

- PRS - Preset
- FNC - Issue function
- EMS - End mass storage operation
- RDS - Read sector
- WDS - Write sector
- LDA - Load address
- DSW - Driver seek wait processing
- DST - Check drive status

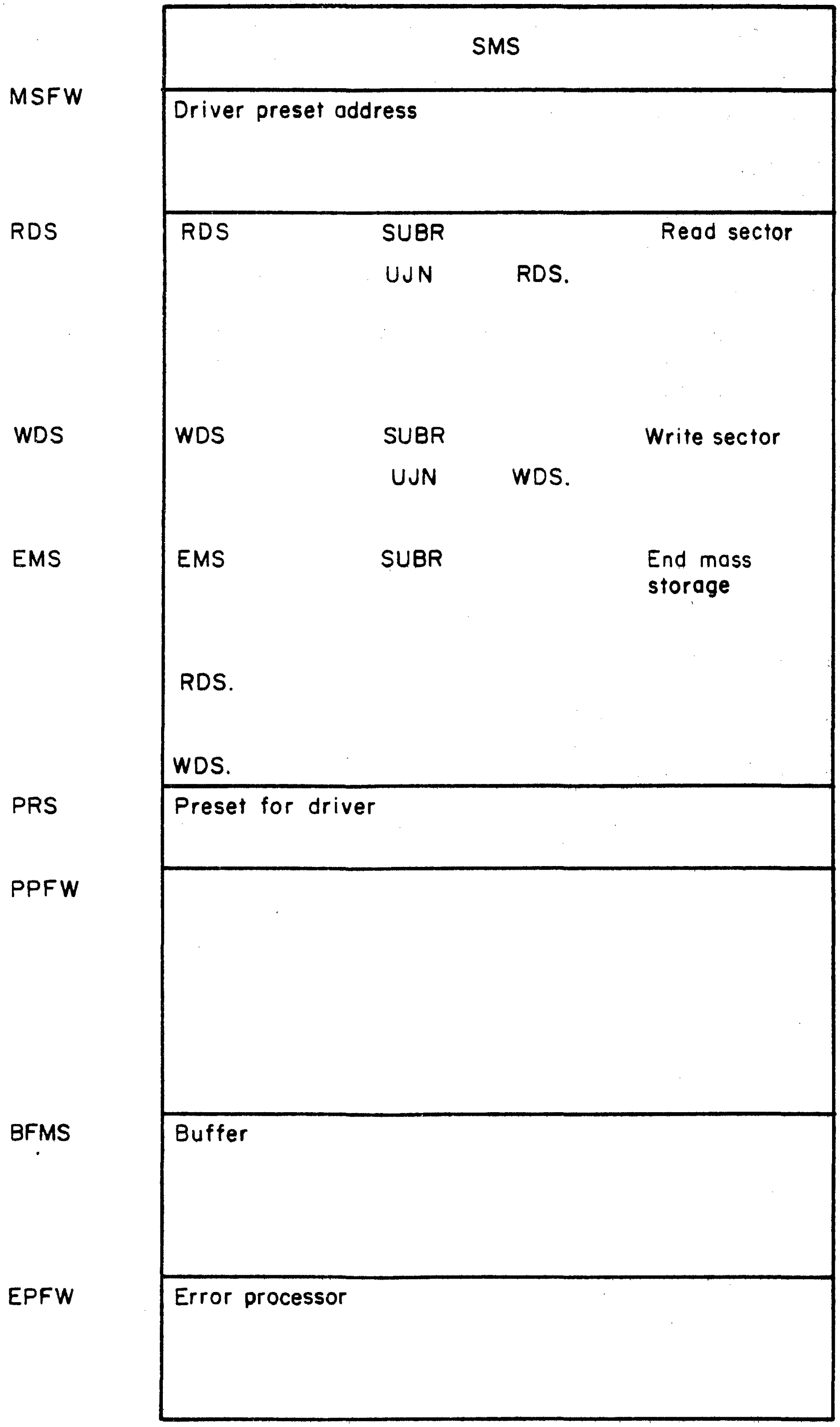


Figure 7-4. MS Driver Core Map

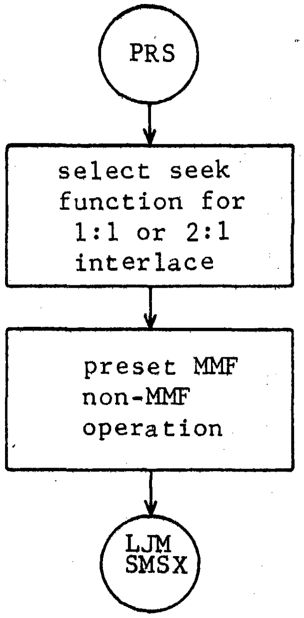


Figure 7-5. PRS - Preset

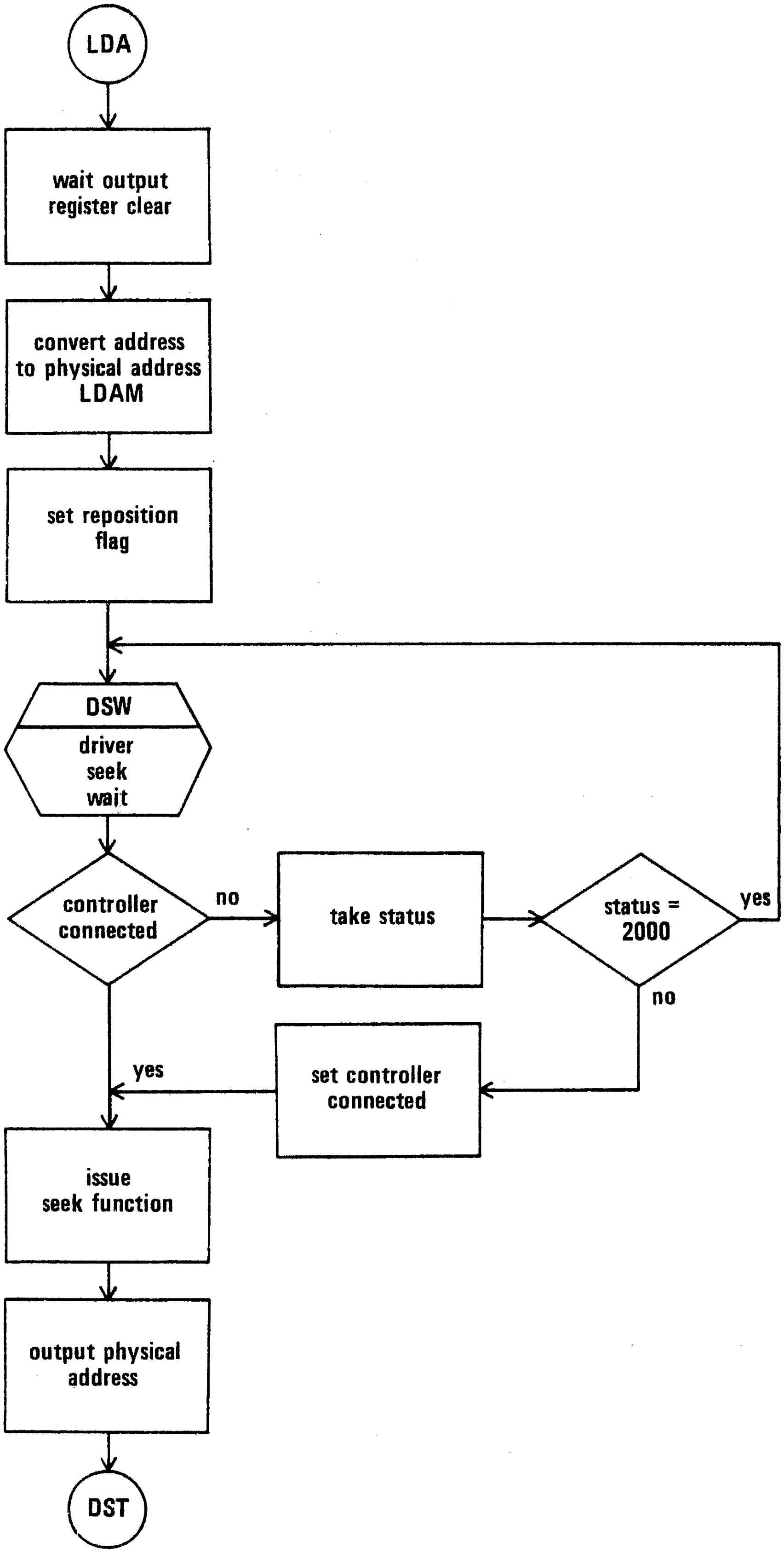


Figure 7-6. LDA - Load Address

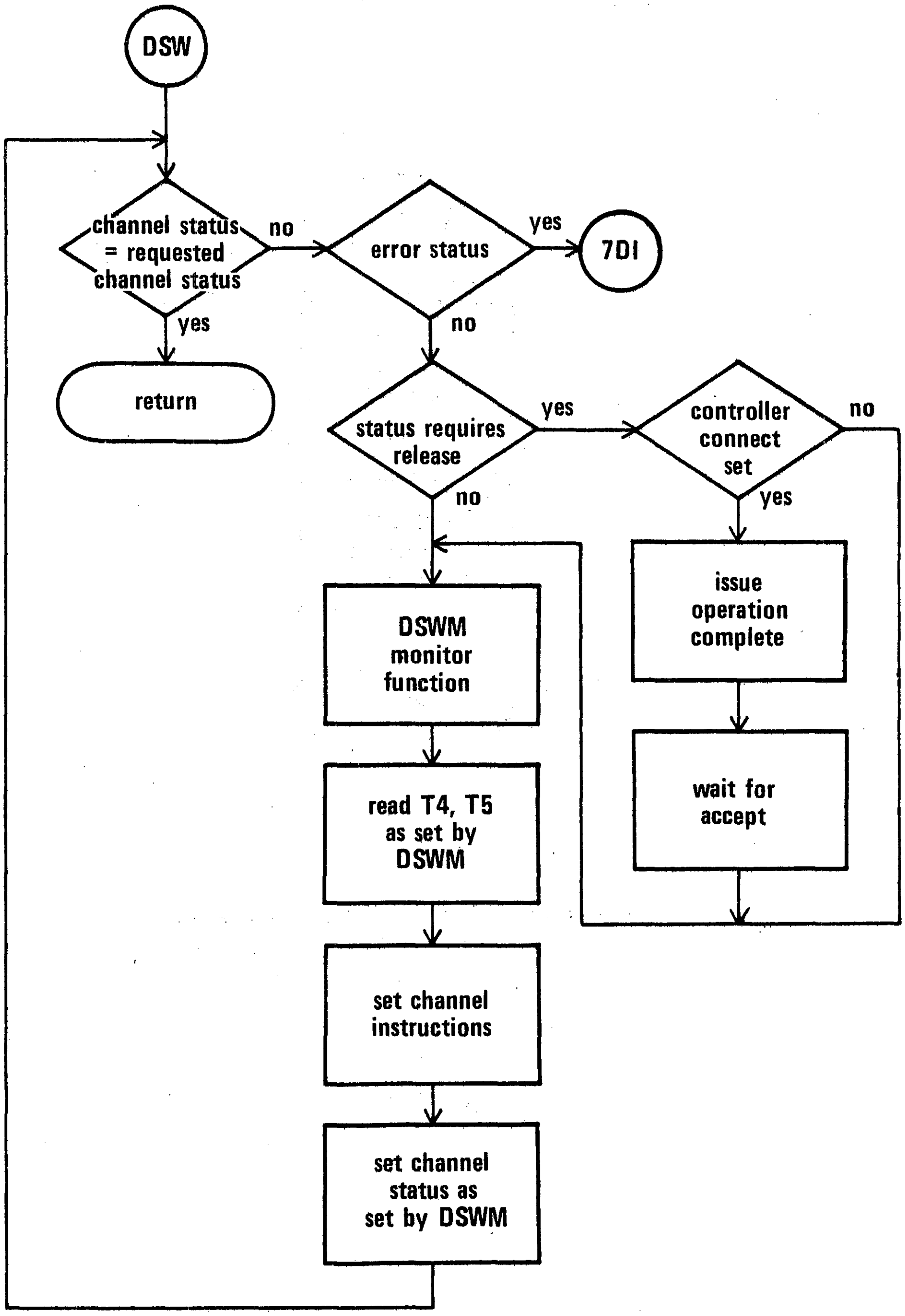


Figure 7-7. DSW - Driver Seek Wait

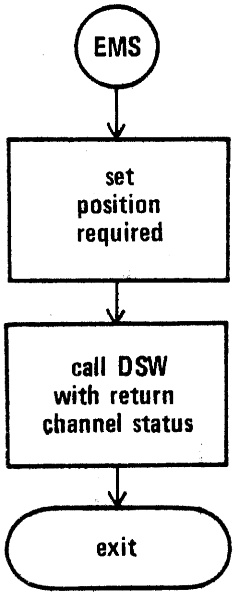


Figure 7-8. EMS - End Mass Storage

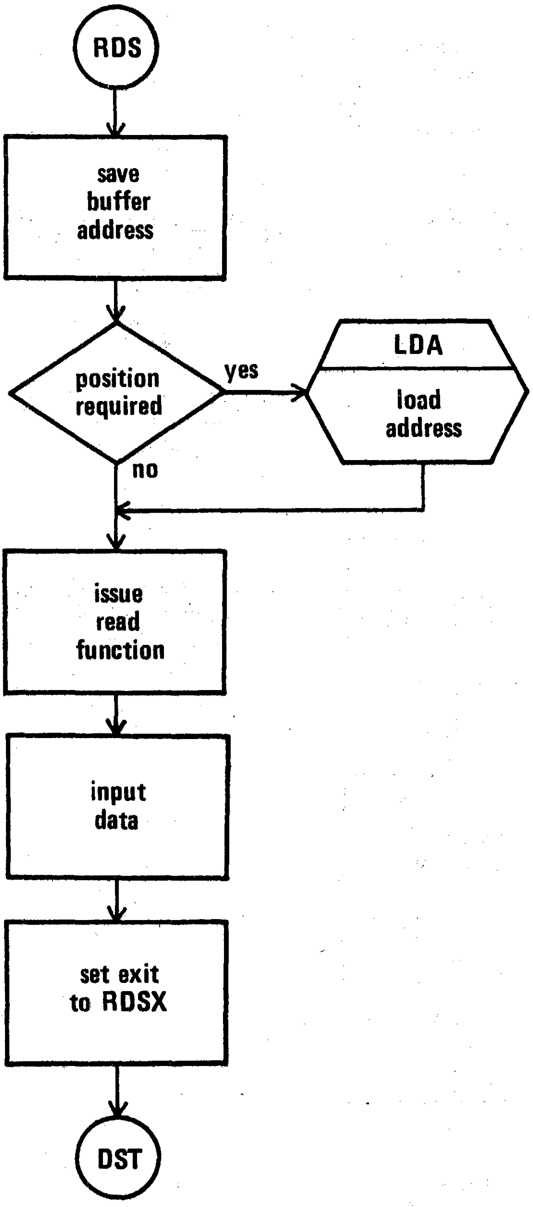


Figure 7-9. RDS - Read Sector

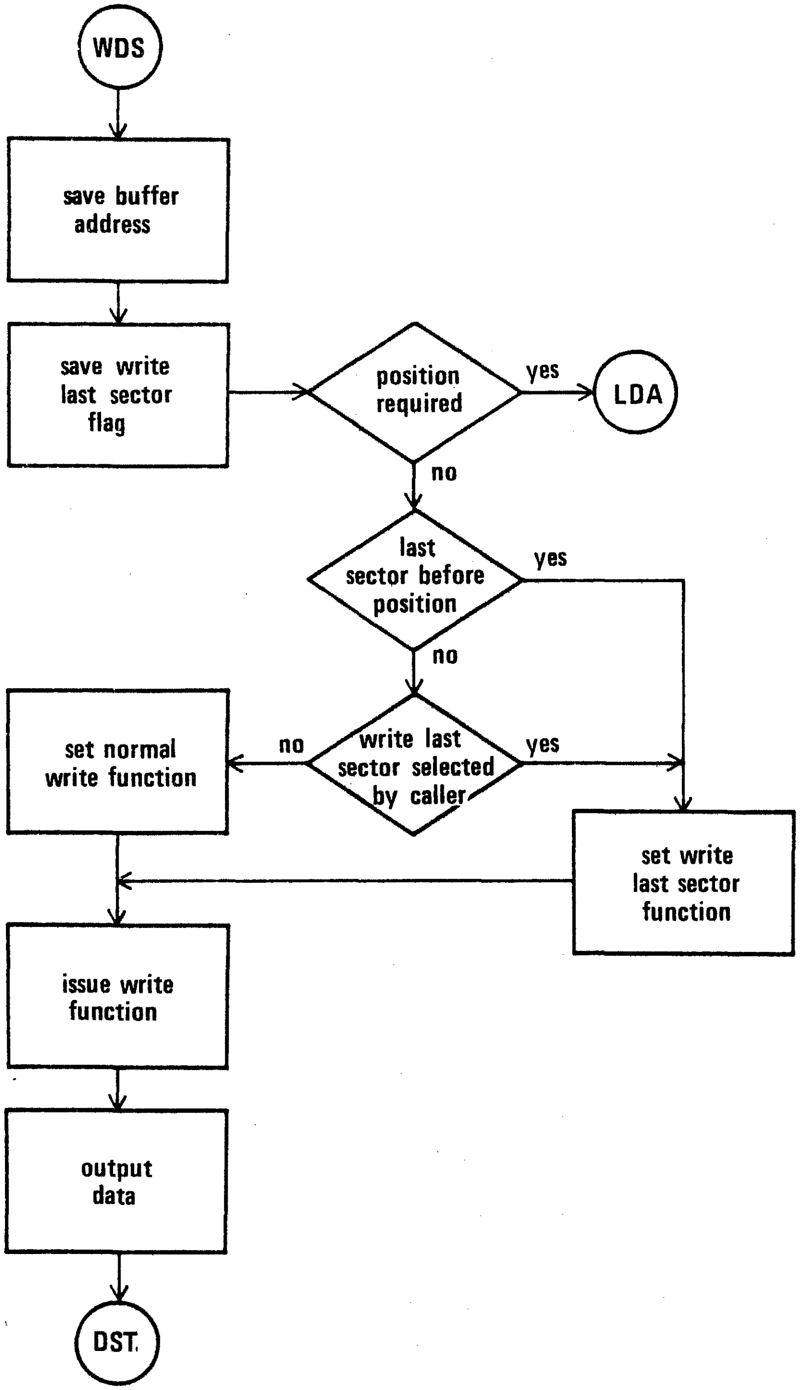


Figure 7-10. WDS - Write Sector

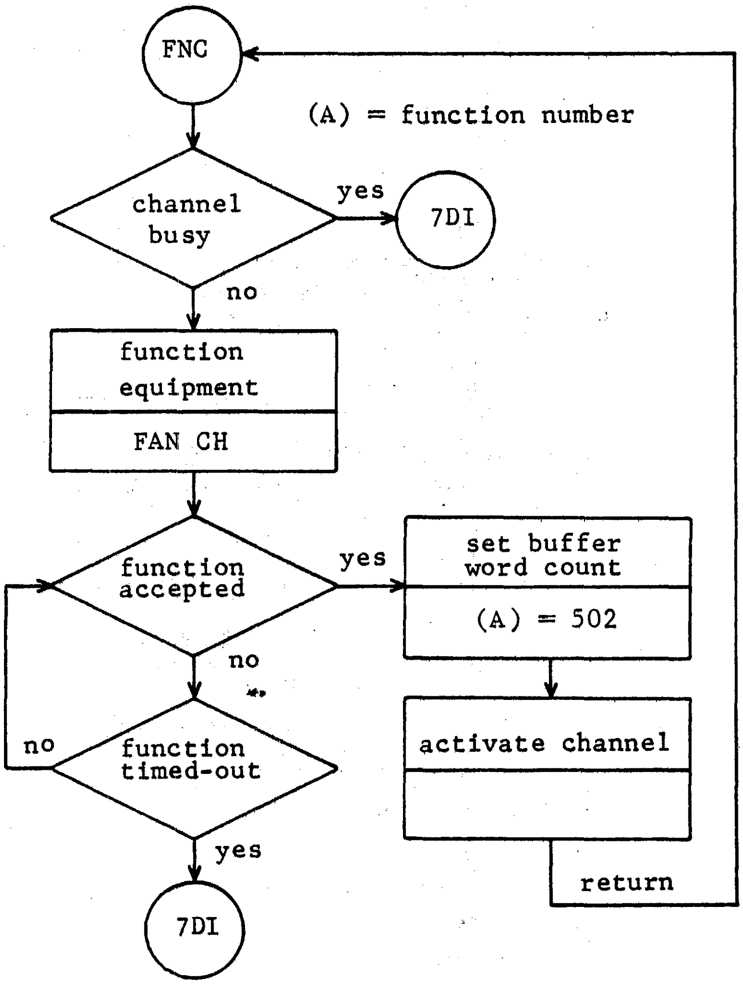


Figure 7-11. FNC - Issue Function

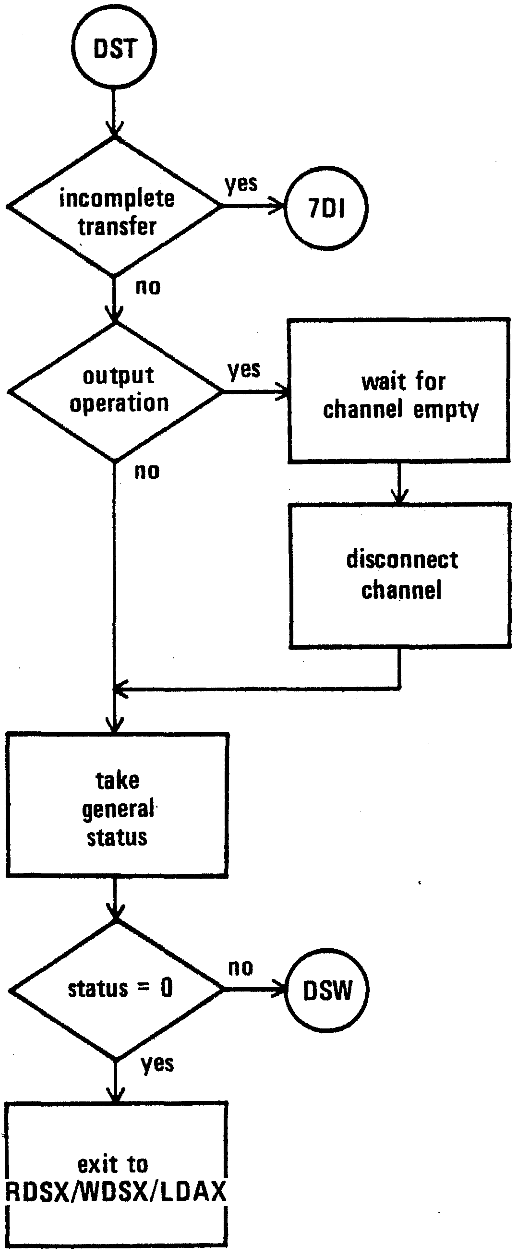


Figure 7-12. DST - Check Drive Status

6DP DDP/ECS DRIVER

Routine 6DP provides the capability to access the ECS I and ECS II secondary storage devices via the DC135 or the parity enhanced DC145 DDP. Routine 6DP performs the basic read/write functions for the DP type equipments.

Whenever an unrecoverable ECS abort occurs, 7DP, the DDP/ECS error processor, is called. If a read function was being processed and a parity error occurred, 7DP calls 7RP. Routine 7RP retrieves the data in error from the DDP port and completes the read of the remaining ECS words (7RP is not called on a write parity error). After 7RP is called on a read parity error, 7RP recalls 7DP. At this time 7DP issues its first DEPM and calls 7EP, causing the initial error message to be issued for the block of data being read or written.

Once control is returned to 7DP, it calls 7SP to reread or rewrite the data one word at a time and compares previously read data with the new data. Routine 7SP calls 7MP to issue intermediate ECS error message whenever the data read does not compare. After issuing the error message, 7MP recalls 7SP to continue the single word read or write error recovery process. Once 7SP completes its single word reads or writes, it recalls 7DP. Routine 7DP issues another DEPM and calls 7EP which issues a final error message for the read or write function, showing an unrecovered or recovered status for the entire operation.

Figure 7-13 illustrates the preceding process.

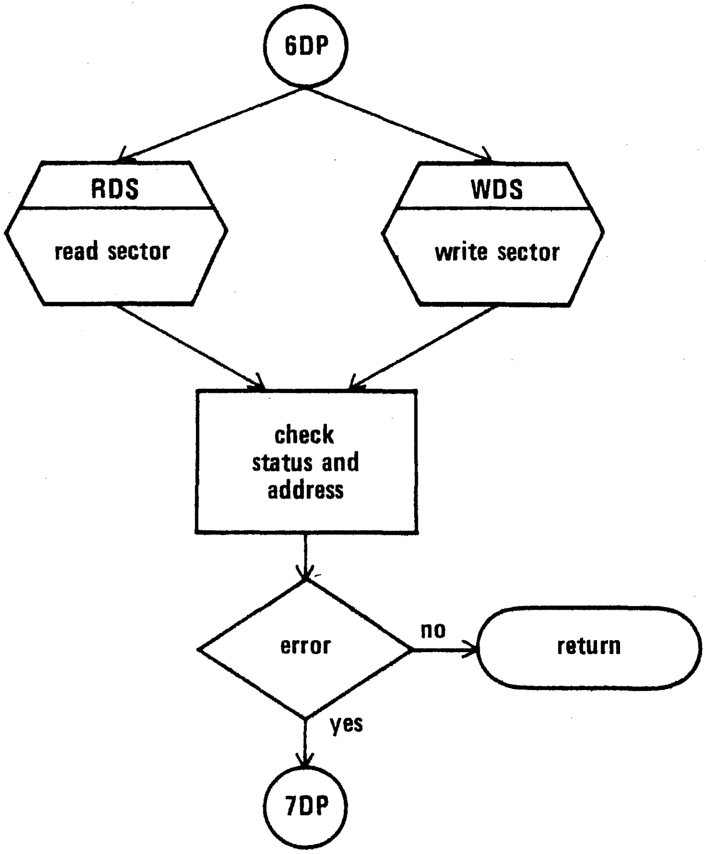


Figure 7-13. 6DP DDP/ECS Driver

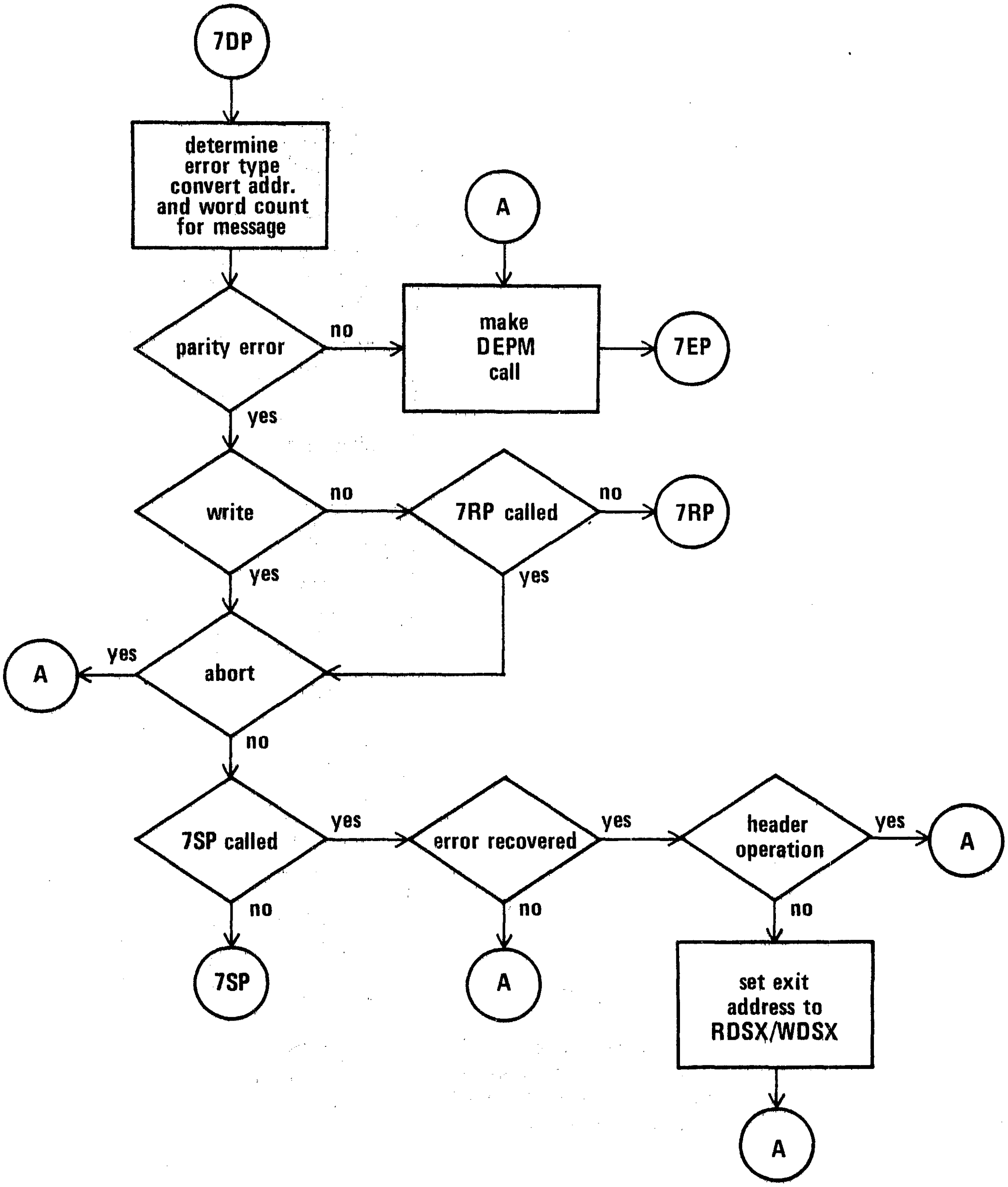
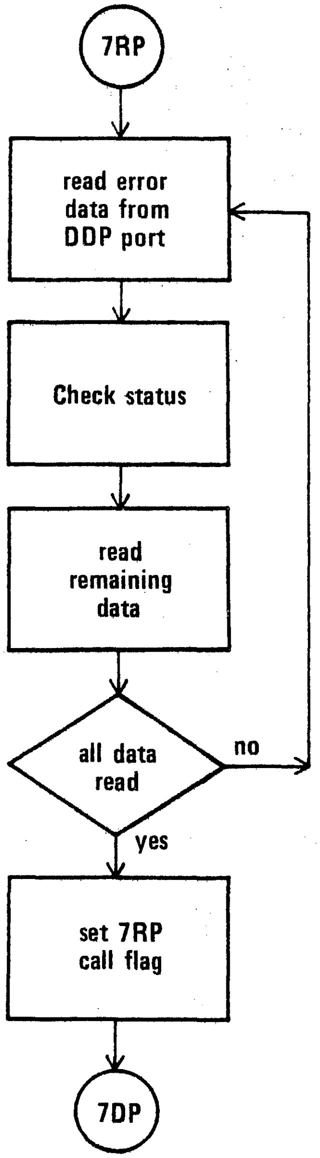


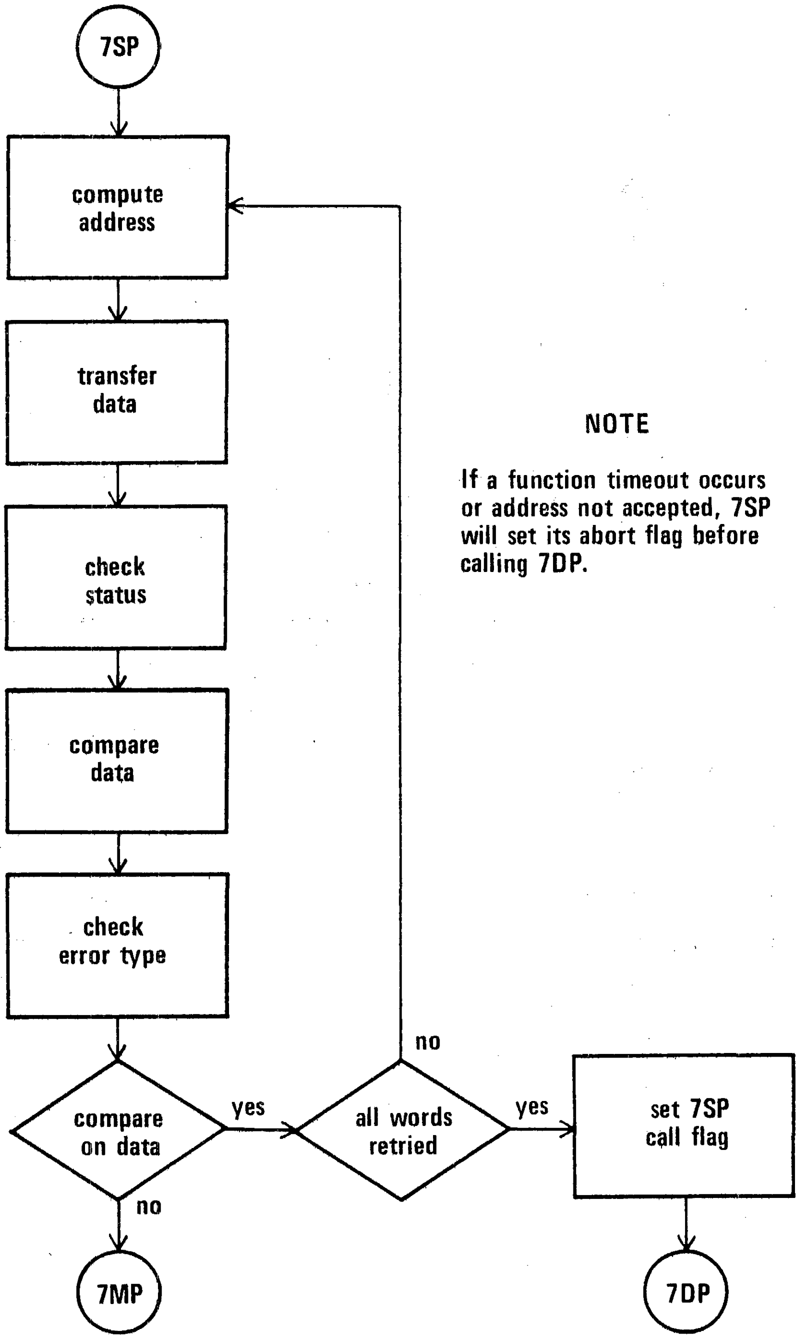
Figure 7-13. 6DP DDP/ECS Driver (Continued)



NOTE

If the channel was previously active, or a function time out occurs, 7RP will set its abort flag before calling 7DP.

Figure 7-13. 6DP DDP/ECS Driver (Continued)



NOTE

If a function timeout occurs or address not accepted, 7SP will set its abort flag before calling 7DP.

Figure 7-13. 6DP DDP/ECS Driver (Continued)

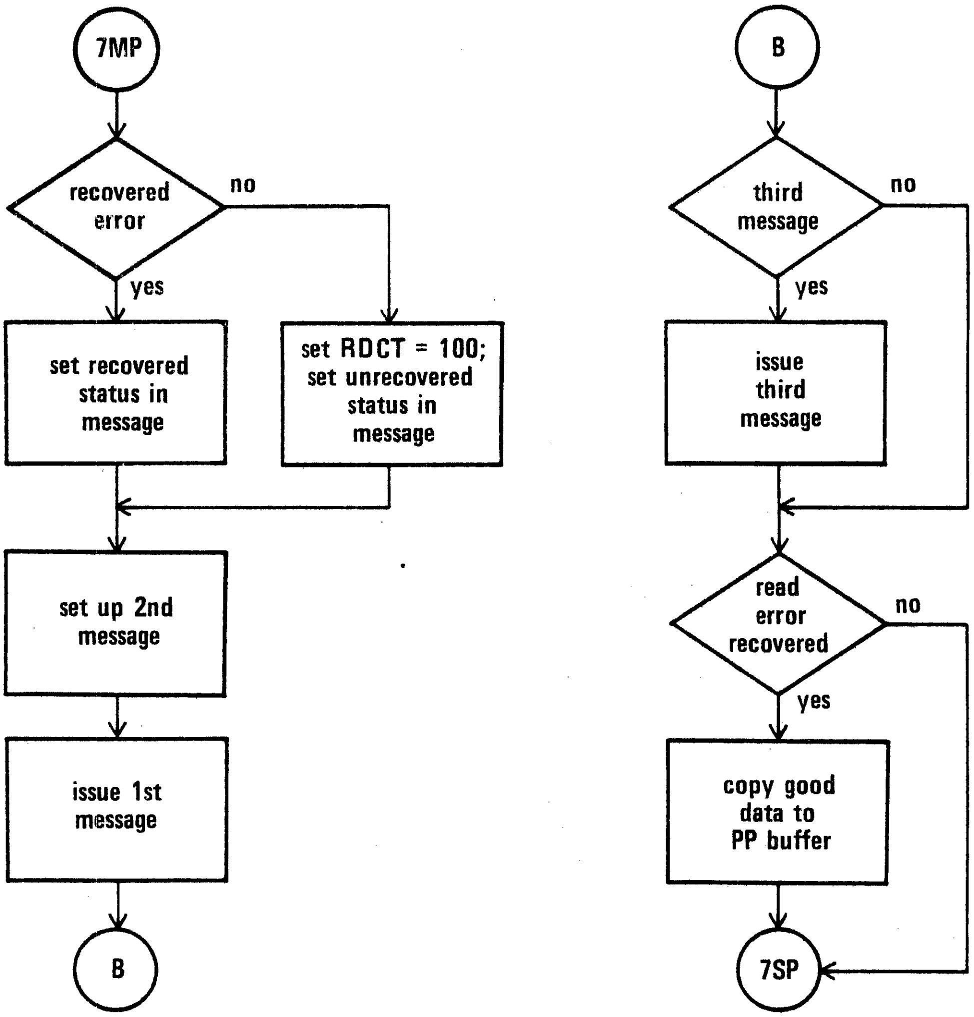


Figure 7-13. 6DP DDP/ECS Driver (Continued)

The initialization and recovery of mass storage devices is controlled by three routines: recover mass storage (RMS), check mass storage (CMS), and system recovery processor (REC). All mass storage devices have a label which contains information identifying the equipment and usages of it such as pack name, device number, and device masks. The usages of a mass storage device are defined through an initialization and these attributes remain with the device until the device attributes are redefined by subsequent initializations.

This section deals with the recovery and initialization of mass storage under NOS 1, and it is assumed that the reader is familiar with NOS mass storage concepts and the format of mass storage tables. The reader should refer to the NOS Installation Handbook, Sections II and IV, for review of the basic mass storage concepts.

MASS STORAGE MANAGER

The recovery of mass storage devices is performed by mass storage manager (MSM). MSM routines build the mass storage tables (MST) for all mass storage devices introduced to the system during deadstart or on-line. The two major routines in MSM - CMS, which controls the on-line device operations, and RMS, which handles the deadstart operations - read the label of each mass storage device and enter appropriate information from the label into the MST/TRT for the device. Since an attempt to read and recover the information contained in the device label is always done, the manipulations done by MSM are generally called recovery. The initialization of a mass storage device is then a subset of device recovery. A device recovery implies that all the information contained in the device label is transferred to the device's MST/TRT; a device initialization may alter some or all of the label information when building the device's MST/TRT.

The contents of the label track for a device are defined in common decks COMSLSD and COMSDSL. The format of a mass storage device label track and label sector are shown in Section 2. The format of the MST is also shown in Section 2 and its contents are defined in PPCOM.

INITIALIZATION AND RECOVERY ROUTINES

RECOVER MASS STORAGE (RMS)

RMS is the deadstart portion of the MSM. It surveys all defined mass storage equipments and attempts to recover them.

RMS is activated by STL as part of the deadstart process with the input to RMS being the level of deadstart recovery selected to be performed. With the deadstart level and information entered into skeletal MSTs from CMRDECK processing, the recovery of mass storage devices may be done.

A flowchart of RMS is shown in Figure 8-1. There are four basic phases to RMS: preset, read device labels, check and recover devices, and call REC into execution.

Preset

The preset phase of RMS scans the equipment status table (EST) for mass storage devices, building two tables: table of equipments to recover (TREC) and table of CM addresses for MST of first unit in the equipment (TEQP). The address of dayfiles to recover is also set at this time. If the system being deadstarted is part of a multimainframe (MMF) complex, the link device is examined for the existence of this machine's ID and its condition in order to determine whether this machine can be recovered or introduced to the MMF complex.

Read Device Labels

Before any recovery (or initialization) can be performed, an attempt is made to read each device's label. This is done by routine read device label (RDL). The flowchart for RDL is shown in Figure 8-2. RDL is also called by CMS, the on-line portion of MSM. If the attempt to read the labels from each unit comprising the device is not successful, a label error status (STLE) is set into the MST for the device for later processing. If the read is not successful because a unit is not ready, a not ready status (STNR) is set into the MST if the device being recovered is removable. The attempt to read the device label is not made by RMS for devices with a total initialize requested.

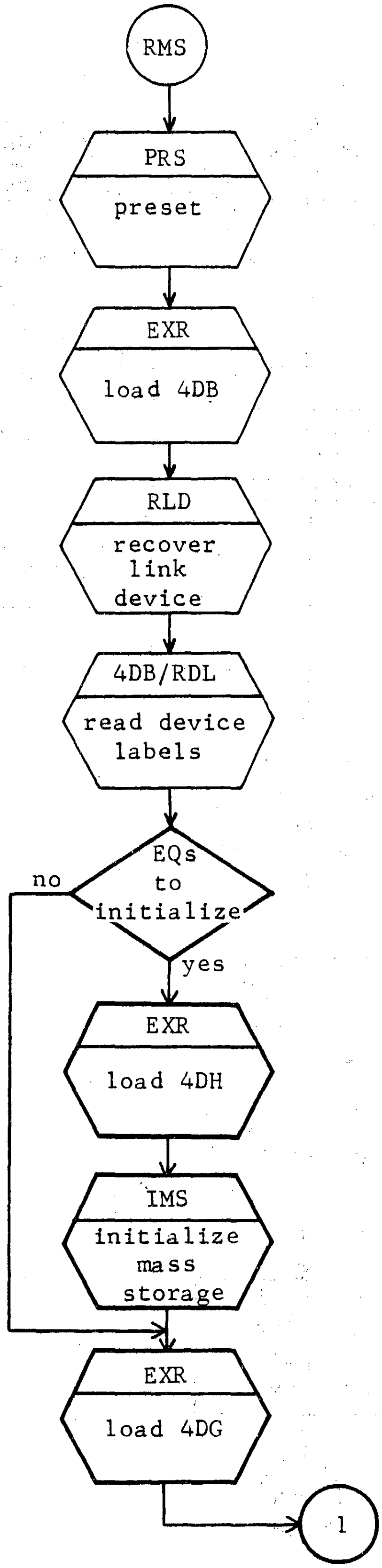


Figure 8-1. Recover Mass Storage (RMS)

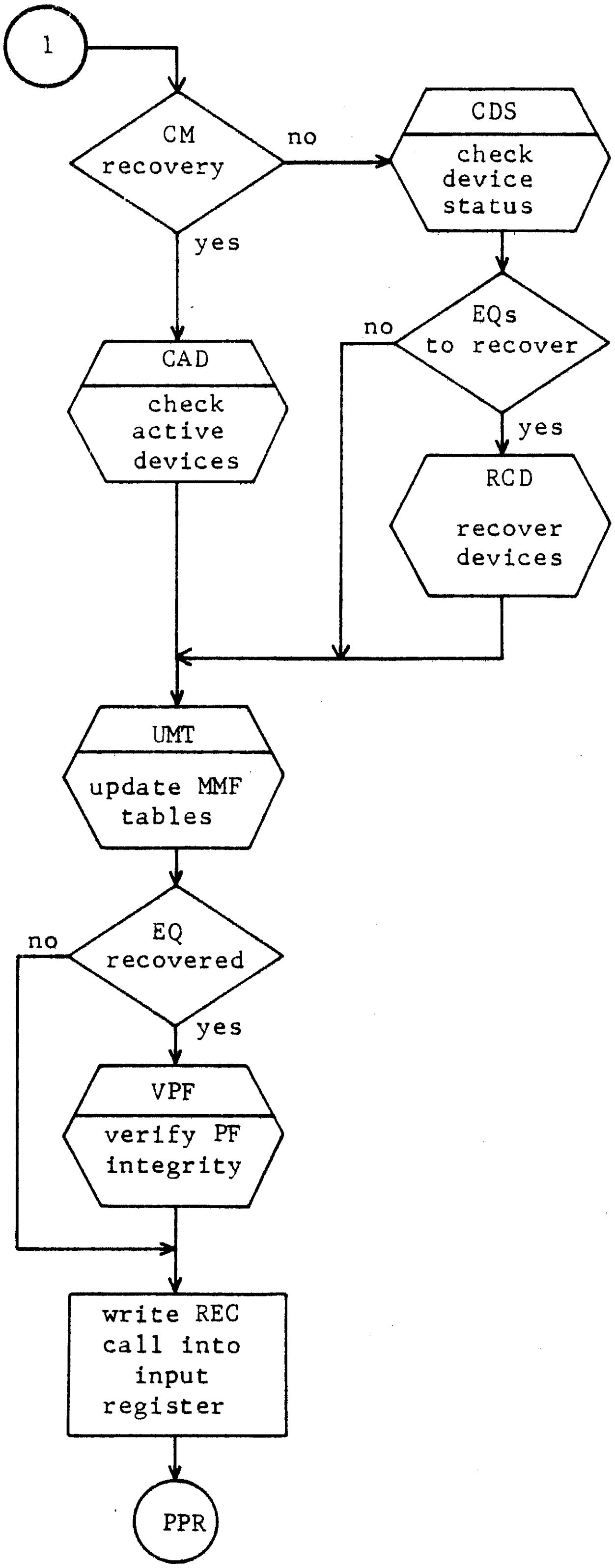


Figure 8-1. Recover Mass Storage (RMS) (Continued)

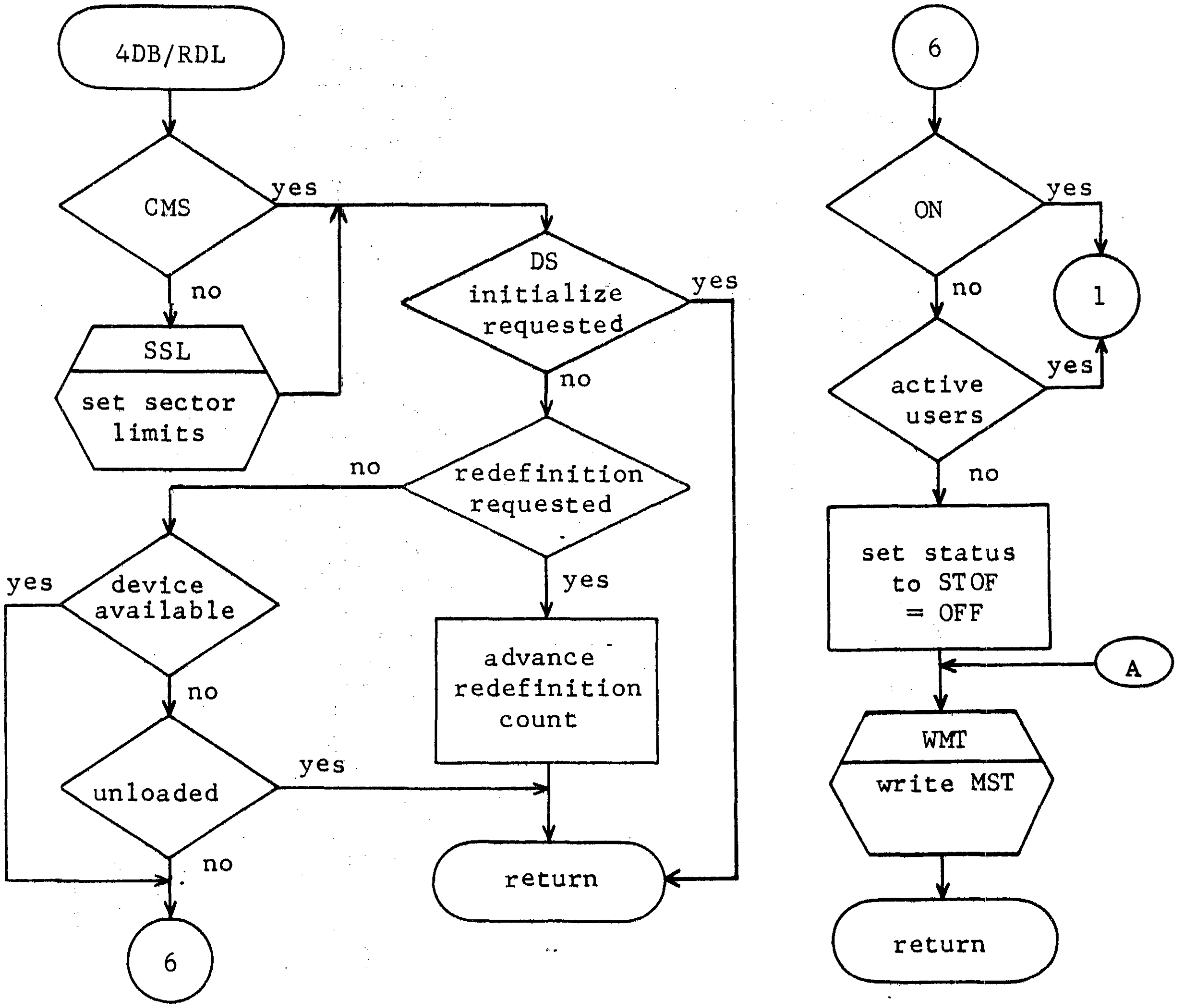


Figure 8-2. Read Device Labels (RDL)

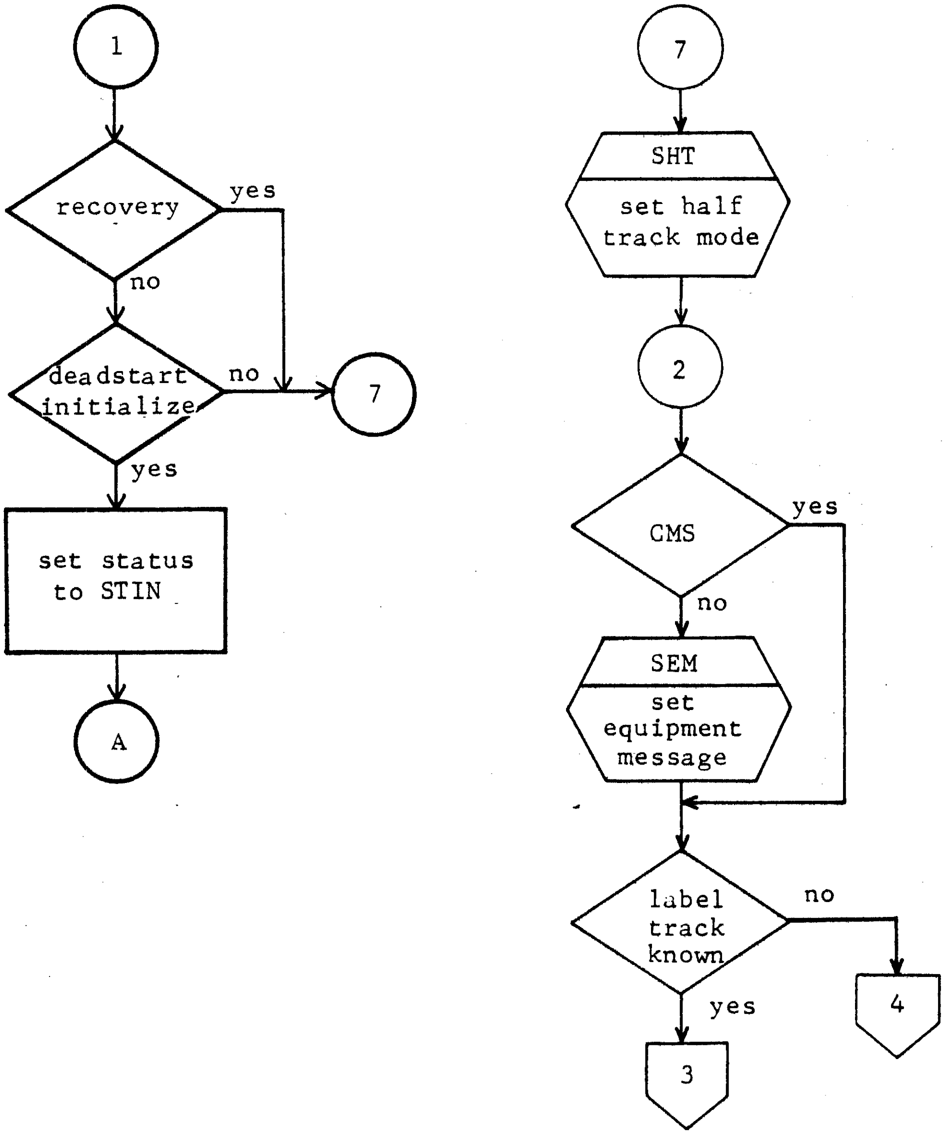


Figure 8-2. Read Device Labels (RDL) (Continued)

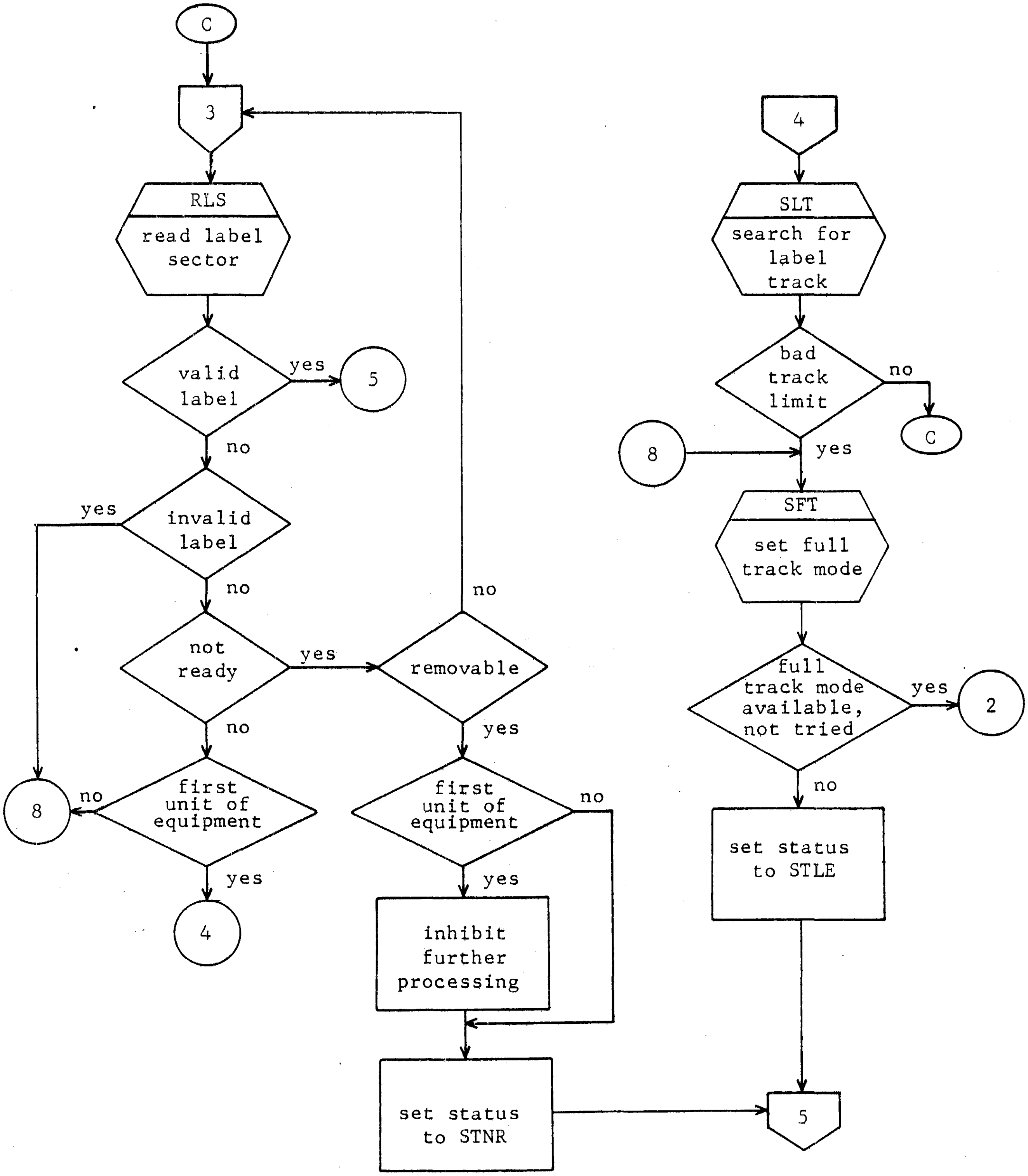


Figure 8-2. Read Device Labels (RDL) (Continued)

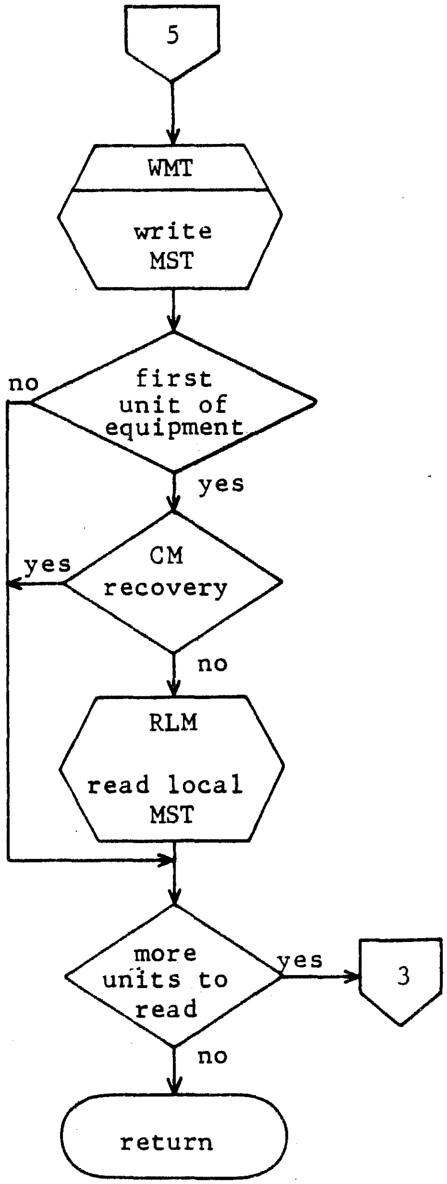


Figure 8-2. Read Device Labels (RDL) (Continued)

Check and Recover Devices

Upon the reading of device labels, the mass storage devices are recovered from their active states (level 3 deadstart) or from their label information (level 0, 1, and 2 deadstarts). Subroutine check active devices (CAD) recovers mass storage devices from their central memory MST/TRT. Figure 8-3 contains the flowchart for CAD. Subroutine check device status (CDS) controls the recovery of the mass storage devices using the information read from the device labels in conjunction with information entered when processing the CMRDECK. Refer to figure 8.4.

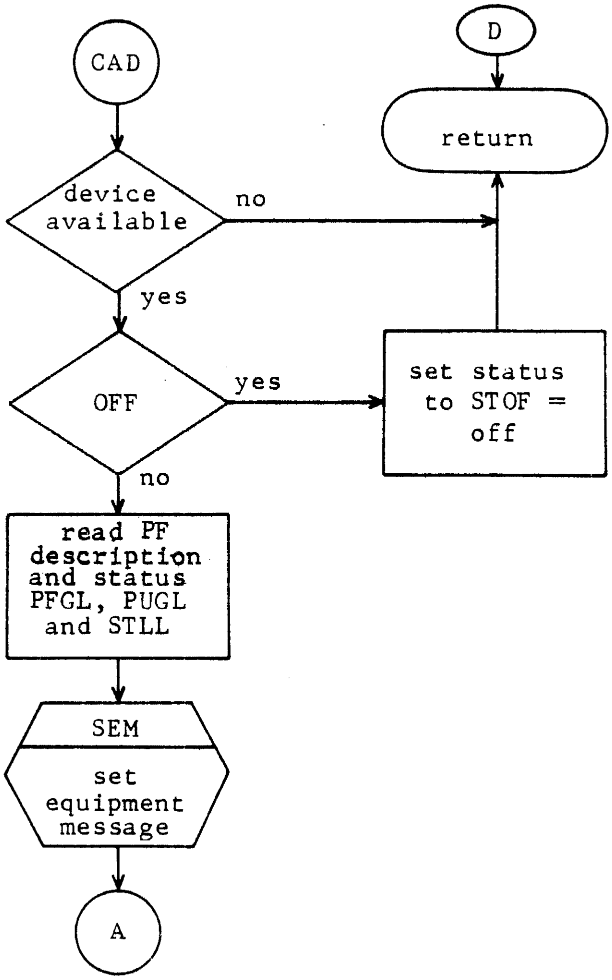


Figure 8-3. Check Active Devices (CAD)

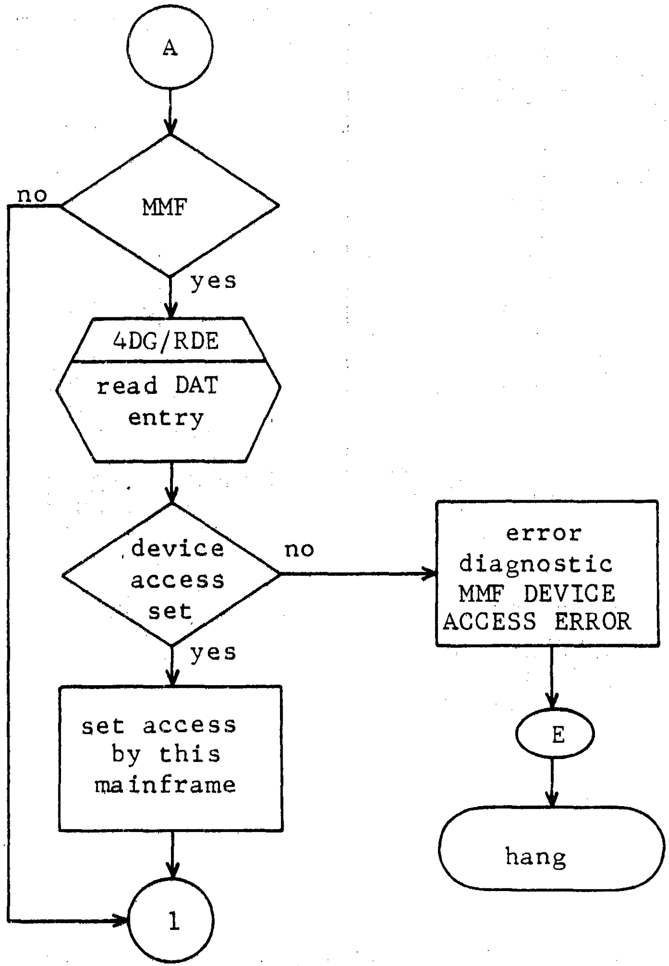


Figure 8-3. Check Active Devices (CAD)
(Continued)

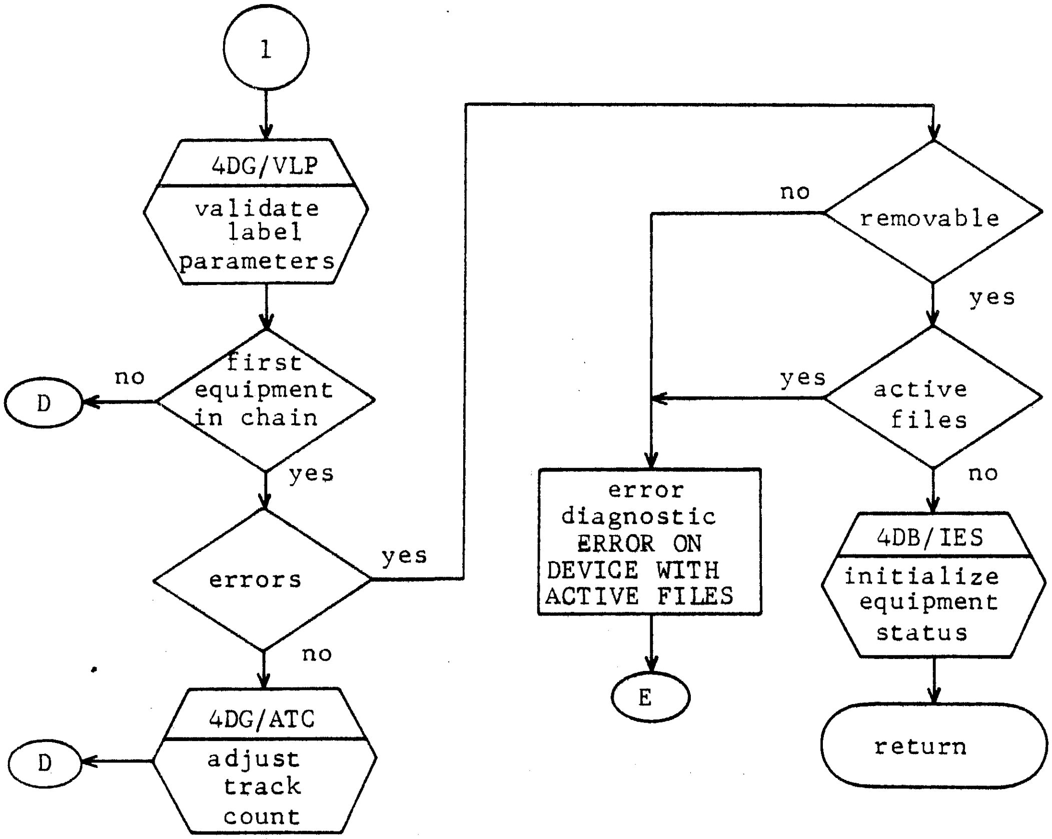


Figure 8-3. Check Active Devices (CAD)
(Continued)

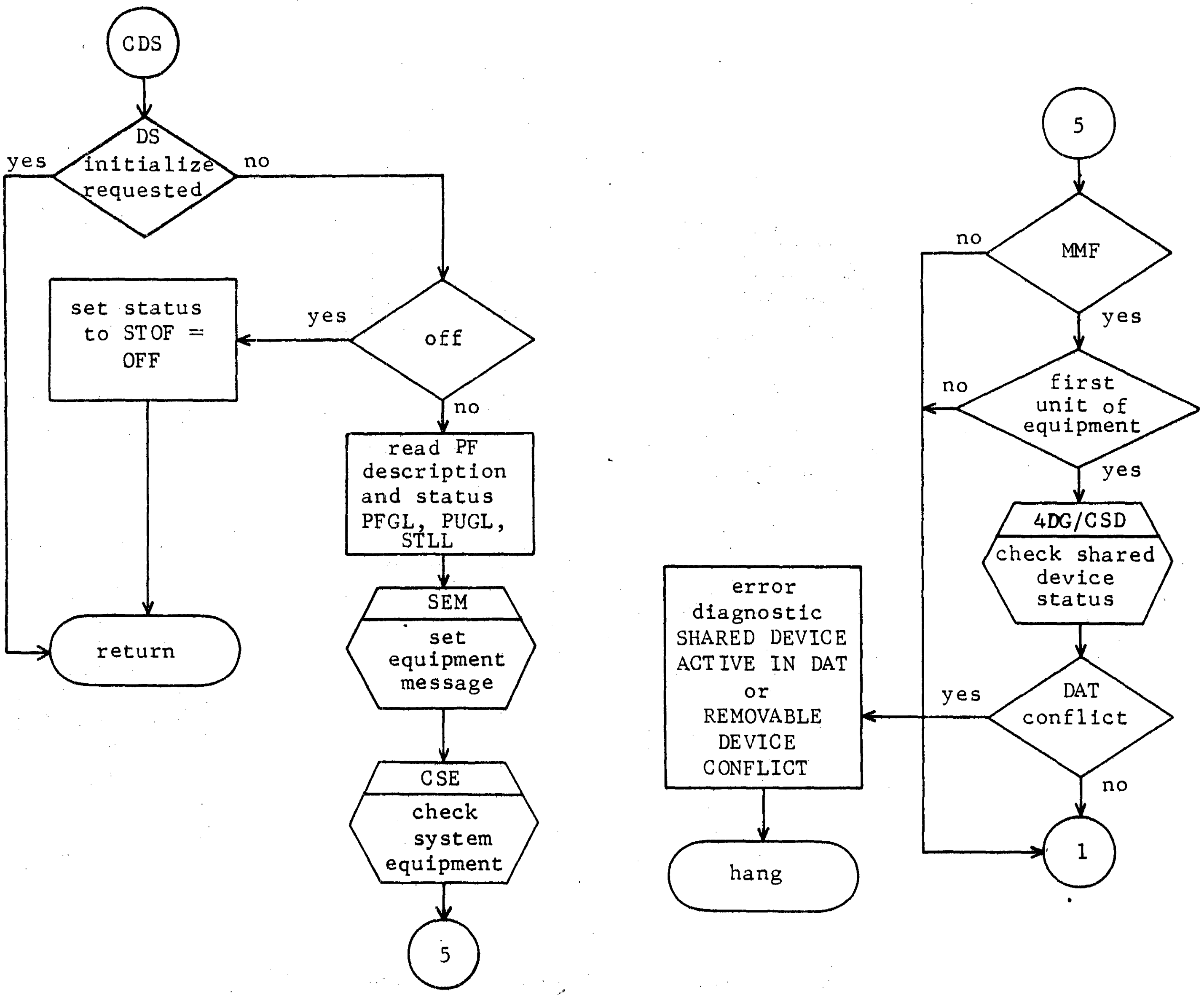


Figure 8-4. Check Device Status (CDS)

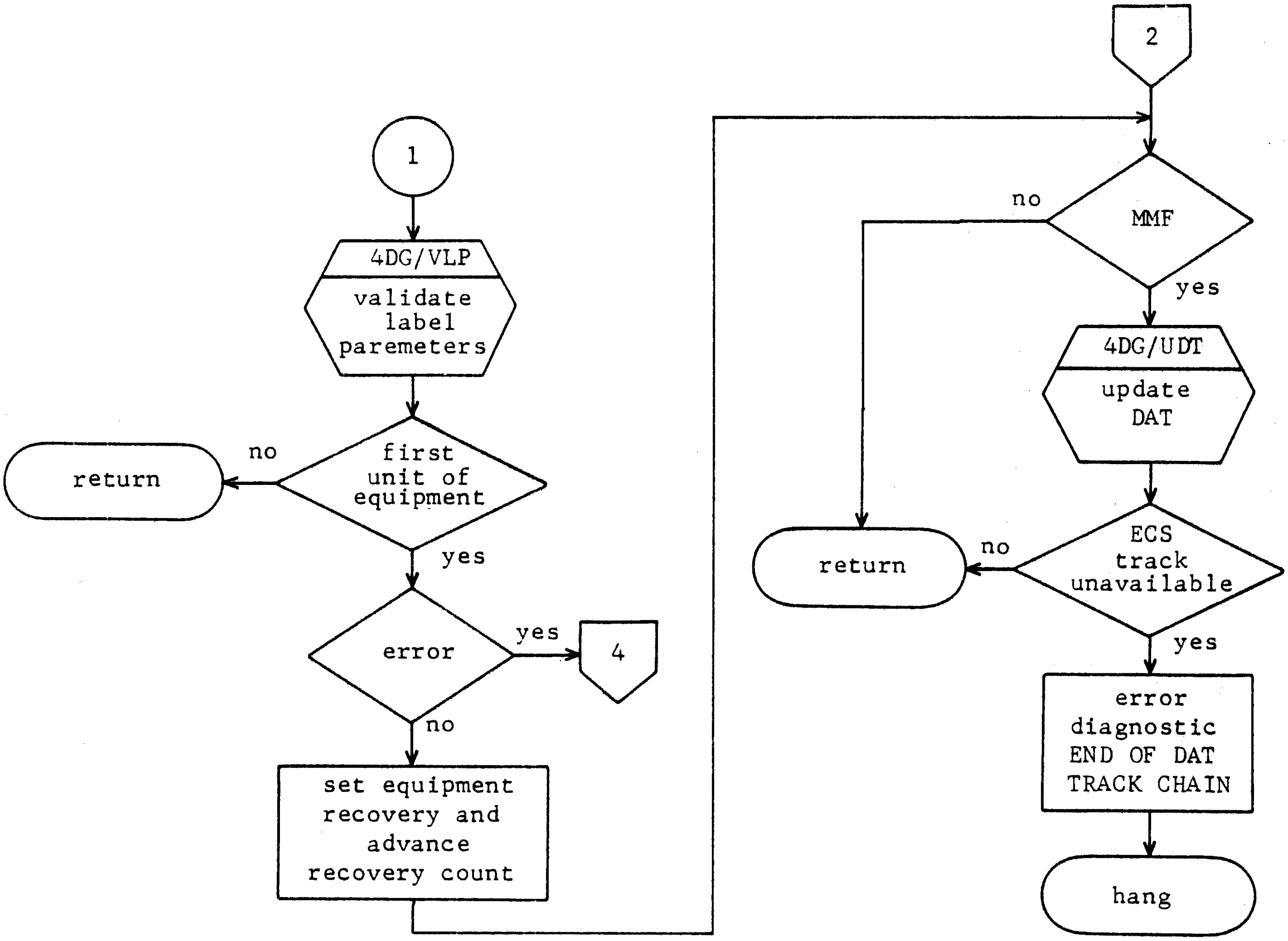


Figure 8-4. Check Device Status (CDS)
(Continued)

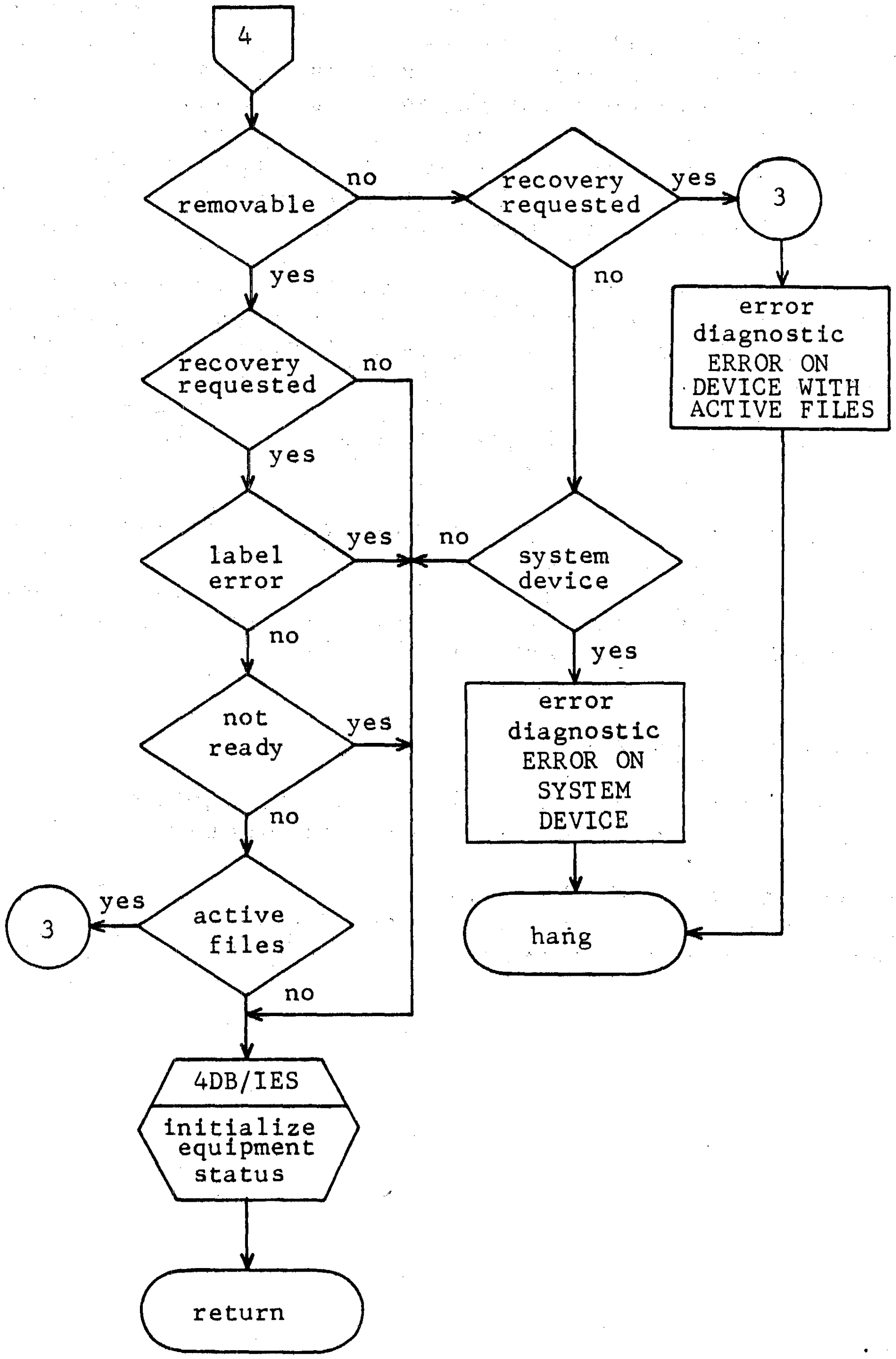


Figure 8-4. Check Device Status (CDS)
(Continued)

Mass storage devices are recovered after executing CDS by subroutine recover devices (RCD), provided that there are mass storage devices to be recovered. RCD recovers the TRT by either copying it entirely or editing it. The editing phase recovers only preserved and flawed tracks, making all other tracks available for system usage regardless of their previous reservations. The editing phase occurs only on a level 0 deadstart. Figure 8-6 is the flowchart for RCD. RCD also initiates the recovery of preserved system dayfiles by invoking subroutine chase dayfile chain (CDC), which recovers the dayfile by reading the mass storage device upon which it resides.

At this point, the mass storage tables have been recovered either from their CM area or from the labels on the mass storage devices. If the system being deadstarted is part of a multimainframe configuration, the MSTs for shared devices are updated in the link device. The details of mass storage recovery for an MMF configuration are described later in this section.

If any equipments have been recovered, a verification of the permanent file subsystem is done. This verification, performed by verify permanent files (VPF), protects against duplicates in pack/family names and device mask bits (master devices) and device numbers within the same family.

Call REC Into Execution

Having completed this phase of mass storage recovery, the remainder of the system deadstart can proceed.

RMS writes a call for PP routine REC into its input register and jumps to PP resident, allowing REC to be loaded and to continue the system recovery process.

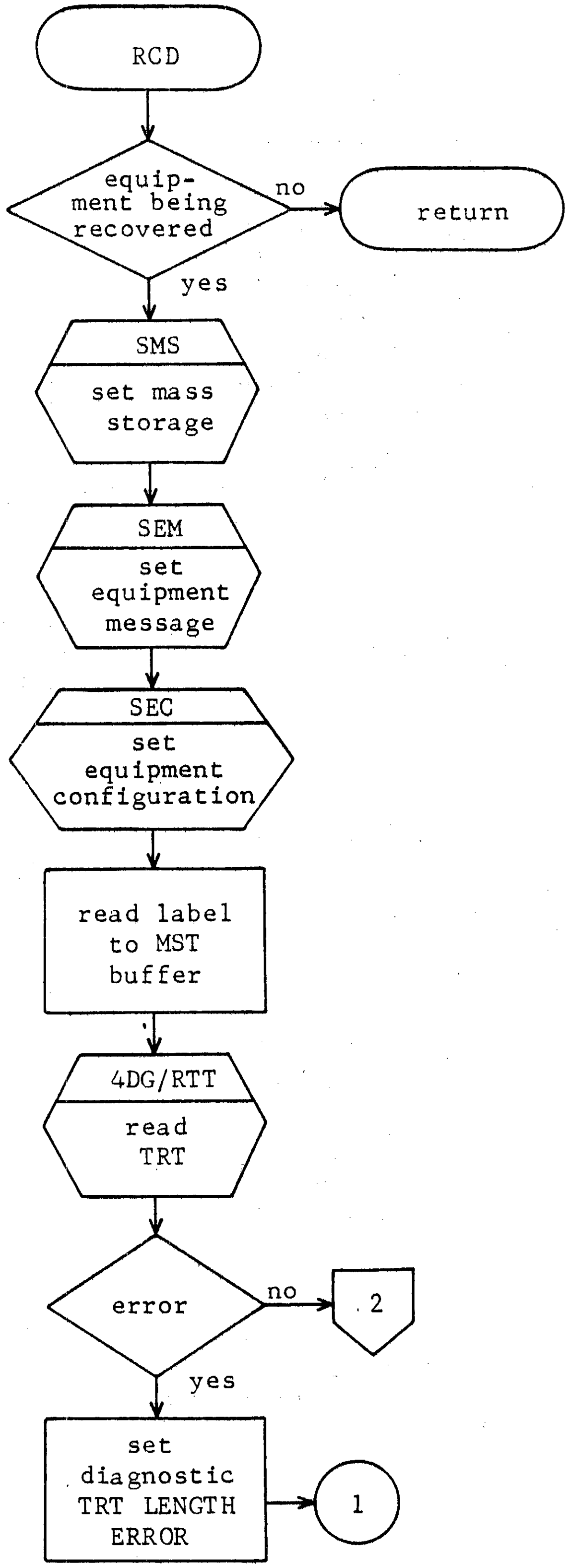


Figure 8-6. Recover Devices (RCD)

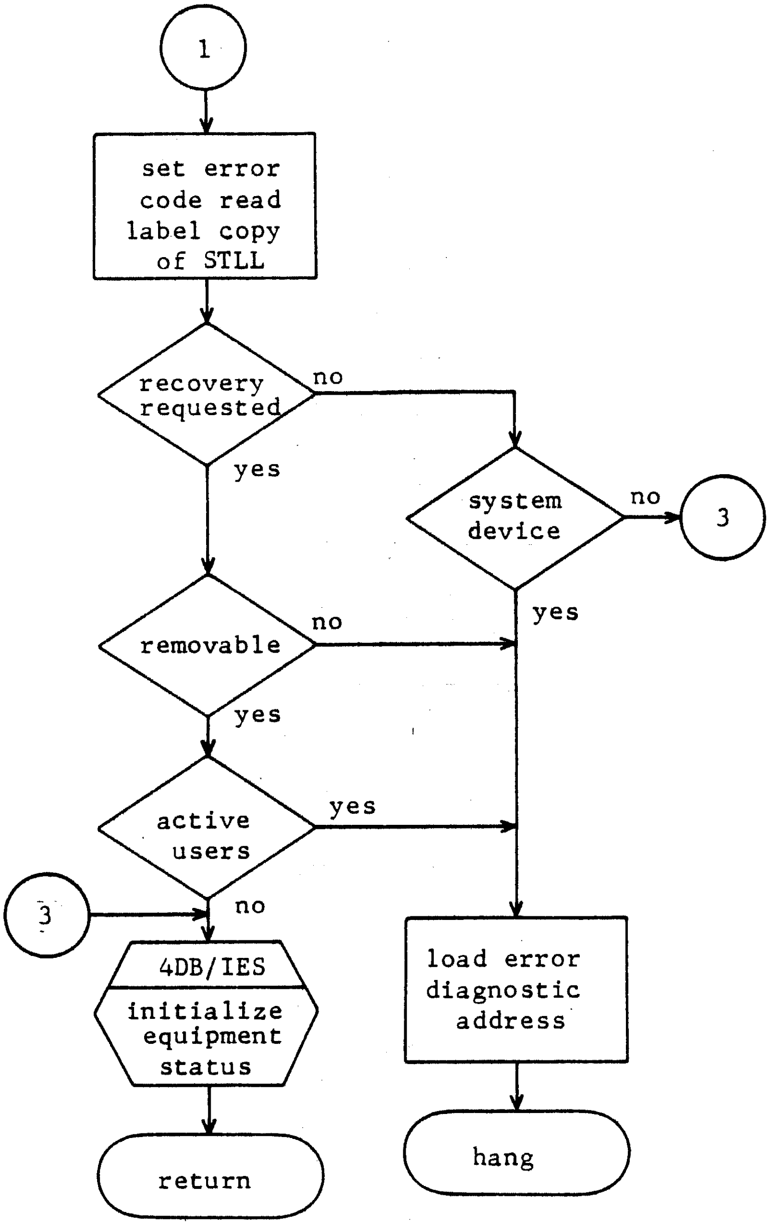


Figure 8-6. Recover Devices (RCD) (Continued)

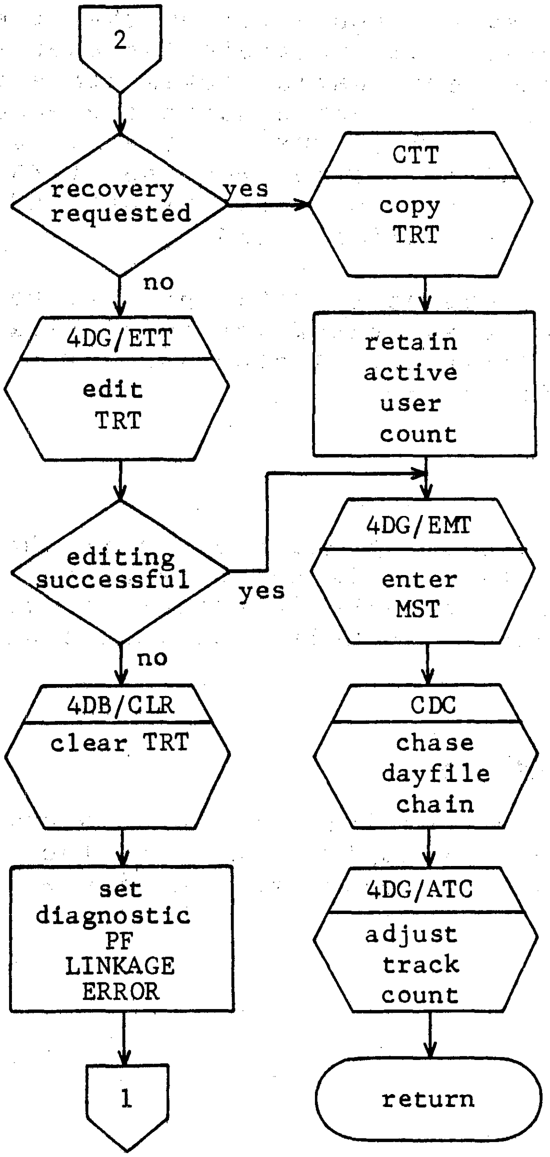


Figure 8-6. Recover Devices (RDC)
(Continued)

CHECK MASS STORAGE (CMS)

CMS is the on-line portion of the MSM. It surveys all defined mass storage devices and verifies that the proper devices are mounted or makes them available for user access if possible. This survey and verification takes place on a periodic basis (60 cycles of 1SP) if removable packs are enabled (with DSD command or IPRDECK directive). CMS is also activated by the on-line mass storage initialization routines (IMS/MSI) after a device has been initialized on-line. UNLOAD and MOUNT commands will also cause 1DS to activate CMS.

There are five phases to CMS: preset, read device labels, check and recover devices, check for initialization requests, and count active families. A flowchart of the main routine of CMS is shown in figure 8-7.

Preset

The preset phase of CMS determines if CMS is being activated for the first time or is being recalled. If being recalled and PFNL is interlocked, the interlock is requested to be cleared. The EST is scanned to build the TREC table in the same manner as RMS.

Read Device Labels

The RDL routine contained in 4DB is modified to exclude portions of RDL that do not apply to on-line label recovery. RDL is then executed. RDL is flowcharted in Figure 8-2.

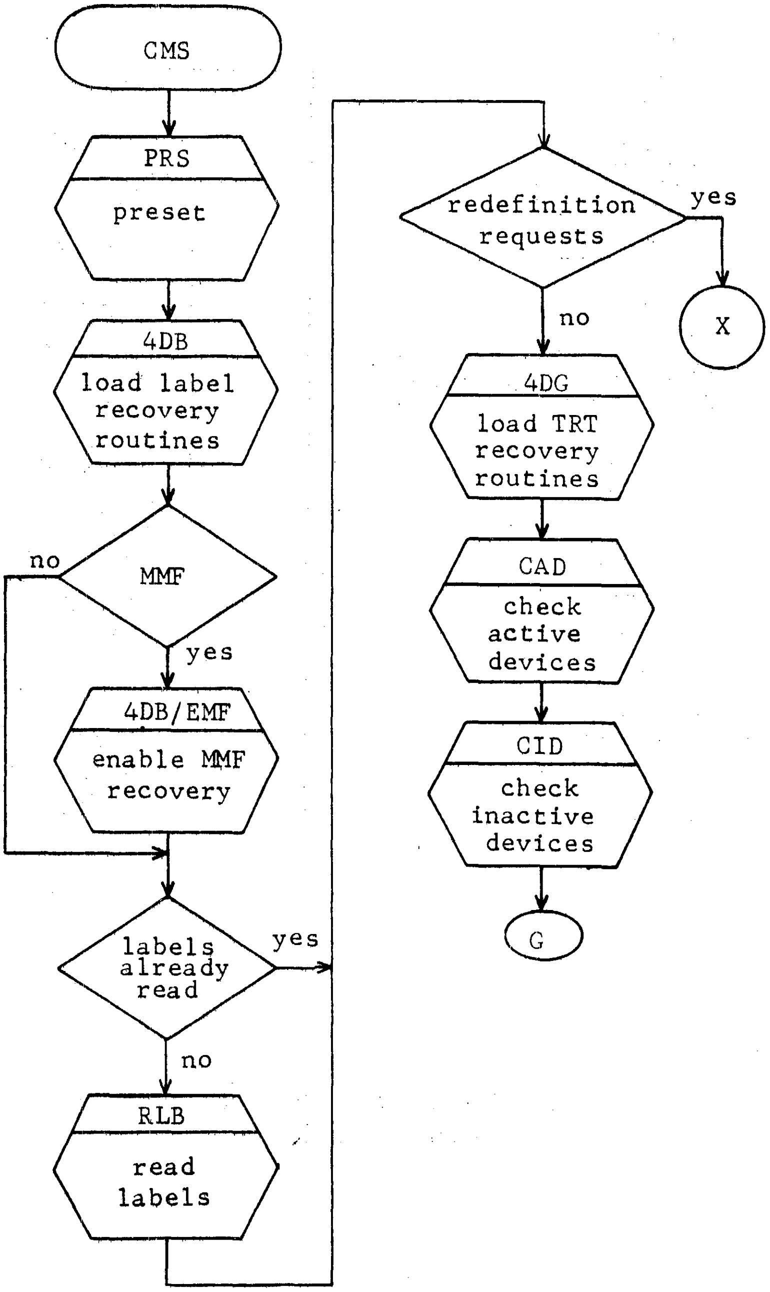


Figure 8-7. Check Mass Storage (CMS)

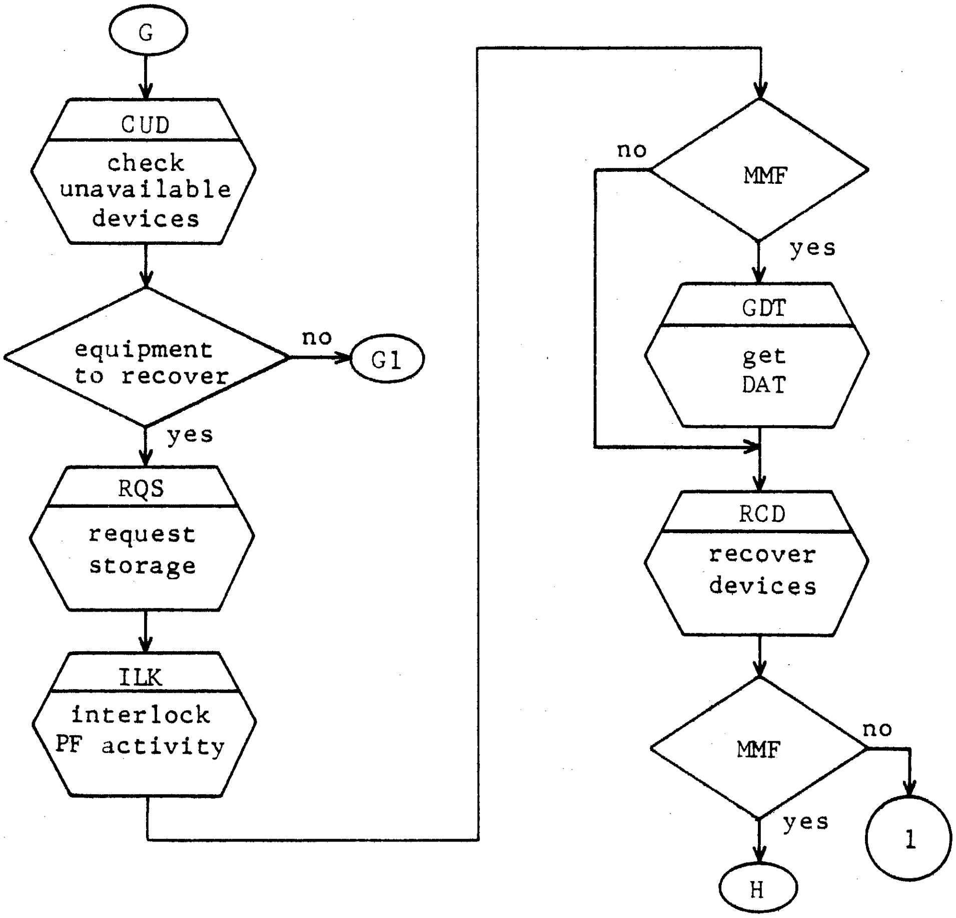


Figure 8-7. Check Mass Storage (CMS) (Continued)

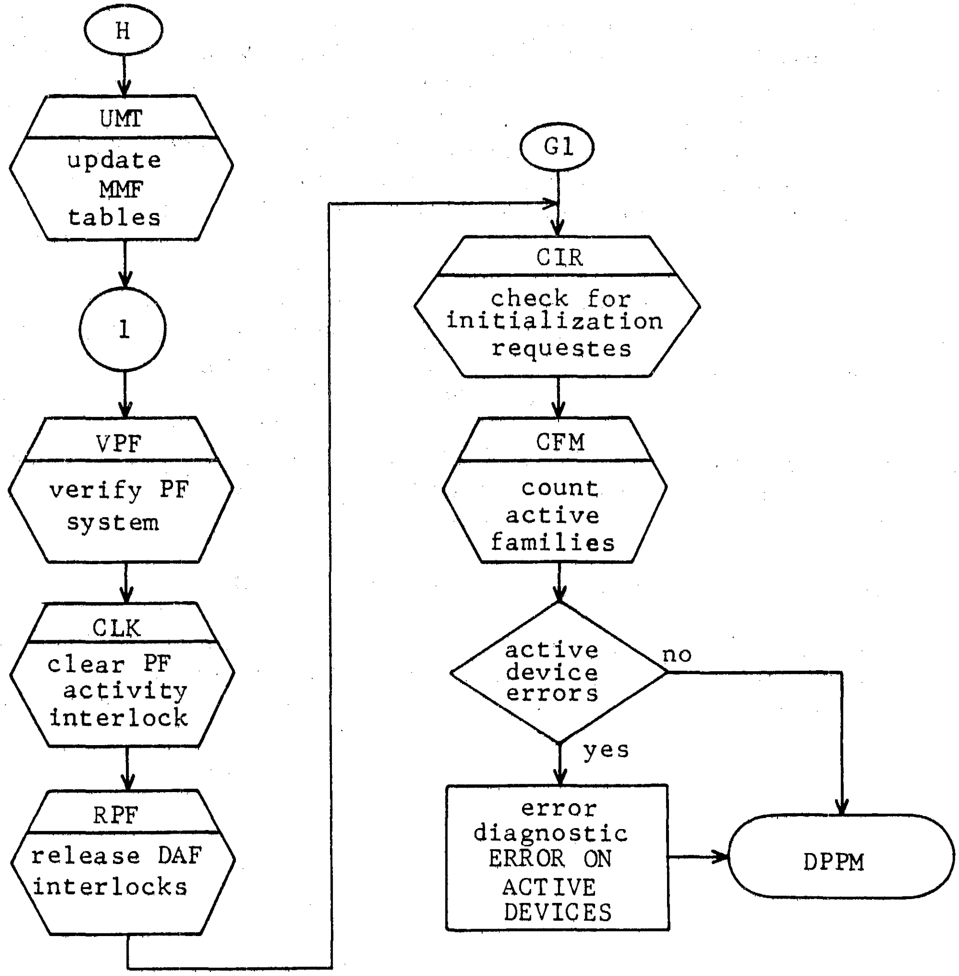


Figure 8-7. Check Mass Storage (CMS)
(Continued)

Check and Recover Devices

This phase of CMS validates the mounted mass storage by verifying active devices, checking inactive and unavailable devices, and recovering those devices that are as yet unrecovered. Subroutine CAD is concerned with those devices that are available, not being initialized, on or off with active users, not removable, or removable with active users, or checkpoint pending and not being unloaded. If the preceding properties are held by the device, then verify label parameters (VLP) is called to validate the device. All other devices are not considered to be active devices. Figure 8-8 is a flowchart of CAD.

Subroutine check inactive available devices (CID) is then executed to check those devices that were excluded by CAD and are removable. The MST for devices processed by CID is initialized and cleared of extraneous data. CID, basically, is the routine that cleans up the MST when a removable device is unloaded and restores the MST of invalid labeled devices to a skeletal and unavailable condition. Figure 8-9 is a flowchart of CID.

Subroutine check unavailable devices (CUD) processes those devices not previously validated by CAD or not unloaded by CID (these should be all devices not active and verified that have an unavailable status). CUD determines if there are any devices to recover. A flowchart of CUD is shown in figure 8-10.

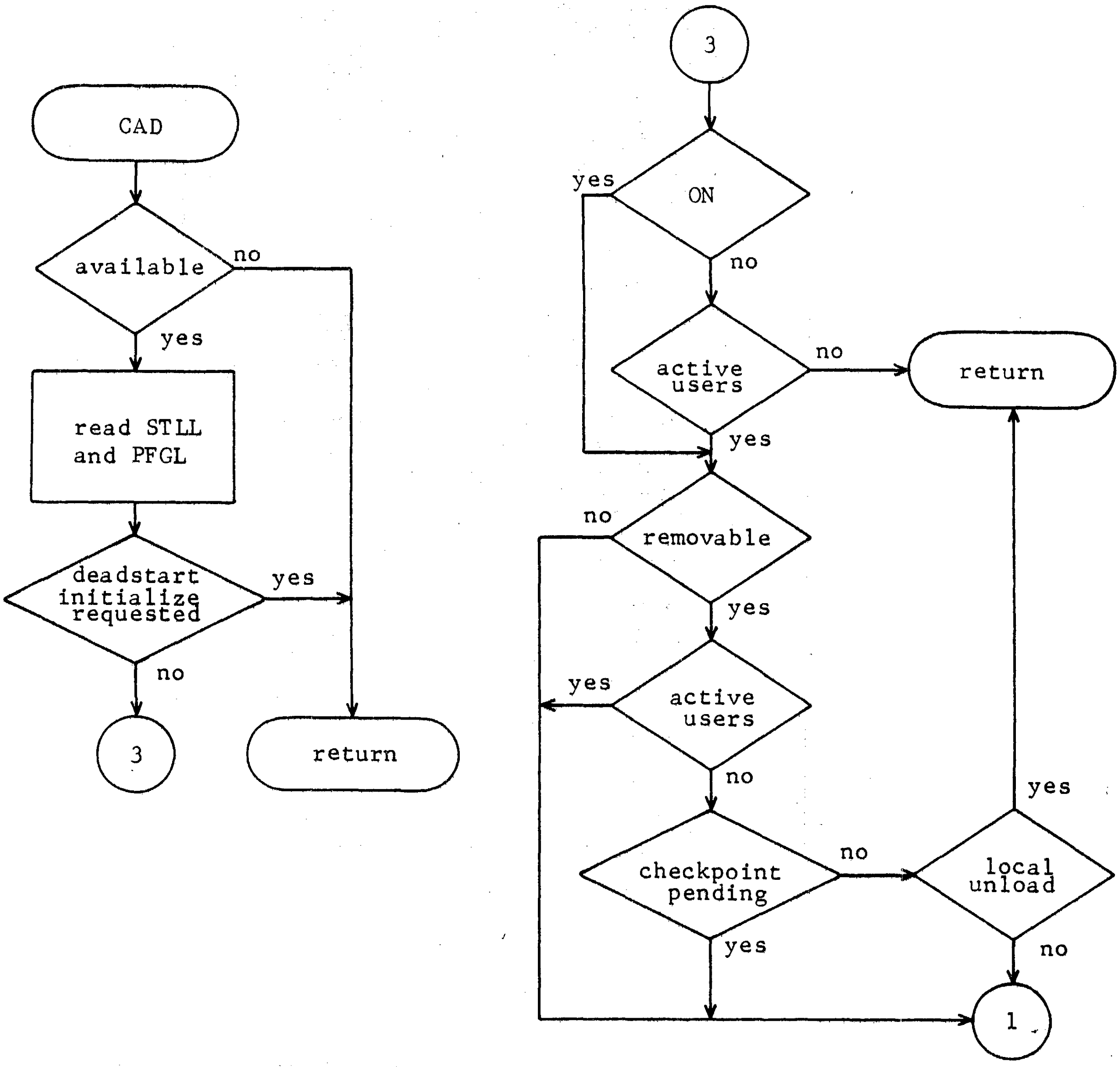


Figure 8-8. Check Active Devices (CAD)

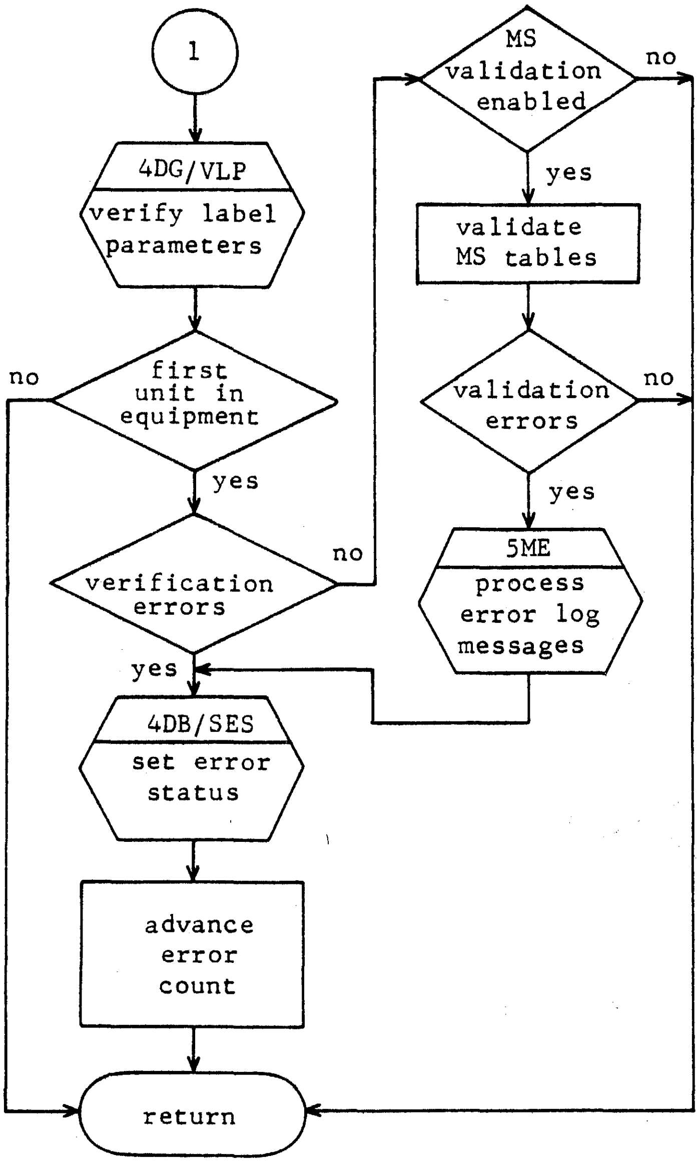


Figure 8-8. Check Active Devices (CAD) (Continued)

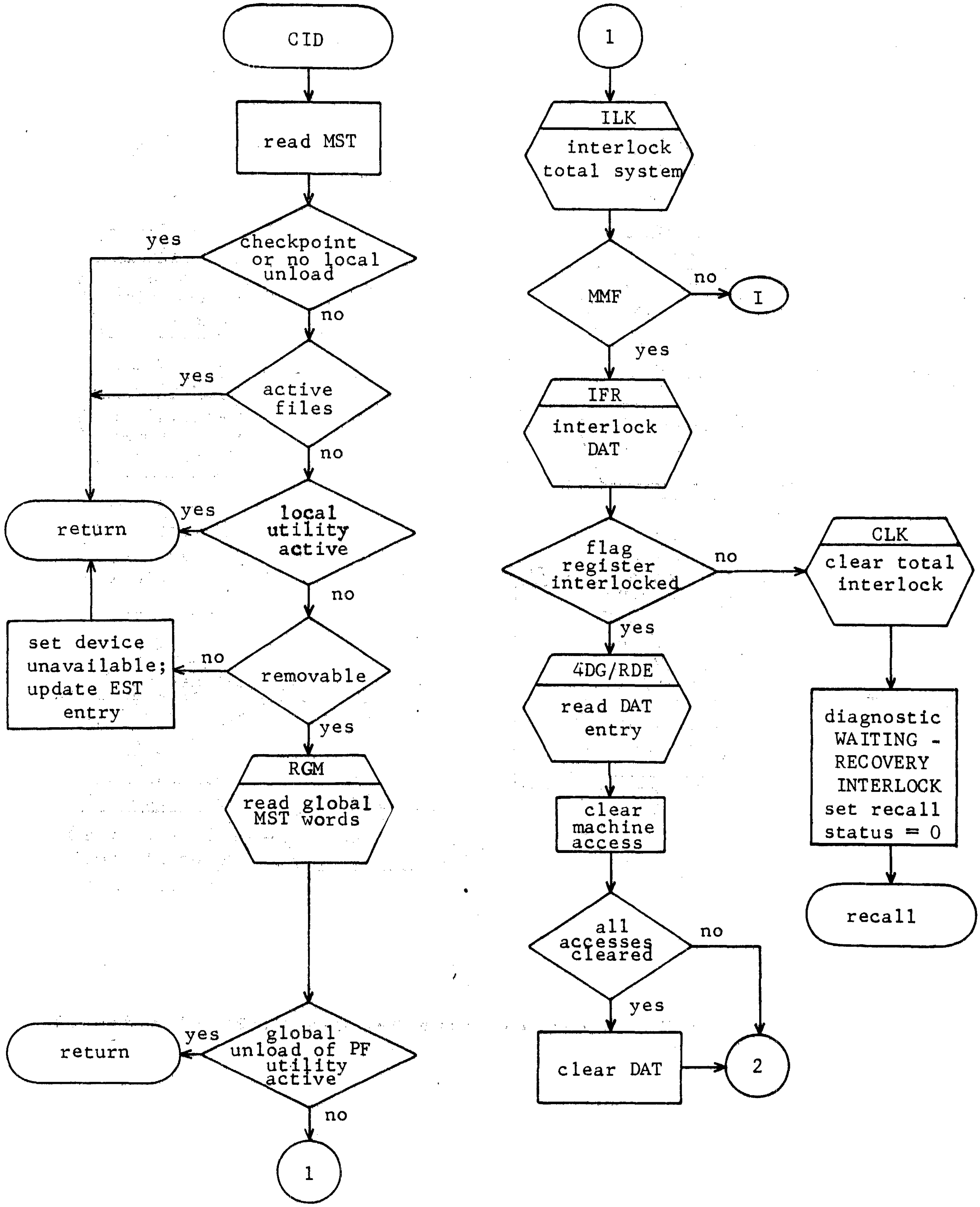


Figure 8-9. Clear Inactive Devices (CID)

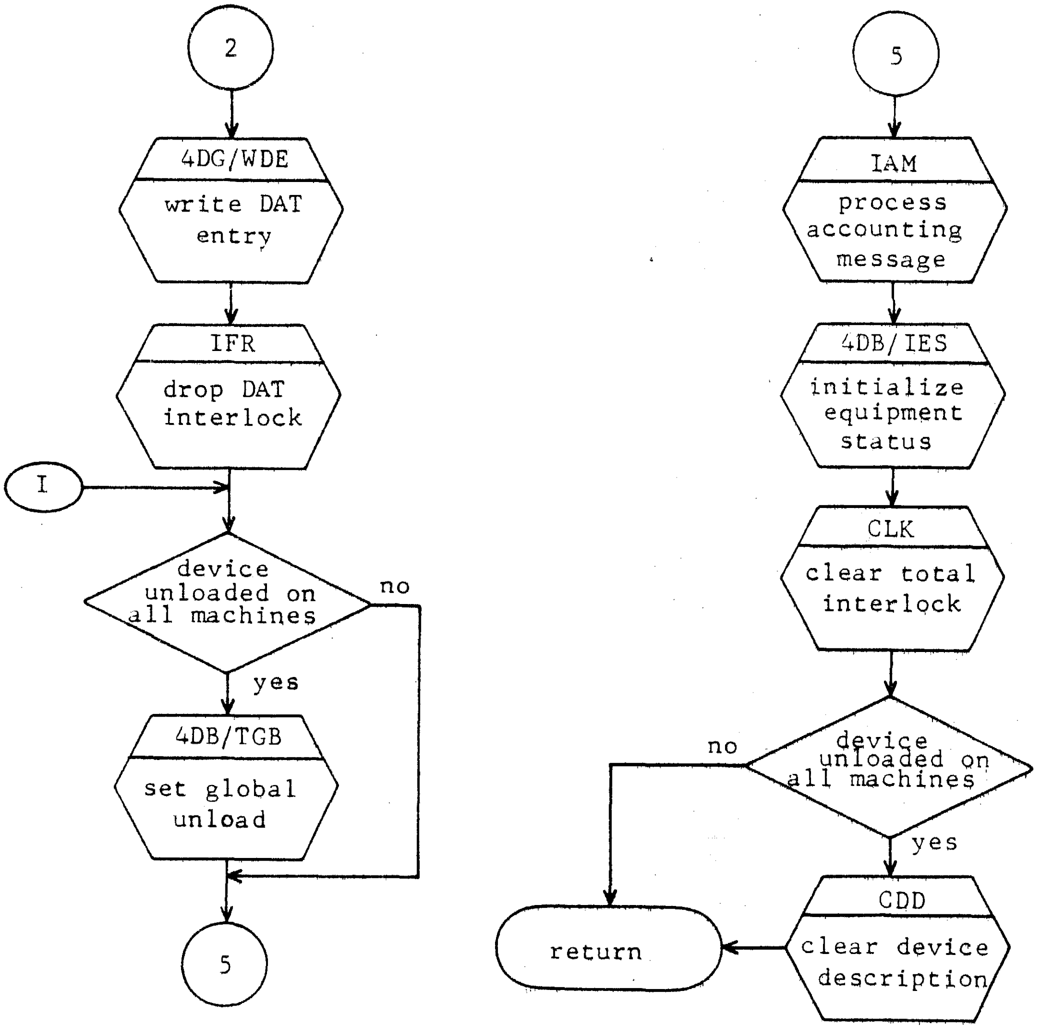


Figure 8-9. Clear Inactive Devices (CID) (Continued)

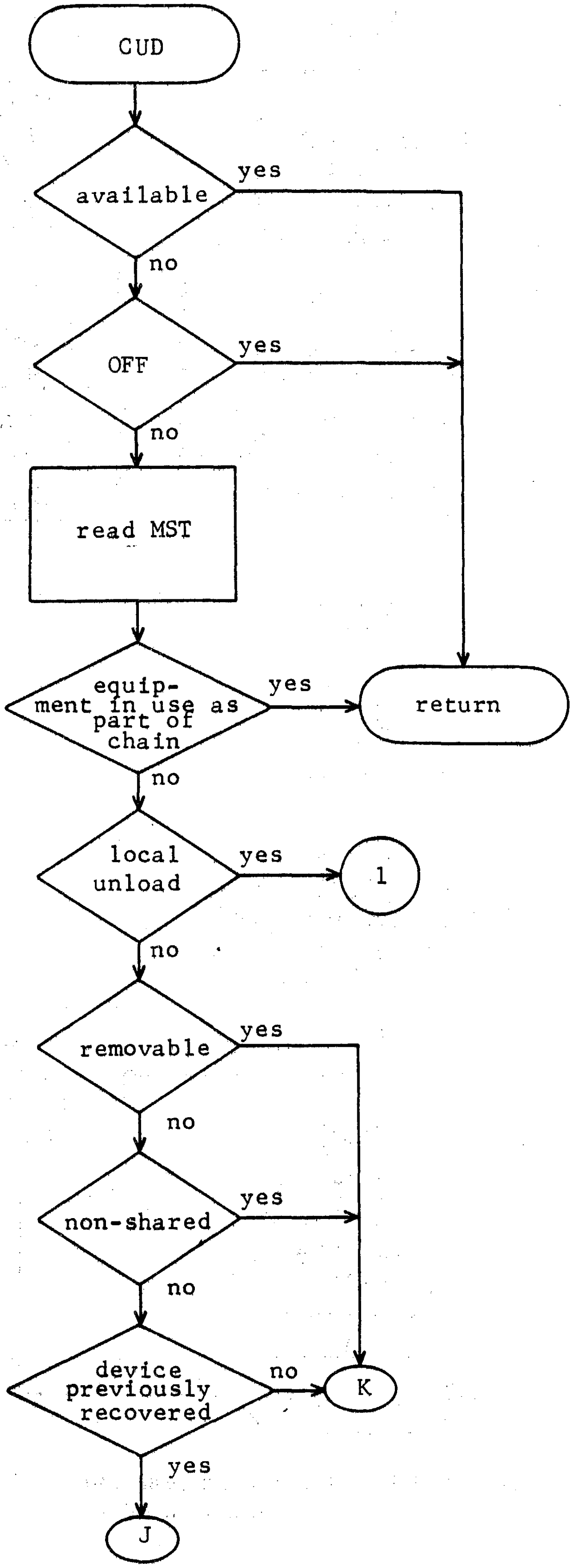


Figure 8-10. Check Unavailable Devices (CUD)

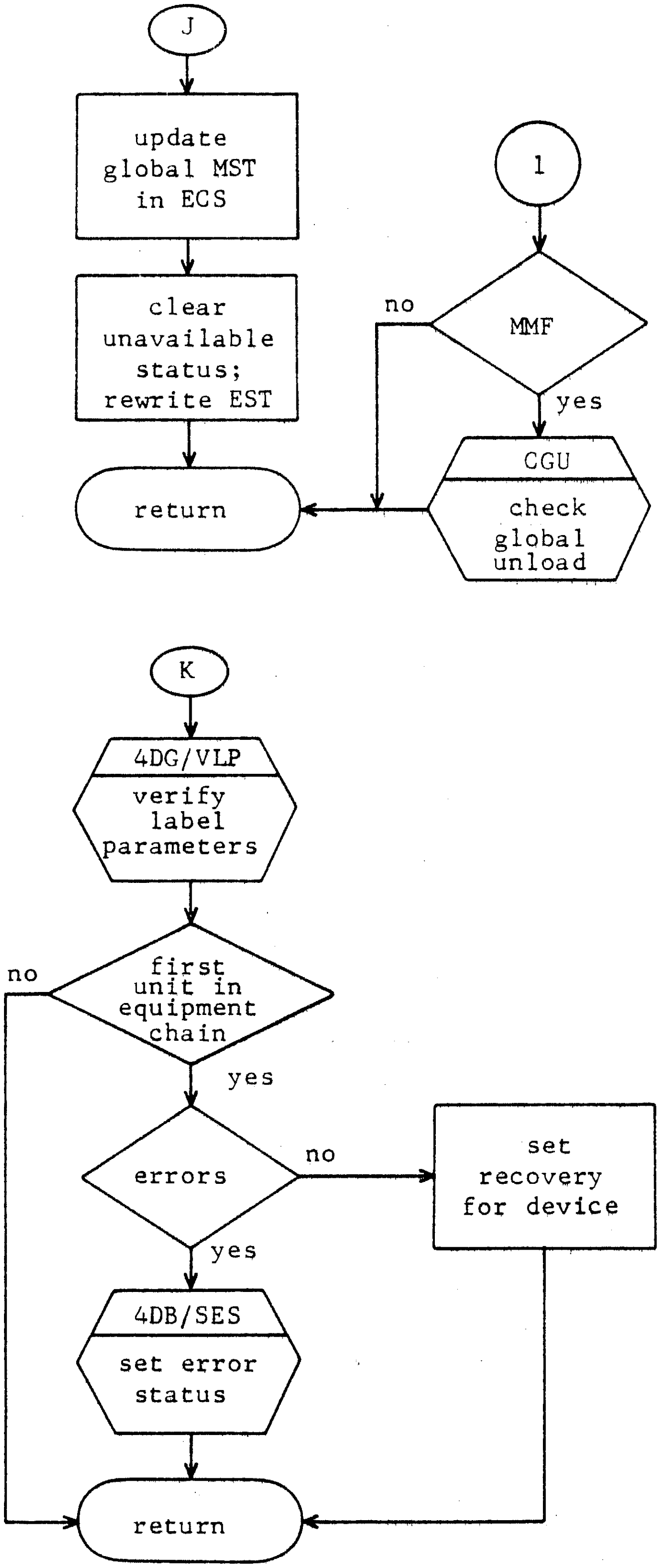


Figure 8-10. Check Unavailable Devices (CUD) (Continued)

If after executing CAD, CID, and CUD, a device is to be recovered, subroutine RCD is called. Subroutine RCD performs the same function for CMS as the RMS subroutine. A verification of the permanent file subsystem is performed as in RMS.

Check for Initialization Requests

After recovering and verifying existing mass storage devices, a check is made to see if there is an initialize request pending for any of the mass storage devices. If initialization requests are present, then CPU routine MSI is activated to process the initialization. The control statement MSI. is entered into control statement buffer of the control point and 1AJ is called to process the next statement. This activity is performed in subroutine check initialization request (CIR) and is flowcharted in Figure 8-11.

Count Active Families

If no initialization requests were present and the permanent file system has not been verified, VPF is called. If the count of active families detected by VPF does not agree with the family count in PFNL, then the family count is updated in PFNL using monitor function IAUM with option IPFS or DPFS to increase or decrease the family count.

Once this phase has been completed, CMS terminates by dropping its PPU.

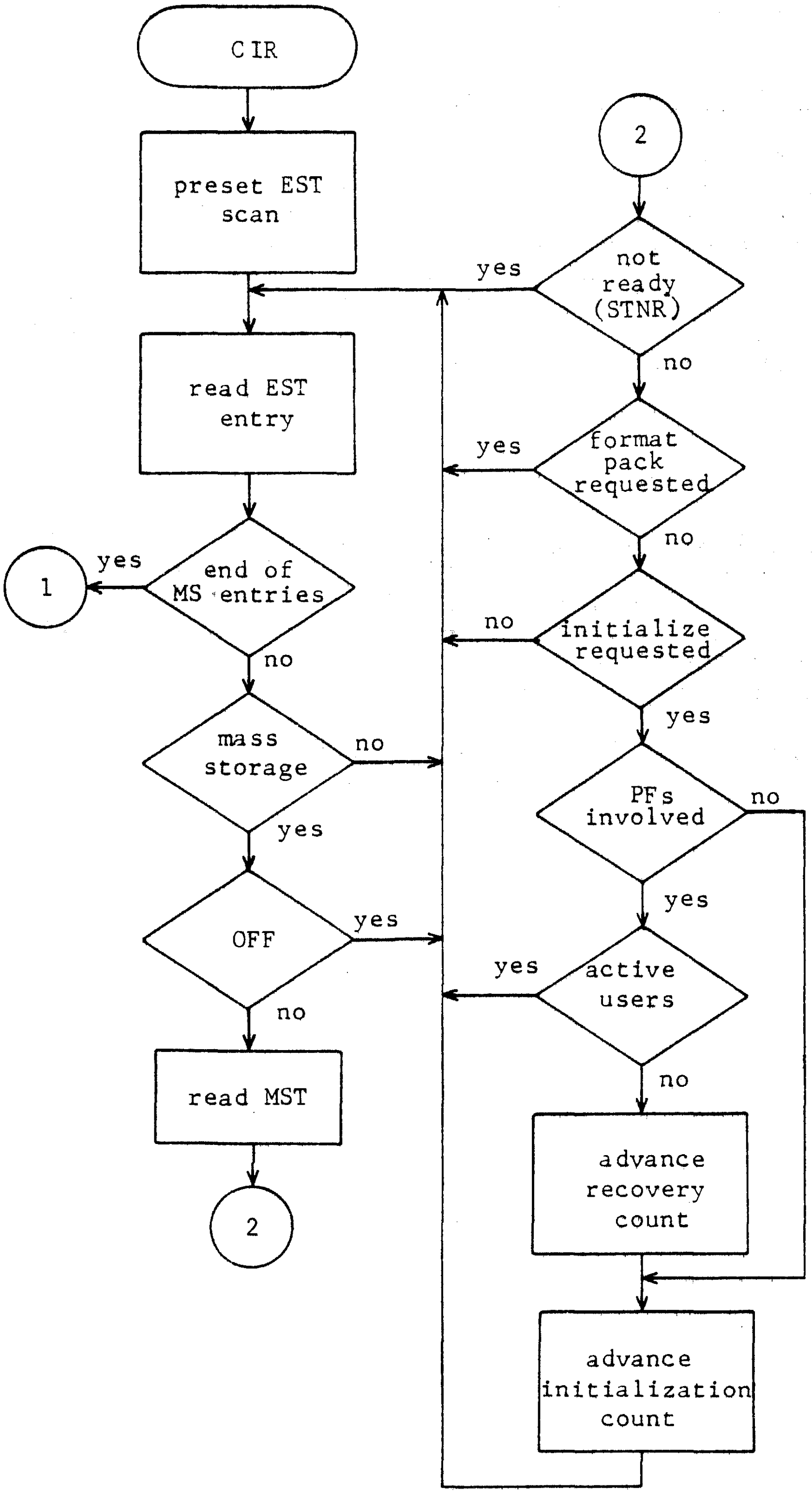


Figure 8-11. Check Initialization Requests (CIR)

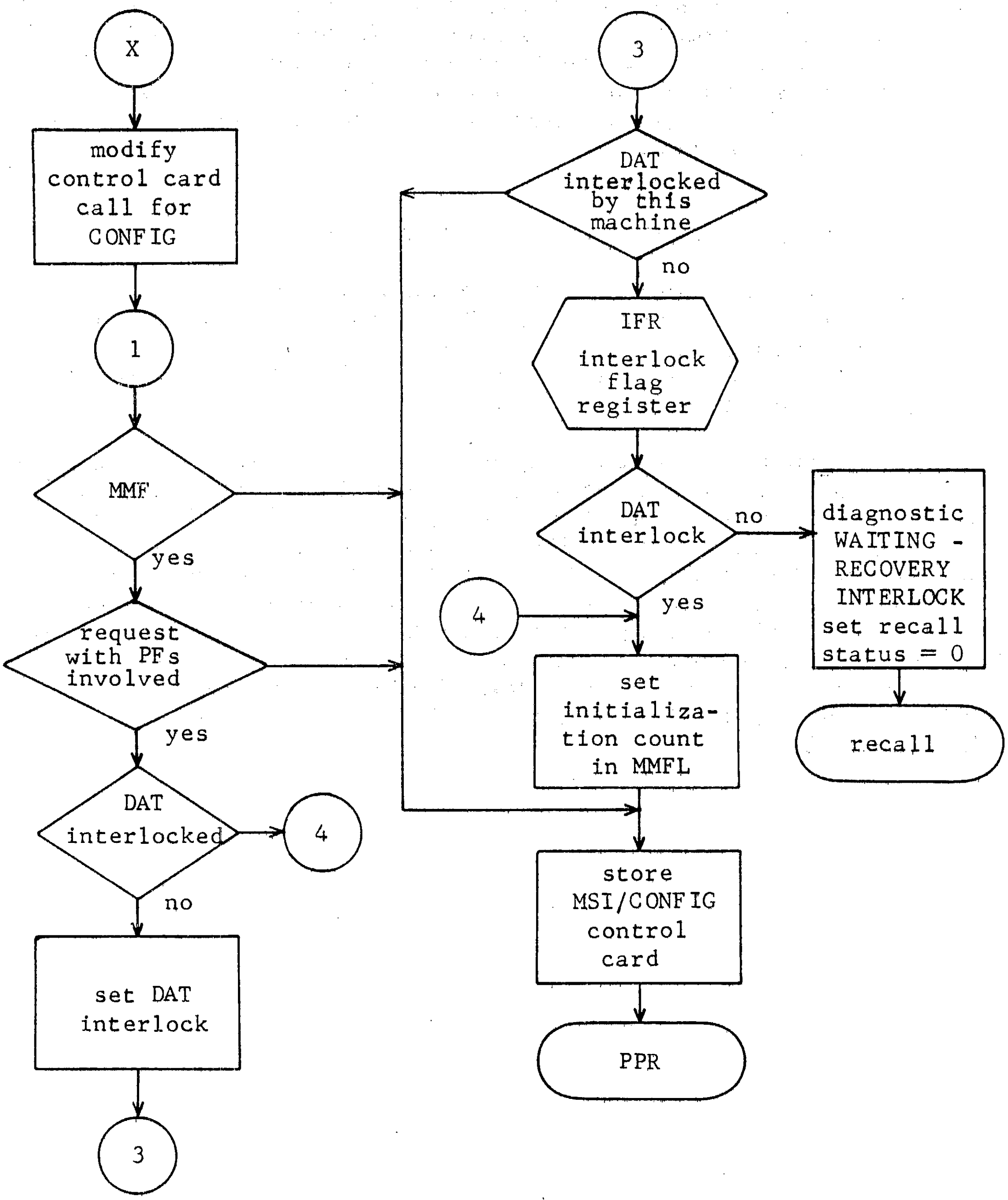


Figure 8-11. Check Initialization Requests (CIR) (Continued)

SYSTEM RECOVERY PROCESSOR (REC)

The third component of mass storage recovery is REC. REC performs many recovery functions but is mentioned in this section only for those activities dealing with the recovery of mass storage, namely the recovery of preserved files. REC loads overlay 4DA (which is part of MSM) to recover preserved files; this includes input/output queue files, and direct access permanent files. If a dayfile is not recovered, REC will establish a new one.

MASS STORAGE RECOVERY IN MMF ENVIRONMENT

For purposes of device usage determination, tables are maintained in ECS that identify the status of all devices in the multimainframe complex. This includes shared and nonshared devices for all machines. These tables are called the device access tables (DAT).

RMS and CMS use similar logic in recovering mass storage devices. When a device is recovered, the DAT is interlocked while a check is made to see if an entry exists for this device. The presence of an entry indicates that another machine is also accessing the device. If an entry is found and the machine recovering the device has not been instructed to share it, an error is indicated and recovery halts with an appropriate message displayed. If the machine already accessing the device is not allowing it to be shared (TRT is not ECS-resident), the same error condition occurs. These situations are illustrated in table 8-1.

TABLE 8-1. RECOVERY OF SHARED DEVICE ERRORS

\ Status Status \	Device Used in Nonshared Mode	Device Used in Shared Mode
Use Device in Nonshared Mode	ERROR Two machines want to use the same device in nonshared mode.	ERROR Machine coming up wants to use a device in nonshared mode that other machines are sharing.
Use Device in Shared Mode	ERROR Machine coming up wants to use a device in shared mode that another machine is using in nonshared mode.	Add accessing status to DAT.

The statuses across the top indicate in which mode the device is being utilized. The statuses down the left side indicate in which mode a machine coming up wants to utilize the device.

When a machine recovers a device it adds an indication to the DAT entry, if the indication does not already exist, that this machine has accessed (recovered) this device.

If the device is shared and another machine has it interlocked, a bit in the DAT is checked to determine if a level 0 type recovery is in progress on the device. Once the recovery is completed, recovery on this machine proceeds as indicated. It is not allowable to attempt a nonlevel 0 recovery on an interrupted machine once the recovery utility is run on another machine to recover the mass storage space of the interrupted machine. When RMS recovers a device on a nonlevel 0 deadstart, the DAT indicates that this machine has accessed the device previously. This status is cleared by the machine recovery utility.

It is the responsibility of each machine to recover its own local MST area off of the device. A bit in the global portion of the MST indicates if the sector of local information exists. In any event, if the local area that exists in the label sector matches the machine ID of the recovering machine, that local area is assumed to be the most up-to-date, regardless if information also exists in the sector of local areas.

If no entry for the device to be recovered exists in the DAT, an entry is made by RMS. A flag register interlock is set to prevent other machines from attempting the same. Once recovery is completed, the flag register interlock is cleared by REC.

Table 8-2 shows the steps involved for mass storage device recovery during the various levels of deadstart. When a device is not shared with any other mainframe, it is termed a standalone device. If the device is shared, the DAT is interrogated and recovery proceeds differently depending on whether it is active (in the DAT) or not active (not in the DAT). Another criterion that denotes which steps are taken is the machine mask field in the DAT which indicates whether or not the device has been accessed previously by this machine. Removable devices recovered on-line are handled the same as devices on a level 0 deadstart.

TABLE 8-2. MASS STORAGE DEVICE RECOVERY DURING DEADSTART

Level of Dead-start	Device Type					
	Standalone Device		Shared-Not Active (Not in DAT)		Shared - Active (In DAT)	
	*	**	*	**	*	**
0	2,4,6,7,8,14	Not appli-cable	1,4,6,7,8,9,10,14	Not appli-cable	3,11	11,12,13
1 and 2	2,4,7	4,7	1,4,5,7,9	Not appli-cable	3,11	11,13
3	Error	4,7	Error	Not appli-cable	Error	11,13

* Device not accessed previously
** Device accessed previously

The numbers in table 8-2 indicate the following.

1. DAT entry not found; make DAT entry that indicates that this machine only is currently accessing the device.

2. DAT entry not found; make DAT entry which shows that this machine is accessing the device but has no MST pointer (not shared).

3. Add indication to existing DAT entry which shows that this machine is accessing the device.

4. Retrieve MST (all local and global portions) from the device and, if shared device, preset into ECS. Retrieve TRT from device.

5. Set MRTs from device into ECS.

6. Edit TRT (that is, release all track chains except the preserved file chains).

7. Clear track interlocks for all machines.

8. Clean up system sectors (interlocks and user counts) for all machines.

9. Set TRT from device into ECS.

10. Clear MRTs for all machines.

11. Retrieve TRT and global MST from ECS. Get local MST from device. Clean up local MST (clear interlocks, reservations and request statuses).

12. Process MRT for this machine and drop local tracks.

13. Process MRT for this machine and clear track interlocks.

14. Build file of inactive queued files.

MSM OVERLAYS

MSM contains the CMS and RMS main programs and overlays that are used by CMS and RMS and by REC as well. The overlays of the MSM, and for mass storage recovery in general, have the name 4Dx. In the descriptions that follow, the mass storage recovery overlays are detailed.

OVERLAY 4DA/RDA

The main routine of 4DA and RDA is flowcharted in Figure 8-12. RDA processes preserved track chains recovering input/output queue files and direct access permanent files. If QPROTECT is enabled, the IQFT file is built from the input/output files recovered. Other subroutines in 4DA include:

CDA	Determines if a track is part of a preserved chain
CQF	Creates an IQFT entry for a recovered input or output queue file
IQF	Creates the system sector for the IQFT file
TQF	Completes the IQFT file (if queues were recovered), causing the IQFT first track to be set in word ACGL of the MST and checkpoints the device if the IQFT is not empty
VFL	Verifies that the file is as long as the track chain for the file indicates
WQF	Writes individual sectors of the IQFT as the IQFT buffer becomes full
IDM	Issues the following messages: EQee xxxx DIRECT ACCESS FILES RECOVERED. EQee xxxx PRESERVED FILE ERRORS. EQee xxxx DIRECT ACCESS FILE ERRORS. EQee xxxx QUEUED FILES RECOVERED. EQee xxxx QUEUED FILE ERRORS. EQee xxxx QUEUED FILES IGNORED.
PFE	Formats the permanent file length error message
BAD	Compute IQFT buffer address
CFL	Change file length
IEM	Issue error log message
RDC	Read disk chain

VSL	Validate sector linkage
VTC	Verify track chain
IRM	Issue recovery messages
CEA	Convert ECS address
CTU	Clear user counts
GDE	Get DAT entry from ECS
WDE	Write DAT entry to ECS

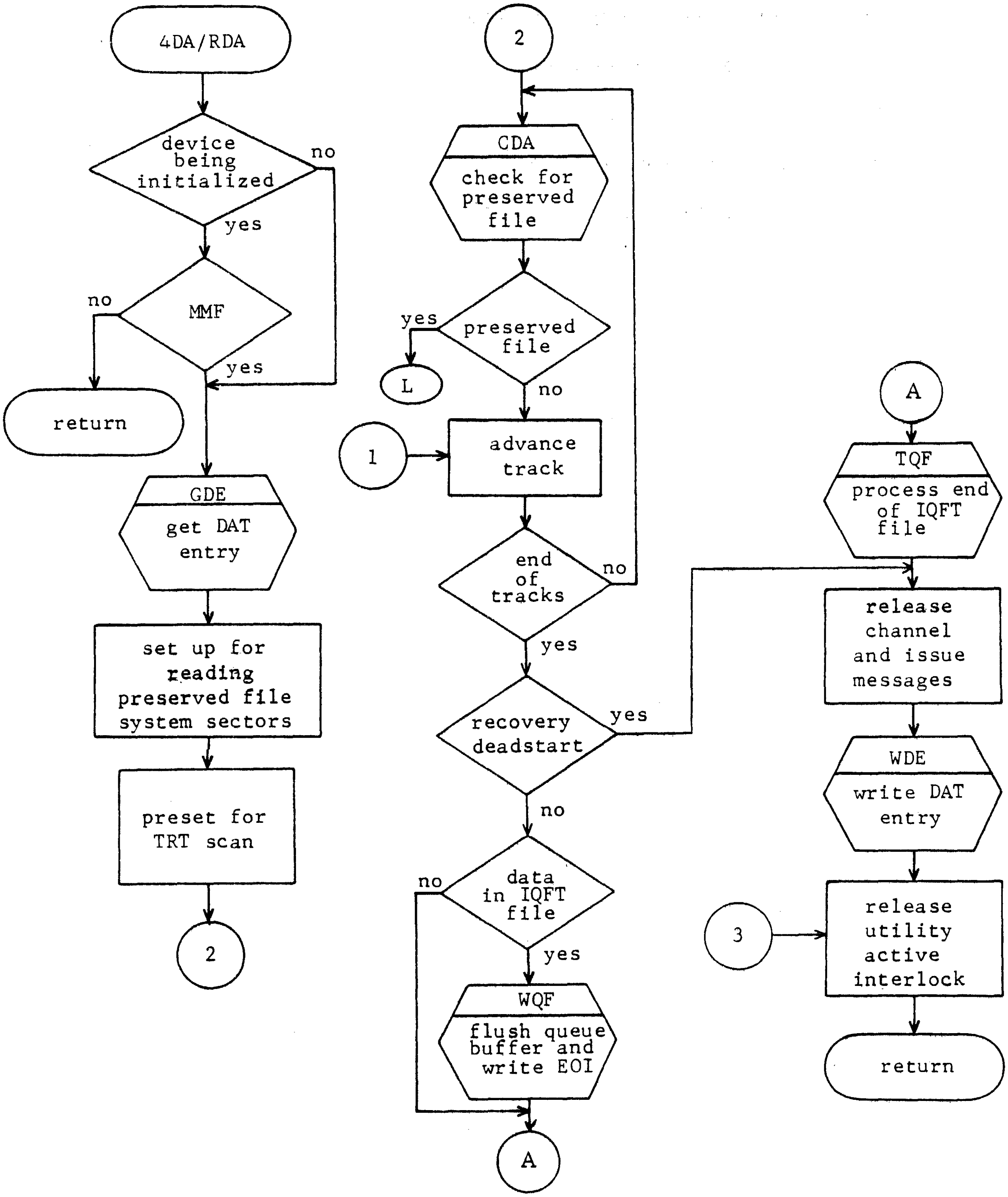


Figure 8-12. Overlay 4DA/RDA

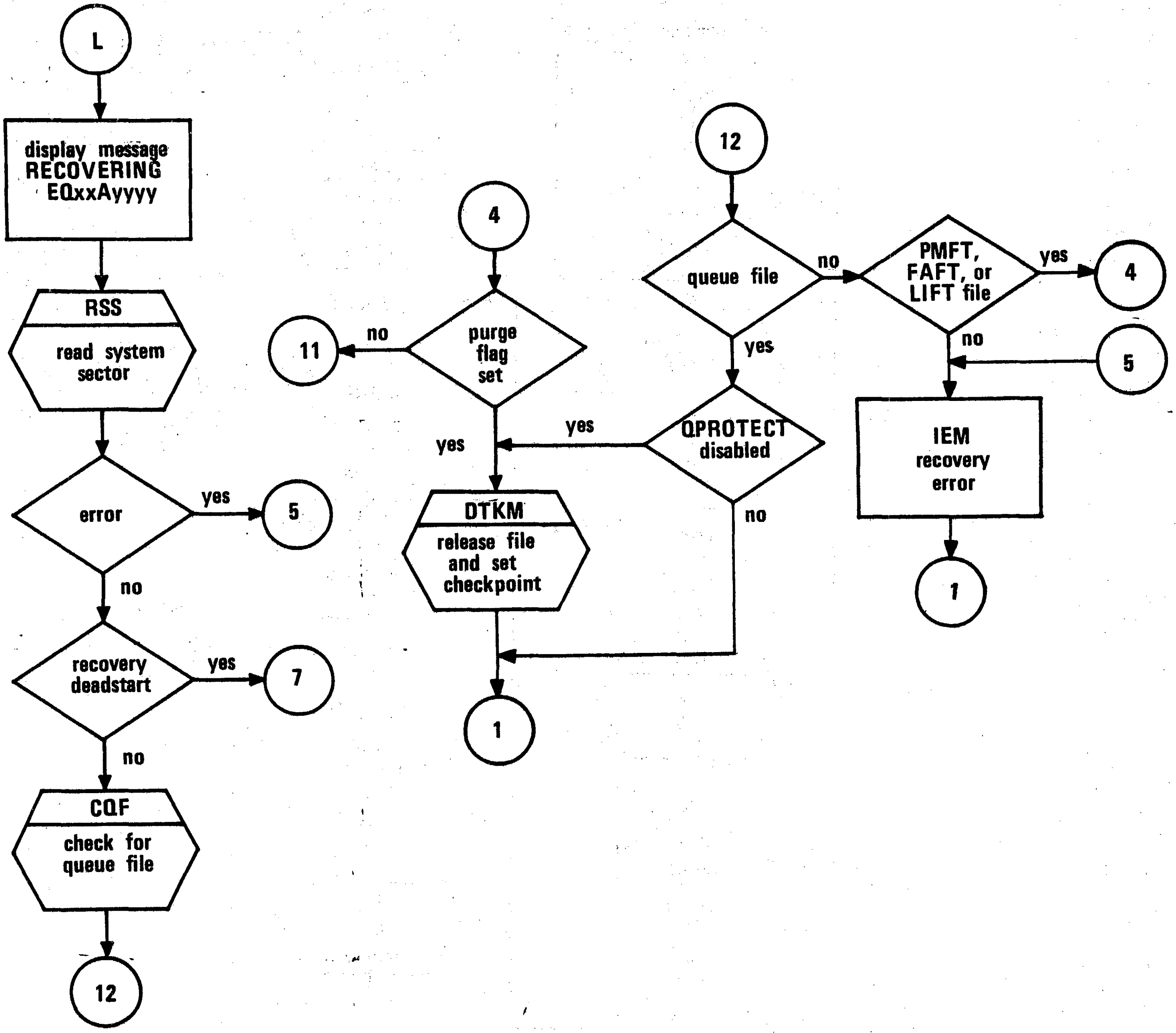


Figure 8-12. Overlay 4DA/RDA (Continued)

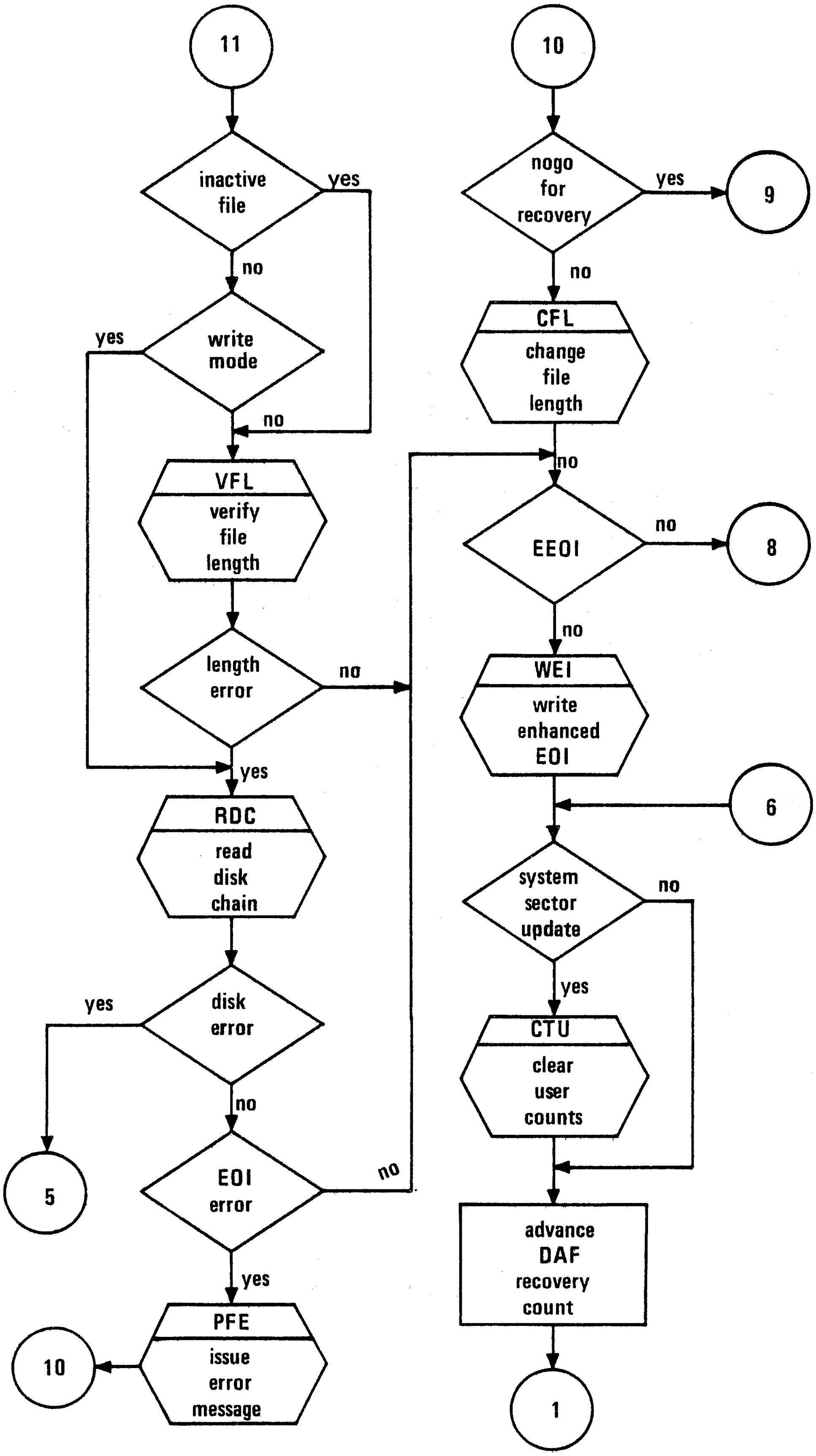


Figure 8-12. Overlay 4DA/RDA
(Continued)

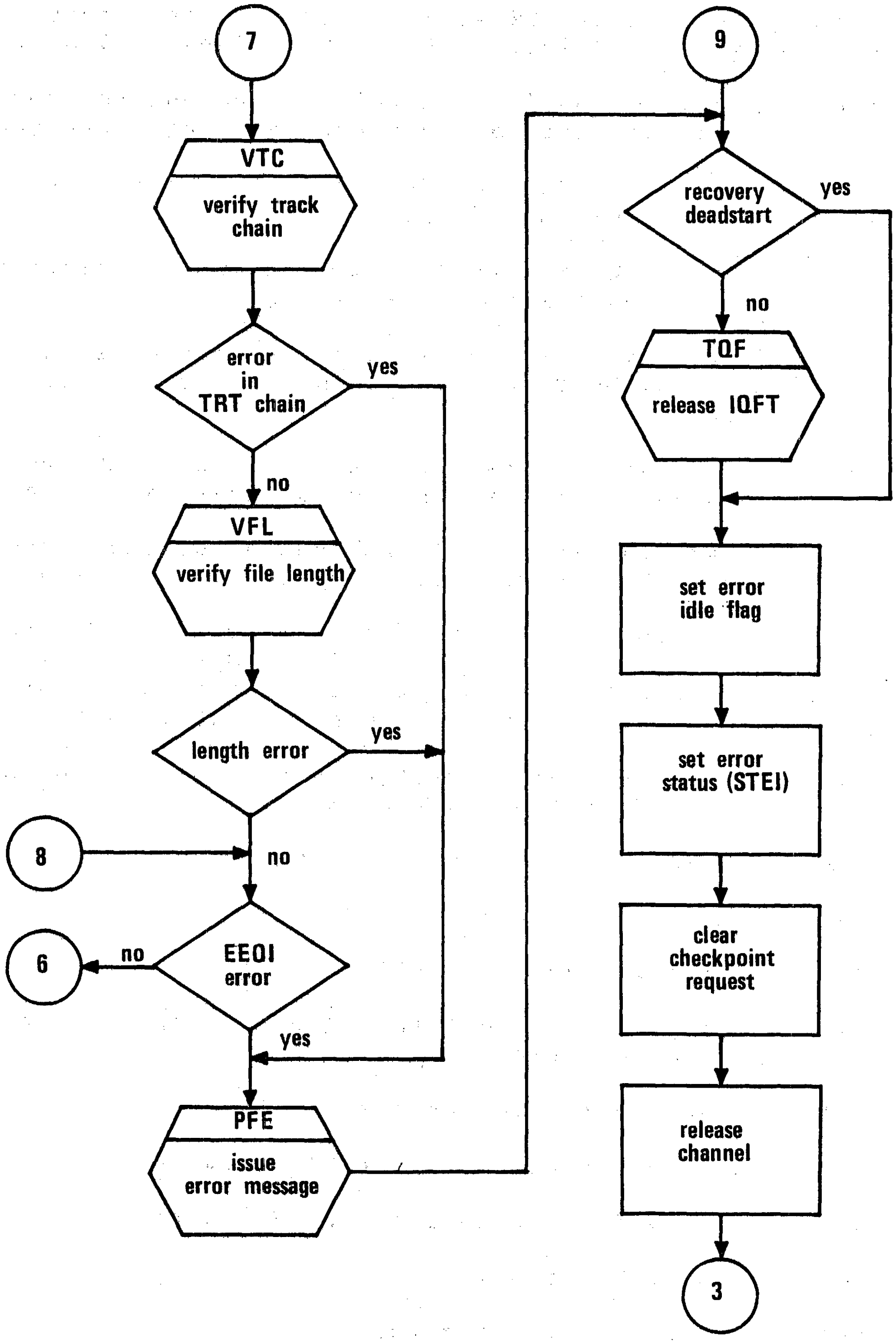


Figure 8-12. Overlay 4DA/RDA
(Continued)

OVERLAY 4DB

The primary routine of 4DB is RDL (read device labels), which is flowcharted in figure 8-2. Other subroutines in 4DB are:

MRL	Provides looping control for subroutines that must be executed once for each device being processed
CLR	Clears the TRT, preserving only those tracks that have been previously flawed
CMT	Clears the MST words ALGL and DULL; initializes MDGL, STLL, and ACGL; and calls CLR
IES	Initializes the equipment status fields in MST word STLL by building the equipment chain values for this device
SSL	Sets the sectors per track values in MST word MGDL depending upon the number of units in the equipment
RLM	Reads the local MST from sector of local areas if the local MST is not the correct MST for the machine ID
SPP	Sets the permanent file attributes of the device into the MST from the label of the device; converts permanent file information from predecessor systems into the current NOS format of this data
WMT	Writes the MST from the working buffer in the PP into the MST area of central memory for the device
CEA	Convert ECS address
SNT	Set next track in DAT chain
WDE	Write DAT entry to ECS
CAM	Change access mode (half/full track or full/half track for LDAM devices
CFT	Clear full track access
SFT	Set full track mode for LDAM device
SHT	Set half track mode for LDAM device
RLS	Read label track
SLT	Search for label track
CSD	Returns to its caller a status indicating the condition of the DAT with respect to this device
CDE	Check DAT entry

RDE	Read DAT entry from ECS
SDT	Search DAT
UDT	Update DAT in ECS
LDT	Load DAT from ECS

OVERLAY 4DC

The subroutines in 4DC are:

VPF	Determines on a family basis that the mounted members of the family have unique device numbers and do not duplicate master device mask bits
CAN	Builds a table of family names currently mounted
CFN	Compare family/pack names
GNE	Get next entry from MS EST
ERR	Issues one of the following diagnostics: EQxx EQyy CONFLICTING DN EQxx EQyy CONFLICTING PN EQxx EQyy CONFLICTING UM and calls 4DB/IES to set the appropriate error code in the MST.

OVERLAY 4DD

The primary subroutine of 4DD is IDF (initialize dayfiles) which is flowcharted in figure 8-13. The other subroutines in 4DD serve entirely as subordinates to IDF. Overlay 4DD is assembled as part of REC and is included for completeness of 4Dx overlays.

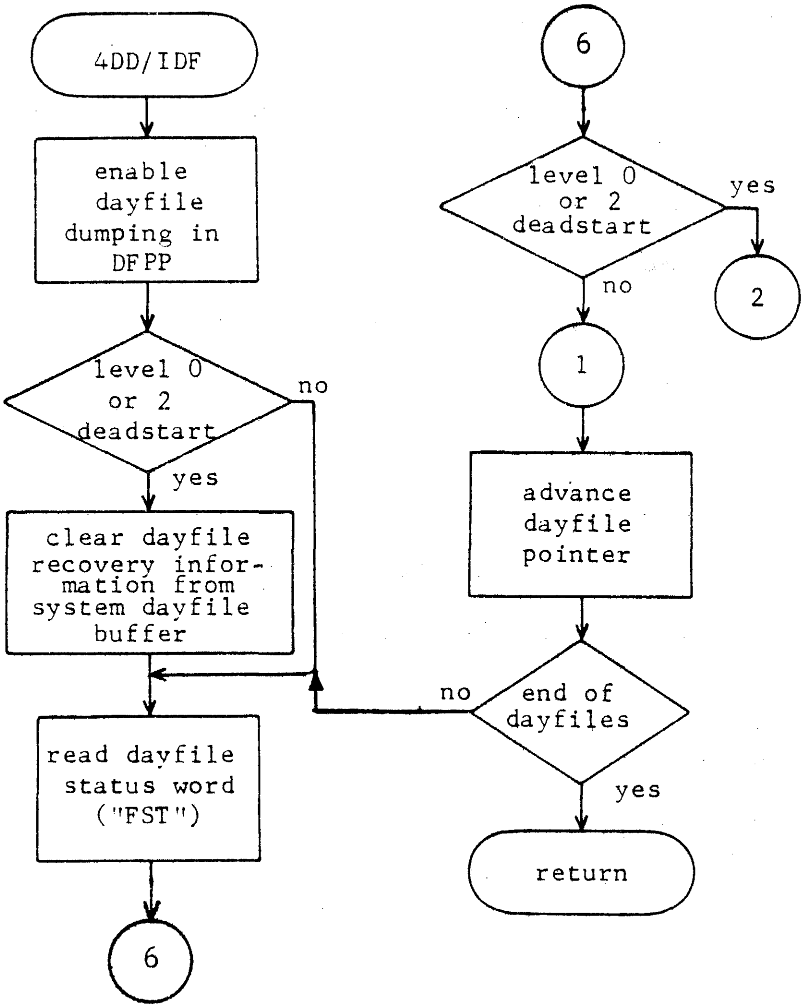


Figure 8-13. Initialize Dayfiles (IDF)

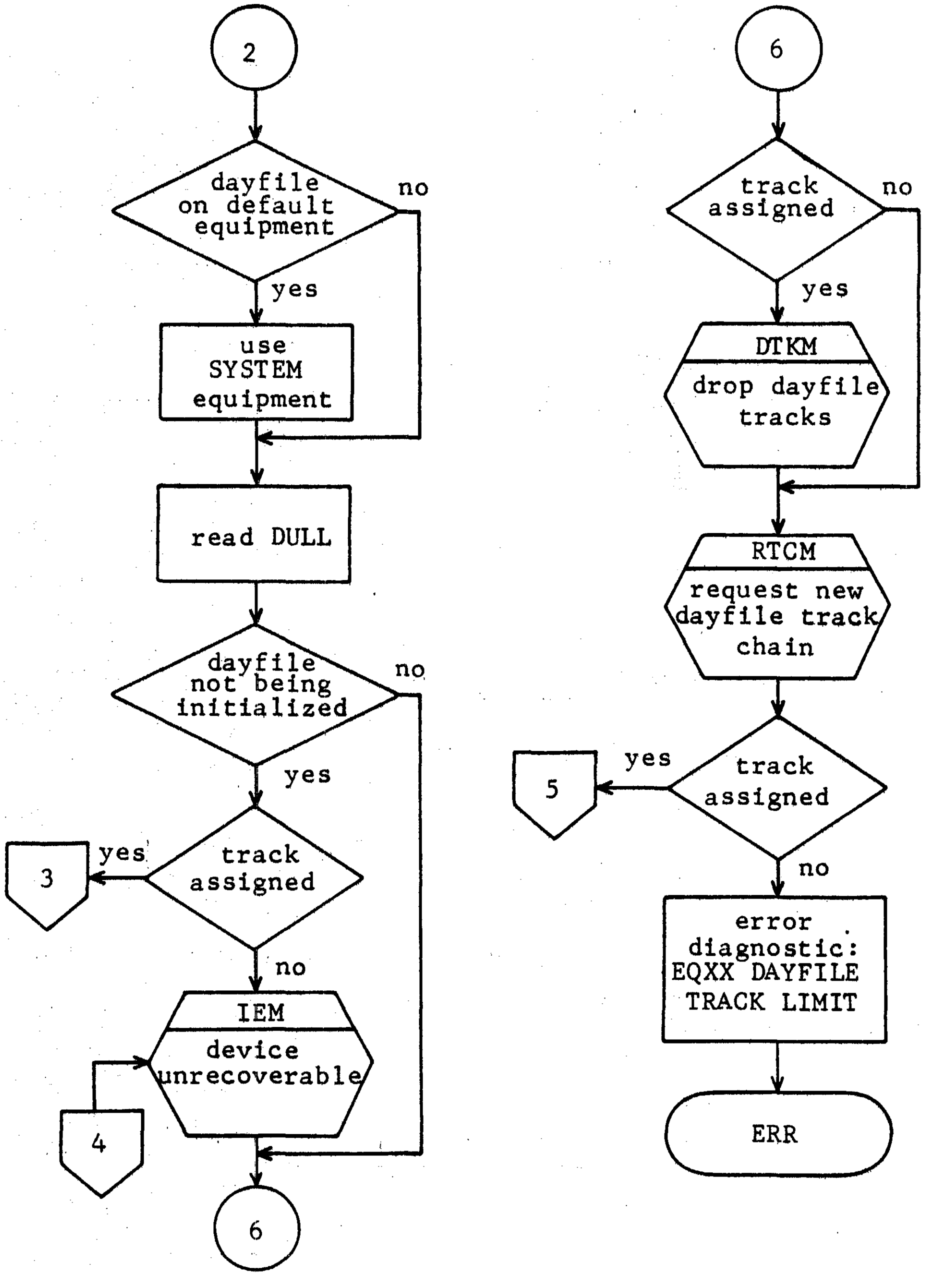


Figure 8-13. Initialize Dayfiles (IDF)
(Continued)

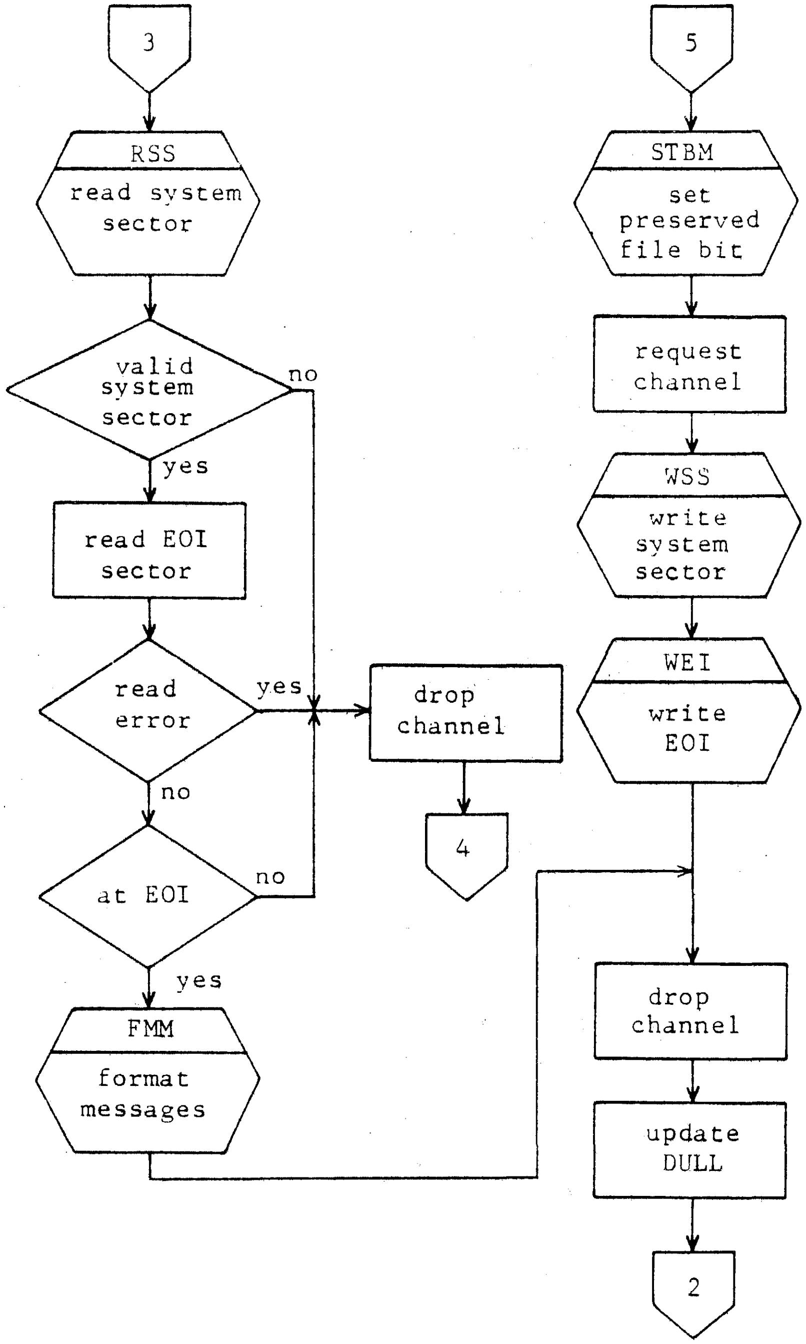


Figure 8-13. Initialize Dayfiles (IDF)
(Continued)

OVERLAY 4DE

The subroutines in 4DE manipulate the user ECS chain. This chain is a contiguous track chain which is then used as an area to allocate user ECS blocks. The routines are:

ECS	Main routine to allocate/recover user ECS area
ACE	Assign contiguous ECS tracks
CAD	Clear allocation data in system sector for specified machine ID
CDI	Clear device interlock on ECS
FAD	Fetch allocation data for specified machine ID from system sector
FSS	Fetch system sector for user ECS chain
FTM	Find area for this machine's subchain in system sector
HNG	Issue message and hang
ISS	Initialize system sector for user ECS area
RLS	Release existing user ECS subchain
SCP	Set up control point areas for access to ECS
SDI	Set device interlock on ECS equipment
SSS	Write system sector for user ECS chain

To support user ECS, a contiguous track chain is reserved by routines in overlay 4DE. This chain is split into subchains that are used by machines in an MMF environment which have machine IDs that match the ID on the subchain. The information describing the subchains is written in the system sector of the user ECS chain. The user ECS system sector format is shown in section 2.

OVERLAY 4DF

The subroutines in 4DF manipulate MMF tables contained in ECS. These routines are:

EDT	Builds the DAET word in ECS
UER	Clears or enters machine recovery tables (MRTs) for the device in ECS
CRT	Clears the MRT of this device
ERT	Edits the MRT
SMT	Stores the MRT and TRT into ECS

OVERLAY 4DG

Overlay 4DG contains routines for manipulating track reservation tables. TRTs are read into central memory area separate from where they reside with the MST for recovery operations. The major routines of 4DG are:

ATC	Adjusts the track count (number of tracks remaining) in word TDGL of the MST.
EMT	Updates the MST and EST to indicate that the device is available updating MST words ACGL, MDGL, DULL and the PF descriptors ALGL, PFGL and PUGL.
ETT	Edits the labels from the recovery buffer in central memory to the TRT area, releasing all track chains that are not preserved or flawed.
RTT	Reads the TRT into the recovery buffer in central memory from its position in the device's label track.
SEC	Sets the equipment configuration by indicating whether a device is part of an equipment chain. While part of an equipment chain, the MST/TRT of the first device in the chain is the primary source of information for the chain.
CCE	Validates the equipment chain to verify that the elements of the chain are correctly linked.
CLP	Compares a set of label parameters consisting of pack name, user number, and number of units with a desired set of these values.

VLP	Validates labels read for a given equipment chain. All devices in the chain must be ready and have correctly read labels as well as satisfying the desired set of pack name/user number/number of unit properties. VLP calls CLP and CCE.
RTC	Reserve track chain.
AUL	Assemble unit list.
CEP	Compare equipment parameters.
VDP	Verify device parameters.

OVERLAY 4DH

Overlay 4DH contains routines utilized by RMS in initializing mass storage equipment. The primary subroutine of 4DH is IDS (initialize device status) which is flowcharted in figure 8-13.1. The other subroutines in 4DH, which are entirely subordinate to IDS, are the following.

CTF	Check track flawed in TRT.
IFM	Interpret flaw map. This routine uses overlay OTI to read the factor flaw map on 844 and 885 type devices and sets the flaws in the TRT.
PFT	Prewrite flawed track. Prewrites any potential label that is flawed.
RCS	Reserve CTI space. This routine reads the deadstart sector, checking for the presence of CTI/MSL, and flaws those areas of the TRT accordingly.

MSM OVERLAY LOAD ADDRESSES

Figure 8-14 details a load map of MSM. The load addresses of the various overlays of MSM are defined as follows:

Routine	Load Address	Definition
4DA	O4DA	Maximum of /CMS/PRSX+5 and OCTL+5
4DB	O4DB	NMSD + maximum of /CMS/PRSX and /RMS/PRSX
4DC	OSOV	End of common subroutines in 4DB
4DF	OSOV	
4DG	OSOV	

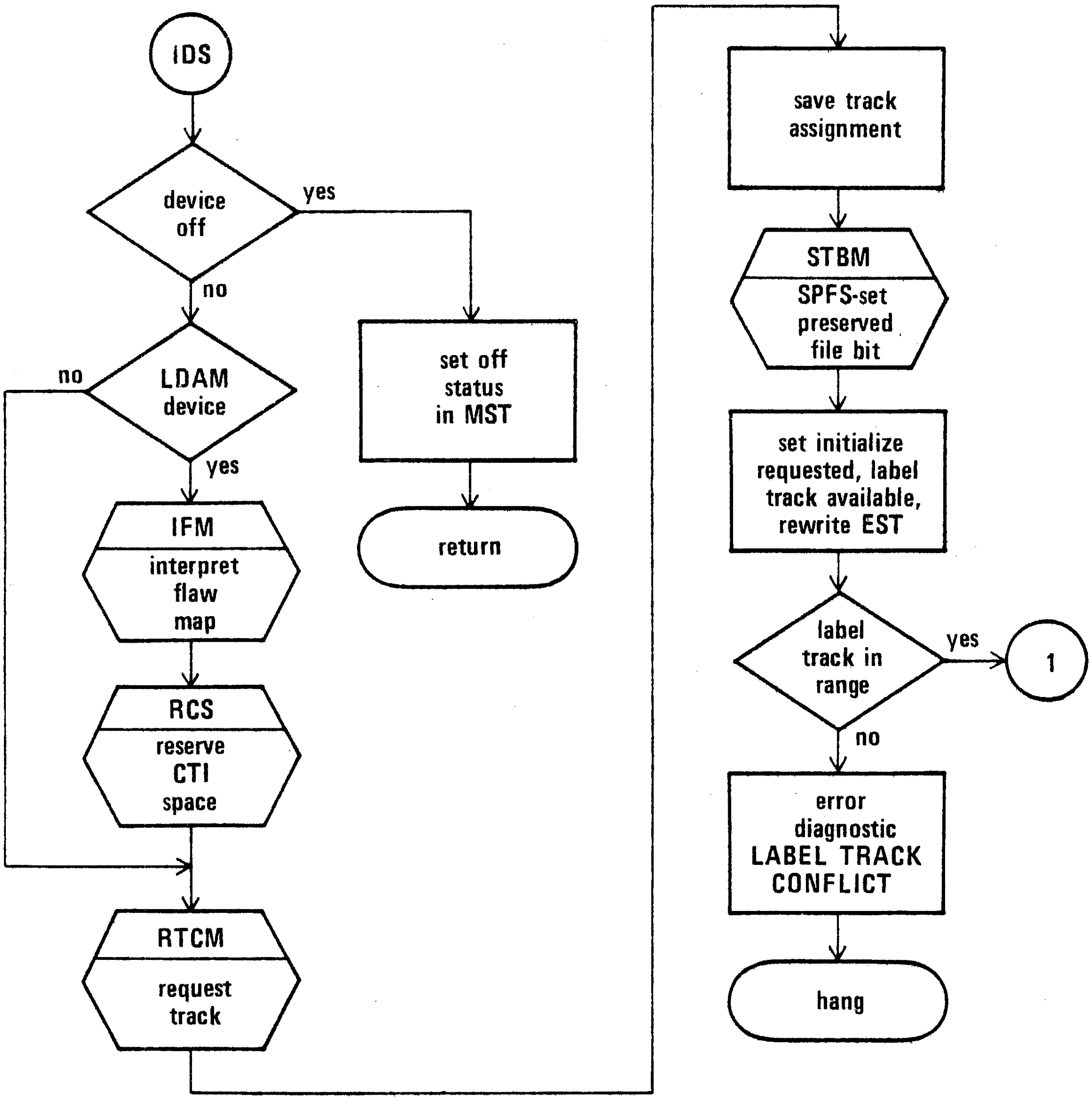


Figure 8-13.1. Initialize Device Status (IDS)

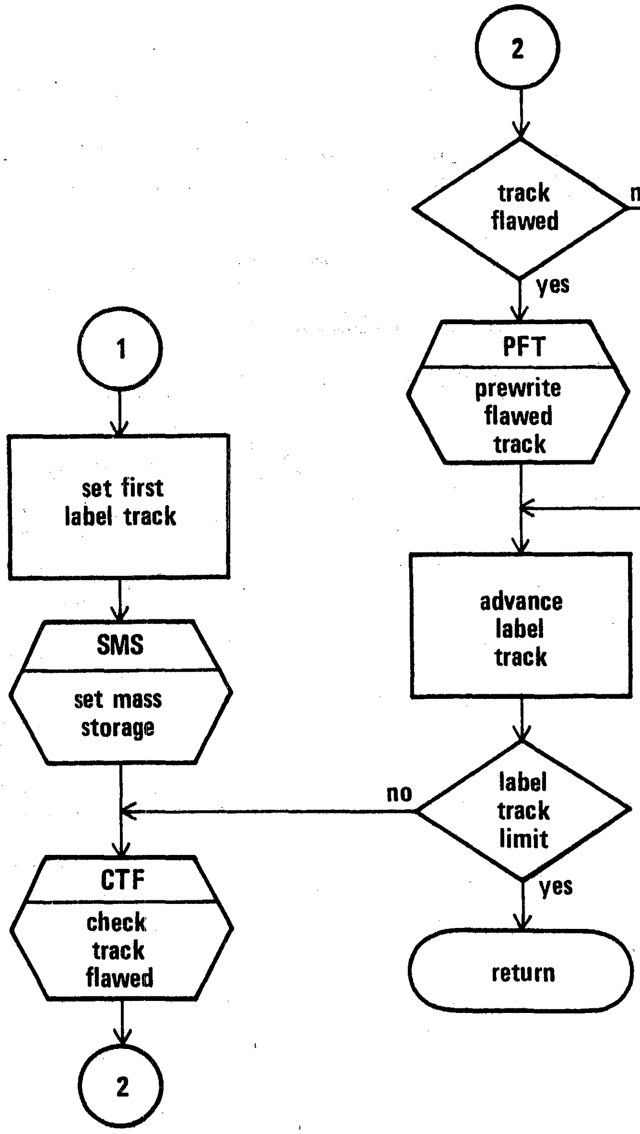


Figure 8-13.1. Initialize Device Status (IDS)
(Continued)

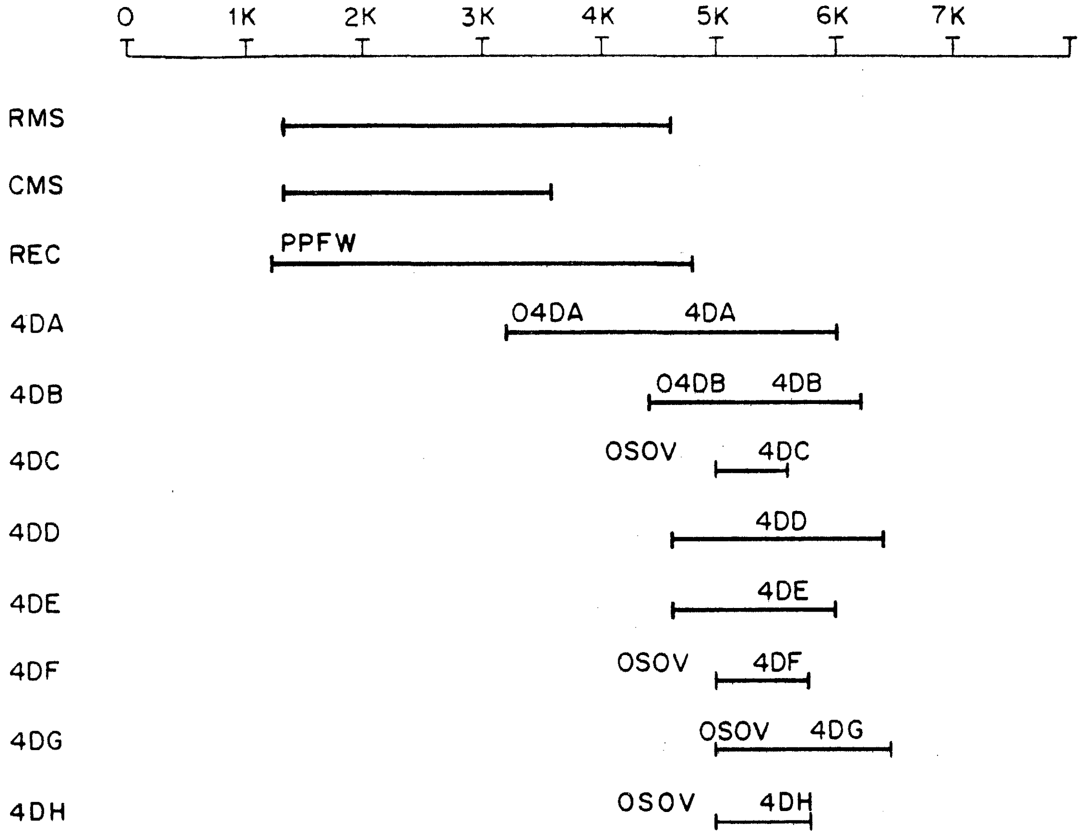


Figure 8-14. MSM Load Map

Table 8-3 shows a cross-reference of the main routines CMS, RMS, and REC and the overlays that they call.

TABLE 8-3. MSM CROSS REFERENCE

Overlay	CMS	RMS	REC
4DA	X		X
4DB	X	X	
4DC	X	X	
4DD			X
4DE			X
4DF	X	X	
4DG	X	X	
4DH		X	
others	OPI		OCI
		OMF	ODF
			OPI
			ORF

DEVICE CHECKPOINT

The information written on a mass storage device, whether during an initialization or during usage, is kept accurate by the means of a checkpoint. The term checkpoint in this section refers to those checkpoints that update the label track of the device. The success of a recovery operation depends almost entirely upon how accurately the information in the device label reflects the actual usage of the device. A checkpoint operation updates the MST and TRT information contained in the device label from the MST/TRT active in central memory.

There are six types of checkpoints:

- A deadstart checkpoint allocates system table space and copies the tables to mass storage, checkpoints the TRT on system devices, and checkpoints local areas on devices with active dayfiles.
- A system checkpoint idles the system and checkpoints system tables and TRTs.
- A device checkpoint checkpoints the TRT for all devices with checkpoint requests set.
- An alternate library checkpoint performs the same function as a system checkpoint without idling the system.
- A local area checkpoint copies the local MST information of a device to the local area sector. The TRT on that device is also checkpointed.
- An initialized device checkpoint performs the same function as a device checkpoint, except the call has been initiated by IMS for a single device.

Checkpoints are performed by PP routine 1CK. For device checkpoints, 1CK is called from 1SP. Routine 1SP examines the MST for mass storage equipments every 30 cycles (about 30 seconds) and calls 1CK if the checkpoint bit is set in STLL.

The major concern for mass storage recovery is the device checkpoint. The device checkpoint writes the current EST and MST in the label sector and copies the entire TRT to the label track. Refer to Section 2 for the format of these areas of mass storage. For all checkpoints, the TRT is written to the label track; this TRT checkpoint is the last operation performed in a checkpoint in order to guarantee the accuracy of the TRT. The flowchart for the TRT checkpointing routine WTT is shown in figure 8-15.

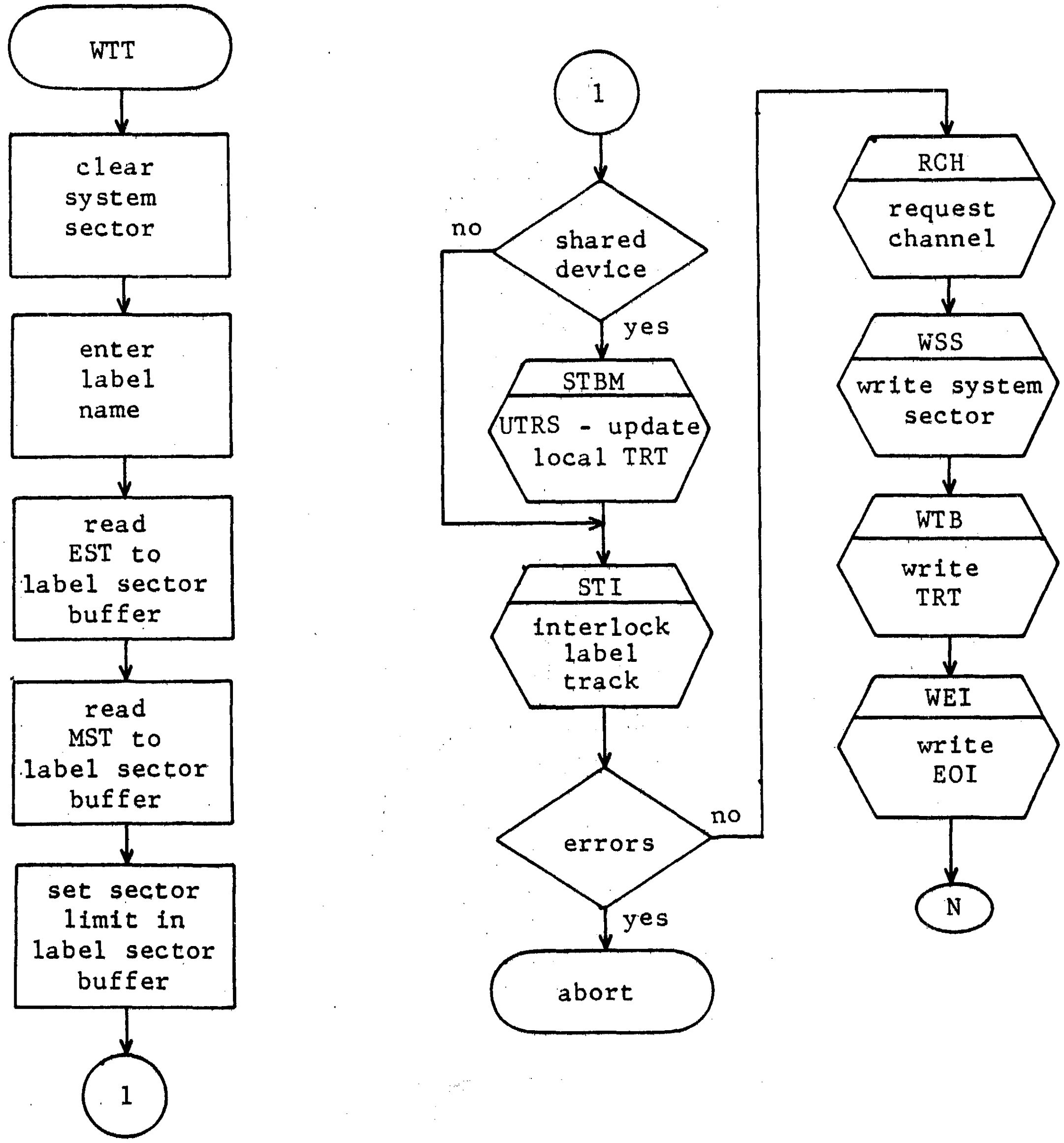


Figure 8-15. Write TRT (WTT)

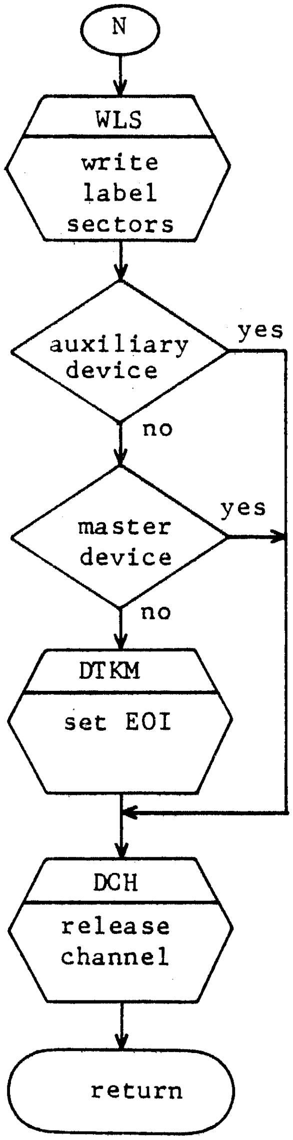


Figure 8-15. Write TRT (WTT)
(Continued)

ON LINE RECONFIGURATION OF RMS

On-line reconfiguration of devices is handled by two routines, RDM (redefine mass storage) and CONFIG (configure mass storage). RDM handles the actual device reconfiguration and table interlocking. CONFIG supplies the operator interface via a K-display for all interactions required while redefining a device. The redefinition process is initiated by the DSD entry REDEFINE,eq. This entry starts a sequence which calls CONFIG to a control point for the actual reconfiguration process.

ROUTINE RDM

The RA+1 call to RDM has the following format.

	59	35	23	17 0
RA+1	RDMP	function number	0	status word address

That status word is necessary for all of the RDM functions. Following is a list of the functions with the status word contents upon entry and the reply, if any, upon exit.

Function 1 - Search for Outstanding Requests

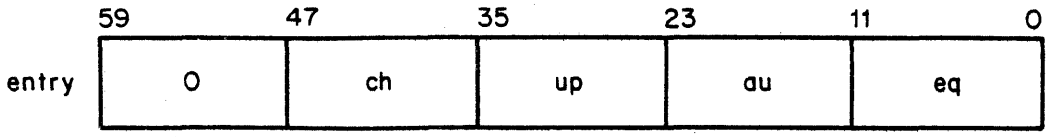
	59	47 0
reply	eq	0

eq	EST ordinal of first shared equipment having reconfiguration reply machine masks in word ACGL; 4000B if no reconfiguration reply machine masks found on any shared equipment

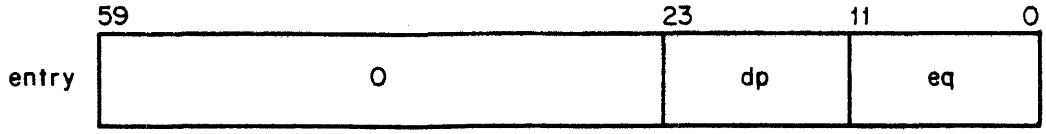
Function 2 - Replace Unit

	59	35	23	11 0
entry	0	dp	au	eq

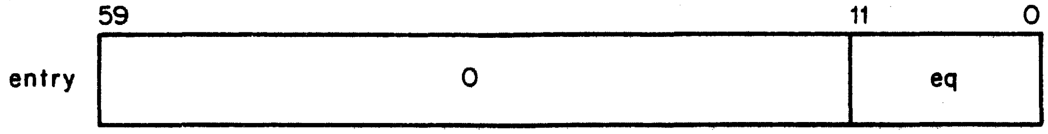
Function 3 - Add Unit



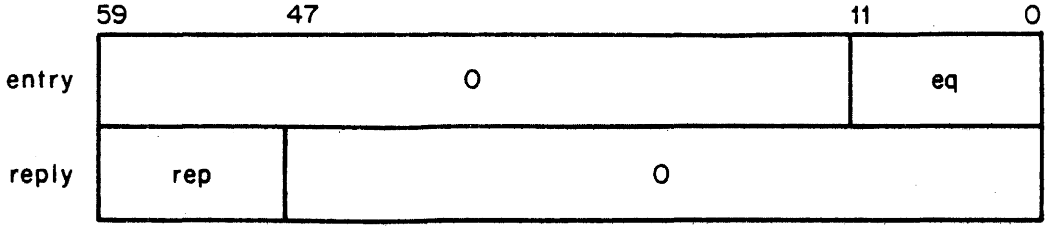
Function 4 - Delete Unit



Function 5 - Clear Request



Function 6 - Ignore Processing of Device



rep	4000B if redefinition requested bit set; 0 if not set.

The following mnemonics describe the parameters in the above entry conditions.

ch	Channel(s) for the unit to add

up	Unit position of the unit to add
au	Unit to add to the unit list
dp	Unit position of the unit to delete
eq	Equipment to process

Function 1 is called by CONFIG only if the machine is running in MMF mode. First, it gets the ECS DAT interlock, if it does not already have it. Next, the mass storage devices in the EST are searched for LDAM type shared devices. When one is found, the MST is updated from ECS. If the redefinition requested bit in word ACGL is set, the reply bits in ACGL are checked. If a reconfiguration reply machine mask is found set in word ACGL, then this shared equipment is currently being reconfigured within the complex, so the equipment number (EST ordinal) is returned in the reply word. This equipment then requires acknowledgement by the operator through CONFIG. If no reconfiguration reply machine masks are found set in the entire EST search, the reply of 4000B is sent to CONFIG. Upon a 4000B reply, CONFIG accepts input for the remaining equipments to be reconfigured, as listed on the K-display list, utilizing functions 2 through 6 to process these equipments.

The replace unit function (function 2), when calld in MMF mode to reconfigure a shared device, first reads the DAT entry for the device from ECS. If an unrecoverable read error is encountered reading the DAT entry, the message ERROR ON LINK DEVICE is issued to the error log, the redefinition requested bit cleared, and the run aborted. After reading the DAT entry, the function requests all channels to the drive in ascending numerical order, set the redefinition in progress bit in word DDLL of the MST, and then drop the channels. This insures that no one from this machine is able to access the device, since software drive reserve status is now set. The reconfiguration reply machine mask then is written in word ACGL of the global MST and the DAT interlock is dropped. Next, the function loops, checking word ACGL until all machines in the complex that are accessing the device have set their reconfiguration reply machine masks. The machine masks from the DAT entry are used to check when all machines have complied.

Once all machines have complied, the EST and MST fields necessary for verification of the pack labels are copied from the equipments being monitored to the pseudo equipment RD, EST and MST. Following this, the old device is monitored for not ready status. The opertor is notified of this if the unit is not already spun down via the flashing B-display message SPIN DOWN UNIT x. Next, the new device is monitored for not ready, then ready status. Again, flashing B-display messages, SPIN DOWN UNIT y and SWITCH PACK/SPIN UP UNIT y, notify the operator of the current operation. These operations insure that both drives have been spun down and the new drive spun up again.

The message VERIFYING EQxx DNyy is now displayed on all machines. By use of the RD pseudo equipment and the standard driver, RDM then verifies that the correct pack has been mounted on the unit by verifying the pack label. This verification is performed on each channel connected to the device, verifying on the channels in ascending order. After verification, the fields in the pseudo equipment EST and MST are reset.

If an unrecoverable read error is detected while reading the label on any channel, the message LABEL READ ERROR/ENTER STOP is flashed to the operator. In this case, all machines in the complex must enter STOP to begin the return to original configuration operation. If the verification of the label on one machine is not valid, the message LABEL VERIFICATION ERROR is displayed on that machine. If this message appears on all machines in the complex, the units should be spun down and the physical packs checked to insure that the correct packs were switched. If the wrong packs were switched, the correct switch should be made and normal operations continued. If the correct switch had been made, or if the error message appeared on only some of the machines in the complex, all the machines should enter STOP to initiate the return to original configuration operation.

On the return to original configuration operation, the return switch of the packs is monitored, flashing the messages SPIN DOWN UNIT y, SPIN DOWN UNIT x, and RETURN PACK TO UNIT x, as necessary, to the operators. If the return verification is good and no unrecoverable read errors were encountered, the reconfiguration reply machine masks are cleared, the redefinition requested and redefinition in progress bits cleared, the message EQxx REDEFINITION ABORTED issued to the error log, and the reconfiguration runs aborted.

If the return verification cannot be completed, the message RETURN ERROR/ENTER OVERRIDE is flashed on the display and each machine has to enter OVERRIDE, causing the masks and bits to be cleared, the error log message issued, and the reconfiguration runs aborted.

On normal operations with no errors, as each machine verifies the label, they clear their own reconfiguration reply machine mask in word ACGL of the global MST and then loop, waiting for the redefinition requested bit to be cleared. The machine which clears the last reconfiguration reply machine mask also clears the redefinition requested bit. The current unit number with the unit position specified in the status word is then replaced in the unit list in word DDLL of the MST by the replacement unit number from the status word. The redefinition in progress bit is then cleared in each machine.

NOTE

> The DSD STOP command may be entered at any time during the reconfiguration run if it is decided that the reconfiguration is not wanted. Necessary clean-up procedures are initiated automatically when STOP is detected at various points within the routine. Aborting a switch run causes the replacement unit to be left undefined (not in the EST).

When called in single mainframe mode or MMF mode on a nonshared device, the replace unit function sets the redefinition in progress bit. The operator is instructed to spin the units down and switch the physical packs and the pack is verified. Finally, the redefinition requested bit is cleared, the current unit number replaced in the unit list by the replacement unit number, and the redefinition in progress bit cleared. All error processing remains the same as in MMF mode on a shared device (described in pevious paragraph). When the call is in MMF mode on a nonshared device, the machine will have and hold the DAT interlock throughout the entire reconfiguration run.

NOTE

> When function 2 has redefinition in progress set, no additional PP calls or overlay loads may be processed and no dayfile messages issued by RDM until the operation is complete.

The add unit function (function 3) checks first the no units bit in word DDLL of the MST of the equipment specified in the status word. If no units exist, the bit is cleared and the status word checked for channel numbers. If channel numbers are found, they are set in the EST. The unit number read from the status word is then positioned according to the unit position, also passed in the status word, in the unit list in word DDLL. A position of 0 means to set the device as the first device of an equipment, 1 as the second device, and so on, up to a position of 7 which means the eighth device of an equipment. Finally, the number of units - 1 count in word DDLL and the original number of units in word STLL of the MST is incremented by one if not adding to null equipment. If in MMF mode, the DAT interlock is held throughout this function.

The delete unit function (function 4) deletes the unit number specified by the unit position given in the status word from the equipment unit list in word DDLL of the MST for the equipment specified in the status word, shifting the following units, if any, to the right in the unit list. It decrements the number of units - 1 count in word DDLL and the original number of units in word STLL of the MST by one if they are the same unless they already are zero, in which case the no units bit in word DDLL is set. If the unit is found to be an unused unit on a device, only the original units count in word SSTL is decremented. If in MMF mode, the DAT interlock is held throughout this function.

The clear reconfiguration run function (function 5) reads the equipment number from the status word and clears the redefinition requested bit in word ACGL of the MST. If running in MMF mode, this function should be called through the use of the CLEAR command only if all processing on the specified equipment is completed. If it is decided that no machine wishes to execute the reconfiguration on the given equipment, whether MMF or not, the CLEAR command must be used to clear out the redefinition requested bit.

The ignore processing on device function (function 6) reads the equipment number from the status word, clears the DAT interlock if it is held by this machine, and then checks, the redefinition reqested bit. If the bit is set, the reply of 4000B is sent to CONFIG. This function is called repeatedly until the bit is found to be clear, in which case a 0 reply is returned to CONFIG.

DEVICE REDEFINITION LOGIC FLOW

A device redefinition sequence would follow the following sequence.

1. Enter DSD entry REFEDINE, eq. This sets redefinition requested status for equipment (eq) MST word (ACGL bit 11).

2. 1SP during its periodic execution checks for bit 11 in ACGL. If the bit is encountered the CMS call sequence executes.

3. CMS detects the redefinition requested status, obtains the DAT interlock (if required due to MMF operation), and calls CONFIG.

   NOTE

   The 1SP, CMS logic enables the redefinition programs to run as part of the MS subsystem.

4. CONFIG scans the MSTs looking for redefinition requested bits set and builds a list of these equipments.

The following is the sequence CONFIG and RDM follow while replacing or switching units on a shared device in a MMF environment. RDM function 2 does not drop out until the reconfiguration completes.

5. The first machine to obtain the DAT interlock (machine A) calls RDM to scan through the MSTs of all shared devices. When RDM finds that none of these MSTs contain reconfiguration reply machine masks in word ACGL (bits 7 to 10), it returns a reply of 4000B to CONFIG.

6. When CONFIG receives the 4000B reply it gets the first equipment to be reconfigured from its list, displays the current configuration for that equipment on the K-display, and accepts and verifies input parameters for the reconfiguration of this equipment.

7. CONFIG calls RDM to process the reconfiguration request on machine A when the GO command is entered. RDM sets the redefinition in progress bit (DDLL bit 59). This insures that any PPs accessing the device execute the LDAM function and get not ready status.

8. RDM on machine A sets the reconfiguration reply machine mask for machine A in word ACGL and waits for the response from other machines in the complex. It does this by dropping the DAT interlock and looping, checking for the reconfiguration reply machine masks of the other machines that are accessing the device (as found in the DAT entry). Meanwhile, CMS in machine B obtains the DAT interlock and calls CONFIG. CONFIG sets up the K-display on machine B and calls RDM to scan the MSTs of shared devices for reconfiguration reply machine masks. When it comes across the device being processed by machine A, it finds the reconfiguration reply machine mask of machine A in word ACGL and returns the equipment number (EST ordinal) to CONFIG.

9. CONFIG on machine B now accepts and processes K-display input and calls RDM when GO is entered. RDM sets the redefinition in progress bit and sets the machine B reconfiguration reply machine mask in word ACGL. Next, it drops the DAT interlock and begins to loop in the same manner as machine A.

10. When all machines in the complex that are accessing the device concur with the replace or switch, having entered their reconfiguration reply machine masks in word ACGL, the messages SPIN DOWN UNIT x and SPIN DOWN UNIT y are flashed to the operator on the B display as necessary until the unit to be replaced and the unit replacing it are both spun down. Then the message SWITCH PACK/SPIN UP UNIT y is flashed on the display.

11. RDM on each machine monitors the pack switch to insure that the correct pack is mounted on the correct unit. While monitoring the switch, the pack label is validated. Each machine clears their own reconfiguration reply machine mask after they have validated the label. After clearing this bit they loop, waiting for the redefinition requested bit to be cleared. The last machine to clear its own reconfiguration reply machine mask then clears the redefinition requested bit.

12. Once the redefinition requested bit is cleared, each machine sets the new unit number in the unit list in word DDLL and clears their redefinition in progress bit. This allows the system to access the device. Control is then returned to CONFIG on all machines and the reconfiguration runs proceed as in step 5.

The following is the sequence CONFIG and RDM follow while replacing or switching units on a nonshared device in MMF mode, or any device in single mainframe mode. If reconfiguring a nonshared device in MMF mode, the machine will have and hold the DAT interlock throughout the entire reconfiguration.

5. CONFIG accepts and verifies the K-display input. When GO is entered, RDM processes the request. The channels connected to the device are attached, the redefinition in progress bit is set.

6. The SPIN DOWN UNIT x, SPIN DOWN UNIT y, and SWITCH PACK/SPIN UP UNIT y messages are flashed to the operator on the B display as necessary.

7. RDM monitors the pack switch and the pack label is validated. The redefinition requested bit is then cleared, the new unit number set in the unit list and the redefinition in progress bit cleared. The system now is allowed to access the device.

USER/CIO INTERFACE

Combined input/output (CIO) processes input/output requests for CPU programs. Data transfer between CIO and the CPU program is handled via a buffer within the CPU program's field length. This buffer is known as a circular buffer because CIO treats the last word and the first word as contiguous. The circular buffer is controlled via a file environment table (FET) which is also within the job's field length. The FET not only describes the buffer, but also holds the request code being issued to CIO. Figure 9-1 shows the relationship between CIO, the FET, and the circular buffer. For a write operation, at least one PRU of data should be in the buffer. For a read operation, the buffer must have room to receive one PRU of data. Less than one PRU of data is transferred only if an end-of-record (EOR) is read or written.

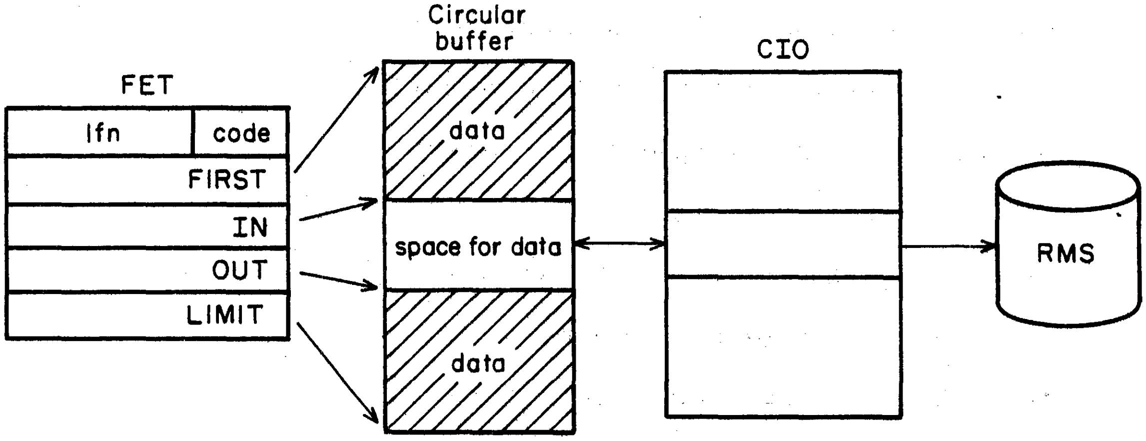


Figure 9-1. User/CIO Interface

The FET formats for mass storage and magnetic tape files are described in detail in the NOS Reference Manual, Volume 2 (refer to preface for publication number).

Equipment which may be accessed by CIO includes:

- Mass storage (MS)
- Magnetic tape (MT or NT)
- Time-sharing terminals (TT)
- Card reader (CR)*
- Card punch (CP)*
- Line printer (LP, LR, LS, or LT) *

Routines used by CIO include:

- 0BF - Begin file
- 0DF - Drop file
- 2LP - Write line printer *
- 2PC - Write card punch *
- 2RC - Read card reader *

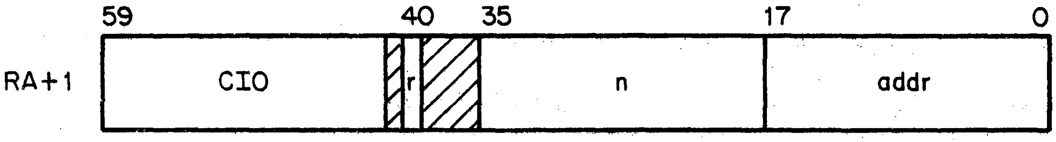
CIO consists of the following overlays:

- CIO - Main routine and termination
- 2CA - Identify special request
- 2CB - Read mass storage
- 2CC - Special mass storage reads
- 2CD - Write mass storage
- 2CE - Special mass storage writes
- 2CF - Position mass storage
- 2CG - Close mass storage
- 2CH - Terminal input/output
- 2CI - Magnetic tape operations
- 2CJ - Multifile label processor
- 2CK - Error processing
- 2CL - Issue dayfile message

---

* On-line drivers for these equipments are part of the maintenance package. They are referenced in this section for completeness only.

The call to CIO is formatted as follows:



r	Autorecall bit
n	Count for skip oprations
addr	Address of the FET

CIO MEMORY ALLOCATION

The allocation of PP memory by CIO is dependent upon the interrelationship of the various functions required by the processing overlays. In many cases, two overlays are associated with the process being performed; for example, write mass storage/special mass storage writes (2CD/2CE), and magnetic tape operation/multifile label processor (2CI/2CJ). In other cases, an overlay will require the read or write overlay; for example, close mass storage (2CG). In these cases, the load address is determined by which additional overlays must also be loaded. The error processing overlay (2CK) is loaded at the next address available after the maximum simultaneous overlay residence has been computed, so that the processing overlays are not disturbed during error processing.

Table 9-1 associates origin (load) addresses with the various overlays and the definition of the origin addresses.

Figure 9-2 illustrates PP memory as allocated by CIO. Figure 9-3 illustrates the CIO main overlay.

TABLE 9-1. ORIGIN ADDRESSES

Routine	Origin	Definition
2CA	OVL	End of tables and overlayable subroutines
2CB	MSDO	End of CIO resident subroutines
2CC	ERMS	End of 2CB
2CD	MSDO	End of CIO resident subroutines
2CE	EWMS	End of 2CD plus track table
2CF	PMSO	Maximum (end of 2CD, end of 2CB); this is not maximum (ERMS, EWMS) since EWMS includes the track table and PMSO does not
2CG	CLOO	Maximum (EWMS, amount loaded in PRUs for 2CD)
2CH	DRFW	Defined as 2000B in COMSCIO
2CI	OVL	End of tables and overlayable subroutines
2CJ	EMTO	End of 2CI
2CK	ERPO	Maximum (EWTO, ERDO, end of 2CF)
2CL	IDMO	ERPO plus end of subroutine /ERP/IMR
2CP	DRFW	Defined as 2000B in COMSCIO
2PC	DRFW	Defined as 2000B in COMSCIO
2RC	DRFW	Defined as 2000B in COMSCIO

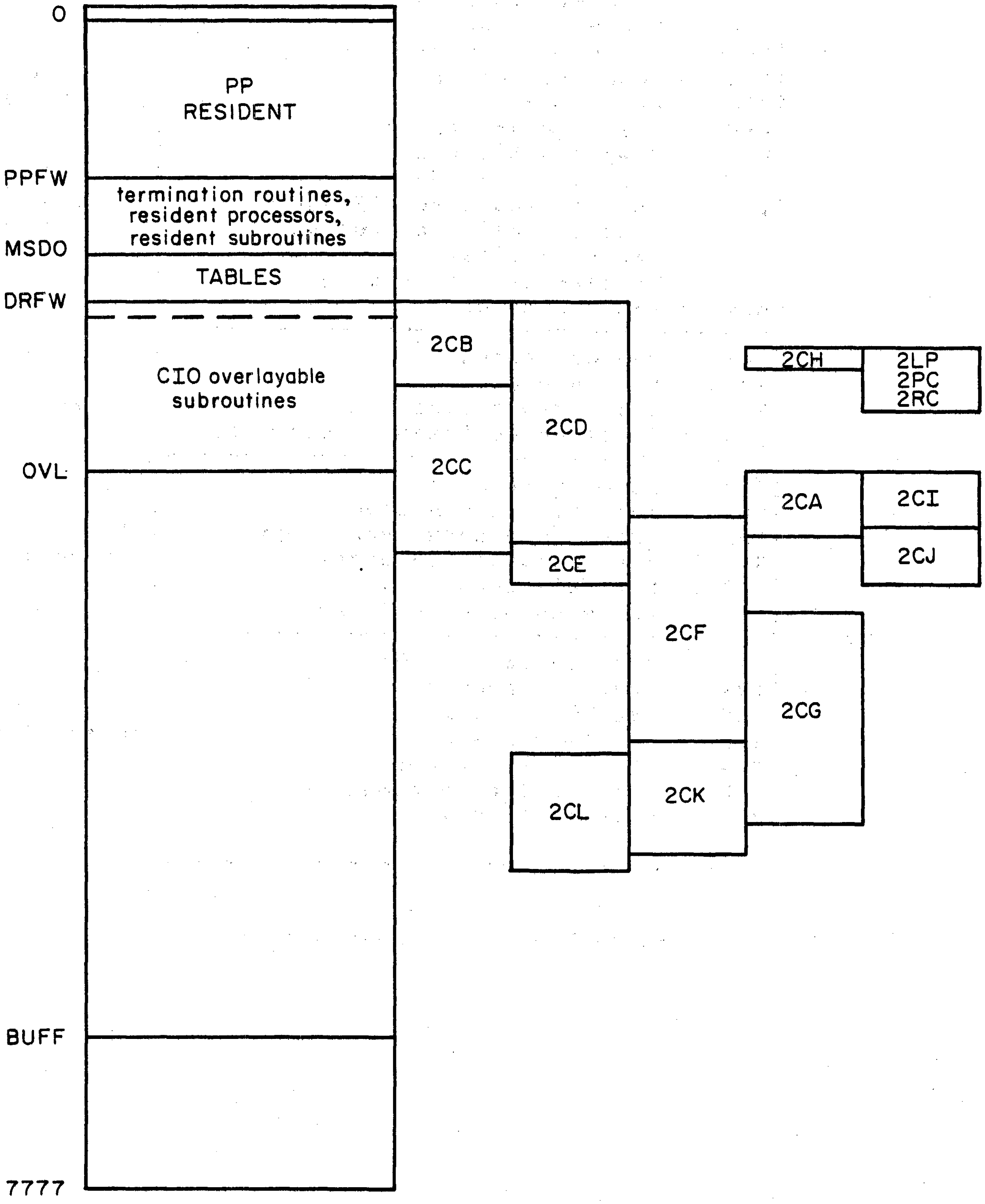


Figure 9-2. CIO PP Memory Allocation

```
 PPFW -----------------------------------------
      |UFS  - Update file status              |Termination
      |RRF  - Reset random FET pointers       |routines
      |IOF  - Set IN=OUT=FIRST                |
      |CFN  - Complete function               |
      |ERR  - Process error                   |
      |---------------------------------------|
      |CNR  - Complete null read              |Resident
      |DRF  - Drop file                       |processors
      |REW  - Rewind mass storage             |
      |MSP  - Complete MS processing          |
      |---------------------------------------|
      |CAF  - Compute absolute FET address    |Resident
      |DCC  - Drop channel when output        |subroutines
      |       register clear                  |
      |IMS  - Initialize mass storage         |
      |MSR  - Process MS error                |
 MSDO |---------------------------------------|
      |TREQ - Request codes                   |
      |TRDO - Read processors                 |
      |TWTO - Write processors                |Tables
      |TFCN - Function equipment processors   |
      |---------------------------------------|
      |EFN  - Enter file name                 |
      |FMS  - Function mass storage           |
      |FUE  - Function unknown equipment      |CIO
      |RUE  - Read unknown equipment          |subroutines
      |SSC  - Set skip count                  |
      |WUE  - Write unknown equipment         |
OVL   |---------------------------------------|
      |CIO1 - Read buffer status              |
      |IRQ  - Identify request                |
      |SAF  - Search for file                 |
      |SFS  - Set file status                 |Initialization
      |CFA  - Check file access               |
      |CBP  - Check buffer parameters         |
      |PFN  - Process function                |
      |                                       |
      |---------------------------------------|
      v                                       v
```

Figure 9-3. CIO - Main Overlay

CIO INITIALIZATION ROUTINES

Figures 9-4 through 9-10 are flowcharts for the following CIO initialization routines:

- CIO1/IRQ
- SAF
- EFN
- SFS
- CFA
- CBP
- PFN

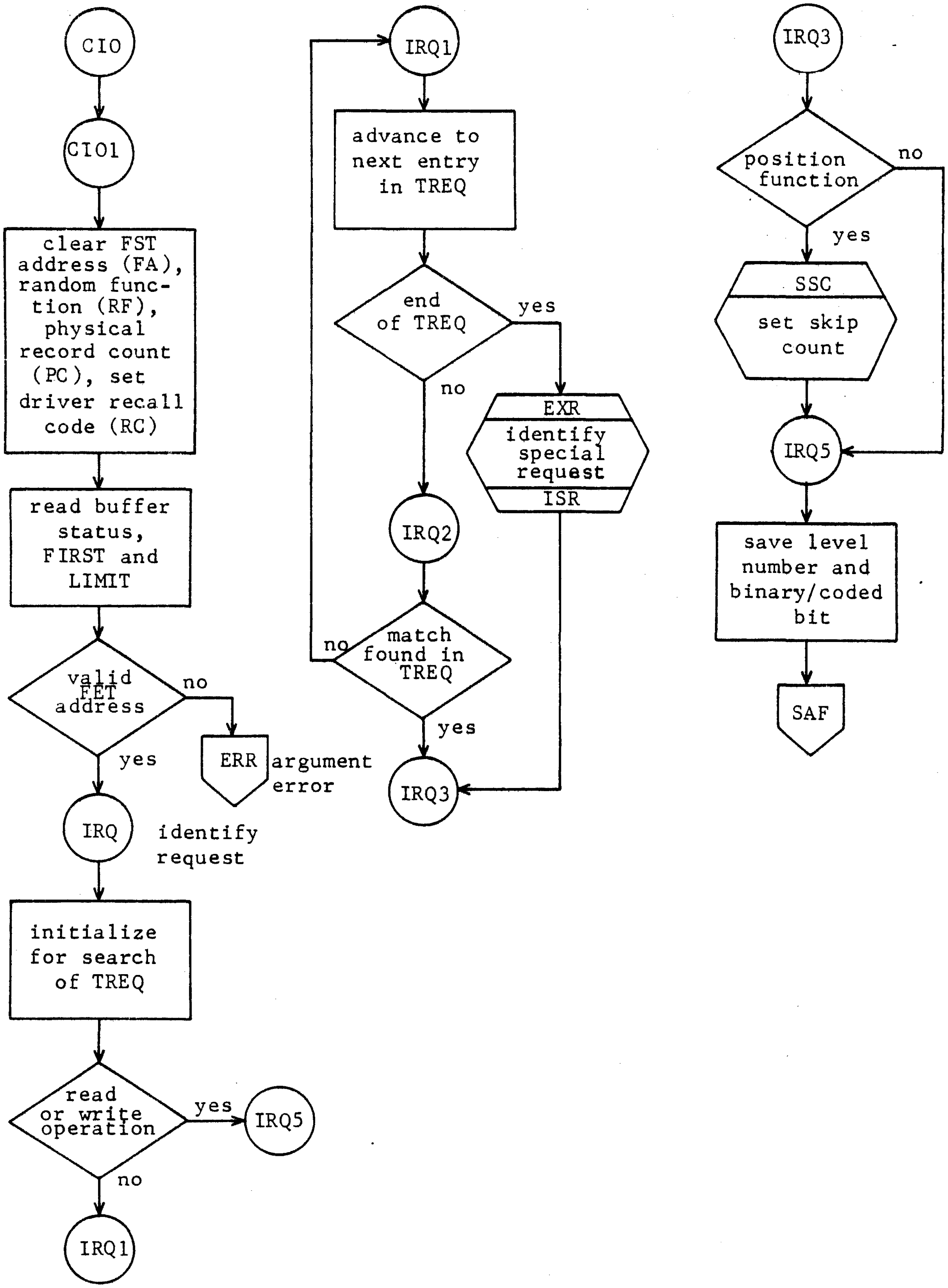


Figure 9-4. CIO1/IRQ - CIO Initialization

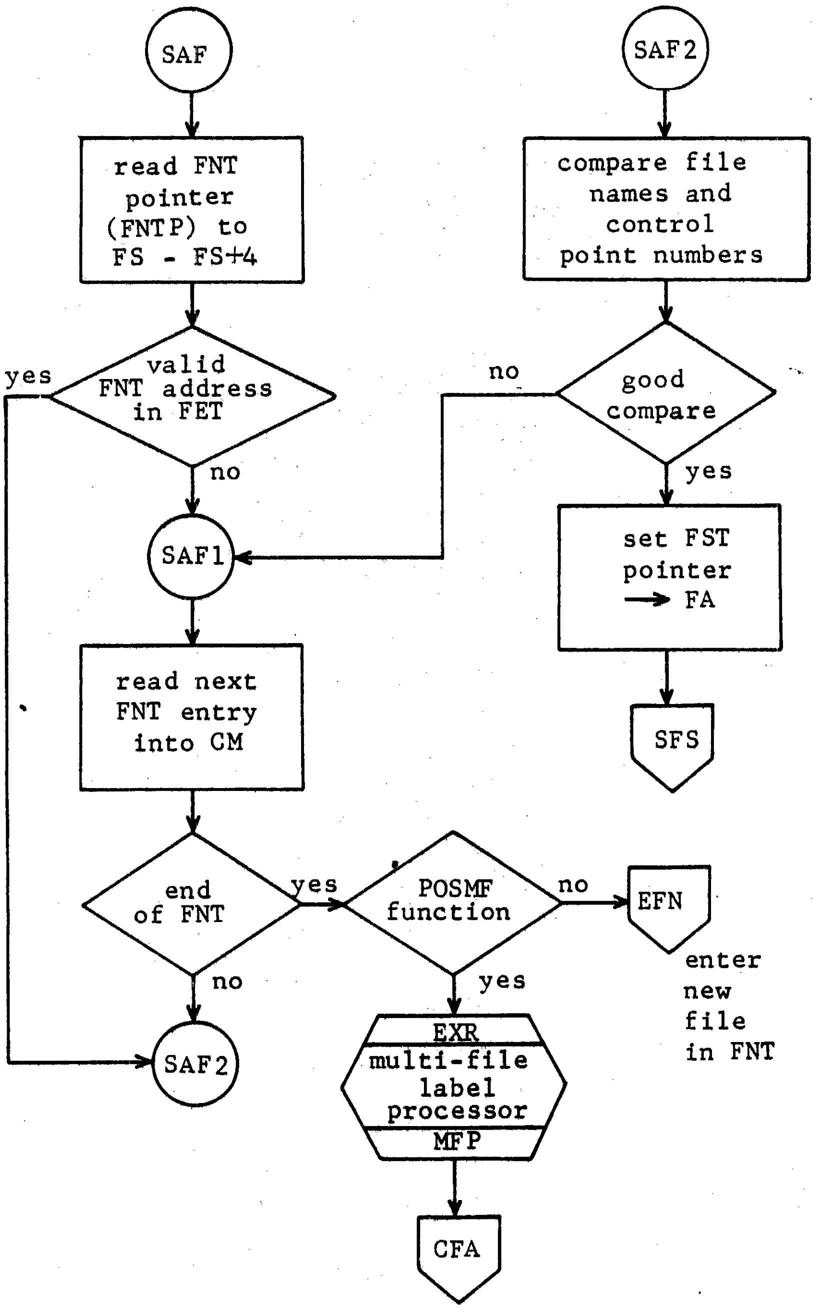


Figure 9-5. SAF - Search for Assigned File

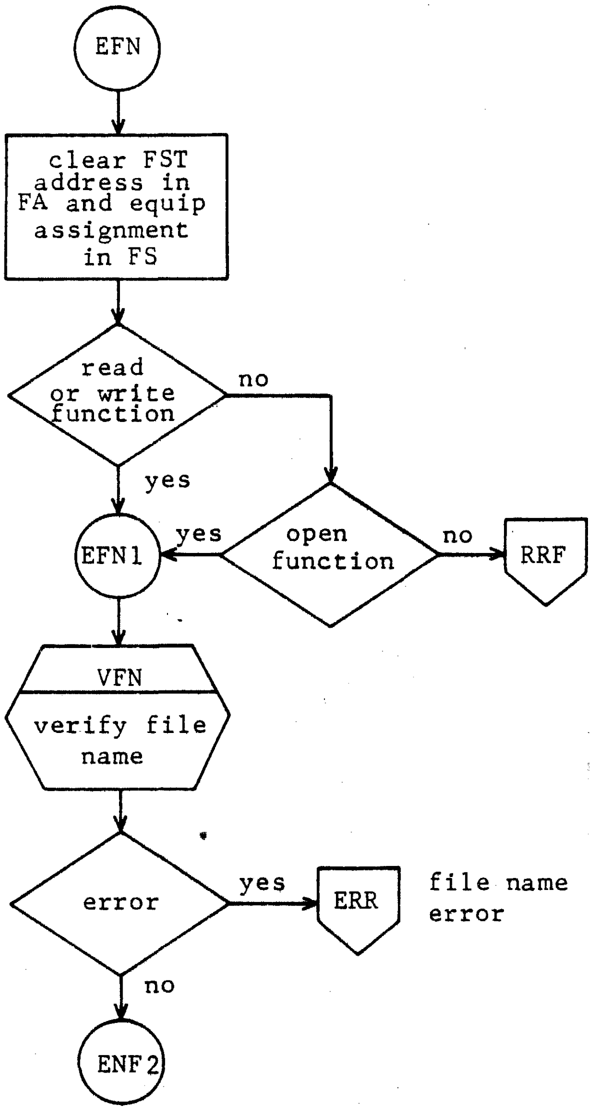


Figure 9-6. EFN - Enter File Name

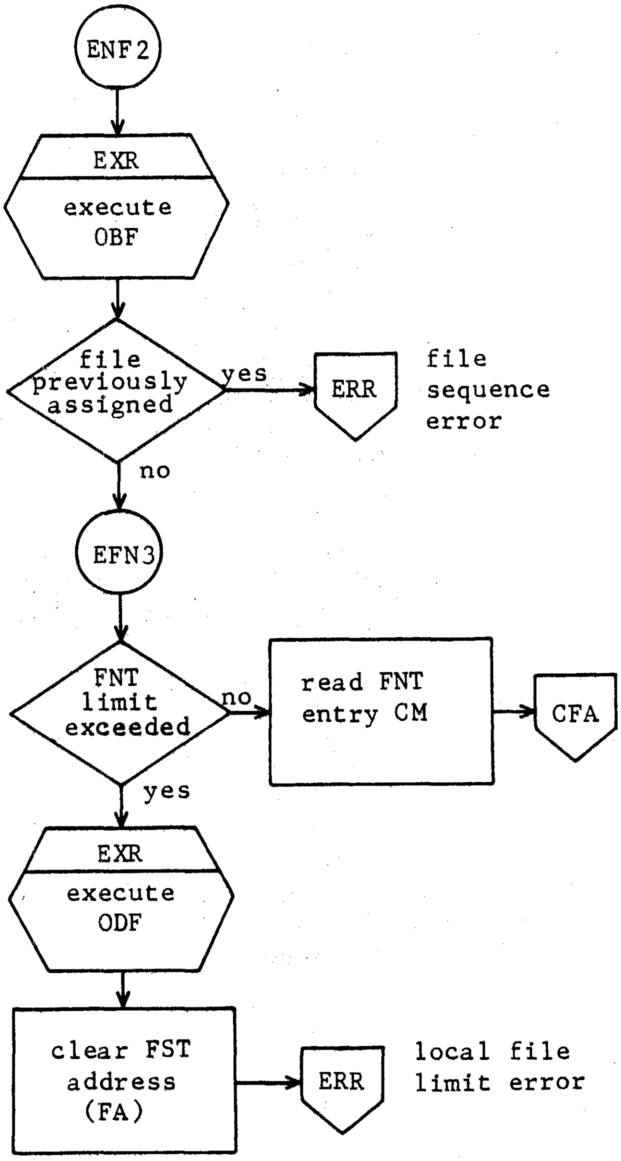


Figure 9-6. EFN - Enter File Name (Continued)

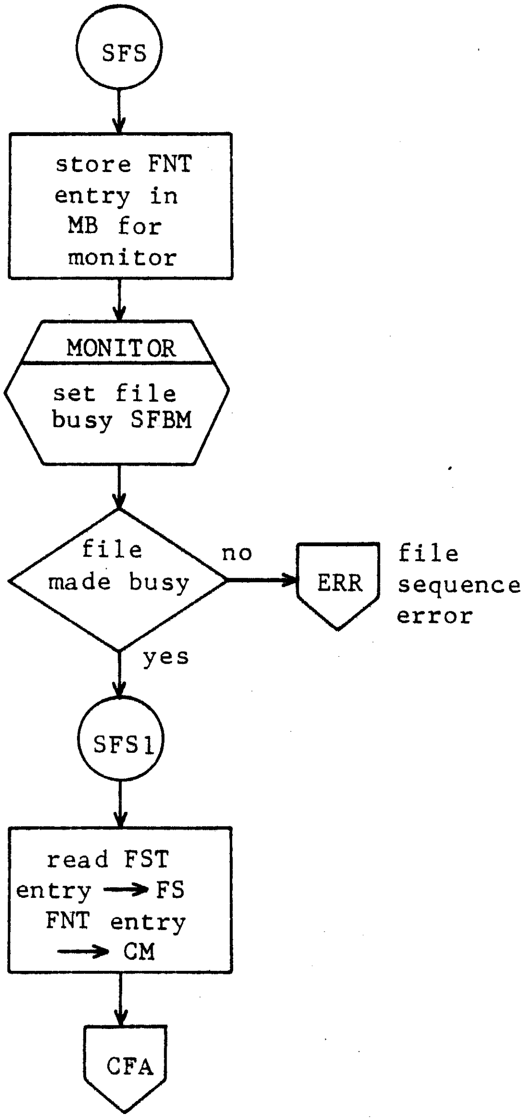


Figure 9-7. SFS - Set File Status

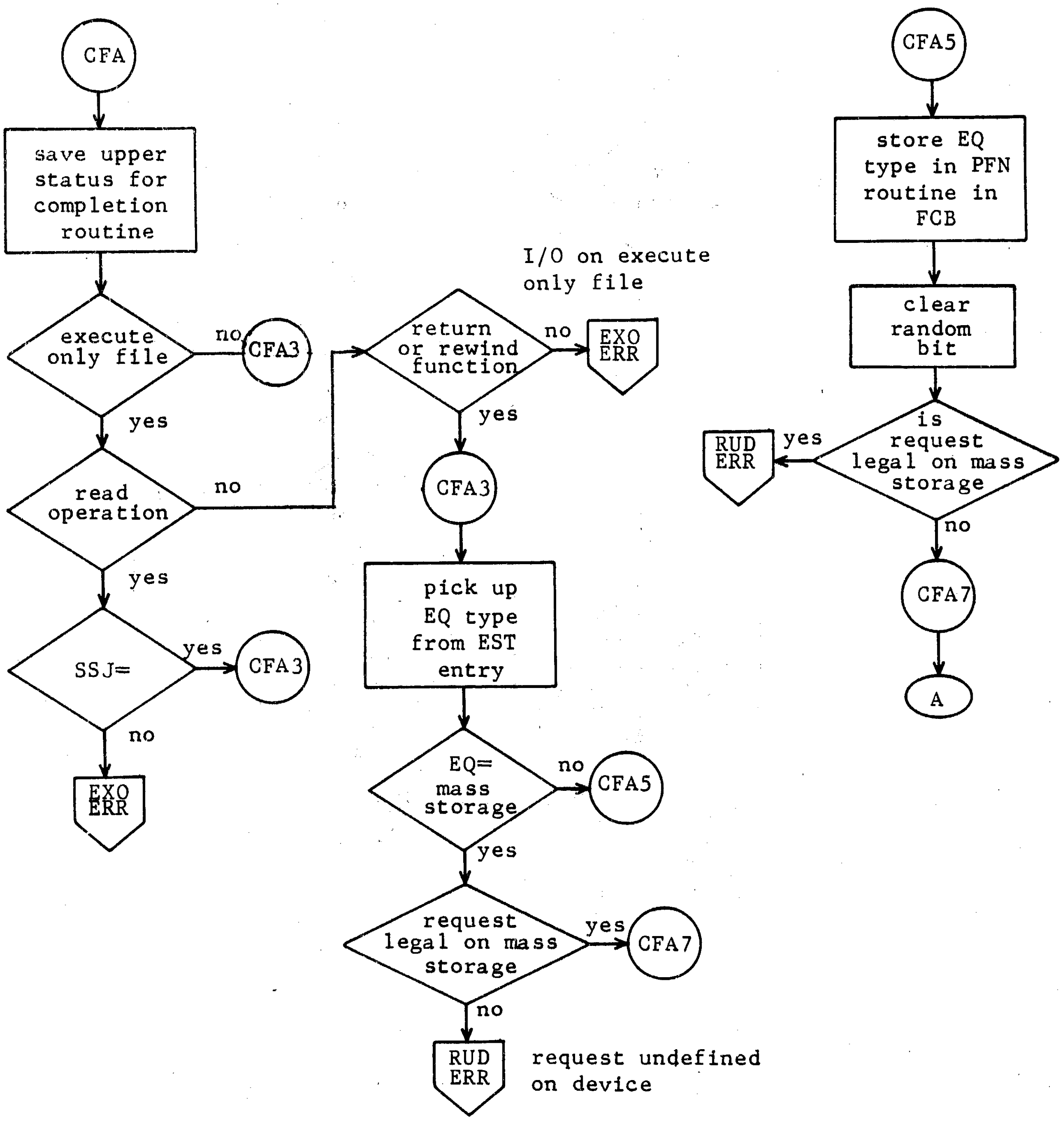


Figure 9-8. CFA - Check File Access

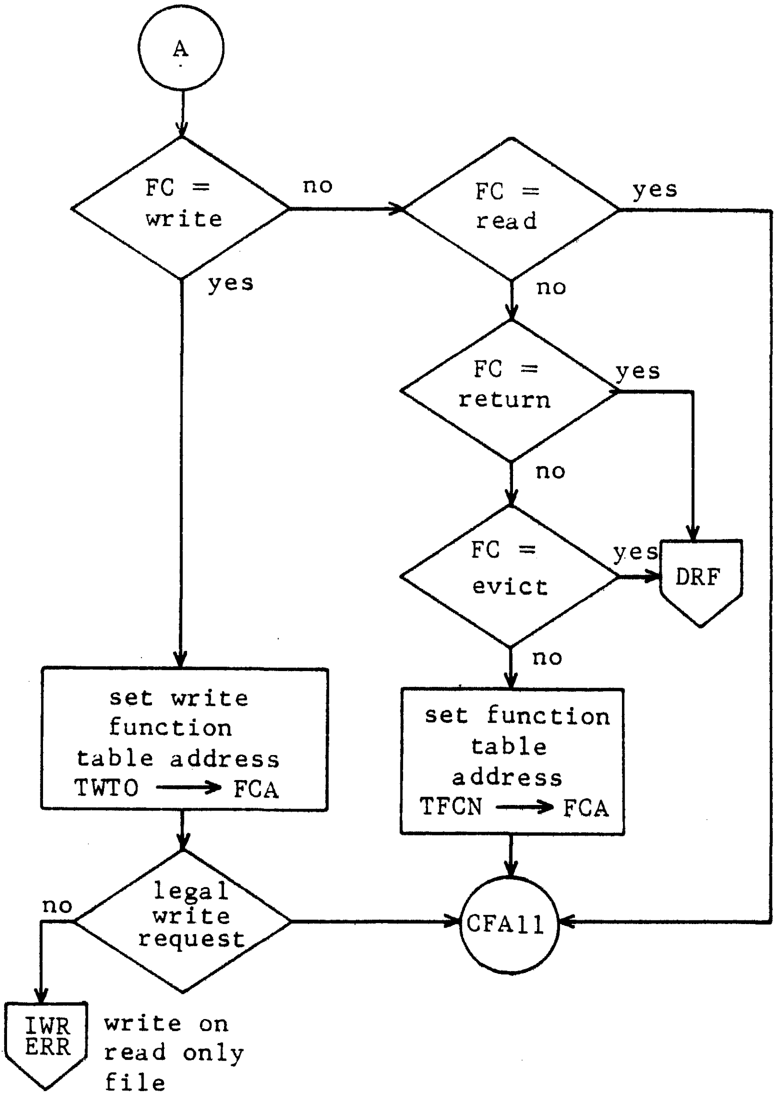


Figure 9-8. CFA - Check File Access (Continued)

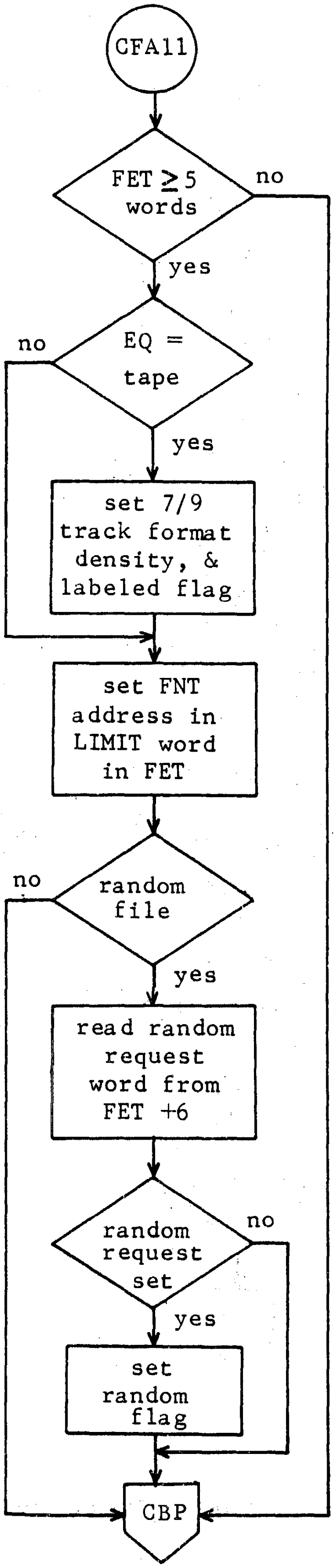


Figure 9-8. CFA - Check File Access (Continued)

Entry - FIRST and LIMIT already read by CIO1

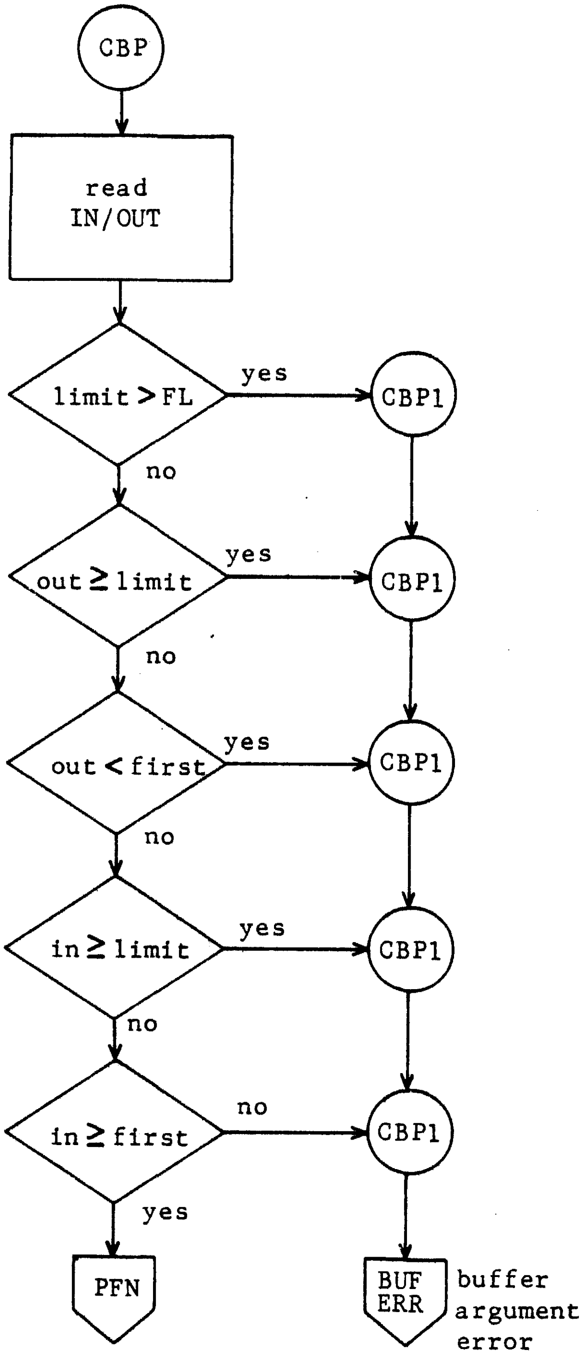


Figure 9-9. CBP - Check Buffer Parameters

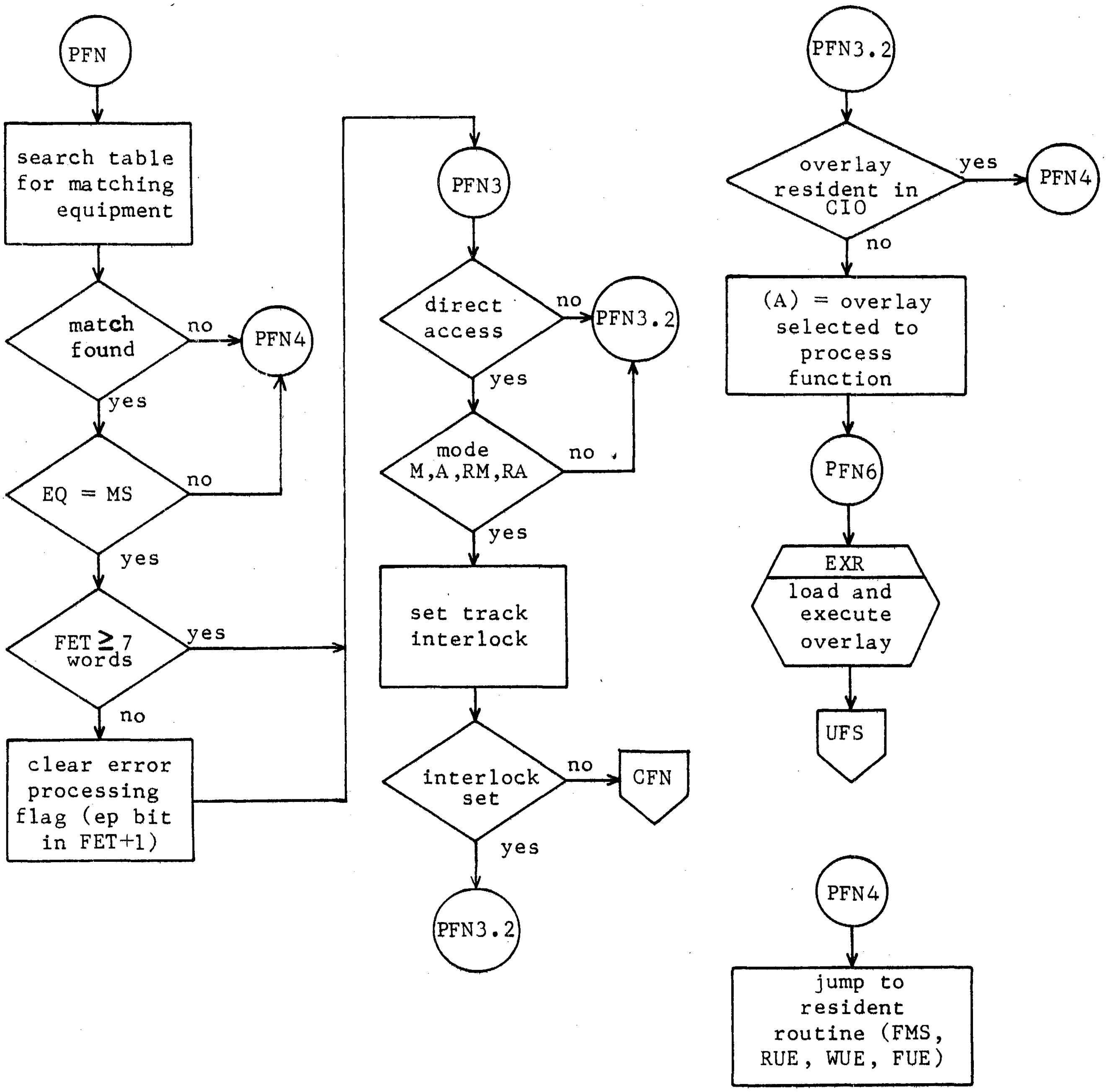


Figure 9-10. PFN - Process Function

The PFN routine searches one of three tables (TRDO, TWTO, or TFCN) to get the name of the overlay to be executed. The three tables are formatted as shown in tables 9-2 through 9-4.

TABLE 9-2. TRDO - TABLE OF READ PROCESSORS

Equipment	Entry Point	Overlay Name
MS	RMS	2CB
TT	TIO	2CH
MT	PMT	2CI
NT	PMT	2CI
CR		2RC
0	RUE	(Resident-read unknown equipment)

TABLE 9-3. TWTO - TABLE OF WRITE PROCESSORS

Equipment	Entry Point	Overlay Name
MS	WMS	2CD
TT	TIO	2CH
MT	PMT	2CI
NT	PMT	2CI
LP		2LP
CP		2PC
0	WUE	(Resident-write unknown equipment)

TABLE 9-4. TFCN - TABLE OF FUNCTION PROCESSORS

Equipment	Entry Point	Overlay Name
MS	FMS	(Resident)
MT	PMT	2CI
NT	PMT	2CI
0	FUE	(Resident-function unknown equipment)

CIO ERROR MESSAGES AND ROUTINES

Error messages from CIO are numbered and identified by a unique three-character name. Subroutines issuing an error message do so with the following code.

```
       LDN         /ERR/xxx

       LJM          ERR
```

All error messages are in overlay 2CK (table 9-5).

TABLE 9-5. OVERLAY 2CK

Name	Message
ARG	FET ADDRESS OUT OF RANGE
BLE	BUFFER CONTROL WORD ERROR ON
BUF	BUFFER ARGUMENT ERROR ON
DRE	DEVICE ERROR ON FILE
EXO	I/O ON EXECUTE ONLY FILE
FLN	ILLEGAL FILE NAME
FPE	FET PARAMETER ERROR ON
FSQ	I/O SEQUENCE ERROR ON FILE
IFE	ILLEGAL EXTENSION OF
IFM	ILLEGAL MODIFICATION OF
IRQ	ILLEGAL I/O REQUEST ON FILE
IWR	WRITE ON READ ONLY FILE
LFL	LOCAL FILE LIMIT, FILE
MFN	MULTI-FILE NAME NOT FOUND
OFL	OUTPUT FILE LIMIT, FILE
PRL	PRU LIMIT, FILE
RAD	RANDOM ADDRESS NOT ON FILE
RUD	REQUEST UNDEFINED ON DEVICE
RWT	INDEX ADDRESS OUT OF RANGE FOR
TKL	TRACK LIMIT, FILE
TNA	M.T. NOT AVAILABLE ON FILE

The logical file name and FET address follow the preceding messages. The error processing subroutine ERR is flowcharted in figure 9-11 and the overlay 2CK called by ERR is flowcharted in figure 9-12.

Entry - (A) = error number

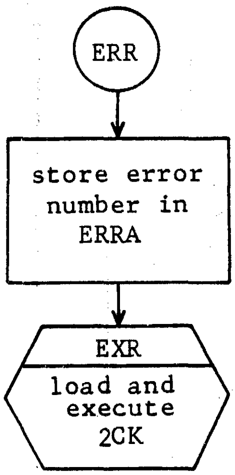


Figure 9-11. ERR - Process Error

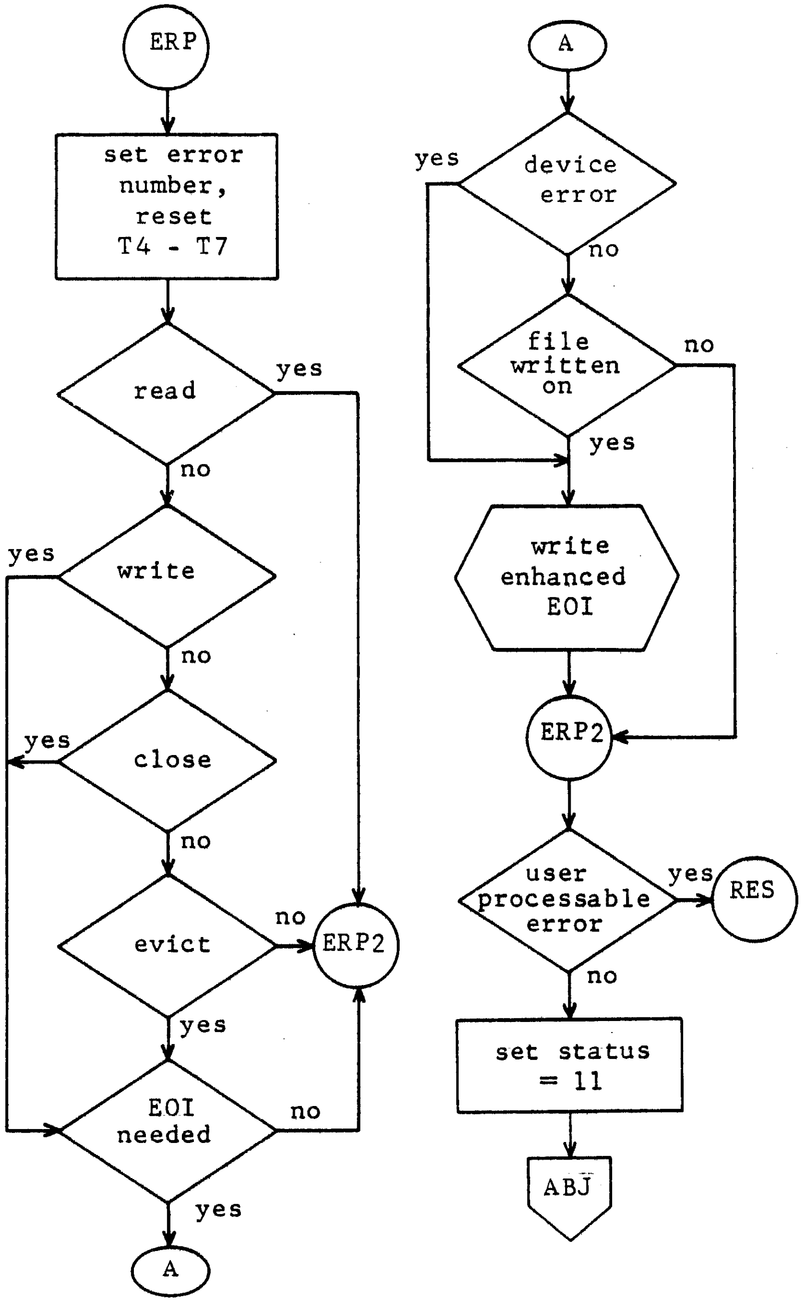


Figure 9-12. ERR - Error Processor (2CK)

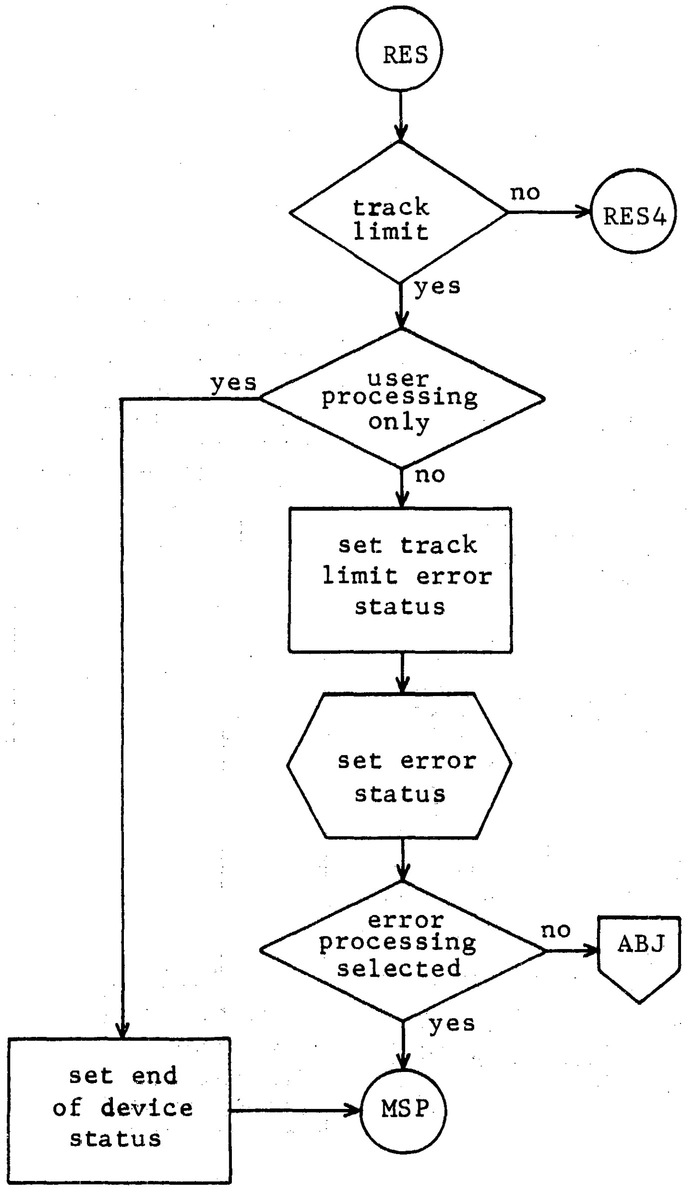


Figure 9-12. ERR - Error Processor (2CK) (Continued)

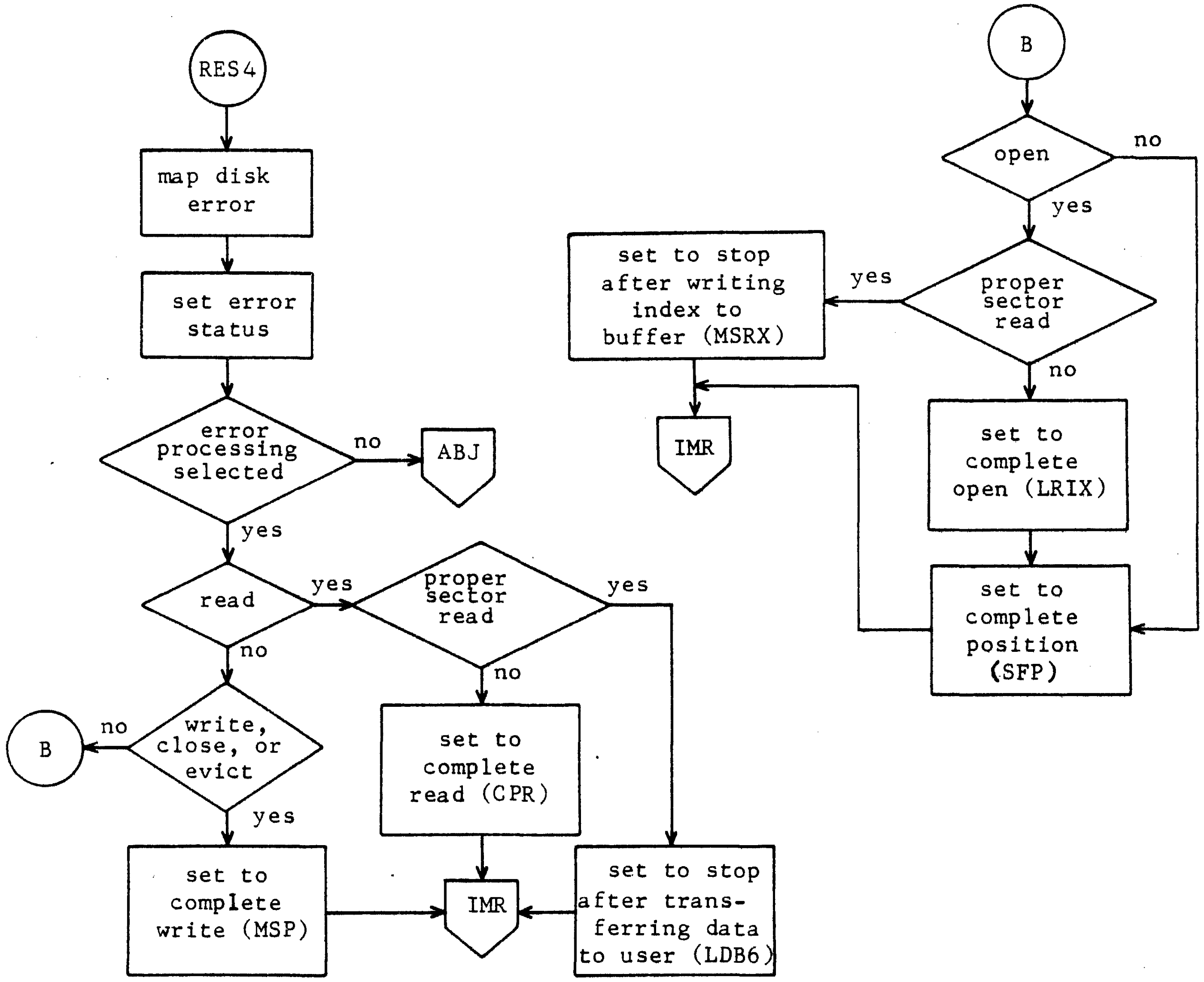


Figure 9-12. ERR - Error Processor (2CK) (Continued)

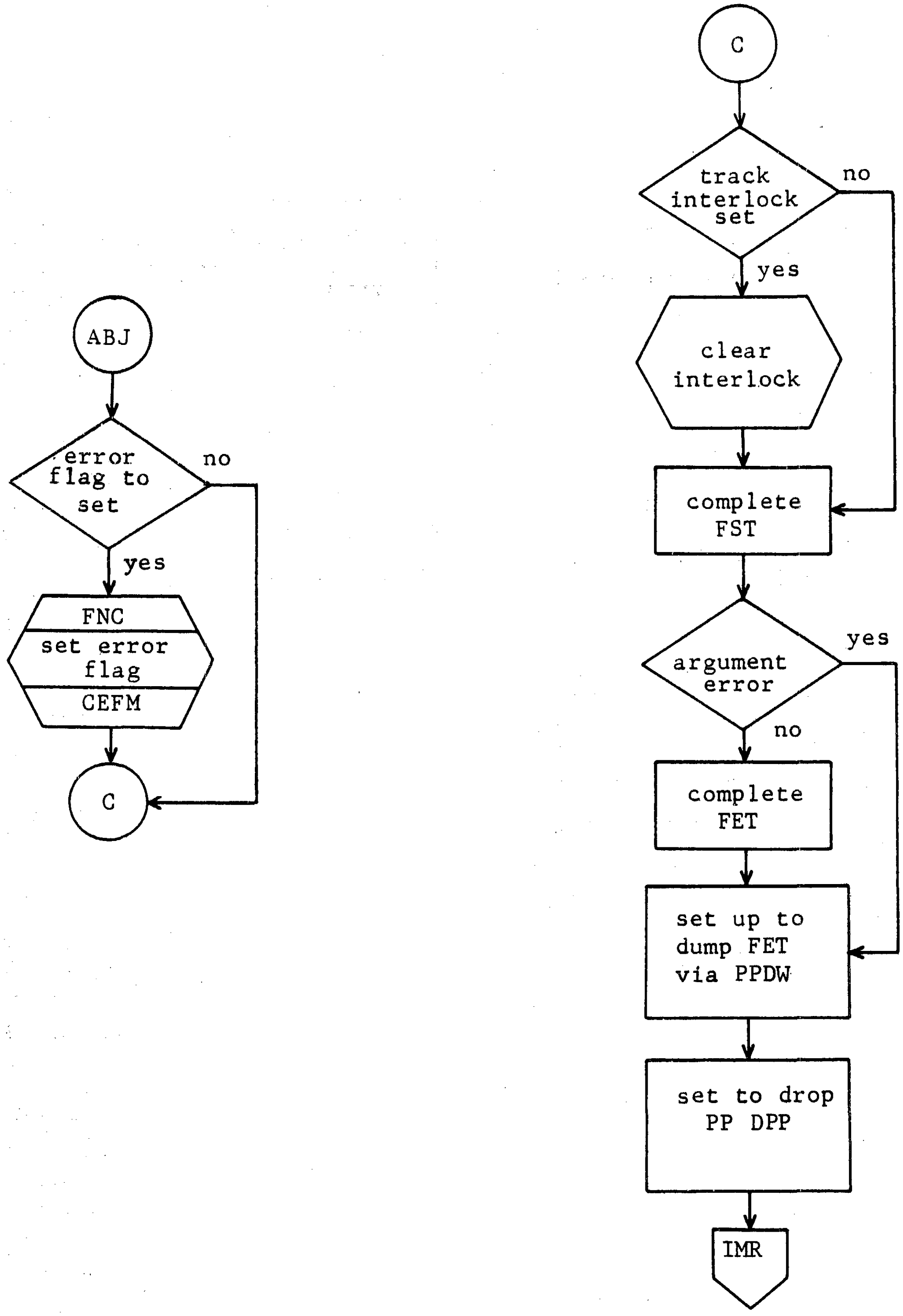


Figure 9-12. ERR - Error Processor (2CK) (Continued)

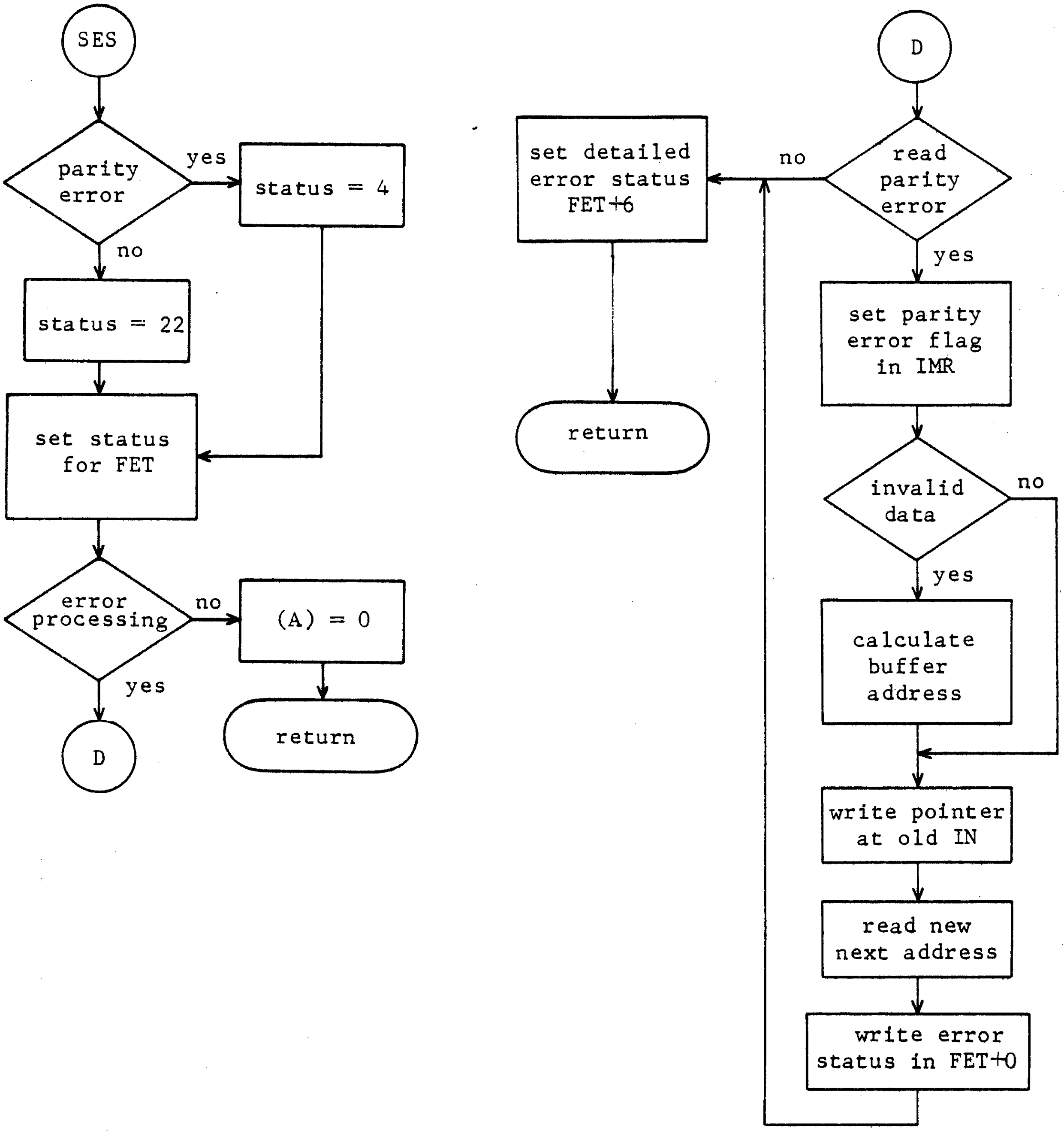


Figure 9-12. ERR - Error Processor (2CK) (Continued)

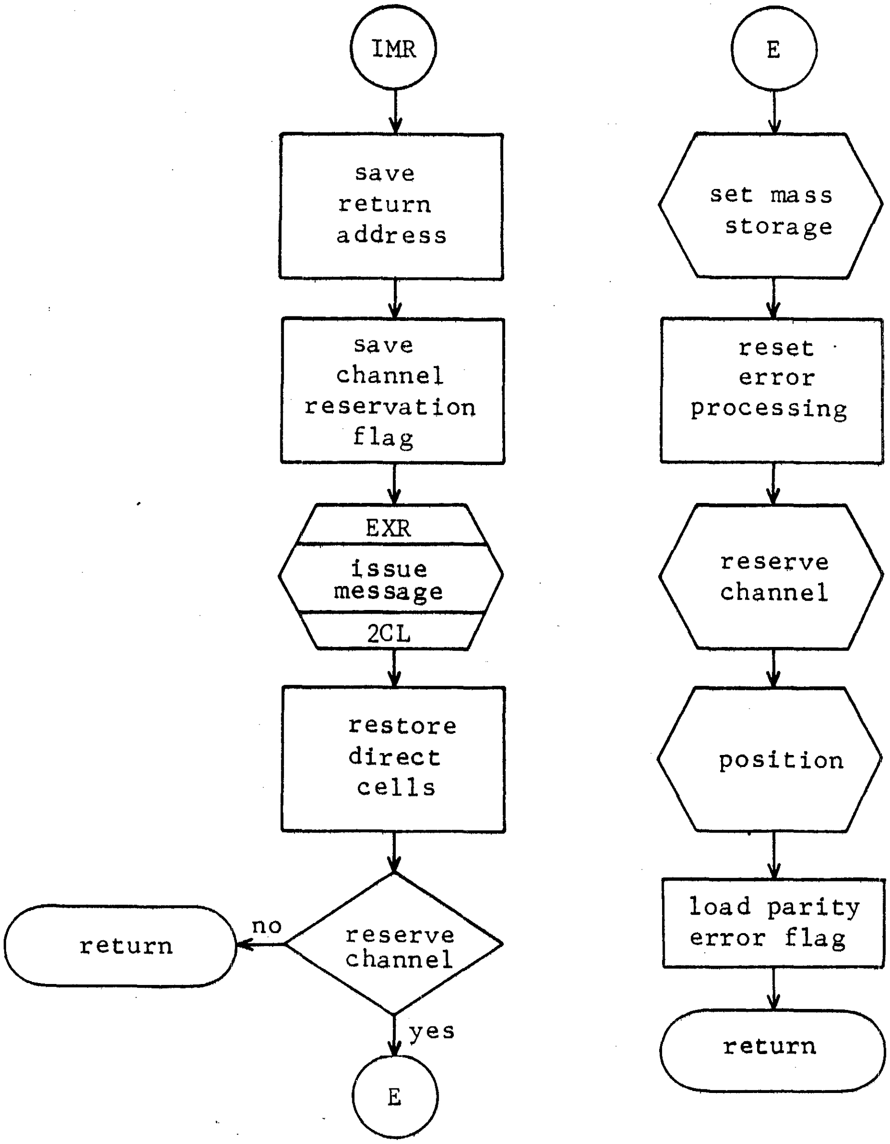


Figure 9-12. ERR - Error Processor (2CK) (Continued)

2CA SUBROUTINES

Figures 9-13 and 9-14 are the flowcharts of the three subroutines residing in overlay 2CA. These are:

- ISR - Identify special request
- EVF - Evict mass storage file
- EPF - Evict permanent file

Table 9-6 is searched to map the request code in BS+4 into a function code stored in PC.

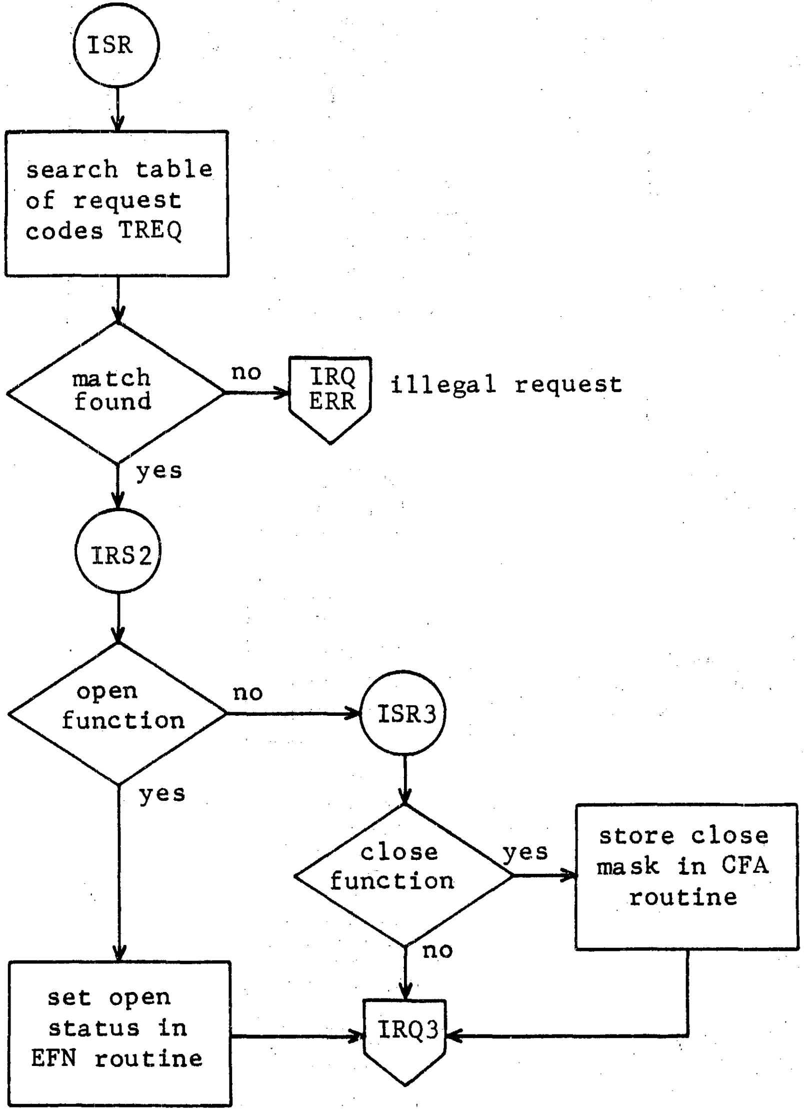


Figure 9-13. ISR - Identify Special Request (2CA)

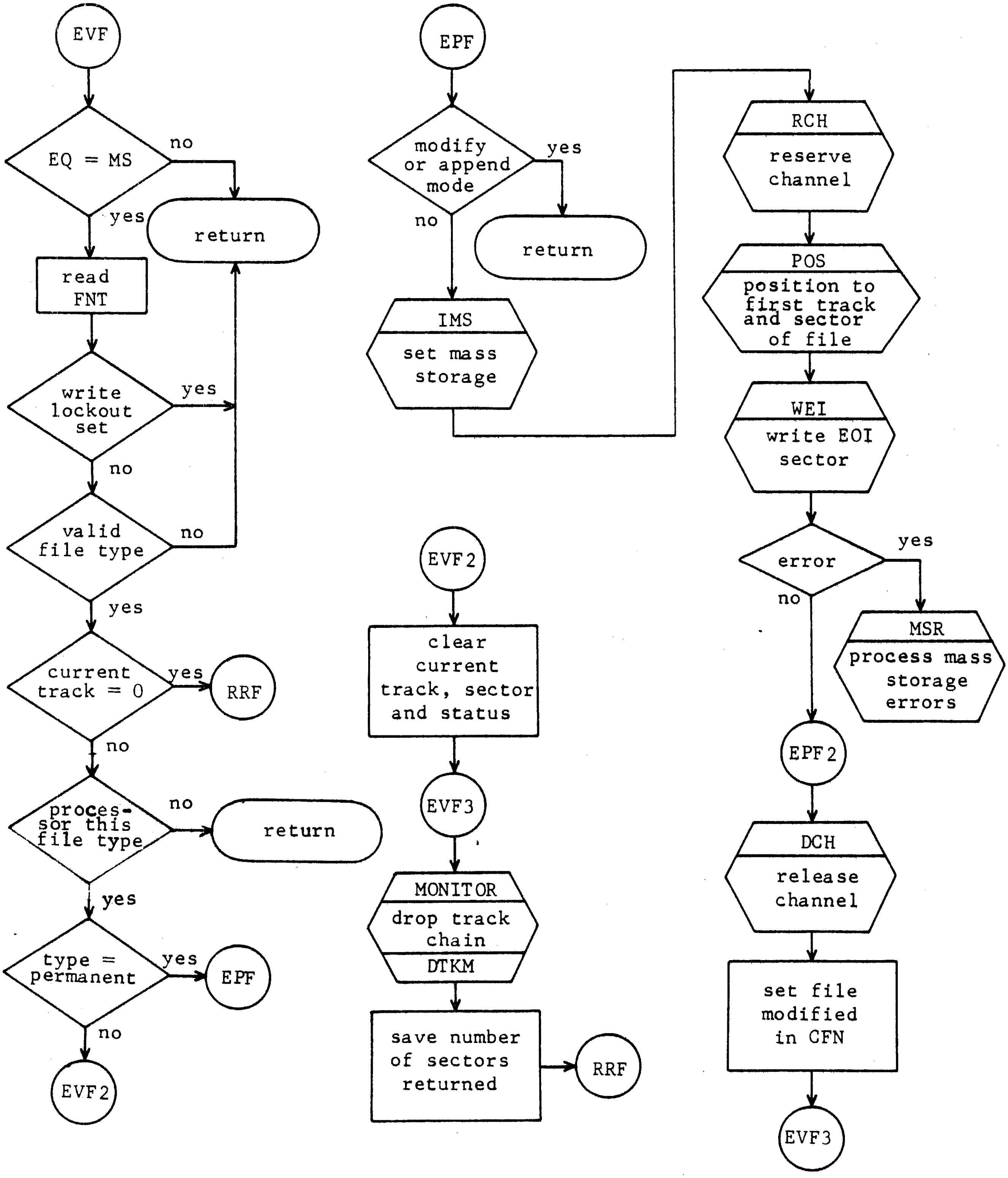


Figure 9-14. EVF/EPF -2CA Subroutines to Evict a Mass Storage or Permanent File

TABLE 9-6. TREQ

Request Code	Function Code Name	Description
0100	OPE	Open, read, no rewind
0104	OPE	Open, write, no rewind
0110	OPE	Position multifile set
0114	EVI	Evict
0120	OPE	Open, alter, no rewind
0130	CLO	Close, no rewind
0140	OPR	Open, read, rewind
0144	OPR	Open, write, rewind
0150	CLU	Close, rewind
0160	OPR	Open, alter, rewind
0170	CLU	Close, unload
0174	CLU	Close, unload, return
0300	OPE	Open, read, no rewind
0330	CLO	Close, no rewind
0340	OPR	Open, rewind
0350	CLU	Close, rewind
0370	CLU	Close, unload

2CB SUBROUTINES

Figures 9-15 through 9-20 are flowcharts of subroutines in overlay 2CB - read mass storage. The following is a list of those subroutines; CBS and SBA are not flowcharted.

- RMS - Read mass storage (main routines)
- LDB - Load CM buffer
- WCB - Write central buffer
- EOF - Process EOF
- EOR - Process EOR
- CPR - Complete read
- CBS - Check buffer space
- SBA - Set buffer addresses

The flow is for a buffer read. If another read function is issued, the flow will be changed.

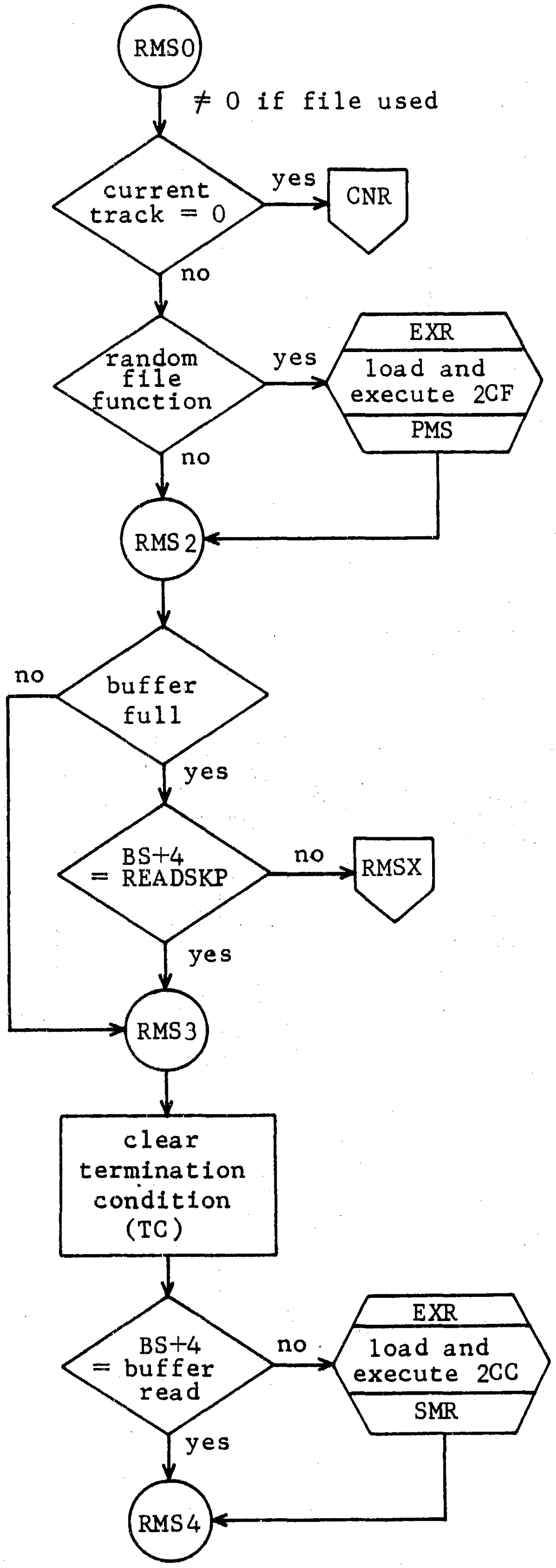


Figure 9-15. 2CB - Read Mass Storage

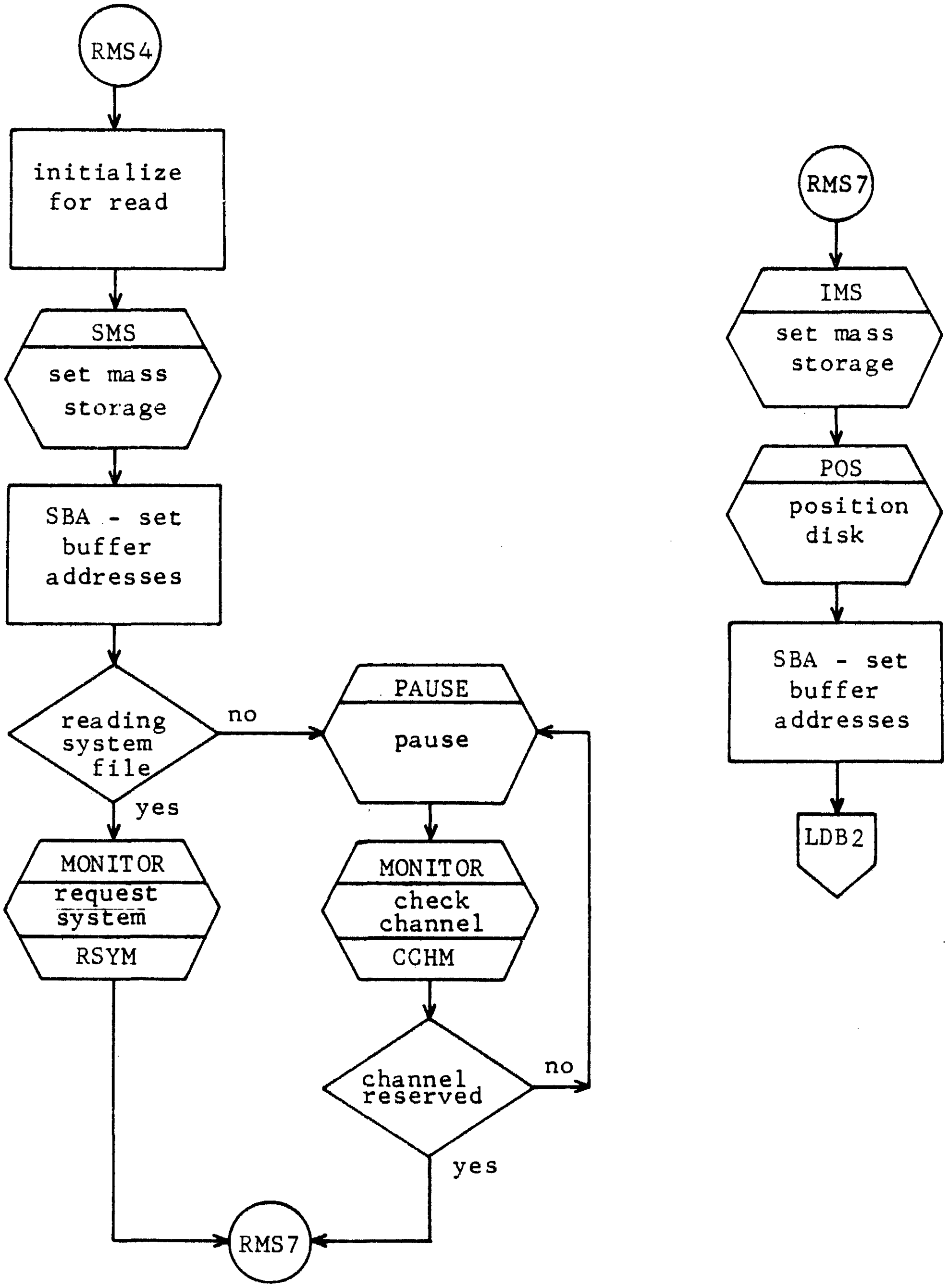


Figure 9-15. 2CB - Read Mass Storage (Continued)

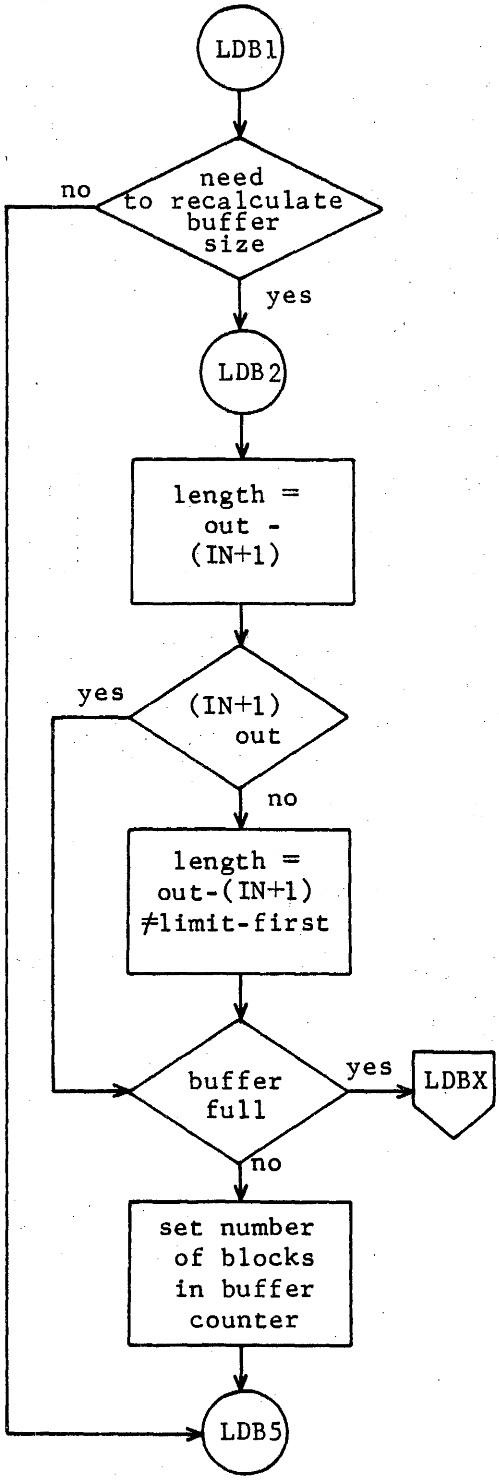


Figure 9-16. LDB - Load CM Buffer

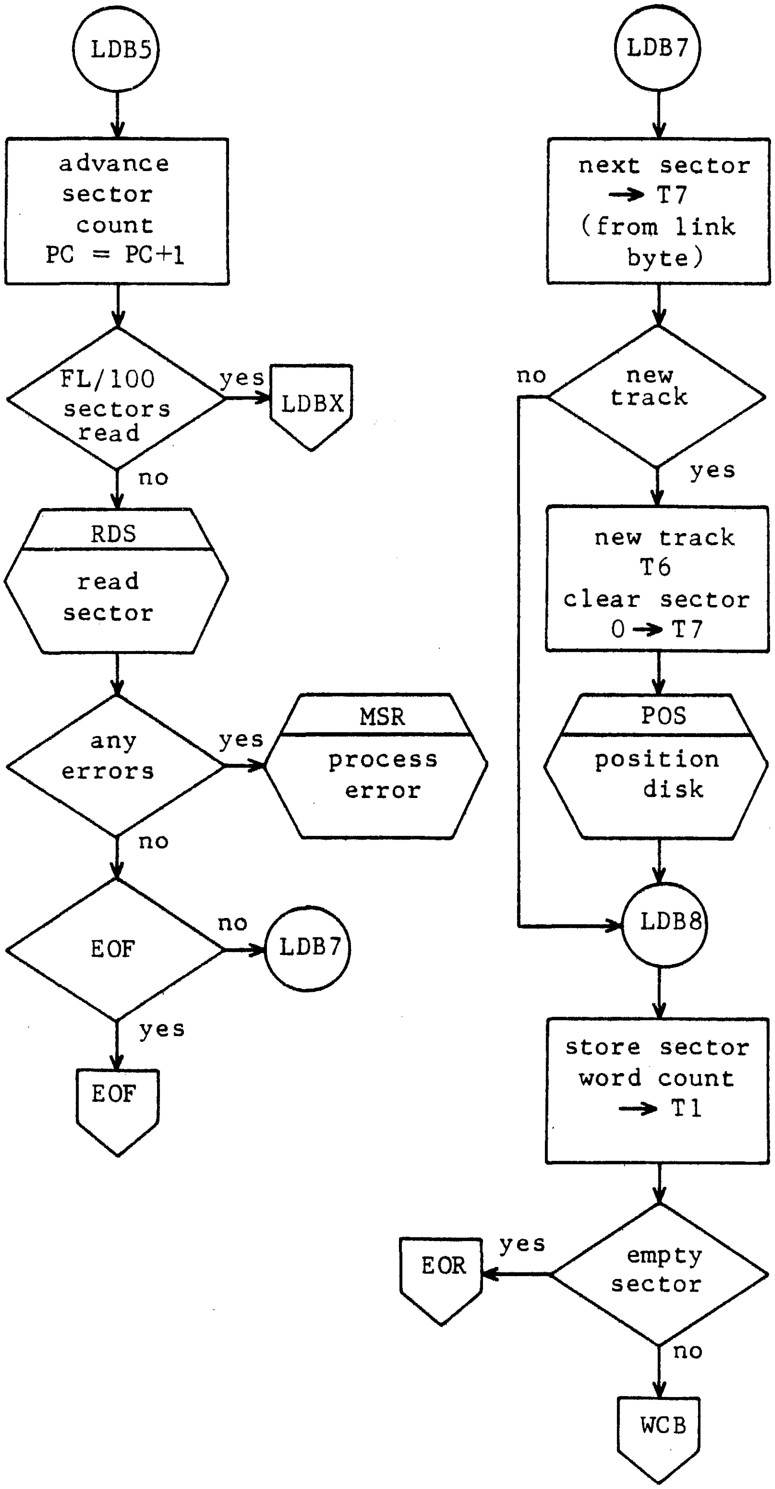


Figure 9-16. LDB - Load CM Buffer (Continued)

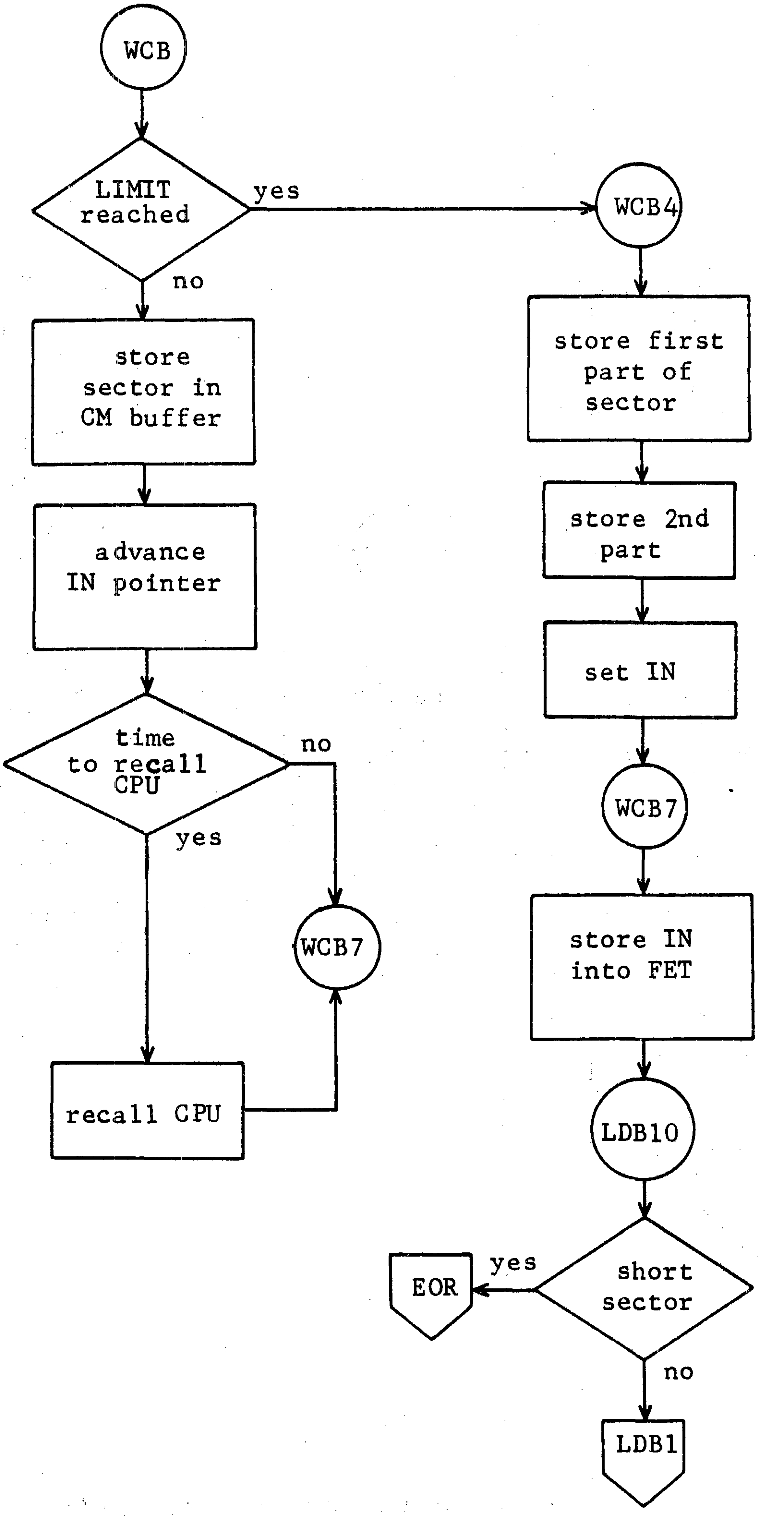


Figure 9-17. WCB - Write Central Buffer

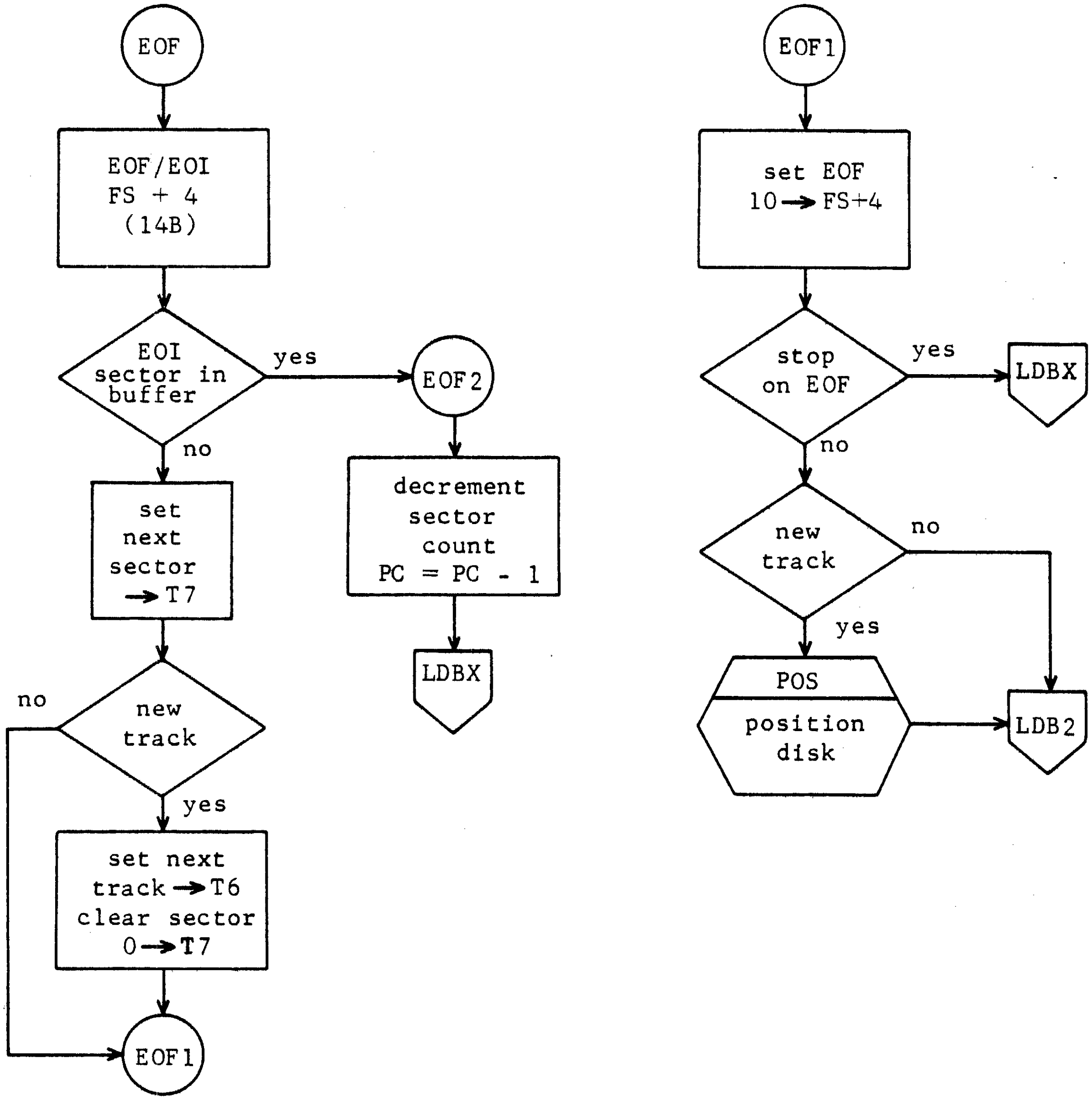


Figure 9-18. EOF - Process EOF

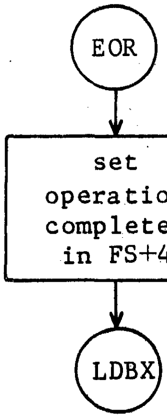


Figure 9-19. EOR - Process EOR

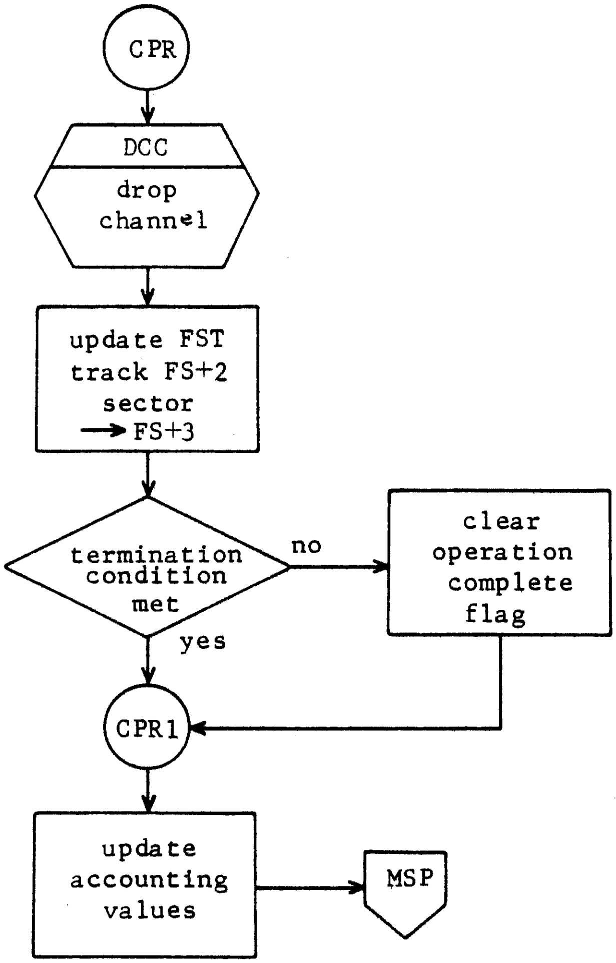


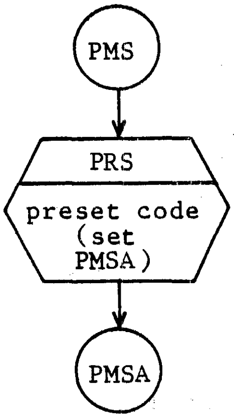
Figure 9-20. CPR - Complete Read

POSITION MASS STORAGE ROUTINE

Figure 9-21 is a partial flowchart of PMS. The position mass storage routine is in overlay 2CF. PMS is called from three places in CIO:

- Resident processor PMS
- RMS in 2CB
- WMS in 2CD

PMS - Position mass storage (2CF)



PMSA = RRD - process random read
RWT - random write
SKF - skip forward
SKB - skip backward
BKS - backspace
PMSX - rewind
OPE - open

Figure 9-21. PMS and Function Processor Return

Function processor return

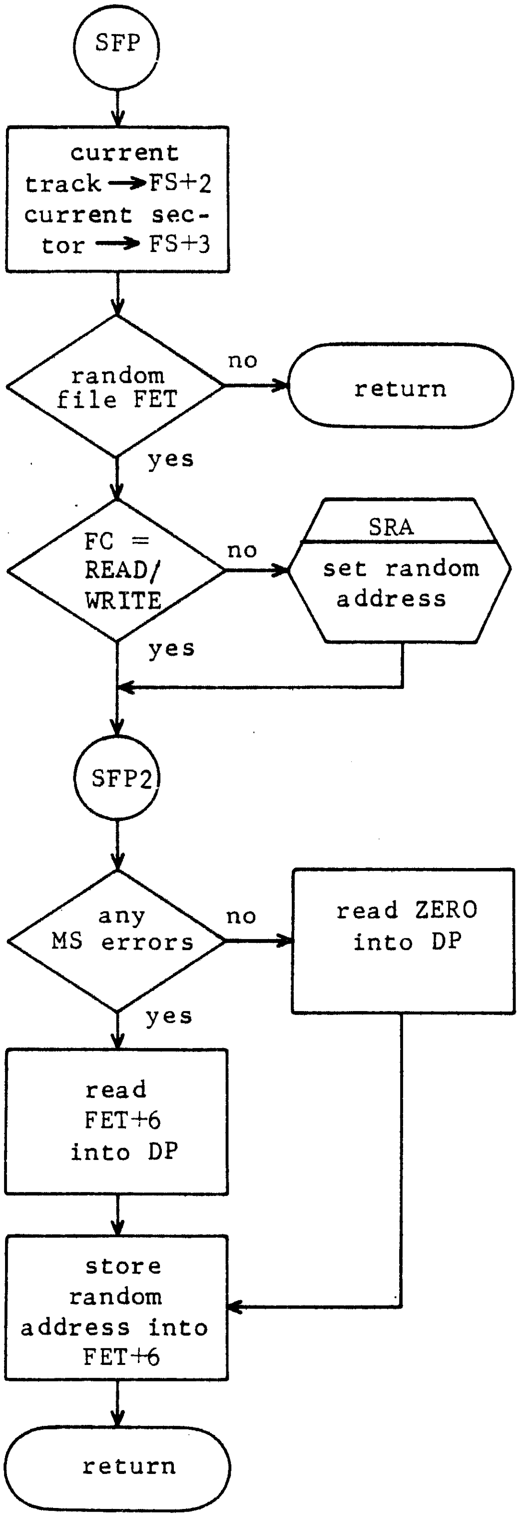


Figure 9-21. PMS and Function Processor Return (Continued)

CIO-TERMINATION ROUTINES

Figures 9-22 through 9-24 are flowcharts of the following CIO termination routines:

- UFS - Update file status
- IOF - Set IN=OUT=FIRST
- CFN - Complete function

Routine RRF (reset random FET pointers) is not flowcharted.

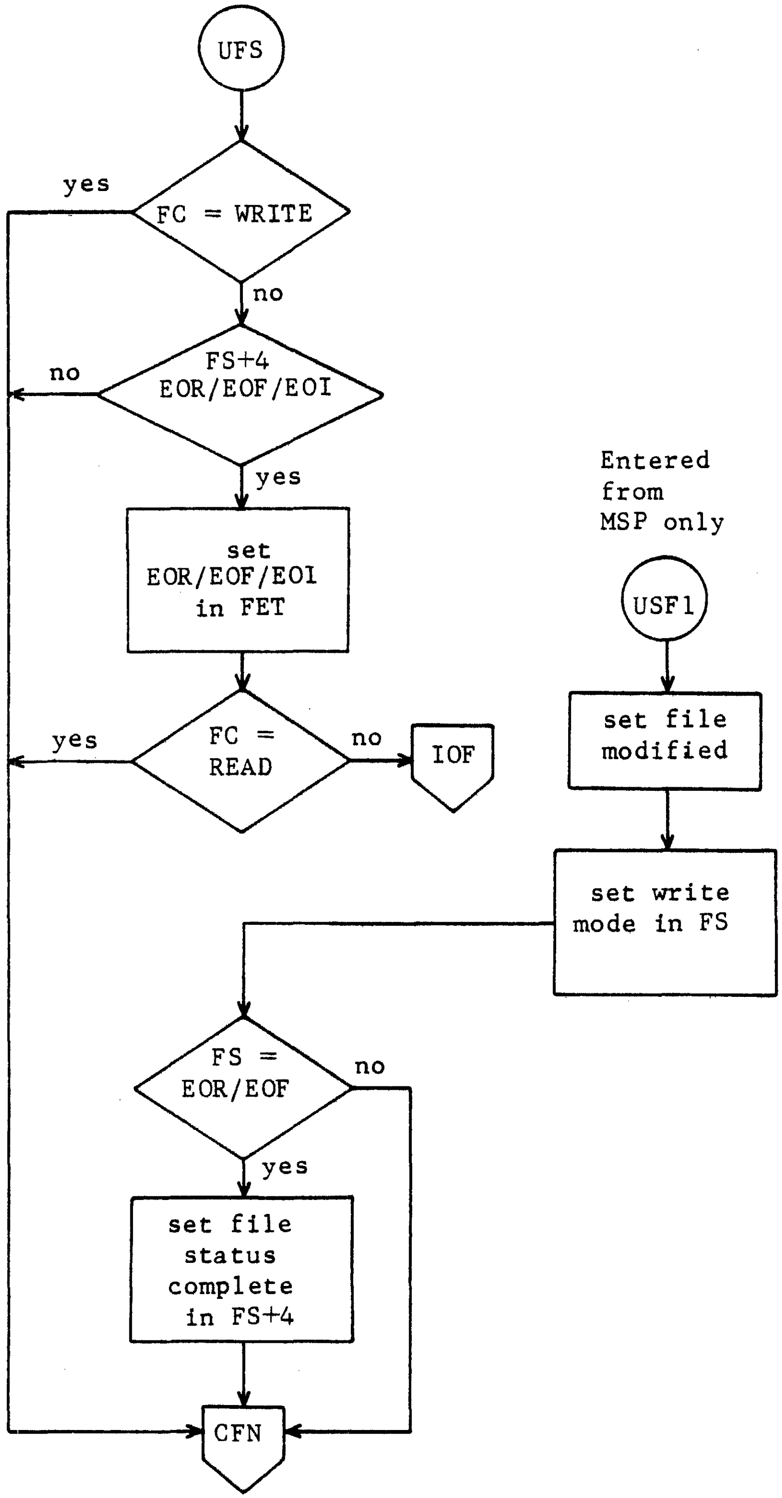


Figure 9-22. UFS - Update File Status

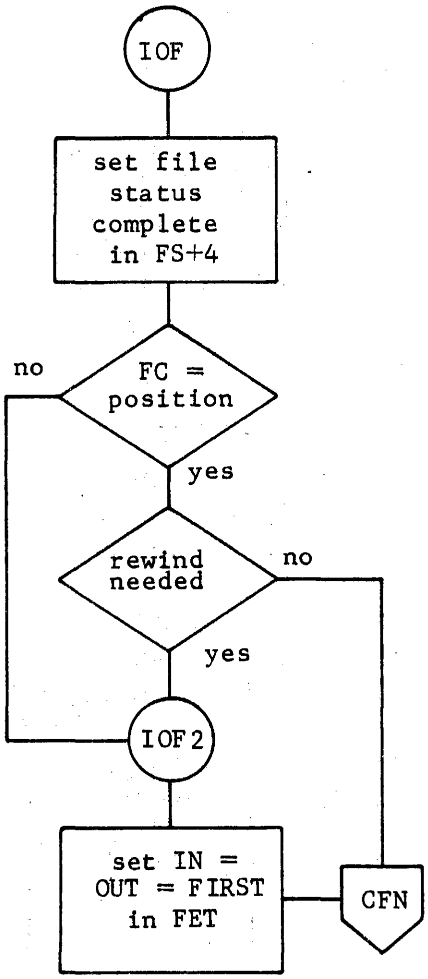


Figure 9-23. IOF - Set IN = OUT = FIRST

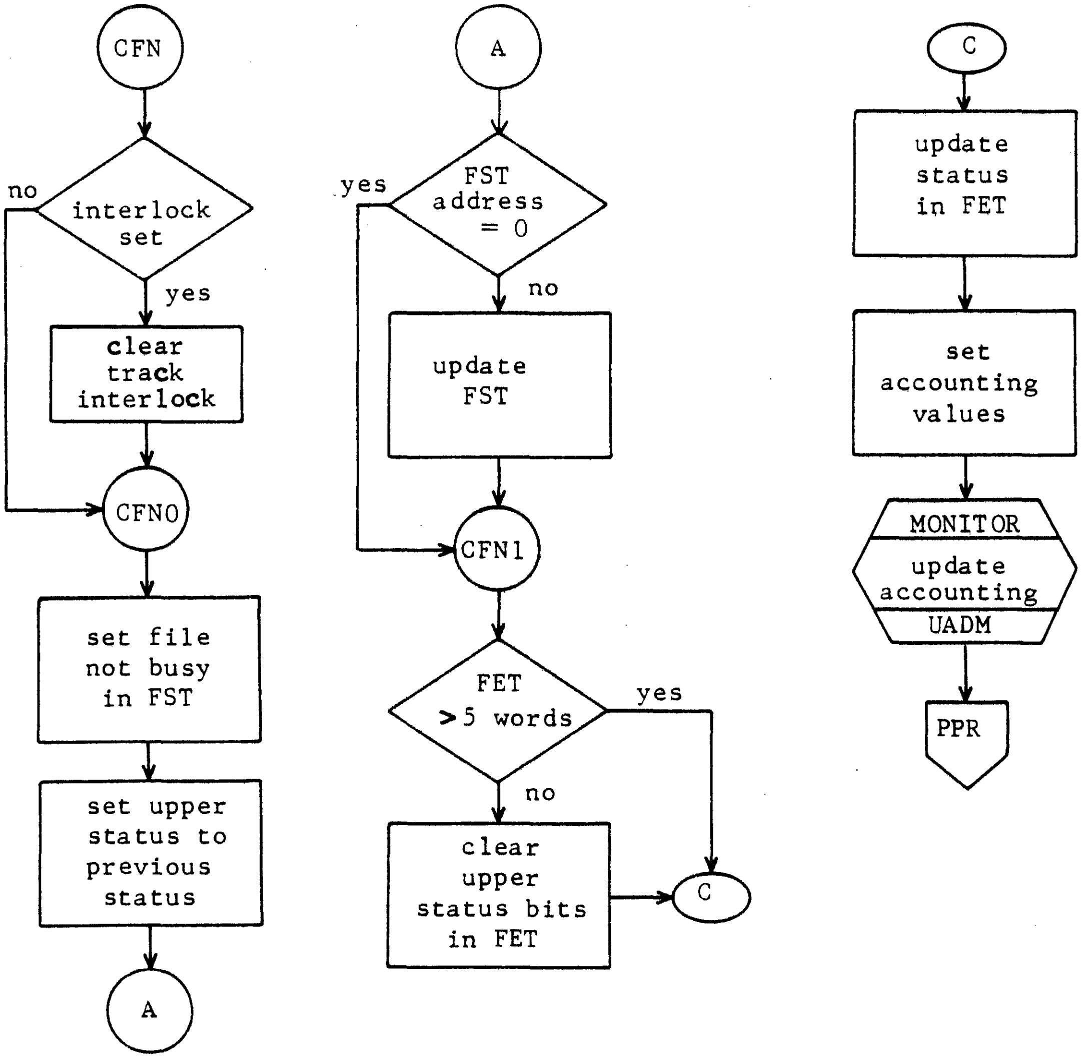


Figure 9-24. CFN - Complete Function

TERMINAL INPUT/OUTPUT ROUTINE TIO

Figure 9-25 is a flowchart of the terminal input/output (TIO) routine. This routine is contained overlay 2CH. TIO is only called from the PFN subroutine.

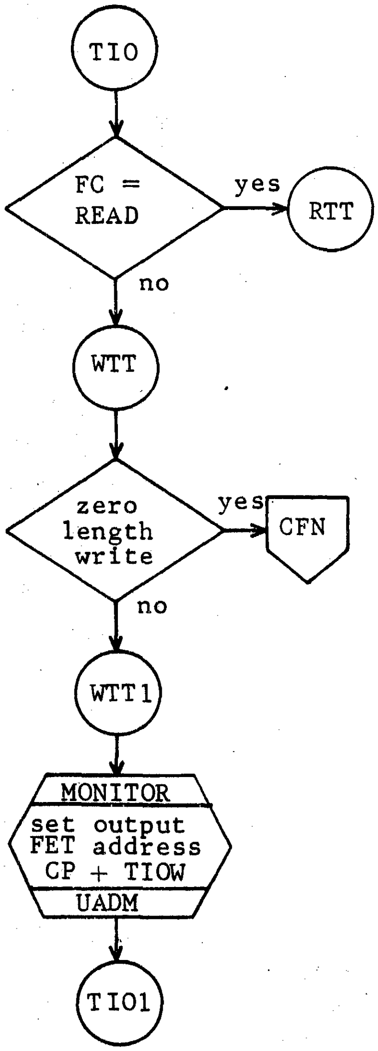
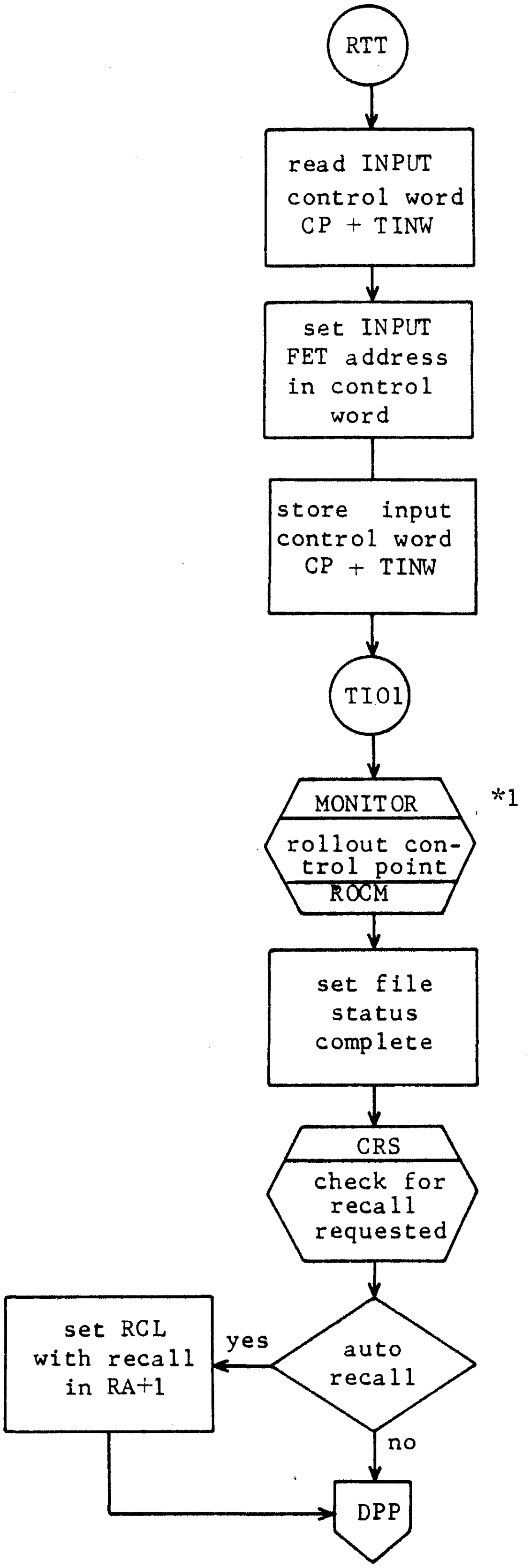


Figure 9-25. TIO-Terminal Input/Output



*1 1RO will write output to disk on terminal's rollout file.

Figure 9-25. TIO - Terminal Input/Output (Continued)

2CI SUBROUTINES

Figures 9-26 through 9-28 are flowcharts of the following subroutines in overlay 2CI.

- PMT - Process magnetic tape operations
- MER - Magnetic tape executive request
- UDT - Unit descriptor table read/write

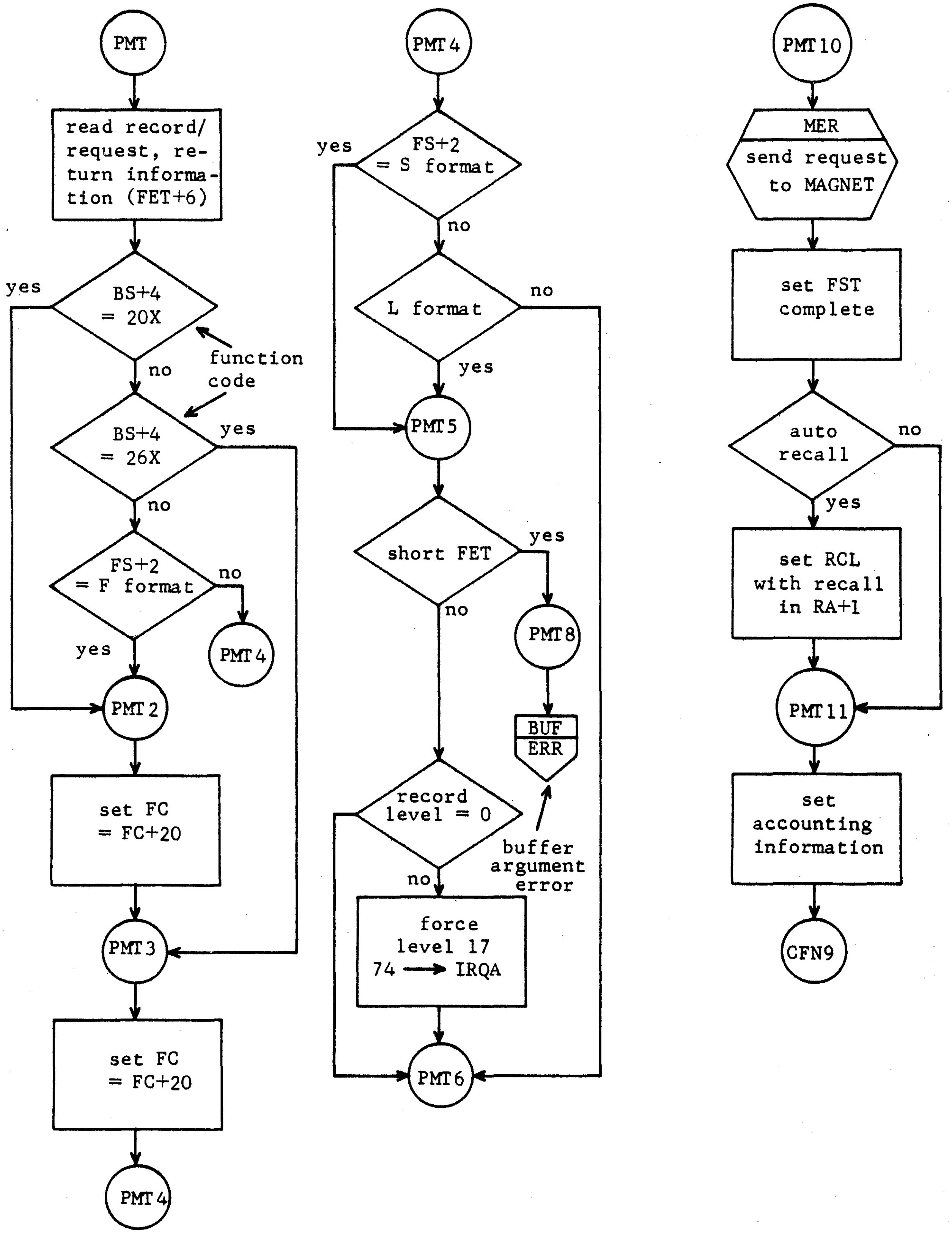


Figure 9-26. PMT - Magnetic Tape Operation

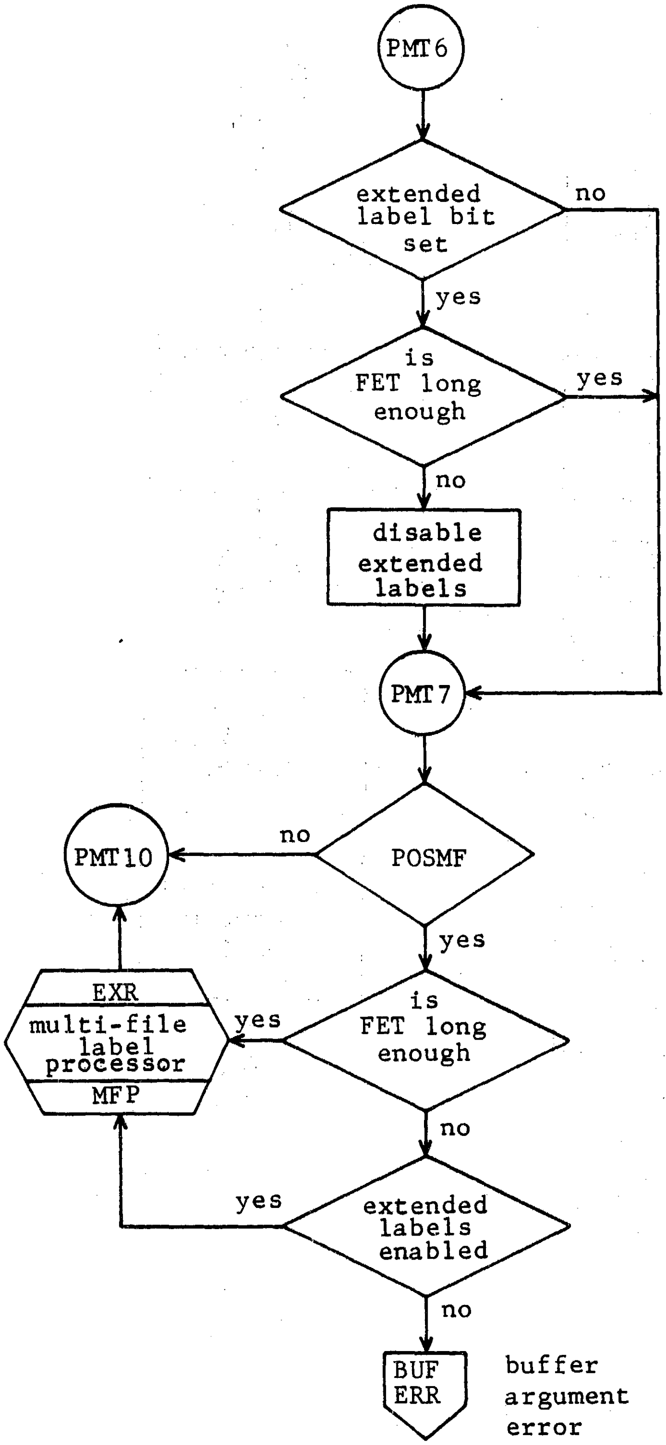


Figure 9-26. PMT - Magnetic Tape Operation (Continued)

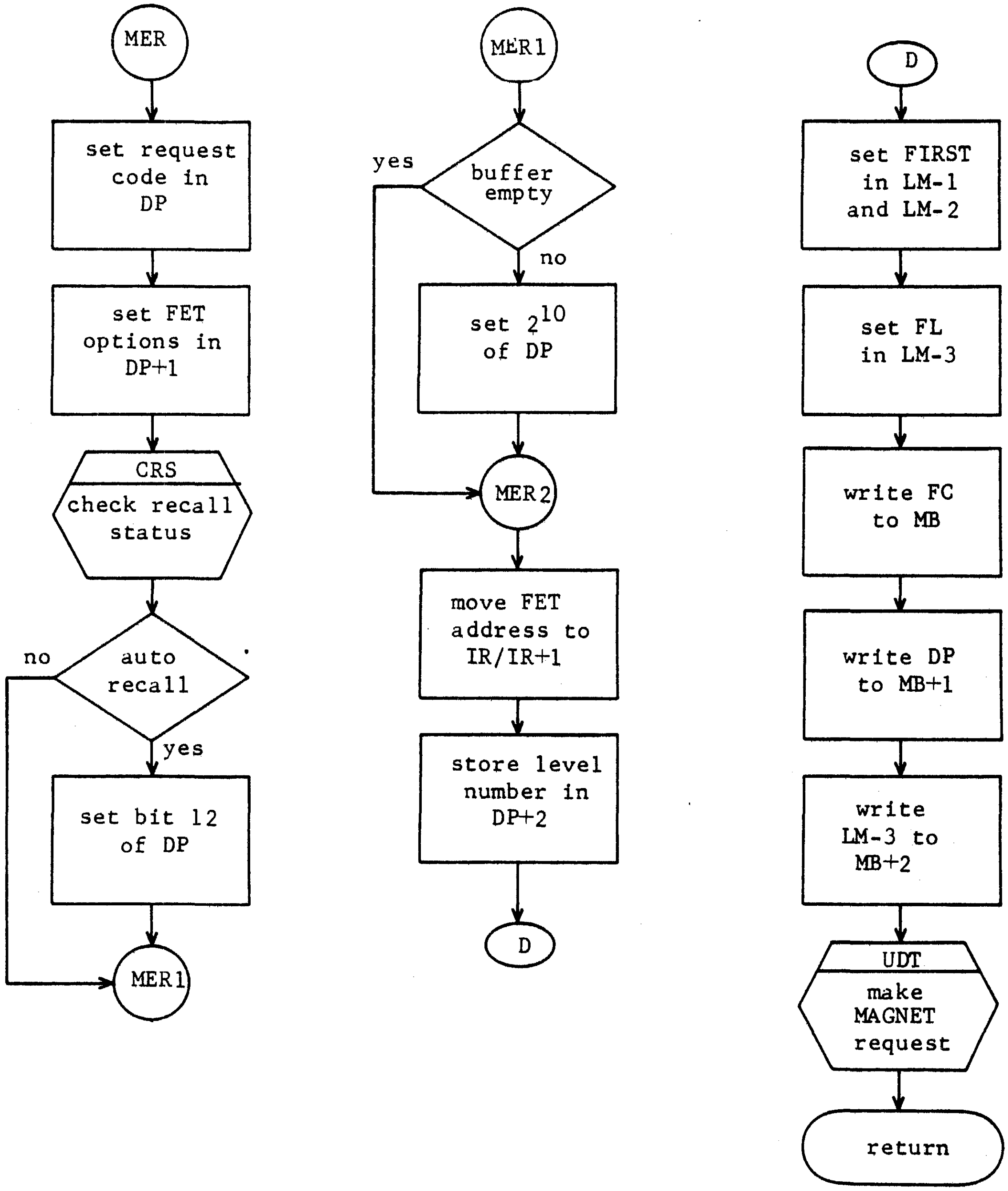


Figure 9-27. MER - Magnetic Tape Executive Request

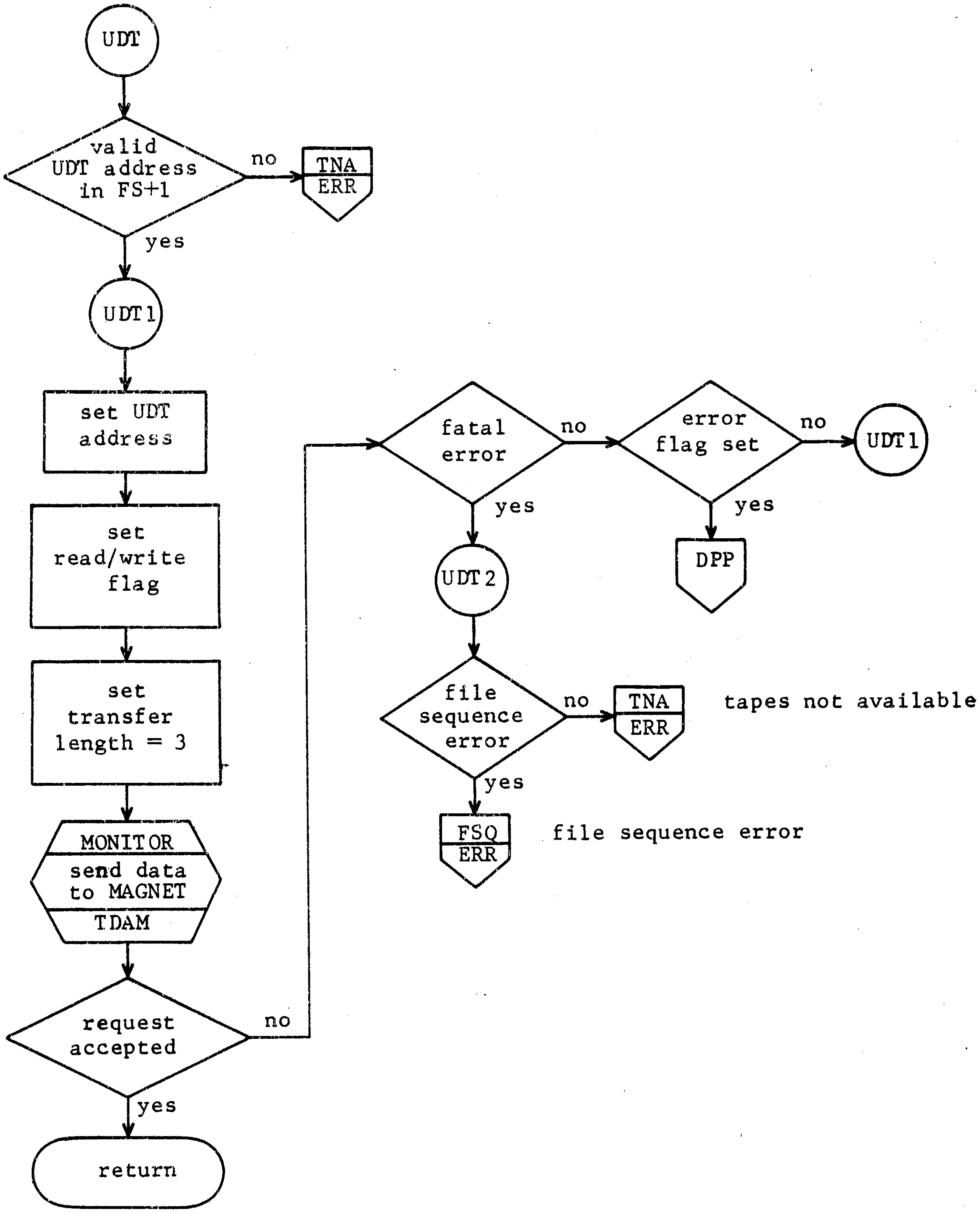
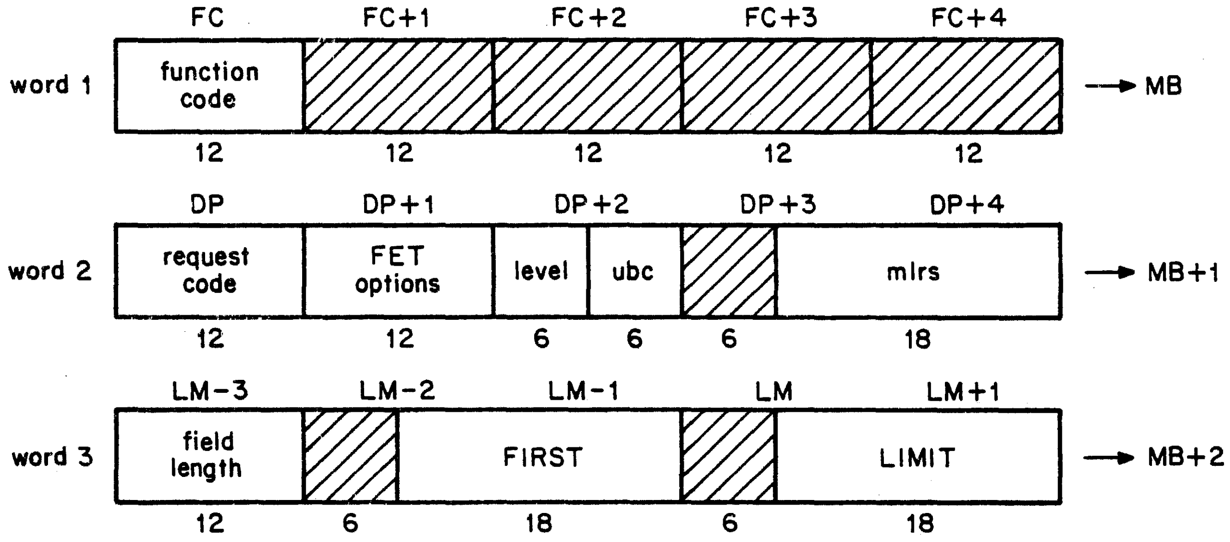


Figure 9-28 UDT - Unit Descriptor Table Read/Write

Flowcharts for the multifile label processor (2CJ) are not shown. Basically, PMT sets up a three-word parameter block and passes that information to MAGNET. The format of the three words is as follows:



ubc	Unused bit count (refer to FET description, NOS Reference Manual, Volume 2; publication number is in preface)
mlrs	Maximum logical record size

The request code is taken from the FET. The upper bit (bit 11) is set if autorecall was specified. Bit 10 is set if the buffer contains data. FET options are from byte 1 of FET+1 and indicate the error processing, user processing, and extended label processing bits.

COMMENT SHEET

MANUAL TITLE: CDC NOS Version 1 Internal Maintenance Specification, Volume 1

PUBLICATION NO.: 60454300 **REVISION:** B

NAME: ____________________

COMPANY: ____________________

STREET ADDRESS: ____________________

CITY: ____________ **STATE:** ____________ **ZIP CODE:** ____________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

CUT ALONG LINE

STAPLE STAPLE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION