CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER 70 COMPUTER SYSTEMS
MODELS 72, 73, 74
6000 COMPUTER SYSTEMS

SCOPE SYSTEM PROGRAMMER'S REFERENCE MANUAL
MODEL 72, 73, 74 VERSION 3.4
6000 VERSION 3.4

CONTROL DATA
CORPORATION

CONTROL DATA®
CYBER 70 COMPUTER SYSTEMS
MODELS 72, 73, 74
6000 COMPUTER SYSTEMS

SCOPE SYSTEM PROGRAMMER'S REFERENCE MANUAL
MODEL 72, 73, 74 VERSION 3.4
6000 VERSION 3.4

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

REVISION RECORD

REVISION	DESCRIPTION
A	Original printing.
(10-15-71)	

Publication No. 60306500

Additional copies of this manual may be obtained from the nearest Control Data Corporation sales office.

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Software Documentation
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

PREFACE

This manual describes the SCOPE 3.4 Operating System for the CONTROL DATA® CYBER 70/Models 72, 73 and 74 and 6000 Series computers. It is written for systems programmers who perform system evaluation or program modification.

Other documents of interest to system programmers using 3.4:

SCOPE 3.4 Reference Manual

SCOPE 3.4 Operator's Guide

Record Manager Reference Manual

LOADER Reference Manual

UPDATE Reference Manual

SCOPE 3.4 Installation Handbook

The SCOPE 3.4 Internal Maintenance Specifications are not published; they are available, however, on listable magnetic tape.

SCOPE 3.4 is an extension of SCOPE 3.3. It includes the following new features and changes:

User library creation and maintenance similar to EDITLIB creation of system libraries

Support of SCOPE format records on 9-track tape

New loader with associated restructuring of SCOPE system libraries and overlay/segment structures

Revised REQUEST control statement parameters

Addition of VSN card to support automatic tape assignment to requesting jobs

Extended error processing

Addition of sequential disk pack capabilities for highspeed access to a sequential file

Renaming of private disk packs to family disk packs

Additional permanent file functions: ALTER, SETP; additional parameters for other permanent file functions

Replacement of SNAP and TRACE debugging aids

New extended core storage system

Addition of Record Manager

Addition of Integrated Scheduler

Addition of File Organizer and Record Manager utilities

Reorganization of system Monitor and CIO

Common files are no longer supported

Numeric values given in this document, unless otherwise indicated, are assumed to be as follows:

Bit numbers	Decimal
Byte numbers	Decimal
Word addresses	Octal
Field/table lengths	Octal
Block/page sizes	Octal

CONTENTS

SCOPE 3.4 INTRODUCTION 1

The operating system for CONTROL DATA® CYBER 70/Models 72, 73 and 74, and the 6000 Series computers provides Supervisory Control of Program Execution (SCOPE). The SCOPE operating system controls the use of CONTROL DATA CYBER 70/Models 72, 73 and 74, and 6000 Series computer hardware. Therefore, SCOPE is in control of the computer. SCOPE accepts input in the form of jobs submitted by users and processes them as directed by control cards accompanying each job as well as by keyboard commands input by the operator.

Efficient processing of user's 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.

HARDWARE CHARACTERISTICS

SCOPE uses peripheral processor units (PP) for system and input/output tasks and a central processor unit (CPU) to execute user and system jobs. Central memory (CM) contains user programs; system software areas are located at the upper and lower ends of central memory. An extended core storage (ECS) unit may be used to contain SCOPE libraries and other items (such as file buffers for RMS and swap files) which may not be contained in CM or on other mass storage devices.

CENTRAL PROCESSOR

The CPU is designed to perform tasks of a computational nature; it has no input/output capability. It communicates with other system components through the central memory. Under SCOPE, 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.

PERIPHERAL PROCESSORS

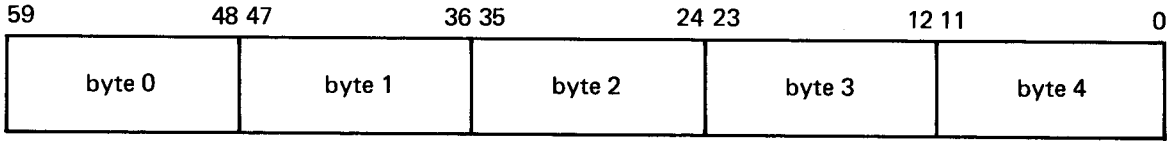
The peripheral processors, of which there may be up to 20 (identified as PP0, PP1, . . . PPn) are identical; they perform many tasks for requesting programs in central memory.

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

Each PP is assigned a block of eight words in the system area of central memory through which communications with the system are conducted. Each block contains an input register, an output register, and a message buffer. Peripheral processors are discussed in section 3.

CENTRAL MEMORY

Central memory words are 60 bits long; each has five 12-bit PP memory words, called bytes. Each 12-bit byte in a CM word is numbered 0 through 4, from left to right:



One or more user programs may be in some state of execution concurrently under SCOPE. These programs are stored in central memory in an assigned user area; a set of system components necessary for the operation of the system is also stored in central memory, forming the central memory resident (CMR) and the record block table (RBT) areas. Central memory is accessible by all PPU's and CPU's and forms the communications link between all processor units in the computer system.

Central memory resident (CMR) contains system communications areas, system tables, CPU resident routines, the library directory, and information about each job currently in execution. The CMR is discussed in section 2 of this manual; the RBT is discussed in section 4.

EXTENDED CORE STORAGE

FEATURES

Under SCOPE 3.4, Extended Core Storage (ECS) is divided into three areas: the System Area, the Dynamic Area, and the Direct Access area. They function as follows:

System Area	Contains system pointers and tables required by ECS software
Dynamic Area	Paged area; contains buffers, library programs, swap files and other files assigned to ECS
Direct Access Area	Assigned to user as result of job card request; used and managed by requestor

Such division of ECS allows the following features to be provided:

I/O buffering through ECS

Library residence in ECS

Job swapping to/from ECS by the Integrated Scheduler

Compatibility with BNL-ECS

ECS PAGING

I/O buffering through ECS with variable size buffers is made possible by the paging of ECS. Deadstart/ Installation parameters define the ECS page size buffer size. IP.EPAG sets the page size and has a default value of 8 PRU (1000 CM words); IP.EBUF determines the ECS buffer size and has a default value of 16 (decimal) pages.

ECS paging provides the following advantages:

More efficient usage of ECS through dynamic allocation/deallocation of space

Availability of ECS to more users by allowing the use of an "over-commitment" algorithm for ECS

Simplification of core management by eliminating garbage collection chores

SOFTWARE ELEMENTS

Two elements are basic to the SCOPE operating system: files and control points.

FILES

A file is an organized collection of data known to the system by a given name. Data is organized in one or more logical records and terminated by an end-of-file indicator. Under the SCOPE operating system, the jobs it processes and all intermediate and final results are contained in files or parts of files. Files are discussed in section 4.

CONTROL POINTS

The system can control execution of several jobs at one time. When placed into CM before execution, each job is assigned a value which is the control point number and the index to a control point. Jobs at control points are assigned to a processor for execution. Each control point has a control point area in the CMR which holds all information necessary to process the assigned job. The control point concept is discussed in section 2.

SCOPE ORGANIZATION

The SCOPE operating system consists of PP programs, CP programs, macro definitions and symbol definitions. The entire system is contained in a magnetic tape file produced by the library maintenance program UPDATE. 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 a deadstart file on tape.

A system monitor is in complete supervisory control of the hardware system. The system monitor is made up of PP overlay OV.MTR which operates in PP0, and CPMTR which is assembled as part of the central memory resident (CMR).

SYSTEM LOADING

To load the operating system into a CYBER 70 or 6000 series computer, the deadstart tape is mounted on a device and a small bootstrap loader program is set up on the hardware deadstart panel switches. When the deadstart button of the operator's console is activated, the bootstrap program is transferred to and executed in PP0. The bootstrap loader reads the PP0 save program (CEA) from the first record on the deadstart file, executes it, then reads the deadstart control program (CED) from the next record into PP0 and sets it into operation. CED determines the type of deadstart to be performed, then loads the required routines into all PPs involved in the deadstart process. The routines include a display routine in PP0 and I/O routines in PP1 and PP2. CMR is read from the deadstart file into CM, and a display shows all deadstart functions and the options that may be selected.

The functions include:

Type of deadstart to be performed

Level of system to be used

Level of processing to be performed on rotating mass storage (RMS) device labels

Status of permanent files to be used

Restriction of permanent files to disk pack devices (854 and 841 only)

Changes to be made to equipment configuration of the system

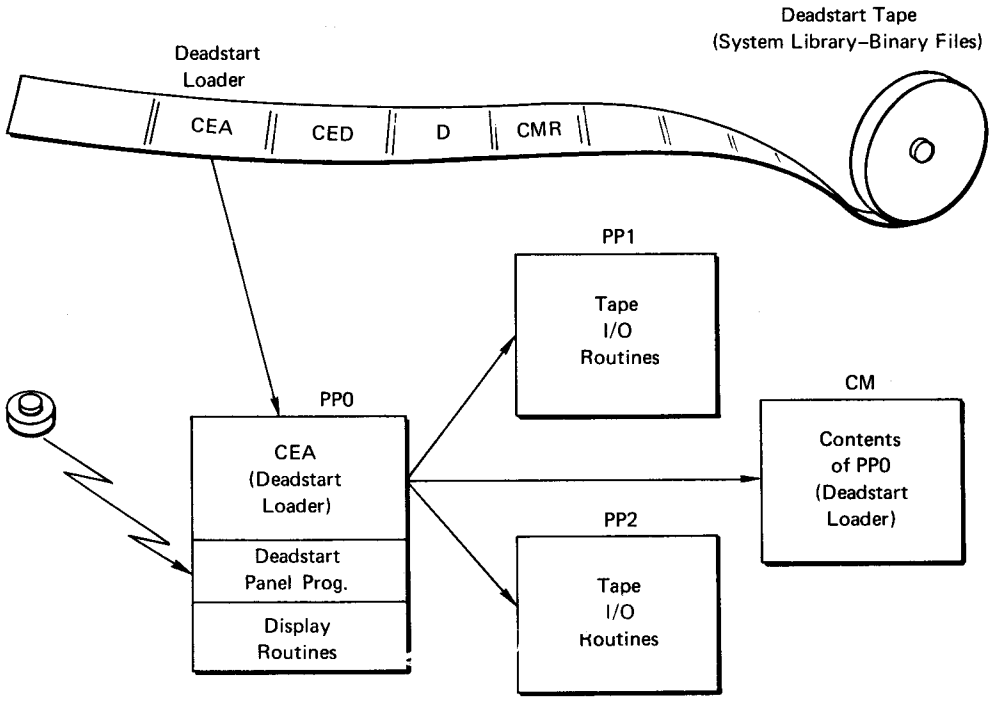
Level of initialization of ECS

Pre-allocation of RMS devices for customer engineering diagnostic programs

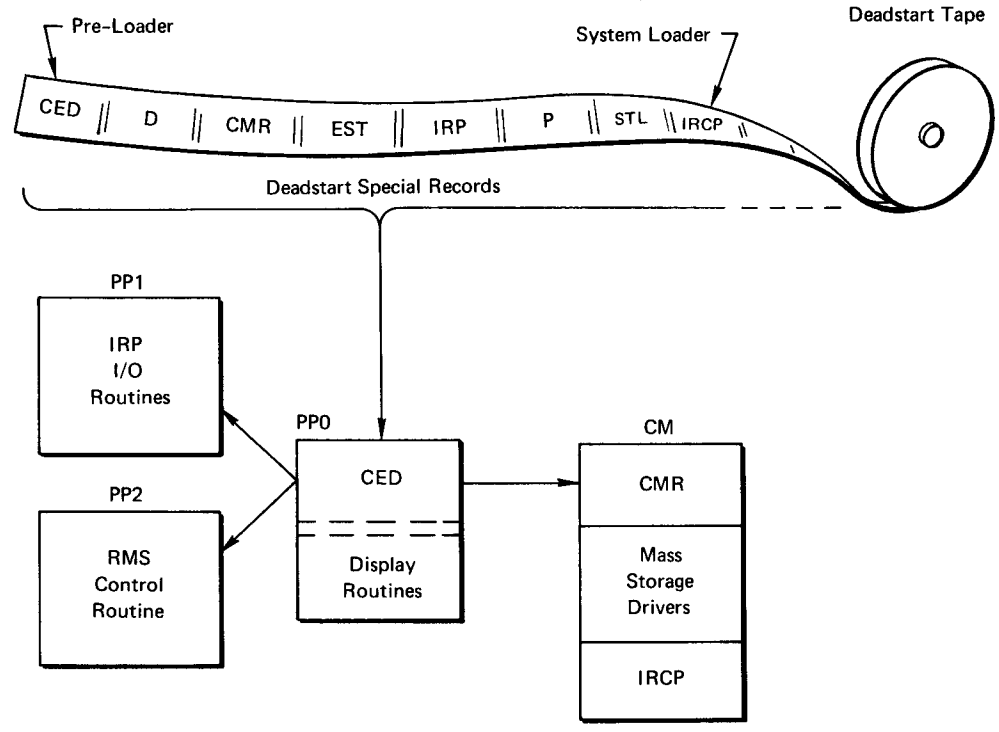
The operator may select specific options or take the default option for each function.

When processing of deadstart options is completed, control is passed to MTR and DSD. The system deadstart tape will be rewound to its load point, and it will not be referenced again during operation unless another deadstart is necessary. The tape may be removed and the tape unit cleared for use in other operations. At this point the system is ready to process jobs (see section 7).

DEADSTART – PHASE 1



DEADSTART – SYSTEM ACTIVATION



Upon completion of system loading, the computer contains the following:

1. Initial system libraries, stored on one or more mass storage devices. Programs can be loaded from any system library into PPs or CM as needed.

2. The central memory resident (CMR), loaded into the low end of central memory. A set of tables in CMR contain information about the system. Some of the tables are used by the system PP programs to communicate with each other. A record block table (RBT) is built in the upper end of CM.

3. Also, some programs in the system are stored in central memory in an area immediately above the CMR. Such programs can be loaded into PPs or into other CM areas much faster than they can be loaded from the system storage device. Storage space in CM is costly and space used to store library programs cannot be used to run user programs; therefore, only the most necessary and frequently used library programs are stored there. For the same reason, CMR is kept as small as possible.

4. The system monitor program (PPMTR in PP0 and CPMTR in CMR) remains in overall control of the system. It controls allocation of system physical resources (CM, ECS, channels, equipments, PPs and CPUs). It handles all communications between user programs and the system and coordinates activities of the other PPUs.

5. PP1 contains the operator console display driver program, DSD. DSD provides a communication path between system and operator. Current system status is displayed by DSD on the two screens in the operator console. The operator can control system operation by typing commands on the console keyboard.

6. Each remaining PP contains pointers to a PP communications area, an area in CMR used for communications between each PP and MTR, and a PP resident program loaded into each PP. The resident is responsible for reading the input register and loading PP overlay programs into its PP as assigned, and for providing communication between the overlays and MTR.

SCOPE SYSTEM TAPE

The released SCOPE 3.4 system (deadstart) tape consists of 21+n records, each followed by an EOR, where n is the number of programs and overlays in the tape:

NAME	Description
CEA	PP0 save program
CED	Deadstart control program
TDR	7-track/9-track driver
D	Deadstart dump control program
DMT	Dump magnetic tape driver
CMR	Central memory resident (n copies)
COM	Deadstart option matrix generator
EST	Deadstart equipment reconfiguration program
IRP	Deadstart RMS driver control program
5CP	Deadstart 6603-I driver
5CQ	Deadstart 6638 driver
5CT	Deadstart 6603-II driver
5CR	Deadstart 865 driver
5CS	Deadstart 854 driver
5CU	Deadstart 814 driver
5CV	Deadstart 821 driver
5CW	Deadstart 841 driver
P	Pre-address 6603-II program
STL	Deadstart system initiation program (PP resident)
IRCP	Deadstart main program
MTR	System monitor program

DSD	Display control program
Directory	Library name table
	PP name table
	PP programs
System Libraries	Entry point table
	External reference list
	External reference table
	Program number table
	Program name table
	CP/PP routines

Any installation may expand the above records in two ways:

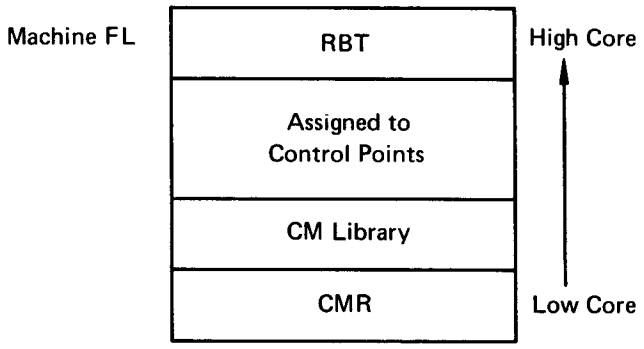
1. By installing CE Diagnostics. The deadstart diagnostic sequencer, CES, will be placed after CEA, followed by diagnostic routines selected for the site (CU1, ALS, FST, etc.), followed by routine CED.

2. By placing up to seven additional CMR records on the system tape for different equipment configuration.

CENTRAL PROCESSOR AND SCOPE 2

CM ORGANIZATION

The allocation of central memory is illustrated below:



Low core is allocated to the central memory resident portion of SCOPE, executable system programs, and INTERCOM buffers. The length of the INTERCOM buffers area varies dynamically when INTERCOM is running. High core is allocated to the record block table (RBT); its length varies dynamically with the load of the system. The remaining area can be assigned to control points.

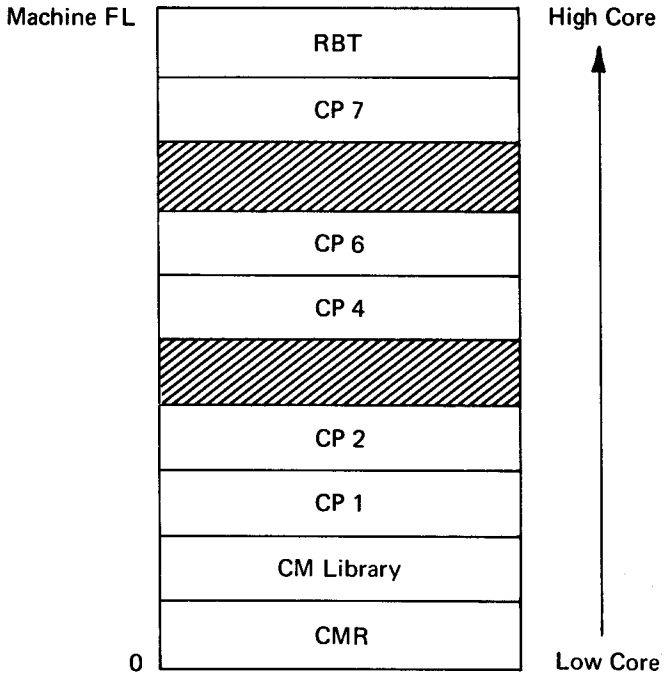
CONTROL POINT CONCEPT

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 bookeeping. The control point number and the control point area, however, are physical quantities that do appear in the system.

Under SCOPE 3.4, any number of control points up to 15 are possible. In the released system, the default value of N.CP is 15 decimal. 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, equipments and processors may be assigned to the control point for use by the job.

Storage assigned to a single control point is contiguous; storage for all control points is not necessarily contiguous. The core 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 always higher than that for control point 2, and so on.

In the following figure, no storage is assigned to control points 3 and 5; unassigned storage appears between assigned storage.



In addition to the n control points used for running jobs, two pseudo control points, numbered zero and n + 1, are used by the system.

Control point zero is used to identify system resources not allocated to a job at a control point; they are unallocated or allocated to the system. If an equipment is assigned to a control point, that number is entered into the system table entry for that equipment.

If not assigned to a job, the equipment is assigned to control point zero and is available to be assigned to a job. All active system files are attached to control point zero. They include the system file, any job files that have been read in and are waiting for scheduling, and all output files waiting to be processed by JANUS. Control point n + 1 is used by monitor (MTR) when it runs a central processor (CP) system program. For example, if MTR needs to move the storage assigned to a control point, it asks CPMTR to initiate the CP storage move program in central memory resident. Since the CP storage move program is not associated with any specific job running at a control point, it is assigned to control point n + 1. There is no control point area in CMR for control point n + 1.

JOB DESCRIPTION NUMBER

During the course of execution, a job might not remain continuously at the same control point. It is possible for the job to be swapped out while it is only partially executed. When a job is swapped out it is not associated with a control point. When a job is swapped back in it is probably associated with a control point other than the control point during its original assignment.

During the time a job is swapped out, the only table in CMR that contains information about the job is the Job Descriptor Table (JDT). When a job is initialized at a control point it is also assigned to an entry in the JDT. The job descriptor number is constant and is used to identify the job during its entire execution.

In order to clarify the difference between job descriptor number and control point number, JDT numbers start at n + 1 where n is the number of control points.

STORAGE MOVES

Since jobs come and go as they finish processing and new jobs begin, or as jobs are swapped 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. MTR keeps a tally of unassigned CM in a CMR word T.UAS. Storage associated with each control point is of two types, either of which may have a zero value.

Allocated storage is defined by the reference address (RA) and field length (FL) of the control point.

Unallocated storage (UAS) lies between the allocated portions of two consecutive control points. This area is associated with the lower of the two control points, but it may be transferred to neighboring control points by moving any intervening allocated storage.

A request for a reduced field length merely transfers storage to UAS (no storage moved). A request for an increased field length, when the total already associated with the control point is adequate, will result in a transfer of unallocated storage to allocated; no storage move will take place.

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

In the diagram shown under Control Point Concept, 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 will be moved downward to provide the storage. Added storage always extends the field length upward.

STORAGE MOVE EXAMPLE:

Control point 5 requests an FL of 300 (all values are increments of 100 octal).

	Before			After		
Control Point	RA	FL	Unallocated Storage (UAS)	RA	FL	UAS
0	0	142	0	0	142	0
1	142	33	0	142	33	0
2	175	31	0	175	31	0
3	226	0	500	226	0	0
4	726	20	130	226	20	0
5	1076	100	0	246	300	430
6	1176	150	0	1176	150	0
7	1346	0	430	1346	0	430

If MTR takes the UAS from control point 7, the 150 units of central memory at control point 6 would have to be moved; however, taking UAS from 3 and 4 requires moving 120 units of central memory at control point 4 and 5. (20 units are moved from 4 to 3 and 100 units are moved from 5 and added on behind the 20 units moved to 3.)

CP - SYSTEM COMMUNICATION

A running CP program must communicate with the system as illustrated in the following examples:

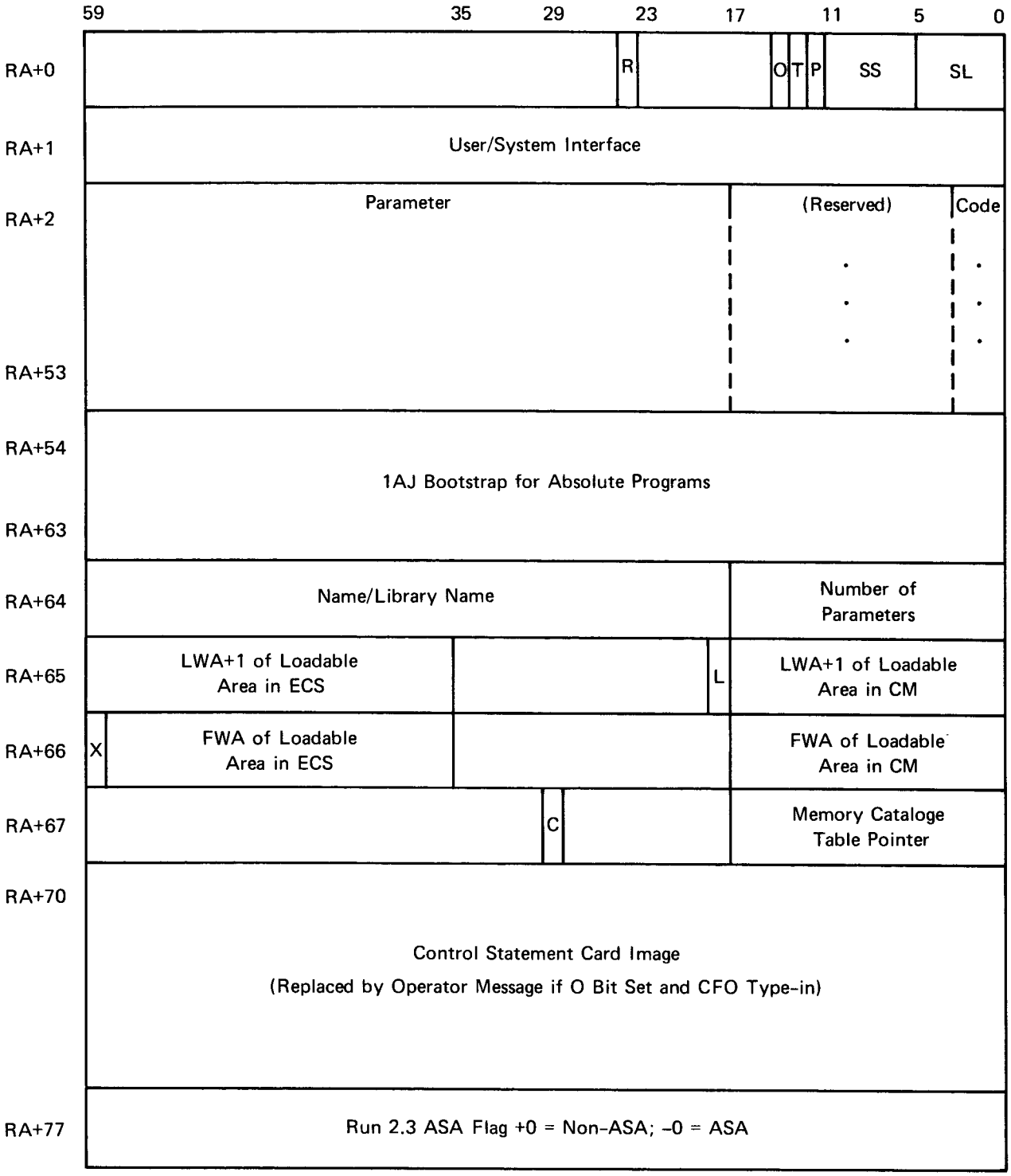
1. When a CP program is loaded and executed as a result of a control card call, the system must place any parameters specified on the control card in an area where they can be read by the CP program.

2. No CP instructions allow a CP program to perform input/output; therefore a CP must send a request to the system, to load a PP program to execute the input/output.

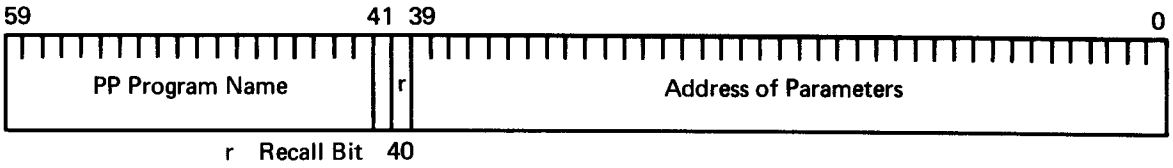
3. When a CP program terminates, it must advise the system that it may process the next control card.

Since a CP program cannot access memory locations outside its field length, any area reserved for communication between a CP program and the system must be within the field length of the job. The first 100 (octal) locations of each job's field length are reserved for this purpose. The first program loaded into a user field length is always loaded at location RA + 100 (for the user, this is location 100).

RA COMMUNICATION AREA



The first word of a user field length (location RA+0) is reserved for use of hardware and software flags in event of error. Other locations in the first hundred words of a user field length store information needed for execution of system program. Monitor regularly scans location RA+1 which is presumed to contain a request from the central processor for monitor to summon a peripheral program. The form of the request is:



Loader information is placed by the first of several loader routines in words RA+64 through RA+67. This information is used and modified as additional loader routines complete specific tasks.

When parameters are encountered on a control card, they are placed in locations RA+2 through RA+63 by 1AJ, which stores the total number of parameters in location RA+64. When the routine or file indicated on the control card executes, it finds in these locations the information needed to direct execution.

CP - PP COMMUNICATION

If a user's program places a call for a PP program in RA+1, CPMTR will pick up the RA+1 call, insert the control point number of the caller into bits 36 through 39 of the word, and clear bit 41. If the central exchange jump (CEJ) installation is available, the user's program should use it immediately after placing a call in RA+1. This will cause CPMTR to begin execution immediately. If CPMTR determines that the RA+1 call should be assigned to a PP it will pass the call on to MTR.

When a PP is available, MTR will write the word into its PP input register, in CMR. Figure 3-2 shows the format of a PP input register for a transient program called from a CP program. The name, the auto-recall bit, and any parameters in bits zero through 35 appear in the input register exactly as they did in RA+1. Parameters are passed from a CP program to a PP program through this parameter field.

For example, if the PP program CIO is called, CIO will find 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 users program does not use the central exchange jump, or the instruction is not available. When a RA+1 call is found MTR initiates CPMTR. Less CPU time is used by letting CPMTR process the call, than if MTR did it directly.

Bit 59 of RA+66 is used to communicate to the user program if the central exchange jump is available. If the hardware for this instruction is available the bit is set on.

PROGRAM RECALL

The recall program status is provided in SCOPE to enable efficient use of the central processor and to capitalize on the multiprogramming capability of SCOPE. Often, a CP 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 CP program were placed in a loop to await I/O completion, a CP program can ask SCOPE to put the control point into recall status until a later time; and the CPU may be assigned to execute a program at some other control point. The job itself may be rolled out or swapped out, as necessary.

Recall may be automatic or periodic. Auto-recall should be used when a program requests I/O or other system action and cannot proceed until the request is completed. SCOPE will 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 CP 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 delay count of the control point in periodic recall will be reduced by the Advance Control Point (ACP) routine of PP monitor. When the count is zero, the PP is restarted and the CPU is again assigned to execute the program at the control point. At this time, the CP program can check completion bit in the FET to see if the I/O is finished. If so, the CP program may proceed with computations. If I/O is not complete, the CP program will put itself back into recall.

AUTOMATIC RECALL

If a CP program makes a request in RA+1 and bit 40 of RA+1 is set to one, 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 MTR after the completion bit in the FET has been set. MTR, not the user, checks the completion bit in the FET.

Recall and auto-recall 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 one, the control point will be put into auto-recall. If bit 40 is set, bits zero through 17 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. When the completion bit is set, MTR will restart the program.

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

Some PP programs (DMP and MSG) set the completion bit only when they are called with auto-recall. Periodic recall cannot be used for these programs.

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

Normally, CP programs use auto-recall for convenience, but only one request involving auto-recall 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. He will issue all the requests without recall (using a separate FET for each request); then go into periodic recall. Each time the CP 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.

Periodic recall may be used also when a CP program can initiate an I/O request and then perform some computation. In some cases, the I/O would be completed before the computation; in others, the computation would be done first. The user would go into recall only when computation was done, and then only if the I/O was 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. Some of the I/O drivers coordinate with MTR so that a program in periodic recall is restarted after one or two PRU's have been processed.

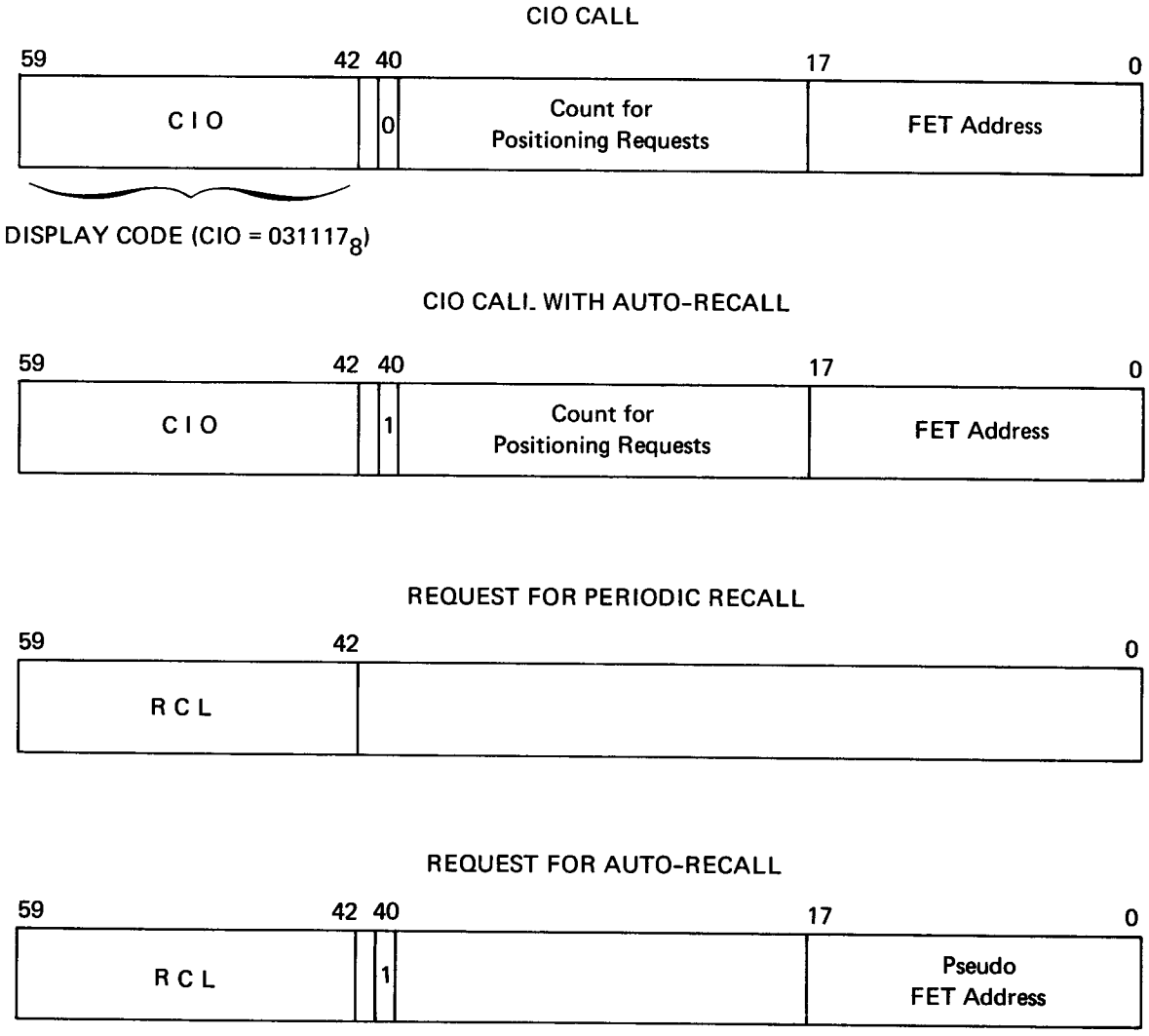


Figure 2-1. Call Formats

CENTRAL MEMORY RESIDENT

The low end of core storage is reserved for the central memory resident portion of SCOPE and the system library portions which reside in central memory. CMR contains pointers, tables and programs. Its length is dependent upon several factors, including the number of peripheral processors and the number of control points, which determine the number of tables in CMR and the length of certain tables. Some CMR tables are optional and may appear only by installation parameter. Figure 2-2 illustrates a typical CMR.

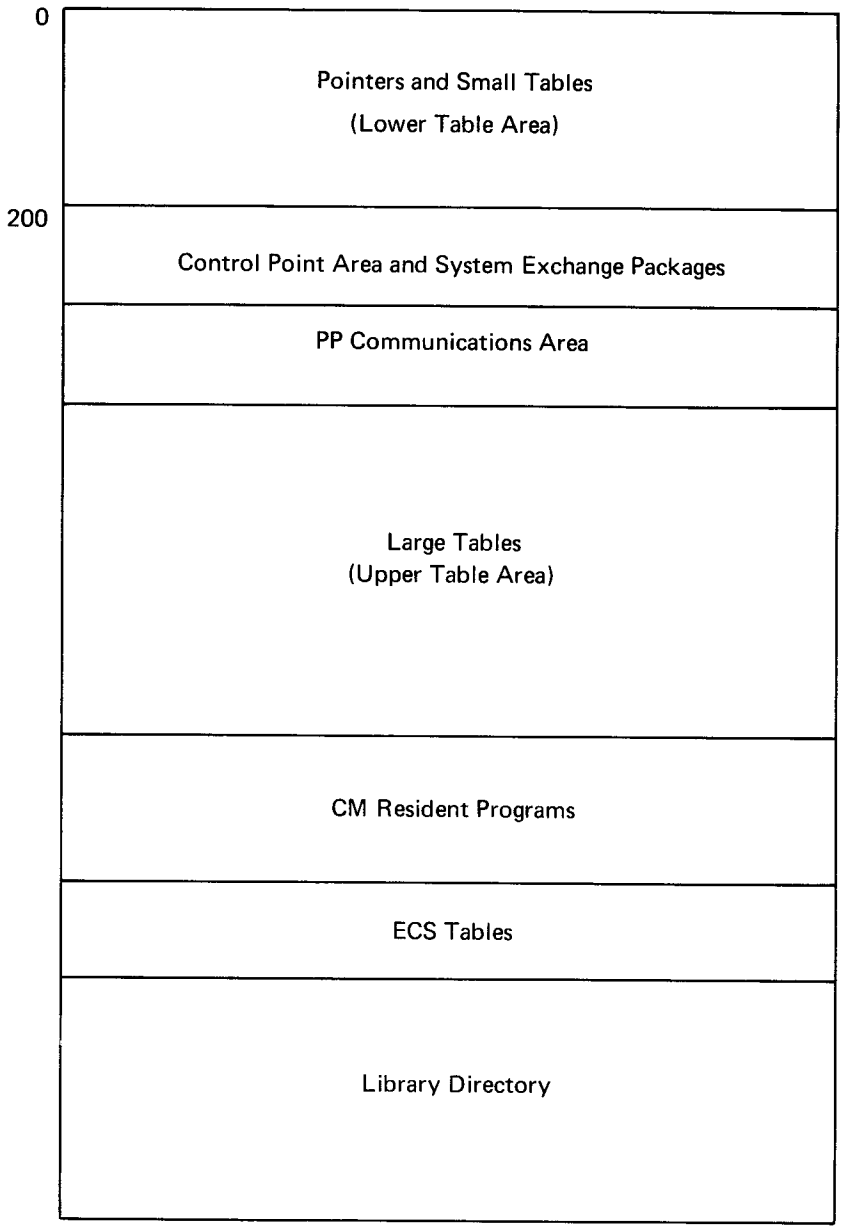


Figure 2-2. Typical CMR Assignments

SUMMARY OF CENTRAL MEMORY RESIDENT AREAS

Lower Table Area Contains pointers to larger tables in the upper table area of CMR, along with various flags, constants, and installation option parameters. It includes accounting information, calendar and Julian dates, the system display label and other small tables. The lower table area occupies the first 200 words of central memory.

Control Point Area Contains a 200-word area for each control point in the system. Each area contains the job name, exchange package, and other information related to the job running at that control point. The system exchange package is also contained in the same area.

PP Communications Area Contains eight words for each peripheral processor in the system, through which they communicate with the system monitor and with each other. Each area contains the PP input and output registers and a 6-word message buffer.

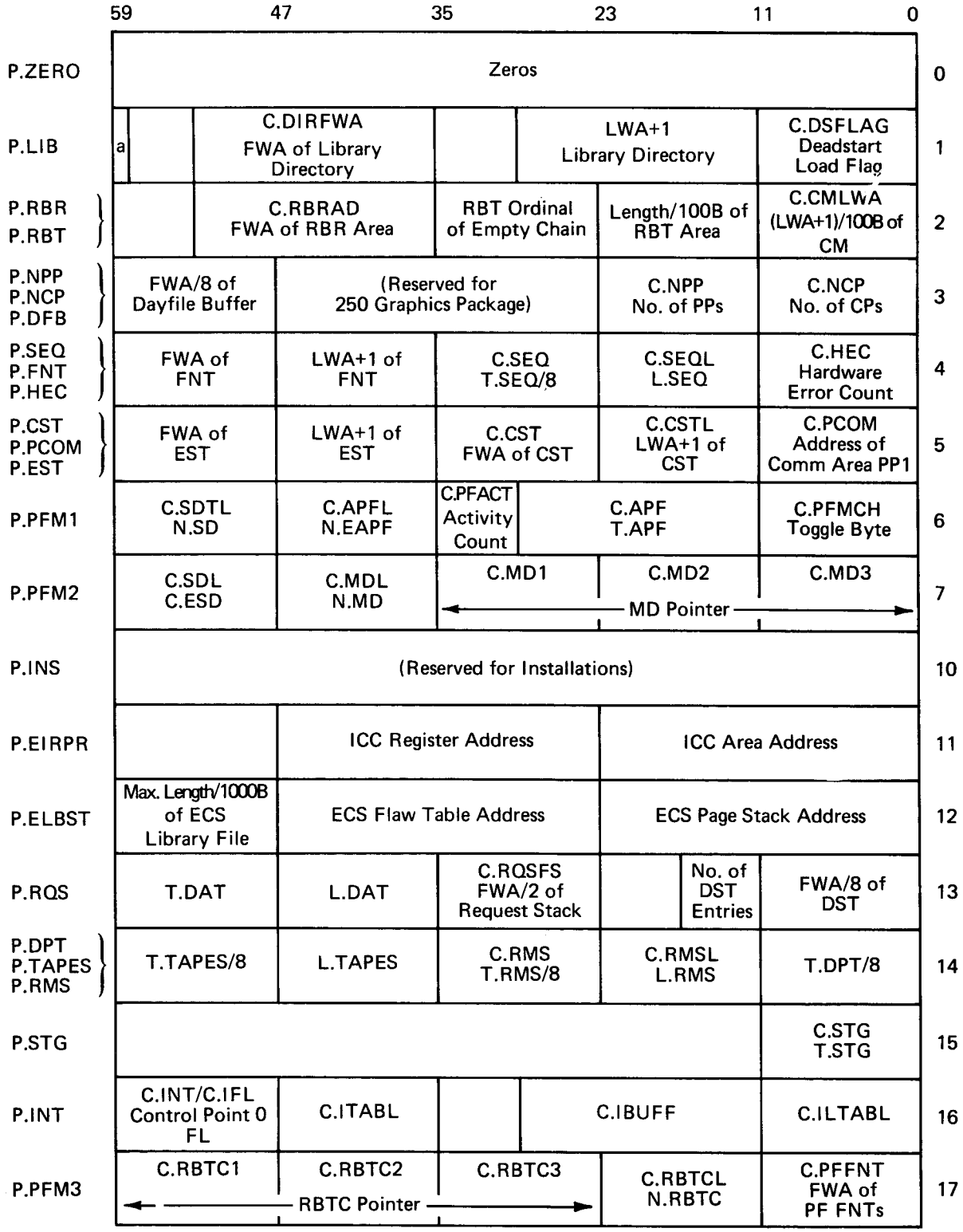
Upper Table Area Contains major tables pertinent to system and job operation.

ECS Tables Area Contains buffers for transferring PP overlays and RMS files.

CM Resident Program Area Contains eight resident programs: CP Monitor; CM Storage Move program; ECS Read/Write and Storage Move programs; memory manager; Integrated Scheduler; CP Circular I/O controller; and Stack Processor Manager.

Library Directory Contains tables related to the system libraries, including library name table, PP program name table and the CM resident library programs.

CMR POINTER AREA



CMR POINTER AREA DETAILS

These descriptions begin with the location number and the symbol. Contents of the words may change in released version of system.

0 P.ZERO

Contains a full word of binary zeros; a PP may clear five bytes at a time by reading this location. Also used as a program stop by CMR and idle program. Changing the contents of this word will usually destroy system operation.

1 P.LIB

Right justified in bytes 0 and 1 is the first word address of the library directory; bytes 2 and 3 contain the right-justified last word address plus one of the library directory. Byte 4 is a deadstart load flag. A library change flag appears in bit 59.

2 P.RBT/P.RBR

Bytes 0 and 1 contain the right-justified first word address of the record block reservation area. Byte 2 contains the record block table word-pair ordinal of the first member of the RBT empty chain. Byte 3 contains the current length of the RBT area in 100-word blocks. Byte 4 contains the current size of central memory in 100-word blocks.

3 P.DFB/P.NPP/P.NCP

Byte 0 contains the FWA/10 of the dayfile buffer area; bytes 1 and 2 are reserved for the 250 graphics package. Bytes 3 and 4 contain the number of PP's and control points in the system, respectively.

4 P.FNT/P.SEQ/P.HEC

Bytes 0 and 1 contain the FWA and LWA+1 addresses of the file name table. Bytes 2 and 3 contain the FWA and length of the diagnostic sequencer table. Byte 4 contains the hardware error count.

5 P.EST/P.CST/P.PCOM

Bytes 0 and 1 contain the FWA and LWA+1 addresses of the equipment status table. Bytes 2 and 3 contain the FWA and LWA+1 addresses of the channel status table. Byte 4 contains the FWA of the communications area for PP1.

6 P.PFM1

The number of permanent file sub-directories, number of empty attached permanent file table entries, permanent file activity count, FWA of attached permanent file table, and the permanent file toggle byte are in bytes 0 to 4, respectively.

Bits in the toggle byte are:

0	RBTC wrap around
1	Attached permanent file interlock
2	Permanent file directory interlock
3	RBTC interlock
4	Utility interlock
5	Master directory interlock
6	TRANSPF lockout.

7 P.PFM2

The number of entries per sub-directory is in byte 0. The length of and pointers to master directory entries are in bytes 1 to 4.

10 P.INS

Reserved for installation use.

11 P.EIRPR

Contains the ICC register address and ICC area address.

12 P.ELBST

Contains the ECS flaw table and page stack processor address. Byte 0 contains the maximum size of the ECS library file in 1000-word pages.

13 P.RQS

Bytes 0 and 1 contain the FWA and length, respectively, of the device activity table. Byte 2 contains FWA/2 of the request stack. Right justified in byte 3 is the current number of device status entries. FWA of the device status table is in byte 4.

14 P.TAPES/P.RMS/P.DPT

Bytes 0 and 1 contain the FWA/10 and length of the tape configuration table. Bytes 2 and 3 contain the FWA/10 and length of the RMS preallocation table. Byte 4 contains the FWA/10 of the device pool table.

15 P.STG

The FWA of the tape staging table is in byte 4. The rest of the word is currently unused.

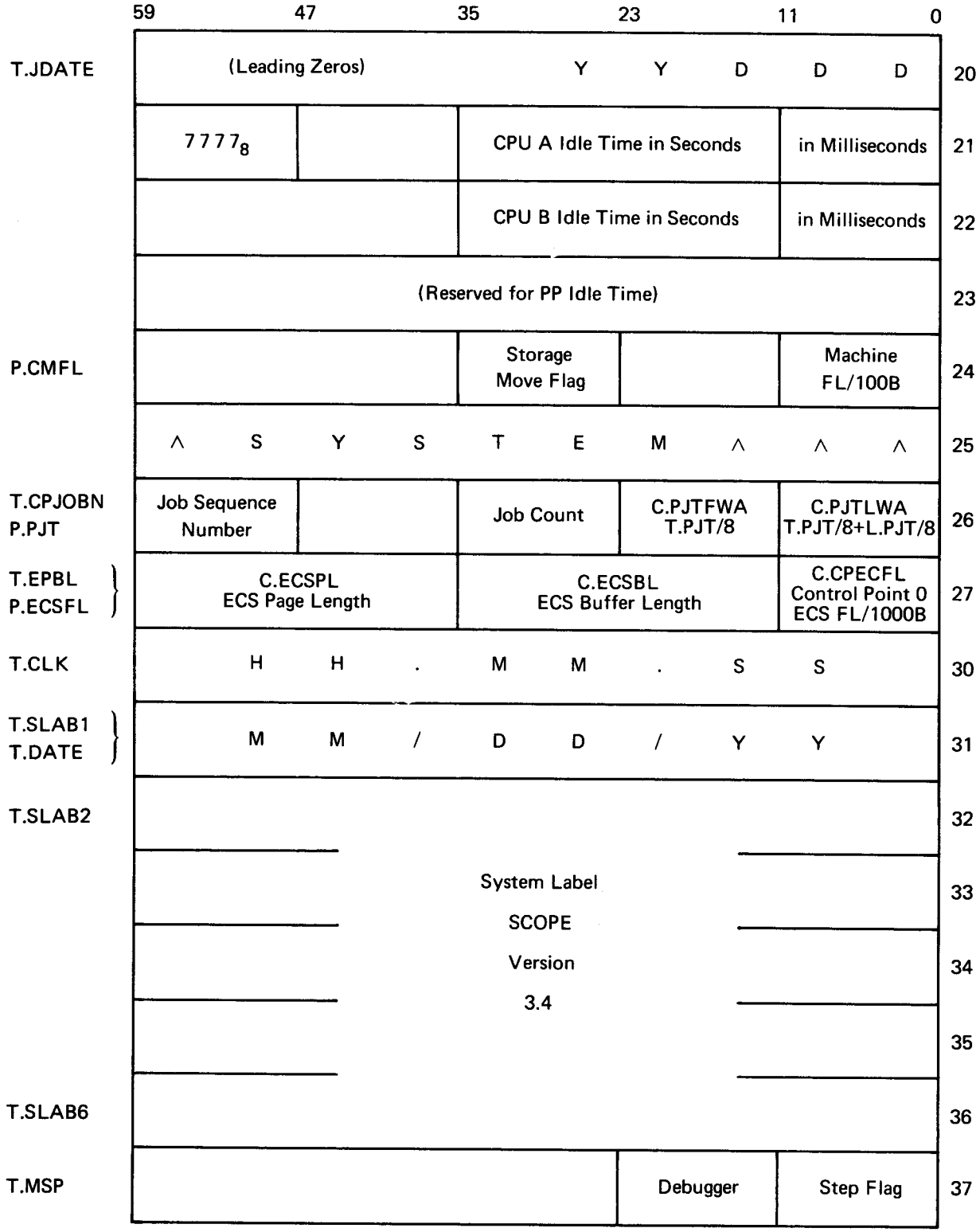
16 P.INT

Byte 0 contains the control point zero field length value; byte 1 points to the INTERCOM table; bytes 2, 3 and 4 point to the INTERCOM reserved core and buffer.

17 P.PFM3

Bytes 0, 1 and 2 contain pointers to the RBTC. Byte 3 contains the number of RBTC entries; byte 4 points to the FWA of the FNT's for the permanent file system.

CMR POINTER AREA



20 T.JDATE

The current Julian day number is stored here in the form yyddd with leading zeros.

21 (W.CPATIM)

Byte 0 contains all ones; bytes 2, 3 and 4 contain CPU-A accumulated idle time at control point zero.

22 (W.CPBTIM)

Bytes 2, 3, and 4 contain CPU-B accumulated idle time at control point zero.

23 (W.PPTIME)

Contains accumulated PP time for control point zero.

24 P.CMFL

Contains in byte 4 the current machine core size in 100-word blocks and in byte 2 the control point zero storage move flag.

25 (W.CPJNAM)

Contains control point zero job name in the form SYSTEM

26 T.CPJOBN/P.PJT

Bytes 0 and 2 contain the job sequence number and job count, respectively. FWA and LWA of the parameter storage area for delayed PP job is in bytes 3 and 4.

27 P.EPBL/P.ECSFL

Right-justified in bytes 0 and 1 is the ECS page length as set by IP.EPAG; right-justified in bytes 2 and 3 is the ECS buffer length set by IP.EBUF. Byte 4 contains the current number of ECS pages assigned to control point zero.

CMR LOWER TABLE AREA DETAILS

Each description begins with the location and symbol.

30 T.CLK

Current display clock time in the format:

hh.mm.ss.

31 T.DATE/T.SLAB1

Current calendar date is in the format:

mm/dd/yy

This is the first of a 6-word system display label.

32 (IP.SYSL1) T.SLAB2
33 T.SLAB3

These two words provide storage for up to 20 characters for the system display label first line, as given by installation parameter.

34 (IP.VER) T.SLAB4

System version identification.

35 (IP.SYSE) T.SLAB5

System edition date.

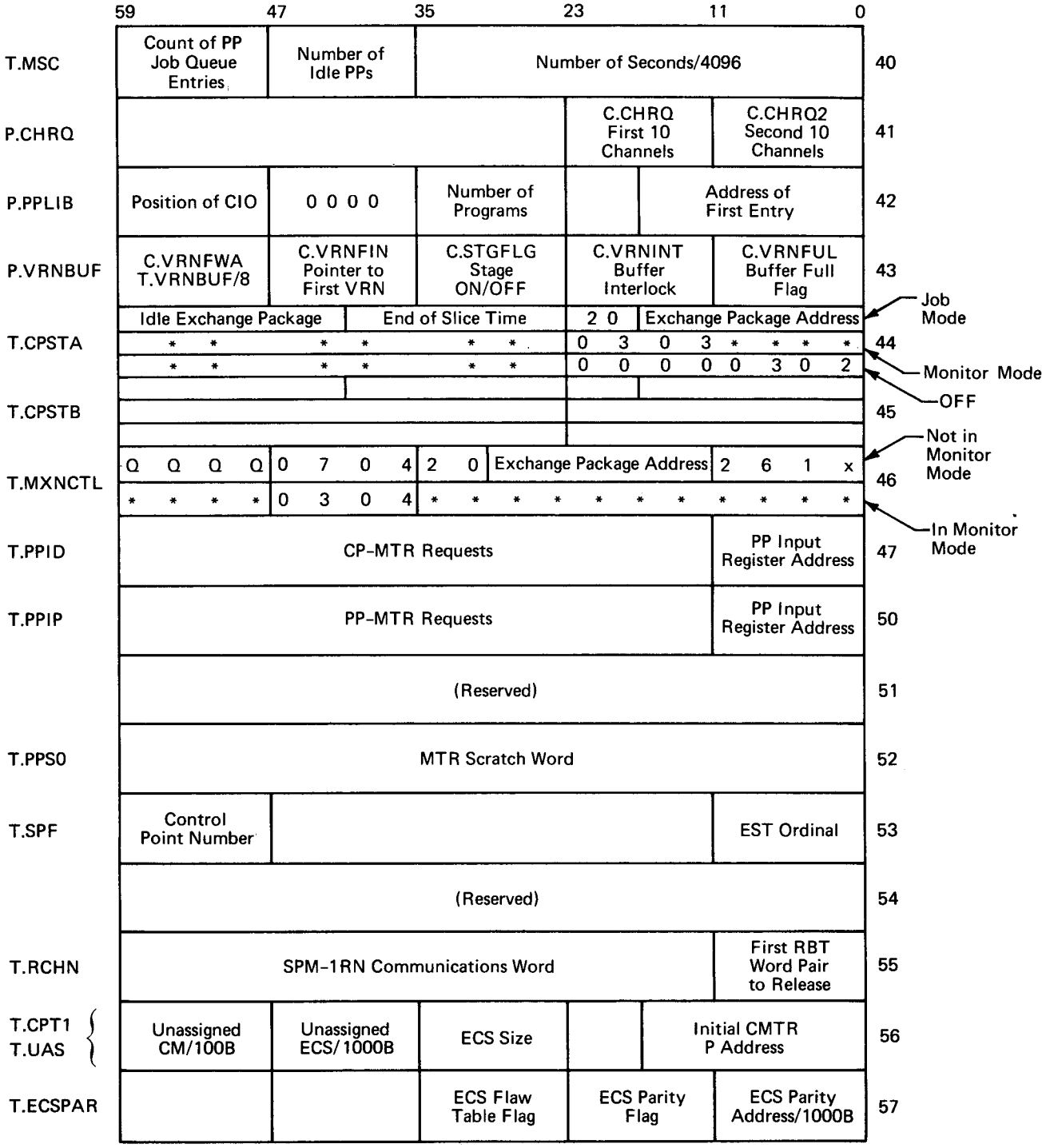
36 T.SLAB6

Unused.

37 T.MSP

Monitor step flag in byte 4, a debug flag is in byte 3.

CMR POINTER AREA



40 T.MSC

Real time clock (microsecond count) in bytes 2 through 4. Count of jobs in PP job queue in byte 0; a count of idle PPs is in byte 1.

41 P.CHRQ

Active channel request queue flags for the first 10 channels in byte 3; for the second 10 channels in byte 4.

42 P.PPLIB

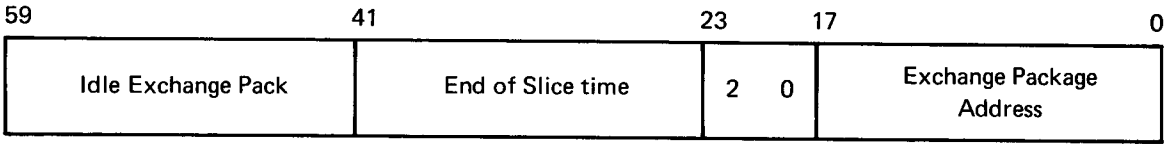
Byte 0 contains the current position of CIO; the number of programs in the system library is given in byte 2. The address of the first library table entry is right-justified in bytes 3 and 4.

43 P.VRNBUF

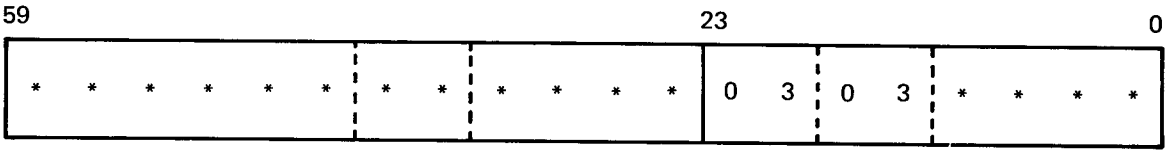
Contains volume serial (visual reel) number buffer information and tape staging flags.

44 T.CPSTA
45 T.CPSTB (one word for each CPU)

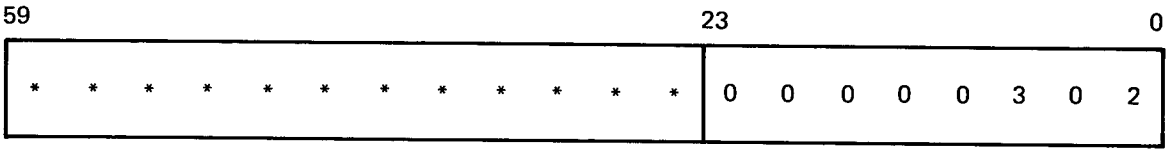
Value while the CPU is running in job mode:



Value while the CPU is running in monitor mode:

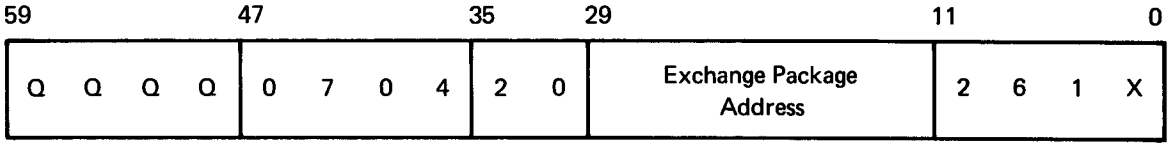


Value while the CPU is turned off:

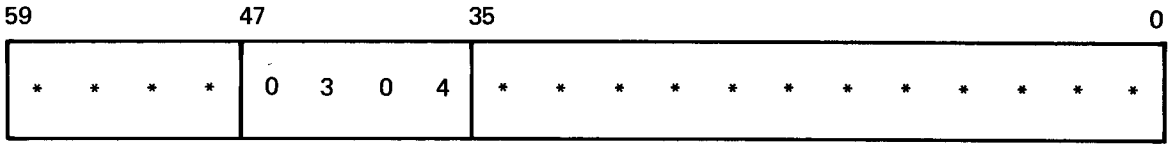


46 T.MXNCTL

Value while not in monitor mode:



Value while in monitor mode:



This word contains PP executable code. It is used to perform the MXN for any PP monitor function.

QQQQ is the complement of the lowest control point job queue priority for which an MXN may be performed. This is used only by time critical jobs. The usual value is 7777 (-0) for MXN systems or 0000 for EXN systems.

X = 0 or 1. The CPU number of the CPU that is running the lower priority job. CMTR sets this and the exchange package address at which the job is running.

47 T.PPID

Contains PP input register address and current monitor request made by CP.

50 T.PPIP

Contains PP input register and current monitor request made by PP.

51 Currently reserved

52 T.PPS0

Used as a scratch word by MTR.

53 T.SPF

Contains the number of the control points currently using CP in byte 0; its EST ordinal is in byte 4.

54 Currently reserved.

55 T.RCHN

First RBT word pointer of the chain to be released in byte 4. The rest of the word contains the SPM-1RN communications word.

56 T.UAS/T.CPT1

Byte 0 contains the number of unassigned CM 100-word storage blocks; byte 1 contains the number of unassigned ECS 1000-word blocks. Byte 2 contains the current size of ECS. Bytes 3 and 4 contain, right-justified, the initial program address of the CM monitor.

57 T.ECSPAR

Bytes 0 and 1 are unused. Byte 2 contains ECS flaw table full flag; byte 3 contains ECS parity flag; byte 4 contains ECS block address in which parity error occurred.

CMR POINTER AREA

	59–47	47–35	35–23	23–11	11–0	
P.SCH	C.SRSL C.LEJDT LE.JDT	C.SRS T.XPSCH/10B	C.JCA T.SCHJCA/10B	C.LJDT L.SCHJDT	C.JDT T.SCHJDT/10B	60
P.STR	C.NFL Needed FL/100B	C.JQP Queue Priority of Job in Counter	C.RFL Reserved FL/100B	C.STMF SCH Recall	C.AFL Available FL/100B	61
T.SCHCP	Interlock Word (Scheduler)					62
T.SCHPP	Interlock Word (PP Routines)					63
P.RPT	T.RPT/10B	L.RPT				64
P.MAIL P.SWPECS P.SCHPT	C.MAILF T.MAIL/10B	C.MAILL L.MAIL	C.SWPECS L.ECSSWP	C.SCHPT T.SCHPT/10B		65
	(Reserved)					66
	(Reserved)					67
	(Reserved)					70
	(Reserved)					71
	(Reserved)					72
	(Reserved)					73
	(Reserved)					74
	(Reserved)					75
	(Reserved)					76
	(Reserved for DDP Simulator)					77

60 P.SCH

Contains information relative to the integrated scheduler. Pointer to job scheduler exchange package is in byte 1; a pointer to the job control area is in byte 2; length and pointer to the job description table is in bytes 3 and 4; length of job description table entries is in byte 0.

61 P.STR

Information relative to job scheduling.

62 P.SCHCP

Interlock word for integrated scheduler.

63 P.SCHPP

Interlock word for PP routines.

64 P.RPT

Pointer to and length of removable pack table in bytes 0 and 1, respectively.

65 P.MAIL/P.SWPECS/P.SCHPT

Pointer to and length of scheduler mail box buffer in bytes 0 and 1, respectively. The ECS swap flags are in byte 2. Pointer to the job scheduler is in byte 3. The ECS swap flag values are:

00	Swap only INTERCOM job on quantum to ECS.
X1	Swap INTERCOM jobs to ECS at end of job.
1X	Swap batch jobs to ECS at end of quantum or when waiting for CM field length assignment.

66-76 Reserved for future use.

77 Reserved for use by DDP Simulator.

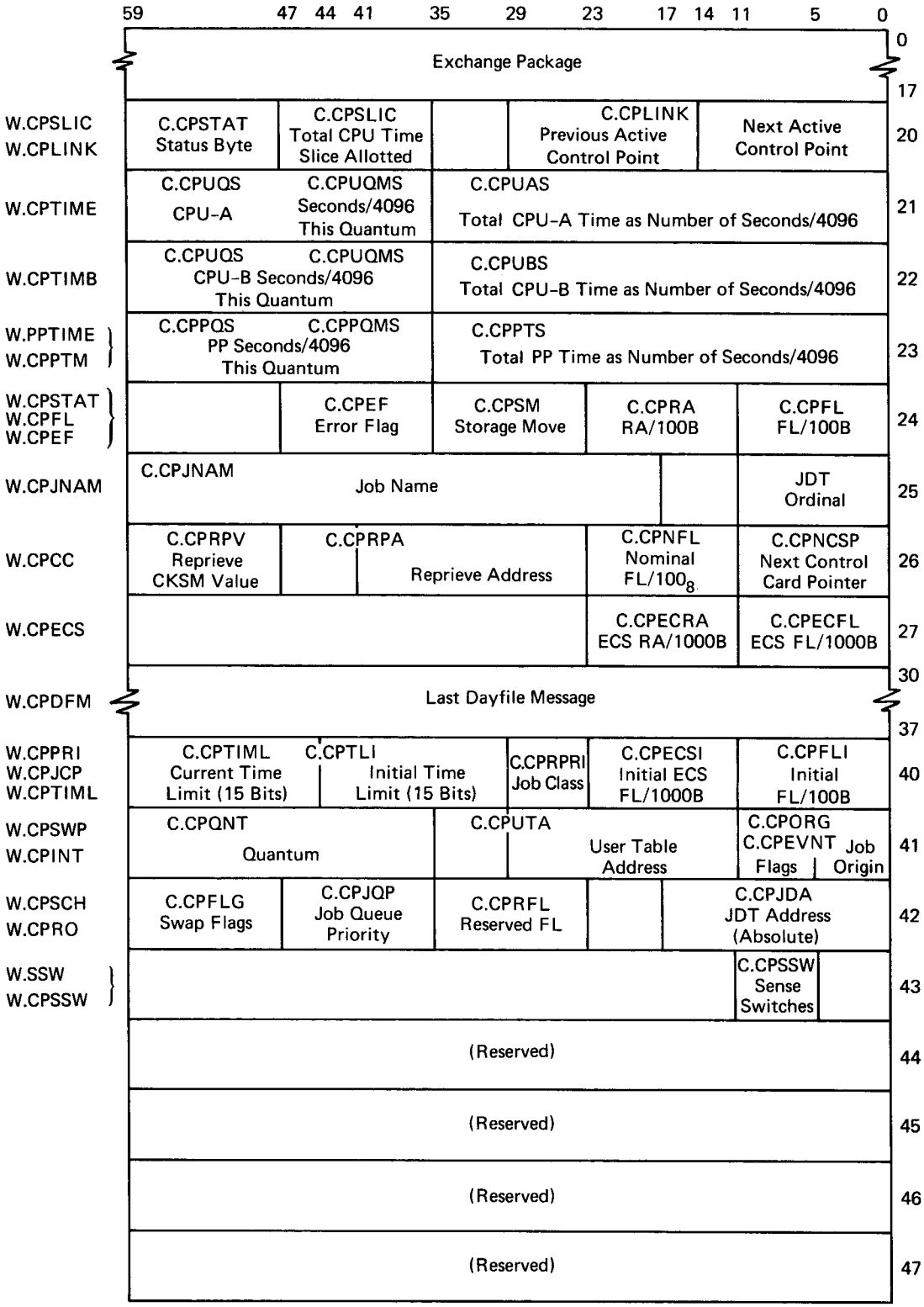
100 T.CST

Entries for the channel status table.

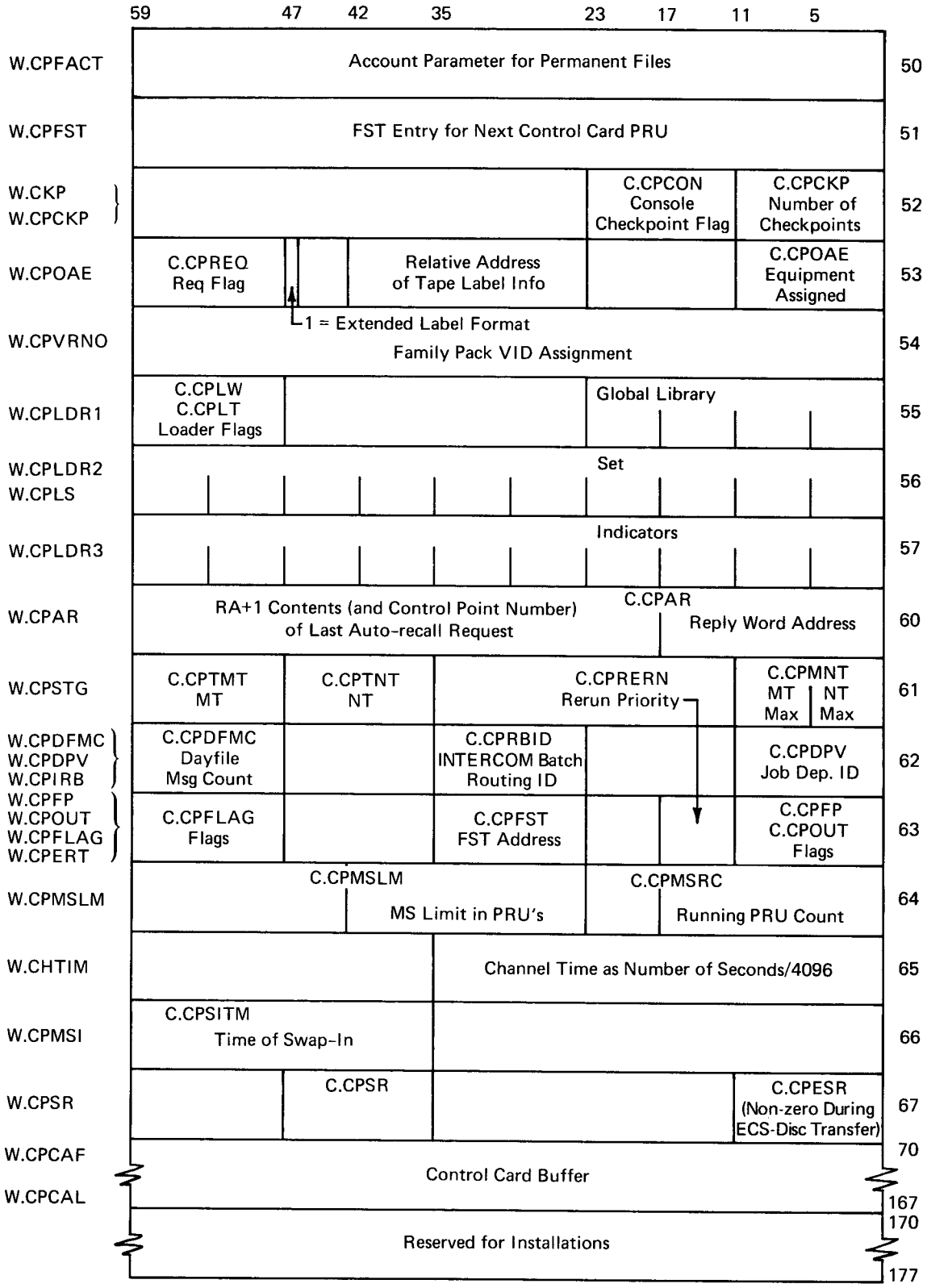
154 T.PPSn

One word entries containing status information for up to 20 PPs.

CONTROL POINT AREA



CONTROL POINT AREA



CONTROL POINT AREA CHART NOTES

MNEMONIC	WORD	BIT
W.CPLINK W.CPSLIC	20	C.CPSTAT Status bits 0 M Move flag move in progress) 1 Y Auto recall 2 A CPU-A assigned only 3 B CPU-B assigned only 4 X Recall status 5 W Wait status 6 R Real-time job (currently not used) 7 C Active - CPUA 8 D Active - CPUB 9 S Activity suspended by job swapper. 10 P Activity suspended by checkpoint
W.CPSWP	41	C.CPORG values (octal) 4 Real time 10 Graphics 20 Multi-user 40 INTERCOM 100 Event flag for swap-out
W.CPSCH	42	C.CPFLG bits 0 unused 1 unused 2 unused 3 S.CP1IB 1IB operating at control point 4 S.CPFFL FNTs in positive FL 5 S.CPEOJ end of job 6 S.CPCLR control point area clear request 7 S.CPRFL storage request in progress 8 S.CPROP rollout in progress 9 S.CPSIP swapin in progress 10 S.CPSOP swapout in progress 11 S.CPSWC swapout complete
W.CPLDR1	55	C.CPLW bits 11-10 W Indicator for loader to be used 9-6 M Map Options 5 R Reduce flag 4 S.CPLT Debugging aid flag 3-2 L Library Set indicator 1 S.CPLP Program loaded from non-system library 0 (Reserved)

W.CPFLAG	63	C.CPFLAG bits 0 (S.CPLDAF) MDI interlock 1 Rerun priority present 2 Private pack overflow 3-12 (S.CPNTNT) If on, do not search FNT) 4-12 (Reserved)
W.CPFP	63	C.CPFP bits 0 S.CPL Reprocess 1 S.CPG Abort 2 S.CPA No rerun 3 S.CPS Sequencer 4 S.CPN Checkpoint taken 5 S.CPX EXIT card encountered 6 S.CPDP Private disk pack 7 S.CPEOR Control card EOR unused 8 S.CPJFL Job card field length assigned 9 S.CPJ JANUS 10 S.CPR Remote batch 11 S.CPE INTERCOM
W.CPEF	24	C.CPEF

CONTROL POINT AREA ERROR FLAG VALUES

1	F.ERTL	CP time limit, sensed by MTR
2	F.ERAR	CP arithmetic error, sensed by MTR
3	F.ERPP	PP abort (M.ABORT), requested by PP
4	F.ERCP	CP abort (ABT in RA+1), requested by program
5	F.ERPCE	PP-call error; abort, sensed by MTR
6	F.EROD	Operator drop type-in
7	F.ERK	Kill, operator type-in
10	F.ERRN	Rerun, operator type-in
11	F.EREX	Control card error, set by 1AJ
	F.ERCC	Control card error for INTERCOM job
12	F.ERECP	ECS parity error, sensed by MTR
13	F.ERJC	Job card error
14	F.ERPA	Pre-abort job before coming to control point
15	F.ERRCL	Auto recall error, bad PP call
16	F.ERHANG	Job hung in auto recall
17	F.ERMSL	Mass storage limit exceeded by stack processor
61	F.ERPARF	Parity error sensed by Scheduler
62	F.ERPAR	Parity error sensed by Scheduler

PP COMMUNICATIONS AREAS

Symbol	Function Description
T.PPC1 T.PPC2 T.PPC3 T.PPC4 T.PPC5 T.PPC6 T.PPC7 T.PPC8 T.PPC9	Nine 8-word areas are set aside for communications with PP units 1 through 9.
T.CPIR	Control point n + 1 input register
T.CPOR	Control point n + 1 output register

The last two words are used for communication between MTR and CP programs; this area is the MTR communications region.

RELATIVE WORD DESCRIPTIONS IN PP COMMUNICATIONS AREA (T.PPCn)

0	W.PPIR	PP input register
1	W.PPOR	PP output register (scanned by MTR)
2-6	W.PPMESn	Six-word message buffer used to pass parameters between CP and PP programs.

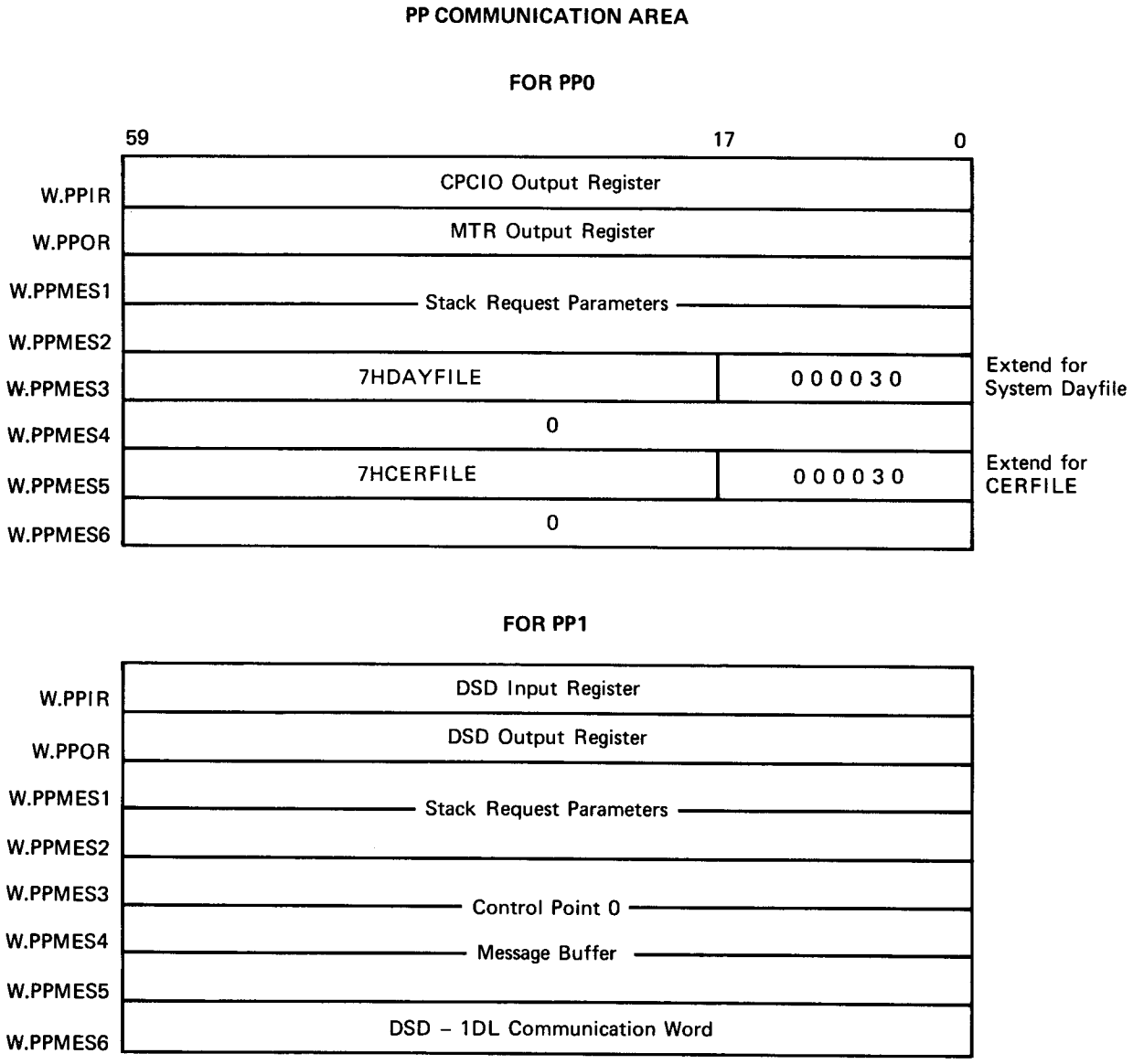


Figure 2-3. PP Communications Areas

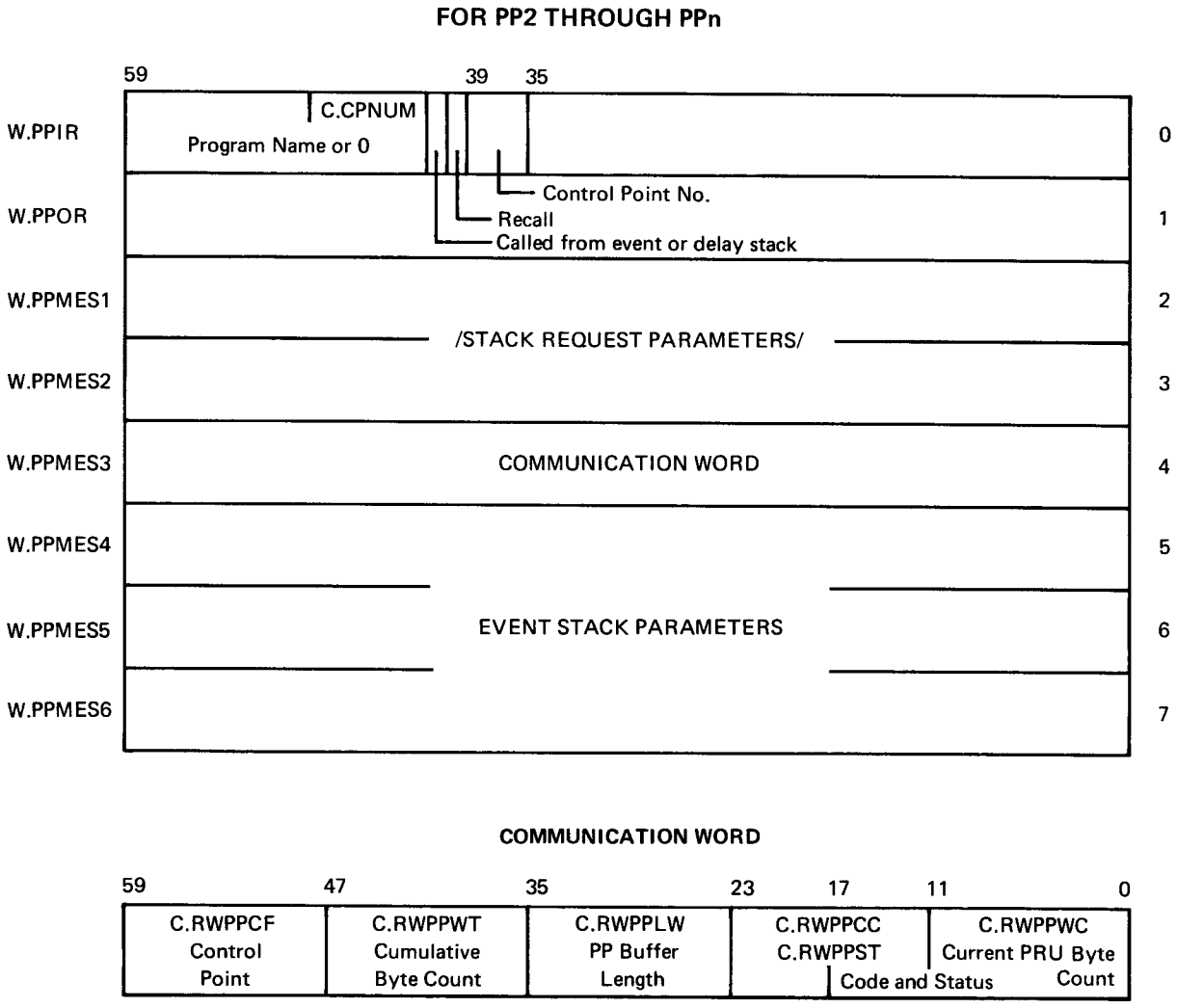


Figure 2-3. PP Communications Areas (continued)

CENTRAL MEMORY RESIDENT

0		Pointers
100		Channel Status Table
154		PP Status Words
200	$T.CPA_n$	Control Point Areas
	T.XPIDLA	System Exchange Packages
	$T.PPC_n$	PP Communication Areas
*	T.EST	Equipment Status Table
*	T.FNT	File Name Table
*	T.ITABL	INTERCOM Table
*	T.DAT	Device Activity Table
*	T.STG	Tape Staging Table
*	T.APF	Attached Permanent File Table
‡	T.RQS	Request Stack
	T.RBR	Record Block Reservation Table (Headers)
	T.RBRBIT	RBR Bit Table
	T.DST	Device Status Table
	T.DPT	Device Pool Table
	T.SEQ	Sequencer Table
	T.RMS	Rotating Mass Storage Diagnostic Table
	T.INS	Installation Area
	T.VSNBUF	VSN Buffer
	T.TAPES	Tapes Table
	T.RPT	Removable Pack Table
	T.MAIL	Scheduler Mailbox Buffer
	T.DFB	Dayfile Buffers
	T.PJT	Parameter Storage for Delayed PP Jobs
	T.SCHPT	(Optional) Scheduler Statistics
	T.SCHJCA	Scheduler Job Control Area
	T.SCHJDT	Scheduler Job Descriptor Table
	T.SUBPG	Subpage Buffer
	T.ECTL	ECS Buffer for RMS—ECS Transfer
	T.EPAGE	Empty Page Stack
		CM Resident Programs
	T.LIB	Library Directory
		INTERCOM Pointer Area
		INTERCOM Small Buffers and User Tables
		INTERCOM Large Buffers

*Table Must Begin Before 10000_8

‡Table Must Begin Before 20000_8

SUMMARY OF TABLES IN UPPER TABLE AREA

Equipment Status Table One entry for each device in the system configuration. Non-allocatable devices can be assigned to one control point at a time; allocatable devices may be attached to many control points simultaneously.

File Name Table Contains an entry for each file in the system, created when the file is created. Several entries are preset and remain in the system for duration; these entries are for the system library (deadstart) file, the system and control point dayfiles, and the hardware error file.

INTERCOM Table Provides multiplexer and port definition information for INTERCOM program use. Assembled only by installation parameter IP.INTCM.

Device Activity Table One-word entry for each RMS device in system. Each entry provides dynamic information related to current activity of the RMS device.

Attached Permanent File Table Provides information for the permanent file manager and job use. Control and status information entries are created when a permanent file is cataloged initially or attached to a qualified CP program.

Request Stack Requests for data transfers, device positioning, or logical file operations. Each allocatable device in system may have one or more 3-word entry in this table when a request for its use is pending.

Record Block Reservation Table Provides continuous information as to assignment/availability of record blocks in which file data is recorded on allocatable devices. Strings of bits in the RBRBIT table denote current status of record blocks in each device.

Device Status Table Directly related to the request stack; contains a 2-word entry for each allocatable device in the system, plus an additional pseudo-entry for unassigned file processing.

Device Pool Table Contains a 10-word area for each RMS device. Used by 1SP and 1EP to pass their internal pool area to each other.

Sequencer Table Contains 30-word entry for each preallocated RMS device for use for CE diagnostic programs.

RMS Diagnostic Table Reflects areas on the system rotating mass storage device preallocated during deadstart to customer engineering diagnostic programs.

Installation Table Reserved for specific needs of installation. Tables are generated in the area only by installation.

Nova Table Created only by installation parameter for support of 250 Interactive Graphics devices.

Tape Staging Table Defines availability, assignment, and demand for tape devices.

Dayfile Buffer Area Contains dayfile buffers and file environment table entries of the system dayfile, the control point dayfiles, and the hardware error file. The control point zero buffer is at the end of this area.

Peripheral Job Table Contains parameters saved for delayed PP jobs.

Mailbox Used for communications between CP monitor and scheduler.

Scheduler Performance Table Optional table used to collect execution data to study the efficiency of the integrated scheduler. Created by installation parameter IP.SPT set to 1.

Job Control Area Contains entries pertinent to the scheduling of jobs by class and queue priorities.

Job Descriptor Table Contains linked entries for each class of job. Entries describe job requirements, current status, accumulated use time of system components, etc. See Job Processing, Chapter 7.

Move Buffer Used for handling ECS/RMS storage moves and partitioning.

Library Directory Falls at the end of the CMR upper table area following the CM resident programs and ECS tables; and is of variable length. It contains 2-word entries in the program name table section and 1-word entries in the entry point table. The directory length may expand or contract as programs are added and deleted or as program residence is changed.

CP Resident Programs

Symbol	Name/Function
CP.MTR	Central processor monitor
CP.SM	Central memory storage move
CP.ECSM	ECS storage move (assembled by installation option parameter IP.MECS)
CP.SPM	Stack processor manager
CP.SCH	Memory manager
CP.SCH1	Integrated scheduler
CP.ECOVL	ECS read/write PP overlay
CP.CIO	Central Processor I/O controller

PERIPHERAL PROCESSORS AND SCOPE 3

PERIPHERAL PROCESSOR ORGANIZATION

When SCOPE 3.4 is loaded into the computer at deadstart time, system monitor, MTR, and the system display program, DSD, are loaded into peripheral processors 0 and 1, respectively, where they reside permanently. All peripheral processors in the system contain a group of permanently assigned storage locations called the PP direct cells; each PP contains a copy of the PP resident program which handles common service functions for the PP programs that may be loaded into the unassigned pool processors.

PP programs are loaded into a pool processor by the PP resident and remain in a PP only until they have completed a specific function. When loaded, the program may load additional overlays to help complete its function; on completion, the program may be overlaid by another transient program loaded by PP resident to perform another, often unrelated, function. Figure (3-1) shows a typical layout of a pool PP loaded with a transient PP program and an overlay. The PP direct cells occupy locations 0 to 77; the PP resident is loaded starting at location 103 to approximately 777. The remainder of the PP is occupied by PP transient programs, except for PP0 and PP1 which contain MTR and DSD, respectively.

Specific assignment of the PP direct cells is detailed in the following chart.

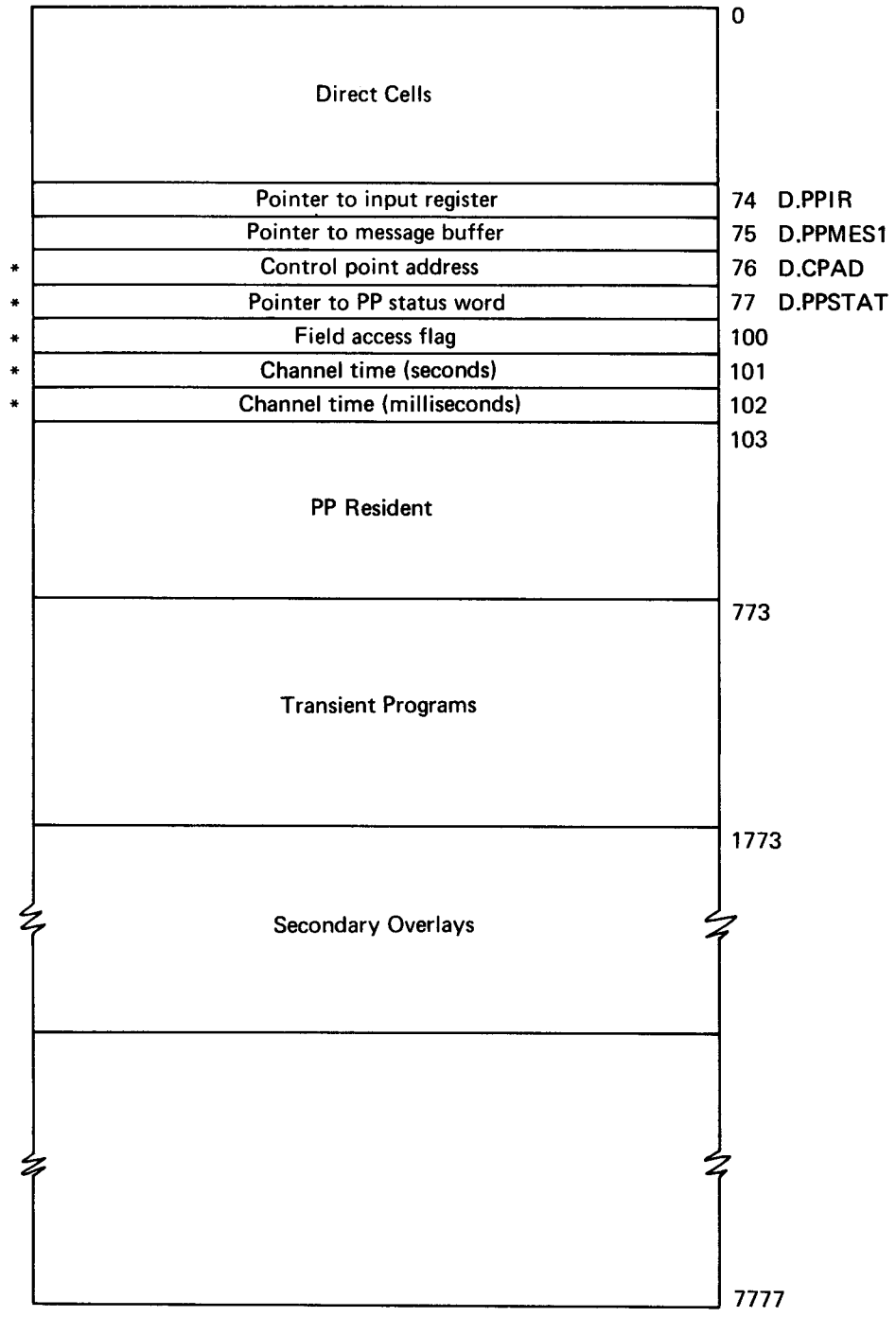


Figure 3-1. Pool PP Layout

* Cells 76 - 102 constitute the five bytes of the PP status word in central memory.

PP DIRECT CELL ASSIGNMENT

Octal	Identifier	Function
0	D.Z0	Temporary storage
1	D.Z1	
2	D.Z2	
3	D.Z3	
4	D.Z4	
5	D.Z5	
6	D.Z6	
7	D.Z7	
10	D.T0	PP output register buffer
11	D.T1	
12	D.T2	
13	D.T3	
14	D.T4	
15	D.T5	Temporary storage
16	D.T6	
17	D.T7	
20	D.FNT/D.TW0	D.FNT through D.FNT+9 contain words 2 and 3 of the FNT, referred to as the file status table (FST).
32	D.EST/D.JPAR D.TH2	D.EST through D.EST+4 contain the EST entry in process. D.JPAR contains a job parity flag.
37	D.DTS/D.JFL D.TH7	D.DTS contains device type code in left 6 bits and allocation type code in right 6 bits; D.JFL contains CM field length requirement returned to caller by 2TJ.
40	D.BA/D.FR0	Contains first word of FET in D.BA through D.BA+4 (buffer address).
45	D.JECS/D.FR5	ECS field length returned by 2TJ to caller.
46	D.JPR/D.FR6	Computed job priority returned to caller by 2TJ.
47	D.JTL/DFR7	Job time limit returned to caller by 2TJ.
50	D.PPIRB/D.FF0	D.PPIRB through D.PPIRB+4 contain PP input register contents.
55	D.RA/D.FF5	Reference address divided by 100 (octal) control point to which PP is attached.
56	D.FL/D.FF6	CM field length divided by 100 (octal) for job at control point to which PP is attached.
57	D.FA/D.FF7	Address of second word of FNT entry in process.
60	D.FIRST/D.SX0	This and next cell contain 18-bit CM address of word FIRST in circular I/O buffer.
62	D.IN/D.SX2	This and next cell contain 18-bit CM address of word IN in circular I/O buffer.
64	D.OUT/D.SX4	This and next cell contain 18-bit CM address word OUT in circular I/O buffer.

Octal	Identifier	Function
66	D.LIMIT/D.SX6	This and next cell contain 18-bit address of word LIMIT in circular I/O buffer.
70	D.PPONE/D.SV0	Preset to constant value plus one (+1).
71	D.HN/D.SV1	Preset to constant value +100 (octal).
72	D.TH/D.SV2	Preset to constant value +1000 (octal).
73	D.TR/D.SV3	Preset to constant value +3.
74	D.PPIR/D.SV4	PP input register address.
75	D.PPMES1	Address of first word of PP message buffer.
76	D.CPAD	Address of control point area in use by PP.
77	D.PPSTAT	Pointer to PP status word.
100		Field access flag.
101		Channel time in seconds.
102		Channel time in milliseconds.

PP COMMUNICATIONS

For each pool PP, CMR has an area used for communication between the PP monitor and the PP. Each area contains a PP input register and a PP output register, each one CM-word long, plus a six CM-word message buffer. In section 2, Figure 2-3 is a diagram of a PP communications area.

When a PP is idle, the input register in the communications area contains zero. When PPMTR assigns a PP to load and run a transient PP program, it will load a request word into the assigned PP input register. Figure 3-2 shows the format of a PP input register for a transient program called from a CP program. CPMTR will have inserted the requesting program's control point number into bit 36-39 of the word and cleared bit 41. Bits 0-35 appear in the input register exactly as they did in RA+1 of the requesting CP program.

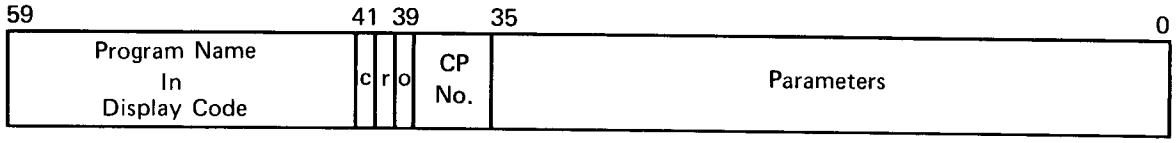


Figure 3-2. PP Input Register

The PP resident in each PP constantly scans its own input register. When it becomes non-zero, the PP resident will search the program name table in the library directory in CMR for the program name specified in the input register. It will load the transient program at location 773 and store the address of the control point area to which the PP is assigned in direct cell D.CPAD, then transfer control to location 1000 to start execution. If the transient needs to load an overlay, it calls a subroutine in the PP resident. PP resident loads the overlay. Since the input register is not cleared until the PP becomes idle, parameters transmitted by PPMTR in the input register can be read by the transient program and/or any overlay. When the PP transient program has completed its function, it sends a request to PPMTR to drop the PP. PPMTR will clear the PP input register and record the fact that the PP is idle and that another program can be loaded into the PP. The transient program terminates by executing a jump to the idle loop of the PP resident which scans the input register for the next assigned task.

When PP resident has a monitor request, it places a message into the PP output register in the PP communication area. The leftmost byte contains a number which identifies the function requested; other bytes may contain parameters for the request. Additional information or parameters for the request may be placed in the message buffer in the PP communications area. After making the request, PP resident waits for the first byte of the output register to be set to zero, signalling that the monitor has processed the request. If CPMTR or PPMTR need to communicate with the PP about the request being processed, it will store the necessary information in the remaining bytes of the output register.

PP RESIDENT

The PP resident program performs two main functions:

Handles all communication between MTR and the transient and/or overlay program

Loads transient programs and overlays and initiates execution of these programs

The PP resident is made up of a series of routines, each of which performs a specific function. (See figure 3-3.) These resident routines are used by the transient and overlay programs as required. The routines, their names, locations, calling sequences and functions are described below.

Field Access Flags	R.FAF
PP Resident Idle Loop	R.IDLE
Load Primary PP Overlay	R.OVLJ
Load PP Overlay	R.OVL
Transmit Data To/From Stack Processor	R.READP R.WRITEP
Special Words Used by Loader	R.RWP R.RWPP
Access Request Stack Entry	R.EREQS
Request Access to Control Point Field Length	R.RAFL R.PAUSE
Terminate Access to Control Point Field Length	R.TAFL
Compare Accumulator to Field Length	R.TFL
Process Monitor Function	R.MTR R.PROCES
Wait for Output Register to Clear	R.WAIT
Reserve Channel	R.RCH
Drop Channel	R.DCH
Mask a Byte into Specified Words	R.STBMSK R.STB
Transmit Dayfile Message	R.DFM

Figure 3-3. PP Resident Routines

(103) R.IDLE PP Resident Idle Loop

Calling sequence: LJM R.IDLE

In the idle loop, PP resident continually scans its input register for an assigned task. R.IDLE destroys direct cells 20 through 22 and some of the temporary storage cells; R.OVLJ and all other PP resident routines destroy only the temporary cells 0 through 17.

(135) R.OVLJ Primary (transient program) Overlay Loader

Calling sequence: Store name of overlay left-justified in D.T6, D.T7
LJM R.OVLJ

When this routine is called, the resident loads a new primary overlay at C.PPFWA minus L.PPHDR and transfers control to location C.PPFWA.

(144) R.OVL Overlay Loader

Calling sequence: Load A register with load point address
RJM R.OVL

A search for the secondary overlay whose name is left justified in D.T6 and D.T7 will be made in the PP program name table (PPNT). If found, it will be loaded into the PP beginning at the address given in the A register. R.OVL is used by both PP resident and PP overlays to load the named overlay. To load disk resident overlays, PP resident does not reference the disk directly, it makes a call to the stack processor by calling R.READP. If the overlay is not located, an OVLERR error message is produced and the control point is aborted.

(316) R.READP Transmit Data Via Channel from Stack Processor

(325) R.WRITEP Transmit Data Via Channel to Stack Processor

Calling Sequence: Load L(request)
RJM R.READP (R.WRITEP)

Computes PP word count from first and last word addresses given in the request formatted at the request location and adds the computed word count, the address of the PP message buffer, and the control point number to the request. The request is entered in the stack, and data is transmitted via channel directly to/from PP memory. Upon exit from R.READP (R.WRITEP), the following information is set:

(D.T3 + C.RWPPLW) LWA + 1 of data transmitted

(D.T3 + C.RWPPST) Upper 6 bits of status

(D.T3 + C.RWPPWT) Number of PP words transmitted

(D.T4 + C.RWPPST) Lower 12 bits of status

(342) R.RWP Special entry point to R.READP used by LDR.

(356) R.RWPP Word in R. READP modfied by LDR.

(431) R.EREQS Enter Stack Request

Calling Sequence:
```
Store L(request) in D.TO
RJM    R.EREQS
```

This routine adds the control point number to the already formatted request and searches the central memory request stack for an empty entry. The monitor function, M.EREQS, is called, and PP resident iterates until MTR accepts the request. If the available flag is set (S.STF+S.STFA of byte C.STFB of the second word of the request), R.EREQS exits to R.IDLE. Otherwise, it returns control to its caller.

(464) R.RAFL Request Control Point Field Length Access

Calling Sequence:
```
RJM    R.RAFL
```

The storage move flag for the control point is tested. If set, a call is made to R.TAFL; when clear, the field access flag in the PP status word is set, the RA in D.RA is reset, and the FL in D.FL is reset.

(R.PAUSE is the same as R.RAFL.)

(517) R.TAFL Terminate Control Point Field Length Access

Calling Sequence:
```
RJM    R.TAFL
```

This routine is called to clear the field access flags in the PP byte R.FAF and in the PP status word.

(533) R.TFL Test Field Length

Calling Sequence:
```
Load relative address
RJM    R.TFL
```

This routine ensures that a relative address is within the field length limits. The 18-bit address is added to the control point reference address and compared with the field length. If the resultant address is out of range, R.TFL exits with a negative A register; otherwise, the A register will contain the resultant absolute CM address (RA+relative address) upon exit. The RA and FL values are found locally in the PP resident, at R.CPRA and R.CPFL, respectively. These locations are initialized when a new transient program is loaded into the PP; the transient and its overlays cannot call R.TFL until R.RAFL has been called.

(544) R.MTR Process Monitor Function

(R.PROCES is identical with R.MTR)

Calling Sequence:
```
Store function parameters in D.T1 to D.T4
Load function code
RJM    R.MTR
```

Places the function code in D.T0, writes D.T0 through D.T4 to the output register, and waits for the output register to clear via a call to R.WAIT.

(555) R.WAIT PP Wait Loop

Calling Sequence:
```
RJM    R.WAIT
```

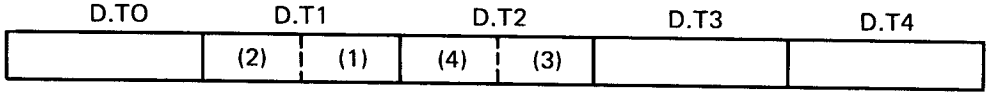
Determines if the monitor function is for MTR or CPMTR. If the MXN instruction is not available, R.WAIT is modified at deadstart causing R.WAIT to assume that all functions are for MTR. If the function is for CPMTR, the PP input register address is written into T.PPID and T.MXNCTL is read in and executed. If the function is for MTR the input register address is written in T.PPIP.

After either action, R.WAIT idles until byte zero of the output register is cleared.

(617) R.RCH Request Channel

Calling Sequence:
```
Load channel numbers
RJM    R.RCH
```

Channel numbers loaded in the A register will be stored in D.T1, the monitor function M.RCH inserted into D.T4, and then D.T0 through D.T4 will be written to the output register for that PP. Channel numbers in D.T1 and D.T2 will be assigned by monitor on the following priority basis:



The highest priority is given to the channel number in the rightmost 6 bits of D.T1; the second highest to the channel number in the leftmost 6 bits of D.T1, etc.

When assigning alternate channels, monitor will discontinue its search of D.T1 and D.T2 when it encounters 6 zero bits. If only one alternate channel is wanted, the programmer must clear D.T2 before calling R.RCH. As an example, the coding for requesting primary channel 12 alternate channel 13 would be:

```
LDN    0
STD    D.T2
LDC    1312B
RJM    R.RRCH
```

Normally, MTR will stop looking for alternate channels after four have been investigated; in the above example, only two channels will be investigated.

When R.RCH is called, the function is not considered complete until byte 0 of the output register is cleared, which signals that a channel has been assigned.

If an M.RCH channel request is made directly to R.MTR, additional action is taken. If MTR cannot assign a channel, it moves byte 4 of the output register to byte 0 and signals the requesting PP in byte 4 of the output register to wait unitl the channel is assigned.

(627) R.DCH Drop Channel

Calling Sequence:

```
Load channel number
RJM    R.DCH
```

The specified channel will be dropped. Since more than one PP can request the same channel at the same time, a MTR request must be used to reserve a channel. Only the PP reserving the channel can release it by making a R.DCH call; the function will modify the CST entry for the channel to indicate that it is free.

(641) R.STBMSK

Address in PP resident of a logical mask used by R.STB routine. This mask is initially 7700 octal; the value should be restored by any routine which substitutes an alternate mask.

(650) R.STB Store Byte

Calling Sequence:

```
Load L(List)
RJM    R.STB
```

List has the form:

```
L(BYTE)
L(WORD1)
L(WORD2)
.
.
.
L(WORDn)
ZERO
```

A logical AND is performed on the mask at location R.STBMSK for each word in the list before the word is exclusive-ORed with word BYTE. R.STB is used primarily to substitute channel numbers in driver overlays.

(656) R.DFM Enter Dayfile Message

Calling Sequence:

```
Load L(message) + flag bits
LOAD R.DFM
```

A message is written to the dayfile and/or displayed on console. The flag bits in the high-order 6 bits of the A register are used to determine message destinations. In the flag bit values given below, one or more bits may be on; all are optional.

Bit 1	Dayfile A display only
Bit 2	Control point 0 (system) message
Bit 4	Dayfile (no A display)
Bit 12	A $ is inserted in 20th character of message sent to system dayfile (installation use only.)

PP PROGRAM NAME RESERVATIONS (3.4)

8AA thru 8A9	Reserved
8BA thru 8B9	Reserved
8CA thru 8C9	Customer Engineering
8DA thru 8D9	DSD
8EA thru 8E9	DSD (7000 overlays)
8FA thru 8F9	Reserved
.	.
.	.
.	.
8WA thru 8W9	Reserved
8XA thru 8X9	DSD
8YA thru 8Y9	Reserved
8ZA thru 8Z9	INTERCOM
9AA thru 9A9	Customer Engineering
.	.
.	.
.	.
9YA thru 9Y9	Customer Engineering
9ZA thru 9Z9	INTERCOM

All PP overlay names (OV.xxx symbols) are listed in Appendix B.

SCOPE SYSTEM MONITOR

In SCOPE 3.4, there are two separate monitors: CPMTR (CP.MTR) which controls CPU monitor mode execution and CPU scheduling, and the PPMTR (OV.MTR) which is in general control of the system and operates in PP0.

DEFINITION OF MONITOR TERMINOLOGY

MODES

The CPU executes in either Monitor Mode or User Mode. The state is determined by a Monitor Mode flag which is either set or clear, respectively. The effect of the flag and the CPU mode determines the behavior of exchange jumps.

In Monitor Mode, the CPU is permitted to execute without interruption until a task has been completed. In SCOPE 3.4, this mode is used by CPMTR, SPM (Stack Processor Manager) and CPCIO to process ECS I/O transfers. Monitor Mode is terminated by an exchange jump to a program in user mode.

All programs, including system programs (with the exception of CPMTR, SPM and CPCIO) run in User Mode which can be interrupted at any time, if the CPU is needed by a Monitor Mode program. Use of the CPU is time-sliced for each User Mode program. If it has not relinquished the use of the CPU within the span of the time slice, the User Mode program is interrupted at the expiration of the time slice.

Separate exchange package areas are reserved for each user mode program, including IDLE. While the CPU is in user mode, the CPMTR exchange package is stored in the exchange package area of the program that is in execution. Refer to the System Program Job Exchange Package Area, Figure 3-4, and Exchange Package, Figure 3-5.

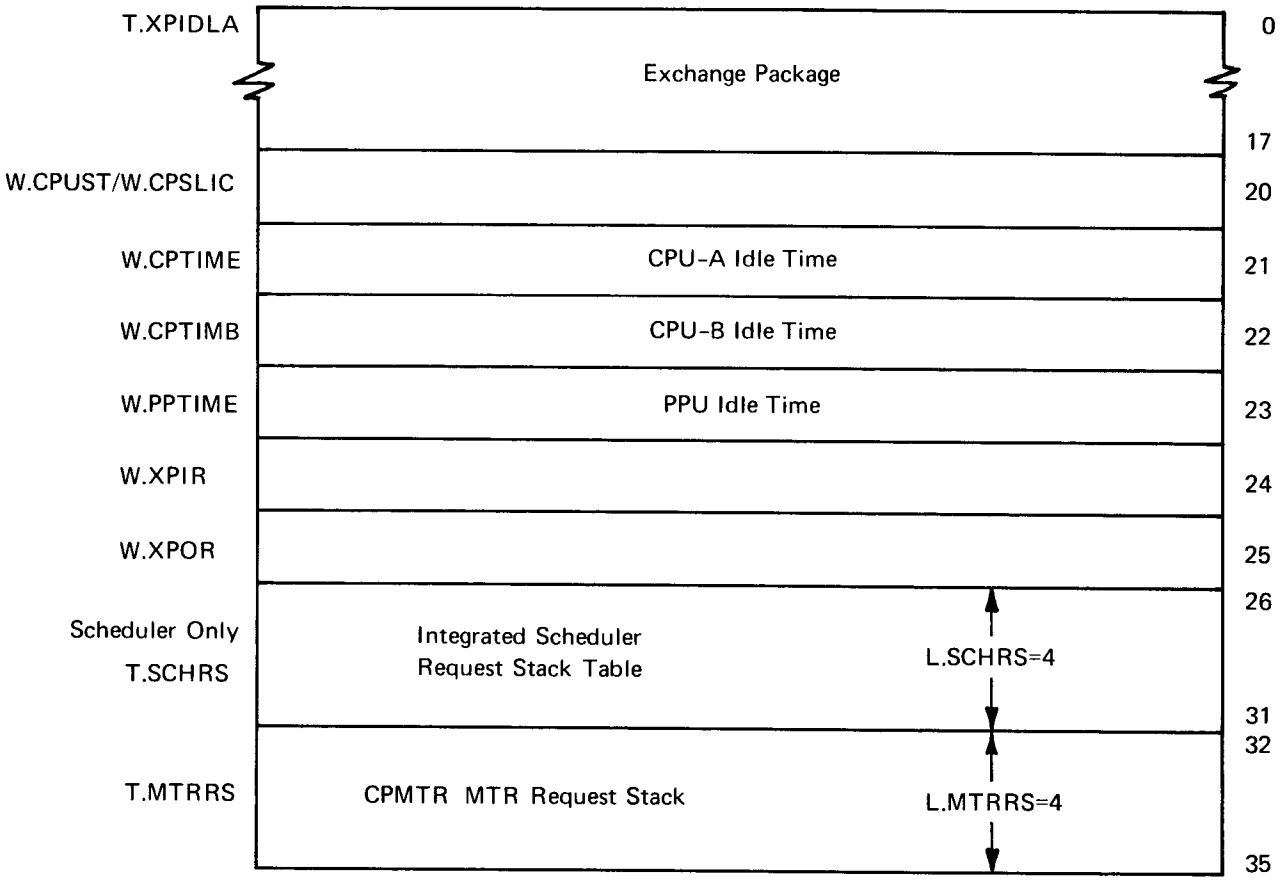


Figure 3-4. System Job Exchange Package Areas

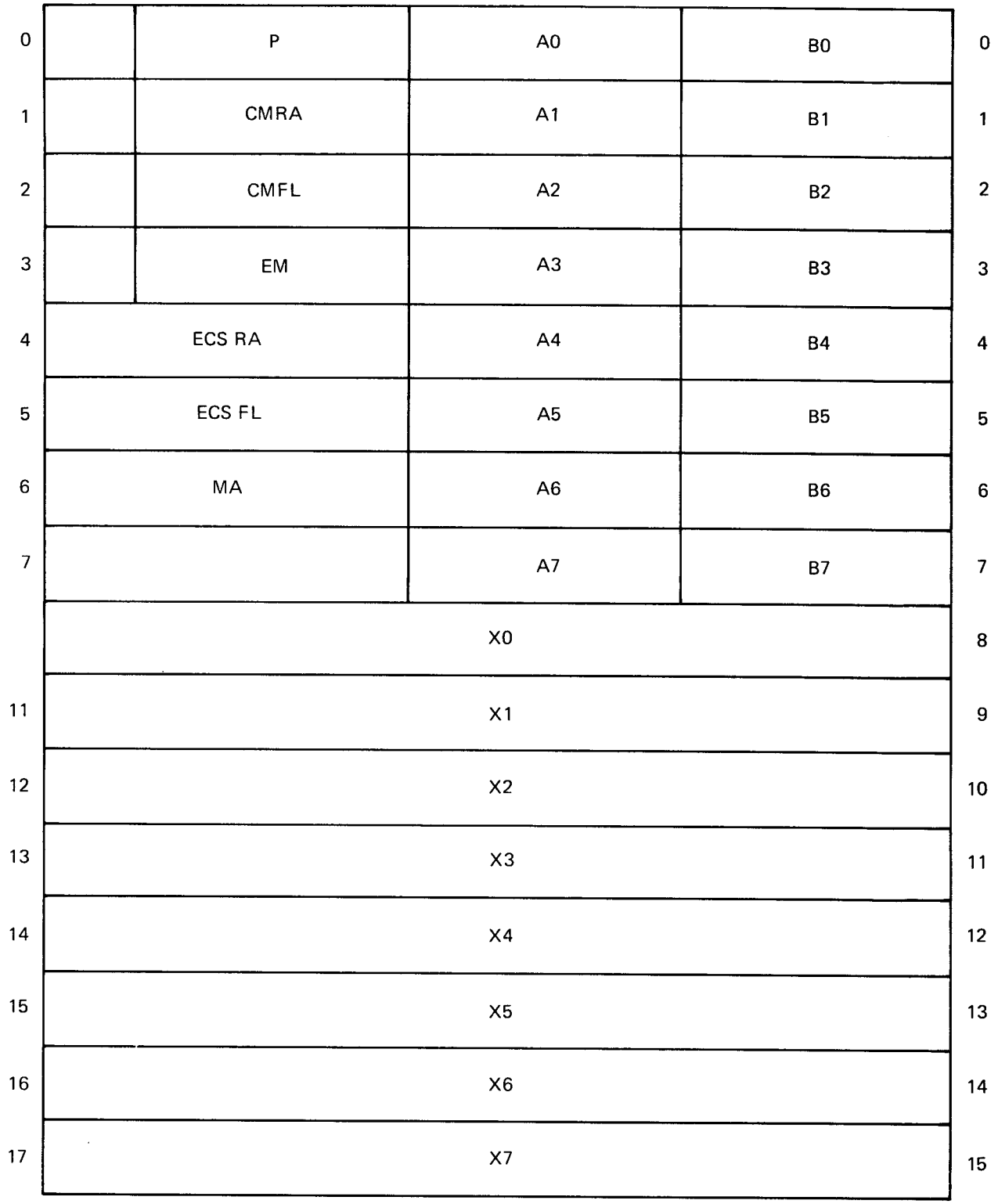


Figure 3-5. Exchange Package

EXCHANGE JUMPS

An installation may make use of the optional hardware instructions MXN (Monitor Exchange) and XJ (Exchange Jump) or EXN (Exchange). SCOPE 3.4 requires either the combination of MXN/XJ or EXN.

MXN

This is a PP initiated exchange jump to place the CPU in monitor mode execution. This exchange jump is conditional upon the setting of the Monitor Mode flag, which represents the execution mode of the CPU. If the flag is set, the CPU is in Monitor Mode and the MXN will be treated as a Pass instruction. If the flag bit is clear, this instruction will set the flag and initiate the exchange of data in the system job exchange package area, placing the CPU into Monitor Mode.

XJ

The Exchange instruction is to be used in conjunction with MXN. The action of the XJ depends upon the mode of the CPU. When the CPU is in User Mode, the CPU executes in Monitor Mode upon completion of the exchange jump. The CPU executes in User Mode upon completion of an XJ issued in Monitor Mode. The CMR Resident programs CP.SM (Storage Move) and CP.SCH1 (Integrated Scheduler) are system programs that run in User Mode and use the XJ when available in the system.

EXN

In an installation without the MXN/XJ instruction set, the EXN instruction is used. It is a PP initiated exchange jump which occurs independently of the mode of the CPU and has no effect of 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 CPMTR in a PP output register, it will:

EXN to the exchange package address given in the CPU status word T.CPSTA in order to initiate CPMTR.

Loop on an RPN instruction until P=0. (Since XJ does not exist, CPMTR will terminate on an jump to zero.)

EXN to the exchange package address given in T.CPSTA and then restore the CPMTR address in the exchange package P register. If there is no job to run, T.CPSTA will point to the idle package. When there are two CPUs in the system, T.CPSTA is used for the first CPU and T.CPSTB is used for the second CPU.

EXCHANGE PACKAGE MANAGEMENT

The transfer of control from one user mode program to another is controlled by CPMTR. CPMTR is initiated either by an XJ from the user mode program in execution or by an MXN to the exchange package whose address is in T.CPSTA (MTR) or T.MXNCTL (PP resident).

The exchange leaves the exchange package of the user mode program in its own area, with CPMTR being executed. CPMTR modifies T.CPSTA and T.MXNCTL to prevent another MXN to the old exchange package address. It then selects which user mode job should be executed next, rewrites T.CPSTA and T.MXNCTL with the new exchange package address and executes an XJ to that exchange package area.

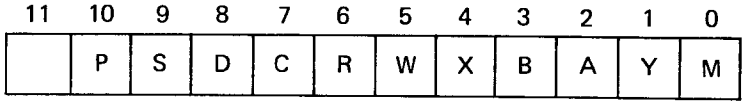
A 6500 or 6700 has an IDLE exchange package area for each CPU. This permits both CPU's to execute the IDLE program simultaneously.

EXCHANGE PACKAGE CONTROL WORD

A word in the lower table area of CMR contains executable PP code used by PP resident to perform the Monitor Exchange Jump (MXN) for any CPMTR function. See description of T.MXNCTL in Section 2.

CONTROL POINT STATUS

The status of the control point is contained in byte C.CPSTAT of word W.CPUST in the control point area.



At any given time, the status of a control point can be determined by the combination of bits set.

M — Set by CPMTR to indicate that a storage move is in progress for this control point.

X, Y — Used to indicate the type of recall associated with a peripheral task request and are set to temporarily relinquish the CPU while the task is in progress. The X-bit indicates periodic recall and is set by the user through the use of RCL. The Y-bit is set by any RA + 1 call in which the auto-recall bit is set.

A, B — Used to indicate the CPU in which the control point may execute. Delegation to CPU A and CPU B are indicated by bits A and B, respectively. The job must be executed by the CPU to which it is delegated.

W — The bit is generally set by an M.RCP function and remains set until an END or ABT is posted from RA + 1 or until an M.DCP function is requested.

C, D	Used to indicate the CPU in which the job is currently executing. CPU A and CPU B are indicated by bits C and D, respectively.
P, S	Used to indicate the suspension of execution of the control point by either the job swapper (S) or checkpoint dump (P).
R	Used to indicate a real-time job (currently used).

All of the control points for which W is set (is assigned to a CPU) and P, S, X, and Y are clear (execution not suspended or in recall), are linked together into a circular chain. At the end of a CPU time-slice period, CPMTR will advance to the next job in the chain for which neither C, D, or M is set (not currently in execution or undergoing a storage move).

CPU STATUS

A control word for each CPU appears in the lower table area of CMR. Each can have a different value when the CPU is running in Monitor Mode, User Mode, or when the CPU is turned off. See Section 2 for words T.CPSTA and T.CPSTB and the description of their contents.

CPMTR

The CP monitor has several specific tasks: Control point status processing, RA+1 request processing, processing CP monitor requests from PPs, reservation and assignment of channels.

STATUS PROCESSING

When a CPU is required for a control point, the Wait (W) status bit is set in the control point status byte C.CPSTAT in the control point area. The control point is linked into the chain of control points requesting a CPU which is then processed by the CPU selection process.

RA + 1 REQUEST PROCESSING

CPMTR distinguishes between system and user CPU programs as follows:

System programs run at control point N.CP+1 with an RA equal to zero, a field length of 377777, depending on CM size. These programs cannot make requests through RA+1; they are initiated by CPMTR and use an output register associated with their exchange package area.

User programs run at control point 1 through N.CP and communicate with CPMTR via RA+1. CPMTR scans periodically the contents of RA+1 in each active program. If RA+1 is non-zero, CPMTR considers it to be a request and processes it before clearing RA+1.

The first three characters in RA+1 determine the request type and action to be taken:

CIO — CPMTR checks the request to see if it is a PP call or a call for CPCIO and processes it accordingly.

END — End of user program and set the control point to zero status. The CPU selection routine is then called.

ABT — Abort the program. Set the error flag to F.ERCP for the control point and set the control point to zero status.

RCL — Enter recall status:

If the automatic recall bit (40) is not set, the control point is put in X status and periodic recall delay is initiated.

If the automatic recall bit is set and the parameter address points to an incomplete status, the control point is put in Y status.

If the automatic recall bit is set and the parameter address points to a complete status, the request is considered to be a no-operation.

TIM — Supply time or date to the user program, depending upon the value transmitted in RA + 1. CPMTR returns the following information to the location specified in the parameter address:

0	Elapsed CPU time (W.CPTIME/ W.CPTIMB)
1	Current date (T.DATE)
2	Clock time (T.CLK)
3	Julian date (T.JDATE)
4	Elapsed time since deadstart (T.MSC)

If the characters in RA + 1 are other than the above value, they are considered to represent a PP program call if the first character is alphabetic. CPMTR takes the contents of RA + 1, clears bits 36-39 and 41, inserts the control number, and then considers this entry to be a PP input register image, so it will place the entry into the Monitor Request Stack (T.MTRRS).

If the automatic recall bit is set and the parameter address points to an incomplete status, the control point is put into X + Y status. Every time the general activity count of the control point is reduced, CPMTR checks the status and puts the control point into W status if the completion bit is set. When such a request is encountered, CPMTR records the status address by writing the complete request from RA + 1 to word W.CPAR in the control point area. If the automatic recall bit is set and the parameter address points to a status already complete, the control point is aborted.

During CP request processing, CPMTR can set the following error flags:

F.ERCP — Control point aborted. Error flag is set in response to an ABT request.

F.ERPCE PP call error detected when a PP program call does not begin with an alphabetic character or when an invalid parameter is transmitted.

F.ERRCL Automatic recall error detected when a PP program call is issued with the automatic recall bit set and the parameter address points to a complete status.

CURRENT CPMTR FUNCTION CODES AND MNEMONICS

01	M.SETST	Set CP status bit
02	M.CLRST	Clear CP status bit
03	M.RCP	Request central processor
04	M.DCP	Drop central processor job
05	M.RCLCP	Recall central processor job
06	M.ICE	Initiate central executive
07	M.CPUST	Change CPU status (IP.MCPU $\neq$ 1)
10	M.SLICE	MTR interrupt at end of job's CPU time slice
11	not used	
12	M.RCH	Reserve channel

In the request format, those bits or bytes indicated by * have no significance to the function.

M.SETST SET STATUS BITS
(0001,BBBB,****,****,00NN)

BBBB	Pattern of bits to be set
NN	Control point number (only if in MTR output register)

Called to set CP status bits in byte C.CPSTAT in control point NN area. May cause linkage to or delinkage from chain of control points actively waiting for a CPU.

M.CLRST CLEAR STATUS
(0002,BBBB,****,****,00NN)

BBBB	Pattern of bits to be cleared
NN	Control point number (only is in MTR output register)

Called to clear CP status bits in byte C.CPSTAT in control point linkage. Will cause linkage to or delinkage from chain of control points actively waiting for a CPU.

M.RCP REQUEST CENTRAL PROCESSOR
(0003,****,****,****,****)

This request is ignored under the following conditions:

Requesting PPU is assigned to control point zero

Error flag is set for the control point

Job is already in the waiting status.

If none of the above conditions exist, CPMTR sets the job in waiting status (W).

M.DCP DROP CENTRAL PROCESSOR JOB
(0004,****,****,****,****)

Execution of the central processor job at control point is stopped. The control point status is set to zero status (Z; the W and X bits are clear), the secondary status, bits M, Y, A, and B are not altered. The control point status bits set prior to M.DCP are returned in byte 1 of the output register of the requesting PPU.

M.RCLCP RECALL CENTRAL PROGRAM
(0005,****,****,****,****)

This request is effective only if the central program associated with the requesting PPU is in recall status, and no error flag is set at the control point. The status of the control point is set to waiting (W). In any other case, the status of the control point is not altered.

M.ICE INITIATE CENTRAL EXECUTIVE
(0006,****,****,****,IIII)

IIII identifies a central memory resident program which will be started by CPMTR upon recognition of this request. This system program will run at control point N.CP + 1 with a priority of 7777 (RA = 0, FL = 377777, ECS RA = 0, ECS FL = 10000000). It will terminate by setting the output register T.CPIR to zero; MTR loops and tests T.CPIR for non-zero. The M.ICE request is delayed if a system program is already active; CPMTR initiates only one system program at a time.

IIII	Program
0	CM storage move
1	ECS storage move
2	Unused
3	Stack Processor Manager
4	Load PP program from ECS library
5	Unused
6	Unused
7	Request ECS buffer
10	Release ECS buffer
11	Request system buffer
12	Release system buffer
13	RMS-ECS move
14	ECS-RMS move
15	Flush ECS buffer

M.CPUST CHANGE CPU STATUS
(0007,***X,****,****,00NN)

(1) N is non-zero
The request is ignored if a CPU is OFF. Otherwise, the control point status is dedicated to CPU X (X = 1 or 2) and will be allowed to use this CPU for the duration of the job.

(2) N = 0, X = 0
If either CPU is off, it is returned to the ON status, unless it was locked off at deadstart load time.

(3) N = 0, X = 1 or 2
CPU X is turned off and dedication of any control point to this CPU is cancelled.

(4) N ≠ 0, X = 0
Control point N is released from dedicated status.

M.SLICE TERMINATE TIME SLICE PERIOD
(0010,****,****,****,****)

Only the PPMTR can issue this function request. It is issued to interrupt an executing user mode program so that CPMTR can reschedule the use of CPUs.

M.RCH REQUEST CHANNEL RESERVATION
(0012,BBAA,DDCC,****,RRRR)

AA 1st choice channel number
BB 2nd choice channel number
CC 3rd choice channel number
DD 4th choice channel number

RRRR	=	0000	Request immediate reply
RRRR	=	0000	No reply until a requested channel has been reserved.

When channel zero is requested, it must be field AA. Zero BB, CC, or DD implies no channel request and no alternate choices. If none of the requested channels is available and an immediate reply is requested, PPMTR will set bytes 0 and 4 of the PPU output register to zero. When a channel is granted, the number of that channel will be returned in the PPU output register byte 1 (location of AA). Byte 4 will be set to a non-zero value.

On exit, if a channel has been reserved, the output register appear:

0000 XXXX TTTT TTTT YYYY

XXXX Channel number

TTTT TTTT Information from the channel status word.

YYYY PP input register address

M.MTRCPU is the symbolic limit for CPMTR functions. Its current value is 12; therefore CPMTR will process only those functions whose values are from 1 to 12, inclusive.

CPMTR/PPMTR COMMUNICATIONS

When a user program posts an RA + 1 call, CPMTR picks it up and determines the processing procedure to be followed. When CPMTR determines that the call is for a PP program, it sets up the request in the next available entry in a 4-word circular buffer, T.MTRRS, known as the monitor request stack and used specifically for CP to PP monitor communications. Any PPMTR function request generated by CM system programs would also be placed in the buffer. The monitor request stack is located adjacent to the exchange package for CP resident programs.

PPMTR

The PP Monitor is in general control of the system. It is loaded into PP0 at deadstart and remains there for the duration of system execution. Primarily, PPMTR controls and co-ordinates system activities to avoid conflicts between various system processors. It allocates peripheral processors, central memory and ECS to the control points; in addition, PPMTR maintains the system clock.

MAIN LOOP

As soon as PPMTR has been loaded into PP0 and initialized, it begins execution of the main loop, consisting of several functions executed on each pass, plus a group of lesser functions, one of which is executed on each pass. The primary functions performed in the main loop are:

Scan the 4-word monitor request stack (T.MTRRS) for a PPMTR function call or a PP program request.

Scan the PP input register (T.PPIP) for a PPMTR function request.

After completing the primary main loop functions, PPMTR will process a slower function taken from a list called DSTRING (dispersal string). Each entry in the list is not necessarily unique: for instance, the advance clock (AVC) routine must be entered once every four milliseconds, hence it will appear five times in the list.

The DSTRING list contains the following functions:

AVC	Advance Clock
ACP	Advance Control Point
PPX	Scan PP Output Registers
CHCIO	Check CP.CIO Output Register

SYSP	System Program Advancement
PDS	Process Delay Stack
PEZ	Test P-register for Zero
CHR	Channel Rejection History
PPM	Process Monitor Request

Advance Clock Routine (AVC)

The function of this routine is to maintain the real time and display code clocks stored in the CMR pointer area at T.MSC and T.CLK, respectively. The real time clock specifies time in increments of 244 microseconds (one 4096th of a second); the other specifies time in hours, minutes and seconds in display code. The AVC routine must be entered every four milliseconds. It interrupts the CPU at the end of every time slice and accumulates idle PP time for use by the Scheduler. If the time limit set for the job has been exceeded, MTR will set the error flag F.ERTL and the control point is put in drop status.

CPU time is accumulated by a hardware clock used by the computer to time hardware instructions and which is connected to channel 14 (octal). There is no entry for channel 14 in the channel status table because more than one PP can read the clock at the same time. Therefore, it is not necessary to have an interlock and to reserve the channel.

Advance Control Point Routine (ACP)

On each pass through the main loop, MTR will analyse the status and activity of one control point by checking for the following conditions:

If the control point is in drop status and no activity remains, 1AJ is called to the control point to process the next control card.

If the control point is in periodic recall (X status set in W.CPSTAT), MTR will decrement the recall delay count associated with the control point. When zero is reached, ACP recalls the control point and exits.

If the control point is in auto-recall and there is no activity at the control point (Y status set in W.CPSTAT), the control point is hung in auto-recall since there is no way the completion bit can be set. MTR will set the error flag F.ERHANG and put the control point in drop status.

If there is activity while in auto-recall, or there is a request in the PPMTR request stack, ACP attempts to restart the control point; ACP exits if there is no auto-recall and there is activity at the control point or an outstanding stack request exists.

SCAN PP OUTPUT REGISTERS (PPX)

In this routine, the PP output registers are scanned for a monitor request made by a PP. This is an indexed scan of active PPs which are linked through APLINK. When PPMTR assigns a PP to process a request, the

routine requested and the PP input register number are entered in the CMR word T.PPIP. This routine enables a request to be processed without its address being entered into T.PPIP.

CHECK CP.CIO OUTPUT REGISTER (CHCIO)

This routine checks the output register T.CPIR of CP.CIO. If a MTR request is present, PPMTR will have it processed.

SYSTEM PROGRAM ADVANCEMENT (SYSP)

This routine scans the event stack and if a request for the Scheduler has been made, it is initiated.

P-REGISTER EQUALS ZERO TEST (PEZ)

When location counter (P register) for a user program is zero, an arithmetic error has been detected and error flag F.ERAR is set by MTR.

Process Delay Stack Routine (PDS)

A PP job is put into the PP job queue when MTR cannot find an available PP into which the program can be loaded. Programs in the job queue are linked together in the same order as they were put into the queue; so that the first program in the chain has been in the job queue the longest. If a job is rolled out when its delay expires, it is re-entered into the delay stack with a new delay period assigned.

When a PP program issues a request to load a PP program after a delay of a certain time period, PPMTR inserts an entry into the PP delay stack. On each pass through the main loop, PPMTR scans the queue of PP job requests issued with a time delay and initiates the PP jobs whose delay has expired. Physically, the PP delay stack, the event stack and the PP job queue are treated as one table. Entries in the delay stack are chained together in order of increasing delay time, and the chain is linked to the end of the PP job queue. When the delay time of an entry in the delay stack becomes zero, it is considered to be part of the PP job queue.

CHANNEL REJECTION HISTORY (CHR)

This routine is the last in the DSTRING list. It updates the record of rejected channels, initializes the channel request register and resets the dispersal index for the next pass through the DSTRING.

PPMTR Function Request Processing (PPM)

All PP requests are made by placing the request number (M.xxxx) in byte zero of the PP output register. Parameters for the various requests are supplied to MTR through bytes 1 to 4 of the output register and/or the PP message buffer. (See figure 3-6.) Upon completion of the request, MTR replies by setting byte zero of the PP output register to zero and passes the response to the requesting PP through the remaining bytes of the output register or through the message buffer.

If the format of the request is incorrect, MTR sets the high-order bit of byte zero in the output register of the requesting PP. As MTR will ignore any PP request with this bit set, the requesting PPU is hung. The following message

ppn NAMcp BAD MTR REQUEST

is inserted into the MTR message buffer to be flashed at the bottom of the right screen.

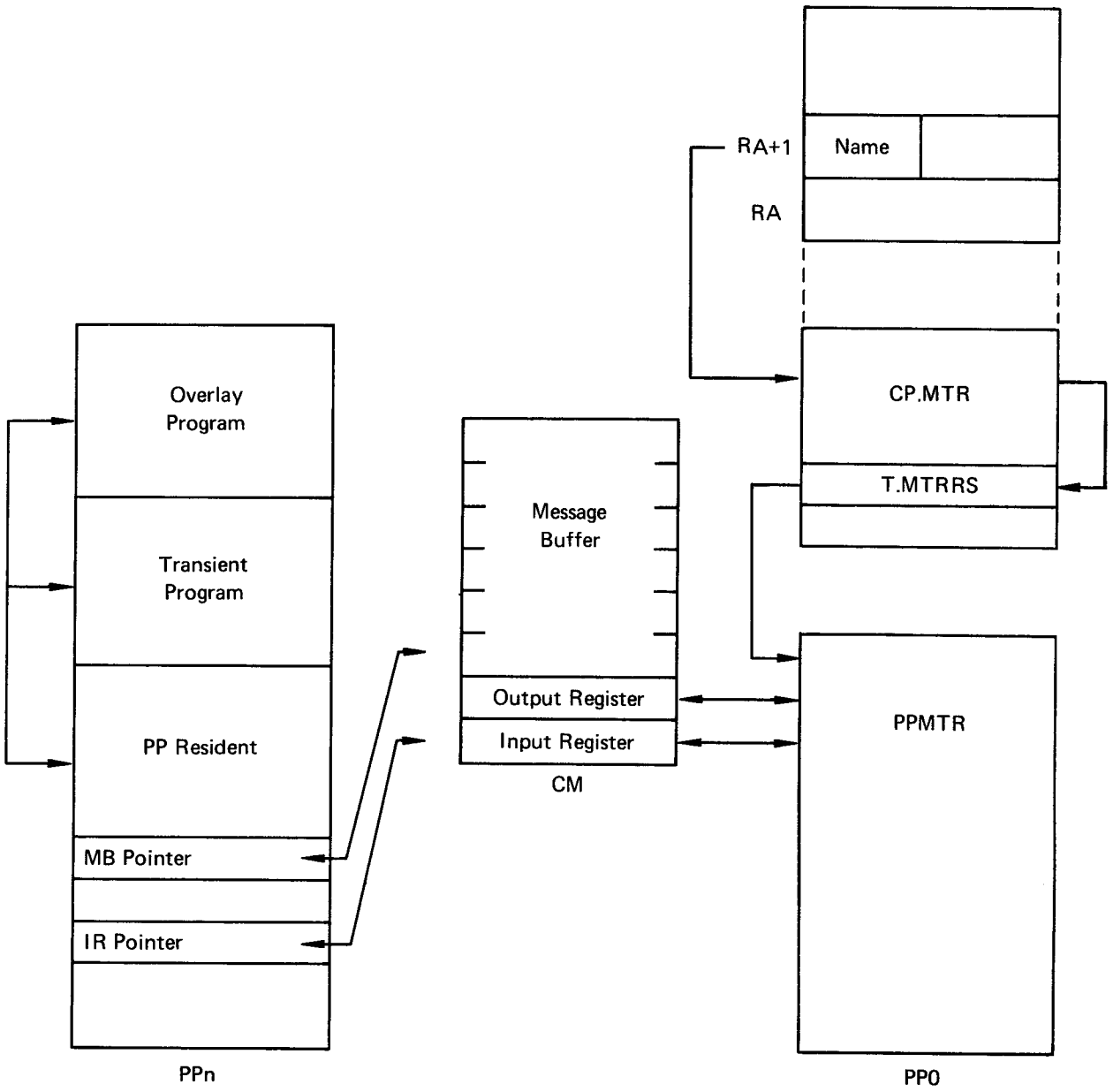


Figure 3-6. Monitor Request Processing

PP MTR Function Codes and Mnemonics

13	M.DFM	Dayfile message
14		Not used
15	M.STEP	Enter step mode
16	M.RBTSTO	Reduce RBT storage
17	M.RSTOR	Request storage
20	M.TSR	Terminate storage request (IP.RTMTR $\neq$ 0)
21	M.DPP	Drop PP
22	M.ABORT	Abort control point and drop PP
23	M.REQP	Request equipment
24	M.DEQP	Drop equipment
25	M.SEQ	Assign job sequence number
26	M.SEF	Set error flag
27	M.ISP	Initiate Stack Processor
30	M.SPRCL	Stack Processor recall
31	M.CCPA	Change control point assignment
32	M.RPJ	Request peripheral job
33	M.EES	Enter event stack
34	M.CPJ	Capture peripheral job
35	M.SCH	Initiate integrated scheduler
36	M.PASS	MTR ignores it
37	M.RACT	Request control point activity
40	M.RPP	Request PP
41	M.NTIME	Enter new time limit
42	M.PPTIME	Assign PP time
43	M.PPCH	CPMTR requests a channel
44	M.BUFPTR	Buffer pointer address
45	M.PATCH	Enter a patch into MTR
46	M.TRACE	Turn on MTR trace mode
47	M.SLPER	Set new time slice period
77	M.KILL	A bad MTR request made
400	M.INPROC	PPMTR is processing a function

The request formats are described below; function parameters are given for each request. Asterisks denote bits which are irrelevant to the function. Requests are listed alphabetically by SCPTEXT mnemonic.

M.ABORT ABORT CONTROL POINT AND DROP PP
(0022,****,****,****,****)

The job at the requesting PPU is terminated. The requesting processor is responsible for the dayfile message. Operation of this function is identical with function M.DPP except that the error flag in the control point area is set to F.ERPP(3) to note the abort function.

M.BUFPTR WATCH BUFFER POINTER WORD
(0044,****,****,00AA,AAAA)

AAAAAA	Buffer pointer address.

I/O drivers use this function to give MTR the absolute address of the buffer pointer that is being updated. MTR monitors the value of that pointer and when it changes, restarts the control point if it is in periodic recall.

M.CCPA CHANGE CONTROL POINT ASSIGNMENT
(0031,****,****,****,***N)

The requesting PPU is released from its current control point assignment in the same as though it had issued an M.DPP function, but its input register is not cleared. The PPU is assigned to control point N, and the new control point number inserted in its input register.

M.CPJ CAPTURE PERIPHERAL JOB
(0034,00XX,XXXX,****,****)

XXXXXX	Buffer address, relative to RA

This request is issued to find a job for a control point either in the event stack or in the PP delay stack. The event stack is searched first; if a job is found, its data is written to the buffer whose address is given in the request. When the end of the delay stack is reached, the function is exited.

M.DEQP DROP EQUIPMENT
(0024,EEEE,****,****,***N)

PPMTR drops equipment ordinal EEEE from the control point and updates the EST entry to indicate this equipment is free for reassignment. PPMTR does not check to ensure that the dropped equipment was assigned to this control point. The parameter N gives the control point number to be considered if the requesting PP is attached to control point zero, otherwise it is irrelevant.

M.DFM PROCESS DAYFILE MESSAGE
(0013,FFFF,MMMM,****,****)

Dayfile flag bits FFFF determine message handling:

0	Do not send to B display
1	Do not send to control point dayfile
2	Do not send to system dayfile (no A display)
3	Flag as an accounting message
4	Send to hardware error file
5	Do not insert job name in system dayfile

When the value of MMMM is greater than that in PPOR, it is taken to be the LWA + 1 of the message in the PP message buffer. When the value is less than that in PPOR, MMMM is taken to be a dump index for a requested dayfile dump.

Value of dayfile dump index:

0	System dayfile dump
1 thru N.CP	Control point dayfile dump
N.CP + 1	Hardware error file dump

M.DPP DROP PPY
(0021,FFFF,****,****)

MTR clears the PP control assignment (the PP status word and the PP input are cleared). If the value of FFFF represents M.DPP, the PP time is not incremented.

M.EES ENTER EVENT STACK
(0033,00AA,AAAA,****,SYTT)

AAAAAA	Word address in Event Stack
Y	Byte address in word AAAAAA
TT	Bit address in byte Y
S	Combined value of F and B:

F=0	Event stack job not assigned (F.ESOFF)
F=4	Event stack job assigned (EESON)
B=0	AAAAAA is an absolute address (F.ESABS)
B=1	AAAAAA is relative to RA (F.ESREL)
B=2	AAAAAA is a control point address (F.ESCPA)

The event stack is similar to the delay stack and is used by the Scheduler. M.EES writes to the peripheral job table, the PP input register, and three parameter words. It then sets up the control point number and linkages in the event stack. The new entry is lined to the oldest prior entry then clears the output register and exits. A separate chain is maintained for each control point.

M.INPROC CPMTR FUNCTION IN PROGRESS
(0400,****,****,****,****)

The CPMTR is processing a function; any CP monitor request made by a PP must wait.

M.ISP INITIATE STACK PROCESSOR
(0027,000X,****,****,CCCC)

CCCC	DST ordinal of stack processor to be initiated
X = 0	Initiate 1S5 only if PP active flag is zero
X ≠ 0	Initiate 1S5 regardless of PP active flag setting

A check is made to see if a PP is assigned to this DST ordinal. If there is none, a check for an available PP is made. If an available one is found, set the PP active flag and place the DST ordinal and 1S5 in its input register. If a PP is found to be assigned to the DST ordinal, the setting of X determines if 1S5 is to be initiated or not. If not, the output register is cleared and an exit made; if yes, proceed as for an available PP. If the PP job stack is full, the routine is exited. The 1S5 program is the PP input register DST ordinal checker and stack processor loader.

M.KILL BAD FUNCTION REQUEST
(0077)

MTR flags the function request as bad and automatically enters STEP 0 mode. The requesting PP is hung.

M.NTIME ENTER NEW TIME LIMIT
(0041,TTTT,T***,****,***N)

A central processor job time limit of TTTTT seconds is entered at the control point. Any previous time limit is superseded. If the requesting PPU is assigned to control point zero, the parameter N will give the number of the control point to be considered; in any other case this parameter is irrelevant.

M.PASS PPMTR IGNORES FUNCTION REQUEST
(0036,****,****,****,****)

Indicates a no-operation by PPMTR which will be cleared by another routine.

M.PATCH INSERT A PATCH IN PPMTR
(0045,AAAA,BBBB,CCCC,DDDD)

The routine inserts a patch in the monitor program at the address indicated.

AAAA	Insertion address for patch BBBB
CCCC	Insertion address for patch DDDD

M.PPCH REQUEST CHANNEL SURVEILLANCE
(0043,BBAA,DDCC,****,RRRR)

This routine is a PP extension of the CPMTR function M.RCH (request channel reservation). When CPMTR processes a channel request and the request is rejected, the M.RCH function is changed to M.PPCH so that a watch can be maintained until the desired channel is available. PPMTR will update the channel reject history (DSTRING function CHR) and when the channel is available, change M.PPCH to M.RCH and initiate CPMTR.

M.PPTIME ASSIGN PPU TIME
(0042,FFFF,****,****,****)

MTR adds the current time, minus the PPU starting time, to the accumulated PPU time in the control point area (word W.PPTIME). If the value of FFFF represents M.PPTIME, the PP time is not incremented.

M.RACT REQUEST CONTROL POINT ACTIVITY
(0037,***N,IIII,****,****)

This request provides the various activity counts of control point N at a given time (N cannot be zero). If IIII is non-zero, the pseudo-activity count will be incremented or decremented by the constant IIII (after sign extension). Monitor replies through the PP output register:

Byte 1	Control point status byte
Byte 2	General activity count
Byte 3	Count of outstanding delayed PP requests
Byte 4	Pseudo-activity count

M.RBTSTO REDUCE RBT STORAGE
(0016,SSSS,****,****,****)

PPMTR sets SSSS*100 as the new RBT starting address.

M.REQP REQUEST EQUIPMENT
(0023,EEEE,****,****,***N)

The parameter EEEE consists of two display-code characters:

if numeric it gives an EST ordinal,
if alphabetic it defines an equipment type.

PPMTR will search the equipment status table (EST) for the EST entry. This entry is updated to reflect the assignment to the control point of the requesting PPU or to control point N if the PPU is assigned to control point zero. Finally, PPMTR places the assigned euqipment ordinal in the first byte of the PPU message buffer. If the equipment is not available, a zero byte is returned.

M.RPJ REQUEST PERIPHERAL JOB
(0032,DDDD,DDDD,****,****)

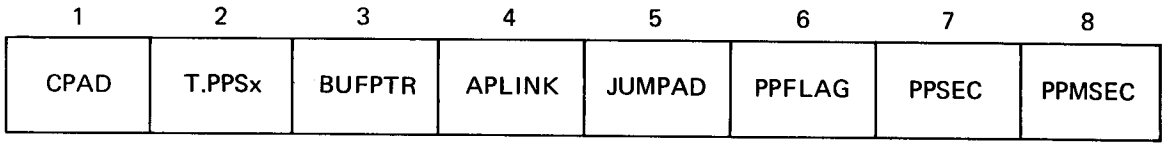
This function requests that another PPU program be initiated after a specified time delay. The first word of the requesting PPU message buffer contains the input register image of the new PPU program. The time delay is DDDDDDDD*244 (decimal) microseconds. If the time delay is zero and no PPU is available, the request is entered in the PP job queue. If no space is available in the PP job queue buffer of PPMTR, the entire request remains pending until a queue entry becomes free.

M.RPP REQUEST PPU
(0040,****,****,****,****)

This function requests immediate initiation of another PPU program. The first word of the requesting PPU message buffer contains the input register image of the new PPU including the control point number to which it should be assigned. The input register address of the assigned PPU is placed in the first byte of the requesting PPU message buffer. A zero byte is returned and the request is rejected if no PPU is currently available.

PPU ASSIGNMENT

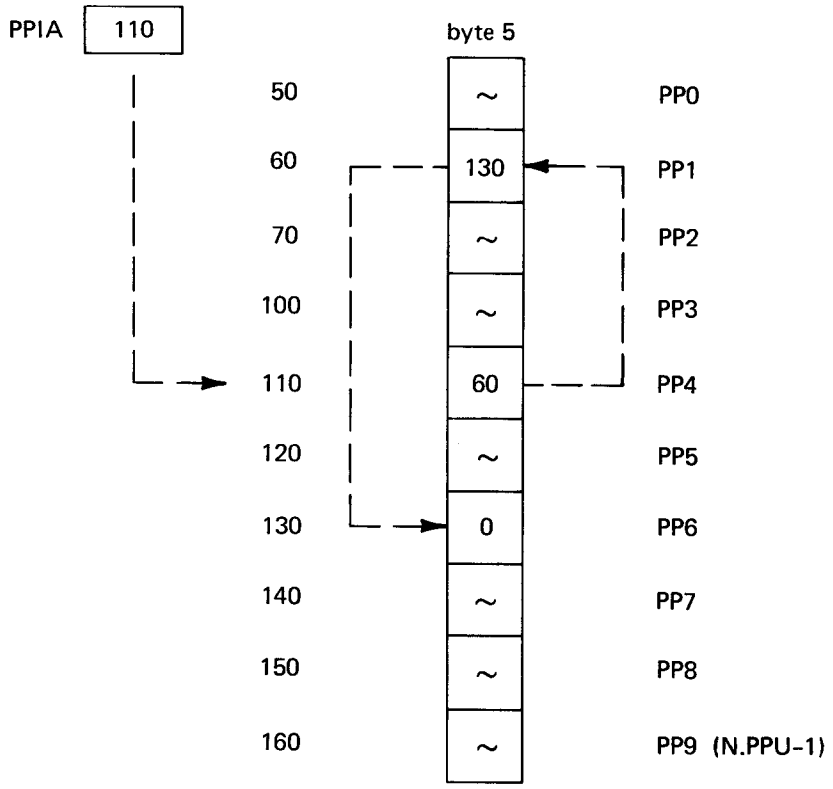
To control PP assignments, PPMTR keeps (in PP zero) an 8-byte status word for the PPU entries which form the PPU Status Table. Each status word has one of two formats, depending upon whether the PP is assigned or unassigned and available.



CPAD	Base address of control point area to which PP is assigned
T.PPSx	CM address of PPU status word
BUFPTR	The low order twelve bits of the buffer pointer the last time that MTR looked at it. (See M.BUFPTR)
APLINK	Active PPU link. This is a pointer to the next member in a chain of active PPUs. The chain always starts with MTR (PP0) and ends with DSD (PP1). The next PPU is identified by its output register index value.
JUMPAD	This is the address saved for re-entry to a partially completed monitor function that has been exited via an RJM MAINLOOP.
PPFLAG	Flag is set when PP contains a stack processor
PPSEC	PP starting time in seconds
PPMSEC	PP starting time in milliseconds

Bytes 1 through 5 form the word written into location T.PPSx when PPx is assigned.

When the PP is unassigned and available, the PPU status word is linked in a chain of unassigned and available PPs, using byte 5 of the status word (PPFLAG). PP direct cell PPIA contains a pointer to the status word at the head of the chain. Byte 5 contains a pointer to the status word of the next available PP. If no more PPs are available, byte 5 contains zero. The following diagram illustrates a chain of three available PPs.



An unassigned PP may exist which is not linked in the available PP chain. This possibility occurs when no PP is assigned to a stack processor (PPFLAG not set) and PPMTR reserves a PP for 1SP.

PPMTR assigns a PP by writing a peripheral job name and a control point number into the PP's input register to:

Satisfy a PP program call issued as an RA+1 request

Answer a PP request for another PP job (M.EES, M.RPP or M.RPJ request)

Initiate a stack processor when an I/O request is issued for a mass storage device to which no stack processor is currently assigned

Call the PP program 1AJ to a control point when all control point activity has ceased

PPMTR maintains a PP queue table containing a maximum of 40 entries, each 4 bytes long. Each entry corresponds to a 4 word entry in the Peripheral Job Table (PJT) in CMR. In the queue, four chains are kept:

A queue of PP jobs which cannot be currently initiated because PPs are not available. This overload queue is a chain of PP input register images.

A queue of PP jobs which must be initiated after a given time delay. This queue is a time-ordered chain of PP input register images, called the delay stack.

A separate queue of PP jobs for each control point which must be initiated after a specified bit has been set or cleared in central memory. This queue is called the event stack.

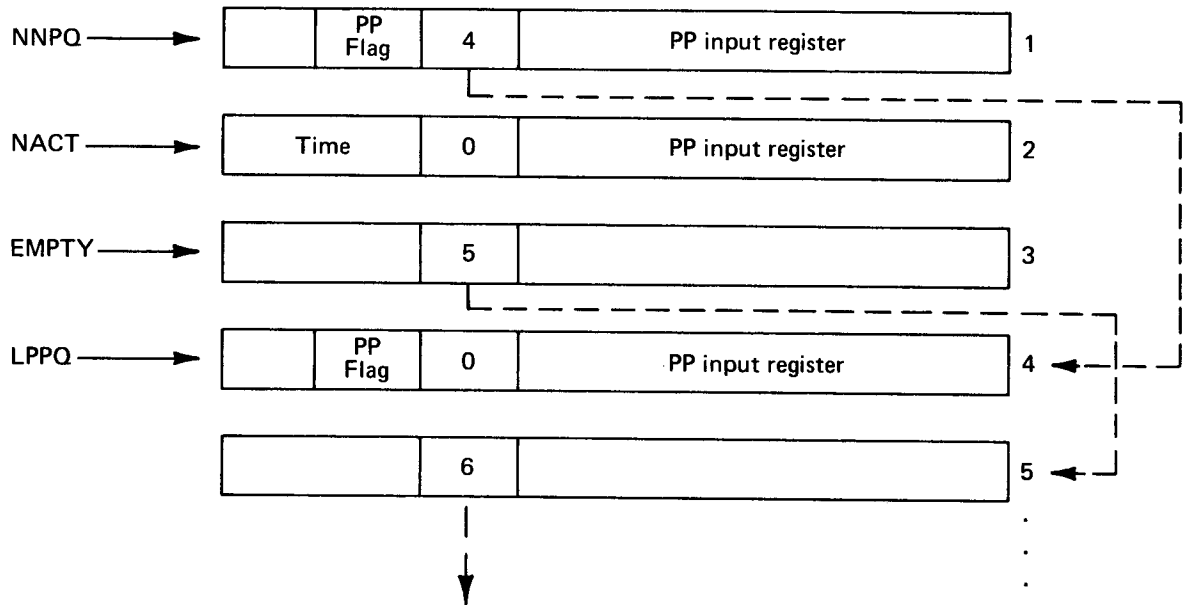
An empty queue of all unused entries in the PP queue.

Three pointers in PPMTR define the beginning of each chain:

NPPQ	PP overload queue
NACT	PP delay stack
EMPTY	Empty chain

The pointers to the event stack are in the control point table EVST. A fourth pointer, LPPQ, defines the end of the PP overload queue. When the time delay expires for an entry in the delay stack, that entry will be transferred to the end of the overload queue.

Chaining of the queue entries uses byte 3 of each 4-byte entry. Bytes 1 and 2 contain the PPFLAG or the maturation time for entries in the delay stack. The end of a chain of entries is signaled by zeros in byte 3. The following illustrates a PP job queue containing a two-entry overload queue, a one-entry delay stack, and the empty chain.



M.RSTOR REQUEST STORAGE
(0017,CCCC,XXXX,00TT,****)

CCCC	Requested CM/100 octal	
XXXX	Requested ECS/1000 octal	
TT	00	CM request only
	01	ECS request only
	02	CM and ECS request
	03	Request reserved CM
	04	Request CM—will await response
	06	Request CM and ECS—will await response
	07	IP.POSFL requested by swapper
	20	Priority storage requested

Assign CCCC central memory and/or XXXX extended core storage to the control point of the requesting PPU. Monitor replies to this request by setting CCCC and/or XXXX to the values actually assigned to the control point and by setting byte 0 to zero. These values should be compared with the original values requested to determine whether these requests have been honored or not. A request for more storage is rejected if not enough storage is available or if a storage move is already in progress. A request for less storage always is honored immediately. If TT = 02, PPMTR can honor part of the request without honoring the remainder.

MEMORY ALLOCATION

Central memory and direct-access extended core storage is assigned in the same sequence as control point numbers. Storage associated with any control point is contiguous.

All storage is associated with the assigned control points. If no user jobs are assigned to control points, all storage is associated with the central memory resident, which operates at control point zero. All storage, therefore, falls into one of two categories:

Allocated storage, defined by the RA and FL of the control point.

Unallocated storage, defined as that which occurs between the allocated storage of two consecutive control points. All unallocated storage is associated with the lower numbered control point.

PPMTR maintains two tables in PP zero which contain the size of allocated and unallocated CM and ECS storage associated with each control point (0 through N.CP). A request for reduced field length will have the effect of reducing the amount of allocated storage and transferring the value to unallocated status; no storage is actually moved. When the total storage associated with a control point is sufficient, a request for an increased field length will cause transfer of the increased value from the unallocated to allocated state; no actual storage move takes place.

If there is insufficient storage associated with a control point to satisfy an increased field length request, unallocated storage may be transferred from adjacent control points. A scan of the unallocated storage table entries for the adjacent control points will locate a combination of blocks which results in moving the least amount of storage.

When a control point RA and/or FL is to be changed, PPMTR will suspend the control point by setting the M bit of the CP status byte. If a CM move is necessary, PPMTR will set the storage move flag and wait for all PPs assigned to the control point to pause and then initiates the system exchange package to start the storage move program. After completion of the move, PPMTR will modify the control point RA and/or FL and resume the control point by clearing the M status bit.

In the case of a request for ECS or a change in FL for control point zero, the storage move flag is not set and the wait for PPU pause is not performed. A PPU pause occurs when the field access flag (C.FAF) in the PP status word is zero. If an ECS parity error occurs during an ECS move, PPMTR will abort both the requesting control point and the control point whose ECS is being moved, if it is no longer intact.

M.SCH INITIATE INTEGRATED SCHEDULER
(0035,000X,****,****,****)

If X=2, call the Scheduler, clear the PP output register and exit. Otherwise, check request stack: if full, call the Scheduler to empty it before an entry can be made; if not full, make an entry. In either case, advance the request stack pointer

M.SEF SET ERROR FLAG
(0026,***N,EEEE,****,****)

Monitor will drop the central program at control point N by putting the program in zero status, and setting the error flag to the value EEEE.

M.SEQ ASSIGN JOB SEQUENCE NUMBER
(0025,****,****,****,****)

Monitor returns in byte 1 of the PPU output register a job sequence number (in display code).

M.SPRCL STACK PROCESSOR RECALL
(0030,00AA,AAAA,000F,CCCC)

This function is called with F non-zero when a stack request has been completed to update the exit count of that control point (CPSR), and to update the I/O channel time information (if IOTIME mods are assembled on) using AAAAAA, which is the millisecond count for PP time inserted by the stack processor. If F is zero, only the IO channel time information is updated.

M.STEP MONITOR STEP CONTROL
(0015,****,****,****,***N)

This control is initiated by a keyboard request. MTR sets an internal step control flag in word T.MSP in the CMR pointer area. At each subsequent request, MTR pauses for console keyboard input. A space from the keyboard causes MTR to process the request. A period from the keyboard causes MTR to process the request and clear the step control flag to resume high speed operation. If N = 0, all PPU requests are stepped. If N is non-zero, only control point N is being placed in step mode and only the requests issued by PPUs assigned to control point N will be stepped.

M.TRACE ENTER MONITOR TRACE MODE
(0046,AAAA,FFFF,NNNN,****)

AAAA	Absolute address of buffer within requesting field length of requesting job
FFFF	Length of buffer
NNNN	Pointer to next available word-pair in buffer

A buffer must be provided by the trace mode requestor into which this PPMTR function will write trace records. Each record is a two-word entry containing function and PP status information. This function is reserved for CDC developmental use.

M.TSR TERMINATE STORAGE REQUEST
(0020,****,****,****,****)

Request is valid if real time monitor is installed (IP.RTMTR is non-zero). Terminates wait period involving an M.RSTOR request.

FILES AND FILE TABLES 4

FILES

SCOPE 3.4 supports the following file formats:

Sequential

Random

Direct Access

Indexed Sequential

A name associated with each file identifies it to the system and to the user. Files are stored on either allocatable or non-allocatable devices. Rotating mass storage units, such as disk or drum, as well as ECS are allocatable because files on these devices may be allocated to more than one control point. Other devices, such as magnetic tapes, card readers, punches, family or sequential packs, and line printers, are non-allocatable because they can process only one file at a time.

Files associated with a job running at a control point are assigned or attached to a control point. Files not associated with running jobs are assigned to control point zero.

Files are associated with one of the following groups: system, input, local, permanent, and output. The following paragraphs describe briefly each of the file groups.

SYSTEM FILES

The files described below are always in the system; they are always assigned to control point zero and reside on allocatable devices called system devices. These files, except for the job dayfile, are maintained on system devices as permanent files.

SYSTEM This file has the name of ZZZZZ04 and contains a copy of the deadstart system tape.

DAYFILE The system dayfile contains a complete record of all activity in the system. Normally, when a message is sent to any job dayfile, it also is sent to the system dayfile. At intervals, the system dayfile can be dumped to a line printer, punch, or magnetic tape.

DFILEn Each user control point in the system has a job dayfile; n is the control point number. These files are assigned to control point zero and a user cannot access them directly. When a user job terminates, the contents of the job dayfile is copied to the end of the job output file. A job dayfile contains images of all control cards processed, appropriate system messages concerning the job run, plus messages sent to the dayfile by the job.

CERFILE If hardware errors are discovered by running programs, a message is written to this file. Periodically, the file is dumped for examination by customer engineers so they can take remedial action.

ZZZZZ06 If the installation parameter IP.ELIB is one, this file is created to contain the ECS library.

ZZZZZ23 This file contains a copy of the CM resident library area. It is created by Deadstart and is updated by EDITLIB. It is not permanently attached to control point zero and is used only for deadstart recovery.

INPUT FILES

Jobs may enter the system from sources such as card reader, magnetic tape, or remote devices. In every case, a job file is read by a system package operating at a control point which then writes the job to a local file on an allocatable device. When the file has been written, its entry in the file name table is altered so that it is released to control point zero as an input file.

The system packages that read in batched local jobs ensure that each job file in the system has a unique 7-character name. The job name from the job card is truncated to the first five characters (or extended with zeros to five characters) and up to two unique sequence characters are added. All numerals and letters may be used as sequence characters, therefore 1,296 sequence combinations are possible for a single five-character job name. Even though unique combinations are exhausted, duplication of names is not significant unless the earlier job has not been completely processed when the duplicate enters the system.

LOCAL FILES

Any files, other than permanent files, attached to a job running at a control point are local files. They may be on allocatable or non-allocatable devices.

Local files assigned to a control point must have unique names. Two local files named INPUT and OUTPUT are associated with each job. INPUT contains the job file; the job name is changed to the name INPUT when the job is assigned to a control point. OUTPUT is assigned to the job when the first reference to it occurs. It has a disposition code which indicates the job output is to be produced on peripheral devices in the area from which the user submitted the job.

When a local file is detached from a control point, disposition depends on its name, control point, disposition code, and device type on which it resides. For local files on non-allocatable devices, the device and the related table space will be released. Assigned storage and related table space is released for local files on allocatable devices, other than private disk packs, having a zero disposition code (except special name files PUNCH, PUNCHB, FILMPR, FILMPL, HARDPL, HARDPR, PLOT, and OUTPUT). For local files with other disposition codes the file name is changed to job name and the file is assigned to control point zero. A local file with the name PUNCH or PUNCHB will be output to a card punch; the file OUTPUT is printed.

PERMANENT FILES

Permanent files are saved across deadstarts and are therefore considered to be permanent to the system. Controls over file access and mode of use are provided to define various degrees of privacy. When a permanent file is created, the privacy defined determines which user can access it and the kind of processing allowed.

OUTPUT FILES

Output files originate from local files on allocatable devices; they have non-zero disposition codes or special names, such as PUNCH and PUNCHB. When a job terminates, such files are assigned to control point zero and given the job name as a file name. These files then form a system output queue which is, essentially, a list of files waiting to be outputted to unit record equipment.

In each output queue entry a field contains the file disposition code. For example, disposition code 40 specifies the file is to be printed on any line printer by JANUS. Octal code 10 specifies the file is to be punched in BCD format. Code 12 specifies punching with a binary format.

Local files can be put into the output queue as follows:

The user can give the file a special name. When a job terminates, the file named OUTPUT is always put into the output queue with print disposition code. A file named PUNCHB automatically will be put into the output queue with a disposition of 12 (punch binary). A file named PUNCH receives a disposition code of 10 (punch BCD).

The user can specify file disposition on a DISPOSE control card or DIVERT command (INTERCOM). The file can have any name. Files on allocatable devices with non-zero disposition are put into the output queue.

Files in the output queue must be on allocatable devices. A file is put into the output queue when the job terminates or when a CLOSE, UNLOAD or RETURN is performed. Since the name of an output queue file is the name of the job which created it, and since a job may create several files which will go into the output queue, names in the output queue often are not unique.

SCOPE I/O TABLES

SCOPE coordinates the requirements of input/output files with the status of input/output devices. File tables and device tables are updated continually to provide interface for user jobs and system programs.

FILE TABLES

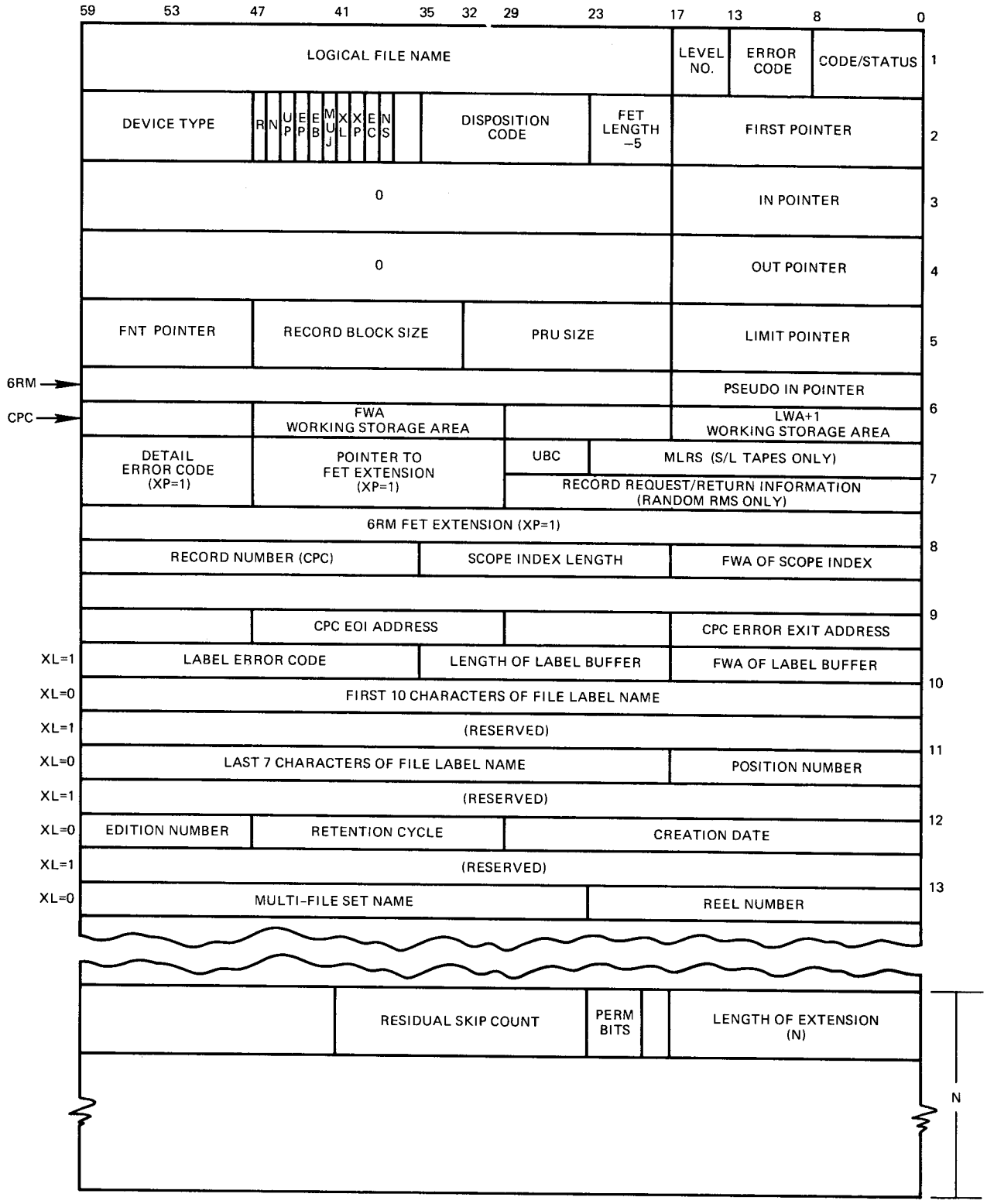
The status and requirements of files is kept in the following tables: File Environment Table (FET), File Name and File Status Table (FNT/FST), File Information Table (FIT), and Record Block Table (RBT). The FET and FIT are created within the job field length, the other tables reside in central memory resident. All are in the upper table area of CMR, except the RBT which resides in the highest addresses of central memory.

The *File Name Table/File Status Table* links all user programs and I/O processing routines. A *Record Block Table* contains chains of record blocks for each file assigned to an allocatable (mass storage) device. The RBT description is included under Device Tables in this chapter.

FILE ENVIRONMENT TABLE

Every file for which I/O is to be performed must have an FET. Each consists of a basic 5-word entry, plus additional words depending upon the manner in which the file I/O is to be performed.

FILE ENVIRONMENT TABLE



LOCAL FILE NAMES
FET Word 1 (bits 59-18)

The following list represents the reserved local file names that appear in a FET for the named 3.4 product set file.

NAME	3.4 PRODUCT SET	USED FOR
ZZZZZ01	EDITLIB	Reset file (permanent file)
ZZZZZ02	EDITLIB	Restore file (permanent file)
ZZZZZ03	EDITLIB	EDITEXTEND file (permanent file)
ZZZZZ04	EDITLIB	System file (permanent file)
ZZZZZ05	EDITLIB	Interpreted EDITLIB directives file
ZZZZZ06	EDITLIB	ECS resident routines library file
ZZZZZ07	EDITLIB	Entry point name table spill file
ZZZZZ08	EDITLIB	Entry point name table spill file
ZZZZZ09	DIAXNOS	
ZZZZZ10	EDITLIB	Program name table spill file
ZZZZZ11		External reference table spill file
ZZZZZ12		External reference collection spill file
ZZZZZ13		Library or deadstart program collection
ZZZZZ14		Scratch
ZZZZZ15		PP program name table spill file
ZZZZZ16		Library name table spill file
ZZZZZIN	PAGE Utility	
ZZZZZOU		
ZZZZZRN		

ZZZZZ17	LOADER	Entry point list
ZZZZZ18	LOADER	Owned global blocks
ZZZZZ19	FORM	
ZZZZZ20	FORM	
ZZZZZ21	FORM	
ZZZZZDF	6RM	
ZZZZZQU	QUERY/UPDATE	
ZZZZZ22	6RM	
ZZZZZ23	EDITLIB	Current directory file (permanent file)
ZZZZZRL ZZZZZOP ZZZZZRM	FTN 4.0	
ZZZZZ24	QUERY/UPDATE	
ZZZZZ25	LOADER	Global library set
ZZZZZ26	GRAPHICS	
ZZZZZ6D	6RM Diagnostics	
ZZZZZ27	LOADER	Overlay generator
ZZZZECS	EDITLIB	System ECS resident library creation job (permanent file)
ZZZZZ28	Debugging Aids	
ZZZZZ31 ZZZZZ32	LOADER	SEGLOAD sort file
ZZZZZRE		Restricted passwords
ZZZZZUN		Unrestricted passwords
ZZZZZSA ZZZZZSB ZZZZZSC ZZZZZSD	SIFT	

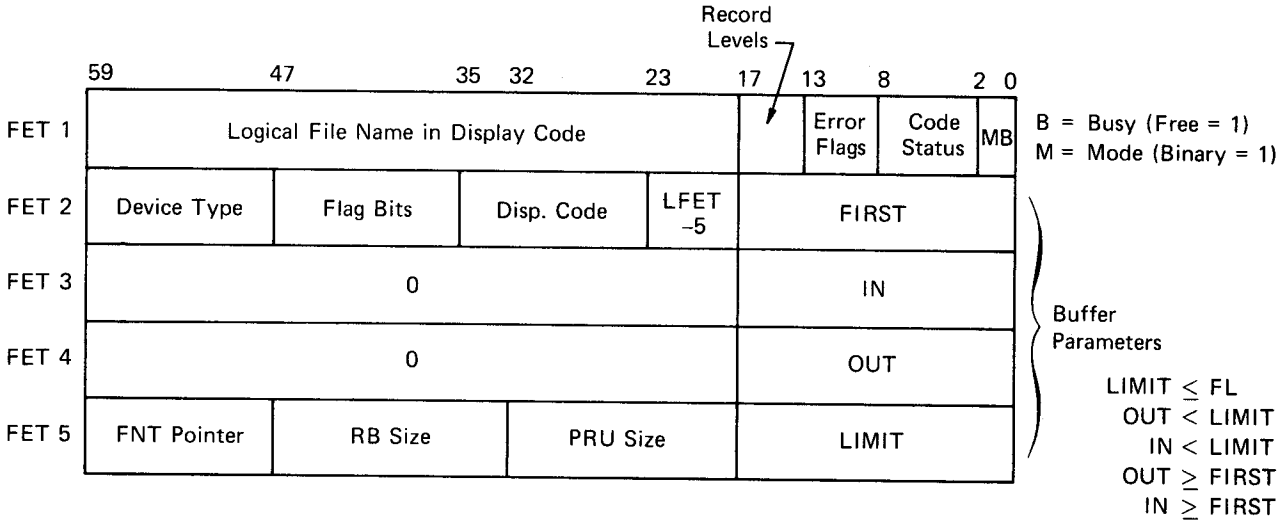
ERROR FLAG LIST
FET Word 1 (bits 9-13)

01	END-OF-INFORMATION
02	END-OF-REEL
04	PARITY ERROR
10	DEVICE CAPACITY EXCEEDED
20	ADDITIONAL ERROR STATUS RETURNED
21	END OF MULTI-FILE SET
22	FATAL ERROR
23	INDEX BUFFER FULL
24	RESERVED
25	INDEX FULL ON RANDOM READ/WRITE OF RECORD n
26	NONEXISTENT RECORD NAMED ON RANDOM READ
27	NONEEXISTENT RECORD NAMED ON RANDOM WRITE AND INDEX IS FULL
30	FUNCTION UNDEFINED ON DEVICE
31	PERMISSION NOT GRANTED
32	FUNCTION ILLEGAL ON PERMANENT FILE
33	RESERVED
34	RESERVED
35	RESERVED
36	RESERVED
37	RESERVED

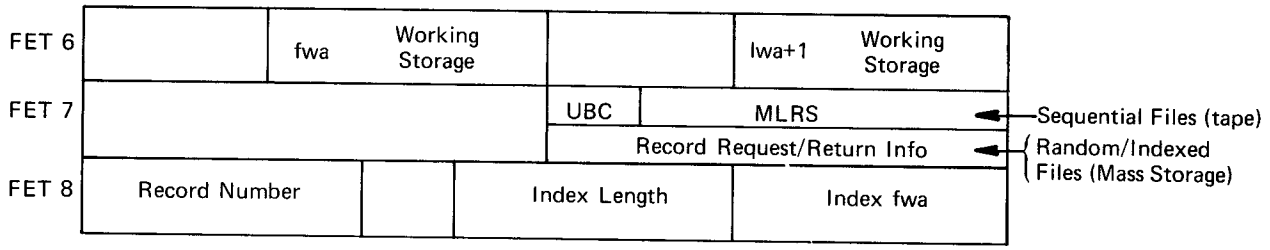
FET FLAG BITS
Word 2 (bits 47-36)

BIT 47	(R)	Process SCOPE index if OPEN/CLOSE; else random read/write
BIT 46	(N)	Release record block as read
BIT 45	(UP)	User processing at end-of-volume
BIT 44	(EP)	User processing on error condition
BIT 43	(EB)	Reserved
BIT 42	(MUJ)	Multi-user job
BIT 41	(XL)	Extended label processing
BIT 40	(XP)	Extended error processing
BIT 39	(EC)	File buffered through or resident in ECS
BIT 38	(NS)	File has non-standard labels; processing of label record(s) is left to the user.
BIT 37		Reserved
BIT 36		Reserved

The basic 5-word FET entry is as follows:



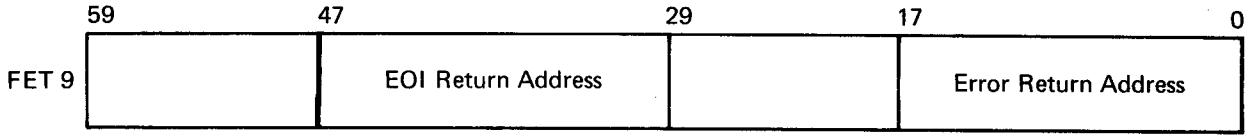
LFET –5 Length of FET minus 5 words for basic entry



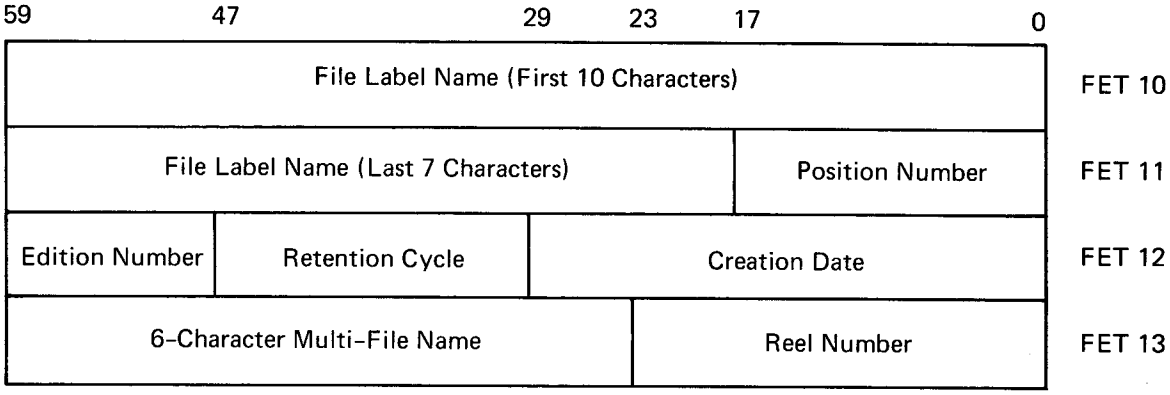
FET6 is used for input/output blocking/deblocking by CPC
FET6 and FET7 are used for S and L tape file processing
FET7 and FET8 are used for indexed file processing by CPC; FET7 is used to pass RMS address between CP programs and system PP input/output routines.

UBC	unused bit count
MLRS	maximum logical record size (S/L tapes only)

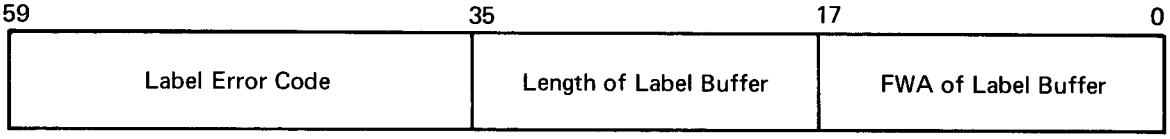
When the UP and/or EP flags are set in FET2, then FET9 contains:



When standard file labels are to be written, the following FET words are filled with information from the LABEL control card or macro. When a labeled file is read, the fields will contain data read from the label.

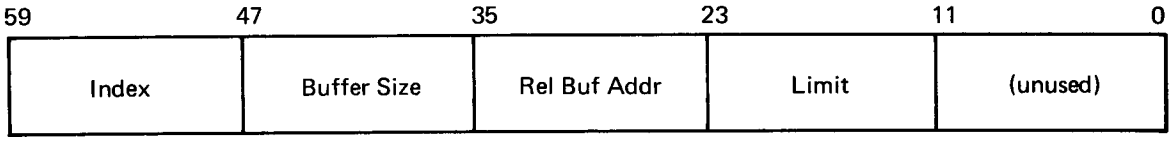


When XL flag in FET2 is set to 1 for extended label processing, FET10 has the following format:



SYSTEM FET ENTRIES

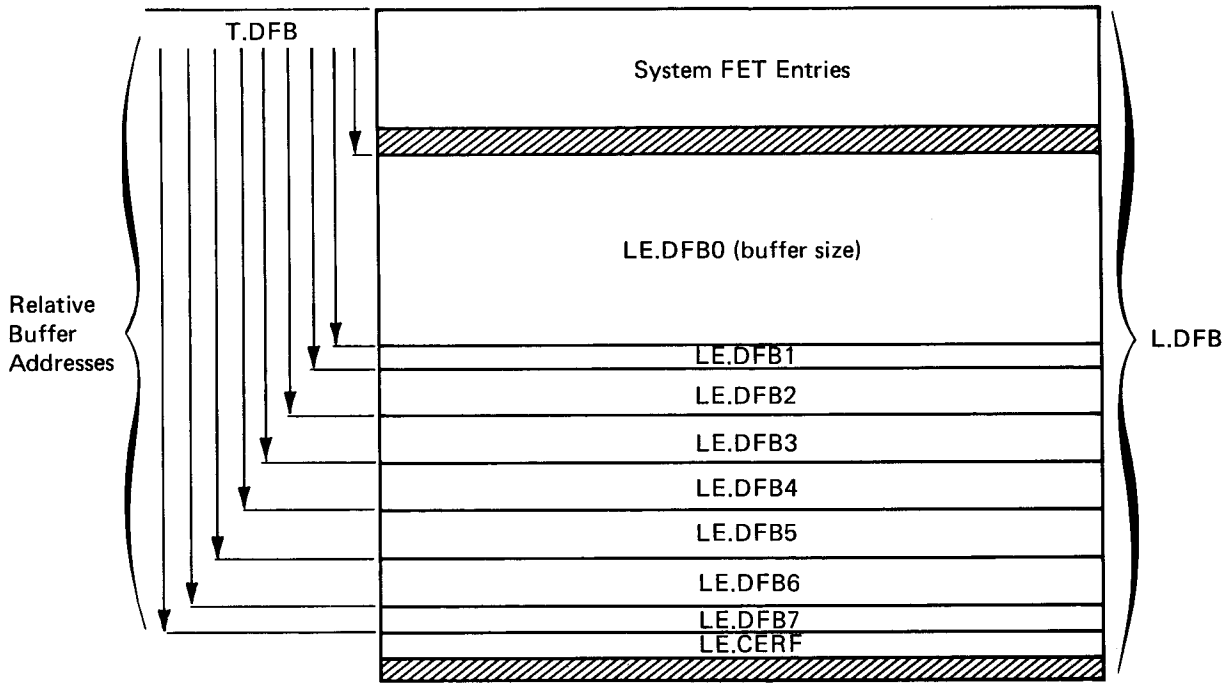
FET entries for the system dayfile, the hardware error file, and the control point dayfiles are kept in the upper table area of CMR, adjacent to the control point zero dayfile buffer. The format of the one-word dayfile FET entries is:



Index is the entry position relative to the table origin (T.DFB). Buffer sizes are set by SCOPE assembly configuration parameters internal to CMR. The origin of each buffer is found by adding the relative buffer address to the T.DFB origin address of the dayfile buffer area in CMR. The CP zero dayfile buffer is preset with a legend in the first eight words as follows: (b represents blank character)

```
bDAYFILE::
bbbNORMALb
(bbbbbbbb)
DEADbSTART
:::::::::::
COPYRIGHTb
CONTROLbDA
TAbCORP.bl
970:::::::
```

The system dayfile area in CMR is diagrammed below:



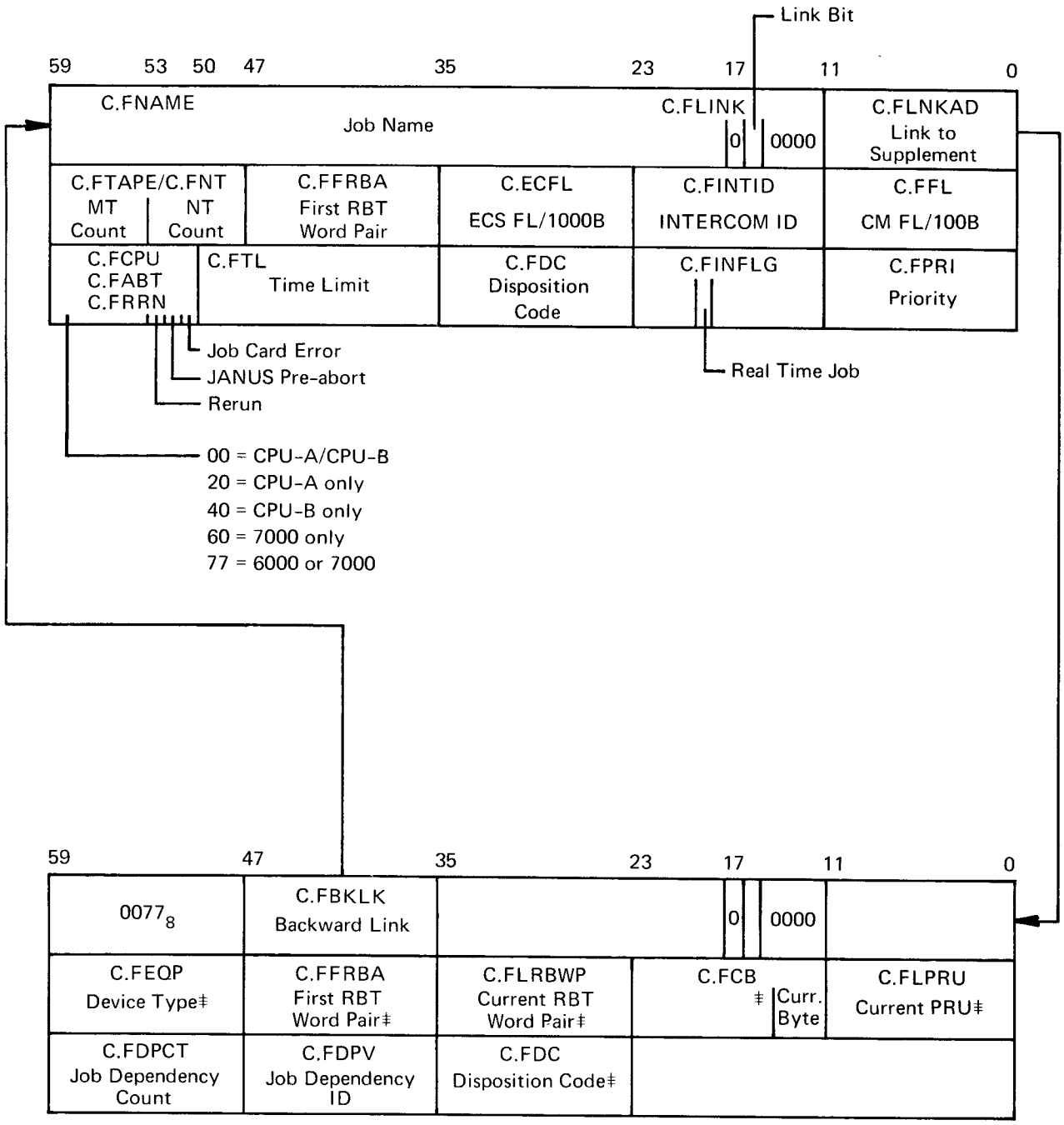
FILE NAME TABLE/ FILE STATUS TABLE (FNT/FST)

To provide linkage between user programs and all I/O processing routines, SCOPE maintains the FNT/FST in CMR upper table area. Each basic entry in the file name table consists of three words; one or two three-word extensions to entries may occur in some instances, extending the entry to 6 or 9 words in length. The first word contains the logical file name, control point number to which it is assigned, as well as other pertinent information. The second and third words constitute the file status table entry; the format of each differs, depending upon the type of file and the equipment residence.

File status entries for each equipment type are as follows. (The C.x(byte) identifiers are listed in the diagrams.)

FILE NAME TABLE

ENTRY AND OPTIONAL SUPPLEMENT FOR FILE IN INPUT QUEUE

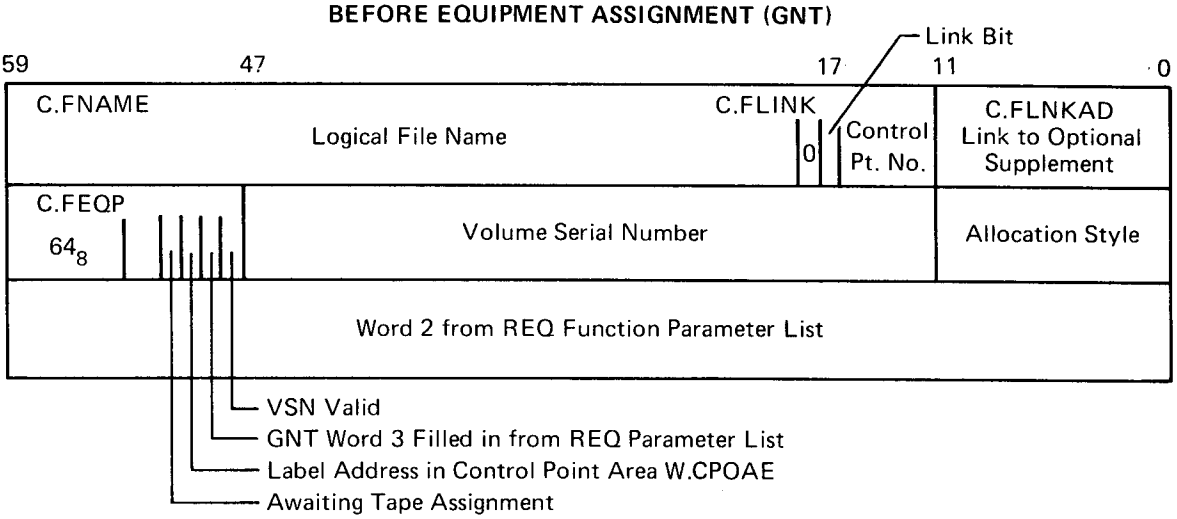


‡If nonzero, fields apply to pre-OUTPUT file

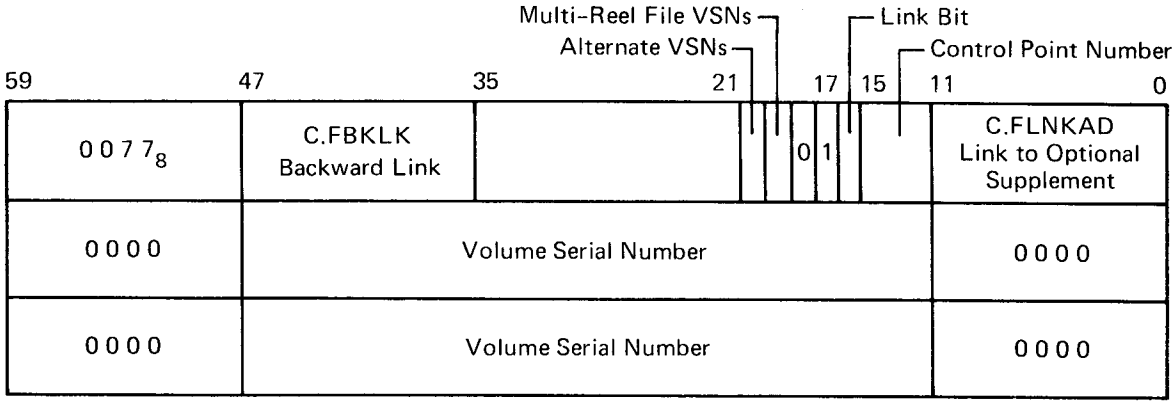
FILE NAME TABLE

TAPE FILE ENTRIES

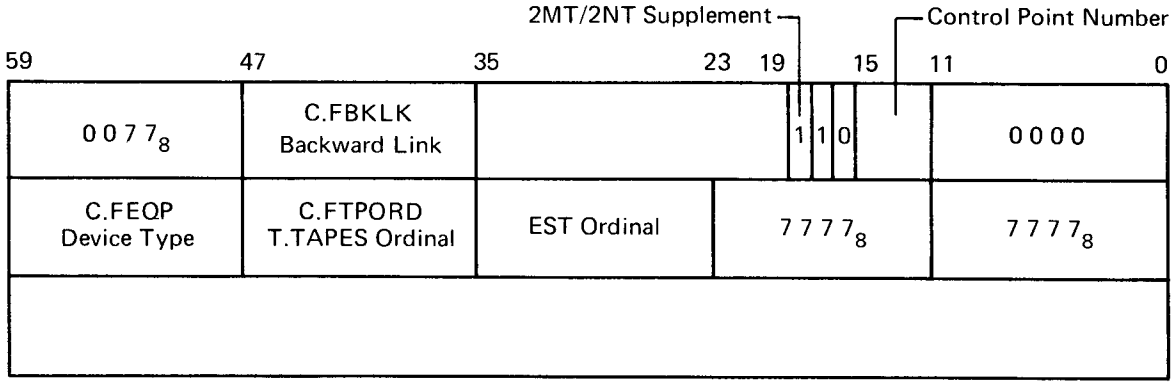
BEFORE EQUIPMENT ASSIGNMENT (GNT)



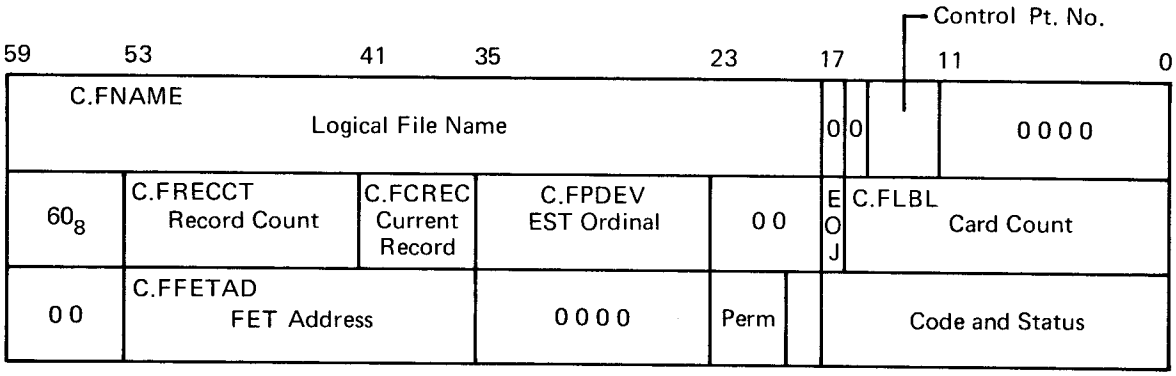
SUPPLEMENT(S) IF MORE THAN ONE VSN GIVEN



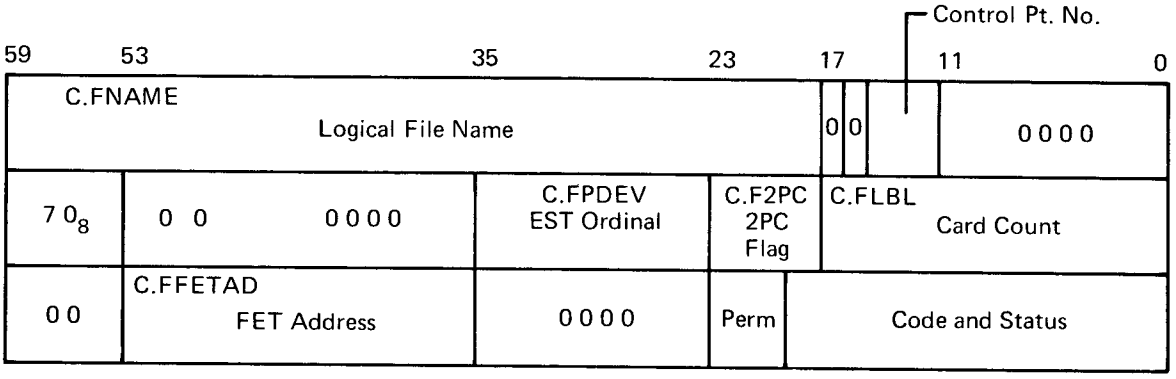
SUPPLEMENT IF 2MT/2NT DECLARED



ENTRY FOR ON-LINE CARD READER FILE

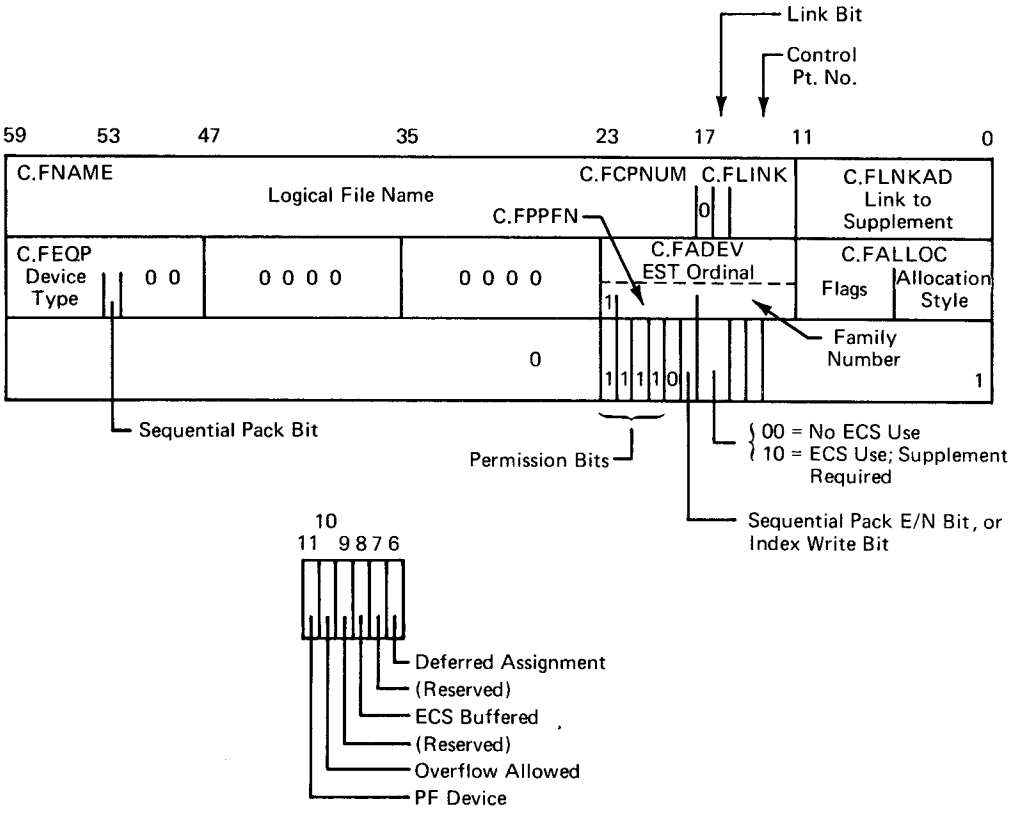


ENTRY FOR ON-LINE CARD PUNCH FILE

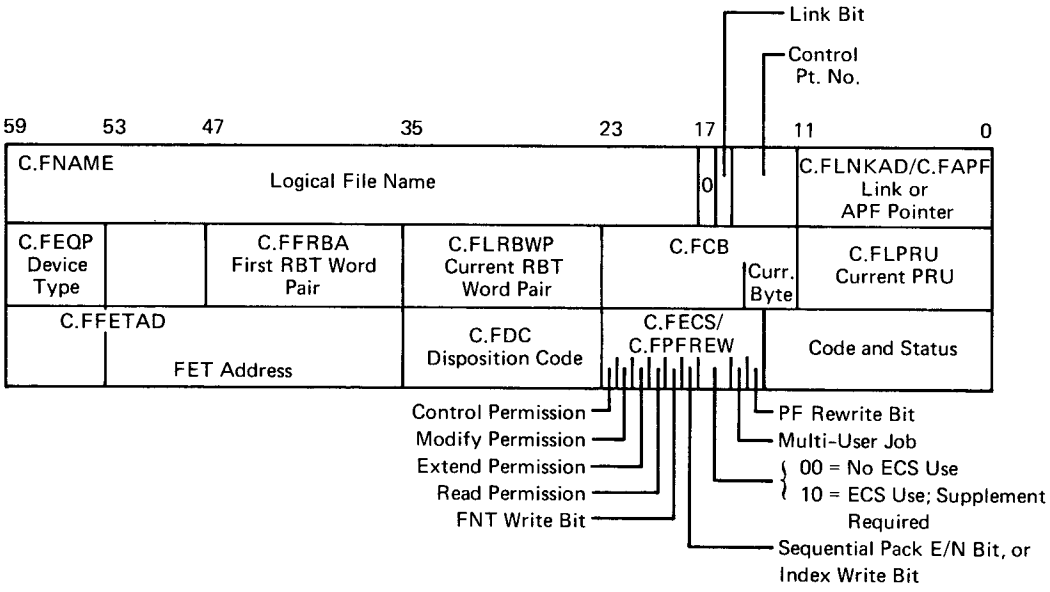


FILE NAME TABLE

ENTRY FOR LOCAL RMS FILE
BEFORE ASSIGNMENT TO A DEVICE



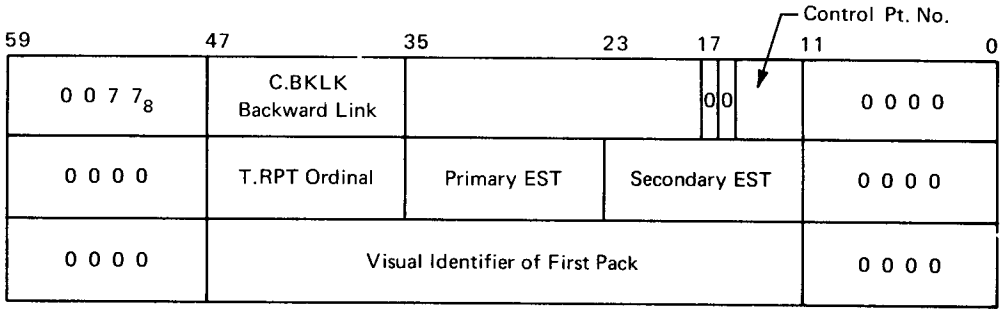
AFTER ASSIGNMENT TO A DEVICE
AND THROUGHOUT PROCESSING



FILE NAME TABLE

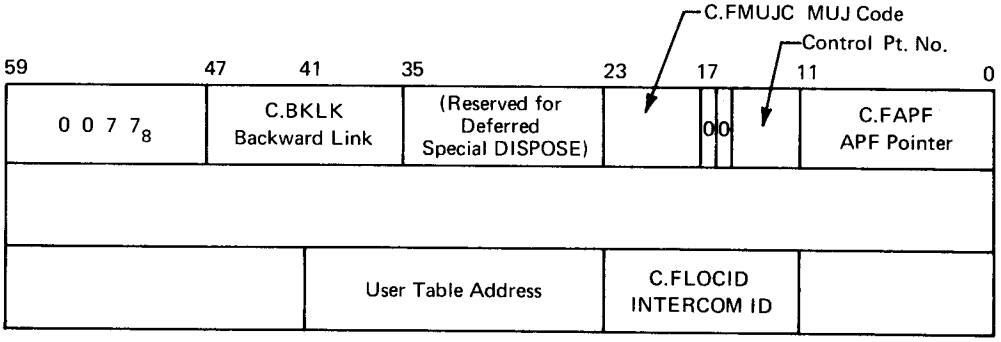
OPTIONAL SUPPLEMENTS FOR LOCAL RMS FILES

SEQUENTIAL PACK FILE SUPPLEMENT

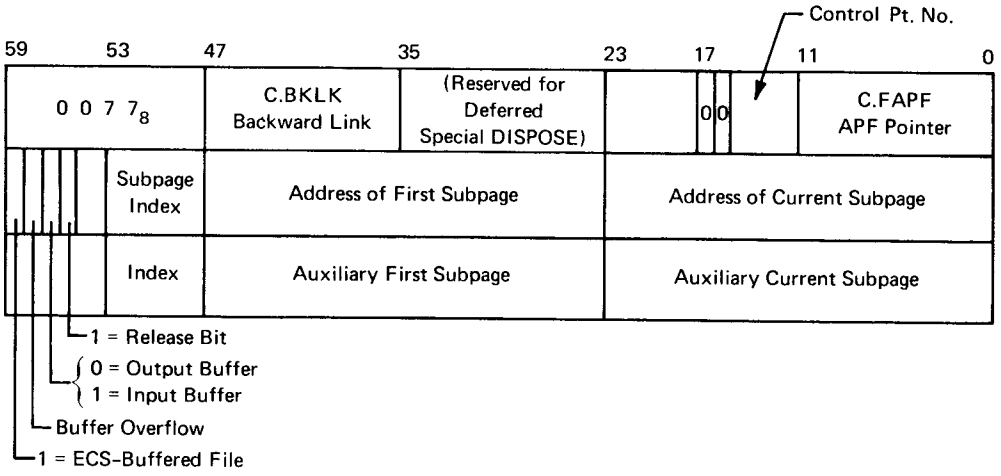


INTERCOM-USER FILE SUPPLEMENT

(Required only when file to be attached to swapped-out job)



ECS FILE SUPPLEMENT



(C.FDC) DISPOSITION CODE VALUES

(Byte 2 of word 3 in FST entries, where appropriate)

Allocatable Devices:

Code	Bit	Description
I	11	INTERCOM user file
RB	10	INTERCOM batch job
	9	Reserved
SD	8	Special name/disposition file
O	7	File named OUTPUT
IP	6	Interrupted print file (not currently implemented)

Non-Allocatable Devices:

Code	Value	Description
CK	xxx1	Checkpoint
IU	xxx2	Inhibit unload
CI	xxx3	Checkpoint and inhibit unload
SV	xxx4	Save
CS	xxx5	Checkpoint and save
	xxx6	Reserved

Low Order 6 Bits:

Code	Value	Description
	01	Reserved
	02	Reserved
	03	Reserved
	04	Input job ready for scheduling
	05	Input tape job
	06	Input tape job on VSN display
	07	Reserved

LOW ORDER 6 BITS OF DISPOSITION CODE (C.FDC)

Code	Value	Output	Description
PU	10	026	Punch 026 character set from display code
† PA	11	029	Punch 029 character set from display code: 66 = 11-8-2; 72 = 12-8-2
P8	12B	Absolute Binary	Punch 80 column binary
† P9	13	029	Punch full 029 character set (96 characters) from extended code set
PB	14	6000/7000 Binary	Punch 6000/7000 format binary
† PZ	15	029Z	Punch with COBOL 0 over-punch conversion: 029 character set from display code: 66 = 11-0; 72 = 12-0
† PF	16	029	Punch folded 029 character set from extended code. Lower case is folded into upper case and 11-8-2,12-8-2 are used

†Not supported by SCOPE 3.4.

Code	Value	Output	Description
† PC	17	029	Punch folded with COBOL 0 over-punch conversion: 029 character set from extended code. Lower case is folded into upper case and 11-0, 12-0 are used
† FR	20		Film print
	21		Reserved
† FL	22		Film plot
	23		Reserved
† HR	24		Hard copy print
	25		Reserved
† HL	26		Hard copy plot
	27		Reserved
† PT	30		Plot
	31-37		Reserved
PR	40	CDC-64	P1 or P2
P1	41	CDC-64	501/505 with 595-1 (CDC) graphic set
P2	42	CDC-64	512 with 595-1 graphic set

†Not supported by SCOPE 3.4.

Code	Value	Output	Description
PE	44	ASCII-95	512 with 595-6 graphic set from extended code input
† AR	50	ASCII-64	A1 or A2
† A1	51	ASCII-64	501/505 with 595-5 graphic set
† A2	52	ASCII-64	512 with 595-5 graphic set
† AF	54	ASCII-64	512 with 595-5 graphic set. Prints 64 characters. Lower case is folded into upper case.
	55-67		Reserved to CDC
	70-77		Reserved to Installations

Notes

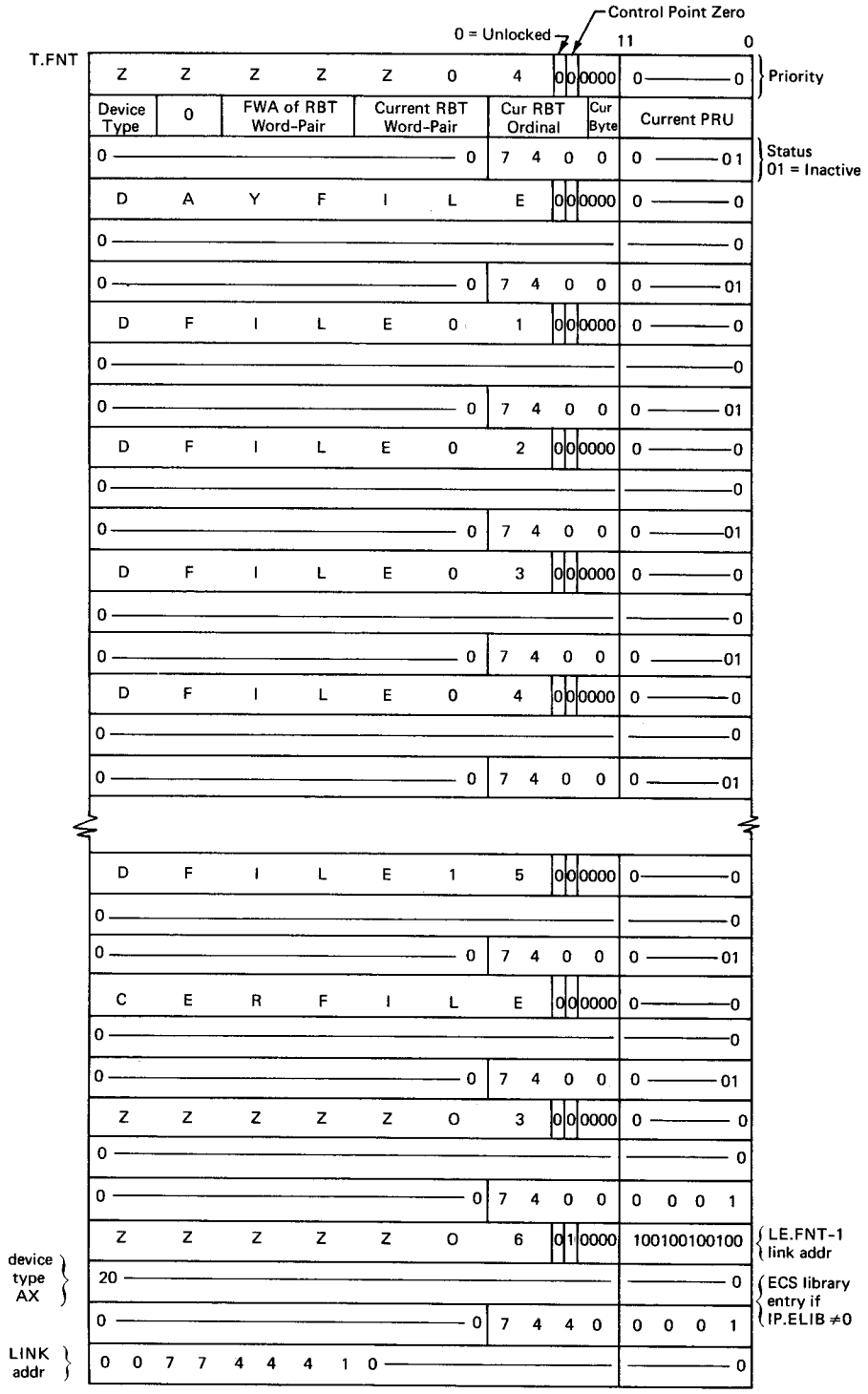
595-1 CDC 63 characters and blank are defined.

595-5 ASCII Subset 63 characters and blank are defined.

595-6 ASCII 95 Character Set 94 characters and blank are defined.

†Not supported by SCOPE 3.4.

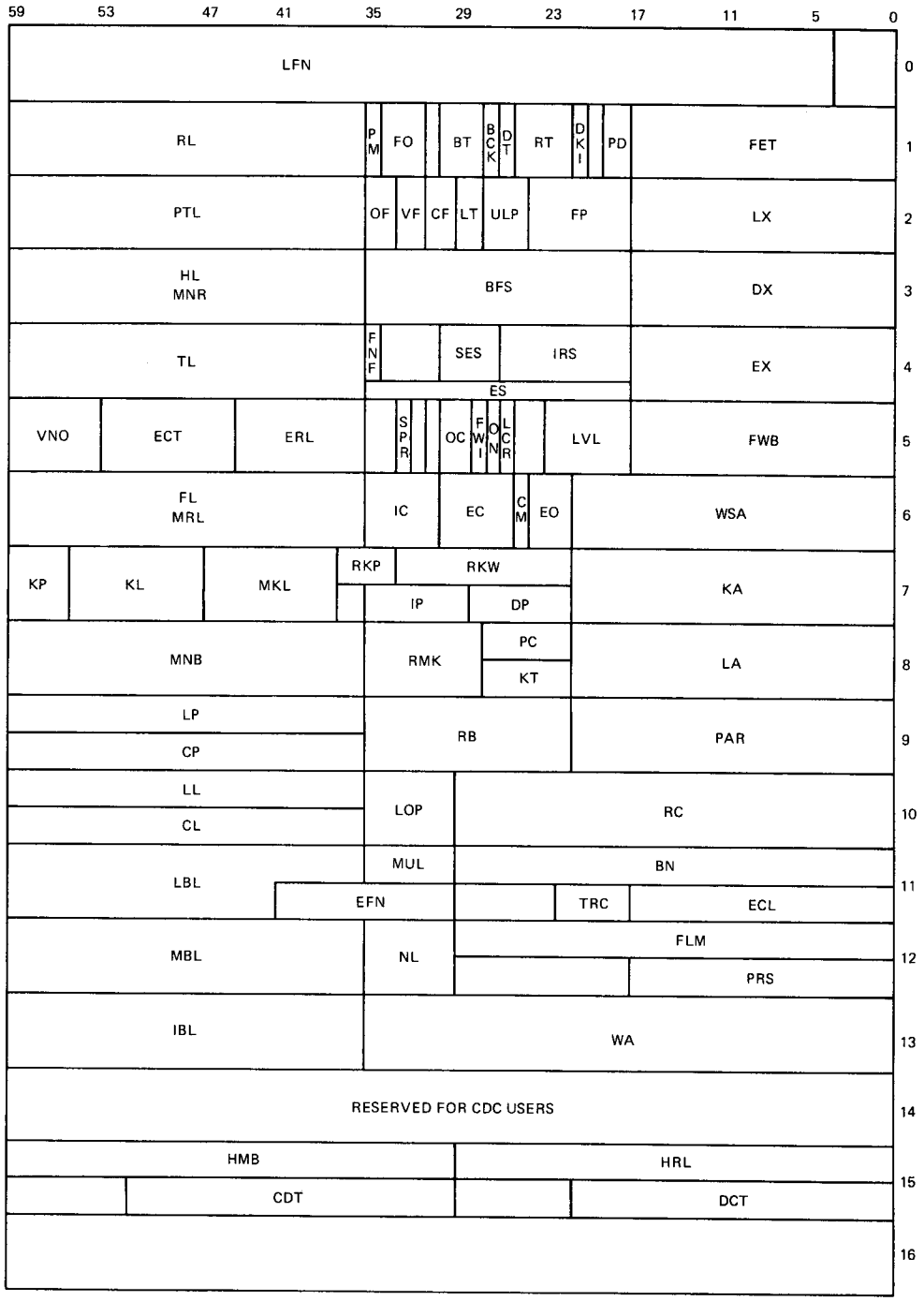
When the system is assembled, several system file entries are built into the FNT/FST for control point dayfiles, system (library) file, and hardware error file. Their initial entries are diagrammed in the following figure.



FILE INFORMATION TABLE (6RM Interface)

The FIT is created in the field length of a job which uses the Record Manager. It serves to describe those files manipulated by Record Manager and is used to that extent in conjunction with the File Environment Table.

FILE INFORMATION TABLE



FIT TABLE

Word 0		
59-0	LFN	Logical file name
Word 1		
59-36	RL	Current record length in number of characters
35	PM	Processing mode (Record Manager only) 0 = sequential 1 = random
34-32	FO	File organization 000 default 001 word addressable (WA) 010 direct access (DR) 011 indexed sequential (IS) 100 library (partition) (LB) 101 direct access (DA)
31		Unused
30-28	BT	Blocking type 001 default/internal (I) 010 character count (C) 011 record count (K) 100 exact records (E)
27	BCK	Block checksum flag 0 = No checksums Block checksum flag 1 = Checksums
26	DT	Device type (6RM only) 0 mass storage 1 on-line tape

25-22	RT	Record type 0000 control word (W) 0001 fixed length (F) 0010 record mark (R) 0011 zero byte (Z) 0100 decimal character count (D) 0101 trailer count (T) 0110 binary character count (B) 0111 undefined (U) 1000 SCOPE logical records (S)
21	DKI	Duplicate key indicator 1 Permission is on an IS file
20-18	PD	Processing directive 000 input (INPUT) 001 input (INPUT) 010 output (OUTPUT) 011 input/output (I/O) 100 output/input (reverse)
17-0	FET	Pointer to FET
Word 2		
59-36	PTL	Partial transfer length, set by GETP, PUTP macros
35-34	OF	Open flags (positioning of file at OPENM time) 00 rewind (R) 01 rewind (R) 10 no rewind (N) 11 extend (E)
33-32	VF	End of volume flags (position of file at volume CLOSEM time) 00 unload (default) 01 rewind (R) 10 no rewind (N) 11 unload (U)

31-30	CF	Close flags (position of file of file CLOSEM time) 00 rewind (R) 01 rewind (R) 10 no rewind (N) 11 unload (U)
29-28	LT	Label type 00 ANSI standard (ST) 01 non-standard (NS) 10 unlabeled (UL) 11 any (ANY)
27-25	ULP	User label processing 000 none 001 VOL/UVL (V) 010 HDR/EOV/EOF (F) 011 VOL/HDR/EOV/EOF (VF) 100 UHL/UTL (U) 101 VOL/UHL/UTL (VU) 110 HDR/EOV/EOF/UHL/UTL (FU) 111 all (VFU)
24-18	FP	File position 000 Mid-record 001 In header label groups (only set during label header processing) 002 Beginning of information (BOI) (only set on SKIPBu in connection with DX) 010 End of section (EOS) 020 End of record (EOR) 040 End of partition (EOP) 100 End of information (EOI)
17-0	LX	Label exit address
Word 3		
59-36	HL	Character length of fixed header of a type T record
	MNR	Minimum character length of type R record
35-18	BFS	CIO buffer size (number of words)
17-0	DX	End of data exit routine address

Word 4

59-36	TL	Character length of trailer portion of type T records
35-18	ES	Error status
35	FNF	Fatal/non-fatal flag (1 = fatal)
30-27	SES	System error subfield 01 Read parity level 1 02 Read parity level 2 03 Read parity level 3 04 Read parity level 4 05 Write parity level 1 06 Write parity level 2
26-18	IRS	Invalid request subfield
17-0	EX	Error exit routine address

Word 5

59-54	VNO	Current volume number of multi-volume sequential file
53-45	ECT	Current error count
44-36	ERL	Error count limit
35-34		Unused
33	SPR	Suppress read ahead (1 = suppress, 0 = no)
32-31		Unused

30-29	OC	Open/close 00 never opened 01 open 10 closed
28	FWI	Forced write indicator for IS file blocks (1 = forced write; 0 = no)
27	ON	Old/new file (1 = new)
26	LCR	Label action on PD tape (PD=I/O) 0 create new labels (N) 1 check existing labels (E)
25-24		Unused
23-18	LVL	Level number of S-type record
17-0	FWB	fwa of user I/O buffer
Word 6		
59-36	MRL	Maximum record length
	FL	Length of an F or Z type record (characters)
35-31	IC	Internal character code 00 CDC display code 01 } not available 37 }
30-26	EC	External character code 00000 CDC display code

25	CM	Conversion mode 1 convert from EC to IC 0 no conversion
24-22	EO	Parity error options 000 terminate job (T) 001 drop erroneous data (D) 010 (unused) 011 accept erroneous data (A) 100 display data and terminate job (TD) 101 unused 110 accept and display erroneous data (AD) 111 (unused)
21-0	WSA	First word address of user work storage area (user record area)
Word 7		
59-56	KP	Starting character position of record locator key specified in KA
55-47	KL	Key length
46-38	MKL	Major key length in number of characters (IS)
37-34	RKP	Relative keyword position (DA files)
33-22	RKW	Relative keyword position in RKW (DA files)
35-29	IP	Index block padding factor (% padding)
28-22	DP	Data block padding factor (% padding)
21-0	KA	Address of record locator keys (key field address)

Word 8		
59-36	MNB	Minimum block length
35-28	RMK	Record mark character
27-22	PC	Padding character for sequential file blocks
	KT	Key type for indexed sequential files 000 symbolic (S) 001 symbolic (S) 010 integer (I) 011 floating (F) 100 computational-1 101 actual (A) 110 computational-2
21-0	LA	First word address of user's label area
Word 9		
59-36	CP	Beginning character position of trailer count field (type T record)
	LP	Beginning character position of record length field (type D record)
35-22	RB	Number of records per K type block (BT=K)
21-0	PAR	Parameter list address
Word 10		
59-36	LL	Length of record length field (RT=D)
59-36	CL	Length of trailer count field (RT=T)
35-30	LOP	Last operation code (6RM only)
29-0	RC	Record count

Word 11		
59-36	LBL	Length of user's label area (number of characters)
35-30	MUL	Multiple of characters per K, E type block
29-0	BN	Block number of current block
47-30	EFN	Error file name word address
23-18	TRC	Number of transactions to be traced (IS or DA files)
17-0	ECL	Error code location (address of SIS V1.0 error code)
Word 12		
59-36	MBL	Maximum block length
35-30	NL	Number of index block levels (IS files)
29-0	FLM	File limit in words per file (FO=WA) or number of records per file (FO=IS)
17-0	PRS	Previous record length (FO=SQ) after a forward sequential GET
17-0	DL	Library file directory length (7DM only)
Word 13		
59-36	IBL	Index block length (words)
35-0	WA	Current position word address set by GET, GETP, PUT, PUTP and SEEK macros

Word 14 Reserved for users

Word 15

59-30	HMB	Number of home blocks of a DA file
29-0	HRL	Address of key hashing routine for a DA file
51-30	CDT	Collating sequence to display code conversion table address for IS file
21-0	DCT	Display code to collating sequence conversion table address for IS file

Words 16, 17, 18, are scratch words used by 6RM.

DEVICE TABLES

Tables in CMR that provide information on input/output equipment and channels are used by SCOPE to make file assignments. Tables included in this section: *Equipment Status Table* (EST), containing entries for all I/O equipment in the configured system; *Device Status Table* (DST) and *Device Activity Table* (DAT), providing information related to mass storage devices and controllers; *Record Block Reservation* (RBR) and *Record Block* Table (RBT) containing information of each record block in a mass storage device. The *Channel Status Table* (CST) provides I/O channel availability information and serves as an interlock for major file tables, which prevents modification of the same table entry by two or more programs. Also included are the *TAPES* table and the *Tape Staging Table* (STG), the *Rotating Mass Storage* (RMS) table, the *Device Pool Table*(DPT), the *Removable Pack Table* (RPT) and the *INTERCOM* table (ITABL).

HARDWARE MNEMONICS AND DEVICE TYPE CODES

Mnemonic	Device Type	Description
AA	01	6303-I disk
AB	02	6638 disk
	03	data cell
AC	04	6603-II disk
AL	05	821 data file
AM	06	841 multiple disk drive
AP	07	3234/854 disk pack drive
AF	10	814 disk file
	11	CDC reserved
AD	12	3637/865 drum
	13-17	CDC reserved
	21-27	CDC reserved
	30-37	Reserved for installations; mass storage only
MT	40	7-track magnetic tape
NT	41	9-track magnetic tape
	42	Member file 7-track tape
	43	Member file 9-track tape
	62	7-track multi-file set tape
	63	9-track multi-file set tape
TR	44	Paper tape reader
TP	45	Paper tape punch
	46	Reserved for installations
	47	Reserved for installations
LP	50	501, 512, 505 line printer
L1	51	501, 505 line printer

Mnemonic	Device Type	Description
L2	52	512 line printer
	53-55	CDC reserved
	56-57	Reserved for installations
CR	60	405 card reader
KB	61	Remote terminal keyboard
	64	Pseudo code for tape staging
	65	CDC reserved
	66-67	Reserved for installations
CP	70	415 card punch
DS	71	6612 keyboard/display console
GC	72	252-2 graphic console
HC	73	253-2 hard copy recorder
FM	74	254-2 microfilm recorder
PL	75	Plotter
	76-77	Reserved for installations
DC		6671 DSC
IX		Reserved for installations
SC		6673/6674 DSC
YC		6676 DSC
CC		6000/7000 channel coupler

EQUIPMENT STATUS TABLE (EST)

The Equipment Status Table resides in the upper table area of CMR and is pointed to by P.EST in the CMR pointer area. Table length is variable depending upon the system configuration at deadstart. Therefore, the CMR pointer word also includes the LWA + 1 address of the EST.

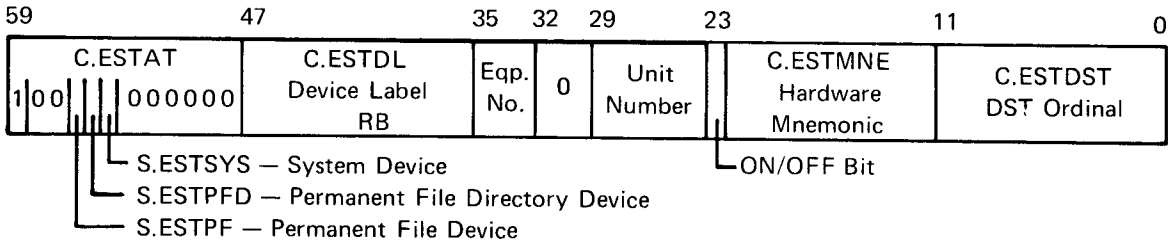
The EST contains a one-word entry for each device configured in the system, including consoles and remote terminal MUX devices. Each entry describes current status of the device and includes the device hardware mnemonic name, channels to which it is attached, device unit number, etc.

Entries in the EST are numbered starting with one; an entry number, called the EST ordinal, is used to identify the table position of each equipment entry. The EST ordinal of the equipment being assigned is given as xx in the operator type-in n.ASSIGNxx.

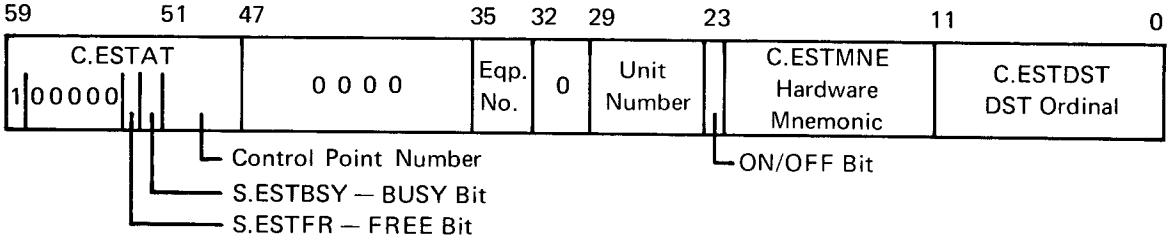
The EST is the basic reference for most other I/O tables. EST ordinals are found in the FNT/FST entries for linking file entries to their assigned equipment. EST ordinals in the TAPES table link tape entries to related equipment entries in the EST. Likewise, EST ordinals are found in the Record Block Reservation (RBR) table, linking that table to the allocatable device it describes.

(T.EST) EQUIPMENT STATUS TABLE

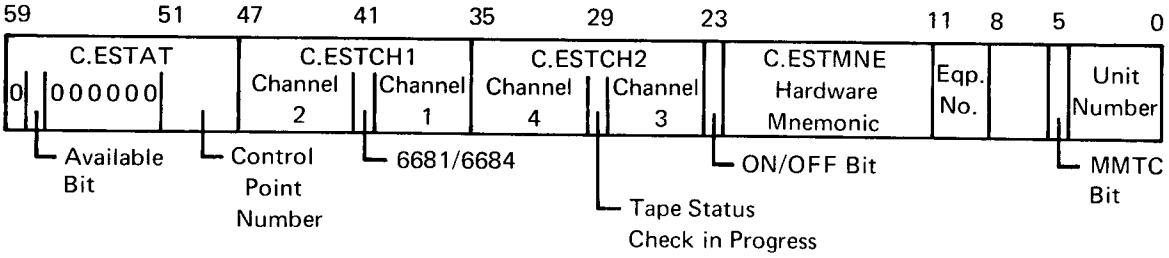
PUBLIC RMS DEVICE ENTRY



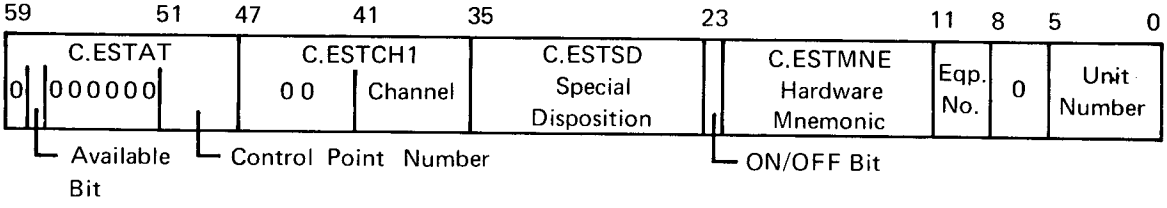
PRIVATE RMS DEVICE ENTRY



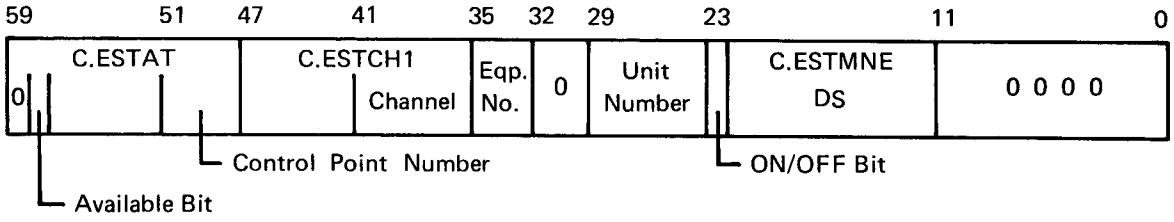
MAGNETIC TAPE ENTRY



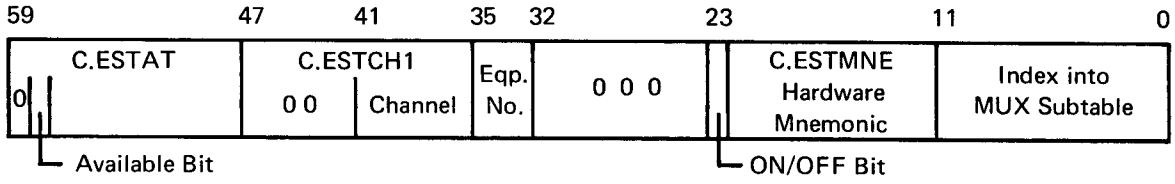
UNIT RECORD EQUIPMENT ENTRY



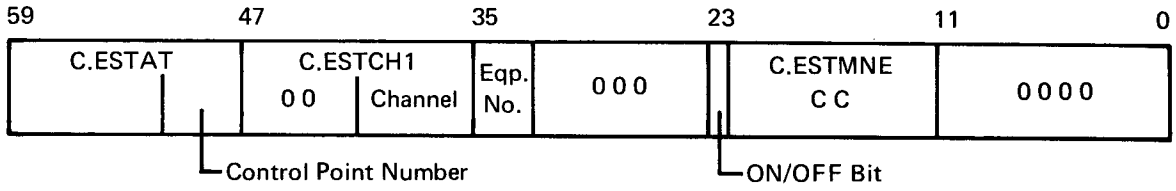
6612 DISPLAY CONSOLE ENTRY



MULTIPLEXER ENTRY



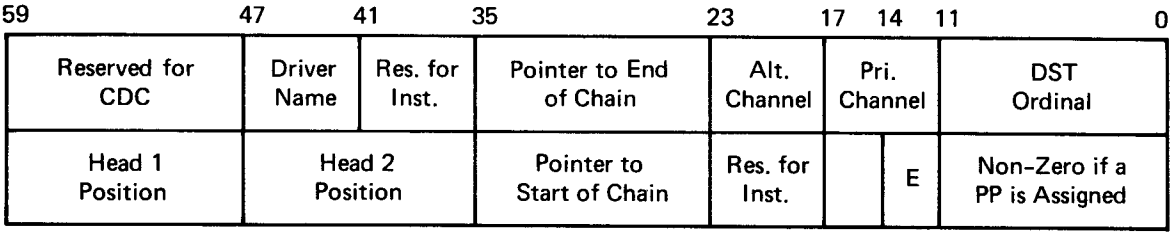
6000/7000 CHANNEL COUPLER



DEVICE STATUS TABLE (DST)

The stack processor uses the device status table in processing of mass storage files. The DST is located adjacent to the request stack in CMR upper table area. Each controller has one DST two-word entry which specifies the overlay to be used by the stack processor (1SP) for each controller, pointers to a chain of requests entered in the request stack for that controller, and device availability information.

Each entry is numbered, starting from 1, to identify DST ordinals. The format of a DST entry is:



Word 1 of the entry contains the driver overlay code which enables the stack processor to have the correct mass storage device overlay loaded. The code is keyed to the type of controller; it is used to form the driver name 3Sx where x is:

P	6603-I Disk
Q	6638 Disk
R	865 Drum
S	854 Disk Pack
T	6603-II Disk
U	814 Disk
V	821 Disk
W	841 Disk Pack

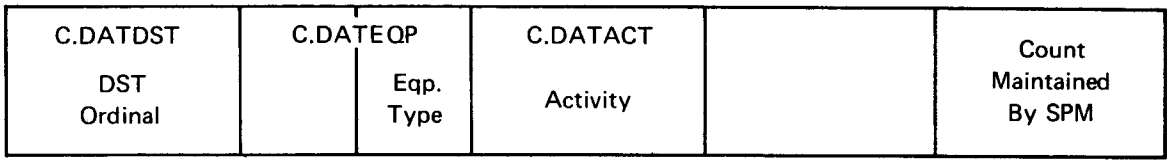
In word 2, if only one unit is attached to the controller, the head position unit 2 byte is not used. Also in word 2, if no 3000 controller is related to the entry, the field is zero; otherwise, it contains the 6681 converter unit number, 1 to 3. If the chain start and chain end pointers are not equal, a PP is assigned to process the stack entry chain. The assigned PP address is given in byte 4 of word 2; when the entry is completed and the PP is dropped, the byte is set to zero.

The device status table is a key table in the processing of allocatable storage files. DST ordinals are found in the Device Activity Table (DAT), the Record Block Reservation (RBR) table header, and the Equipment Status Table (EST). A DST ordinal appears in each DST entry; it is placed into the input register of the PP assigned to process an entry for that device in the request stack.

DEVICE ACTIVITY TABLE (DAT)

The device activity table is directly related to the device status table. It has one entry for each DST entry and is referenced by the mass storage device open overlay (3DO) in determining the best RBR to assign to a new or overflowing file.

Format of one-word DAT entry:

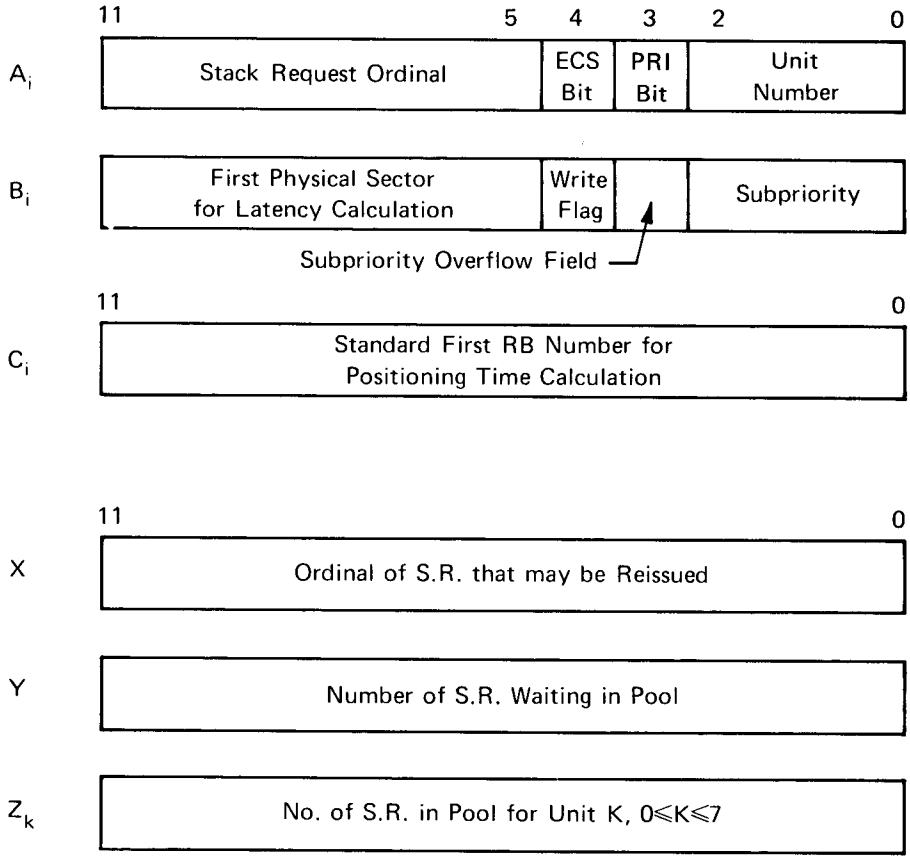


DEVICE POOL TABLE (DPT)

The device pool table contains a 10-word area for each RMS device. Both ISP and IEP routines use this area for passing their pool tables to each other.

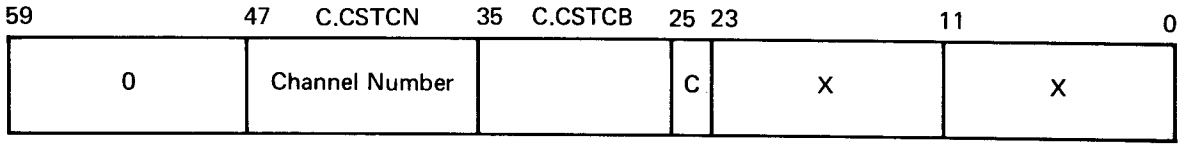
Internal Information Passed Between IEP and ISP Each Time One Calls the Other

T.DPT	59	47	35	23	11 0
1	A_1	A_2	A_3	A_4	A_5
2	A_6	A_7	A_8	A_9	A_{10}
3	A_{11}	A_{12}	B_1	B_2	B_3
4	B_4	B_5	B_6	B_7	B_8
5	B_9	B_{10}	B_{11}	B_{12}	C_1
6	C_2	C_3	C_4	C_5	C_6
7	C_7	C_8	C_9	C_{10}	C_{11}
8	C_{12}	(Reserved)	X	Y	Z_0
9	Z_1	Z_2	Z_3	Z_4	Z_5
10	Z_6	Z_7	(Reserved)	(Reserved)	(Reserved)

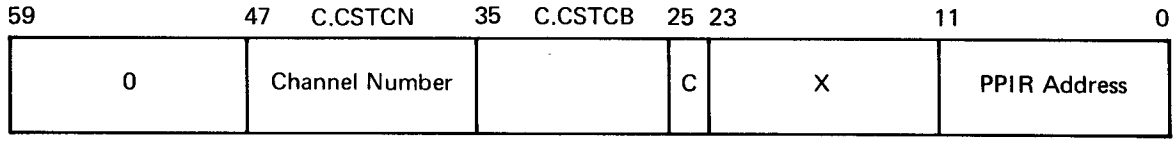


CHANNEL STATUS TABLE (CST)

The Channel Status Table, residing in the lower table area of CMR, contains a one-word entry for each hardware channel and each pseudo channel in the system. For a reserved channel, the PP reserving the channel is identified in the entry. When a channel is not reserved, the CST entry format is:



If a channel is being used by a PP program, the entry format is:



X	Address of this word
C	MMTC conversion table flags
	00 No table
	01 EBCDIC
	10 ASCII
	11 Reserved

The channel number is obtained by a PP program from the EST entry for the type of equipment. The length of the CST includes entries for a minimum of 12 hardware channels (optionally 24 maximum) and 13 pseudo channel numbers. Channel number assignments are listed in the following table.

CHANNEL NUMBERS

00 thru 13	Hardware Channels
14	CH.FST (controls access to FST)
15	CH.FNT (controls access to FNT)
16	CH.LIB (controls access to LIB for 250 Graphics software only)
17	CH.RBT (controls access to RBT)
20-33	Hardware Channels
34	CH.CPA (control point area interlock)
35	CH.PFM (Permanent file manager channel)
36	CH.INS (for use by installation)
37	CH.DMP (represents dump channel)
40	CH.EST = CH.TAPE (controls access to EST/TAPES table)
41	CH.ICTPT (INTERCOM control point channel)
42	CH.IEMBF (INTERCOM empty buffer channel)
43	CH.IUSER (INTERCOM user table channel)
44	CH.SCH (Scheduler channel)

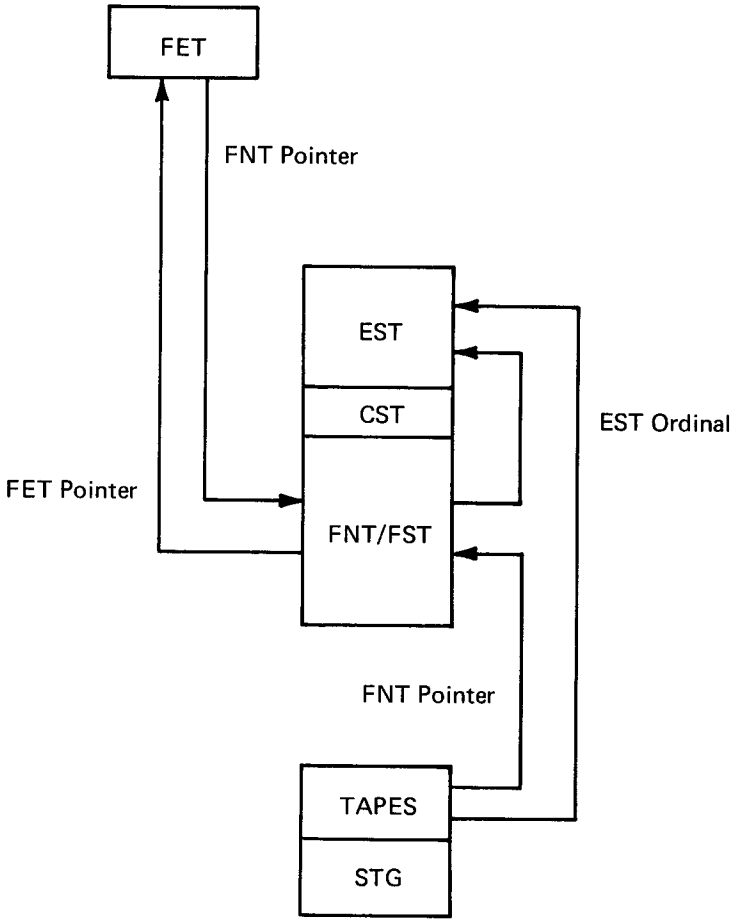
Access to the FST/FNT/RBT is controlled by an interlock scheme which prevents two or more programs from attempting to modify the same table entry at the same time. Not all table accesses require pseudo-channel reservations. Some of the conditions which require pseudo channels are:

Entry is added to FNT

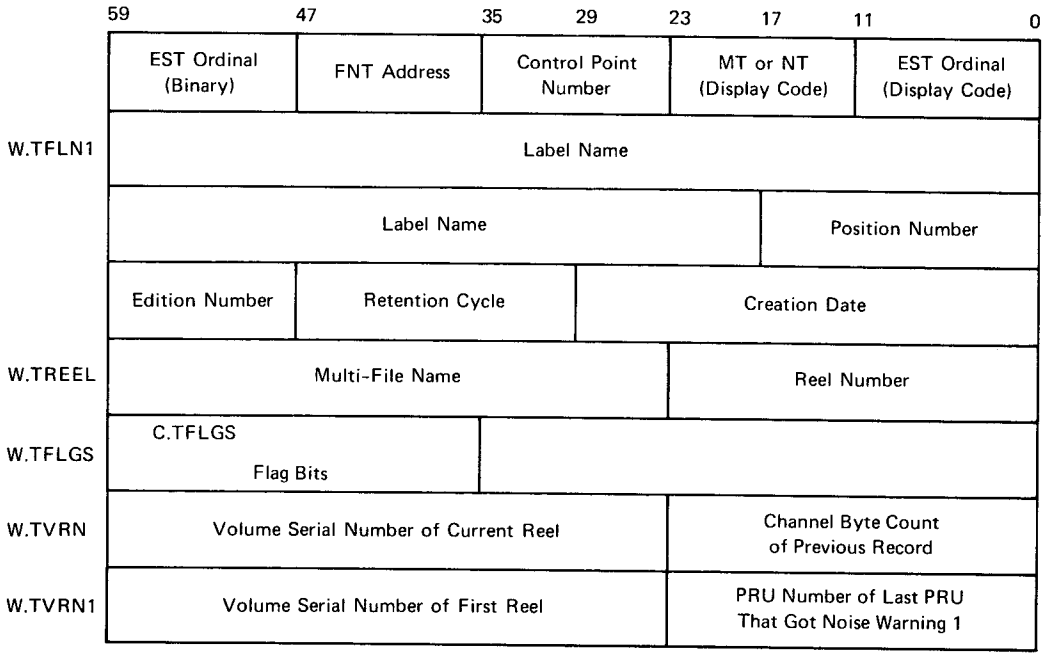
File is assigned to a control point, causing FNT modification

FST code/status byte is initialized

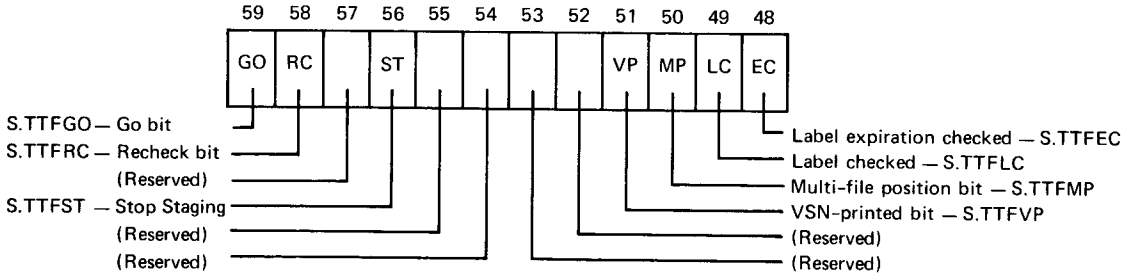
Tables related to file processing on non-allocatable devices:



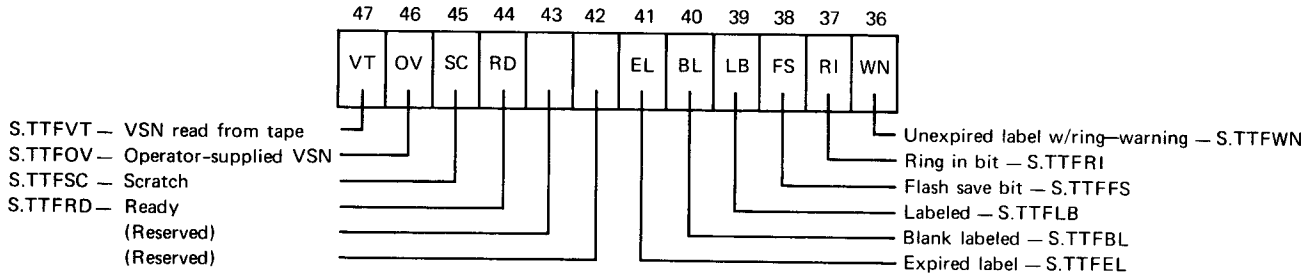
T.TAPES TABLE



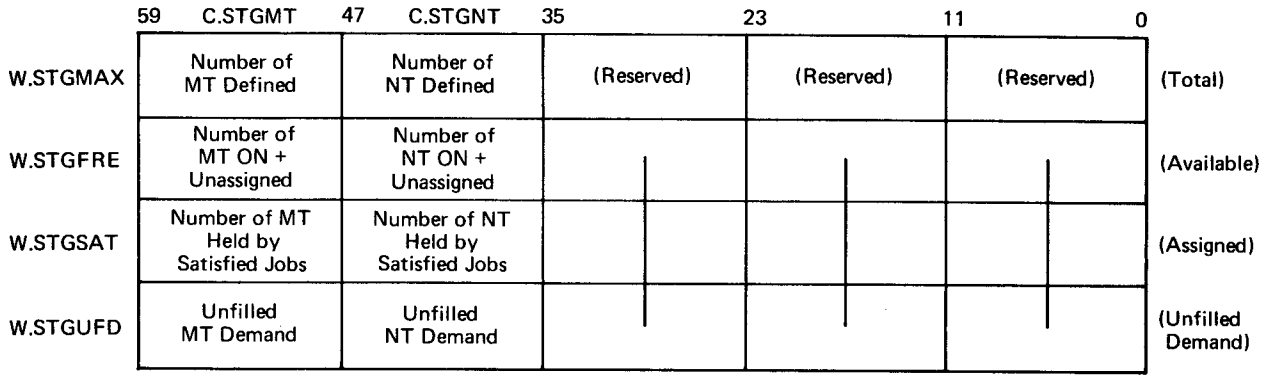
C.TFLGS — Job-Oriented Flag Bits



C.TFLGS+1 — Unit-Oriented Flag Bits



TAPE SCHEDULING TABLE



NON-ALLOCATABLE DEVICES

TAPE DRIVE SCHEDULING

The purpose of tape drive scheduling is to improve the overall system throughput, particularly as it relates to tape job setup and execution. The features of the tape scheduling scheme are discussed in the next paragraphs.

AUTOMATIC TAPE DRIVE ASSIGNMENT

Because ANSI tape labels include a volume serial number (VSN) field, it is possible for the user to have tape drives assigned automatically to his ANSI-labelled tapes by specifying the VSN as a parameter on the REQUEST control card. This feature does not encroach upon automatic assignemnt by label as in SCOPE 3.3. The VSN parameter declares that the tape label is either type U (full ANSI-standard label) or type Z (existing SCOPE "standard" label). The Z labels are not ANSI standard, as the recording density field (character 12 of the volume header label) is not standard.

Automatic tape assignment by VSN allows a form of automatic assignment of unlabeled tapes. In such cases, the VSN must first be entered by the operator, whereupon the tape will be assigned automatically to any and all jobs which name its VSN. U-labeled (formerly Y-labeled in SCOPE 3.3) tapes do not contain VSN information. They may be automatically assigned by label or by VSN, if the operator first enters the VSN through the console.

VSN CONTROL CARD

The VSN card relates the physical tape reel number (VSN) to the logical file name and also provides information required for the tape job prescheduling display. When used with the REQUEST and/or LABEL control cards or the REQUEST function, it serves to relate a VSN to a logical file name, which is important to automatic equipment assignment. Used by itself, it serves no purpose other than supplying data for subsequent use in the job prescheduling display.

TAPE JOB PRESCHEDULING

The tape job prescheduling display is an extension of the P-display and lists, by VSN, the tape reels required by each tape job. A tape job is defined as one having MT and/or NT parameters on its job card. All incoming tape jobs are entered in a prescheduling queue, a subset of the input queue. The purpose of having a prescheduling queue is to advise the operator of tape reel requirements and to hold jobs in abeyance until such reels can be obtained from the tape library. This arrangement also allows the operator some control over the selection of tape jobs for execution.

The operator communicates with the prescheduling queue through DSD type-ins and the P-display. Each time the P-display is requested, tape jobs having the highest priority are displayed. A job requiring tapes is not placed in the normal job input queue until the operator releases it with a type-in. Once released, the job will be considered by assignment to a control point and execution; it will no longer appear in the prescheduling display.

JOB SCHEDULING WITH TAPE DRIVE OVERCOMMITMENT

Job scheduling based on tape drive overcommitment assumes that a tape job does not always need its maximum tape requirement for the duration of the job. Overcommitment takes advantage of the assumption that must of the processing activity of a job is done using less than the maximum number of drives neccessary for job completion; therefore a job is assigned only those drives it needs to continue execution at any instant in time. Its unneeded drives at that instant are made available to run other jobs.

Such a job scheduling algorithm would permit the total tape requirements of all active jobs to exceed the tatal number of drives in the installation, which could result in too many jobs requesting too many tapes. For example, two or more jobs whose maximum tape requirements have not been met can create a system deadlock by typing up all the tape units in the installation. The only way to resolve such a deadlock is to kill one of the competing jobs and rerun it at a later time.

A deadlock prevention algorithm is provided in SCOPE 3.4 as a part of REQUEST processing. Deadlock prevention is a function of tape assignment, not one of job scheduling, although the job scheduling algorithm has some built-in deadlock prevention features. The function of deadlock prevention is to refuse any assignment (manual or automatic) of a tape to a job if that assignment creates a potential deadlock. A potential deadlock is detected when two or more jobs whose maximum tape requirements have not yet been met have been assigned tapes in such a manner that there are not enough free units remaining to satisfy the maximum demand of any one of them. Tape jobs could be scheduled at random and the deadlock algorithm would evade such deadlocks, but the resulting refusal of tape assignments would cause operator confusion and loss of efficiency. Job scheduling based on tape drive overcommitment, therefore, attempts to create an optimal situation while deadlock prevention avoids the worst-case situations.

DYNAMIC TAPE DRIVE STATUS CHECKING

Information concerning the physical status of tape drive units is entered into the TAPES table and kept up to date by periodic checks of unassigned units for a ready/not-ready status. The period for status checking is set by the installation; it must be short enough to preclude the occurrence of an operator dismounting a tape from a drive unit and mounting another without detection. Such periodic checking of unassigned tape drives makes automatic assignment more efficient and flexible.

When a tape drive is set to not-ready status, the fact is noted in the TAPES table and the unit becomes available for assignment to a job. When a drive is put in the ready status, the TAPES table is updated with information read from the tape label. (If the tape is unlabeled, this fact is noted in the table.) A search is then made for a job that needs the tape and the tape is assigned to it, providing such an assignment will not cause a deadlock. This action holds true both for labeled tapes and tapes qualifying as scratch, per IP.TSG.

Whenever a requested tape cannot be located immediately, the requesting job is rolled out until the operator mounts the tape. When the tape is found, it will be automatically assigned to the requesting job and the job rolled back into CM to continue processing. While the job is rolled out, the operator may make a manual tape assignment which will cause the job to be rolled in automatically.

Dynamic tape drive status checking permits the automatic assignment of unlabeled tapes by volume serial number. A VSN entered by the operator is recorded in the TAPES table; as long as that drive remains in the ready status, the system knows that the tape is still mounted and that it may be assigned without operator intervention to any job requesting that VSN.

Dynamic status checking also provides an improved scheme for reel swapping of multi-reel files. The 2MT parameter is no longer required on the REQUEST card; however, it is supported for the sake of compatibility. Whenever an end-of-tape before end-of-file mark is encountered, SCOPE will automatically search the TAPES table for the next reel of that file. If the needed tape is located, the unit on which it is located will be assigned to the job and the original unit deassigned. If the tape cannot be found, an appropriate message is posted to the operator; he may respond by either mounting the required tape or making a manual assignment.

Reel swapping will take place regardless of whether the file was scheduled by VSN or by label reading. For input tape reel swapping by label, SCOPE searches the TAPES table for a duplicate of the current label but with the reel number incremented by 1. For output files, SCOPE simply looks for another scratch tape. In reel swapping, deadlock considerations are irrelevant because one tape drive is beign assigned while another is deassigned.

ROTATING MASS STORAGE DEVICES

Figure 4-1 illustrates the various types of RMS devices, the definitions of which follow:

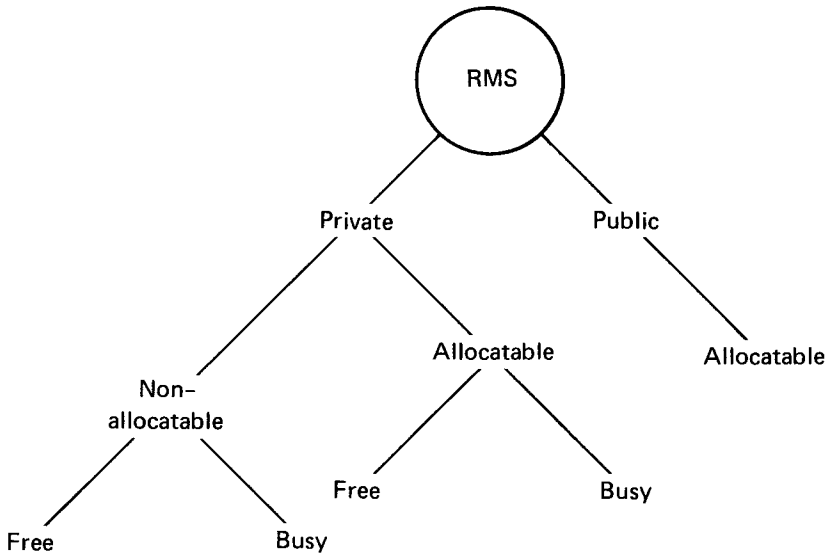


Figure 4-1. RMS Device Types

Private Device	A device which can only be assigned with an explicit request. A private device is logically removable, therefore may be used for family packs (formerly called private packs) and for sequential packs.
Public Device	A device which can be assigned by default. Public devices are allocatable and logically non-removable. Examples are system device, permanent file device, and permanent file directory device.
Non-Allocatable	A device which can be assigned only to one job at any one given time. Sequential packs and family packs are private, non-allocatable devices: assigned by explicit request, assigned only to one job at a time, and logically removable.
Allocatable Device	A device that can be shared by more than one job at a time. Allocatable devices may be either private or public. A public allocatable device may be a system device, PF device, or PFD device, which are assigned by default. A private allocatable device example would be a system permanent pack.
Free	A term used only for private devices which indicates the device is logically not in use.
Busy	A term used only for private devices. Indicates that the device is logically in use; its existance and type of data is know to the system.

DISK PACK DEVICES

A disk pack device may be designated at system deadstart as a public, private, or system permanent device. A public device is treated the same as any other disk pack device; it may contain one or more files of various types which may be assigned to different control points. The device itself is not assigned to any control point. A disk pack on a public device is not removable.

FAMILY PACKS

A family pack may contain one or more files which are referenced by only the control point to which it is explicitly assigned by an RPACK control card. Assignment of a family pack to a private device causes FNT entries to be made for each file resident on the pack; no further action is necessary to reference the files. New files may be created on the pack by using a REQUEST card; unwanted files may be removed by using a REMOVE card. At job termination, all information about each file residing on the pack is written back to the pack before the information is cleared from memory.

A single family of files may reside on as many as five disk packs. All packs in the family must be mounted on private devices and remain mounted until the job terminates. Subsequent disk pack devices will be assigned automatically as files are being created on the packs, up to the limit of five.

SEQUENTIAL PACKS

One or more disk packs may be used to contain a single file whose size may span as many packs (volumes) as are needed. A private, non-allocatable (removable) disk pack device must be explicitly assigned to the file by a REQUEST card; the file may be referenced by only the control point making the request. Only the pack (volume) containing the file portion to be used needs to be mounted on the assigned device.

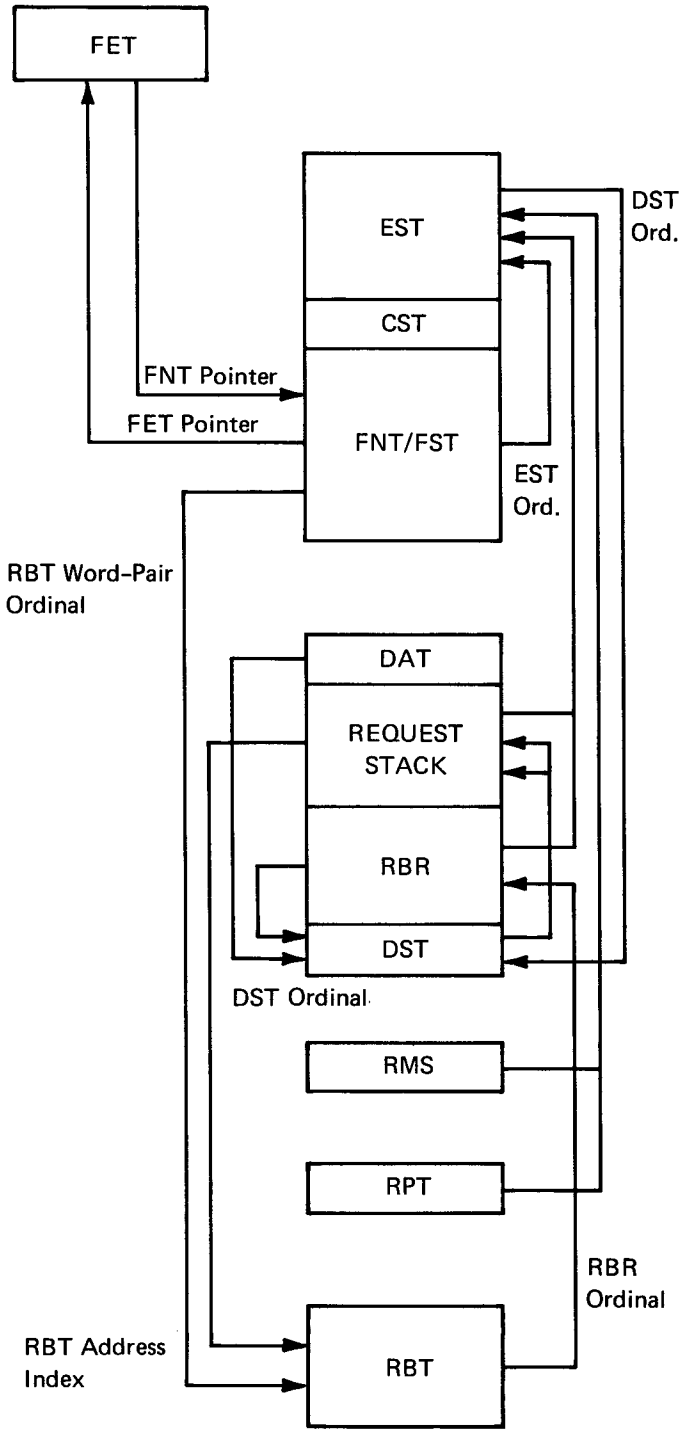
The method used for volume swapping depends on the manner in which equipment used for the file is requested. If a single device is requested, the operator must mount packs in sequence on that device; if two devices are requested, the operator must mount packs in ping-pong sequence, beginning with the first device assigned. Volume swapping will occur automatically or may be controlled by the user. Processing on a given volume may be terminated prematurely by a CLOSER function; the next volume will then be available. A return to the first volume of the file may be effected by a CLOSE function.

SYSTEM PERMANENT PACKS

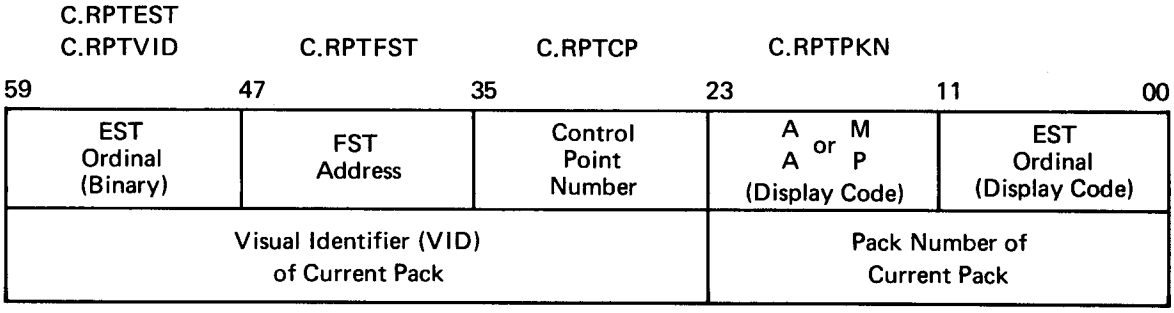
System permanent packs allow the addition of permanent files to a system without requiring loads from tape, provided that these files have been created on physically removable disk packs. The system permanent pack feature is a new addition to SCOPE.

All packs to be used as system permanent packs must have been declared to be permanent file devices during deadstart, since these packs will contain all permanent files known to the system. The number of packs cannot exceed the number of available devices, as all packs must remain mounted while the system is running.

Tables related to file processing on allocatable devices:



REMOVABLE PACK TABLE ENTRY (RPT)



RMS TABLES

To comprehend the functions of the various tables involved, the methods of mass storage space allocation must be understood. Terms are defined below:

SECTOR: the smallest accessible physical space increment on a track of a rotating mass storage device.

PRU: (physical record unit): the smallest amount of data a user may access; it is 64 decimal (100 octal) central memory words and is usually equal to 1 sector.

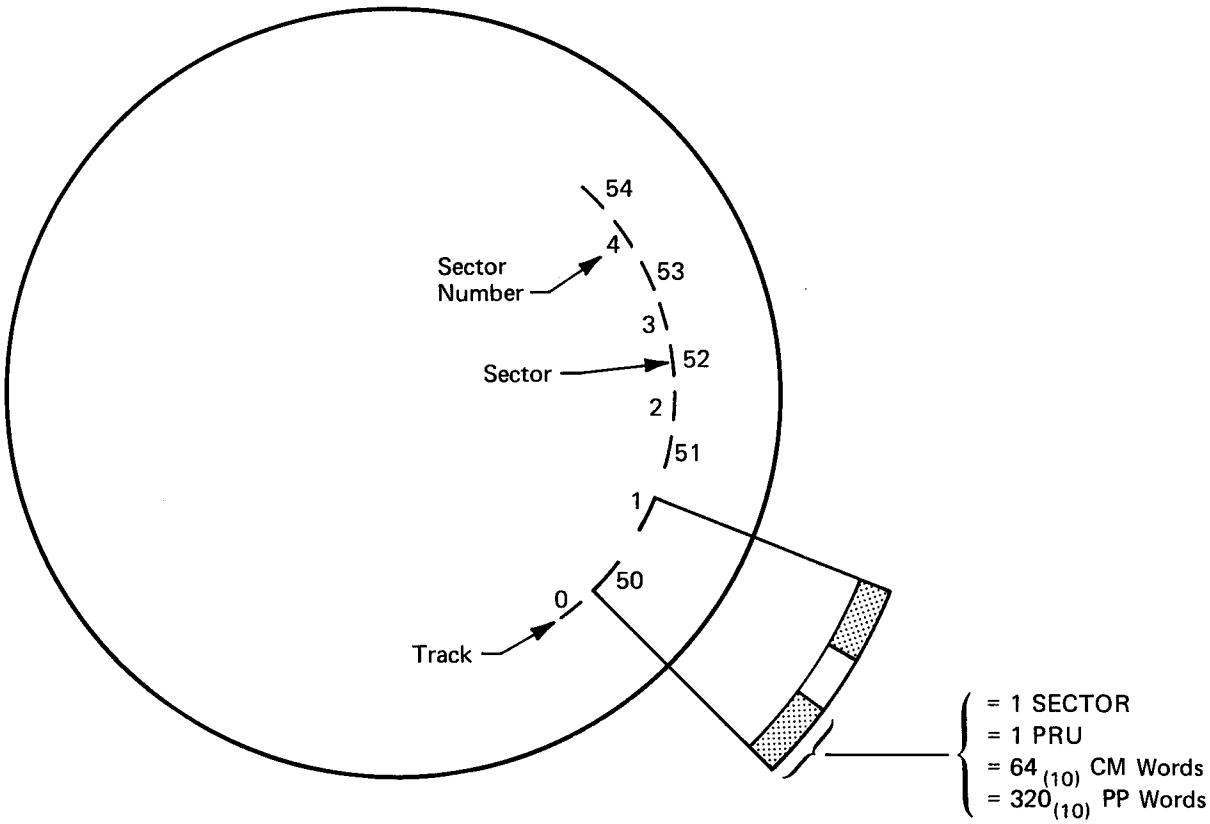
RB (record block): an increment of allocatable space on a mass storage device (RMS or ECS), usually a number of PRUs.

RBR (record block reservation): a bit-coded table which indicates those RBs on a device which are assigned to a file, flawed (defective), or available for assignment. A 0-bit indicates that the specific RB is available for assignment. The standard number of physical record units per record block (PRU/RB) is given in the following table:

Device	RMS Type	Type/Device Codes	PRU/RB
6603-I	DISK	AA 01	**
6638	DISK	AB 02	50
6603-II	DISK	AC 04	**
865	DRUM	AD 12	21
863	DRUM	AE 11	21
814	DISK	AF 10	62
821	DISK	AL 05	320 †
841	DISK PACK	AM 06	56
854	DISK PACK	AP 07	4

** INNER ZONE IS 50 PRU/RB
OUTER ZONE IS 64 PRU/RB

† 2 RBRs ARE REQUIRED



Half-Track Recording Technique

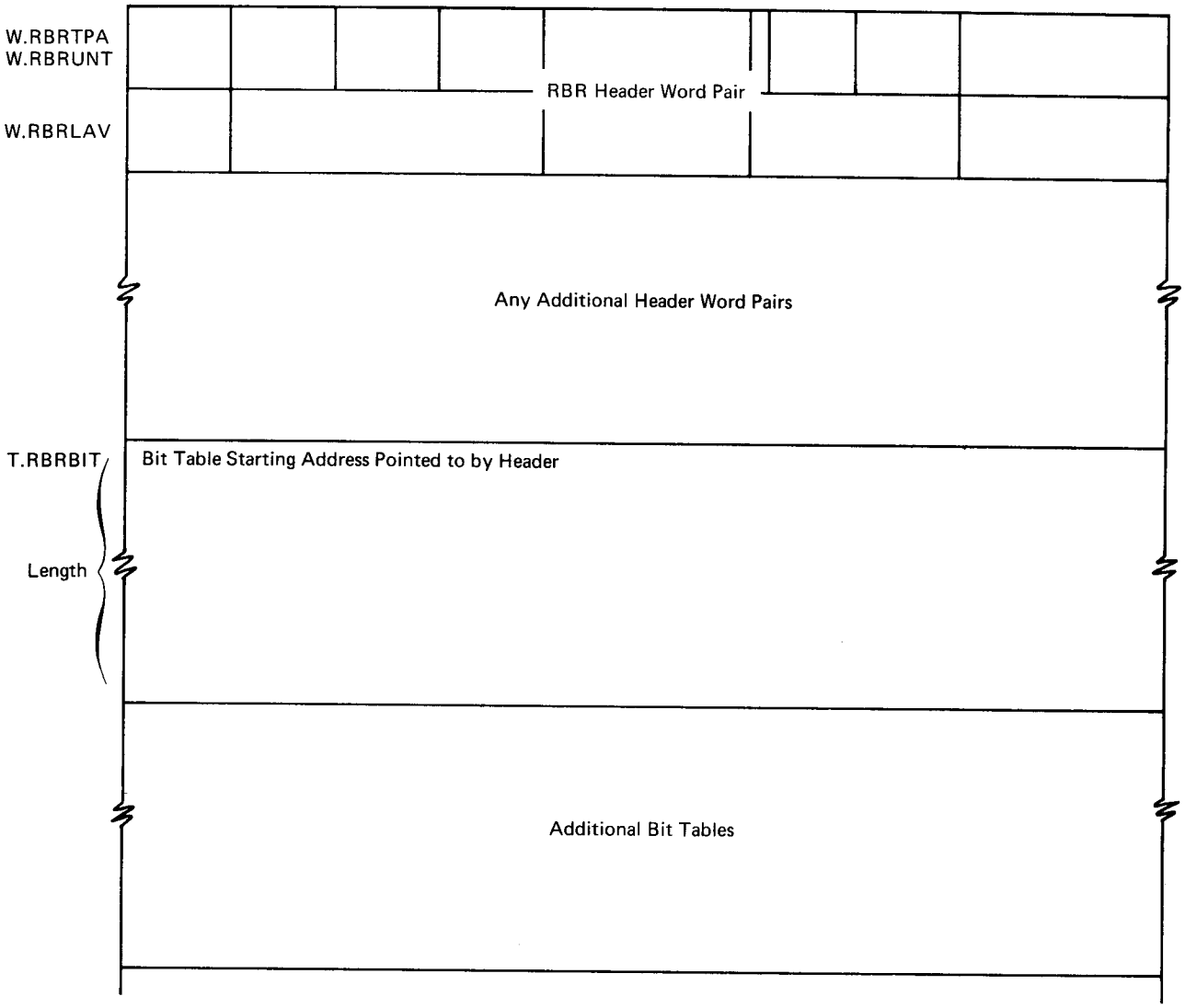
Alternate sectors of a physical recording track on 6603 or 6638 disks are numbered consecutively; intervening sectors are numbered the same way. Essentially, one recording track is divided into two half tracks, one containing the odd-numbered sectors, the other even-numbered sectors. The time required to move to the next numbered sector is used to set up parameters and load or unload PP memory in preparation for the read/write of the next sector.

RECORD BLOCK RESERVATION TABLE

A record block on a mass storage device is allocated to a file before any data can be written to that file. As data is written and a record block is filled, another record block must be assigned. Before the stack processor can select a record block to assign to a file, it must determine availability of record blocks. A record block reservation table maintained in CMR provides this information.

Each mass storage device is represented by at least one entry in the RBR. Several RBRs can be generated for a single device, each describing a unique area on the device. Each entry is made up of a two-word header and a variable length bit table. Each bit represents the availability of the corresponding record block; if a bit is zero, the RB is available for assignment; if a bit is one, the RB is not available.

Format of the RBR:

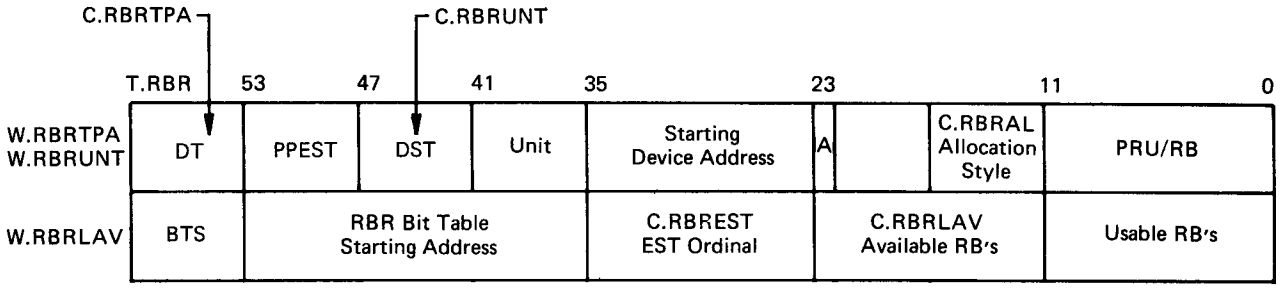


The first word of each RBR header contains a 6-bit allocation style code and a default bit; both are supplied as parameters to the RBR macro when the CMR is assembled at an installation. Unique allocation style codes for each standard RB may be set by the installation; this code can be used to direct a file to the RBR with the correct RB size and/or recording technique. If the default bit is set in an RBR header, file allocation requests that do not specify allocation style may be assigned to that RBR.

Standard (default) RB size and RBR bit table length for each mass storage device are found in the table given under MASS STORAGE I/O PROCESSING.

Format of two-word header:

RECORD BLOCK RESERVATION TABLE



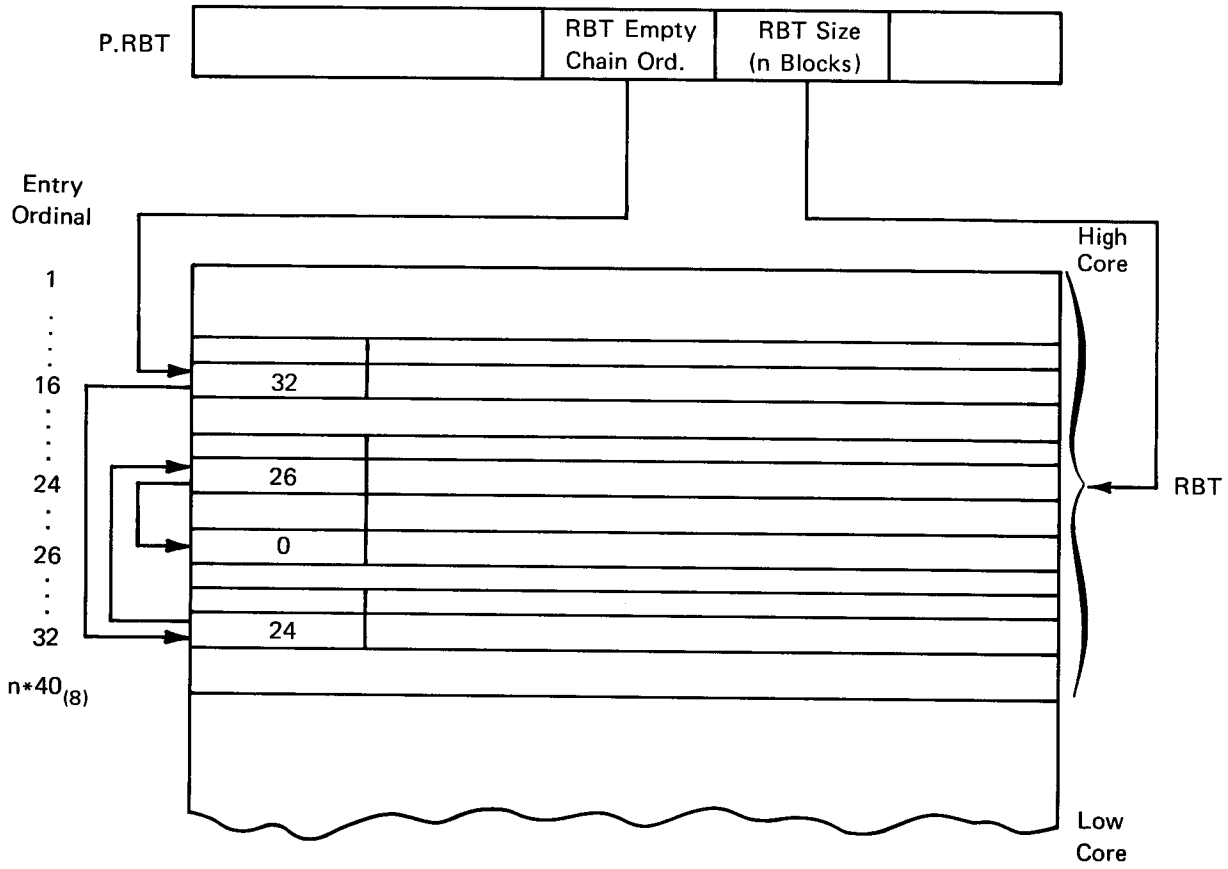
DT	Device type code
PPEST	Family pack EST ordinal (first pack in family)
DST	DST ordinal
UNIT	Unit number
A	If one, files without specific allocation style are permitted on the device
BTS	Bit table size divided by two
Alloc. Style	Allocation style for this RBR

RECORD BLOCK TABLE (RBT)

The Record Block Table (RBT) is file-oriented. Each mass storage file in the system has an associated RBT entry. The RBT, located in the highest address end of central memory, consists of several word pairs, one for each file that exists on an allocatable device currently recognized by the system. The RBT expands and contracts by 100 octal word blocks as files are allocated and released. A maximum of 8192 (decimal) central memory words may be assigned to contain all the RBT entries active at any one time.

When a mass storage file is established, a two-word RBT entry is created for that file; additional entries are assigned and linked in a chain as the file expands and entries are needed. Each entry consists of ten 12-bit bytes, some are used as pointers to additional entries in the chain and to other tables. Remaining bytes in the entry contain the RB number of a record block assigned to the file. RB numbers are placed in sequential RB bytes in order of their assignment. An RB number serves as the address of a bit in the RBR bit table representing the availability of that record block; it is also the address of the corresponding physical record block on the mass storage device. RBT entries are addressed by RBT word-pair ordinals. The word-pair ordinals are numbered sequentially starting from the highest address in central memory.

The CMR pointer word P.RBT contains the current size of central memory in 100-word blocks, as well as the current length of the RBT in 100-word increments. The same word also contains the RBT word pair ordinal of the first member of the RBT empty chain. Unused word pairs in the RBT are linked together to form the empty chain. As record blocks are released from an evicted file, the dropped word pairs are linked to the end of the empty chain. Word pairs are assigned to entering files from the head of the RBT empty chain and the new first-member word pair ordinal is entered into the CMR pointer word.

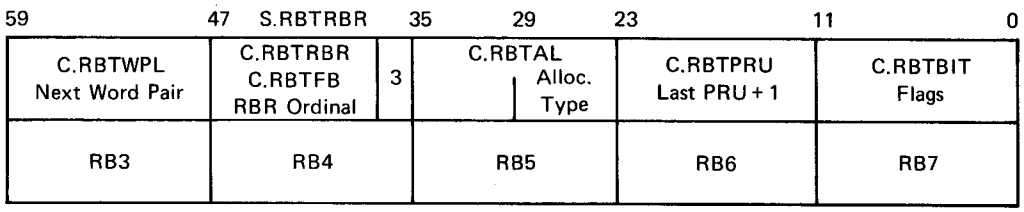


Word 1 of the first word pair (figure 4-2) assigned to a file contains ordinals, flags, etc. The RB bytes (up to 8 usable in any word pair) denote the record blocks assigned to the file. These bytes, initially zero, are set as each record block is assigned. The values in the RB bytes are interpreted as either RB numbers (for RBR bit table use) or as physical record block addresses (see figure 4-4). As a file expands, additional RB addresses are entered into the RB bytes until the word pair is filled or no more record blocks are assignable from the RBR table. In either case, another word pair is assigned to the file and linked to the current word pair. Bytes 0 and 1 in the succeeding word pair contain RBT and RBR ordinals. If an overflow occurred, the RBR ordinal is replaced by the value of 777. The remaining 8 RB bytes contain assigned record block addressed.

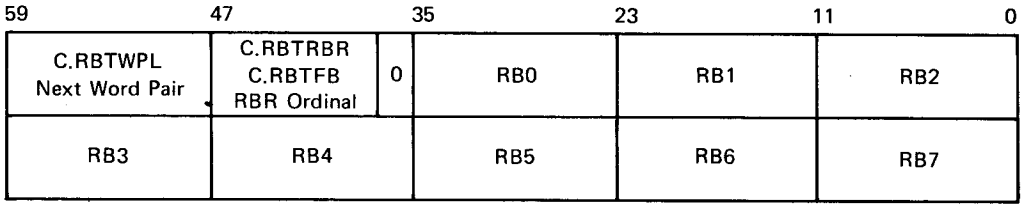
As a file is evicted or record blocks are dropped, the RB bytes are cleared. When an entire word pair is emptied, it is linked to the end of the RBT empty chain.

RECORD BLOCK TABLE (RBT) ENTRY FORMAT

First RBT Word Pair:



Other Word Pairs:



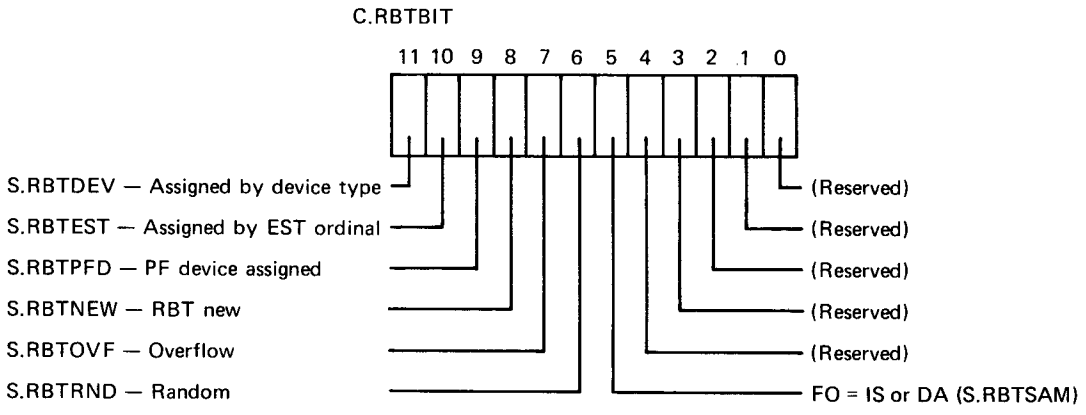


Figure 4-2. RBT Entry Format

The end of a file's RBT chain is a word pair having zeros in byte 0 of the first word. The last word pair in the empty chain contains all zeros.

The interrelationship of the FNT, RBT and RBR entries for a file assigned to an outer zone half track on a 6603 RMS disk is illustrated in figure 4-3. Pointers in the lower table area of CMR give the first word addresses of the FNT and the RBR area. The RBT size, in 100-word blocks, is multiplied by octal 40 to yield the number or word pair ordinals in the RBT. Multiplying any RBT ordinal by 2 and subtracting it from the LWA + 1 address of central memory will produce the address of the word pair entry.

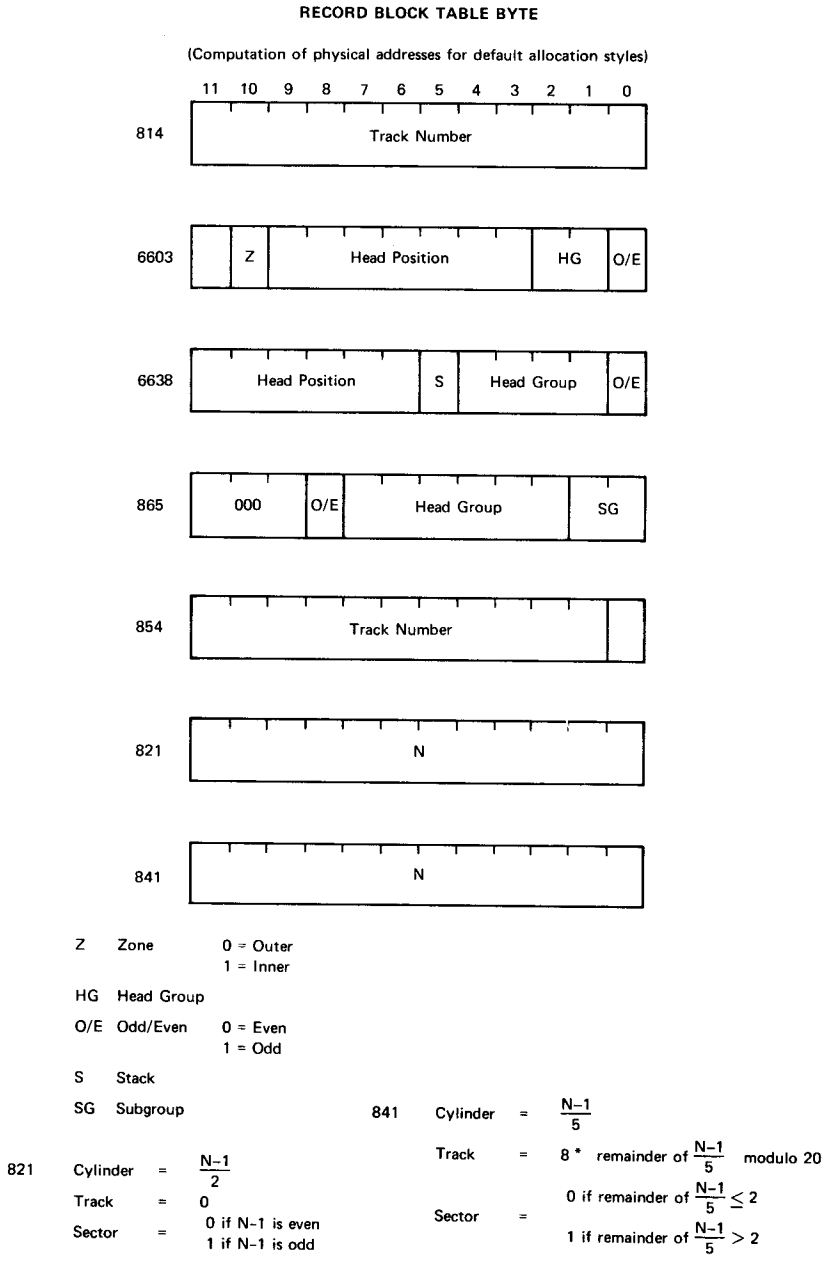


Figure 4-4. Record Block Address Format

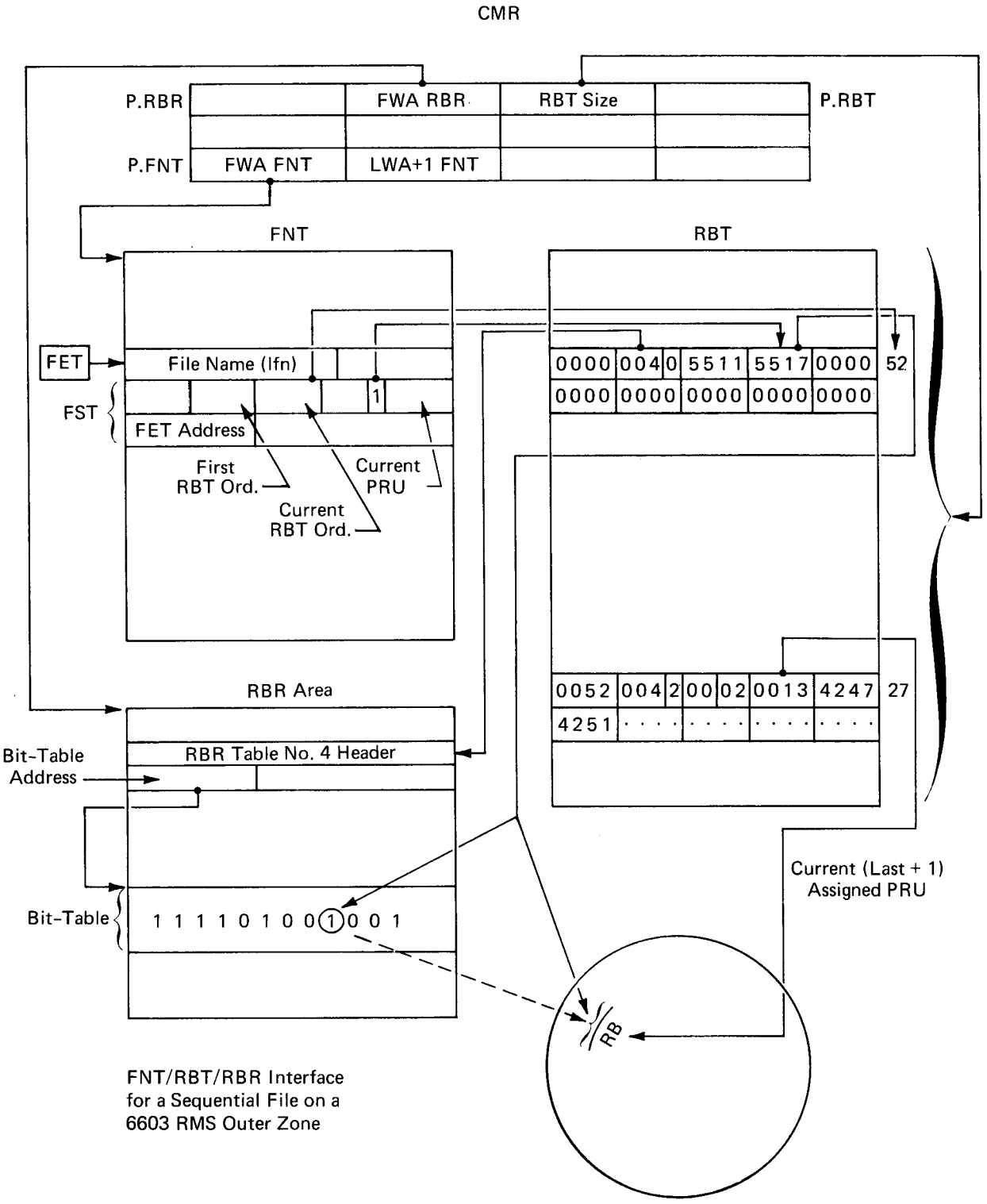
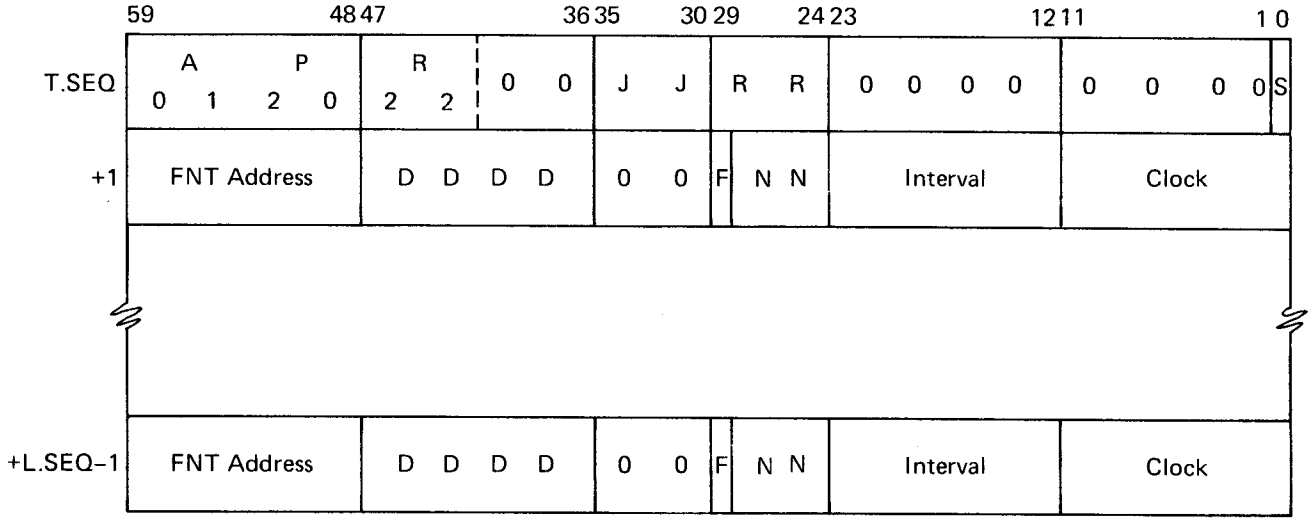


Figure 4-3. File Table Interfaces

When the file is established for a job, an entry is made in the FNT; and a pointer to this entry is placed in the job's FET for the file. The FST entry in the FNT for a mass storage file contains the RBT ordinal of the first word pair assigned to the file (27 in this example), and the current word pair assigned. It also includes the RBT ordinal, RBn byte number representing the current position of the file, and the current PRU number in the record block being accessed. In the example, the current RBT ordinal is 52 and the RBn byte is RB1 (byte 3 in the word). Byte 1 in the first word of any file RBT word pair contains the ordinal of the RBR table entry used by the file. The ordinal points to a two-word table header in the RBR area which, in turn, points to the bit string for the area on the device to which the file is assigned.

The current record block in the file is found by using the RB number in the RBn byte of the RBT word pair currently being accessed. These RBT pointers are in the FST entry; the example showing word-pair ordinal 52, byte RB1. Only 3 bits in the FST are needed to point to one of the 8 RB bytes in the RBT word pair. The RB number in byte RB1 is 5517; when interpreted according to the procedure for finding the corresponding bit in an RBR bit table, the number points to bit 47 in word 26 of bit table 1 for the device. The RB number also is translated into a physical record block address on the 6603. According to the RB address format for a 6603 disk, (figure 4-4), the physical record is in the outer zone, head position 151 of head group 3, and consists of odd-numbered PRUs. Pointers in the RBT and the FST point to, respectively, the PRU currently assigned (last + 1) and the PRU in the record block currently being accessed.

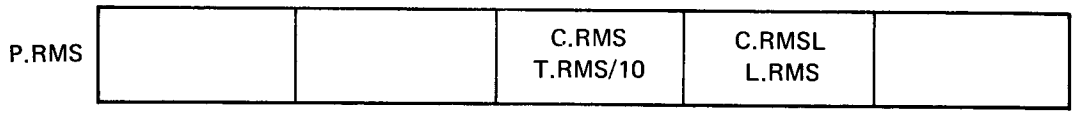
SEQUENCER TABLE



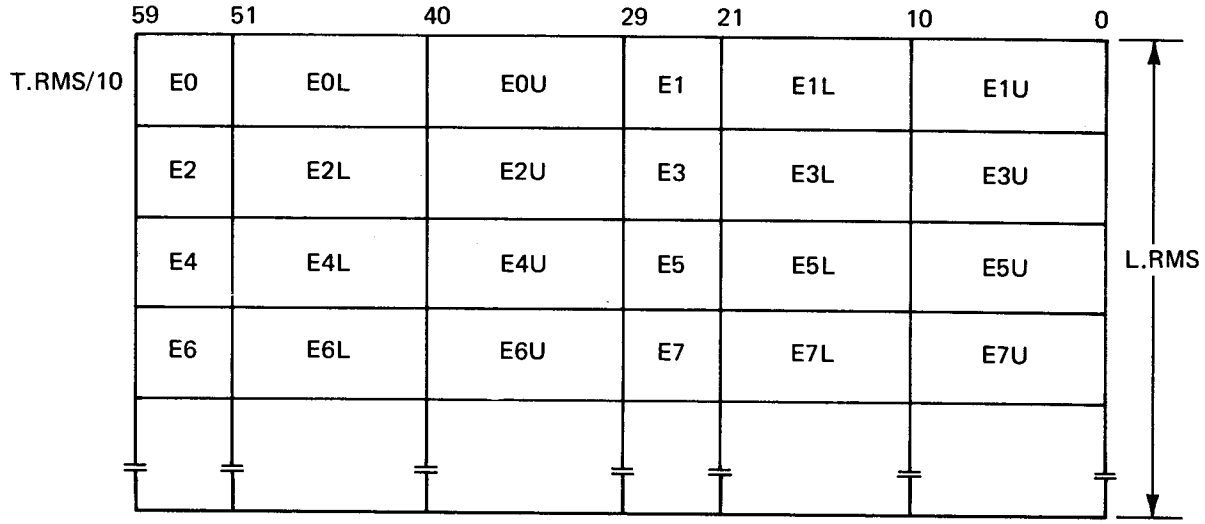
JJ = Number of jobs allowed under sequencer (L.SEQ-1)
RR = Number of jobs running under sequencer
s = Sequencer ON/OFF flag (1=ON, 0=OFF)
DDDD = Diagnostic bits

Bit 0 = CT3
Bit 1 = MY1
Bit 2 = CM6
Bit 3 = CU1
Bit 4 = ALS
Bit 5 = FST
Bit 6 = EC2
Bit 7 = ALX
Bit 8 = (Reserved)
Bit 9 = (Reserved)
Bit 10 = (Reserved)
Bit 11 = (Reserved)

F = Drop Flag
NN = Job number (5 bits)

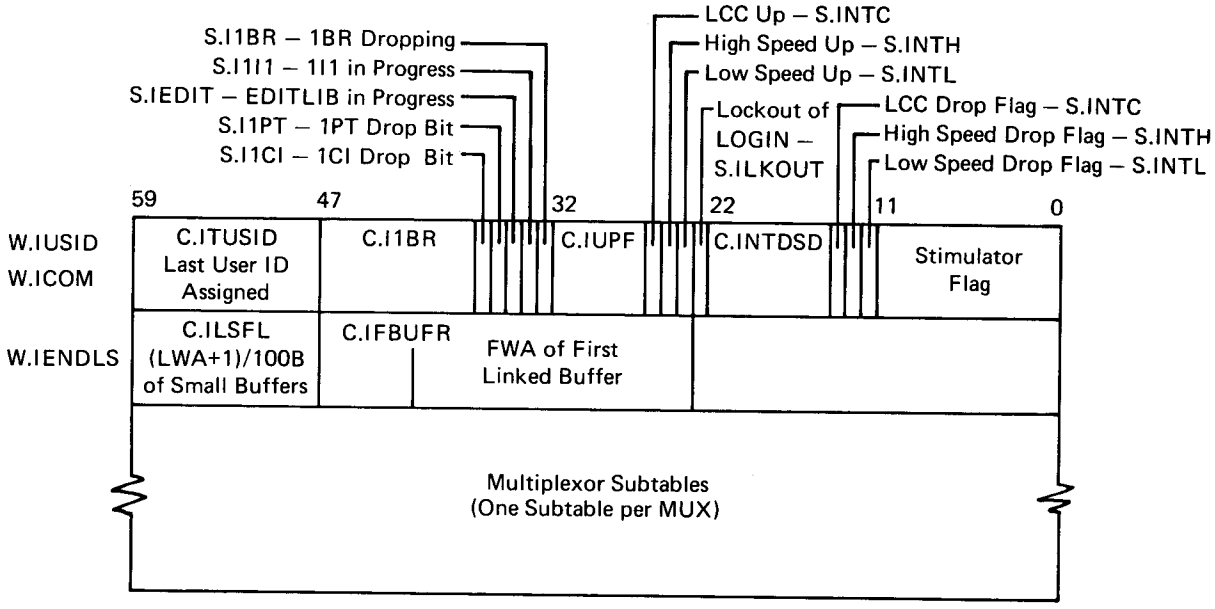


ROTATING MASS STORAGE DIAGNOSTIC TABLE

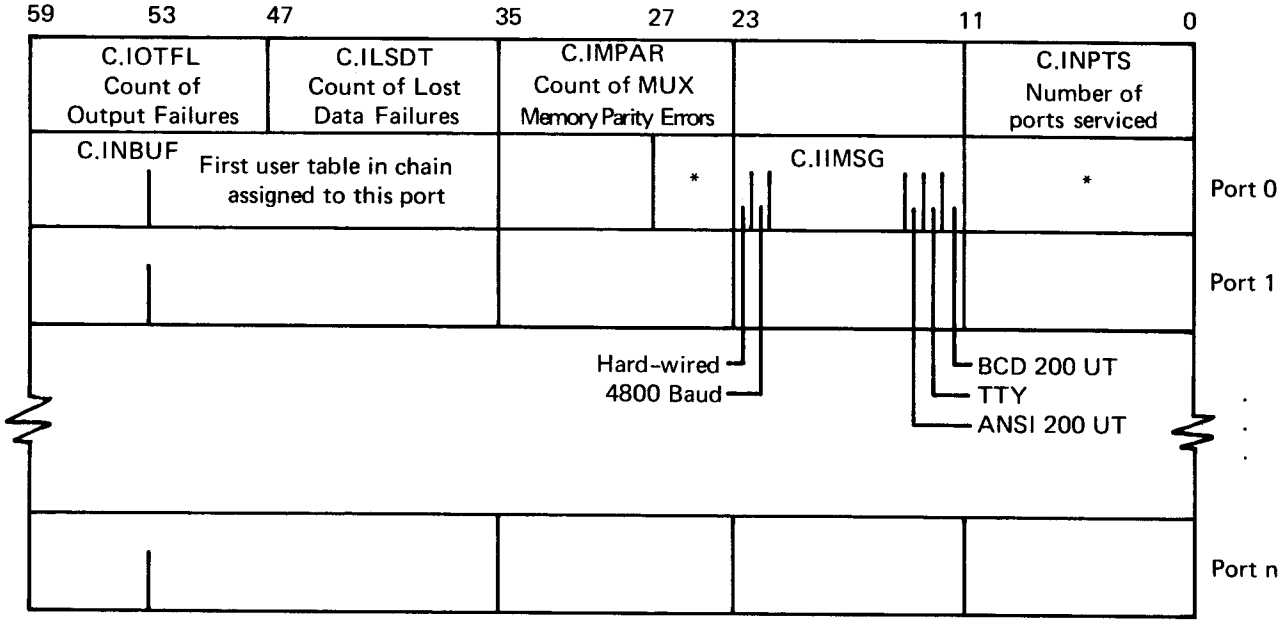


E0, E1, E2, etc. — EST ordinal of preallocated RMS device
E0L, E1L, etc. — Lower cylinder boundary of preallocated area
E0U, E1U, etc. — Upper cylinder boundary of preallocated area

INTERCOM TABLE



6671 MULTIPLEXER SUBTABLE

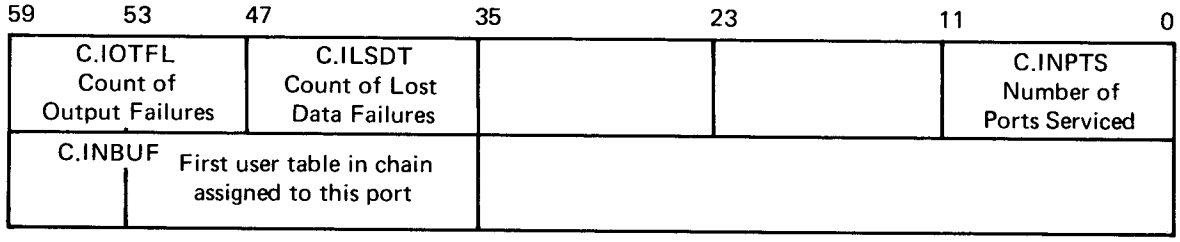


*Treated as one contiguous field (bits "15" - 0)

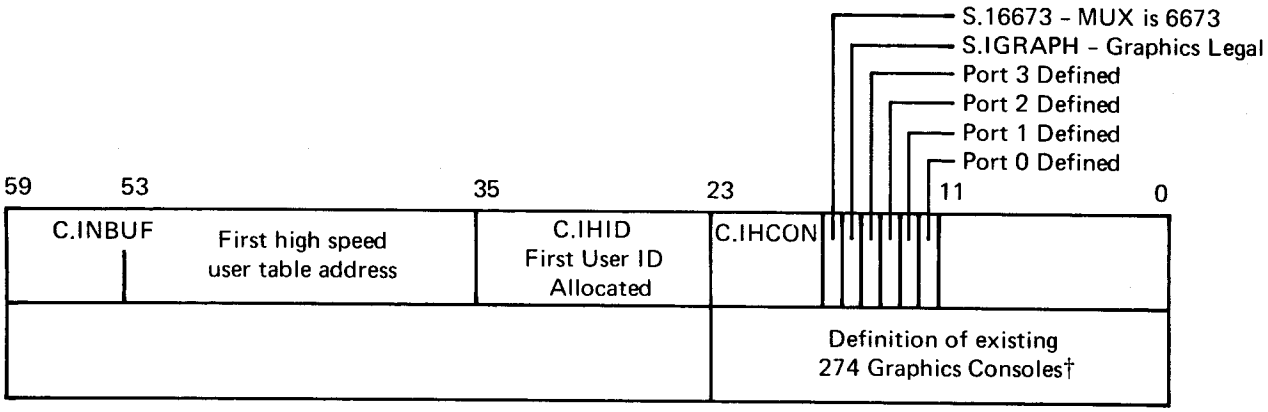
Bits "15" - 0 = 0 : dial-up line

* "15" - 0 ≠ 0 : hard-wired: site n exists if bit n = 1

6676 MULTIPLEXER SUBTABLE



6673/6674 MULTIPLEXER SUBTABLE

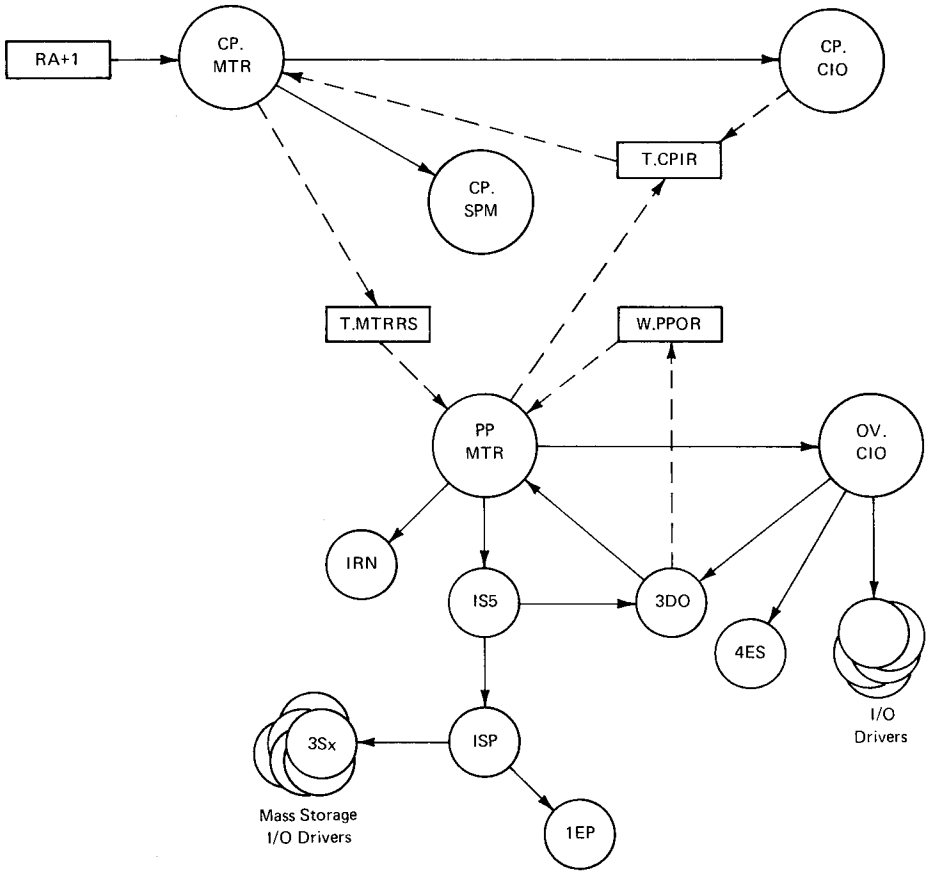


†Word 2 present if S.IGRAPH = 1

I/O PHILOSOPHY

Input and output request processing depends upon the source of each request. Active user CM programs issue RA+1 requests for I/O which are cycled through CPMTR. PP programs request I/O by placing a monitor request into their PP output register. System programs, which run at control point N.CP+1, cannot make monitor requests through RA+1. Since they run as CM service functions for PP programs, they make such requests through the output register of the PP servicing the program.

CPMTR assigns the I/O request to CP.CIO which, in turn, assigns it to the proper processor, CIO or ISP. CIO (circular input/output) processes requests for magnetic tape, Teletype, and unit record I/O; and ISP (stack processor) processes all requests for mass storage I/O.



Another I/O processor, JANUS, exists in SCOPE, but its function is limited to processing unit record I/O for the system input and output queues. The queues contain job input and output files and are related to the job processing activities of SCOPE. JANUS is discussed under the job processing section of this manual.

CIO

The circular input/output processor consists of the central memory program CP.CIO, the PP program OV.CIO and several PP I/O drivers. A system programmer can write his own input/output software, or he can have his program generate a call to CIO. Before calling CIO, the program must set up circular buffer parameters and the CIO operation code in the file environment table (FET) for the file. The relative address of the FET is placed in the CIO call.

A PP routine places a CIO call in its PP output register; PPMTR passes it through the CP input register for the CP.MTR. A CP program places a CIO call in the CP request register (RA + 1). When PPMTR accepts the CIO call, it assigns a PP and clears byte 0 of the PP output register.

When CP.MTR detects a CIO call, it passes it to CP.CIO for validation and selection of the proper CP.CIO routine to supervise execution of the function. The CIO call is then reissued via the request stack and CP.MTR to be processed by the required PPCIO driver; byte zero of the RA + 1 register is cleared. When the I/O operation is completed CP.CIO adds one to the code/status field of FET word one. As all CIO codes placed in the FET code/status field are even numbers, an odd number in that field signals completion of the operation (or that the file is not busy).

SCOPE CIO CODES (3.4)

All codes indicated by * are illegal; all reserved codes are illegal. All codes are octal for coded mode operations; add 2 for binary mode. Example: 010 is coded READ, 012 is binary READ.

000	RPHR	054	*	130	CLOSE,NR
004	WPHR	060	UNLOAD	134	*
010	READ	064	*	140	OPEN
014	WRITE	070	*	144	OPEN,WRITE
020	READSKP	074	*	150	CLOSE
024	WRITER	100	OPEN,NR	154	*
030	*	104	OPEN,WRITE NR	160	OPEN
034	WRITEF	110	POSMF	164	*
040	BKSP	114	EVICT	170	CLOSE,UNLOAD
044	BKSPRU	120	OPEN,NR	174	CLOSE,RETURN
050	REWIND	124	*		

200 Series for Special Read or Write (reverse, skip, non-stop, rewrite, etc.)

200	READC	230	*	254	*
204	WRITEC	234	REWRITEF	260	READN
210	READLS	240	SKIPF	264	WRITEN
214	REWRITE	244	*	270	*
220	*	250	READNS	274	*
224	REWRITER				

300 Series for Tape OPEN and CLOSE

300	OPEN,NR	324	*	354	*
304	*	330	CLOSER	360	*
310	*	334	*	364	*
314	*	340	OPEN	370	CLOSER,UNLOAD
320	*	350	CLOSER	374	*

400 Series (Reserved for CDC)

500 Series (Reserved for Installations)

600 Series

600	*	630	*	654	*
604	*	634	*	660	*
610	*	640	SKIPB	664	*
614	*	644	*	670	*
620	*	650	*	674	*
624	*				

700 Series (Reserved for CDC)

CIRCULAR BUFFER

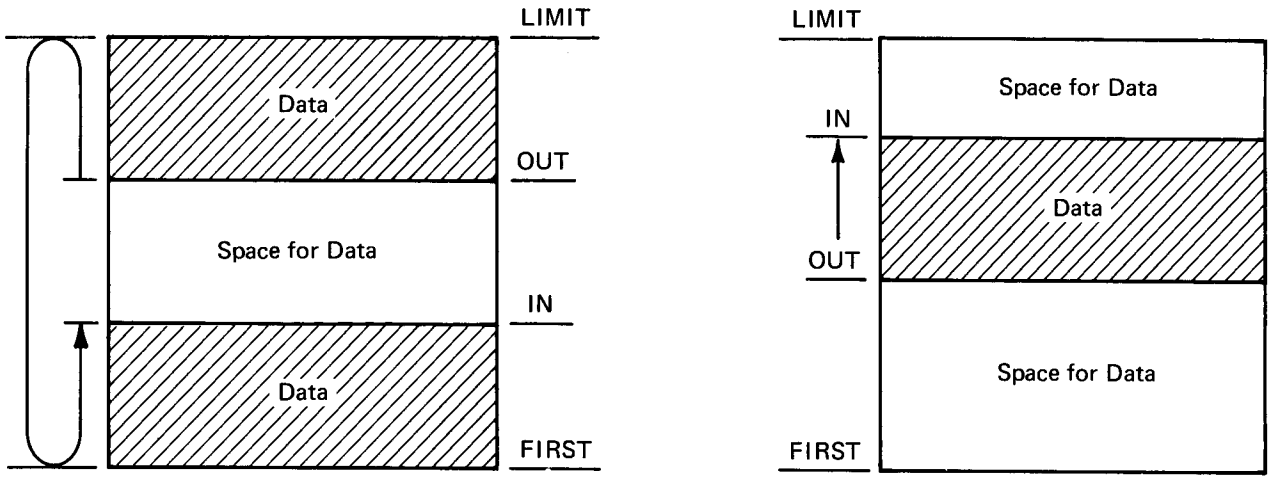
A circular buffer is a temporary storage area in central memory through which data passes during I/O operations. It is termed circular because I/O processing routines treat the last word and the first word of the buffer area as contiguous.

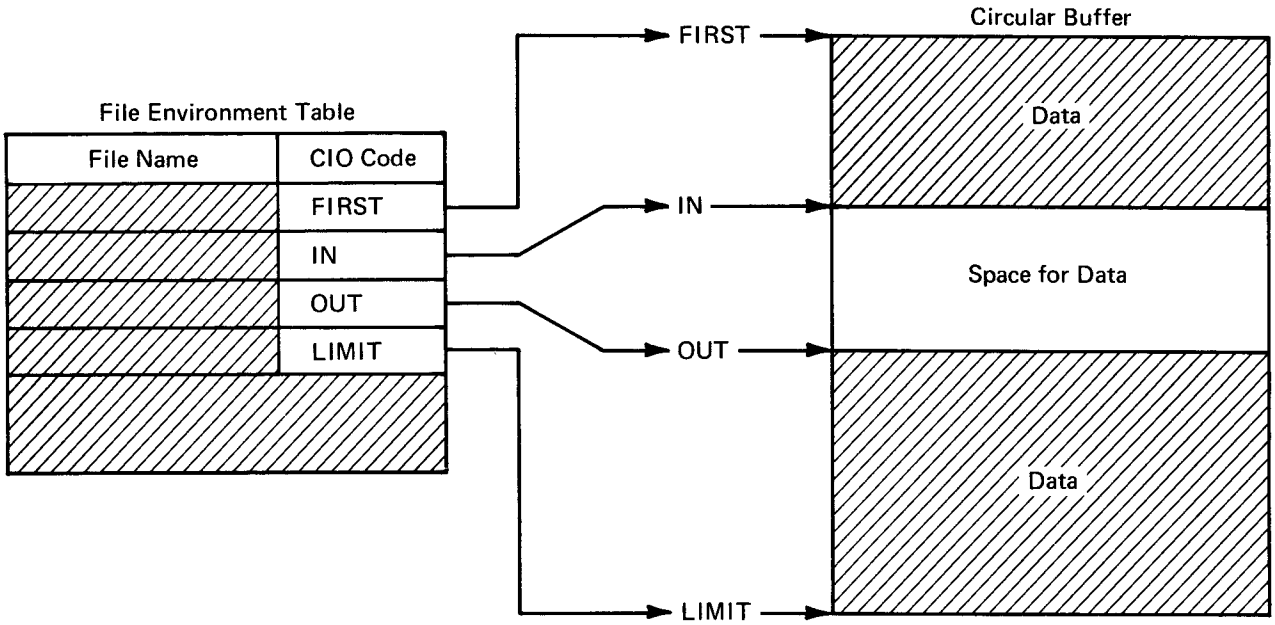
FIRST is the first word address of the circular buffer. Routines that process I/O never change the value of FIRST.

LIMIT is the last word address + 1 of the buffer area. No data is stored in this word. When LIMIT is reached, the next address accessed is FIRST. Routines that process I/O never change the value of LIMIT.

OUT is the next location from which data is removed from the circular buffer. CIO or the calling program changes OUT depending on whether the operation is read or write.

IN is the next location into which data is written. CIO or the calling program changes IN depending on whether the operation is read or write. When IN = OUT-1, the buffer is full. A partly filled buffer extends from OUT to IN-1.





The circular buffer must be at least one word larger than the length of one PRU. 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 may be transmitted only if an end-of-record is read or written.

CIO OPERATION

When MTR initiates CP.CIO to perform file I/O, CP.CIO locates the FNT for the file. If the FNT pointer in the FET is non-zero, CP.CIO checks the FNT entry indicated by the pointer to determine if the file name in the FNT entry is the same as the file name in the FET; it will also check that the file is assigned to the job control point. If the names do not match or if the FNT pointer is zero, CP.CIO will search the entire FNT for a file assigned to that job control point with a matching name. If the file is not found, CP.CIO will create a FNT entry for the file. Such files are always local and assigned to allocatable devices. Once the FNT entry is found or created, CP.CIO stores the address of the FNT entry in the FET. The FNT pointer in the FET facilitates the FNT search.

If file status is busy, CP.CIO posts the request for rescheduling and exits. Otherwise, CP.CIO checks the code field in the FET against the last code/status field in the FNT to ensure the requested operation can legally follow the preceding operation. If not, CP.CIO replaces the RA+1 call with a request for the PP program CEM which handles error messages, then reissues the RA+1 call to be processed again by CP.MTR. If the operation is legal, CP.CIO transfers the code/status field in the FET to the last code/status field in the FNT. The proper CP.CIO routine is selected to supervise function execution.

When the file is opened, CP.CIO determines if the file is on an allocatable or non-allocatable device or is ECS resident by checking the device code in the second word of the FNT. If the file is on an allocatable device, CP.CIO puts the request in the I/O request stack in CMR. The stack processor CP.SPM schedules I/O on allocatable devices; it will perform the I/O and set the completion bit. OV.CIO and its overlays process I/O requests for non-allocatable and ECS buffered files.

When OV.CIO is required, PPMTR assigns an available PP and causes OV.CIO to be loaded and initialized. Depending upon the operation, OV.CIO will call one or more of the following overlays.

Function routines:

1CL	File close
1OP	File open
1MF	Multifile positioning
1RP	Reel close
3DO	Mass storage device file open
4ES	Enter stack request (mass storage I/O)
6WM	Write error message

Tape Drivers:

1RS	Read 7-track stranger (S) tapes
1RT	Read 7-track SCOPE standard labeled tapes
1MT	Read/write other tapes (7-track)
1NR	Read 9-track stranger (S) tapes
1NW	Write 9-track stranger (S) tapes
1WS	Write 7-track stranger (S) tapes
1WI	Write 7-track SCOPE standard labeled tapes
1TF	Move tape forward (except long record (L) tapes)
2TB	Move tape backward (except long record (L) tapes)
1R9	Read 9-track SCOPE standard labeled tapes
1W9	Write 9-track SCOPE standard labeled tapes

Unit record drivers:

2PC	On line card punch
2RC	On line card reader
2LP	On line printer

Tape error recovery drivers:

1P1	Write error recovery – tape positioning
1P2	Write error recovery – erase/rewrite
1P3	Write error recovery – verification driver
1P4	Write error recovery – final driver

1RV	Initialize/terminate read error recovery
1R2	Read parity error recovery
1R3	Read error recovery – reposition/reread
1NO	Noise error recovery – read error processing
1N2	Noise error recovery – read recovery driver
1N3	Noise error recovery – skip noise record
1CS	Write CM data – 7/9 track stranger tape read recovery
1CT	Write CM data – 7/9 track SCOPE standard tape read recovery
1CR	Write CM data – 7-track other tape read recovery

If the file device code is for a non-allocatable device, PPCIO loads an I/O driver into its PP to perform the actual I/O. The overlay selected is determined by the operation requested. For example, if a user issues a request to read data from a file on a SCOPE standard format 7-track tape, CIO will call the overlay 1RT into its PP. 1RT will reserve one of the hardware channels connected to the equipment. It then issues the function codes to connect the controller and tape drive. 1RT issues functions to transmit one PRU of data from the tape drive over the data channel.

1RT accumulates the PRU of data in a PP buffer. When the entire PRU is transmitted or an end-of-record (short PRU) is encountered, 1RT picks up the pointers to the circular buffer in central memory from the FET. 1RT continues to transfer PRUs of data from the tape through the PP buffer to the circular buffer until the buffer is full or an end-of-record is encountered. 1RT updates the PRU count in the file FNT, releases the channel, sets completion bits in the FNT and FET, and drops out.

The following charts depict the logical sequence of events during various CIO tape operations.

READ

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. Exit if not enough room in buffer for one maximum size physical record.	x	x				
2. Exit if not enough room in buffer for MLRS words.			x	x	x	x
3. Read one physical record into PP.	x	x	x	x		
4. Read one physical record into CM.					x	x
5. If physical record exceeds maximum allowable, return error status DEVICE CAPACITY EXCEEDED and perform error procedures.	x	x				
6. If physical record exceeds maximum logical record size, return error status DEVICE CAPACITY EXCEEDED and perform error procedures. If a long record is encountered, excess information is discarded without notification to user.			x	x	x	x
7. If end-of-file mark was read, perform end-of-file mark procedures.	x	x	x	x	x	x
8. If noise records encountered, go to 3.	x	x	x	x	x	x
9. If parity error, perform parity procedures.	x	x	x	x	x	x
10. If end-of-tape reflective spot was encountered and tape is unlabeled, perform end-of-reel procedures.			x	x	x	x
11. If short PRU was read, strip level number.	x	x				
12. If zero length PRU was read, go to 21.	x	x				
13. When 6681 is present, convert data in PP from BCD to display code.		x		x		
14. When 6681 is present, convert data in CM from External BCD to display code.						x
15. Convert 1632 line terminator to 0000.		x				
16. Transmit data to CM.	x	x	x	x		
17. Update IN.	x	x	x	x	x	x

READ (Continued)

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
18. Fetch OUT from CM.	x	x				
19. Place in word 7 of FET the number of unused bits in the last data word.			x	x	x	x
20. If full PRU, go to 1.	x	x				
21. If last record was level 17 of tape mark, set end-of-file status.	x	x	x	x	x	x
22. Set end of record in status field of FET and exit.	x	x	x	x	x	x

READN

	S Binary	S Coded	L Binary	L Coded
1. Fetch size of MLRS from word 7 of FET.	x	x	x	x
2. Exit if not enough room in circular buffer for one logical record plus header word. Buffer size must be > ℓ(record) + 1 (header) to avoid OUT = IN when buffer is full.	x	x	x	x
3. Read one physical record into PP.	x	x		
4. Read one physical record into CM.			x	x
5. If physical record exceeds maximum allowable, return error status DEVICE CAPACITY EXCEEDED and perform error procedures.	x	x		
6. If logical record exceeds MLRS, return error status DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x
7. If end-of-file (tape mark) was read, perform end-of-file mark procedures. Go to 18.	x	x	x	x
8. If noise records encountered, go to 3.	x	x	x	x
9. If parity error, perform parity procedures.	x	x	x	x
10. If end-of-tape reflective spot was encountered on unlabeled tape, perform end-of-reel procedures.	x	x	x	x
11. When 6681 is present, convert data in PP from BCD to display code.		x		
12. When 6681 is present, convert data in CM from BCD to display code.				x
13. Transmit data to CM.	x	x		
14. Update IN in PP memory.	x	x	x	x
15. Place in buffer header word, length of record and number of unused bits in last data word.	x	x	x	x
16. Update IN.	x	x	x	x
17. Fetch OUT.	x	x	x	x
18. If last record was tape mark, set end-of-file status and exit.	x	x	x	x
19. Go to 2.	x	x	x	x

READSKP

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. Read one physical record into PP.	x	x	x	x		
2. If physical record exceeds maximum allowable (512 CM words, etc), return error status DEVICE CAPACITY EXCEEDED and perform error procedures.	x	x	x	x		
3. Read one physical record directly from tape to CM buffer, stopping without error when available buffer space is full.					x	x
4. If end-of-file (tape mark) was read, perform end-of-file mark procedures.	x	x	x	x	x	x
5. If noise records encountered, go to 1.	x	x	x	x	x	x
6. If parity error, perform parity procedures.	x	x	x	x	x	x
7. If end-of-tape reflective spot is encountered on unlabeled tape, perform end-of-reel procedures.			x	x	x	x
8. If short PRU was read, strip level number.	x	x				
9. If zero length PRU was read, go to 10.	x	x				
10. When 6681 present, convert data in PP from BCD to display code.		x		x		
11. When 6681 present, convert data in CM from BCD to display code.						x
12. Convert 1632 line terminator to 0000.		x				
13. Transmit data to CM. If record exceeds circular buffer, stop without error at buffer full.	x	x	x	x		
14. Place number of unused bits in last data word in word 7 of FET.			x	x	x	x
15. Update IN.	x	x	x	x	x	x
16. Fetch OUT from CM.	x	x				
17. If any unused space in circular buffer, go to 1.	x	x				

READSKP (Continued)

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
18. If last record was full PRU, set n = 1 and proceed to SKIPF.	x	x				
19. If L is less than 17, set L = 0.			x	x	x	x
20. If record was end of file mark (tape mark), assume level = 17.			x	x	x	x
21. If level number is less than 1, set n = 1 and proceed to SKIPF.	x	x	x	x		
22. If level number is less than L, set n = 1 and skip to first end-of-file mark (tape mark).					x	x
23. If last record was level 17, set end-of-file status and exit.	x	x	x	x	x	x
24. If last record was not level 17, return end-of-record status and exit.	x	x	x	x	x	x

RPHR

		Standard Binary	Standard Coded
1.	Set OUT = IN.	x	x
2.	Exit if not enough room in buffer for one maximum size physical record.	x	x
3.	Read one physical record into PP.	x	x
4.	If physical record exceeds maximum allowable, return error status DEVICE CAPACITY EXCEEDED and perform error procedures. If a long record is encountered, excess information is discarded without notification to user.	x	x
5.	If end-of-file mark was read, perform end-of-file mark procedures.	x	x
6.	If noise records encountered, go to 3.	x	x
7.	If parity error, perform parity procedures.	x	x
8.	If zero length PRU was read, go to 13.	x	x
9.	Transmit data to CM.	x	x
10.	Update IN.	x	x
11.	If last record was level 17 or tape mark, set end-of-file status.	x	x
12.	Exit.	x	x

WRITE

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. Exit if not full PRU.	x	x				
2. If data from OUT to IN exceeds maximum logical record size from FET, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
3. Fetch number of unused bits in last data word from FET and adjust record length. If record length constitutes a noise record, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
4. Read one PRU of data starting at OUT from CM to PP.	x	x				
5. Read data contained between OUT and IN from CM to PP. Adjust by unused bit count.			x	x		
6. When 6681 present, convert display code to BCD in PP memory.		x		x		
7. When 6681 present, convert from display code to BCD in CM.						x
8. Convert zero byte line terminator to 1632.		x				
9. Write record to tape.	x	x	x	x		
10. Write, from CM to tape, data contained between OUT and IN, adjusted by unused bit count.					x	x
11. When 6681 present, convert data in CM buffer back to display code.						x
12. If parity error, perform parity procedures.	x	x	x	x	x	x
13. If end-of-tape reflective spot, perform end-of-reel procedures.	x	x	x	x	x	x
14. Update OUT.	x	x	x	x	x	x
15. Exit.			x	x	x	x
16. Fetch IN from CM.	x	x				
17. Go to 1.	x	x				

WRITER

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. If IN = OUT, exit.			x	x	x	x
2. If PRU not full, insert level number in PP buffer.	x	x				
3. If data from OUT to IN exceeds maximum logical record size from FET, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
4. Fetch number of unused bits in last data word from FET and adjust record length. If record length constitutes a noise record, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
5. Read one PRU starting at OUT or between OUT and IN, whichever is smaller, from CM to PP.	x	x				
6. Read data between OUT and IN from CM to PP. Adjust by unused bit count.			x	x		
7. When 6681 is present, convert display code to BCD in PP memory.		x		x		
8. When 6681 is present, convert display code to BCD in CM.						x
9. Convert zero byte line terminator to 1632.			x			
10. If IN = OUT, write zero length record. Go to 12.	x	x				
11. Write record to tape.	x	x	x	x		
12. Write data between OUT and IN from CM to tape, adjust by unused bit count.						
13. When 6681 is present, convert data in CM buffer to display code.						x
14. If parity error, perform parity procedure.	x	x	x	x	x	x
15. If end-of-tape reflective spot, perform end-of-reel procedures.	x	x	x	x	x	x
16. Update OUT.	x	x	x	x	x	x
17. Exit.			x	x	x	x
18. If full PRU is not written, exit.	x	x				
19. Go to 1.	x	x				

WRITEF

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. If no data from OUT to IN, go to 23.	x	x				
2. If no data from OUT to IN, go to 19.			x	x	x	x
3. If not full PRU, insert 0 level number.	x	x				
4. If data from OUT to IN exceeds maximum logical record size, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
5. Fetch number of unused bits in last data word from FET and adjust record length. If record length constitutes a noise record, return DEVICE CAPACITY EXCEEDED and perform error procedures.			x	x	x	x
6. Fetch one PRU of data starting at OUT or data between OUT and IN, whichever is smaller, from CM to PP.	x	x				
7. Read data contained between OUT and IN from CM to PP. Adjust by unused bit count.			x	x		
8. When 6681 is present, convert display code to BCD in PP memory.		x		x		
9. When 6681 is present, convert display code to BCD in CM.						x
10. Convert zero byte line terminator to 1632.		x				
11. Write record to tape.	x	x	x	x		
12. Write data between OUT and IN from CM to tape, adjust by unused bit count.					x	x
13. When 6681 is present, convert data in CM buffer to display code.						x
14. If parity error, perform parity procedures.	x	x	x	x	x	x
15. If end-of-tape reflective spot, perform end-of-reel procedures.	x	x	x	x	x	x
16. Update OUT.	x	x	x	x	x	x

WRITEF (Continued)

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
17. Write end-of-file mark and exit.			x	x	x	x
18. If full PRU is not written, write zero length level 17 record and exit.	x	x				
19. Go to 3.	x	x				
20. If last operation was WRITE, write zero length PRU.	x	x				
21. Go to 17.	x	x				

WRITEN

	S Binary	S Coded	L Binary	L Coded
1. If OUT = IN, exit.	x	x	x	x
2. Fetch header word from OUT. Set PPOUT = OUT + 1. Set PPIN = PPOUT + number of CM words in logical record. If PPIN has passed IN, exit.	x	x	x	x
3. If data from PPOUT to PPIN exceeds maximum physical record size, return DEVICE CAPACITY EXCEEDED and perform error procedures.	x	x		
4. Adjust record length by number of unused bits in last data word (from header word). If noise record, return DEVICE CAPACITY EXCEEDED and perform error procedures.	x	x	x	x
5. Fetch data contained between PPOUT and PPIN. Adjust by unused bit count.	x	x		
6. When 6681 is present, convert display code to BCD in PP memory.		x		
7. When 6681 is present, convert display code to BCD in CM.				x
8. Write record to tape.	x	x		
9. Write data between OUT and IN from CM to tape, adjust by unused bit.			x	x
10. When 6681 is present, convert data in CM buffer back to display code.				x
11. If parity error, perform parity procedures.	x	x	x	x
12. If end-of-tape reflective spot, perform end-of-reel procedures.	x	x	x	x
13. Update PPOUT.	x	x		
14. Update OUT. Fetch IN. Go to 1.	x	x	x	x

WPHR

	Standard Binary	Standard Coded
1. If IN = OUT, exit.	x	x
2. If more than 512 words in buffer, return DEVICE CAPACITY EXCEEDED to FET.	x	x
3. Fetch data from OUT to IN, or 512 words from OUT, whichever is smaller.	x	x
4. Write record to tape.	x	x
5. If parity error, perform parity procedures.	x	x
6. If end-of-tape reflective spot, perform end-of-reel procedures.	x	x
7. Update OUT and exit.	x	x

SKIPF

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. If n = 0, set n = 1.	x	x	x	x	x	x
2. If L is less than 17, interpret as L equals 0.			x	x	x	x
3. Read a physical record.	x	x	x	x	x	x
4. If noise record encountered, go to 3.	x	x	x	x	x	x
5. If end-of-tape reflective spot encountered on unlabeled tape, perform end-of-reel procedures.			x	x	x	x
6. If record is full PRU, go to 3.	x	x				
7. If end-of-file mark encountered on unlabeled tape, assume level number equals 17.			x	x	x	x
8. If record is not end-of-file mark, assume level number equals 0.			x	x	x	x
9. If end-of-file mark encountered on labeled tape, perform end-of-file procedures.	x	x	x	x	x	x
10. If level number is less than L, go to 3.	x	x	x	x	x	x
11. Subtract 1 from n. If n $\neq$ 0, go to 3.	x	x	x	x	x	x
12. Return end of record to status. If last level number was 17, return end of file to status. Exit.	x	x	x	x	x	x

SKIPB

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. If n = 0, set n = 1.	x	x	x	x	x	x
2. If L is less than 17, interpret as L equals 0.			x	x	x	x
3. If reel is at beginning of data (either physical load point or zero physical record count), set beginning of information and exit.	x	x	x	x	x	x
4. Read one physical record backward.	x	x	x	x	x	x
5. If noise record encountered, go to 4.	x	x	x	x	x	x
6. If record was full PRU, go to 3.	x	x				
7. If this is first read backward, go to 3.	x	x				
8. Position forward over short PRU.	x	x				
9. If end-of-file mark encountered, assume level number = 17. Otherwise, assume level number = 0.			x	x	x	x
10. If level number is less than L, go to 3.	x	x	x	x	x	x
11. Subtract 1 from n. If n is not equal to zero, go to 3.	x	x	x	x	x	x
12. Exit.	x	x	x	x		

BSKP

The BKSP function is identical to SKIPB with n = 1 and L = 0.

BKSPRU

	Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded
1. If at load point or PRU count = 0, set beginning of information in FET and exit.	x	x	x	x	x	x
2. Backspace one physical record.	x	x	x	x	x	x
3. Subtract 1 from n. If n not equal to 0, go to 1.	x	x	x	x	x	x
4. Exit.	x	x	x	x	x	x

ALLOCATABLE DEVICE I/O

Most files in the system are stored on allocatable units. The system library ZZZZZ04 is stored on an allocatable unit known as the system device. Each time a non-resident CP program or a PP overlay is to be loaded from that library, I/O must be performed on the system device. The job and system dayfiles and the CE error files are stored on allocatable units, as are all input and output queue files and all files created by CIO.

A request for I/O on a mass storage allocatable unit must be placed in a table, called the request stack. The stack is searched, and the request which will require the minimum amount of overhead to access the data is chosen. Overhead involves switching head groups on the disk or physically moving heads. By using a priority-incrementing scheme for scheduling disk I/O, overhead is kept to a minimum.

All mass storage units are connected to controllers which are connected to hardware channels of the computer. For some disk devices, the controller and the disk unit form one piece of equipment. In most cases, however, the controller and the disk are physically separate units. All mass storage units connected to a single controller must be of the same type.

READC

The READC function is intended primarily for system use with mass storage files. Since READC uses inter-sector time to the maximum while reading high-speed mass storage devices, it does not include checks for erroneous programming and control words. READC would only be used by system programmers.

READC lfn,recall

READC transmits PRU's continuously to the circular buffer, with a control word preceding each PRU. READC is a function applicable to all mass storage devices. Reading continues until:

Buffer does not have enough room for the next PRU and its control word.

An error condition occurs.

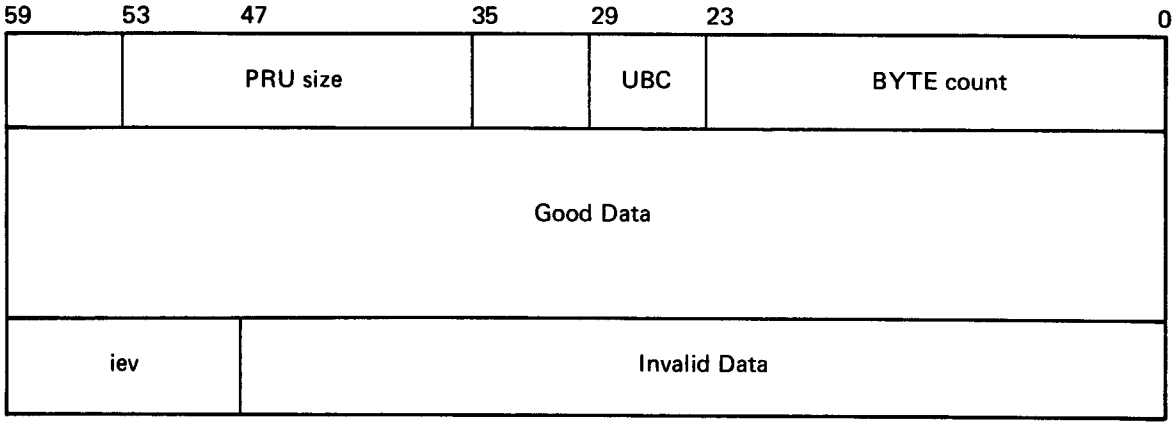
End-of-information is encountered.

Code and status on completion; where x depends on file mode:

00020X	Normal completion
03380X	Error code 33
74123X	EOI

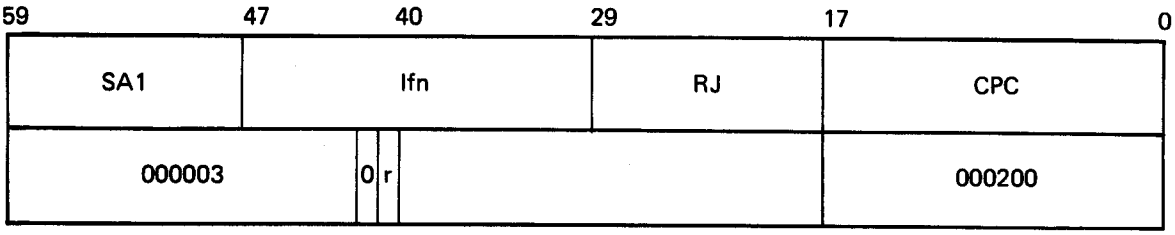
On mass storage, the same amount of data is transmitted for every PRU: the control word and one device standard PRU. The last 12 bits of the control word and the entire standard PRU length are exactly the physical data recorded on the device, including system control information.

Format of the PRU is shown below.



PRU size	Number of central memory words in each PRU on the device
UBC	Unused bit count; always 0
Byte count	Count of the number of 12-bit bytes of data. It must be equal to 5 times the number of central memory words occupied by the data. The value is recorded on disk as 12 bits, but expanded here to 24 bits.
lev	SCOPE logical record level number. If the byte count divided by 5 is less than the PRU size, a short PRU is involved.

The READC macro generates the following code:



READLS

The READLS function is applicable only to mass storage files. READLS reads several random records into the file circular buffer according to the list of direct access addresses provided by the user. No information in the buffer will reveal boundaries. This function should be used by system programmers only.

READLS lfn, recall

Before READLS is called, bits 0-17 of the ninth word of the FET should be set to the address of the list of addresses to be read. Since this field is the OWNCODE error exit field under other circumstances, the EP bit (bit 44 of the second word of the FET) should be set to 0.

Reading continues until:

List of addresses is exhausted.

End-of-information is encountered while reading a record.

The buffer is full.

An error condition occurs.

The request is discountinued for device repositioning.

Code and status on completion, where x depends on file mode:

1v022x	Request terminated on end-of-record level lv
00021x	Request terminated in middle of logical record
74023x	End-of-file encountered
74123x	End-of-information encountered
0ee21x	Error code ee occurred

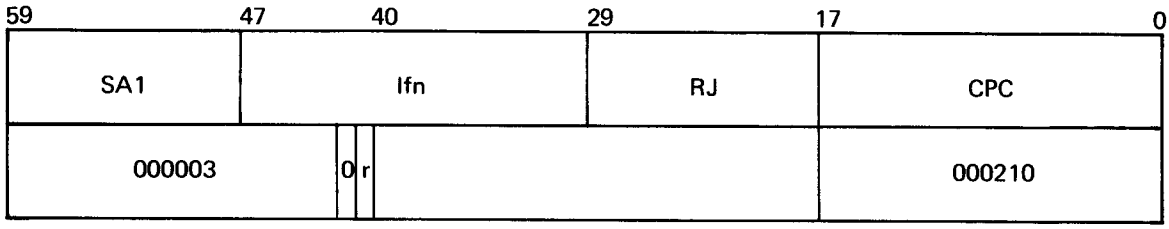
The address pointer is updated by the system when a READLS terminates so that the function can be reissued by the user without the user changing the pointer. The updated pointer will reflect the next record to be read. If reading stopped in the middle of a record, the pointer will reflect the next position to be read.

The words in the list of addresses to be read can have one of two formats, but the format of the entire list must be the same. A word of all zeros must terminate the list.

Bits 36-59 contain a PRU number, the same as used in SCOPE indexes. These are the numbers the system returns to the record request/return information field (bits 0-29 of word 7) of the FET when records are written on a mass storage device. Bits 0-35 are zero. A user list in this format will be converted by the system to the next format.

Bits 0-35 contain the SCOPE internal direct access address (RBTA/RBB/PRU) address RBT.

The READLS macro generates the following code:



CONTINUOUS WRITE (WRITEC)

The WRITEC function is intended primarily for system use. Since it uses inter-sector time to the maximum on high speed mass storage devices, it does not include checks for erroneous programming and control words.

WRITEC lfn,recall

WRITEC transmits PRU's from the circular buffer to a mass storage device. Each PRU in the buffer must be preceded by a control word. Writing continues until:

Buffer is empty

An error occurs

The diagram of the PRU and control word appears with the discussion of READC. Let n be the device PRU size. W must always be n. The 24 low-order bits of the control word and the n full CM words are written to the device.

On all SCOPE-type files, BC must be a multiple of 5. If BC is less than 5n, the high order 12 bits of the next CM word after the good data must be a binary level number, which constitutes the end-of-logical-record. The level number range is $0 \leq$ level $\leq$ 17 octal.

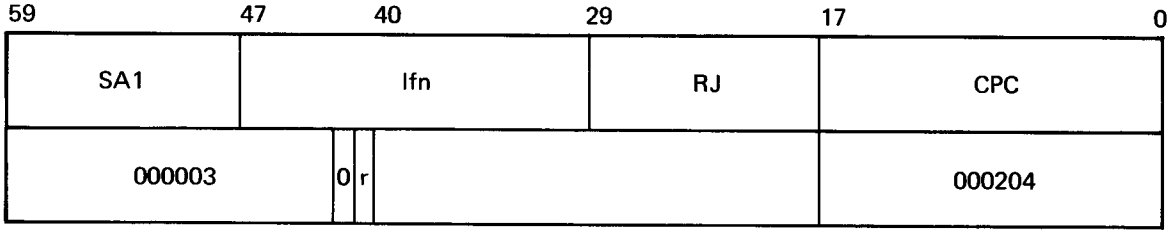
Level 17B: this is logical-end-of-file. If the file has any data at all, it must be terminated by some end-of-logical-record; the level 17 octal must appear as a zero-length logical record.

W is the count of CM words occupied in the user's buffer; must be device PRU size for mass storage, if not, serious errors will result.

BC is the count in 12-bit bytes of good data in this PRU and must be a multiple of 5. If BC/5 is less than device PRU size, then the next 12-bit byte after the good data is the SCOPE level number in binary. L must be in the range $0 \leq 1 \leq 17$ octal.

The unused bit count field (UBC) in the header word represents the number of unused bits in the last data word of a PRU. Since mass storage files are in SCOPE logical record format, data resolution is to the nearest full CM word so that the UBC field will always be zero. This field is reserved for future expansion.

The WRITEC macro generates the following code:



STACK PROCESSOR

The Stack Processor consists of the CM resident manager CP.SPM, the PP program 1SP and its various overlays. Basically, the components of the Stack Processor and their functions are:

CP.SPM	The Stack Processor Manager
OV.1S5	Examines DST ordinal and loads 1SP
OV.1SP	The Stack Processor driver supervisor
OV.1RN	Requests/releases mass storage; releases used chains to the empty chain
OV.3DO	Assigns an RB to a new or overflowing file

CP.SPM is called to enter, terminate and reissue stack requests. Setting and clearing of RBR bits and execution of the EVICT function are done by CP.SPM rather than by 1SP since device access is not required. CP.SPM is activated by an M.ICE request with a value of EX.SPM, placed in T.CPOR by PPMTR. The stack request itself is placed into the output register W.PPOR of the PP communications area (T.PPCi) of the PP making the request for CP.SPM. Consequently, CP.SPM searches for an empty stack request entry, copies the stack request from T.PPCi into the empty entry, inserts the stack request ordinal, then adds the stack request entry to the DST chain for 1S5.

PPMTR loads the basic transient program 1SP into a PP by using the stack processor loader, 1S5, which is assigned to the same PP into which 1SP is to be loaded. PPMTR also sets the active flag in the DST and places the DST ordinal into byte 4 of the PP input register for 1S5. When 1SP is loaded, it checks the same PP input register and references the DST entry to determine the name of the driver overlay to be loaded. Driver overlay 3SX will be loaded; x is a display code letter in word 1 bits 42–47 of the DST entry. The DST entry also contains channel and equipment numbers for this device. 1SP executes all requests in the stack for the equipments on the controller; all such requests have been linked into a chain by the stack processor manager, CP.SPM. 1SP assigns itself to different control points as necessary, always returning to control point zero whenever a request is completed or reissued to the stack.

The code for each driver is contained in a common deck in the program library file. The common deck and driver overlay names are listed below:

Common Deck	1SP Overlay	Device Type
RMSA	3SP	6603-I disk
RMSB	3SQ	6638 disk
RMSD	3SR	865 drum
RMSP	3SS	854 disk pack
RMSC	3ST	6603-II disk
RMSF	3SU	814 disk
RMSL	3SV	821 disk
RMSM	3SW	841 disk pack

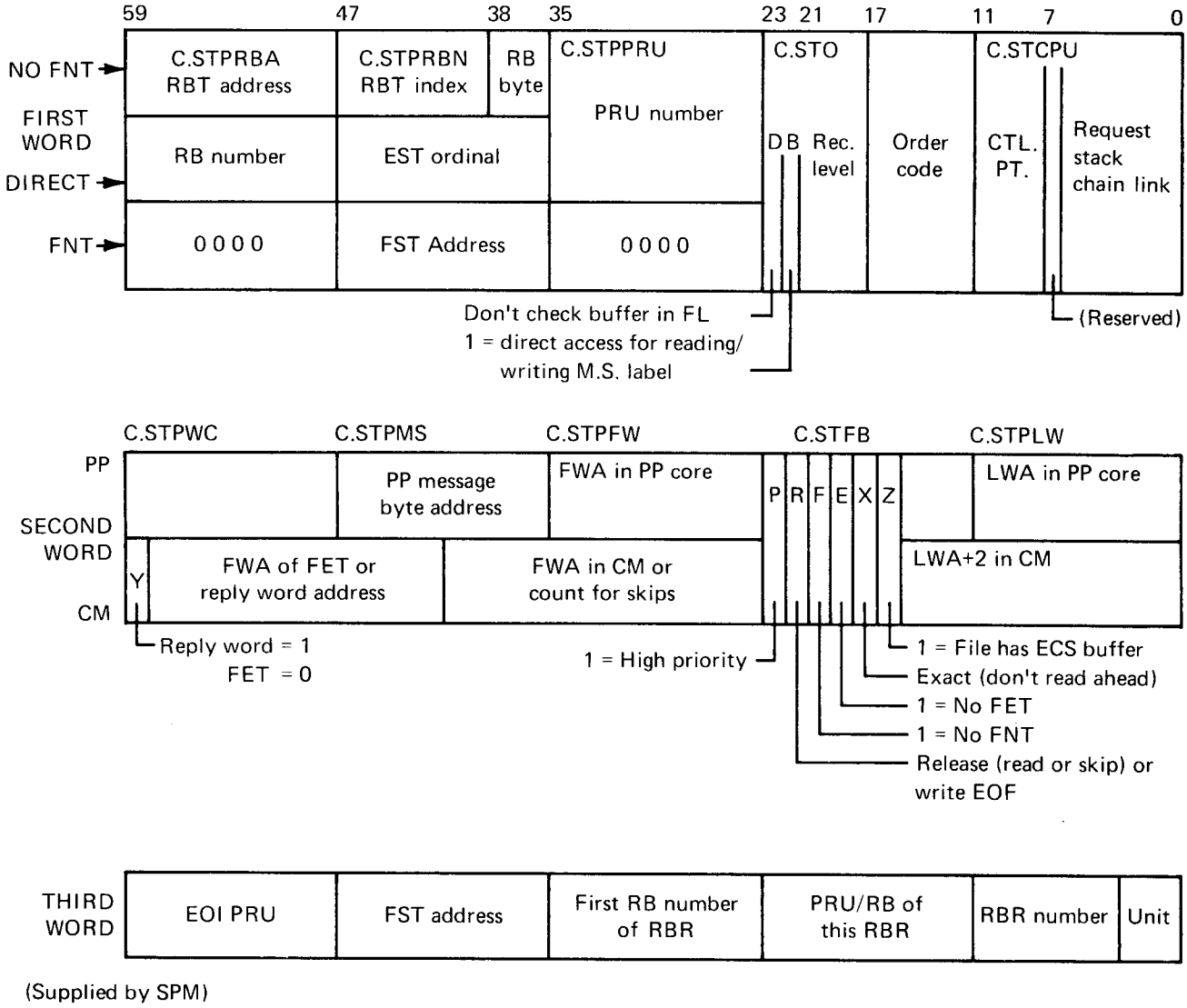


Figure 5-1. Request Stack Entry Formats

The stack processor completes its function, releases its PP, and zeros the active flag in its DST entry when all the following conditions are satisfied.

1. DST entry pointer to start of chain equals pointer to end of chain; no requests in the stack references this DST entry.

2. PP Job Queue entry count (maintained by MTR in byte 0 of CM word T.MSC) is non-zero; at least one task is waiting for an available PP.

3. Active flags are non-zero in at least two DST entries; at least one other ISP is either running in a PP or waiting to be assigned to a PP.

The first two conditions minimize unnecessary dropping-out and reloading of the stack processor; the third condition, together with suitable provisions in MTR, ensures that at least one PP contains a stack processor at all times, or one PP is reserved for that purpose. This requirement is necessary so it is always possible to load a PP overlay that resides on mass storage.

The format of the request stack entries is given in figure 5-1.

STACK PROCESSOR ORDER CODES

Order codes used in mass storage I/O request stack entries differ from those used in the CIO code/status fields of FET and FST entries. The three groups of codes correspond to the formats for the second word of a request stack entry. Order codes and standard system symbols are given below:

Central Memory Read/Write Orders

00 O.READ — Corresponds to READ macro, CIO codes 010 and 012. Read data from device to CM until (a) end of information is reached, (b) a short PRU is read, or (c) next PRU will not fit into the buffer.

01 O.RDSK — Corresponds to READSKP macro, CIO codes 020 and 022. Read as for O.READ until (a) or (b) above, or (c) the CM buffer is completely full; then change to O.SKF with N = 1 unless reading was stopped by (b) with record level ≥ request level.

02 O.RCMPR — No corresponding CPC macro or CIO code. Read as for O.READ but do not transmit first three CM words of first PRU. Used for loading programs from a system library in which first three CM words of each record contain information of interest only to EDITLIB and deadstart.

06 O.RMR — No corresponding CPC macro or CIO code. Read several records for which disk addresses are given in a table; pointer to table is in FET+8. Address of table must be in the user's field length. Read records until EOR is reached, buffer capacity is exceeded, or the addresses are exhausted.

03 O.RDNS — Corresponds to the READNS macro, CIO codes 250 and 252. Read data from device into CM buffer until (a) end of information is reached, (b) a short PRU with record level 16 or 17 has been read, or (c) next PRU will not fit into CM buffer. Used by loader when reading a relocatable binary field, since it does not stop at an ordinary end of logical record.

24	O.WCTNU O.WCTU	Corresponds to WRITEN macro, CIO codes 264 and 266. Provides non-stop writing of tapes without releasing/reloading PP between records. User's buffer must contain at least two records. Writing stops when buffer is empty or at end-of-reel condition.
20	O.RCTNU O.RCTU	Corresponds to READN macro, CIO codes 260 and 262. Provides non-stop reading from tapes without releasing/reloading PP between logical records. User's buffer must provide space for at least two records and their header words.
04	O.WRT	Corresponds to WRITE macro, CIO codes 014 and 016. Write data from CM to device until CM buffer contains less than a full PRU.
05	O.WRTR	Corresponds to WRITER macro, CIO codes 024 and 026. Write data from CM until CM buffer is empty, ending with short PRU (zero-length if necessary) with level number specified in request. If EOF flag bit is set, corresponds to WRITEF system macro, CIO codes 034 and 036. Same as above, but short PRU is followed by zero-length level 17 record (logical end of file mark).

Peripheral Processor Read/Write Orders

10	O.RDP	Same as O.READ, except read data from device into requesting PP's memory.
11	O.RDPNP	Same as O.RCMPR, except read data from device into requesting PP's memory. Used for loading mass storage resident PP programs and overlays.
14	O.WRP	Same as O.WRT, except write data from requesting PP's memory to device.
15	O.WRPR	Same as O.WRTR, except write data from requesting PP's memory to device.

Positioning Orders

12	O.SKF	Corresponds to SKIPF macro, CIO codes 240 and 242. Skip forward until N short PRU's with level ≥ the level specified in the request have been read, or end of information is reached. With N = 777777, the file is positioned at end of information.
13	O.SKB	Corresponds to SKIPB macro, CIO codes 640 and 642. Skip backward one or more PRUs until N short PRUs with level ≥ the level specified in the request have been read, then move forward over the last of these. With N = 777777, the file is positioned at beginning of information (rewound).
16	O.BPRU	Corresponds to the BKSPRU macro, CIO codes 044 and 046. Skip backward N PRUs. This repositioning is by physical record units rather than logical records.
17	O.RCHN	Release allocatable storage and RBTs (processed by SPM).

The following table is a summary of stack processor orders:

Octal Code	System Symbol	Order Function
00	O.READ	Read into central memory
01	O.RDSK	Read-skip into central memory
02	O.RCMPR	Read into central memory, drop first 3 CM words
03	O.RDNS	Read nonstop
04	O.WRT	Write from central memory
05	O.WRTR	Write EOF/EOR from central memory
06	O.RMR	Read multiple records to central memory
07		Not currently defined
10	O.RDP	Read into PPU memory
11	O.RDPNP	Read into PPU, drop first 3 CM words
12	O.SKF	Skip forward
13	O.SKB	Skip backward
14	O.WRP	Write from PPU memory
15	O.WRPR	Write EOF/EOR from PPU memory
16	O.BPRU	Backspace n PRUs
17	O.RCHN	Evict
20	O.RCTNU O.RCTU	Read nonstop (comparable to tape READN)
24	O.WCTNU O.WCTU	Write nonstop (comparable to tape WRITEN)

STACK PROCESSOR/SYSTEM INTERFACE

The system tables, system routines, and MTR functions used by the stack processor are described in this section.

TABLES

Control Point Areas: Control point error flag, storage move flag, RA, and FL. The stack processor accesses but never changes these fields.

DST: All fields of the DST entry whose ordinal is placed by MTR in the 1S5 input register are used. MTR changes the first word, and 1SP changes the second word. In other entries, only the active flag is checked.

EST: Mass storage flag, unloaded flag, off flag, and DST ordinal are checked but not altered.

FET: Code and status field in first word, error processing flag in second word, and IN and OUT pointers in third and fourth words are accessed. Code and status is marked busy (even value) before a request enters the stack and is marked complete (off value) when the request is executed.

FST: RBT/RB/PRU position pointers in first word, code and status field in second word are accessed. The code/status field has been processed the same as for the FET.

RBR area: All the first header word, the EST ordinal, and available RB count bytes in the second header word are used. 1SP assigns record blocks for a write request by searching the RBR table for available bits. When the request is terminated or re-issued, SPM sets the corresponding bits in the RBR for all record blocks assigned for the write operation. SPM also clears RBR bits when record blocks are released and updates the available RB count.

RBT: All fields. The pseudo channel CH.RBT is reserved only when RBT word pairs are being removed from the RBT empty chain.

R.CPRA and R.CPFL (PP Resident): CM reference address and field length, in 100-word block increments, for the control point to which 1SP is currently attached.

R.STBMSK (PP Resident): Contains appropriate mask when calling R.STB; always returned to 7700 octal, its normal value.

T.MSC (CM Resident): PP job queue entry count is looked at but not changed by stack processor.

SYSTEM ROUTINES/PROGRAMS

PP Programs

1SX	Stack Processor Auxiliary program is called by MTR request for tasks that 1SP cannot handle or does not have time to do. For example, 1SP does not issue dayfile messages, because if the dayfile buffer is full, 1SP and MTR would loop endlessly waiting for each other.
MTR	System Monitor calls 1SP initially (via 1S5) when a request has been made for an inactive DST entry and performs various functions for 1SP while it is processing the request.

PP Resident Routines:

R.DCH	Releases a channel reservation.
R.IDLE	Entered when 1SP releases its PP.
R.MTR	Used for all MTR functions other than reserve or drop channel.
R.TAFL	Terminates access to the control point field lenght. When necessary, it is used to interlock storage moves during execution of a request, and when a request is terminating (except at control point zero) to switch 1SP back to control point zero.
R.OVL	Loads driver overlay 3SX.
R.RCH	Reserves a channel.
R.STB	Inserts controller equipment number into device function codes and channel number into I/O instructions.
R.TFL	Computes an absolute CM address from one that is relative to a control point's RA, and checks whether or not a relative CM address is within the control point's FL.

Monitor Functions:

M.DPP	Releases PP assignment.
M.ICE	Used by the ECS driver overlay 3SX to initiate CP.ECOVL and by 1SP to initiate stack processor manager (CP.SPM).
M.RCH	Used (rather than R.RCH) with zero in byte 4 to reserve pseudo channel CH.RBT only if it is immediately available.
M.RPJ	Used for calling 1SX to another PP.
M.KILL	Used when a bad monitor request has been made.

STACK PROCESSOR ERROR CONDITIONS

This section describes the error conditions that can be detected by the stack processor. In some cases, the action depends on debug mode. With IP.DEBUG = 0, these conditions are treated similarly to other errors: an error code is placed in the code and status field of FET and FST entries and the control point is aborted if error processing (EP) bit in FET is zero. With IP.DEBUG $\neq$ 0, an invalid MTR function is issued with the 1SP output register having 77 in byte 0 and an error code in byte 1.

END OF INFORMATION

Puts 01 into bits 9-13 of code/status, but does not issue a message or abort control point. (Non-fatal condition.)

PARITY ERROR

A parity error is reported when any possibly recoverable device error occurs during a read or write operation. These include actual parity error, lost data, mispositioning, and dropping out of ready status during data transmission. The PRU is reread or rewritten up to 62 attempts. (3 for ECS). Whether success is attained or not, request execution continues after setting a flag. When request execution is completed, or the request is about to be reissued to the stack, the flag is examined. If the error was recovered, 1SX is called with code 03 (dayfile message RECOVERED PARITY ERROR), but this condition does not affect code/status or abort the control point. If all 62 attempts had failed, 1SX is called with code 04 (dayfile message UNCORRECTABLE PARITY ERROR), 04 is put into bits 9-13 of code/status, and control point is aborted if EP bit is zero. A request at control point 0 is not aborted.

INVALID RBR NUMBER

This error occurs when a request is processed that references an RBT containing an RBR number greater than N.RBR-1. In debug mode, issues bad MTR request (77,05). Otherwise, calls 1SX with code 05 (dayfile message NON-EXISTENT RBR REQUESTED), puts 05 into bits 9-13 of code/status, and aborts control point if EP bit is zero.

BUFFER PARAMETER ERROR

This error occurs when a request is processed that references an FET when not all the following conditions are satisfied:

$0 \leq \text{FIRST} < \text{LIMIT} \leq$ field length
$\text{FIRST} \leq \text{IN} < \text{LIMIT}$
$\text{FIRST} \leq \text{OUT} < \text{LIMIT}$

Calls 1SX with code 11 (dayfile message BUFFER ARGUMENT ERROR), puts 22 into bits 9-13 of code/status, and aborts control point if EP bit is zero.

NOT ASSEMBLED FOR ECS

This error occurs when a request references a DST entry for an ECS device. 1SP issues a bad MTR request (code 77).

UNDEFINED ORDER CODE

A request contains order code 07. Calls 1SX with code 22 (dayfile message INVALID STACK ENTRY), puts 22 into bits 9-13 of code/status and aborts control point if EP bit is zero.

NO FET FOR O.RMR

A request contains order code 06 (O.RMR), but no FET was specified. Calls 1SX with code 11 (dayfile message BUFFER ARGUMENT ERROR), puts 22 into bits 9-13 of code/status and aborts control point if EP bit is zero.

CONNECT REJECT

A request references an I/O unit that cannot be connected or is not ready. If the control point error flag is zero, calls 1SX with code 73 (display REJECT – ee STATUS xxxx yyyy) and reissues the request with bypass count set to 3. Otherwise, terminates the request with no message. Does not apply to 6603 or 6638.

Address out of FL for O.RMR: address for table of disk addresses for O.RMR is out of field length. Call 1SX with code 22 (dayfile message INVALID STACK ENTRY), puts 22 into bits 9-13 of code/status and aborts control point if the EP bit is zero.

ECS-BUFFERED I/O

Reading and writing of RMS files is greatly enhanced by the use of ECS buffers. Such operations involve the user of a small CM buffer in the user's field length, a large user's buffer in ECS, and a double buffer in CM for system use. The folllowing describes a write sequence involving an ECS buffer; a read sequence is essentially the reverse.

The user requests ECS buffering on a file-by-file basis through the REQUEST control card or REQUEST macro. It should be noted that an installation option will automatically assign an ECS buffer for RMS file I/O. On the control card, the user includes an EC parameter in addition to the normal parameters. The parameter is written either as:

EC	for a default (IP.BUF) size buffer; or,
EC(xxxx) EC(xxxxK)	for a buffer of xxxx-thousand (8) words (K means 1000 and if omitted, and P is not specified, is assumed); or,
EC(xxxxP)	for a buffer of xxxx(8) pages.

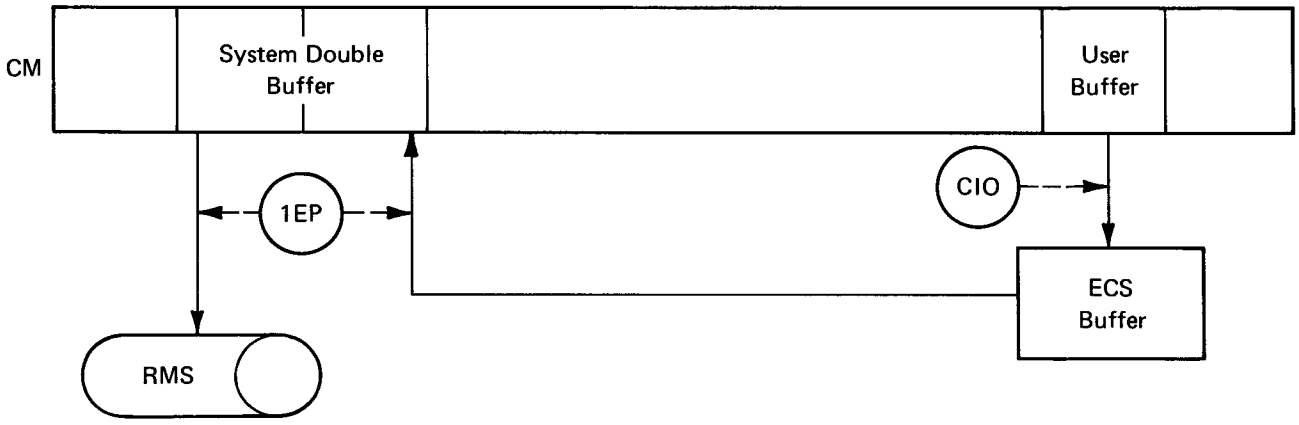
In the REQUEST macro, the user must set bit 33 to one in the second word of the parameter list and in the fourth word of the parameter list, set the buffer size into bits 0-11 and either the display code character K or P into bits 12-17.

In a write sequence, the user first puts data into his CM buffer, which needs to be only about 2000 words long, then issues a CIO call through RA+1. If he codes an XJ instruction after placing the request in RA+1, he will be exchange-jumped out of execution and CP.MTR will be started up to process the request.

CP.MTR recognizes the CIO call and passes it to CP.CIO for processing. For unbuffered files, the request is passed to the PP program CIO for processing. ECS-buffered file I/O will cause CP.CIO to perform a validity check on the FET and cause the proper CM-resident ECS driver to be activated. The data is then written directly from the user's CM buffer to his buffer in ECS.

The above process continues until the user's ECS buffer is full, at which time a stack request is generated by CP.CIO, requesting that the ECS buffer be written to an RMS device. When the stack request is processed by the stack processor 1SP, it will discover that the request is for a ECS-buffered file and will call the ECS stack processor overlay 1EP into the same PP to handle the request.

1EP requests a system double buffer to be set up in CM. The double buffer is used in ping-pong manner: half of the buffer is filled from the user's ECS buffer. While 1EP is writing this data out to the RMS file, the second half of the buffer is being filled. By the time the first half has been written out, the second half is full and 1EP can start writing it out while the first half is being filled. This ping-pong use of the system double buffer continues until the ECS buffer has been emptied and all data has been written out to the RMS file. The following illustrates the general flow of output to an ECS-buffered RMS file.



PERMANENT FILES 6

A permanent file is a mass storage file cataloged so that its location and identification are always known to the system. A rotating mass storage device is designated by an installation to contain files made permanent by system users. Any file, regardless of content, which is not already permanent, may be made permanent by explicit request. Files on magnetic tape or ECS may not be made permanent.

PERMANENT FILE FUNCTIONS

The following permanent file functions are available as system macros:

CATALOG	an existing local mass-storage file, thereby making it a permanent file.
ATTACH	a previously cataloged file to a control point.
RENAME	a file in the permanent file directory.
EXTEND	a currently attached file by making permanent an extension to it.
PURGE	a file from the permanent file directory.
SETP	an established file position to which an attached file will be set when attached.
ALTER	allows the logical end-of-file to be set to the current file position.
PERM	allows a running program to determine what permissions have been granted to an attached file.
FDB	generates a file description block required for interface with the permanent file system.

In addition, two SCOPE system macros and control cards may be used on an attached permanent file to logically detach it prior to job completion.

macro:	CLOSE lfn, UNLOAD or CLOSE lfn, RETURN
control card:	RETURN (lfn) or UNLOAD (lfn)

MACRO REQUESTS

The permanent file functions CATALOG, ATTACH, RENAME, EXTEND, SETP, ALTER and PURGE are available either as control cards or running program macro calls. The same parameters are used for both; they differ in the call format and in the capability to test status in the running program.

All permanent file macro requests share a common format:

name	**fdbaddr,RC,RT,NR**	
name	Macro name written in a COMPASS instruction mnemonic field	
fdbaddr	Address of fifth word in a file definition block (FDB)	
RC,RT,NR	Optional parameters	
	RC	Returns a code to the FDB and gives control to the requestor on non-fatal errors.
	RT	Returns a code to the FDB and inhibits permanent file queuing. (Queueing occurs when a file cannot be attached immediately.)
	NR	Specifies no recall. (All permanent file macro calls are issued in recall unless NR is specified.)

Error messages will be written to the job dayfile, unless the RT or RC parameter is specified.

PARAMETERS

The parameters described below are common to both control cards and macro functions. With the exception of lfn and pfn, parameters may be given in any order. Each is written in the form: cc = value or password where cc is a two-letter parameter code. The lfn and pfn parameters are position dependent; if one or the other is omitted, the lfn and pfn are considered to be the same string of characters. In the case when a pfn contains more than 7 characters in the parameter, the lfn will be the first 7 characters given in the pfn.

lfn	Logical file name; 1-7 character alphanumeric name (first character alphabetic) by which a file is known and referenced at a control point. Once a permanent file is attached to a control point, it is referenced by this name.
pfn	Permanent file name; 1-40 alphanumeric characters, assigned by file creator, under which a file is cataloged.
RP	Retention period in days, (0-999) specified by creator; indefinite retention indicated by 999. Installation default value is defined by the installation.

AC Account parameter; 1-9 alphanumeric characters.

PP Privacy procedure parameter; 1-9 characters, used to pass information to installation-defined procedure.

CY Cycle number (1-999) assigned by creator. Default value on initial CATALOG is 1, and on ATTACH, the highest number cataloged.

LC Lowest cycle number cataloged is referenced when value given is non-zero; default is highest number cataloged.

PW List of passwords used to establish user's access permission. Written as:

$$PW = psw_1, psw_2, psw_3, \ldots\ldots, psw_n$$

PW also is used in a CATALOG request when a new cycle is added or PUBLIC file created, and in a PURGE request in permanent file name mode; up to five passwords allowed, each 1-9 alphanumeric characters.

TK	Turnkey	
CN	Control	Password definition
MD	Modify	1-9 alphanumeric
EX	Extend	characters
RD	Read	
XR	Common	

MR If non-zero, gives read permission only which will permit read access of the file by other users.

RW If non-zero, multi-read with single rewrite will be allowed. Installation parameter can permit multi-read with multi-rewrite. If zero, and either CN, MD or EX permission is requested, exclusive access will be given.

ID Identifies file creator; 1-9 alphanumeric characters. The ID name PUBLIC is reserved for PUBLIC files, and SYSTEM is reserved for SYSTEM files.

ST Reserved for 6000/7000 compatibility

FO File structure (ordering of data) is checked so that extend and modify permissions will have meaning for direct access (DA) or indexed sequential (IS) files.

PS If non-zero, file is attached and positioned at a point established by SETP function.

EC	Allocates ECS for permanent file I/O buffer.	
	EC=K	Allocate a standard number of 1K blocks.
	EC=nnnn	Allocate an octal number of 1K blocks.
	EC=nnnnK	Same as above. If K is omitted and P is not given, K is assumed.
	EC=nnnnP	Allocate an octal number of ECS pages.

UNIVERSAL PERMISSION

An installation may define a combination of one or more permissions to be granted automatically by activating the universal permission option (IP.UP). A nine-character password is defined by the installation for such a permission combination. When this password is given on an ATTACH request, the permissions are granted.

PUBLIC FILES

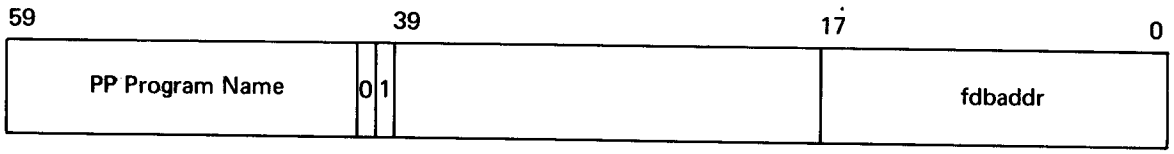
If the public permission password is correctly specified, a file may be cataloged under the ID of PUBLIC; then the user can omit the ID parameter on all permanent file requests. Attaching a PUBLIC file does not preclude the necessity of using correct permission codes.

As all cycles of a file share the same ID, when the first cycle of a file is cataloged as PUBLIC, all subsequent cycles will become PUBLIC. With the RENAME function, a PUBLIC file can be given a new owner ID, making it a private file.

MACRO REQUEST CALLS

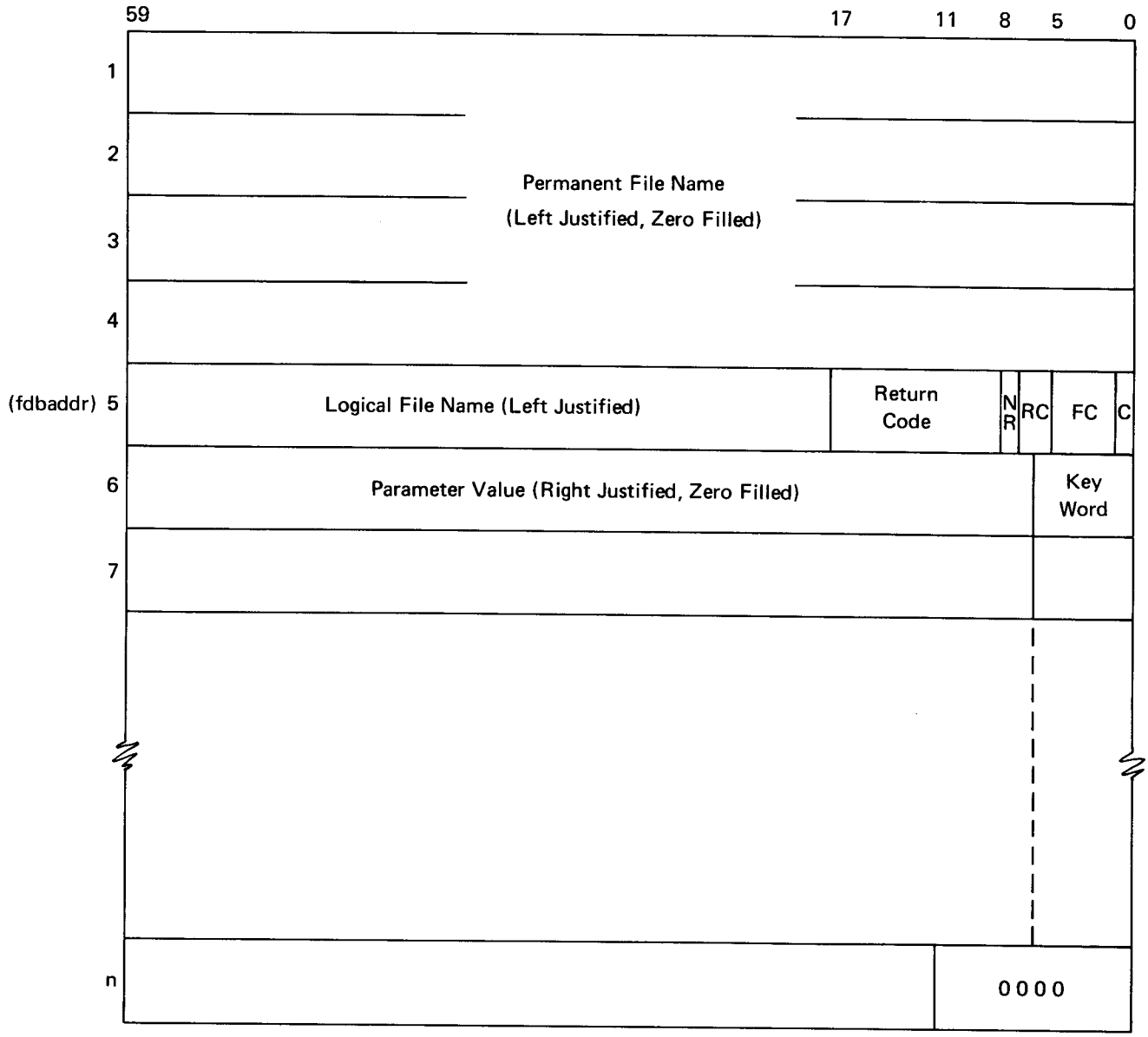
Each permanent file request macro expansion (except FDB) will generate an RA+1 call to the permanent file PP program. All permanent file macro requests are issued with recall status set (bit 40 in word 2 of the request) unless NR (no recall) parameter is present in the macro call.

An RA+1 call to a permanent file PP program has the following format:



FILE DEFINITION BLOCK

Parameters necessary for the execution of a permanent file function are contained in a central memory table called the file definition block (FDB). Error codes are returned to the user via the FDB. The FDB is generated automatically for control card calls, but it must be generated by an FDB macro for use by other permanent file macro calls. The format of the FDB follows:



pfn Up to 40 display code characters, left justified with zero fill.

lfn Logical file name of 1-7 alphanumeric characters, left adjusted with zero fill.

The macro for generating an FDB has the following format:

fdbaddr	**FDB lfn,pfn,parameters**
fdbaddr	is the symbol associated with word 5 of the FDB; it must be present in the location field.
parameters	are separated by commas, and the list is terminated by a blank. Parameters include any of the 2-letter parameter codes listed above. Parameters are entered into the FDB as they are encountered in the list.

The FDB is generated in-line during assembly whenever the macro is called.

If the RC and RT parameter is specified in a macro call, a return code will be available to the user in word fdbaddr, bits 9-17.

Word 5, bits 0-8, contain the PFM request code stored in the following format:

Bit	Content (when set)
8	NR option given
7	RT option given (implies RC given also)
6	RC option given when bit is zero
5-2	Function code (see list given below)
1	Not used
0	Function completed

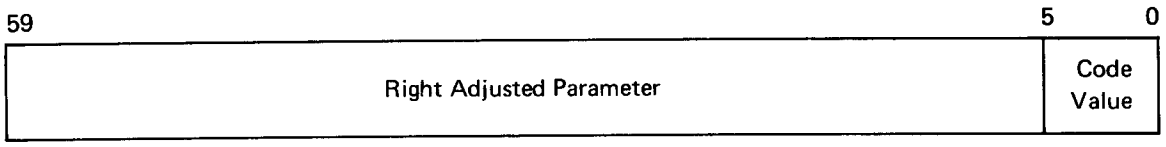
Binary Function Code:

0001	SETP
0010	ATTACH
0100	CATALOG
0110	EXTEND
0111	ALTER
1000	PURGE
1010	RENAME
1100	PERM

Request code examples (octal):

020	Catalog; return error code and control to user on non-fatal error.
130	Extend file; abort job on any error.
210	Attach file; return error code on any error and inhibit file queuing.

The parameter words of the FDB have the following format:



As shown above, parameters are stored, one per word in any order. Parameter code values are placed in bits 0-5. The format of values entered in the parameter field is indicated in parentheses in the following list.

00		end of FDB list
01	PP	privacy procedure parameter (display code)
02	RP	retention period in days (binary)
03	CY	cycle number (binary)
04	TK	turnkey password (display code)
05	CN	control pasword (display code)
06	MD	modify password (display code)
07	EX	extend password (display code)
10	RD	read password (display code)
11	MR	multi-read access only (binary)
12	SD	sub-directory (ignored in 3.4)
13	XR	control, modify, and extend password definition (display code)
14	ID	user identification (display code)
15	RN	automatic rename (ignored in 3.4)
16	AC	account name (display code)
17	EC	request ECS buffering of I/O (display code)
20-24	PW	submitted passwords (display code)
25	FO	file organization (display code)
26	PS	position setting (binary)
27		used internally for checkpoint/restart
30	PF	permanent file (display code)
31	LC	lowest cycle (binary)
32	ST	6000/7000 compatibility
33	RW	multi-access rewrite (binary)

Word 5, bits 9-17, contain a 9-bit return code which can assume the following octal values; the associated message is written to the job dayfile.

000	Function successful
001	ID error
002	lfn already in use
003	Unknown lfn
004	No room for extra cycle
005	RBTC full
006	No lfn or pfn
010	Latest index not written for a random file
011	File not on PF device
012	File not in system
013	(Not used)
014	(Not used)
015	Cycle number limit reached. Maximum value of cycle number is 999
016	PFD is full
017	Function attempted on non-permanent file
020	Function attempted on non-local file
021	(Not used)
022	File never assigned to a device
023	Cycle incomplete or dumped
024	File already attached
025	File unavailable
027	Illegal lfn
033	ALTER needs exclusive access
035	File already in system

070	PFM stopped by system
071	Incorrect permission
072	FDB address invalid

On control card requests, all errors are fatal; on macro requests, unless the RC or RT parameter is specified, all errors are fatal. If RC or RT is specified, all error codes less than 070 will result in control being returned to the user. If 070 or greater, installation parameter IP.PFABT will determine whether or not the job is aborted.

CATALOG FUNCTION

CATALOG fdbaddr,RC,RT,NR

For this request, the required parameters in the FDB are lfn, pfn and ID. If the permanent file name is unique to the ID specified, the request is considered an initial catalog. The initial catalog defines the passwords necessary to access any of up to 5 cycles that may be cataloged with the same pfn and ID. If there is no CY parameter specified, it is assumed to be 1. The following parameters are relevant on an initial catalog:

CY	Cycle number
XR	Control, modify, extend, common password definition
CN	Control password
MD	Modify password
EX	Extend password
RD	Read password
TK	Turnkey password
RP	Retention period
FO	File validity check
RW	Read with rewrite permission
MR	Multi-read access
PW	Password list
AC	Account parameter
PP	Privacy procedure parameter

If a file with the same pfn and ID has already been cataloged, the request will be considered a new cycle catalog. If a CY parameter is not specified, it is assumed to be one larger than the highest cycle. Control permission must be established to do a new cycle catalog. The following parameters are relevant on a new cycle catalog:

CY	Cycle number
PW	Password list
RP	Retention period
FO	File validity check
RW	Read with rewrite permission
MR	Multi-read access
PP	Privacy procedure parameter

If RC is specified, the user is notified of a non-fatal error condition by an error return code at fdbaddr in the FDB.

If RT is specified, the call is regarded as real-time. Specifying the RT option forces the RC option. In both cases, informative and diagnostic messages to the dayfile are suppressed.

Initial Catalog Example:

```
FDBA            FDB             LF1,MFILE,CN=Z,MD=X,TK=Y,ID=ABC
                .
                .
                .
                CATALOG         FDBA,RC
```

LF1 is assumed to exist on a valid permanent file device as a local file to this control point.

The CATALOG macro references FDBA which contains the necessary parameters to make LF1 permanent.

Since RC is specified on the CATALOG macro, control will be returned to the user on a non-fatal error; and a return code will be made available in location FDBA (bits 9-17). If RC is not specified, all errors result in termination and a diagnostic message.

New Cycle Catalog Example:

```
                CATALOG         FDB5
                .
                .
                .
FDB5            FDB             LF16,MFILE,CY=12,PW=Y,Z,ID=ABC
```

This job will add a second cycle to permanent file MFILE, created in the preceding example. File LF16 is assumed to be a valid local file on a permanent file device. The PW parameter is used to submit the passwords needed to obtain control permission. Had the initial catalog attempt aborted, MFILE would not exist; and this new cycle attempt would be processed as an initial cataloging. If successful as an initial catalog, the file would be unprotected as no passwords are defined in the FDB. An alternate form of the FDB could be used:

```
FDB5            FDB             LF16,MFILE,CY=12,PW=Y,Z,ID=ABC,TK=Y,CN=Z,MD=X
```

The above FDB would perform equally as well for a new cycle catalog because the TK, CN and MD parameters would be ignored. If initial cataloging had failed, this FDB would catalog the file with protection and the PW parameter list would be ignored.

ATTACH FUNCTION

ATTACH fdbaddr,RC,RT,NR

The ATTACH request requires the lfn, pfn and ID parameters in the FDB. The following parameters are optional:

CY	Cycle number to be attached
PP	Privacy procedure
PW	Password list
MR	Multi-read access
LC	Lowest cycle number
PS	Position value
RW	Multi-read/rewrite access
EC	ECS buffering for I/O

RC parameter is the same as for CATALOG. When RT is specified, a code of 25 is returned to the FDB if the file is currently in use. If the RT parameter is not specified, the following circumstances will cause a job issuing an attach request to be queued for the desired file:

File not available for exclusive access by requesting job

Attached permanent file (APF)table full

Archived file temporarily unavailable. The ATTACH request will cause a LOADPF job to be set up and scheduled through the tape scheduler. The job requesting the ATTACH will be swapped out until the file is available. Permanent file utility is running.

If the CY parameter is zero or not present and the permanent file has multiple cycles, the default cycle attached is the one with the largest cycle number, presumably the latest cataloged. If the CY parameter is present and that particular cycle number is not known to the system, the request cannot be honored. If both LC and CY are specified, LC is ignored and the conflict is resolved.

System evaluation of passwords establishes the type of access granted to the user for each file. Subsequent to ATTACH, the user cannot access the file in any way for which he does not have permission. For example, if ATTACH results in only READ permission, the user cannot subsequently attempt to use MODIFY or EXTEND.

ATTACH does not preclude opening the file. The success of an OPEN request depends upon the permission granted when the file is attached. If the file is attached to another control point and multi-read access is not possible, PFM will wait for access to the file.

Attach Example:

```
FDBZ            FDB             LF,MFILE,MR=1,PW=Y,ID=ABC,CY=1
                .
                .
                ATTACH          FDBZ,RC,RT
```

The permanent file MFILE is referenced again. Explicitly stating CY = 1 ensures that cycle one will be attached.

In the FDB, only the password for turnkey appears; EXTEND and READ permissions will be granted by default. In this example, the macro contains MR = 1; therefore all permissions except READ are ignored, making the file available for multi-read access.

In the macro request, the presence of RC and RT parameters would result in the octal code 25 being returned at location FDBZ if the file is unavailable.

```
                CATALOG         FDB1
                CLOSE           LF1,UNLOAD,RECALL
                .
                .
                .
                ATTACH          FDB1
                .
                .
                .
FDB1            FDB             LF1,PERMF,TK=T,MD=M,EX=E,CN=C,PW=T,ID=ABC
```

The above example illustrates several points. Assuming that local file LF1 has been created, it is cataloged as cycle 1 (by default) of permanent file PERMF. The file is protected by turnkey, control, modify and extend passwords. The PW parameter in the FDB is ignored in cataloging.

After cataloging is complete, a CLOSE/UNLOAD logically detaches the file from the job.

As illustrated, the file PERMF now can be re-attached. Although not mandatory, the same FDB is used to conserve CM space. When PERMF is attached, the default cycle number is the largest cataloged; therefore cycle 1 is the only cycle present. The PW parameter contains the turnkey password giving READ access permission by default. This example illustrates an implicit read-only attach.

The same example is shown with two FDBs:

```
FDB1            FDB             LF1,PERMF,TK=T,MD=M,EX=E,CN=C,ID=ABC
FDB2            FDB             LF1,PERMF,PW=T,ID=ABC
                .
                .
                .
                CATALOG         FDB1
                CLOSE           LF1,UNLOAD,RECALL
                .
                .
                .
                ATTACH          FDB2
```

ALTER FUNCTION

ALTER fdbaddr,RC,RT,NR

The ALTER function causes the current position of an attached permanent file (designated by lfn in the FDB) to be recorded as end-of-information in the RBTC of that file. Permissions needed to perform the ALTER function depend upon the context in which function is issued. If the current position of the file is less than the file EOI (as attached), modify permission, extend permission and exclusive access are required. If the current position is greater than the file EOI, ALTER operates similarly to the EXTEND function and extend permission is required. The user can assure exclusive access with the RW parameter on the ATTACH request.

SETP FUNCTION

SETP fdbaddr,RC,RT,NR

The SETP function causes the current position of an attached permanent file (designated as lfn in the FDB) to be recorded in the PF tables. A subsequent attach request can specify a non-zero PS parameter, causing the file to be attached at the point established by SETP. If a SETP function has not been executed for the file, or the PS parameter value is given as zero, the file is attached at the beginning of information. SETP records a position only for the attached file cycle; if the file has other cycles, they will not be affected.

The SETP function requires EXTEND permission.

RENAME FUNCTION

RENAME fdbaddr,RC,RT

Any or all information cataloged by the user can be replaced through the RENAME function. The file must be attached to the requesting job with all permissions granted. A file owner can change permanent file name, cycle numbers, passwords, and even the user ID.

In the FDB for this request, lfn is the only required parameter. The specified parameters will cause replacement of existing parameter information if they contradict the cataloged information. If they duplicate the cataloged information, the parameters are ignored.

Parameters which could result in replacement:

pfn	Permanent file name
ID	Owner identification
RP	Retention period
CY	Cycle number
TK	Turnkey password
RD	Read password
EX	Extend password
MD	Modify password
CN	Control password
AC	Account name
XR	Common password definition

The PW parameter may be specified to submit the public password if the file ID is to be renamed PUBLIC. Other parameters in the FDB will be ignored. Changing the ID, pfn, or passwords for any cycle cataloged will change all cycles. No ID, pfn, or CY changes are permitted if any of the cycles have been dumped (mode 2 dump) or archived as retrieval of such files would be impossible. RC and RT are as for CATALOG.

Rename Example:

```
                ATTACH          FDBA
                RENAME          FDBB
                .
                .
                .
FDBA            FDB             DFILE,MFILE,PW=Z,Y,X,ID=ABC
FDBB            FDB             DFILE,PFILE2,RD=W,MD=,CN=ZZ
```

Assuming that MFILE was cataloged with X, Y and Z as passwords for modify, turnkey and control, read access would be given by default when DFILE is attached as a local file. By RENAME action, the cataloged permanent file name will be replaced with PFILE2. A new password, ZZ, will replace the existing password for control permission; a read password, W, will be cataloged for the non-existent read password. The password for modify permission will be removed, and none will replace it. The owner's ID remains unchanged. Since no cycle number was given in FDBA, the cycle with the largest number will be attached; renaming will not change the existing cycle number, as no replacement is given in FDBB.

```
FDB1            FDB             LFILE,MFILE,CY=9,RD=Y,ID=ABC
FDB2            FDB             LFILE,MFILE,CY=8,PW=X,Y,Z,ID=ABC
FDB3            FDB             LFILE,MFILE,RD=Z
                .
                .
                .
                ATTACH          FDB2
                RENAME          FDB1
                .
                .
                .
                RENAME          FDB3
```

This example illustrates that for renaming purposes, the same file can be called more than once in a job. If the read password was originally cataloged as X, it is changed to Y when the file is renamed as cycle 9, and then finally changed to Z. The appearance of an identical ID parameter in FDB1 will be ignored.

EXTEND FUNCTION

EXTEND **fdbaddr,RC,RT,NR**

Local extensions can be written at the end-of-information point of an attached permanent file, and an extend function issued, thus extending the length of the permanent file. The file must be attached with EXTEND permission granted.

In the FDB, the required parameter is lfn; the extend password, if defined, must appear in the PW list. The extended section of the file will acquire the privacy controls of the permanent file.

If lfn is an indexed file, the current index will be assumed to be the only valid index for the entire file. Random files must be closed before an EXTEND request is made.

Extend Example:

```
                ATTACH          FDBX
                .
                .
                .
                EXTEND          FDBX
                .
                .
                .
FDBX            FDB             LF1,PROGLIB1,ID=XYZ
```

The program that attaches permanent file PROGLIB1 as the local file LF1 writes beyond the end of information. Assuming that no password was required for extend permission, it would be given by default. Prior to program termination, the EXTEND request would make permanent any additions written to the file.

PURGE FUNCTION

PURGE fdbaddr,RC,RT,NR

A cycle of a file can be removed from the catalog of permanent files by the PURGE function. In the macro, fdbaddr is required; RC and RT are as for CATALOG. In the FDB, the lfn is the only required parameter if the file was be attached before the purge function was issued. The optional parameters are: AC, CY, PP, LC, EC and PW. Control permission must be granted, or the job will be terminated. For the macro request, the FDB referenced may be one used at attach time; or it may be a new FDB. Only one cycle of a file may be purged at a time. When the last cycle of the file is purged, the entire permanent file name entry is removed from the directory and catalog.

A user attempting to purge a permanent file already attached must specify the lfn under which the file was attached. This purge is by local file name, and the user need provide only the lfn in the FDB.

To purge a file not already attached, the user must specify a local file name not in use at his control point. The permanent file name, ID and password list must be given.

PERM FUNCTION

The PERM function is available only as a system macro. A running program can determine if a file is a non-permanent local file or what permissions have been granted to a currently attached permanent file.

PERM fdbaddr,RC

The lfn of an attached permanent file should be given in the FDB. This macro produces a 5-bit code in the fdbaddr return code field. The bits represent the following information:

Bits 0-3

1000	CONTROL permission
0100	MODIFY permission
0010	EXTEND permission
0001	READ permission

Bit 4

1	non-permanent file
0	permanent file

A return code of zero signifies that the lfn was not in the FNT for the control point (a non-existent file) or that some other error was detected.

Perm Example:

```
FDBA        FDB         DFLN,PFILE,CY=1,PW=XXX,ID=ABDC
            .
            .
            .
            ATTACH      FDBA,RC
            .
            .
            .
            PERM        FDBA
```

Assuming the file had been cataloged with passwords required for control and modify permissions, the ATTACH request would have generated control permission by the password XXX and read and extend permissions by default. A subsequent PERM request would cause an octal 13 value to be returned to the FDB. The code indicates a permanent file is attached with read, extend and control permissions.

PERMANENT FILE UTILITY ROUTINES

The utility routines provide the capability for dumping permanent files to tape, loading dumped files from tape, transferring permanent files and tables from one mass storage device to another, and producing printed reports on the status of each permanent file.

Four routines perform these functions:

DUMPF Copies all mass storage resident permanent files to tape. The purpose is either to clear all permanent files from mass storage devices or to provide periodic back-up files for an installation.

LOADPF Counterpart of DUMPF; required to reload permanent files from tape to mass storage.

TRANSPF Enables permanent files and/or related tables to be transferred to a public RMS device. The device from which they are transferred is no longer a PF/PFD device.

AUDIT Produces a formatted output file containing statistics on all permanent files in the system.

A permanent file utility job deck should contain job card and program call cards for utility routines. The job card specifies name, priority, time limit, field length and tape requirements. AUDIT requires a field length of 35000. DUMPF requires 20000; LOADPF requires 31000; TRANSPF requires 10000.

A permanent file named DUM, having an ID of PUBLIC, and a public permission password, must be cataloged in the system before a DUMPF or TRANSPF function may be called. Control cards calling these functions must include required passwords for the appropriate file, for which an implicit attach will be performed. The LOADPF and AUDIT routines reside on the SCOPE system library; they are called with a standard program call card.

In default mode, the DUMPF and LOADPF routines request the operator to assign a labeled SCOPE internal tape written at 800 bpi density. The tape will be rewound prior to writing or reading, and it will be unloaded at job termination.

GENERAL RESTRICTIONS

1. The permanent file system must be inactive when DUMPF and TRANSPF are running. If the system is active when the first copy of the dump or transfer routine is loaded, the permanent file utility job will go into recall until all permanent file manager activity has ceased.

2. The utilities must be called by program call card.

3. Dumping of a specific unit when IP.ARCH=0 and MO=2 (all RBs associated with the files are released) will cause all permanent files dumped to be unavailable until reloaded from the dump tape by the LOADPF routine. Consequently, these files may not be attached and, therefore, cannot be purged.

4. If the device containing the PFD/RBTC is dumped, all permanent files in the system must be dumped.

5. An archived dump (MO=2 and IP.ARCH=1) sets flags in the PFD and RBTC to signal that a file is saved on a dump tape. Record blocks assigned to the file are released. Archived files can be retrieved by the ATTACH function, depending upon the IP.ARCH setting. An archived file can be purged by not reloading it.

DUMPF

To execute the DUMPF utility, the control card must inlcude passwords required to obtain read permission for the cataloged file DUM. An explicit attach should not be performed, as an internal attach occurs when the DUMPF utility is called.

To allow simultaneous execution of DUMPF routines at several control points, DUM is internally attached with MR=1 to achieve multiple access of the file. The control card may include a mode parameter and a type parameter; in addition, the passwords for file DUM must be specified. The control card parameters are order independent and are written:

DUMPF(MO=n,keyword=type,PW=passwords)

Mode Parameters	Description
MO=1	Record blocks will not be released after being dumped; the file will be available. (Default)
MO=2	Record blocks will be released as they are dumped. Depending on the archive installation parameter, the file may still be available; if not, it must be explicitly reloaded to become available.

Type Parameters	Description
UN=EST ordinal	All files residing completely or partially on the device indicated by the unit EST ordinal will be dumped.
DA=Julian date	All files rewritten, extended, renamed, altered, or cataloged after a specified date will be dumped. Date is in the form yyddd.
ID=name	All files having the ID name specified will be dumped.
DP=A	All files in the system will be dumped. (Default)
DP=C	All files created or changed after the last dump was taken will be dumped.
DP=X	All expired files will be dumped.
IN=n days	All files that have not been attached in n days are considered inactive and will be dumped.

JN = Julian date	All files that have not been attached since (and including) the date given will be dumped. Date is in the form yyddd.
TI = time	A time value to the nearest minute may be used in conjunction with the JN and DA parameters. Time is in the form hhmm.

Several copies of the dump routine can execute concurrently at different control points if all copies have the same mode and type parameters.

The TR = 7 or TR = 9 parameter can be used with any mode or type parameter to indicate whether the dump tape is 7-track (800 bpi) or 9-track (1600 bpi); TR = 7 is the default parameter.

If MO = 2 and the unit specified by the UN type parameter contains the PFD. the operator has the choice of typing GO for a full dump or typing DROP. DUM is dumped under MO = 1 but not under MO = 2.

DUMPF writes the dump to a multireel magnetic tape file having the format shown in the following figure:

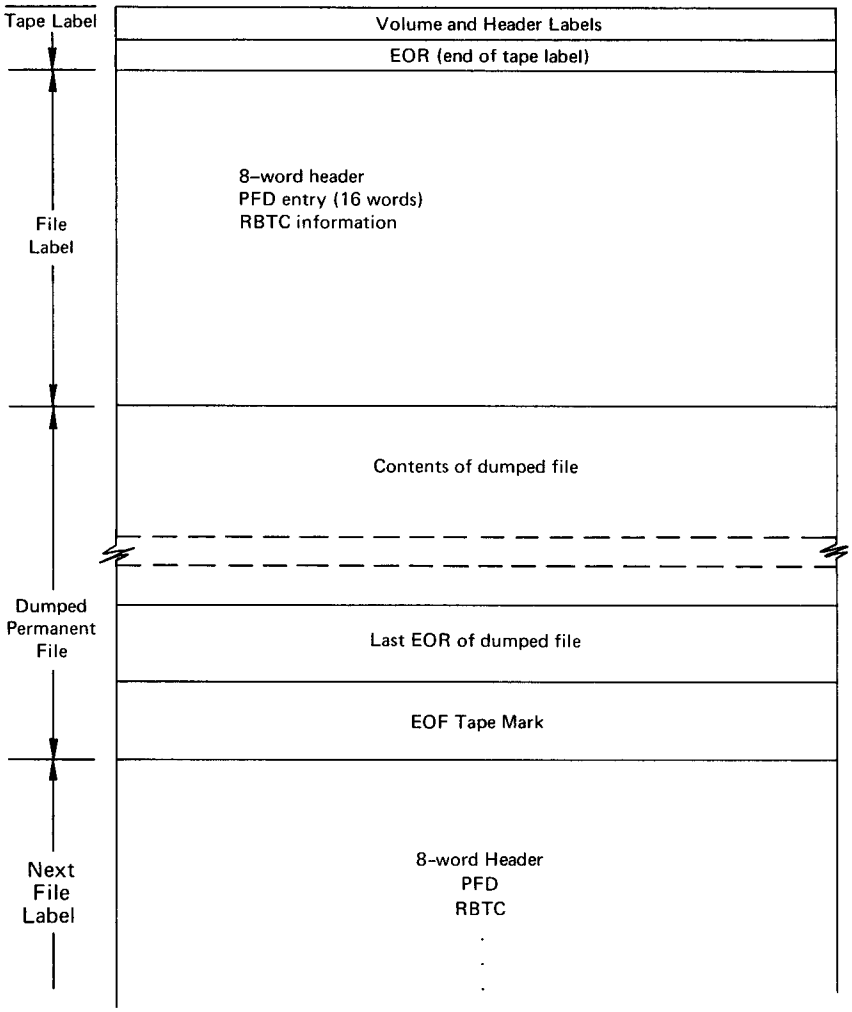


Figure 6-1. Permanent File Dump Tape Format

8-Word Header (Format is same as file header label on tape)

59	53	47	41	35	29	23	17	11	5	0
•	Δ	Δ	Δ	Δ	Δ	Δ	0	0	0	
1	0	0	0	1	Δ	Δ	Δ	Δ	0	
1	Δ	B	I	R	T	H	Δ	N	E	
V	E	R	X	0	0	0	0	0	0	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
PFD Entry of Dumped Format										
RBTC Entry up to and Including the First Word of the RBT Chain										

Information in the first 20 characters of tape label:

HDR1DUMPF TAPE- NEW

Or:

VOL1DUMPF TAPE- UNIT

FIELD LENGTH (No. of characters)	FIRST CHARACTER POSITION	FIELD DESCRIPTION	NAME OF FIELD
3	1	HDR	Label identifier
1	4	1	Label numer
17	5	DUMP TAPE OF P.F.	File label name
6	22	(blank)	Multifile identification
4	28	0001	Reel number
4	32	0001	Multifile position number
4	36	(blank)	Reserved for tape compatibility
2	40	01	Edition number
1	42	(blank)	Reserved for tape compatibility
5	43	BIRTH	Creation data
1	48	(blank)	Reserved for tape compatibility
5	49	NEVER	Expiration Date
1	54	X	Security
6	55	000000	Block count
20	61	(blank)	Reserved for tape compatibility

LOADPF

With this utility, the user can reload permanent files that have been previously dumped to tape. He can select specific files to be reloaded from a dump tape. Ordinarily, reloaded files will be restored to the device from which they were dumped. If it is not possible, the files will be loaded on another device having the same allocation type. If no such type device exists, the files will be reloaded to any other PF device in the system, if one exists.

The job deck for loading dumped files does not require a REQUEST card for the dump tape; the tape is requested by the main reload routine, LPF. The reload utility is called by the program execution (call) card in the form:

LOADPF(keyword = parameter)

Mode Parameters	Description
MO = I	Full load of files of dump tape. (default)
ID = name	All files having the specified ID name will be loaded.
PF = name	Used only in conjunction with the ID = name parameter. To be loaded, the files must have both the specified ID and permanent file name.
CY = name	Used only in conjunction with the ID and PF parameters. To be loaded, the file must have specified ID, permanent file name and cycle numbers.

Type Parameters	Description
LP = X	Expired files will not be loaded; normally, all files will be loaded.
LP = R	Replace existing versions of files. Normally, files are not replaced.
LP = O	Dump tape is not in 3.4 format.
UN = EST ordinal	Files will be loaded into a specified unit; normally, they are not loaded on a particular unit.
TR = 7 TR = 9	Dump tape to be loaded is specified as 7 (800 bpi) or 9-track (1600 bpi). Default is 7-track.

The control card may contain a mode parameter and five type parameters. If a parity error occurs during loading, the file will be flagged in the permanent file table. Several copies of LOADPF can run concurrently.

ARCHIVE FEATURE

The archive feature is automatic, and the user will be unaware of this system activity when loading is from tape. When an attempt is made to attach an archived permanent file, the operator may type n.GO which causes the file to be loaded from the appropriate dump tape. An archived file can be purged without actually loading the file.

TRANSPF

When calling the file transfer utility, the control card must contain the EST ordinals of the devices involved and the password list for the permanent file DUM.

Control card format:

TRANSPF(E1 = est1,E2 = est2,PW = password list)

est1	Indicates the device containing the permanent files and/or tables to be transferred. If it is a PFD device, the PF tables will be transferred.
est2	Device to receive files and/or tables. Must be a PF device if the permanent files are to be transferred.

If the transferred data overflows the device, the overflow will be written to another PF device. If no device is available, the file being copied will not be transferred. Permanent file tables will be transferred only if they exist on the device referred to by the est1 parameter. TRANSPF will do a destructive copy of the old permanent file tables. If the tables cannot be written to a contiguous area within the first RBR on the est2 device, the job will be dropped.

Several copies of TRANSPF can execute simultaneously at different control points while transferring permanent files, as TRANSPF specifies MR = 1 in its internal attach of the file DUM.

AUDIT

The AUDIT utility provides printed reports containing basic accounting information. AUDIT runs in two modes. Full mode produces a printed report containing all the information listed below. A partial audit produces only the items marked with *.

*Permanent file name
*Owner ID
*Cycle number
*Creation date
*Expiration date
*Date of last attach
*Date of last alteration
*Account name
*Size in number of PRUs
*Public device EST ordinal
Number of attaches
Number of extends
Number of rewrites/alters
File size (number of RBs)
Subdirectory
Archive flag (VSN printed)
Time of last attach
Time of last alteration
VSN(s) of files dumped
Loading parity error flag
Positioning flag
Random flag
SAAM flag
New version flag

The type of audit produced is determined by the type parameter given on the AUDIT control card, the format of which is:

AUDIT(keyword=parameter, ...)

AUDIT output is normally directed to the job OUTPUT file, unless another file is specified in the destination parameter on the control card. Only one mode parameter may be elected; if none is given, an audit of all files is assumed.

Destination Parameters	**Description**
LF=OUTPUT	Audit information is written to job OUTPUT file. (Default)
LF=lfn	Audit information is written to file designated by name lfn.

Type Parameters	**Description**
AI=F	Full audit files. (Default)
AI=P	Partial audit.

Mode Parameters	**Description**
MO=A	Audit all files. (Default)
MO=R	Audit archived files.
MO=X	Audit all expired files.
MO=D	Audit inactive file cycles.
MO=I	Audit incomplete file cycles.
MO=P	Audit files with parity errors.
ID=name	Audit all files having ID name.
UN=est ordinal	Audit all files on unit having EST ordinal.

Examples:

```
DUMPF(MO=1,ID=ABC,PW=X,Y,Z)
```

All permanent files with the ID of ABC will be dumped and the record blocks will not be released. The passwords X, Y, Z indicate file DUM is to be attached with permissions corresponding to the three passwords defined when the file was cataloged. The read password should be included.

```
LOADPF(UN=02)
```

Permanent files on the dump tape will be restored to the device with the EST ordinal 02.

```
TRANSPF(E1=03,E2=15,PW=B)
```

File DUM will be attached with the permission corresponding to password B which should establish read permission, at least. Permanent files and tables on the mass storage device with EST ordinal 03 will be transferred to the mass storage device having EST ordinal 15 if E1 is a PFD device and E2 is a PF device.

```
AUDIT(MO=X,AI=P,LF=YOUR)
```

All expired permanent files will be written to the output file YOUR; a partial audit will be produced.

	All Files	Archived Files	Expired Files	Files of Same ID	Files on a Certain Unit	Partial Audit	Full Audit
Owner ID	X	X	X	X	X	X	X
Permanent File Name	X	X	X	X	X	X	X
Cycle Number	X	X	X	X	X	X	X
EST ORDINAL (If on a Public Device)	X		X	X		X	X
Creation Date (Julian)	X	X	X	X	X	X	X
Expiration Date (Julian)	X	X	X	X	X	X	X
Date of Last Attach (Julian)	X	X	X	X	X	X	X
Date of Last Alteration (Julian)	X	X	X	X	X	X	X
Account Parameter	X	X	X	X	X	X	X
Number of Attaches	X	X	X	X	X		X
Number of Extends	X	X	X	X	X		X
Number of Rewrites	X	X	X	X	X		X
File Organization	X	X	X	X	X		X
Flag Archived File By Printing VSN	X	X	X	X	X		X
Length (No. PRUs determined by Inst. Par.)	X		X	X	X	X	X
Flags	X	X	X	X	X		X
Subdirectory	X	X	X	X	X		X
Length in RBs	X		X	X	X		X
Time of Last Attach	X	X	X	X	X		X
Time of Last Alteration	X	X	X	X	X		X

Figure 6-2. Table of AUDIT Outputs

Permanent File Utility Function Codes (octal):

AUDIT	166
DUMPF	172
LOADPF	174
TRANSPF	176

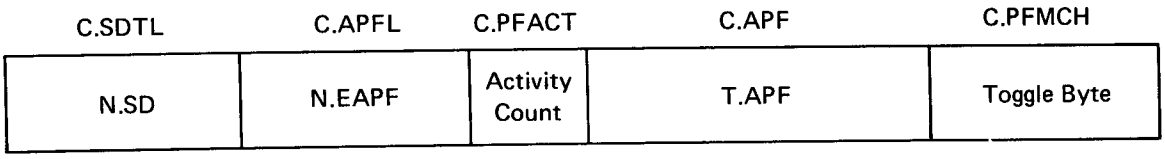
Permanent File Utility Keyword Codes (octal):

Keyword	Code	Description (...) indicates format of values in FDB
TR	57	Dump tape type 7 or 9 track (binary)
MO	60	Load or audit dump mode (display code)
UN	61	Dump, load or audit unit EST ordinal (binary)
DA	62	Date parameter for file dumps (display code)
DP	63	Dump mode parameter (display code)
IN	64	Inactive file dump parameter (binary)
E1	66	Transfer files tables from unit EST ordinal (binary)
E2	67	Transfer files tables to unit EST ordinal (binary)
LP	70-73	Load type parameter (display code)
JN	74	Date parameter for file dumps (display code)
TI	75	Time parameter for file dumps (display code)
AI	76	Audit type parameter (display code)
LF	77	Audit output file parameter (display code)
ID	14	Load by file ID name parameter (display code)
PF	30	Load by ID and pfn names parameters (display code)
PW	20-24	Passwords to attach file DUM (display code)
CY	03	Load by ID, pfn and cycle parameters

PERMANENT FILES/SYSTEM INTERFACE

Three pointer words in the CMR pointer area contain pertinent file system information as follows:

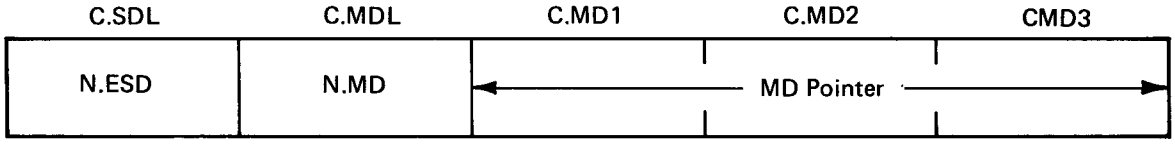
P.PFM1 (word 6 in CMR)



C.SDTL	N.SD is the number of subdirectories in the system
C.APFL	N.EAPF is the number of entries in the APF table
C.PFACT	Number of files being attached or cataloged not having APF entries
C.APF	T.APF is the FWA of the APF table (18 bits)
C.PFMCH	PF toggle byte used for interlocking disk and CM tables. Interlocks in effect are indicated by bit settings:

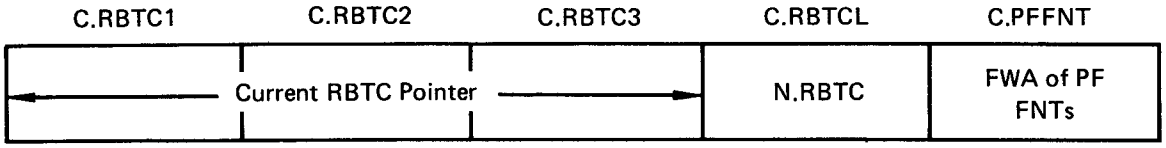
a	S.RBTCW	RBTC Wraparound
b	S.APFIL	APF Interlock
c	S.PFDIL	PFD Interlock
d	S.RBTCIL	RBTC Interlock
e	S.PFUTIL	Utility
f		Not used
g	S.TRANIL	TRANSPF Interlock

P.PFM2 (word 7 in CMR)



C.SDL	N.ESD is the number of available slots in each subdirectory (4 per PRU)
C.MDL	N.MD is the number of master directories.
C.MD1,2,3	Pointers to the master directories

P.PFM3 (word 17 in CMR)



C.RBTCL	N.RBTC is the number of PRUs in the RBTC divided by 16 decimal
C.RBTC1,2,3	Pointers to current end of information in the RBTC
C.PFFNT	Pointer to first word of permanent file directory FNT area

SYSTEM PERMANENT PACKS

An installation may choose to have its permanent file system on 854 or 841 disk pack devices; this would facilitate the transfer of the permanent file system from one hardware configuration to another without the use of the permanent file utilities. One of the packs must be declared a PFD device and the other packs declared PF devices. Deadstart has an option to provide for the addition of these packs. In either configuration, all packs must be mounted while the software system is running. The system files created at deadstart may be on an RMS device that is not a PF device; such a device, however, must be a system device.

PERMANENT FILE TABLES

There are four system files created as permanent files during system deadstart: the system dayfile, the CE error file, the edit-extension file, and the system (library) file. Each has an entry in the permanent file directory (PFD) and the RBT catalog (RBTC).

PFD and RBTC are of fixed length, and reside on the same mass storage device. At deadstart, an FNT entry is made for the RBTC as 0RBTC, and multiple entries are made for PFD as 0SD001, 0SD002, . . . , corresponding to each of the subdirectories. The RBTC and all PFD/FNT entries are attached to control point zero and entered at the high end of the FNT. This arrangement prevents an unauthorized user from interfering with the operation of the Permanent File Manager (PFM).

The APF Table is CM resident and consists of two word entries, up to a predefined size for the installation (L.APF, default of 30 decimal).

The PFD, detailed in figure 6-3, consists of a header at the beginning of subdirectory 1, followed by a number of subdirectories of equal length. The PFD header is a copy of the PFD's RBT chain for use in recovery. Each subdirectory consists of PFD entries as shown in figure 6-4.

Up to five cycles or versions of each permanent file can be maintained. The PFD entry, therefore, can have up to five pointers to the RBTC, each corresponding to an entry for a cycle. Each pointer has a byte (word 7 of PFD entry), which contains two 1-bit flags and the cycle number. The first flag, if on, signifies that this cycle has been dumped and is unavailable; the second signifies the entry is incomplete as no RBTC entry exists.

When cataloging, the user may specify the number for the cycle or permit a cycle number to be generated automatically. Unless the user specifies the cycle when attaching a file, he will get the one with the highest cycle number. Cycle numbers range from 1 to 999.

The permanent file name (pfn) can consist of up to 40 characters. The PFD entry also contains the passwords given when the file was cataloged. PFD entries are placed into appropriate subdirectories by a hashing algorithm applied to the file's ID.

The flag in word one of the PFD entry signifies that all cycles have been dumped. (used by utilities only.) The entry-in-use flag is set when an entry is built. Detailed formats of the RBTC and RBTC entry appear in figures 6-5 and 6-6.

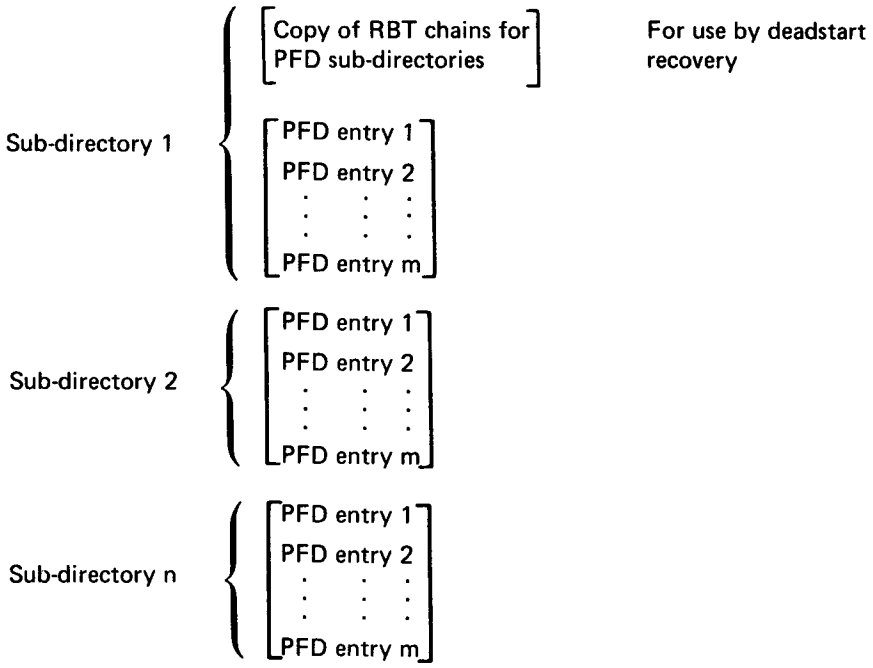
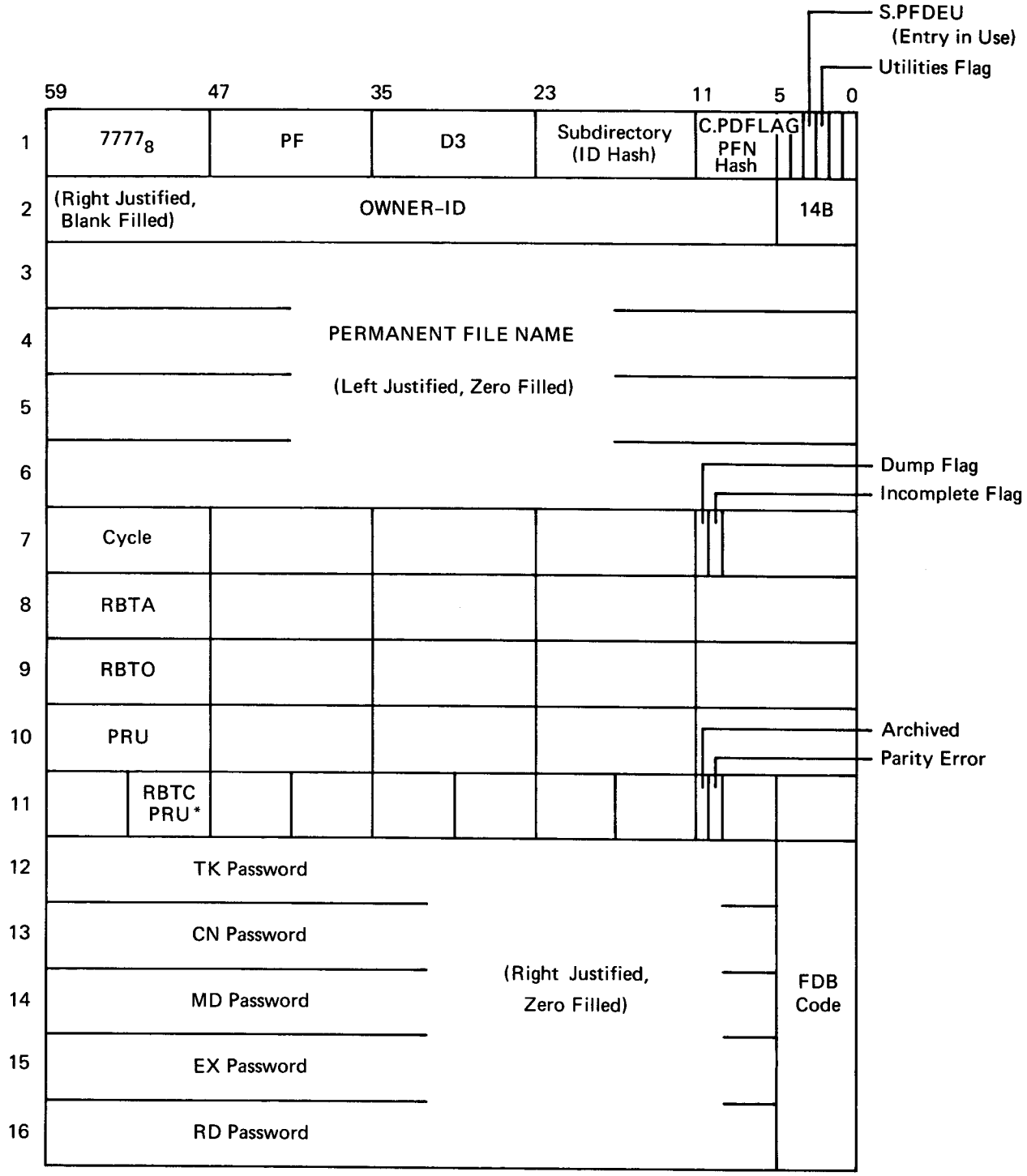


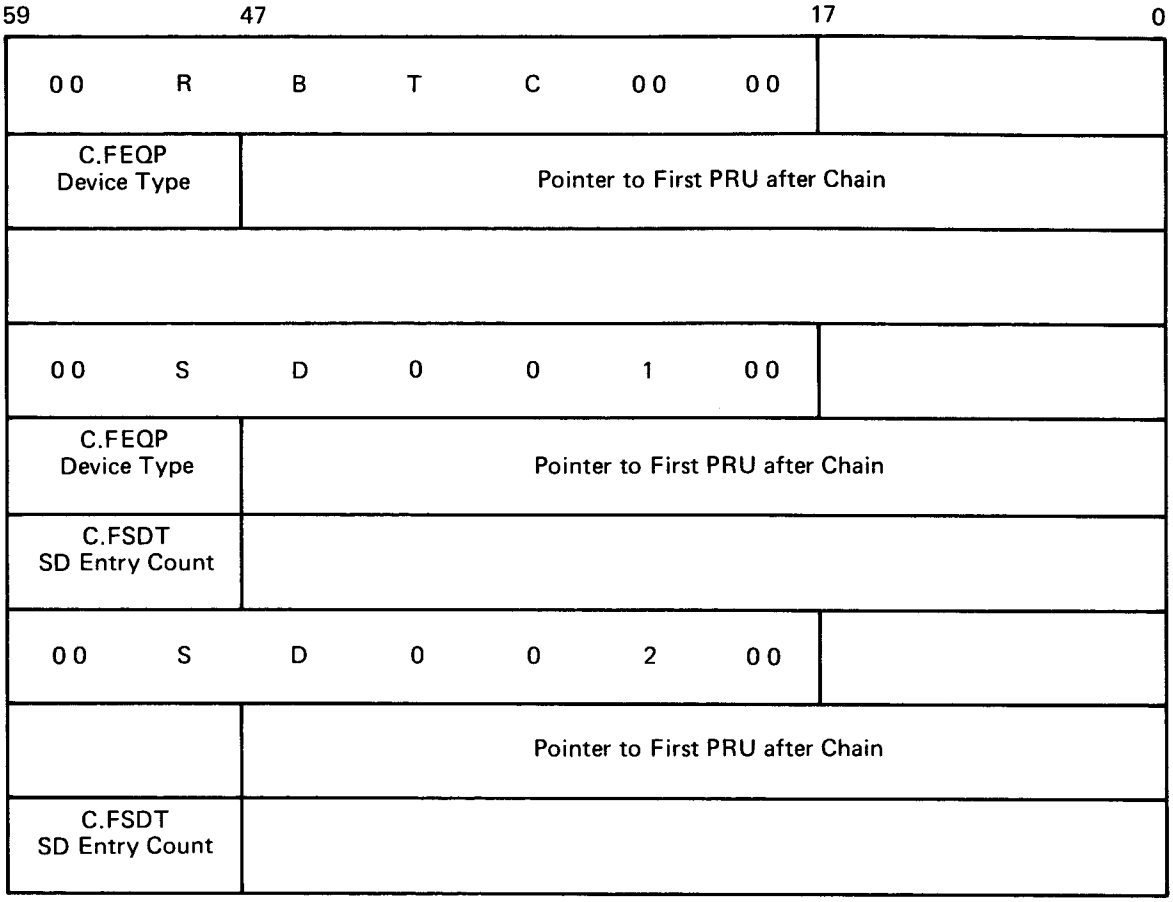
Figure 6-3. Permanent File Directory



* Byte number of RBTC PRU first word. (Word contains an index to the start of the RBTC entry in that PRU.)

Figure 6-4. PFD Entry

ENTRIES FOR PERMANENT FILE TABLES
(LAST THREE ENTRIES IN FNT)

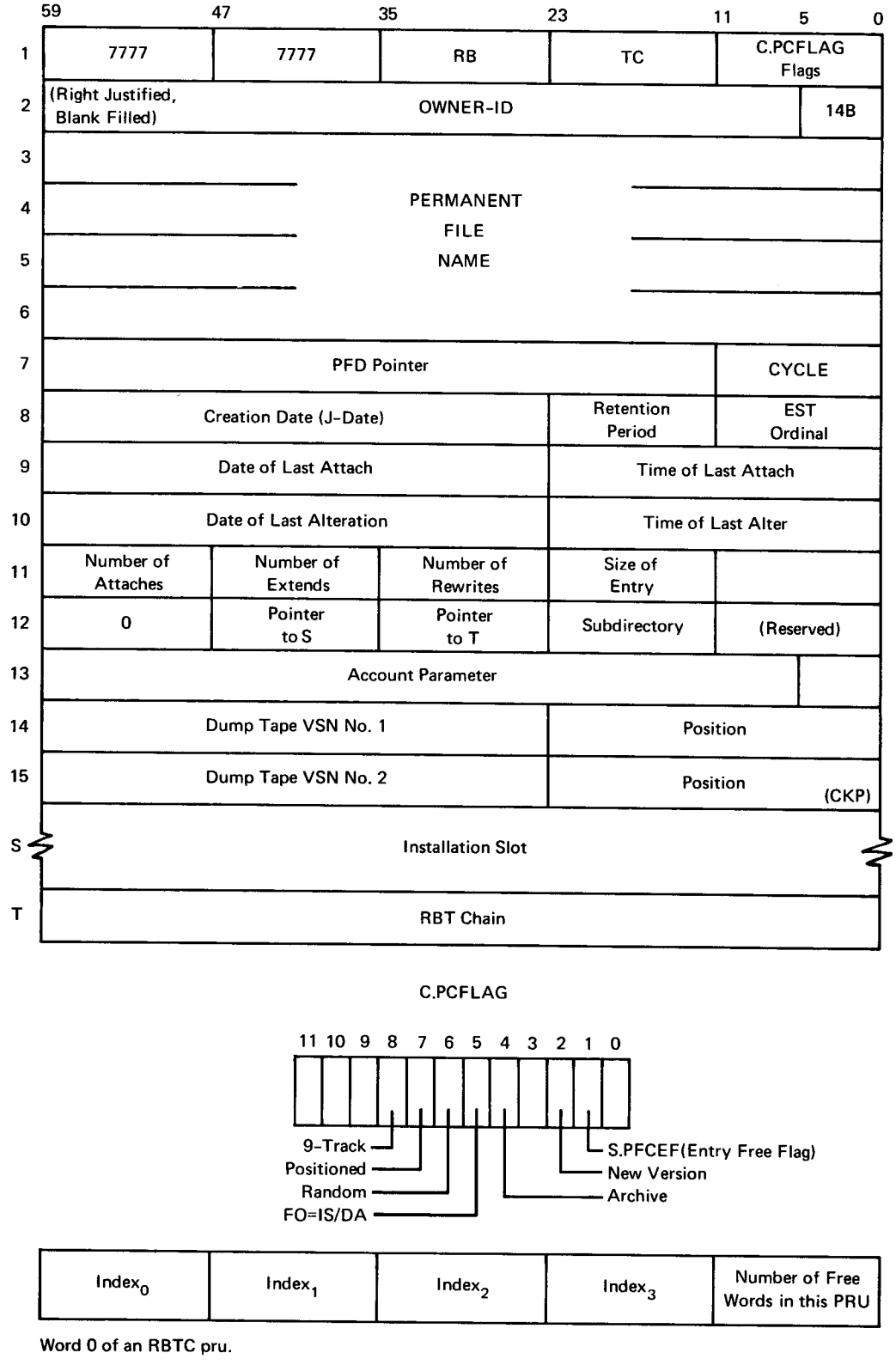


Label for RBTC
Copy of RBT

Chain assigned to RBTC for recovery purposes

RBTC entry
RBTC entry
RBTC entry
. .
. .
. .
RBTC entry

Figure 6-5. RBT Catalog



Word 0 of an RBTC pru.

Figure 6-6. RBTC Entry

The RBTC consists of a header and a number of variable length RBTC entries which are not necessarily contiguous. If an RBTC entry will not fit entirely into one PRU, it is forced to start at the beginning of the next PRU. The header is a copy of the RBT chain for the entire catalog used by deadstart in recovery. Each RBTC entry has a copy of the pfn as well as other relevant information as shown in figure 6-6. Figure 6-2 indicates the information recorded in the RBTC entry that is printed for a file audit, depending on the mode and type of the audit.

Figure 6-7 is the format of the APF entry. Whenever a permanent file is attached to a control point, an entry is made for it in the APF table. A pointer in the first word of an FNT entry for a permanent file relates that entry to the corresponding APF entry. The APF table is an interlock list which allows multi-read access without reloading the RBT chains and also controls queuing for exclusive use of the file.

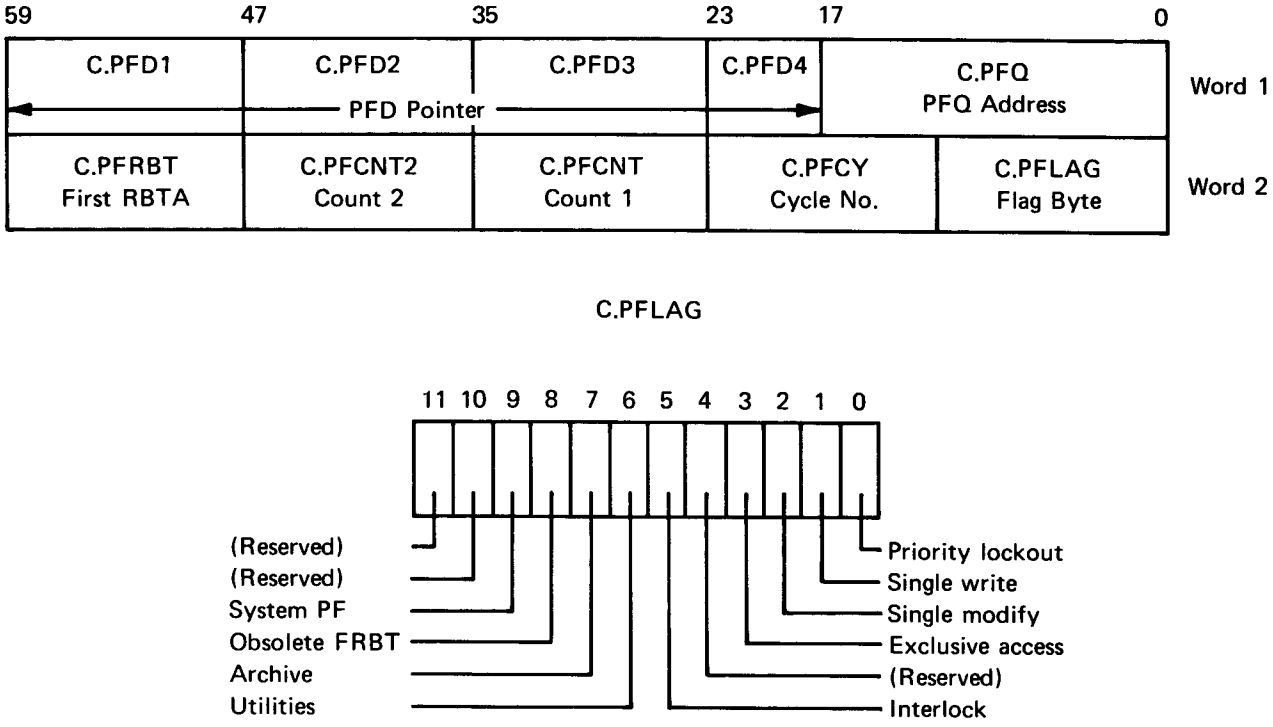


Figure 6-7. APF Table Entry

INCOMPLETE CYCLE

An incomplete cycle is a file that has a PFD entry but no RBTC entry. It can result from either a CATALOG or LOADPF job that does not terminate normally. An incomplete cycle may be attached and purged if the CY parameter is used to specify the cycle and control permission is established.

JOB PROCESSING 7

JOB FLOW

SCOPE processes jobs in the system in three sequential, independent stages: Input, Scheduling, Output. Many jobs may be in any one of the three stages.

Jobs can be loaded into the computer by reading card decks into the system using the system package JANUS. Alternately, they may be input from tape with the tape loader (1LT) or from a user terminal through INTERCOM. When the job has terminated, JANUS is assigned to process the OUTPUT, PUNCH, and PUNCHB files produced by the job.

A job comes under scheduler control after all system resources specified on the job card have been assigned to it. When a job is under scheduler control, it has an assigned job description table entry and is in one of the scheduler queues; or it has been assigned to a control point. When it is assigned to a control point, the control point status values will indicate either the type of activity or that activity at the control point has been suspended. With the job swapping/rolling features of SCOPE 3.4, several jobs may be considered to be assigned to each control point.

A job is in execution only when it is assigned a control point and central memory and is actively using a central processor. Then SCOPE executes the job as directed by the control card record of the job's input file.

JOB INPUT QUEUE

As each job is read into the computer by JANUS, the tape loader 1LT, or INTERCOM, it is placed into the job input file and a FNT entry for the job is created in the job input queue.

JANUS, 1LT and INTERCOM call the PP overlay 2TJ to translate the JOB control card. 2TJ scans the card, checks the parameters for validity, computes a priority value for the job, determines the job class, and passes all this information back to the PP program which called it.

The PP program 1IB scans the job input queue, looking for jobs eligible for execution scheduling. To be eligible, all resources requested on the job card, must be available, except for central memory. A job will remain in the input queue if it is dependent on a job which has not run, it is waiting for operator tape staging, or the job is not to be brought to a control point, or the amount of ECS it requested is not available.

TAPE JOB SCHEDULING

All incoming jobs requiring a specific number of 7 and/or 9-track tapes (as defined by MT or NT parameters on the job card) are entered in the pre-input queue, a subset of the input queue. The operator communicates with the pre-input queue through DSD type-ins and the P display. Each time the operator requests a P display, the jobs displayed are the highest priority tape jobs in the queue. A job requiring tapes is not placed in the normal job input queue until the operator releases it with a type-in. Once released, the job will be considered for assignment to a control point and execution; it no longer appears in the prescheduling display. Tape drive scheduling is described in Section 4 under non-allocatable devices.

LOADING JOBS FROM TAPE

The PP routine 1LT is used to load jobs onto disk via magnetic tape. 1LT is called to a control point by DSD when the operator initiates a keyboard entry in the format:

n.LOAD,g,p.

n is the number of the control point to be used.

g is an optional parameter specifying the number of the file group to be loaded.

p is an optional parameter specifying the number of the job in file g to be loaded.

Both g and p are one or two octal digits.

A standard circular buffer technique is used to process input files. The first card of the first record in each file is presumed to be a JOB card, and it is processed accordingly. This assumption is made for each job on the tape. CIO and its overlays are called to read the tape into memory and write memory to disk.

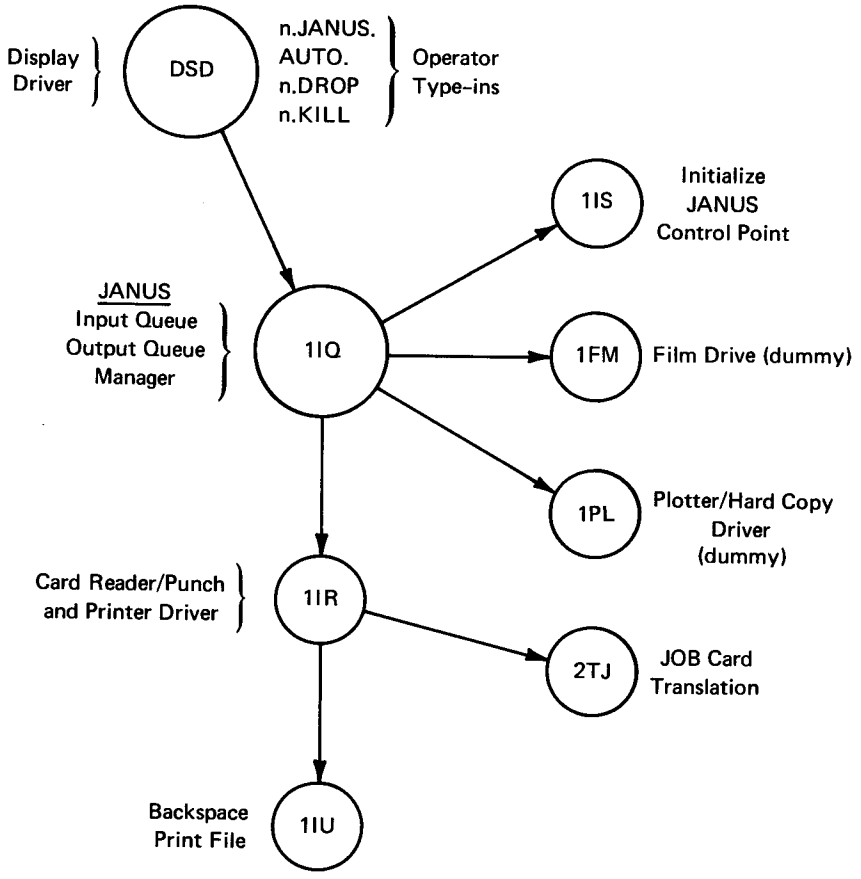
1LT is released when all jobs from a tape have been processed. A double end-of-file on tape signals the end of a group of jobs; when it is encountered or when an error condition is detected at the control point, control is transferred to the PP idle loop.

The number of control cards which may appear in a given job is not limited. If the user fails to place an end-of-record (7/8/9) card behind the control card record of a job, the omission will not be detected by either 1LT, JANUS or INTERCOM until the job runs.

JANUS

The unit record I/O package - JANUS - consists of the following PP programs:

1IQ	Input/output queue manager; sets up a control point to read jobs from card readers and to print or punch job output files; handles DSD type-ins and displays
1IS	Called by 1IQ to load a group of overlays into the field length for use by 1IR
1IR	Card reader, card punch, and printer driver
1IU	Called by 1IR to backspace print files when necessary
1PL	Dummy output file driver for plotter
1FM	Dummy output file driver for film/hard copy devices



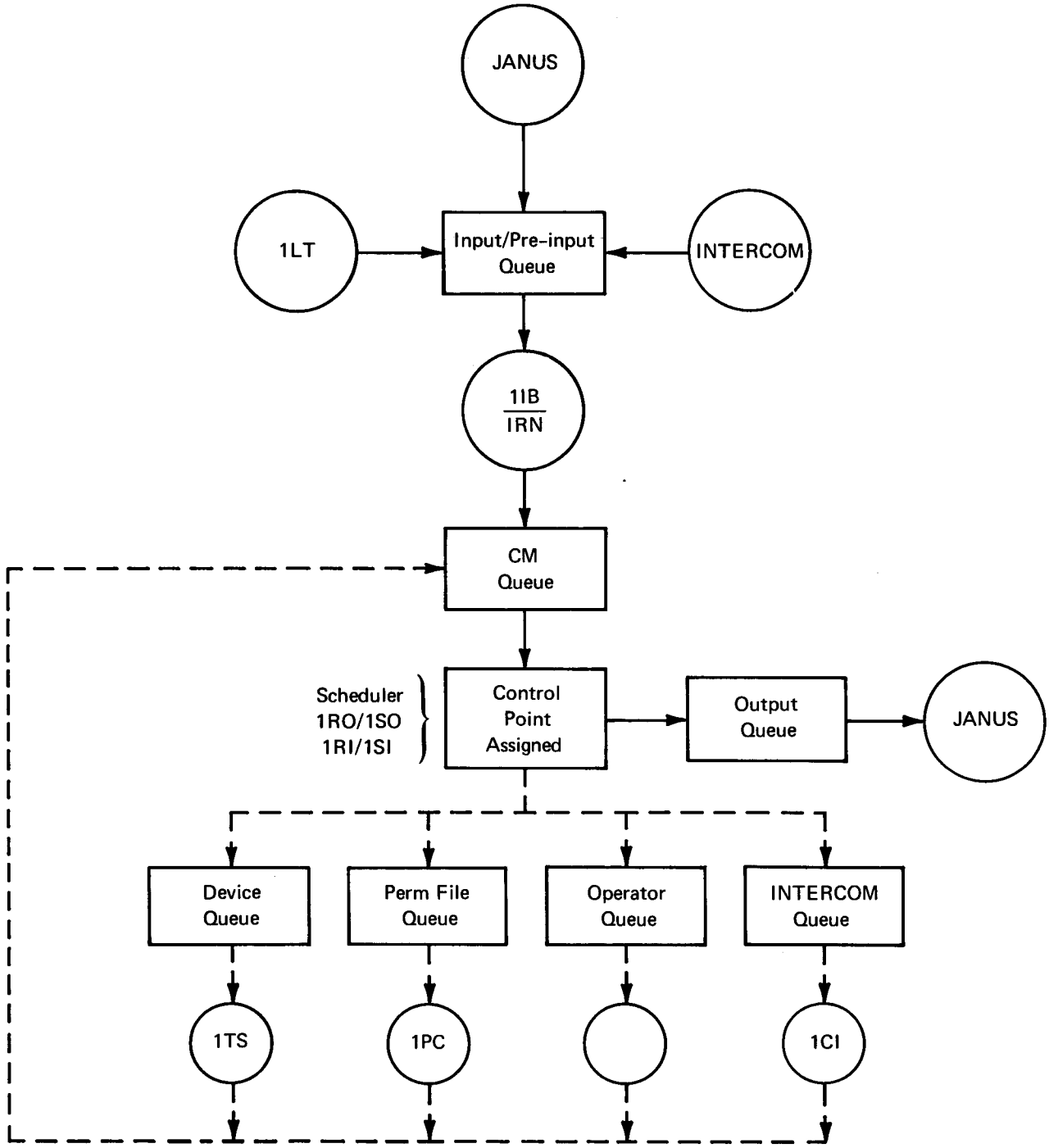
JANUS Interfaces

INTEGRATED SCHEDULER

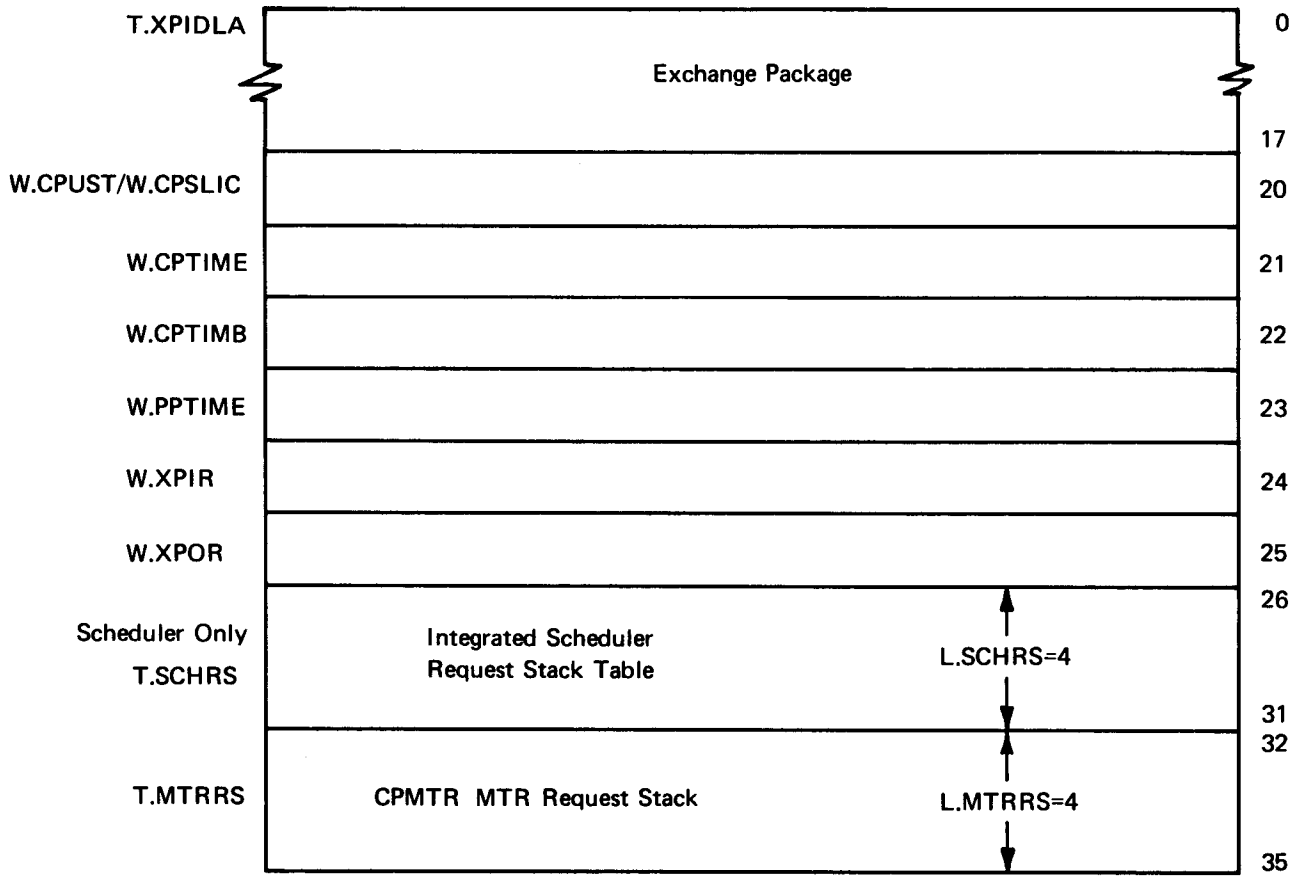
The job scheduling scheme implemented in SCOPE 3.4 is made up of the following:

allocator 1RA is replaced by the following:

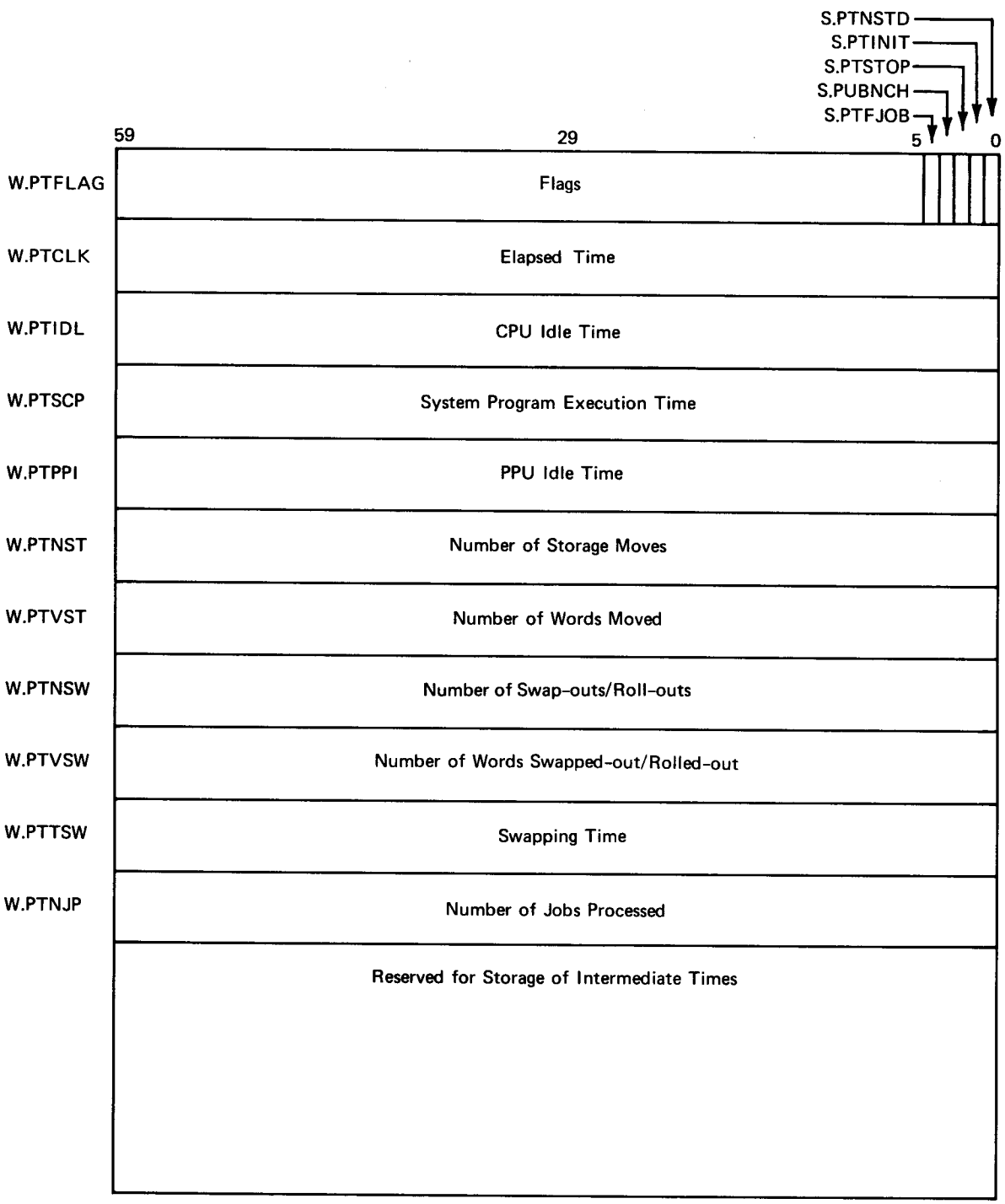
`Scheduler (MMGR)`	CPU program operating at control point zero in 2-to 3-millisecond cyclical intervals. it allocates control points and central memory to jobs and determines which jobs should be swapped/rolled in and/or out.
`1IB`	Brings jobs from input queue to CM queue; builds JDT entries for jobs.
`1RN`	PP program operating at control point zero at about one-second intervals. One of its functions is to age jobs in the input and output queues.
`Swappers`	PP programs called to swap or roll jobs in (1SI) or out (1SO) and to initiate new INTERCOM jobs.
`1DM`	PP program which processes job entries in the device queue.
`1PF`	PP program which processes job entries in the permanent file queue.
`1CI`	PP program which processes job entries in the INTERCOM queue.


JANUS
1LT
Input/Pre-input Queue
INTERCOM
1IB / IRN
CM Queue
Scheduler 1RO/1SO 1RI/1SI
Control Point Assigned
Output Queue
JANUS
Device Queue
Perm File Queue
Operator Queue
INTERCOM Queue
1TS
1PC
1CI

SCHEDULER REQUEST STACK IN SYSTEM EXCHANGE PACKAGE AREA



SCHEDULER PERFORMANCE TABLE (IP.SPT OPTION SET)



S.PTNSTD — Manual/benchmark option selected
S.PTINIT — Initialization of TSPT
S.PTSTOP — Stop data gathering
S.PUBNCH — Benchmark option selected
S.PTFJOB — First Job execution started

JOB SCHEDULING

In SCOPE 3.4, user jobs in execution can be either swapped or rolled in or out of central memory by the integrated scheduler or by the operator. This feature of the job scheduling scheme allows dynamic allocation of control points and central memory. If the scheduler needs a control point or central memory, it need not wait for a job to terminate. Such a scheme permits more efficient servicing of a mixed group of jobs, such as INTERCOM, remote batch, local batch and time-critical. System jobs such as JANUS and LOADER can not be swapped or rolled.

ROLLING

A job will be rolled out instead of swapped out under the following conditions:

Job has direct-access ECS assigned

Job has non-allocatable equipment assigned

Job is in device queue

Job is in operator action queue

When a job is rolled out, its field length is written to an allocatable file and all of its memory, except RA+0 through RA+77, is released. The control point remains assigned to the job and its JDT entry is placed in the appropriate queue. When a job is rolled in, it begins active use of its control point.

SWAPPING

A job may be swapped out under the following conditions:

Job is waiting for CM queue

Job is in permanent file queue

Job is waiting for INTERCOM terminal I/O (job is in INTERCOM queue)

When a job is swapped out, all information concerning the job is written to an allocatable file. This information includes the job field length, control point area, dayfile buffer, dayfile pseudo FET, FNT entries assigned to the control point, and entries in the PP event stack and delay stack. Both the control point and central memory are released and made available for reassignment. The JDT is placed in the appropriate queue; it contains a pointer to the swap file containing the job. The job, when swapped in, may be assigned a different control point and a different area in central memory.

The scheduler can be initiated by a PP program using a M.SCH or M.ICE monitor request. The scheduler is called in the following cases:

1. When a PP program has an entry to be added to the central memory queue, the scheduler will determine if the job has a priority high enough to cause any programs to be swapped out. If so, the scheduler initiates the swap-out and assigns the job to a control point. Otherwise, the scheduler adds the entry to the central memory queue.

2. When the field length of a job executing at a control point is reduced, the scheduler is called to search the central memory queue for a new job to be scheduled.

3. The scheduler is called periodically to determine if any job assigned to a control point has completed its quantum time period for CP and/or PP and if any jobs in the central memory queue have been aged enough to cause a swap-out. A job will be swapped out during its quantum time period only if a job with a very high class priority entered the central memory queue.

JOB CONTROL AREA

Jobs entering the system for execution are placed into one of five classes:

1. Batch jobs using no non-allocatable resources
2. Batch jobs using one or more non-allocatable resources
3. INTERCOM (interactive) jobs
4. Multi-user jobs
5. Express jobs (for which operator has entered DROP, KILL or RERUN)

An entry for each job class appears in a CMR table called the job control area. Each entry contains parameters affecting the scheduling of jobs in that class. These parameters include:

Minimum class priority

Maximum class priority

Aging rate (time) for incrementing job priority

Quantum priority value

Quantum time length value

Pointer to first JDT in job class scheduling chain

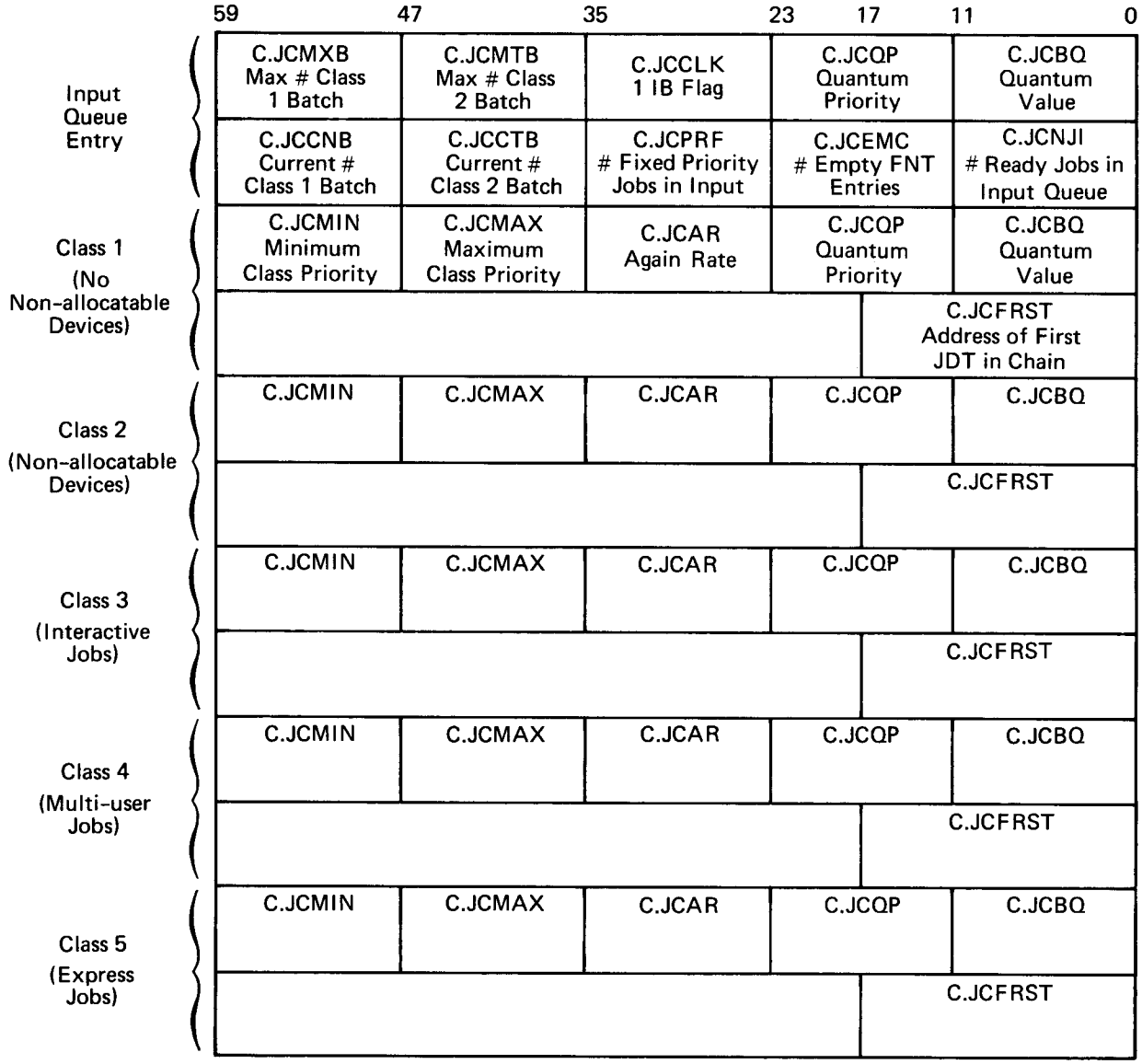
When a job is swapped out, it is given a queue priority value equal to the minimum queue priority for its class. This value is incremented with time at the aging rate for the class. When the incremented priority value reaches that of the maximum queue priority for the class, aging stops.

When a job is swapped in, it is given a queue priority value equal to the quantum priority for the class. When the quantum time length for the class has elapsed, the job priority is set to the minimum priority for the class.

A system operator can control job swapping overhead by adjusting the quantum time lengths for the different job classes in the job control area. Special DSD commands and displays allow the operator to adjust the job control area parameters.

The pointer to the job control area is T.SCHJCA which is in byte C.JCA of word P.SCH of the CMR pointer area. The following is the format of the job control area.

JOB CONTROL AREA



JOB DESCRIPTOR TABLE

Each job to be scheduled for execution, has a job descriptor table (JDT) entry containing information pertinent to its scheduling. If a job is not active at a control point, its JDT must be in one of the scheduler queues. Each job JDT in the CM queue is linked into one of the five job class scheduling chains originating in the job control area. A linking pointer to the next JDT in the chain appears in the first word of the JDT.

Entries in one of the job class scheduling chains of the CM queue are linked in order of their entry into the chain. Each entry contains the time when it was added to the chain. The current priority of an entry is computed by multiplying the length of time the job has been in queue by the aging rate for its class and adding the minimum class priority. Job JDT entries in the permanent file queue are also linked; JDT entries in other queues are not linked.

The job status code in the JDT is of prime importance. When a job is assigned to a control point, its JDT ordinal is in byte 4 of word W.CPJNAM in the control point area. If a job is not assigned to a control point, its status code in word W.JDMGR of the JDT indicates it to be in one of the following job scheduling queues:

Central Memory Queue

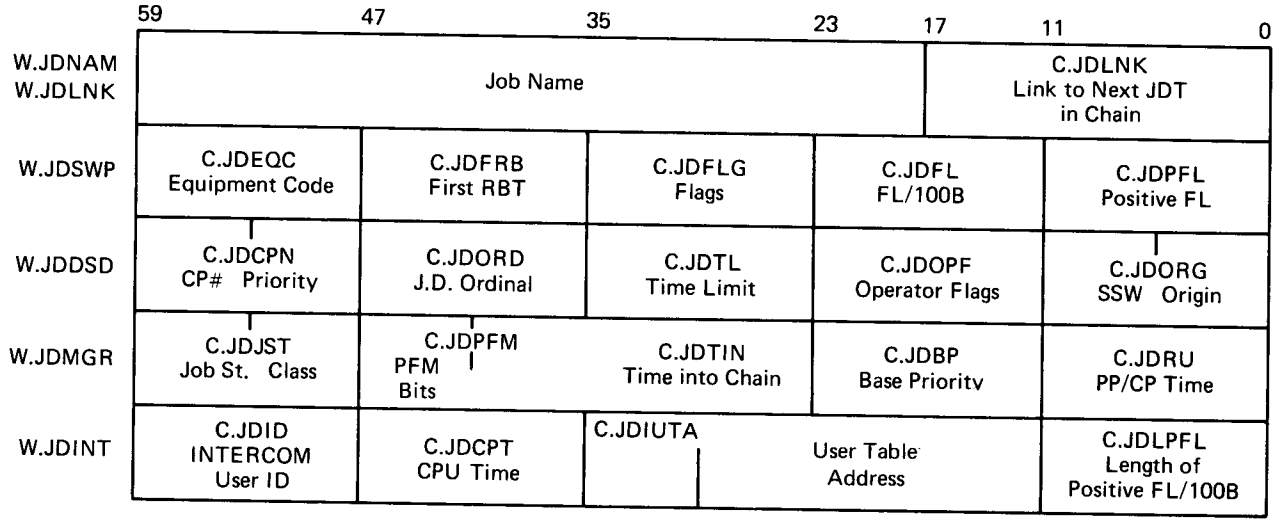
Device Queue

Permanent File Queue

Operator Action Queue

INTERCOM Queue

Format of a JDT entry:



JDT FIELD DESCRIPTORS

W.JDLNK

Bits 00-17	Contain address of next job descriptor in the chain. (C.JCFRST in JCA points to first entry in chain; a zero field denotes last entry in chain)
Bit 17-59	Job name as found in job input FNT

W.JDSWP

C.JDEQC, C.JDFRB	Contain the swap file address (F.JDSWI in word W.JDMGR is set)

C.JDFLG

Bit 11	S.JDBCB	Set if recovery took place
Bit 10	S.JDNRR	Set if job cannot be rerun
Bit 09	S.JDLGI	Set if job is a LOGIN command
Bit 08	S.JDLGO	Set if no swap file is to be generated for this INTERCOM job (LOGOUT command)
Bit 07	S.JDNJ	Set to indicate control cards must be read from INTERCOM area
Bit 06	S.JDECS	Set if swap file is on ECS
Bit 05	S.JDSKFL	Set if FL is to be skipped on swap file
Bit 04	S.JDROLL	Set if job cannot be swapped out
Bit 03	S.JDNFNT	Set if 1AJ should not search FNT table
Bit 02		unused
Bit 01		unused
Bit 00		unused

C.JDFL — Contains the job FL/100, including the job control block which is created when job is swapped out

C.JDPFL — Contains the relative starting address/100 of the job control block when present for job swapping; otherwise, contains zero.

W.JDDSD

C.JDCPN — Contains control point number in upper 6 bits. (Set when job is in execution or rolled out.) In lower 6 bits, job urgency or rerun priority.

C.JDTL — Contains job time limit as typed in by operator (truncated to 12 bits)

C.JDORD — Contains job descriptor (JDT) ordinal

C.JDOPF — Contains flags set by operator

Bits 09-11	0	
Bit 08-07		Currently not used
Bit 06	S.JDEXP	If job is to be placed in express queue
Bit 05	S.JDGO	If operator typed GO
Bit 04	S.JDNS	If job must not be swapped out or rolled out when at a control point
Bit 03	S.JDLOK	If job must not be brought to control point
Bits 00-02	Error codes: F.JDKILL, F.JDDROP, F.JDRRUN, F.JDRRNP	

C.JDORG

Bits 06-11: Sense switches

Bit 11	S.JDINT	Set for a standard INTERCOM job
Bit 10	S.JDMUJ	Set for a multi-user job
Bit 09	S.JDGR	Set for a graphics job
Bit 08	S.JDRT	Set for a real time job

Bits 00-05 — Currently not used

W.JDMGR

C.JDJST

Bits 48-53 Contain the job class values

05	F.JDEXP	If express handling was requested for this job
04	F.JDMUJ	For multi-user job
03	F.JDINT	For standard INTERCOM job
02	F.JDBNA	For batch job with non-allocatable device requirements
01	F.JDBAT	For batch job with no non-allocatable device requirements

Bits 54-59 Values describe the job status.

0x	F.JDLMB	If job is waiting for entry in a scheduling structure (in limbo)
1x	F.JDWCM	If job is waiting for CM (in CM queue)
2x	F.JDWPF	If job is waiting for a permanent file availablility (in permanent file queue)
3x	F.JDWDA	If job is waiting for device assignment (in device queue)
4x	F.JDWOA	If job is waiting for operator action (in operator action queue)
5x	F.JDWIA	If job is waiting for INTERCOM action (in INTERCOM queue)

x1	F.JDSWI	If job is waiting for swap in (is swapped out)
x2	F.JDACT	If job is currently executing at a control point (is active)
x3	F.JDWCC	If job is waiting for scheduler action (queue action completed)
x6		If job is swapped/rolled out

C.JDPFM

Bits 42-47 Contain information used by permanent file manager

C.JDTIN

Bits 24-41 Contain the time at which job descriptor was attached to job class chain

C.JDBP Contains job base priority

C.JDRU Contains ratio of PPU time and CPU time used by job during its last execution

W.JDINT

(assembled if INTERCOM is installed as part of the SCOPE 3.4 system)

JOB SCHEDULING QUEUES

CENTRAL MEMORY QUEUE

The CM queue is made up of jobs waiting for central memory and control point assignment. They are rolled/swapped jobs waiting to be reassigned a control point and/or central memory.

The PP routine 1IB scans the job input queue for jobs to be assigned to a control point. When such a job is found, 1IB creates the JDT entry and requests scheduler to set the entry in the CM queue. Jobs in other scheduling queues, such as the permanent file queue, for which requested action has been completed also are eligible to re-enter the CM queue.

DEVICE QUEUE

If a job executing at a control point requests a tape that must be mounted, the REQ routine will put the JDT into the device queue and call a swapper to roll out the job. The PP program 1TS is responsible for detecting when the requested device is ready and for calling the PP program IDM to put the JDT entry back into the CM queue. When the job is rolled back in, REQ will be recalled to create a FNT entry for the file.

Pseudo channel CH.SCH is used as an interlock in the device queue. Jobs in the queue are rolled out rather than swapped out so that the message to the operator will appear on the B display.

PERMANENT FILE QUEUE

When a job executing at a control point makes an ATTACH request for a permanent file, the PP routine PFA is called. If PFA determines that the file cannot be attached and that the job will have to wait, it places the JDT entry into the permanent file queue and calls a swapper to swap out the job.

Whenever an attached permanent file is detached, 1PF is called. It checks the permanent file queue for jobs waiting for the detached file. If any exist, 1PF will select the highest priority waiting for the file and call the scheduler to put its JDT back into the CM queue. When the job is swapped back in, PFA will be called to attach the permanent file to the job.

If a permanent file is purged, the PURGE routine will search the permanent file queue and call the scheduler to put all jobs waiting for that file back into the CM queue. As each job is swapped in, PFA will be called. It will detect that the file has been purged and respond accordingly.

All entries in the permanent file queue waiting for a given permanent file will be linked together in order of priority. The APF table entry for the file will contain a pointer to the first JDT entry in the chain. When PFA adds an entry to a chain in the queue, it will age all entries in the chain and insert the new entry. Pseudo channel CH.PFM will be used to interlock the queue.

If a job requests a permanent file to be attached with read-only permission and the requested file is already attached to a job with multi-read access, the requesting job can attach the file only if it has a higher priority than any job in the permanent file queue waiting to attach the file.

OPERATOR ACTION QUEUE

When a job makes an operator action request, its pause bit is set in the control point area, its JDT entry is placed into the operator action queue, and then it is rolled out. Such jobs are rolled out rather than swapped out, so that the message to the operator will appear on the B display. Whenever the scheduler finds a job pause bit set, it calls a swapper to put the job in the operator action queue, unless the PP routine processing the request indicates the job is to be put into some other queue. For example, REQ will set the pause bit and then request the job to be put into the device queue rather than the operator action queue.

When the operator takes appropriate action, the job is again eligible for the CM queue; it will be rolled in and initiated at its control point.

INTERCOM QUEUE

Jobs waiting for input from an interactive INTERCOM terminal or waiting until output can be sent to a terminal are swapped out and put into the INTERCOM queue.

JOB ADVANCING

When the scheduler has set up a job at a control point, the job advancing routine 1AJ is called into a PP. 1AJ reads one PRU of data from the control card record of the job's INPUT file into the control card buffer in the control point area, starting at location W.CPAF. Since the file INPUT is on an allocatable device, one PRU will be 100 CM words long and will extend to location W.CPCAL in the control point area. If the control card record has more than one PRU, 1AJ stores the INPUT file FST entry for the next PRU of the control card record in word W.CPFST in the control point area.

The job card has been processed by the routine 2TJ; 1AJ scans the control card buffer for the card following the JOB card. When it is found, 1AJ sets a pointer in word W.CPCC of the control point area to the next control card and then processes the first card found. The processing sequence for control cards is illustrated in figure 7-1 .

When the first control card has been processed, 1AJ is loaded again. Using the next-control-card pointer, it will read that card from the buffer, delineate the next card, set the pointer to the next card, and then process the control card just read. When all cards in the buffer are processed, 1AJ reads another PRU from the INPUT file control card record into the buffer and resets the next-control-card pointer to the beginning of the buffer.

1AJ continues to process control cards until all have been processed, the job is terminated because of error, or an EXIT card is encountered.

CONTROL CARD PROCESSING

When 1AJ begins to process a control card in the control point buffer, it copies the card to the job field length, at locations RA+70 through RA+77. Parameters on the control card are stored one in each word, starting at RA+2. 1AJ places the control card verb in PP direct cells ELM through ELM+4.

1AJ searches a table of control card names to match the verb. If the name is found, 1AJ jumps to a routine to process the card or calls a PP program to complete the processing. If the verb is not found in the table, 1AJ jumps to a separate routine to search the file name table for a matching file name then to search the program name table for matching absolute (NUCLEUS) library program name. If the verb is the name of an attached local file in the job FNT, 1AJ calls the loader to load the file. If the verb is the name of a system program, such as COMPASS, the program is loaded by 1AJ and the CPU requested.

Control cards with verbs in the list of control cards in 1AJ are processed as follows:

COMMENT	The comment card, less the keyword COMMENT, is issued as a dayfile message. If the message is longer than 40 characters, it is issued in two parts.
MAP	The map option bits in the loader flag byte C.CPLM in word W.CPLDR1 in the control point area are set.
MODE	The exit mode is set as specified on the MODE card. If the mode value is greater than 7, a control card error message is issued.
LIMIT	This card sets a limit on the amount of mass storage that can be allocated to the job. The value is given as a decimal number of 4096 CM-word blocks and is stored in the control point area, word W.CPMSLM, as the mass storage limit in number of PRUs. If the job does not contain this card, an installation defined value is used. A running PRU count is kept in the same control point word; if the limit is exceeded, the job is terminated.
SWITCH	The parameter value causes a corresponding bit to be set in the C.CPSSW byte of word W.SSW in the control point area.
REDUCE	The reduce flag is set in the loader flag byte C.CPLR of the control point area word W.CPLDR1.
RFL	The new field length value is set in byte C.CPNFL of word W.CPCC. The new field length must not exceed the field length given on the job card.

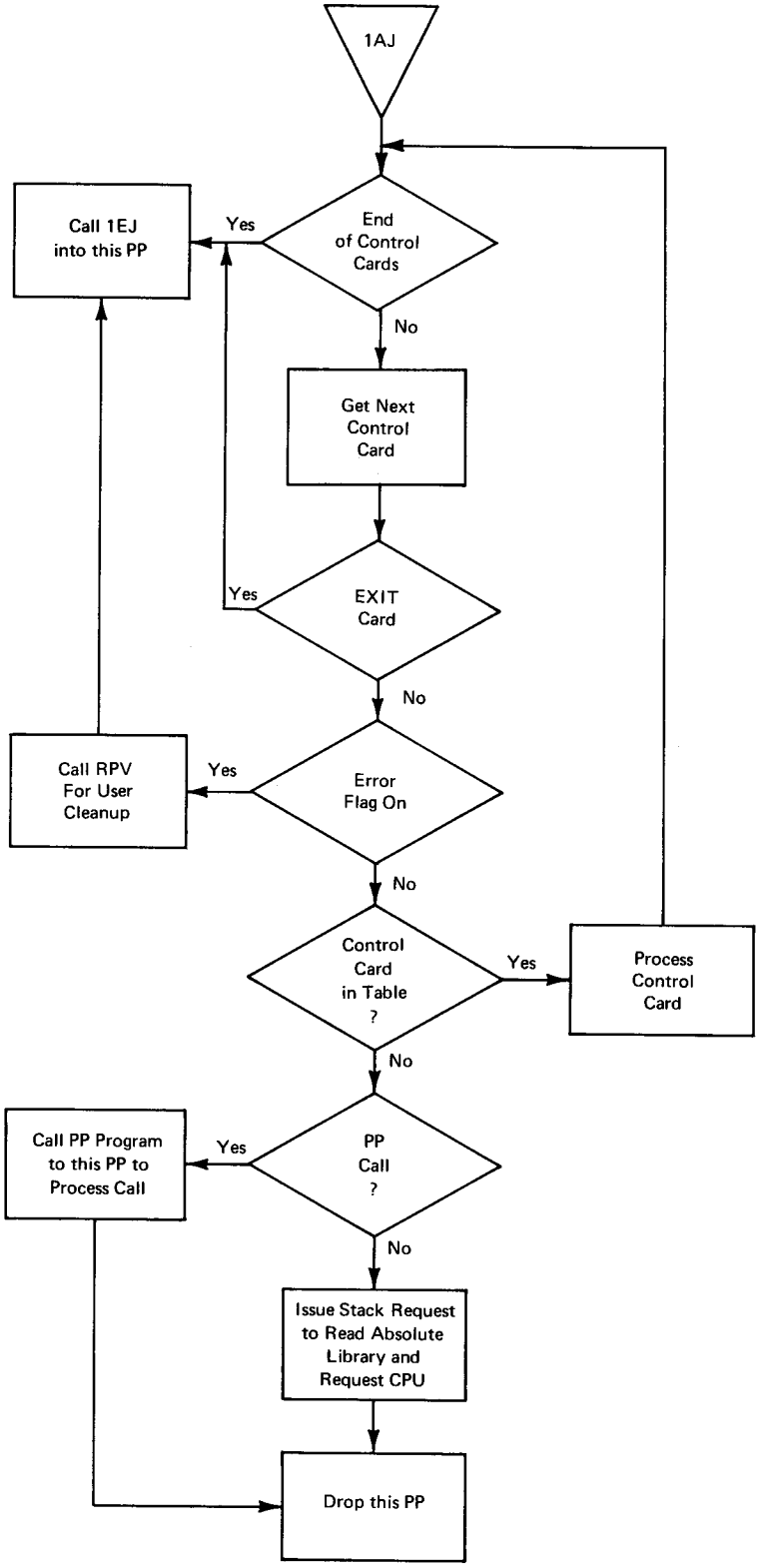


Figure 7-1. Control Card Processing Flowchart

CATALOG ATTACH EXTEND RENAME PURGE, etc.	Each permanent file control card is copied into the job field length, starting at RA+70. No dayfile message for the cards is issued. A CM program PFCCP is loaded to process the card.
RPACK	The family pack name is placed in the PP direct cell D.BA, the visual identifier and label type parameters are placed into D.FIRST and D.FNT. PP routine 1DA is called to process the parameters.
REMOVE	The local file name is placed in the PP direct cell D.BA; if present, the family pack name is left in D.EST. The PP routine 5DA is called to process the request.
EXIT	The exit flag bit is in byte C.CPFP of word W.CPFP in the control point area. If a control card error occurs or if a binary deck containing fatal assembly or compilation errors is loaded, the abort flag bit in the same byte is set. If the EXIT card contains a parameter, the exit flag bit is set when the error flag in byte C.CPEF of word W.CPEF of the control point area is set. If the EXIT card does not contain a parameter, the exit flag bit is set only if the error flag is set and the abort bit is not set. The PP routine 1EJ is called to do end-of-job processing.
VSN	The ordered sets of parameter are placed into RA+2 throught RA+n-1. The PP overlay VSN is called to process the parameters from the VSN card.
DMP	The arguments are converted to binary and the PP routine DMP is called to process the request.

JOB TERMINATION

NORMAL TERMINATION

Normal job termination occurs when the error flag value is zero, and all control cards for the job have been processed (an end-of-record mark for the control card record on the job INPUT file has been read) or when an EXIT or EXIT(S) card has been encountered. The end-of-job processor, 1EJ, is called by 1AJ to terminate the job as follows:

1. Dispose of all files associated with the job as follows:

   Drop local files and all equipment or storage associated with them.

   Attach non-zero disposition files to control point zero for further processing of output files.

   Call overlay 1PC for disposal of permanent files.

   Update the TAPES table; rewind and unload tapes as appropriate.

   When tape scheduling is used, return reserved tapes to the system pool.

2. Transfer the job dayfile to the OUTPUT file associated with the job.

3. Release storage for the job, set the job name to NEXT (or to zero if the clear bit is set), and zero appropriate words in the control point area.

Files are disposed of as follows:

FAMILY DISK PACK FILES

This type of private file is disposed of before any others. Information, such as file name and the RBT chain for the file is written to the disk pack. The file, itself, remains on the pack, but the FNT entries for both file and pack are deleted. RBT chains for disk pack files are released. 1EJ also calls 1DA to the control point while it flashes a message to the operator on the B display indicating the released packs. 1DA keeps recalling itself until the operator acknowledges the message by typing n.DROP. 1EJ then is called back to change the job name at the control point to NEXT if the clear flag is not set.

PERMANENT FILES

Each time a permanent file is to be disposed of, an entry is made in a list to be processed by 1PC. 1PC will delete the FNT/FST entry for the file; normally, the file will still exist on a mass storage device, unless purged by the user/owner. The APF entry for the file will be updated or deleted, as appropriate.

INPUT FILE

The job input file is not dropped until after the dayfile has been transferred to the job OUTPUT file and the OUTPUT file has been released to JANUS. Then the disk space for the job input file is evicted and its FNT/FST is cleared.

OUTPUT FILE

At the start of 1EJ operation, the buffer of all output files are flushed. After the dayfile has been copied to the job OUTPUT file, the OUTPUT file FNT is modified so that the file name is replaced by the job name, and the file is assigned to control point zero and rewound. The file is then released for processing by JANUS.

If the output file is on a non-allocatable device or pack, the equipment is dropped and the FNT is zeroed because JANUS cannot process files on non-allocatable devices.

If the file is on an allocatable device, the output buffers are flushed, the FNT is rewritten so that the file name is replaced by the job name, and the file is assigned to control point zero and rewound; its disposition code is retained. For INTERCOM batch files, the user ID is picked up from the control point area and stored in the FNT.

OTHER LOCAL FILES

If the file is on a non-allocatable device, the equipment is dropped. If the file is on an allocatable device, the file space is evicted. In either case, the FNT/FST entry is cleared.

DAYFILE

After the dayfile has been copied to the job OUTPUT file, its FET is reset. If part of the dayfile is on disk, the disk space is released. The FNT is rewritten to make the file local to control point zero, and the FST is cleared.

ABNORMAL TERMINATION

Abnormal termination of a job occurs when error flag value is non-zero. If the error flag value is other than KILL (value 7) or RERUN (value 10), 1EJ will dump the contents of the exchange package and the contents of memory. The dump includes location 100 before the error stop location through 100 words after the location (P-100 to P+100) to the OUTPUT file. A dayfile message describes the error. If an error flag value other than KILL or RERUN occurs and an EXIT or EXIT(S) card is contained in the job, the preceding will occur in addition to the flushing of output buffers for files with non-zero disposition codes. 1EJ then calls 1AJ to process the control cards appearing after the EXIT card.

If the operator has killed the job, the error flag value is set to 7. Permanent files and family disk pack files are processed as for normal termination. The FNT is zeroed and disk space or equipment is released for all other files, including output and non-zero disposition files. JOB KILLED is issued to the system dayfile, the control point area is reset and the job name is cleared. No output is produced for killed jobs.

If an operator has typed a RERUN command, the error flag value is set to 10 and the job input file is returned to the job input queue. Permanent files and family disk pack files are processed as for normal termination; all other files are dropped. A pre-output file is created as a local file to control point zero; it is not rewound; it has a file name which is the same as the job name. The dayfile is copied to the pre-output file, which becomes the output file for the job when it is rerun. Before dropping out, 1EJ resets the control point area and changes the job name to NEXT. However, if the clear request flag in byte C.CPFLG of the control point area is set, the job name will zeroed.

JOB POST-PROCESSING UTILITIES

During execution of a user program, various errors, such as arithmetic mode error or exceeding time limit, may cause the program and the job to terminate abnormally. At this point, the operating system will place a message in the dayfile, dump the exchange package and central memory (100 words on each side of the error stop location) and either resume the job processing at an EXIT or EXIT(S) card or terminate the job.

At the request of the user, the operating system will return control to the job after normal termination or after an error condition is detected and before EXIT card processing is performed. At that time, the system will return control to a user supplied post-processor routine which will print out data in appropriate format, etc. The user's post-processor routine may specify procedures to be performed after normal or abnormal job termination.

REPRIEVE

The reprieve function RPV is a PP program called by the user and by 1AJ. It allows the user to receive control at a location inside the job field length after program termination, at which time a cleanup/recovery routine may be executed.

The user should call RECALL during the job initialization phase, at which time the RPV call, generated by RECALL as follows, is placed in RA+1:

```
VFD  18/3HRPV,6/20B,12/mask,24/fwa
```

`RPV`	Is the PP routine name
`20B`	Sets the CPU in auto-recall
`mask`	Is a bit list of error classes for recovery
`fwa`	Is the first word address of the recovery routine in the user job

Job terminations are divided into classes; each class is assigned an octal value which may be specified singly or in combination to form the mask generated as part of the call. The values are:

0001	Arithmetic mode error
0002	RA + 1 error (PP call error, auto-recall error)
0004	Time and/or storage limit exceeded
0010	Operator drop or rerun
0020	System abort (PP abort; ECS parity error)
0040	CP abort
0100	Normal termination

The first word address points to a 17-word (decimal) block which will receive the exchange package and RA + 1 contents. Control is returned to fwa + 21.

Upon a user call to RPV, if the value of the upper 30 bits of the word at fwa is non-zero, it is assumed to be the last word address of the recovery routine in the user's job. A checksum of the area fwa + 21 to lwa will be made and the sum stored in fwa + 1.

The call to RPV by 1AJ occurs when 1AJ is entered with both error flag and reprieve mask non-zero. 1AJ calls RPV with bit 41 of the input register set to one.

Upon exit from a user call to RPV, the mask and fwa fields of the call will be stored in the control point area, in word W.CPCC. The contents of the word at fwa will be cleared; and if a checksum was performed, fwa + 1 will contain the sum.

Upon normal exit from a call by 1AJ, the mask in the control point area will be cleared, the exchange package with the error flag set in register B0 will be in fwa to fwa + 17, and the contents of RA + 1 will be in the next word, fwa + 20. If the remaining central processing time is less than five seconds and no time extensions have been grated, the time limit will be extended an additional five seconds in order to perform the user's recovery routine.

RECOVR

The central memory program RECOVR is an interface between user program and operating system for users who wish to gain control at the end of a job step, regardless of the manner in which it terminated. RECOVR should be called during the initialization phase of the job, at which time the address of the post-processing routine is given. RECOVR formats the arguments in the call into a stack entry, checksums the user post-processor routine if requested, saves the checksum in the stack, and then calls the PP program RPV to inform the system that the job should be restarted in case an error occurs.

When the system detects an error condition, it will issue two messages to the dayfile: the first will report the error condition and the second will report that the job has been restarted only if flags set in RECOVR call match those that cause the abort. It will give control to RECOVR along with the contents of the exchange package and RA+1 and the code for the condition that caused the system to stop program execution. RECOVR will call the post-processor program in a last in-first out order of calls in the entry stack and pass to it the address of the array area in which the exchange package, RA+1 contents and system error code has been placed. Upon termination of the user's post-processing routine, RECOVR will make an abort request to the system so that EXIT card processing will follow, unless the post-processor has specified that the program be terminated normally.

The calling sequences for RECOVR:

FORTRAN: `CALL RECOVR(name,flags,checksum)`

COMPASS: `RECOVR name,flags,checksum`

`name` Name of the user routine to which control is returned after a system abort

`flags` Octal value specifying condition under which control is to be returned.

001	Arithmetic mode error
002	RA+1 error
004	Time/storage limit exceeded
010	Operator drop or rerun
020	System abort
040	CP abort
100	Normal termination

`checksum` Should be zero for no checksumming; otherwise, it should be the last word address+1 of the recovery program to be checksummed.

The user post-processing routine to which control is returned by RECOVR should provide for three arguments. (1) The address of a 17-word array to receive the contents of the exchange package upon abort in the first 16 words and the contents of RA+1 in the 17th word. (2) A flag to indicate normal (ENDRUN) termination of the job or abnormal (ABORT) termination after post-processing is completed. A non-zero value indicates normal termination. (3) An address of a word to contain the contents of RA+0. If the P register contains zero in the exchange package, the address of the program error stop will be returned in bits 30-47 of RA+0.

Under no circumstance should the post-processor routine store into or modify the contents of the first argument.

The system error code is returned in bits 0-17 of word 1 of the first argument. The octal codes are:

0	Normal termination
1	Time limit exceeded
2	Arithmetic mode error
3	PP abort (illegal parameter list passed to a PP)
4	CP abort (program posted an ABORT request)
5	PP call error (garbage in RA+1 or PP program called is not in library)
6	Operator drop
10	Job to be rerun
12	ECS parity error
15	Auto-recall error
16	Job hung in auto-recall

Octal codes 7,11,13 and 14 exist but control will not be returned to the user when they occur.

If the user program is executed in overlay mode, both the call to RECOVR and the post-processing routine should reside in the 0,0 level overlay.

When the system stops normal execution because the job time limit has been exceeded, the time limit will be incremented by a value set per installation option so that the post-processing routine can be executed. Post-processing of an error condition should not be confused with OWNCODE (user processing of parity errors); the two concepts are not related.

EDITLIB 8

INTRODUCTION

The purpose of SCOPE 3.4 EDITLIB is to:

Create deadstart tapes

Create and maintain system libraries

Manipulate the running system to effect temporary changes

Modify the running system or the deadstart tape by merging in permanent changes and creating a new deadstart tape.

Create and modify a user library consisting of group of central processor routines or overlays.

The above is consistent with the philosophy of SCOPE 3.4. Assuming a user is starting from scratch, he will obtain an unconfigured deadstart tape from Software Distribution. This tape will contain the binary decks of the entire SCOPE operating system, to be used to deadstart a brand new system. EDITLIB is then called upon, via selected control cards and directives, to modify the current system after the appropriate assemblies and compilations have been performed. Thus a new deadstart tape and/or running system may be created which reflects the system as it is physically configured. (See figure 8-1.)

A major difference with previous system is that SCOPE 3.4 no longer requires all system code to be located in one massive system library. Object code may now reside in either a system library or a user library. Thus EDITLIB may be used in two distinct forms: SYSTEM EDITLIB, which is discussed in this section; and USER EDITLIB, which is contained in the SCOPE 3.4 Reference Manual.

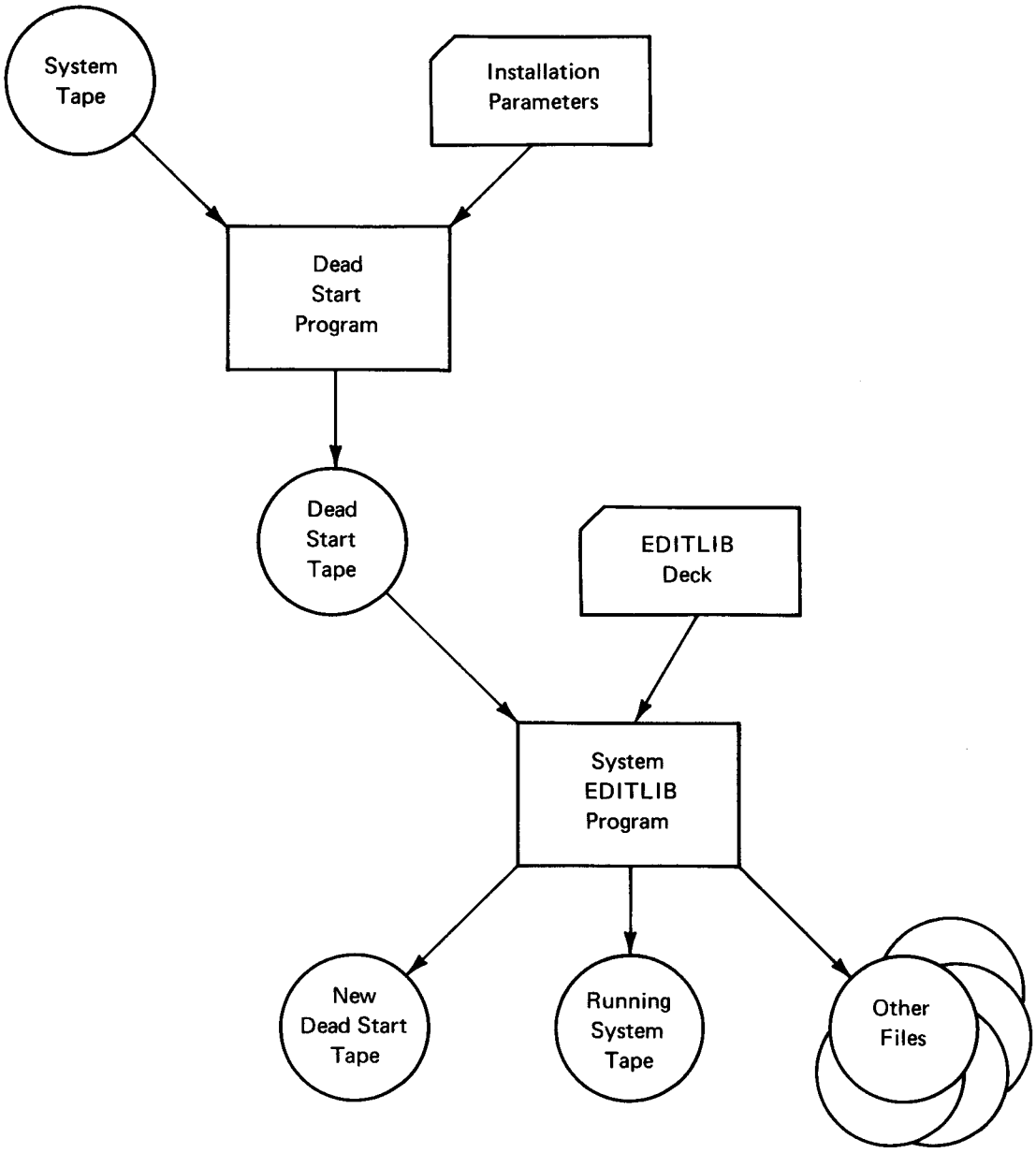


Figure 8-1. Basic Usage of System EDITLIB

CHARACTER SET

The EDITLIB character set is that which is supported by SCOPE 3.4. EDITLIB will interpret the $ ( ) = + - * / , and . as delimiters. Blanks are free characters and will be removed. From the character set, the user is able to create:

1. Symbol – a single character or a string of characters

2. Vocabulary Word – a symbol that has meaning as defined by the semantics.

3. Name – a symbol that is defined by syntax and semantics.

When any delimiter and/or a blank is used to form a symbol, the symbol must be enclosed between $ signs. (see Example 1). When a $ is to be used as a character within a symbol, an additional $ must be written immediately adjacent to each $ desired. The only character that will be removed is the first $ of each pair of $ encountered. (See example 2.)

Whenever a name could be interpreted as a number or a vocabulary word, the name should be enclosed between $ signs.

Examples:

$--DSD---$

Symbol is --DSD---

$D$$I$$$$$$--$

Symbol is D$I$$$--

$$$CIO$

Symbol is $CIO

The input to EDITLIB is the first 72 columns of an 80 or 90 column card image. A directive must be completely contained on one card.

SYNTAX AND SEMANTICS

Presented here is the format of the various directives and the meaning associated with the symbols. The syntax as illustrated in the examples must be followed.

Any directive having parameters must be terminated by a right parenthesis. If the directive has no parameters, it must be terminated by a period.

Whenever a violation of syntax is encountered, EDITLIB will set the abort flag. After EDITLIB has completed a given task, such as checking all directive formats, it will check the abort flag. When set, EDITLIB will abort. EDITLIB will continue to check a directive even if it has located an error in that directive. EDITLIB will proceed to the next directive only after it has completed analyzing the current directive or after it is unable to determine the syntax of the directive.

SYMBOLS USED IN SYNTAX

[ ]	The enclosed items are optional.	
⊥	or	
( )	The enclosed item is the actual symbol.	
p	name of a routine	
lfn	Logical file name	
n	positive or negative integer	
pn	library name	
r	residence:	
	CM	Central Memory
	DS	Disk
	ECS	Extended core storage; if no ECS available, then default to disk.
	EM	Extended Core storage; if no ECS available, then Central Memory.
	Default	Disk

Interval in a directive will be interpreted to mean

$P_1 + P_2$ or

P_1-P_2 or

$P_1/P_2/P_3$ or

P_1

where P_i are single program names. It is acceptable to replace either P_1 or P_2 with [*], but not both. EDITLIB will set the abort flag for the following conditions:

unable to locate P_1

unable to locate P_2

unable to locate P_1 when P_2 = [*]

unable to locate P_2 when P_1 = [*]

unable to locate P_1 when there is no interval.

* This may be used to replace p in the syntax. When * is encountered, EDITLIB will consider all remaining routines on the file proceeding from the current position. If the file is a library, the current position is the first entry in the PNT.

\+ This symbol is used to designate an inclusive interval.

Example: Given the ordered set of records A, B, C, D, E, F, and G on a file; then A + G represents all members of the file.

– This symbol is used to designate an exclusive interval.

Example: Given the ordered set of records A, B, C, D, E, F, G, H, I, and J on a file; then D-H represents all members of the file except D, E, F, G, and H. This leaves A, B, C, I, and J.

Notes

1. EDITLIB will assign a number to each directive it processes.

2. EDITLIB will list each input card it receives on the OUTPUT file. The directive as interpreted by EDITLIB will appear directly below the input card. The interpreted directive will be prefaced by the number assigned to this directive. All comments about the directive will follow the interpreted directive. Error messages will be generated by referencing the number that is assigned to each directive. Error messages will be issued whenever EDITLIB finds it cannot comply with the user's request. EDITLIB will also issue messages when it is unable to determine what the user is trying to accomplish.

3. The directives ADD, DELETE and REPLACE will produce a CONTENT directive on the routines involved.

4. Any interval specified in a directive will be interpreted to mean:

   In a reference to a file which is a library, the interval will be formed by the entries in the order they appear in the Program Name Table.

   In a reference to a file that is not a library, the interval will be formed by the order of the routines on the file.

   The interval begins when p_1 is encountered and ends at which ever occurs first: p_2 or EOF or the last entry in the PNT. The table or file is not handled circularly.

5. In a System EDITLIB, all directives that do not specifically name a particular library and are not contained between a LIBRARY and FINISH directive will refer to the PP library. This means that the PP library is the default library when no library is specified.

6. EDITLIB will not support TEXT and/or DATA input (7000 LIBEDT). If text/data input is found, it will be flagged and the job will be aborted.

7. Miscellaneous 7000 LIBEDT directives: TYPE, ERROR, PCOPY, PMOVE, MOVEC, COPYC, and DELETEC. 6000 EDITLIB will ignore these, print a warning message on each directive, and send one warning message to the dayfile, if present in run.

SYSTEM EDITLIB

EDITLIB FILES

EDITLIB will use the following lfn:

lfn	Cataloged	Use
ZZZZZ01	yes	RESET
ZZZZZ02	yes	RESTORE
ZZZZZ03	yes	SYSTEM EXTEND
ZZZZZ04	yes	SYSTEM
ZZZZZ05	no	Interpreted directives
ZZZZZ06	no	ECS resident routines library file
ZZZZZ07	no	Entry Point Name Table spill file
ZZZZZ08	no	Entry Point Name Table spill file
ZZZZZ10	no	Program Name Table spill file
ZZZZZ11	no	External Reference Table spill file
ZZZZZ12	no	External Reference Collection spill file
ZZZZZ13	no	Library or Deadstart program collection
ZZZZZ14	no	Scratch
ZZZZZ15	no	PP Program Name Table spill file
ZZZZZ16	no	Library Name Table spill file
ZZZZZ23	yes	Current Directory File

System EDITLIB will produce the following type of files:

File Use	lfn	pfn
RESET	ZZZZZ01	ZZZZZ01
RESTORE	ZZZZZ02	ZZZZZ02
SYSTEM EXTEND	ZZZZZ03	ZZZZZ03
SYSTEM	ZZZZZ04	ZZZZZ04
ECS LIBRARY	ZZZZZ06	
Current Directory	ZZZZZ23	ZZZZZ23

The files will be cataloged using the following keys:

File Use	ID	RP	TK	CN	MD	EX
RESET	EDITLIB	999	SSSSSSSSS	XNOX	XNOX	
RESTORE	EDITLIB	999	SSSSSSSST			
SYSTEM EXTEND	SYSTEM	999	SSSSSSSSU	XNOX	XNOX	XNOX
SYSTEM	SYSTEM	999	SSSSSSSSV	XNOX	XNOX	XNOX
Current Directory	SYSTEM	999	DIRECTORY	XNOX	XNOX	

In SCOPE 3.4, the allocation of ECS is controlled by Monitor. Since the allocation of ECS involves the creation of tables and pointers by Monitor, the Deadstart Loader will not duplicate this effort to set up ECS files. Due to this change, a new procedure will be required to place routines into ECS at initial deadstart time.

Another task that EDITLIB performs in conjunction with deadstart is that of recorery. Deadstart will call EDITLIB to accomplish a C type recovery. In this situation, no operator intervention is required. The Deadstart Loader will set bit S.SYSEDT in byte C.DSFLAG in word P.LIB. This bit when set informs MDI that it can go ahead and change the directory without issuing the GO/DROP message. When MDI has moved the directory from the CP to CMR, it will clear S.SYSEDT and set bit S.EDTRUN in the same byte. S.EDTRUN will be set by EDITLIB whenever the directory is changed.

EDITLIB will also maintain a permanent file (ZZZZZ23) for the Deadstart Loader. This file will contain the current system directory. This record represents the directory as it appears in CMR.

The files ZZZZZ01, ZZZZZ03, ZZZZZ04, ZZZZZ06, and ZZZZZ23 will be cataloged such that by attaching them, CN (Control) and MD (Modify) permission are not granted. That is, the above permission will be granted if the correct passwords are specified, but not by just specifying the ID (Identification) and TK (Turnkey). EDITLIB will request the CN or MD options on file ZZZZZ23. When EDITLIB modifies the running system, it will do so by adding the new routines to the end of the file called ZZZZZ03. By using this method, EDITLIB will not keep others from accessing the files.

CONTROL CARDS

PROGRAM CALL CARD

EDITLIB(SYSTEM,t_1,t_2, . . .t_n)

where t_i can be

RESET

RESTORE

I=lfn

L=lfn

LDIS

MSGL=n

SYSTEM — This parameter is a password. It is used by EDITLIB to determine what kind of EDITLIB run is to be performed. This key is a program parameter. The key can not exceed nine characters and will allow restricted access to the running system directory.

RESET — This parameter instructs EDITLIB to replace the current system directory with the directory as it appeared after initial deadstart. Whenever the RESET parameter is encountered, EDITLIB will use the directory from the RESET permanent file to replace the directory presently in CMR. It will then change this directory as prescribed by the directives in the input file. EDITLIB will then replace the current directory with the directory it just modified. A copy of the current directory will be saved on the RESTORE permanent file before the change takes place.

RESTORE — This parameter informs EDITLIB that a user wishes to replace the current system directory with the directory as it appeared before the last EDITLIB took place. Whenever the RESTORE parameter is encountered, EDITLIB will use the directory on the RESTORE permanent file to replace the directory presently in CMR. The reset of the procedure EDITLIB follows is the same as for an EDITLIB (SYSTEM, RESET) except that the current directory will not be saved on the RESTORE file.

I=lfn — This symbol is used to designate the file containing EDITLIB directives. If it is not present, the lfn that will be used is INPUT. If INPUT is used, any binary input to EDITLIB on this file must be in the order that it will be requested. This file will not be searched.

L=lfn — This symbol is used to designate the file of listable output from EDITLIB. If it is not present, the lfn that will be used is OUTPUT.

LDIS — This is a decommittal request to EDITLIB to issue a message to the dayfile to have the operator bring the L display to this control point. The message issued will be EDITLIB REQUEST L-DISPLAY. This parameter will allow the operator to enter EDITLIB directives from the display console. If this option is selected, directives from the console will be accepted before those in the input file. Each directive entered at the keyboard will be placed, in turn, in the user's field length and analyzed for syntax. The user may not enter directives until the word NEXT. is displayed. The word WAIT. will be displayed when EDITLIB is processing a type-in.

MSGL=n — where n = 0 or 1. The default value is 0. Under this condition all EDITLIB messages will be sent to the output file only. If n=1, EDITLIB will send its messages to the output file and to the system and user's dayfile. Thus all messages will appear on the system A display. If the value of MSGL is 0, then all error messages, including fatal error messages, will only appear in the output listing.

OPTIONAL PARAMETERS

Optional parameters are not positional.

AL=l — where l = 0 to 7777B. Default = 0. AL stands for access level and is a parameter of the ADD and REPLACE directives. 11 of the 12 bits are meaningful to INTERCOM only. The right most bit of this field is used by the system to regulate access to program through control card requests. When this bit is set to 0, the routine or entry cannot be called from a control card. The setting of this 12-bit field is determined by the binary representation of the octal digits specified by AL. INTERCOM has divided the 12-bit field into 3 sections. The first section is one bit in size and is the right most bit of the field. The remaining 11 bits are divided into two sections according to the installation parameter, IP.ACES. IP.ACES = n defines the number of bits to be reserved for the access level. The remaining bits (11-n) will be used as permission bits. When a user requests the use of a command, his access level is checked to determine whether it is greater than or equal to the access level of the command.

If it is, the permission bits of the user are compared against the permission bit of the command. If a match is found, the user is given access to the command. If either test fails, permission is not granted.

FL=k — where k = 0-377777. Default = 0. FL is a parameter of the ADD and REPLACE directives and stands for Field Length. It is meaningful for CM programs only. It is used by INTERCOM to determine the amount of CM needed to execute a program. This field length must taken into account the CM needed by all programs, overlays and segments that could be loaded by the specified program.

FLO=m — where m = 0 or 1. Default = 0. This parameter can only appear on an ADD or REPLACE directive. It is used to set the FL override bit. This bit is used by INTERCOM to determine whether the field length entered by the user at a terminal may override the field length specified in the PNT.

LIB=pn — where pn is the name of a library. This is a parameter which can only appear on an ADD or REPLACE directive. It is used when the lfn specified in the directive is SYSTEM (meaning the running system) or a deadstart file. This parameter specifies what library on the deadstart file will be used to satisfy the directive.

NEW — Mandatory parameter for the LIBRARY directive and optional parameter for the READY directive. When a deadstart file is being created, the lfn specified will contain a new library or directory.

OLD — Mandatory parameter for the LIBRARY directive and optional parameter for the READY directive. It is required in these directives when an existing library or an existing deadstart file is referenced by the lfn parameters.

r — Represents residency as follows:

CM	Central Memory
ECS	Extended Core Storage; if no ECS available, then default to disk.
DS	Disk (default)
EM	Extended Core Storage; if no ECS available, then Central Memory.

Examples:

```
EDITLIB(SYSTEM)

EDITLIB(SYSTEM.RESET)

EDITLIB(SYSTEM.LDIS.MSGL=1.RESTORE)

EDITLIB(SYSTEM.LDIS.I=INFILE.MSGL=0)

EDITLIB(SYSTEM.I=INFILE.L=OUTFILE)
```

ADD(Interval,lfn[,r][,(AL=)l][,(FL=)k][,(LIB=)pn][,(FLO=)m])

This directive adds the specified program(s) from the lfn to the library under construction or modification. If the lfn contains a library, the Program Name Table of the library will be searched to locate the specified program(s) to be added. If the file is not a library, the search for the specified program(s) will begin at the current record position on the file and proceed to EOF. If the program(s) are not found, the file will be rewound and the search will continue until all programs have been searched. If the program(s) are not found in the library or on the file, an error message will be issued. If the program(s) to be added already exist(s) in the library, an error message will be issued. It is not the function of an ADD directive to replace the current program with the new program. This function is accomplished by a REPLACE directive.

Example: ADD(HIGH,LOW,ECS,AL=70,FL=1000)

ADD(HIGH,LOW,FLO=1)

ADD(HIGH,SYSTEM,LIB=SCOPE,ECS,AL=123)

CHANGE(pn[,r])

This directive is used to change the residence of a library. If the library already has the specified residency or if the library is not part of the system, EDITLIB will issue an appropriate comment.

Example: CHANGE(FNT,CM)

COMPLETE.

This directive informs EDITLIB that the directive list for the directory whose name appeared in the READY directive has been exhausted. Simply, it means that there are no more modifications or additions to be performed on the directory. This directive must have been preceded by a READY directive.

CONTENT(Interval,lfn)

The CONTENT directive is used to obtain information about a program or series of programs on a specified file (lfn). The information is obtained from the programs expanded Prefix Table and the program itself. The CONTENT directive applies to all files other than a deadstart file or a LIBRARY file.

Example: CONTENT(ONE,FILE1)

DELETE(Interval)

This removes all entries from the library tables that refer to the specified programs.

Example: DELETE(FEAR)

DELETE(FEAR+HATE)

END.

This directive is a request for decommittal and designates that the last directive has been entered from the L-display. EDITLIB will now resume as if the LDIS option had not been requested.

ENDRUN.

This directive indicates that execution of directives should stop. All directives following ENDRUN will be checked, but not processed. This is an optional directive; if it is not found, one will be generated. If this directive is found and EDITLIB finds an error in a directive after the ENDRUN card, it will be flagged but will not cause an abort.

FINISH.

This directive designates the end of the list of directives that will affect the library defined by the last LIBRARY directive. Each LIBRARY directive must have a FINISH directive. It is permitted to have multiple (LIBRARY, FINISH) sequences within a single directive sequence.

INCLUDE(pn,lfn[,r])

This directive will add the library which is on the specified file to the directory. This allows a user to make his user library a part of the running system.

Example: INCLUDE(SMOG.AIR.ECS)

INCLUDE(SMOKE.LUNGS)

LIBRARY(pn,(OLD)⊥(NEW)(,r))

The LIBRARY directive during a system EDITLIB run performs two functions. When it is encountered outside of a READY and COMPLETE directive, the action taken is equivalent to a user's EDITLIB. When it is between a READY and COMPLETE directive, the action is also equivalent to that of a user's EDITLIB, but the LIBRARY will be found using the Library Name Table (OLD) or will be added to the Library Name Table (NEW).

Example: LIBRARY(FTN,OLD)

LIBRARY(RUN,NEW,ECS)

LIBRARY(COBOL,NEW)

LISTLIB(Interval,lfn,[pn])

The LISTLIB directive is used to list a library which can be a user's library, running system, new system and/or deadstart file.

Note: LISTLIB(COPY,SYSTEM,SCOPE)

When between READY and COMPLETE, the lfn that the list will be performed on is the running system and/or the new system under construction. When this directive appears outside of a READY and COMPLETE, only a user's library and a deadstart tape can be listed. pn is meaningful when listing the running system, new system or deadstart file. If not present, the PPLIB is assumed. LISTLIB cannot be present between a LIBRARY and a FINISH directive.

These directives will send information about the specified program(s) on the lfn or on the lfn and in the pn to the output file. The information will come from the object deck being processed.

Example: LISTLIB(COPY,FILE1,SCOPE)

LISTLIB(1AJ,SYSTEM)

LISTLNT.

The LISTLNT directive must appear between a READY and a COMPLETE directive. It will list the Library Name Table of the file referenced on the READY directive. There are no parameters.

MOVE(Interval,r)

This directive will change the residence of CP programs within a system library. Movement of CP programs depends on the residence of the system library of which the CP program is a part. If CM resident, movement is unrestricted; but for DS and ECS resident libraries, the new residence can be specified only as DS or ECS.

Example: MOVE(STOCKS,DS)

MOVE(BONDS+CASH,CM)

READY(lfn[,(OLD)⊥(NEW)])

This is the first directive of any directive list pertaining to the lfn specified. Only those directives between a READY and a COMPLETE will affect the lfn. The READY directive is used in conjunction with a system EDITLIB run. The READY directive specifies that the operation will be performed on a directory. The lfn is either the running system or the name of the file which will contain the new system. If the name SYSTEM appears as the lfn, it will mean the running system when the OLD parameter is specified or missing. Directives following the READY directive can be classified into four major functions: Those that modify the Library Name Table; those that modify the PP library; those that modify the system library (they must have LIBRARY and FINISH directives to delimit them); those that do not change any part of the system directory.

Examples: READY(FUTURE,NEW) READY(FUTURE)

READY(SYSTEM,OLD) READY(SYSTEM)

REMOVE(pn)

This directive removes the library from the running system. The library's entry in the Library Name Table is removed.

Example: REMOVE(FTN)

REPLACE(Interval,lfn[,r] [,(AL =)l] [,(FL =)k] [,(LIB =)pn] [,(LFO =)m])

This REPLACE directive in a system EDITLIB will retain the r value of the replaced routine if r was not specified. All other option parameter values will not be saved.

Example: REPLACE(IBM,CDC,AL = 70)

REPLACE(XDS-RCS,NCR,LIB = NUCLEUS)

REWIND(lfn)

The result of this directive will be the positioning of the specified file at the first record on the file. To accomplish this, EDITLIB will issue a CIO request to do a REWIND. In system mode, the file name SYSTEM refers to the running system.

SETAL(Interval,I)

where I = 0 to 7777 (octal). This directive permits the user to change the access level of a program already in the directory.

Example: SETAL(IBM + NCR,7770)

SETAL(CDC,1)

SETFL(Interval,k)

where k = 0 to 377777 (octal). This directive permits the user to change the field length required to run the specified program already in the directory.

Example: SETFL(BYTE,100000)

SETFL(BIT+WORD,100000)

SETFLO(Interval,k)

where k = 0 or 1. This directive permits the user to change the field length override bit of program already in the library. The default value is 0 (override is not permitted).

Example: SETFLO(BIT11,0)

SETFLO(BYTE-WORD,1)

SKIPF/SKIPB(N,lfn,[F]⊥[P])

This is the new skip-forward and skip-backward directive. The meaning of the parameters is as follows: If N is numeric, it represents a number of records to be skipped unless the third parameter is present; then the meaning is files. If N is alphanumeric, then it means that the lfn will be positioned before the record of that name (N). If N is not found, original position is maintained. Alphanumeric form is applicable to sequential files only. While in system mode, the file name SYSTEM means the running system.

TRANSFER(Interval⊥n,lfn)

This directive is used to create deadstart tapes. It will move records whose prefix table has been removed or is to be removed. A check will be made to determine if the records being moved have a prefix table. Because a program name can be all digits it is necessary to delimit such programs for this directive; e.g. program name is 026, the directive will be TRANSFER(026,SYSTEM).

Example: TRANSFER(*,SYSTEM)

TRANSFER(10,SYSTEM)

*/comment

This symbol is used to denote a comment field; whenever encountered in columns one and two, EDITLIB treats it as a comment and will not check the card.

Example: */THIS ADDS A PROGRAM THAT WILL DO A/B*C

LIBRARY DIRECTORY ACCESS

PP ROUTINES

When a PP routine wishes to access the library directory, it will first check bit 59 of P.LIB. When bit 59 is set to 1, EDITLIB is waiting to change or is changing the directory. Thus, the directory is unavailable. When bit 59 is set to 0, the directory is available.

CP ROUTINES

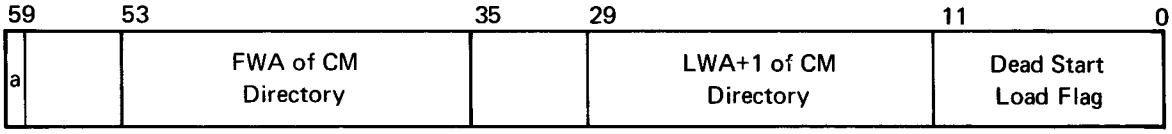
All CP routines accessing the directory should set bit S.CPLDAF in byte C.CPFLAG in word W.CPFLAG of their control point area. To access the directory the following procedure should be followed. First check bit 59 in P.LIB; if not set, then set bit S.CPLDAF. Then check bit 59 in P.LIB again. If it is now set, clear S.CPLDAF and wait until bit 59 in P.LIB is clear. After access is obtained and directory access is complete, clear bit S.CPLDAF.

EDITLIB

When EDITLIB wishes to access the directory, it will issue a monitor request to have the PP routine MDI assigned to its control point. By passing MDI the appropriate code, MDI will try to change the directory. The procedure is as follows:

MDI will set bit 59 in P.LIB to 1 to lock out all other routines from initiating a directory access. It then checks bit S.CPLDAF in byte C.CPFLAG in word W.CPFLAG in each control point area. If the bit is set, it means the job has not completed its last directory access, and MDI will go into the event stack until the bit is cleared. After MDI has completed its task, it will adjust the FWA and the LWA + 1 pointers in P.LIB and set bit 59 to 0.

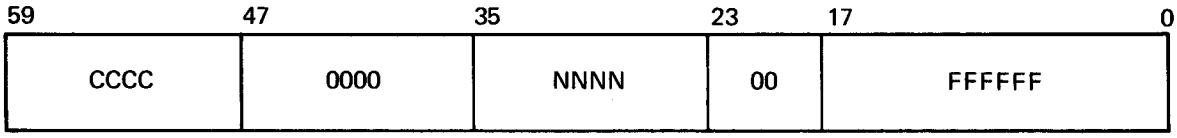
LIBRARY DIRECTORY POINTER FORMAT



a = Library Change Flag

P.LIB — The use of the first word has been expanded to hold a Library Change Flag (1 bit). This flag was added to increase system integrity and effiency. The Library Change Flag tells the PP and any CP program using the directory that an EDITLIB run wishes to change the directory or is changing the directory. Only when the flag in the control point areas is zero will EDITLIB proceed to change the directory.

PP LIBRARY POINTER FORMAT



P.PPLIB FFFFFF is the address of the first entry in the PP Program Name Table. NNNN is the number of entries in the PP Program Name Table. CCCC is the position of CIO in the PP Program Name Table. This value specifies the number of entries preceding CIO.

MDI will set this word to zero and wait one second before changing the directory.

FILES

SYSTEM FILES

A copy of all the routines referenced by the directory will reside on a permanent file called ZZZZZ04 after initial deadstart. If a program or library subsequently is to be included into the directory, it will be placed in the permanent file called ZZZZZ03. EDITLIB will never write, modify or extend the permanent file ZZZZZ04. The new version of replaced routines will reside on the file ZZZZZ03. Since EDITLIB will not alter any record currently on the file ZZZZZ04 or the file ZZZZZ03, routines may be read from either file while EDITLIB is adding new records to ZZZZZ03.

USER FILES

The user is responsible for the contents of his library. EDITLIB will not overwrite any record on the file. It will only add records at EOF/EOI. This means that the user will have unused records on his file when he deletes are replaces programs on his library. It is up to the user to remove this dead space by building a new library using the old one as a source. EDITLIB will issue no permanent file request. The user must attach, catalog and extend his file. If the library is on tape, the user must build a new library each time an EDITLIB is performed on the library, using the old library as a source.

GENERAL

Index to user's libraries on a random file (six words):

WORD	ADDRESS
1	Entry Point Name Table
2	External Reference Table
3	Program Number Table
4	Program Name Table
5	Entry Point and External Reference List
6	Unreferenced PRU Count (not an address)

Format of user's library on a sequential file:

RECORD	CONTENTS
1	This record is three words long: 1. ***LIBRARY 2. Library Name (blank filled) 3. Date of creation
2	Entry Point Name Table
3	Entry Point and External Reference List
4	External Reference Table
5	Program Number Table
6	Program Name Table
7 to EOF	Routines

SYSTEM SECURITY

EDITLIB will use the reprieve function to protect the system files and its scratch files. Under any normal or abnormal termination condition, EDITLIB will have control turned over to a program that will return or release all EDITLIB internal files.

MDI (MOVE SYSTEM DIRECTORY)

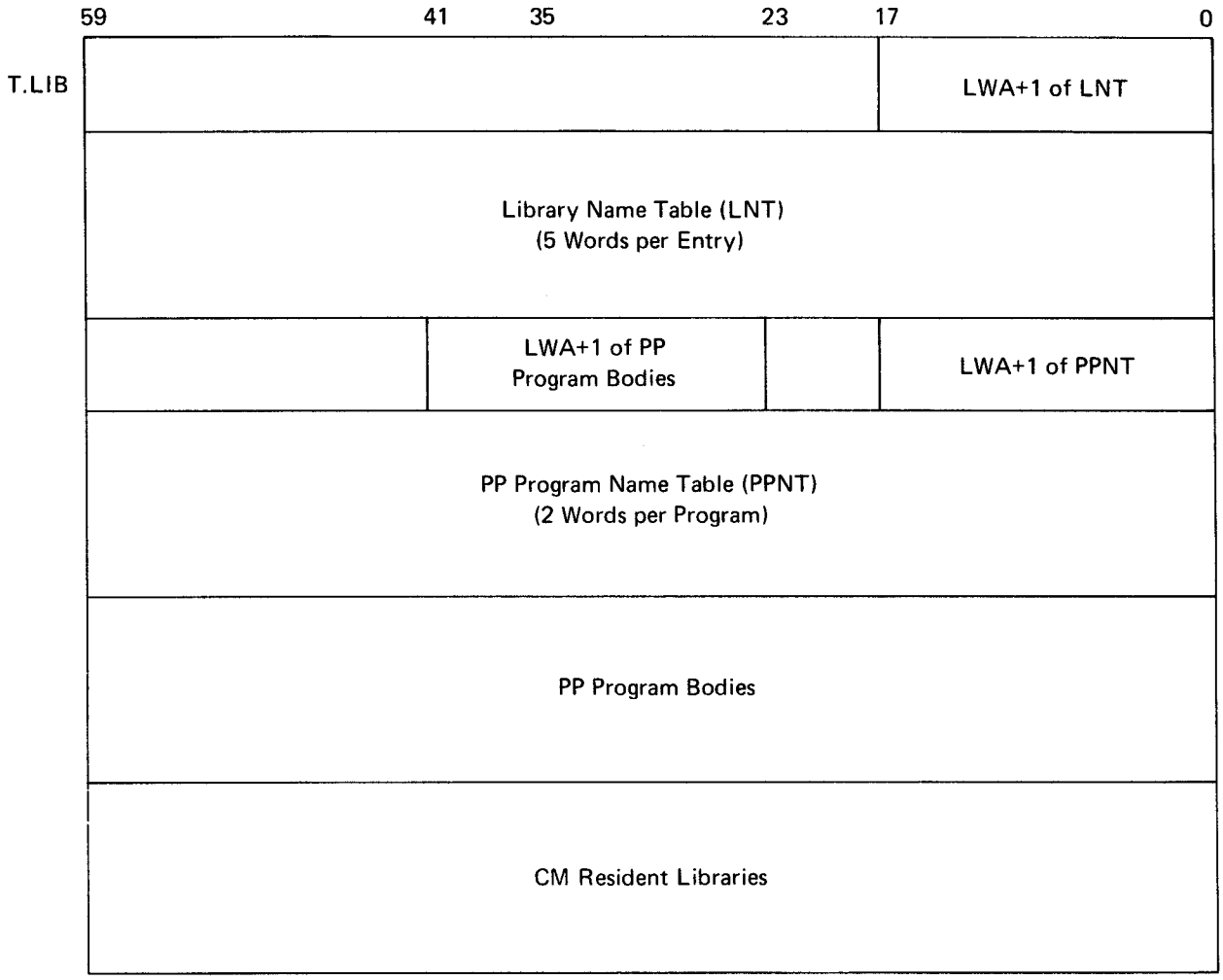
This PP overlay includes the program 6MD which is used to control changes to the CMR directory. If 6MD is deleted, the running system may not be changed, but all other functions can be performed. MDI will perform the following tasks:

1. Request field length for directory in CMR.
2. Set directory change flag.
3. Get length of directory.
4. Place CM address into LNT for CM resident libraries.
5. Copy directory to control point.
6. Copy directory to CMR.
7. Check CP activity flags at each control point.
8. Write current directory file.
9. Attach FNT to ZZZZZ06 to EDITLIB control point.
10. Monitor EDITLIB error flag after GO.

A parameter list will be provided in EDITLIB for MDI. This list will contain MDI request code and all necessary information. MDI will return information to EDITLIB in this list. This list is the MDI INPUT/OUTPUT register.

TABLE FORMATS

CMR Directory

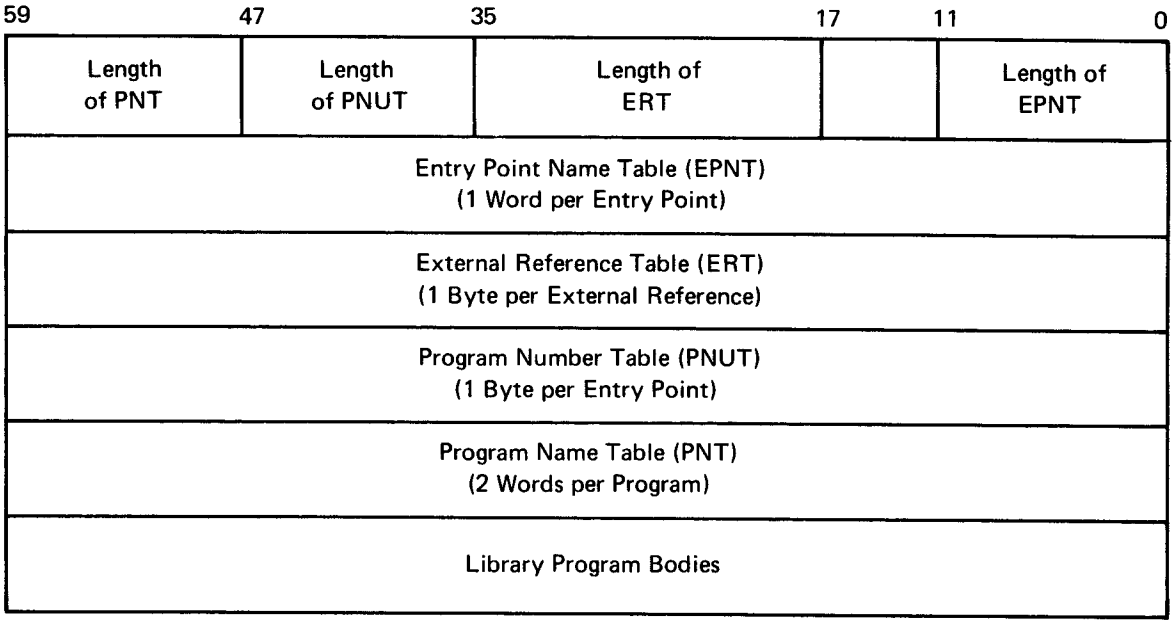


CMR DIRECTORY NOTES

ENTRY	FLAG BITS	
LNT		
	- 0	CM resident library
	- 1	ECS resident library
	- 2	unused
	- 3	EDITEXTEND/SYSTEM library
	4-16	unused
	- 17	1 = use disk address only

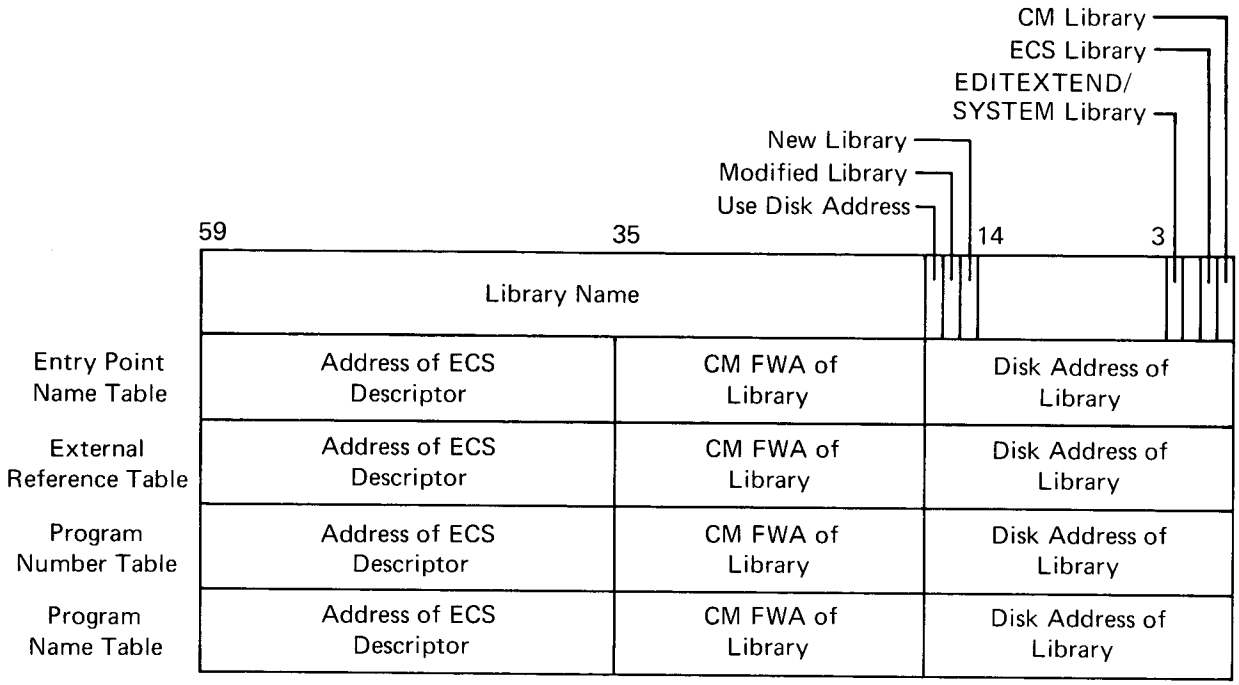
PPNT		
	0	EDITEXTEND/SYSTEM record
	1	ECS resident
	2	CM resident
	3	control card callable
EPNT		
	0-2	000 = relocatable program 001 = overlay 010-111 = unused
	- 3	control card callable
PNT		
	11-0	EFL command can override FL specification
	12-14 R	xx0 program on EDITEXTEND xx1 program not on EDITEXTEND 00x disk address 01X CM address 10x ECS address

CMR Library Format



Notes: All information that is a part of the system directory and resides in ECS will be on the ECS file called ZZZZZ06.

Library Name Table Format



Word 1:

Bit 0 — 0 Library is not CM resident.

1 Library is CM resident.

Bit 1 — 0 Library is not ECS resident.

1 Library is ECS resident.

If bits 0 and 1 are both zero, the directory is on disk.

Bit 3 — 0 Library on file ZZZZZ04

1 Library on file ZZZZZ03

Bit 15 — Library is new.

Bit 16 — Library is modified.

Bit 17 — 0 Residency addresses are valid.

1 Use disk address only.

Bits 15, 16, and 17 are set and cleared by EDITLIB only. They are used when EDITLIB is changing a directory that contains many libraries. Any routine that is accessing the library that has these bits set must use the disk address to obtain any library table and any routine that is part of this library. These bits are necessary due to the dynamic allocation of ECS and the nature of the new directory.

Bits 18-59 The library name; can contain up to 7 alphanumeric characters of which the first must be alphabetic.

Word 2:

Address word for the Entry Point Name Table.

Word 3:

Address word for the External Reference Table.

Word 4:

Address word for the Program Number Table.

Word 5:

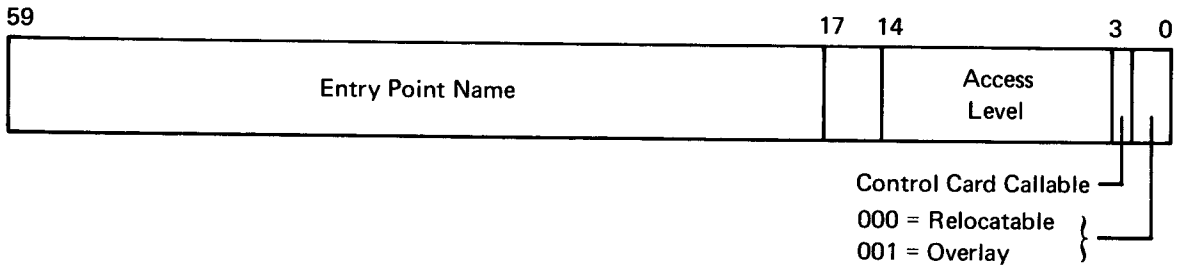
Address word for the Program Name Table.

Example:

Bits 2	1	0	
0	0	0	Library is disk resident.
0	0	1	Library is CM resident.
0	1	0	Library is ECS resident.

Note: Fields not being used contain zero.

Entry Point Name Table Format



Bits 0-2

000 The entry point belongs to a relocatable program

001 The entry point belongs to an overlay.

Bit 3

0 This entry point cannot be called from a control card.

1 This entry point can be called from a control card.

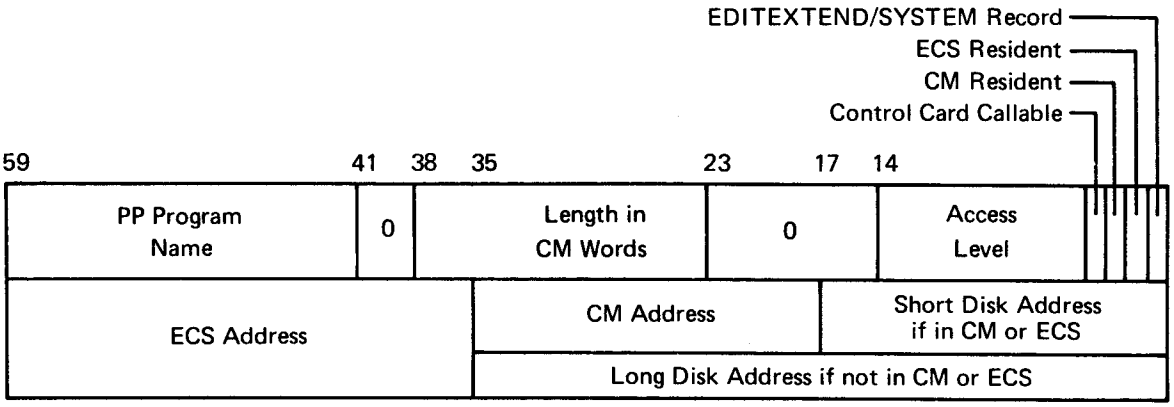
Bits 4-14

These are the Access Level bits for INTERCOM.

Bits 18-59

Entry Point Name value is equal to its position in the table relative to the start of the table. The table is sorted alphabetically.

PP Program Name Table Format



Word 1:

Bit 0 0 If the record is on (pfn) ZZZZZ04

1 If the record is on (pfn) ZZZZZ03.

Bit 1 0 Record/routine is not ECS resident.

1 Record/routine is ECS resident.

Bit 2 0 Record/routine is not CM resident.

1 Record/routine is CM resident.

Bit 3 0 This program cannot be called from a control card.

1 This program can be called from a control card.

Bits 4-14 Access Level bits for INTERCOM.

Word 2:

If the routine is disk resident, there will be a special disk address in bits 0-35 for use by PP resident. The CM address is absolute.

Fields not being used contain zero.

External Reference Table Format

Entry Point Number + 1	Entry Point Number + 1	Entry Point Number + 1	Entry Point Number + 1	Entry Point Number + 1 or Continuation

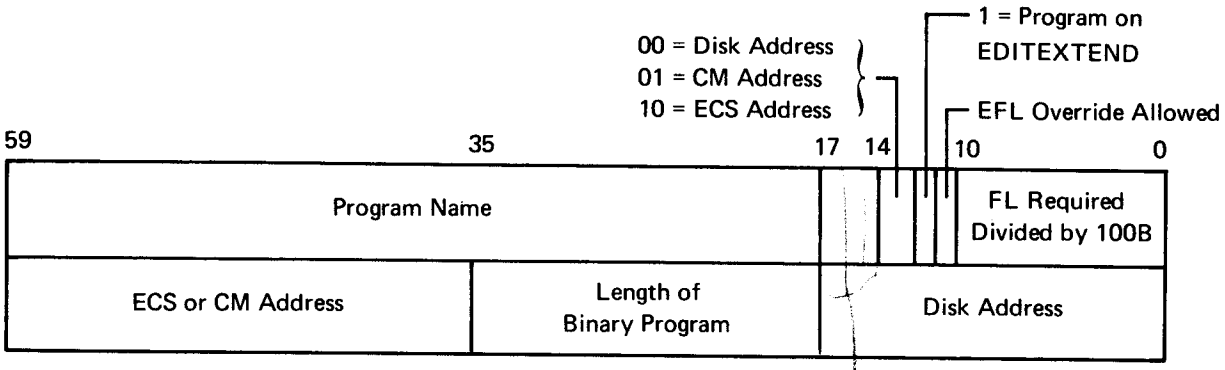
The first entry of this table is associated with the first entry in the Entry Point Name Table. The chain of External References is terminated by 12 bits of zero or by the 5th parcel not having bit 11 set to 1. If bit 11 is set to 1, the value contained in this parcel, less bit 11, is the relative address of the continuation word. The address is relative to the FWA of the External Reference Table plus the Entry Point Name Table. All external calls made directly by a routine will be shown in this table.

Program Number Table Format

Parcel 0 Relative PNT Address	Parcel 1 Relative PNT Address	Parcel 2	Parcel 3	Parcel 4
Parcel 5	Parcel 6			Parcel n

This table holds pointers relative to the Program Name Table for each Entry Point Number. The Entry Point Number is the parcel number in this table. There are 5 parcels per word, stored from left to right in the sequential bytes of a word. The value in each parcel is the relative address of the programs's entry in the Program Name Table. Address is relative to the start of the Program Name Table.

Program Name Table Format



Word 1:

Bits 0-10 — Contain the amount of field length divided by 1000 to execute this program. If not specified during an EDITLIB run, it will be set to IP.SFL/1000.

Bit 11 — INTERCOM field length override bit:

- 0 do not override
- 1 may be overridden

Bit 12

- 1 Program is on (pfn) ZZZZZ03.
- 0 Program is on (pfn) ZZZZZ04.

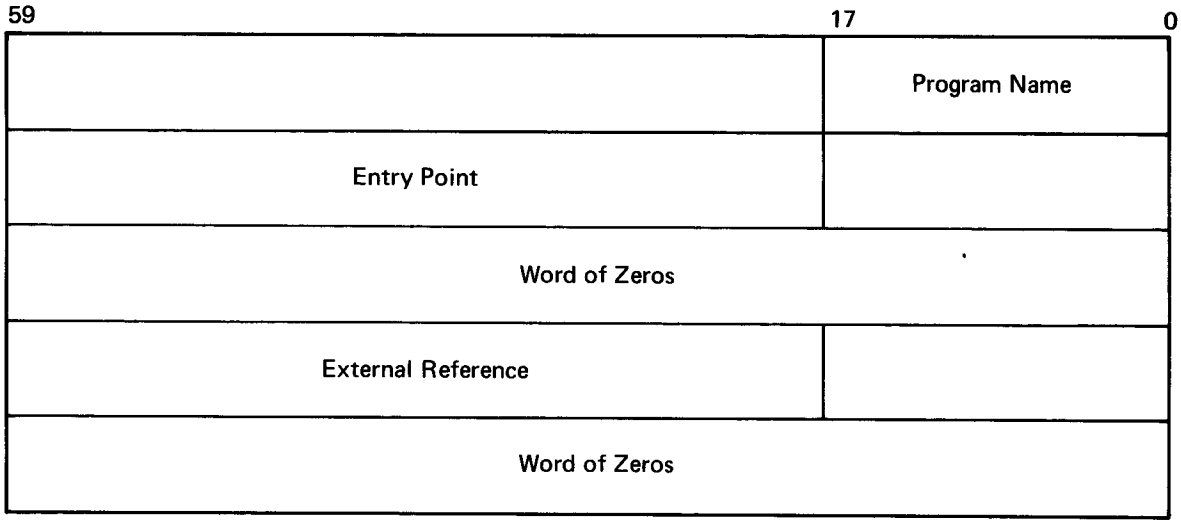
Bits 13-14

- 00 Disk address
- 01 CM address
- 10 ECS address

Bits 18-59 — Program Name in display code.

The CM address is relative to the LWA+1 of this table. Fields not being used contain zero.

Entry Point and External Reference List Format



The Entry Point and External Reference List has an entry for each program that is a part of the library. This list will be maintained by EDITLIB. The entry point and external references for each routine will be in alphabetical order and will be left justified with zero fill.

The index to this record, when the library is not part of the running system, will be in the index record for the random file. See figure 8-2. When the library becomes part of the running system, the index to the Entry Point and External Reference List for the library is destroyed. The Entry Point and External Reference List will be placed following the Entry Point Name Table record. By using the index to the Entry Point Table it will be possible to access the External Reference List. A CIO Read Skip request will be used with the index to the Entry Point Name Table. Then a read with a zero index will be used to obtain the Entry Point and External Reference List.

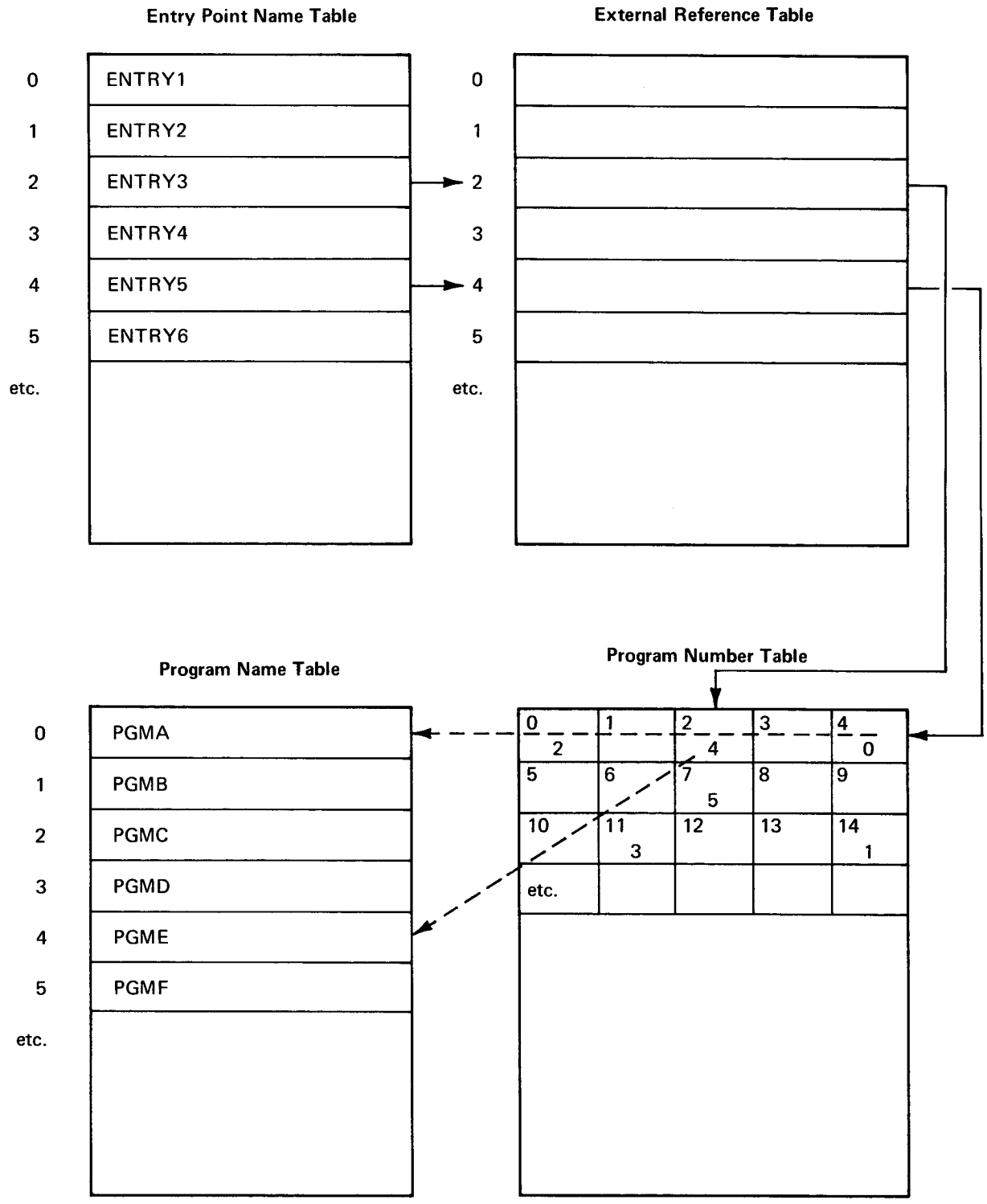


Figure 8-2. Library Table Interfaces

EXAMPLES

Deadstart Creation

```
READY(NEWSYS,NEW)
TRANSFER(17,SYSTEM)
TRANSFER(CMR,NEWCMR)
SKIPR(1,SYSTEM)
TRANSFER(*,SYSTEM)
LIBRARY(NUCLEUS,NEW)
ADD(*,SYSTEM,LIB=NUCLEUS,AL=1)
REPLACE(EDITLIB,LGO,AL=1,FL=40000)
REPLACE(LOADER,LGO,AL=1,FL=20000)
FINISH.
LIBRARY(FORTRAN,NEW)
ADD(*,NEWFTN,AL=0,FL=45000)
FINISH.
LISTLIB(*,NEWSYS,NUCLEUS)
LISTLIB(*,NEWSYS,FORTRAN)
ADD(*,SYSTEM)
REPLACE(1SP,LGO,AL=0,CM)
LISTLIB(*,SYSTEM)
LISTLIB(*,SYSTEM)
LISTLIB(*,SYSTEM,NUCLEUS)
LISTLIB(*,SYSTEM,FORTRAN)
COMPLETE.
ENDRUN.
```

System EDITLIB

```
READY(SYSTEM,OLD)
ADD(*,NEWPP)
LIBRARY(SCOPE1,NEW)
ADD(*,SYSTEM,LIB=NUCLEUS,DS,AL=1,FL=40000,FLO=1)
REPLACE(LOADER,NEW,DS,AL=1,FL=20000)
FINISH.
LISTLIB(*,SYSTEM,NUCLEUS)
COMPLETE.
READY(SYSTEM,OLD)
ADD(1ZZ,LGO,AL=0)
LIBRARY(NUCLEUS,OLD)
FINISH.
COMPLETE
ENDRUN.
```

STANDARD SCOPE CHARACTER SETS A

The character set selected when the system is installed should be compatible with the printers.

With an installation parameter, the installation keypunch format standard can be selected as 026 or 029; the installation parameter can also allow a user to override the standard; a user may select a keypunch mode for his input deck by punching 26 or 29 in columns 79 and 80 of his JOB card or any 7/8/9 end-of-record card. The mode remains set for the remainder of the job or until it is reset by a different mode selection on another 7/8/9 card.

CDC 63-CHARACTER SET

Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD	Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD
00	(none)†			16	40	5	5	5	05
01	A	12-1	12-1	61	41	6	6	6	06
02	B	12-2	12-2	62	42	7	7	7	07
03	C	12-3	12-3	63	43	8	8	8	10
04	D	12-4	12-4	64	44	9	9	9	11
05	E	12-5	12-5	65	45	+	12	12-8-6	60
06	F	12-6	12-6	66	46	-	11	11	40
07	G	12-7	12-7	67	47	*	11-8-4	11-8-4	54
10	H	12-8	12-8	70	50	/	0-1	0-1	21
11	I	12-9	12-9	71	51	(	0-8-4	12-8-5	34
12	J	11-1	11-1	41	52	)	12-8-4	11-8-5	74
13	K	11-2	11-2	42	53	$	11-8-3	11-8-3	53
14	L	11-3	11-3	43	54	=	8-3	8-6	13
15	M	11-4	11-4	44	55	blank	no punch	no punch	20
16	N	11-5	11-5	45	56	, (comma)	0-8-3	0-8-3	33
17	O	11-6	11-6	46	57	. (period)	12-8-3	12-8-3	73
20	P	11-7	11-7	47	60	≡	0-8-6	8-3	36
21	Q	11-8	11-8	50	61	[	8-7	8-5	17
22	R	11-9	11-9	51	62	]	0-8-2	12-8-7	32
23	S	0-2	0-2	22	63	:(colon)†	8-2	8-2	00*
24	T	0-3	0-3	23	64	≠	8-4	8-7	14
25	U	0-4	0-4	24	65	→	0-8-5	0-8-5	35
26	V	0-5	0-5	25	66	v	11-0 or 11-8-2	11-0 or 11-8-2	52
27	W	0-6	0-6	26					
30	X	0-7	0-7	27	67	∧	0-8-7	12	37
31	Y	0-8	0-8	30	70	↑	11-8-5	8-4	55
32	Z	0-9	0-9	31	71	↓	11-8-6	0-8-7	56
33	0	0	0	12	72	<	12-0 or 12-8-2	12-0 or 12-8-2	72
34	1	1	1	01					
35	2	2	2	02	73	>	11-8-7	0-8-6	57
36	3	3	3	03	74	≤	8-5	12-8-4	15
37	4	4	4	04	75	≥	12-8-5	0-8-2	75
					76	¬	12-8-6	11-8-7	76
					77	;(semicolon)	12-8-7	11-8-6	77

†When the 63-Character Set is used, the punch code 8-2 is associated with display code 63, the colon. Display code 00_8 is not included in the 63-Character Set and is not associated with any card punch. The 8-6 card punch (026 keypunch) and the 0-8-4 card punch (029 keypunch) in the 63-Character Set are treated as blank on input.

*Since 00 cannot be represented on magnetic tape, it is converted to BCD 12. On input, it will be translated to display code 33 (number zero).

CDC 64-CHARACTER SET

Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD	Display Code	Character	Hollerith (026)	Hollerith (029)	External BCD
00	: †	8-2	8-2	00*	40	5	5	5	05
01	A	12-1	12-1	61	41	6	6	6	06
02	B	12-2	12-2	62	42	7	7	7	07
03	C	12-3	12-3	63	43	8	8	8	10
04	D	12-4	12-4	64	44	9	9	9	11
05	E	12-5	12-5	65	45	+	12	12-8-6	60
06	F	12-6	12-6	66	46	-	11	11	40
07	G	12-7	12-7	67	47	*	11-8-4	11-8-4	54
10	H	12-8	12-8	70	50	/	0-1	0-1	21
11	I	12-9	12-9	71	51	(	0-8-4	12-8-5	34
12	J	11-1	11-1	41	52	)	12-8-4	11-8-5	74
13	K	11-2	11-2	42	53	$	11-8-3	11-8-3	53
14	L	11-3	11-3	43	54	=	8-3	8-6	13
15	M	11-4	11-4	44	55	blank	no punch	no punch	20
16	N	11-5	11-5	45	56	, (comma)	0-8-3	0-8-3	33
17	O	11-6	11-6	46	57	. (period)	12-8-3	12-8-3	73
20	P	11-7	11-7	47	60	≡	0-8-6	8-3	36
21	Q	11-8	11-8	50	61	[	8-7	8-5	17
22	R	11-9	11-9	51	62	]	0-8-2	12-8-7	32
23	S	0-2	0-2	22	63	%	8-6	0-8-4	16
24	T	0-3	0-3	23	64	≠	8-4	8-7	14
25	U	0-4	0-4	24	65	→	0-8-5	0-8-5	35
26	V	0-5	0-5	25	66	∨	11-0 or 11-8-2	11-0 or 11-8-2	52
27	W	0-6	0-6	26					
30	X	0-7	0-7	27	67	∧	0-8-7	12	37
31	Y	0-8	0-8	30	70	↑	11-8-5	8-4	55
32	Z	0-9	0-9	31	71	↓	11-8-6	0-8-7	56
33	0	0	0	12	72	<	12-0 or 12-8-2	12-0 or 12-8-2	72
34	1	1	1	01					
35	2	2	2	02	73	>	11-8-7	0-8-6	57
36	3	3	3	03	74	≤	8-5	12-8-4	15
37	4	4	4	04	75	≥	12-8-5	0-8-2	75
					76	¬	12-8-6	11-8-7	76
					77	;(semicolon)	12-8-7	11-8-6	77

†This character is lost on even parity magnetic tape.

*Since 00 cannot be represented on magnetic tape, it is converted to BCD 12. On input, it will be translated to display code 33 (number zero).

ASCII 64-CHARACTER SUBSET*

Display Code	Character	Hollerith (026)	Hollerith (029)	ASCII Code	Display Code	Character	Hollerith (026)	Hollerith (029)	ASCII Code
00	: †	8-2	8-2	072	40	5	5	5	065
01	A	12-1	12-1	101	41	6	6	6	066
02	B	12-2	12-2	102	42	7	7	7	067
03	C	12-3	12-3	103	43	8	8	8	070
04	D	12-4	12-4	104	44	9	9	9	071
05	E	12-5	12-5	105	45	+	12	12-8-6	053
06	F	12-6	12-6	106	46	-	11	11	055
07	G	12-7	12-7	107	47	*	11-8-4	11-8-4	052
10	H	12-8	12-8	110	50	/	0-1	0-1	057
11	I	12-9	12-9	111	51	(	0-8-4	12-8-5	050
12	J	11-1	11-1	112	52	)	12-8-4	11-8-5	051
13	K	11-2	11-2	113	53	$	11-8-3	11-8-3	044
14	L	11-3	11-3	114	54	=	8-3	8-6	075
15	M	11-4	11-4	115	55	blank	no punch	no punch	040
16	N	11-5	11-5	116	56	, (comma)	0-8-3	0-8-3	054
17	O	11-6	11-6	117	57	. (period)	12-8-3	12-8-3	056
20	P	11-7	11-7	120	60	#	0-8-6	8-3	043
21	Q	11-8	11-8	121	61	' (apostrophe)	8-7	8-5	047
22	R	11-9	11-9	122	62	!	0-8-2	12-8-7	041
23	S	0-2	0-2	123	63	%	8-6	0-8-4	045
24	T	0-3	0-3	124	64	" (quote)	8-4	8-7	042
25	U	0-4	0-4	125	65	_ (underline)	0-8-5	0-8-5	137
26	V	0-5	0-5	126	66	]	11-0 or 11-8-2	11-0 or 11-8-2	135
27	W	0-6	0-6	127					
30	X	0-7	0-7	130	67	&	0-8-7	12	046
31	Y	0-8	0-8	131	70	@	11-8-5	8-4	100
32	Z	0-9	0-9	132	71	?	11-8-6	0-8-7	077
33	0	0	0	060	72	[	12-0 or 12-8-2	12-0 or 12-8-2	133
34	1	1	1	061					
35	2	2	2	062	73	>	11-8-7	0-8-6	076
36	3	3	3	063	74	<	8-5	12-8-4	074
37	4	4	4	064	75	\	12-8-5	0-8-2	134
					76	^ (circumflex)	12-8-6	11-8-7	136
					77	; (semicolon)	12-8-7	11-8-6	073

†This character is lost on even parity magnetic tape.

*BCD representation is used when data is recorded on even parity magnetic tape. In this case, the octal BCD/display code correspondence is the same as for the CDC 64-character set.

SCOPE SYMBOL DEFINITION B

SCOPE Text (SCPTEXT) contains all system macros, micros, and symbols used by COMPASS CPU and PPU programs that comprise the SCOPE 3 Operating System. SCPTEXT is made up of three integral common decks; PPSYS, ACTCOM, and CPSYS.

PPSYS provides the definition of all macros, micros, and SCOPE symbols, and contains the PP macros prototypes.

ACTCOM contains all of the CPU program system action request macro prototypes.

CPSYS contains the central processor input/output macros prototypes which call the central processor library routine CPC (Central Program Control).

Peripheral processor text (PPTEXT) contains all the SCOPE symbol definitions and PP macro prototype required by the COMPASS programs CEA (deadstart PP0 save routine) and DSD (system display driver). PPTEXT is made up solely of the common deck PPSYS.

Symbol definitions can have either of the following forms:

name	EQU	absolute expression
name	=	absolute expression

where = is the equivalent of the EQU instruction and the absolute expression is either an absolute value or an expression in which all symbols have absolute values assigned.

PPTEXT symbolic names have the format:

i.mn

i	Identifier; one or two alphabetic characters specifying the category to which the symbol belongs.
mn	Mnemonic; one to six alphanumeric characters suggesting the meaning of the symbol.

PPTEXT IDENTIFIERS

Identifiers that appear in PPTEXT symbolic names follow:

Identifier	Description
C	Byte number in CM word (0-4). C identifiers are used for flags and parameters of 12 bits or less.
CH	Pseudo channel assignments
D	Direct cells
EX	M.ICE parameter values
F	Error flag values
L	Lengths
LE	Length of table entries
M	PPU request of monitor
N	Numbers (quantities of things)
O	Stack processor orders
OV	PPU overlays; mnemonic is the overlay name
P	CM location of pointer words
R	PPU resident entry points
S	Number of bits to right-shift a parameter to right-justify it in a PPU word
T	First word address of CM tables. The system programmer should use the P. definition rather than access the table directly with the T. definition
W	Relative positions in CM tables
X	Executive processor codes

PPTEXT SYMBOLS

The values associated with these symbols are subject to change in the released version of SCOPE 3.4.

C.AFL (4) Byte 4 of word P.STR in CMR area containing the available field length; used by scheduler

C.APF (3) Byte of word P.PFM1 containing the lower 12 bits of FWA of the APF table.

C.APFL (1) Byte of word P.PFM1 containing the maximum number of entries in the APF table.

C.CHRQ (3) Byte in which each bit represents hardware channels 0-13. MTR sets the bit in the byte when a request is made for a reserved channel. JANUS and Stack Processor check this byte to determine whether of not to drop the channel.

C.CHRQ2 (4) Byte in which each bit represents hardware channels 20-33.

C.CMLWA (4) Byte of word P.CMFL containing the length of CM in 100 (octal) word blocks.

C.CPAR (3) Byte of word W.CPAR containing address of last auto recall request.

C.CPCKP (4) Byte of word CPCKP containing a count of the number of check points taken.

C.CPCON (3) Byte in word W.CKP containing console checkpoint request flag.

C.CPDFMC (0) Byte of word W.CPDFMC containing dayfile message count for the job. When count is zero, job is aborted.

C.CPDPV (4) Byte of word W.CPDPV in the control point area containing 2-character dependency identifier associated with this job.

C.CPECFL (4) Byte in word in control point area containing number/1000B of ECS words assigned to control point.

C.CPECRA (3) Byte in word W.CPECS containing RA of ECS direct access area in 1000 (octal) word blocks.

C.CPECSI (3) Byte of word W.CPJCP containing ECS field length from the job card. Used if job is rerun.

C.CPEF (1) Byte in word in control point area containing error flag. If zero, there is no error at this control point; otherwise, C.CPEF may contain a value defined by the F.x symbols.

C.CPEVNT (4) Byte of word W.CPSWP in control point area containing the swapper event bit S.CPEVNT.

C.CPFL(4) Byte in word in control point area which contains central memory field length/100 assigned to this control point.

C.CPFLAG (0)	Byte within word W.CPFLAG in control point area which contains JANUS and rerun flags.
C.CPFLG (0)	Byte in word W.CPSCH containing job swapping flags.
C.CPFLI (4)	Byte of word W.CPJCP containing CM field length from the job card. Used if job is rerun.
C.CPFP (4)	Byte of word W.CPERT containing flags for use by job processing. The JANUS, INTERCOM and EXPORT/IMPORT bits are in this byte.
C.CPFST (2)	Byte in word W.CPERT in control point area. Contains the pointer to the FST entry of the input file assigned to the control point.
C.CPJDA (3)	Job descriptor address in control point area.
C.CPJCA (3)	Left 6 bits in byte of word W.CPSCH containing index for job control area.
C.CPJDT (4)	Bytes in word W.CPJNAM containing absolute address of job descriptor table.
C.CPJNAM (0-3)	Bytes in control point area word W.CPJNAM containing job name (7 characters).
C.CPJQP (1)	Byte in word W.CPSCH containing job queue priority for scheduler.
C.CPLAI (2)	Byte of word W.CPLDR1 containing the loader-already-in flag.
C.CPLDR2 (2)	Byte in word W.CPLDR2 for temporary use by loader.
C.CPLDR3 (3)	The byte in word W.CPLDR1 of the control point area which contains flags to indicate the value of the map option, whether to use the old or new loader, flags for DEBUG, and flags for REDUCE.
C.CPLINK (3)	Byte in word W.CPLINK containing 2-way link to other control points.
C.CPLIB (4)	Byte in word W.CPLDR4 for temporary use by loader.
C.CPLIBP (4)	Byte in word W.CPLS containing a flag set to 1 when 1AJ loads an absolute overlay from the system library.
C.CPLW (0)	Byte in word W.CPLW containing loader flags.
C.CPMSLM (1)	Byte of word W.CPMSLM in the control point area containing mass storage limit in PRUs.
C.CPMNT (4)	Byte of word W.CPSTG containing job 7 and 9-track tape requirements from job card.
C.CPMSRC (3)	Byte of word W.CPMSLM in the control point area (together with byte 4) containing the running PRU count.
C.CPNCSP (4)	Byte in word W.CPCC of control point area. Contains pointer to next control statement to be processed.

C.CPOUT (4) Byte in word W.CPERT chich contains flag indicating JANUS control point.

C.CPPQMS (1) Byte in W.CPPTM containing PP quantum in milliseconds.

C.CPPQS (0) Byte in W.CPPTM containing PP quantum in seconds.

C.CPPRI (2) Byte in word W.CPPRI in control point area; denotes class quantum of job.

C.CPPTMS (4) Byte in W.PPTIME in control point area containing job's PP use time in milliseconds.

C.CPPTS (2) Byte in W.PPTIME in control point area containing job's PP use time in seconds.

C.CPQNT (0) Byte in word W.CPSWP containing scheduled quantum (accrued CP/PP time).

C.CPRA (3) Byte in word W.CPSTAT in control point area which contains reference address (RA)/100 of control point.

C.CPRBID (2) Remote batch job ID field used by INTERCOM; in word W.CPIRB.

C.CPREQ (0) Byte of word W.CPOAE used as a flag to DSD to call REQ when the operator assigns equipment.

C.CPRERN (3) Byte in word W.CPERT containing job rerun priority value.

C.CPRFL (2) Byte in word W.CPSCH containing size of FL reserved by scheduler for the control point.

C.CPRO (4) Byte in word W.CPRO temporarily not in use.

C.CPRPA (1) Byte of word W.CPCC in the control point area which, together with byte C.CPRPA+1, contains the address of the post-recovery (reprieve) function, relative to RA.

C.CPRPRI (2) Byte in word W.CPOAE in control point area which contains rescheduling priority.

C.CPRPV (0) Byte of word W.CPCC in the control point area containing reprieve function check-sum value.

C.CPSITM (0-1) Bytes in W.CPTIME containing swap-in time.

C.CPSLIC (1) Byte in W.CPSLIC containing total CPU time slice allotted.

C.CPSM (2) Byte in word W.CPSM in control point area which contains storage move flag. MTR sets this field non-zero when storagd is to be moved. All PPs at this control point should pause if C.CPSM is non-zero.

C.CPSR (1) Byte of word W.CPSR containing the entry count of the number of stack requests which cause CM access for this control point.

C.CPNFL (3)	Byte of word W.CPCC containing job nominal field length in 100 (octal) word blocks.
C.CPNUM (1)	Byte in PP input register word which contains control point number originating request.
C.CPOAE (4)	Byte of word W.CPOAE containing EST ordinal of equipment assigned by operator.
C.CPORG (4)	Byte of word W.CPSWP containing job origin for scheduler use.
C.CPSSW (4)	Byte in word W.CPSSW contain setting of simulated sense switches.
C.CPSTAT (0)	Byte in a word W.CPSTAT in control point area which MTR uses to note status of control point.
C.CPTIML (0)	Byte in word W.CPTIML in control point area which contains job time limit in 15 bits.
C.CPTLI (1)	First byte of a 15-bit field in word W.CPTIML containing the time limit from the job card to be used if job is rerun.
C.CPTMT (0)	Byte of word W.CPSTG containing current number of 7-track tapes assigned.
C.CPTNT (1)	Byte of word W.CPSTG containing current count of 9-track tapes assigned.
C.CPUAMS (4)	Byte in word W.CPTIME contains CPU-A use time in milliseconds.
C.CPUAS (2)	Byte in word W.CPTIME contains CPU-A use time in seconds.
C.CPUBMS (4)	Byte in word W.CPTIMEB contains CPU-B use time in milliseconds.
C.CPUBS (2)	Byte in word W.CPTIMEB contains CPU-B use time in seconds.
C.CPUQMS (1)	Byte in word W.CPTIME contains CP quantum time in milliseconds.
C.CPUQS (0)	Byte in word W.CPTIME contains CP quantum time in seconds.
C.CPUTA (2)	Byte in word W.CPSWP contains scheduler user table address.
C.CST (2)	Byte in word P.CST in CMR which contains FWA of the channel status table.
C.CSTCB (2)	CST/MMTC conversion table address.
C.CSTCN (1)	CST channel number.
C.CSTL (3)	Byte in word P.CSTL in CMR which contains LWA + 1 of the channel status table.
C.DATACT (2)	Byte of the device activity table entry referring to the current activity of the device.
C.DATDST (0)	Byte of the device activity table entry giving the device status table entry.

C.DATEQP (1)	Byte of the device activity table entry giving the equipment type.
C.DIRFWA (0)	Leftmost of two bytes in P.LIB which contains first word address of library directory.
C.DIRPRU (4)	Byte in second word of a directory entry which contains number of first PRU assigned to record.
C.DIRRBA (2)	Byte in second word of a directory entry containing linkage to RBT word pair defining record in which the record starts.
C.DIRRBN (3)	Byte in second word of a directory entry containing original RBT word pair and byte defining block in which record starts.
C.DIRUNT (1)	Byte in second word of directory entry containing physical unit number (DST ordinal).
C.DPT (4)	Byte contains block number address of T.DPT.
C.DSFLAG (4)	Deadstart parameter byte in P.LIB.
C.ECFLG (0)	Flag byte in ECS file descriptor.
C.ECSBL (2)	Left byte of ECS buffer length field in T.EPBL in CMR pointer area.
C.ECSPL (0)	Left byte of ECS page length field in T.EPBL.
C.ESTAT (0)	Status/assignment byte in EST.
C.ESTCH1 (1)	Equipment channel assignment field in EST.
C.ESTCH2 (2)	Equipment channel assignment field in EST.
C.ESTDST (4)	DST ordinal field in EST entry.
C.ESTMNE (3)	Hardware mnemonic field in EST entry.
C.ESTSD (2)	Special disposition field in 3000 unit record EST entry.
C.ESTDL(1)	Device label RB field in allocatable device EST entry.
C.FABT (0)	Byte of FNT word 3 containing the abort flag.
C.FADEV (3)	Byte of FNT word 2 for an allocatable device file to which no RBT has been assigned. Contains the EST ordinal of the device to which it has been assigned.
C.FALLOC (4)	Byte in FNT word 2 containing allocation style for a file not yet assigned to mass storage.
C.FAPF (4)	Byte of FNT word W.FTYPE containing the APF pointer for a permanent file.
C.FBKLK (1)	Backward link in FNT supplement entries.
C.FCB (3)	Byte in word in FNT mass storage file entries which contains the current RB byte.

C.FCPNUM (3)	Byte in word 1 of a file name table entry containing file's control point assignment.
C.FCPRU (3-4)	Two-byte field in tape file FNT entry containing current PRU count.
C.FCPU (0)	CPU-A/B flag field in input file entry of FNT.
C.FCREC (1)	Record count in card file entry of FNT.
C.FCS (3)	Leftmost of two bytes in word 3 of FNT entry containing code and status field.
C.FDC (2)	Byte in word 3 of FNT entry containing file disposition code.
C.FDEVTP (0)	Device type/equipment code field in FNT entries.
C.FDPCT (0)	Byte of word 3 of the input file name table entry containing the job dependency count.
C.FDPV (1)	Byte of word 3 of the input file name table entry containing the 2 character job dependency identifier.
C.FECFL (2)	Byte of FNT word 2 containing the ECS field length for a job in the input queue.
C.FECS (3)	ECS flag bits in FNT entry (local file).
C.FEQP (0)	Byte in word 2 of FNT entry containing file equipment code.
C.FFETAD (0)	FET address in mass storage entry in FNT.
C.FFL (4)	Byte of FNT word 2 containing the CM field length for a job in the input queue.
C.FFRBA (1)	Byte in word 2 of FNT entry containing address of first RBT word pair for a file on allocatable device.
C.FINFLG (3)	Input flag field in FNT entry for input file.
C.FINTID (3)	INTERCOM user ID field in FNT entry supplement (input file).
C.FIOTID (1)	INTERCOM user ID field in output file FNT entry for output queue.
C.FLBL (3)	Byte in word 2 of FNT entry. For magnetic tape, byte C.FLBL contains upper 12 bits of current physical record unit (PRU) count; lower 12 bits of PRU count are in byte C.FLBL + 1. For punched cards, byte C.FLBL is 12-bit byte containing upper 5 bits of card count within a record being punched; lower 12 bits of card count are in byte C.FLBL + 1.
C.FLINK (3)	Byte in FNT first word containing link bit.
C.FLNKAD (4)	Byte in first word of FNT entry containing link pointer.

C.FLOCID (3) Byte in FNT entry containing INTERCOM user ID (local file).

C.FLOCK (3) Byte in word 1 of FNT entry containing the lock bit.

C.FLPRU (4) Byte in word 2 of FNT entry containing current physical record unit (PRU) position of a file on an allocatable device.

C.FLRBEB (3) Byte in word 2 of FNT entry containing RBT entry and byte at which a file on allocatable device is currently positioned.

C.FLRBWP (2) Byte in word 2 of FNT entry containing address of current RBT word pair for file on allocatable device.

C.FMUJ (3) Multi-user job flag bit field in INTERCOM user FNT entry.

C.FMUJC (3) Multi-user job code in INTERCOM user FNT entry supplement.

C.FNAME (0) Leftmost of four bytes in word 1 of FNT entry containing seven-character file name.

C.FNT (0) Input file tape requirements field in FNT.

C.FPDEV (2) Byte in FNT word 2 of tape FNT entry containing primary device number (EST ordinal).

C.FPFREW (3) Permanent file rewrite bit in local file flag field of FNT.

C.FPPFN (3) Private pack family no. or EST ordinal for local mass storage FNT entry.

C.FPRI (4) Byte in FNT word 1 containing priority of input or output file.

C.FRECCT (0) Card file record count field in word 2 of FNT entry.

C.FRRN (0) Byte of FNT word 2 containing the rerun flag for a job in the input queue.

C.FSC (3) Byte in FNT word 3 containing file security code, indicates whether or not file is open.

C.FSDC (1) Byte in FNT word 3 used when a file has a special disposition.

C.FSDT (0)

C.FSPDIS (3) Special dispose ID in FNT supplement for output file during printing.

C.FTAPE (0) Input file tape requirements in FNT entry word 2.

C.FTL (0) First byte of a two-byte field in FNT word 3 containing the time limit for a job in the input queue.

C.FTPORD (1) T.TAPES ordinal field in word 2 of tape file FNT entry.

C.FVSN (1)	Volume serial no. in FNT supplement of unassigned tape file.
C.F2PC (3)	Punch file 2PC flag in word 2 of FNT entry.
C.HEC (4)	Byte of word P.HEC containing the hardware error count.
C.IBSTAT (4)	
C.IBUFF (2)	Byte in CMR word P.INT.
C.IFBUFR (1)	Byte containing fwa of INTERCOM low speed buffers.
C.IFL (0)	Byte in word P.INT in CMR pointer area.
C.ILSFL (0)	LWA + 1/100 of INTERCOM buffer area.
C.ILTABL (4)	Length of INTERCOM table.
C.INBUF (0)	Address of next linked INTERCOM buffer.
C.INT (0)	INTERCOM up flag.
C.INTDSD (3)	INTERCOM/DSD communications byte.
C.INUSER (2)	Number of INTERCOM users.
C.ITABL (1)	Byte in word P.INT in CMR pointer area.
C.IUAUT (2)	INTERCOM auxiliary user table pointer.
C.IUCCA (0)	INTERCOM user control card buffer pointer.
C.IUEQC (0)	INTERCOM equipment code.
C.IUEST (0)	INTERCOM multiplexor EST pointer.
C.IUFILE (4)	INTERCOM user flags field.
C.IUFL (2)	INTERCOM user job fl/100 (includes positive fl).
C.IUFRB (1)	INTERCOM user-address of first record block of job.
C.IUJDA (0)	INTERCOM job descriptor entry address.
C.IUMJS (0)	INTERCOM user MUJ status.
C.IUPF (2)	INTERCOM partially up flag.

C.IUPFL (3)	FWA/100 of INTERCOM job positive fl.
C.IUPORT (0)	INTERCOM port number.
C.IUSID (2)	INTERCOM user job ID.
C.JCA (2)	Byte in CMR area word P.SCH containing pointer to job control area/10.
C.JCAR (2)	Job aging rate entry in job control area.
C.JCBQ (4)	Job base quantum value.
C.JCCLK (2)	Flag in job control area set by IIB when it fails to bring up a job.
C.JCCNB (0)	Current count of batch job entries in job descriptor table.
C.JCCTB (1)	Current count of batch jobs with non-allocatable devices in JDT.
C.JCEMC (2)	Count of empty FNT entries kept in job control area.
C.JCMAX (1)	Maximum queue priority value.
C.JCMIN (0)	Minimum queue priority value.
C.JCMTB (1)	Maximum number of batch jobs with non-allocatable devices.
C.JCMXB (0)	Maximum number of batch jobs.
C.JCNJI (4)	Number of ready batch jobs in input queue.
C.JCNTJ (3)	Number of ready tape jobs in input queue.
C.JCPRF (2)	Priority flag byte in job control area.
C.JCQP(3)	Quantum priority value in control point area.
C.JDBP (3)	Job base priority in job descriptor entry.
C.JDCPN (0)	Control point number assigned to job in this JDT entry.
C.JDCPT (1)	Accumulated CPU time by job in this JDT entry.
C.JDEQC (0)	Equipment code byte in JDT entry.
C.JDFL (3)	Job fl including positive field length.
C.JDFLG (2)	Job flag word in JDT entry.

C.JDFRB (1)	First RBT word pair for job.
C.JDID (0)	User ID for remote batch job entries in JDT.
C.JDIFLG (2)	INTERCOM user flag field in JDT entry.
C.JDIUTA (2)	Address of new user table for INTERCOM user.
C.JDJST (0)	Job status and job class byte in JDT entry.
C.JDLNK (3)	Address of next job descriptor entry in queue.
C.JDLPFL (4)	Length/100 of positive fl.
C.JDOPF (3)	Operator flag byte in JDT entry.
C.JDORD (1)	Job descriptor ordinal.
C.JDORG (4)	Job origin of job having this JDT entry.
C.JDPFL (4)	Positive fl address.
C.JDPFM (1)	Permanent file permission bits.
C.JDRU (4)	Resource usage (PP/CP) by job.
C.JDT (4)	Pointer to JDT in word P.SCH in CMR pointer area.
C.JDTIN (1)	Time job entered job descriptor table (time in).
C.JDTL (2)	Time limit for job.
C.JQP (1)	Job queue priority byte in word P.STR in CMR area.
C.LEJDT (0)	Byte in P.SCH which contains length of each JDT entry.
C.LJDT (3)	Byte in P.SCH which contains the length of the Job Description Table.
C.LRD (3)	Bytes in CMR containing logical record definition table address and length.
C.MAILF (0)	Byte in CMR word P.MAIL containing pointer to scheduler mailbox buffer.
C.MAILL (1)	Byte in CMR word P.MAIL containing size of scheduler mailbox buffer.
C.MVFG (1)	Storage move flag for real time programs.
C.NCP (4)	Byte of word P.NCP containing the number of control points.

C.NFL (0) Needed field length (Scheduler) in word P.STR in CMR area.

C.NOVA (1) Byte of word P.NOVA containing the FWA/10 of the NOVA (Interactive Graphics) table.

C.NOVAL (2) Byte of word P.NOVA containing the length of the NOVA (Interactive Graphics) table.

C.NPP (3) Byte of word P.NPP in CMR containing the number of PPs in the configuration.

C.PCFLAG (4) Flag byte in RBTC header word.

C.PCOM (4) FWA of PP communication areas.

C.PDFLAG (4) Flag byte in PFD header word.

C.PFACT (2) Permanent file activity count.

C.PFCNT (2) APF entry byte count #1.

C.PFCNT2 (1) APF entry byte count #2.

C.PFCY (3) File cycle number attached (APF entry).

C.PFD1 (0) APF entry byte

C.PFD2 (1) APF entry byte Permanent file

C.PFD3 (2) APF entry byte directory pointer.

C.PFD4 (3) APF entry byte

C.PFFNT (4) FWA of permanent file FNTs.

C.PFLAG (4) Attached permanent file usage flag byte.

C.PFMCH (4) Permanent file interlock (toggle) byte in word P.PFM1.

C.PFQ (4) APF entry byte – PFQ address.

C.PFRBT (0) First RBT address of attached permanent file cycle (APF).

C.PJTFWA (3) FWA of peripheral job delay stack parameter list.

C.PJTLWA (4) LWA of peripheral job delay stack in P.PJT of CMR pointer area.

C.PPFAF (2) The byte in the PP status word which contains the field access flag.

C.PPFWA (1000) Location in PP memory at which PP resident is to begin execution of a primary overlay. Used both as the second parameter in IDENT card and in address field of ORG on all primary PP overlays.

C.PPSWA (2000) Location in PP memory for beginning execution of a secondary overlay. Used same as C.PPFWA.

C.PPTWA (3000) Location in PP memory at which to begin execution of a tertiary overlay. Used same as C.PPFWA.

C.PP4WA (4000) Location in PP memory at which to begin execution of a fourth level overlay.

C.PP5WA (5000) Location in PP memory at which to begin execution of a fifth level overlay.

C.PP6WA (6000) Location in PP memory for beginning execution of a sixth level overlay.

C.PP7WA (7000) Location in PP memory at which to begin execution of a seventh level overlay.

C.RBRA (3) Byte in RBR header containing permissable allocation.

C.RBRAD (0) Byte in word P.RBR containing FWA of the RBR area.

C.RBRAL (3) Byte of word W.CBRUNT of the RBR table that designates the allocation stype. If the high order bit of this byte is set then files without specified allocation style may go on this device.

C.RBREST (2) Byte of word W.RBRLAV of the RBR table containing the EST ordinal.

C.RBRLAV (3) Byte in second word of each RBR header containing count of unassigned record blocks defined in the RBR.

C.RBRTPA (0) Byte in first word of each RBR header defining device type referenced.

C.RBRUNT (1) Byte in word one of RBR header containing DST ordinal.

C.RBTAL (2) Byte in first word of RBT word pair containing allocation type of file.

C.RBTBIT (4) Byte of first word of the RBR word-pair containing flags.

C.RBTCL (3) Byte in word P.PFM2 containing the number of PRU/8 in RBTC.

C.RBTC1 (0) See C.RBTC3.

C.RBTC2 (1) See C.RBTC3.

C.RBTC3 (2) These bytes in P.PFM3 contain the current end-of-information pointer for the RBTC used by the permanent file system.

C.RBTFB (1) Byte in first word of RBT word pair containing byte number of first record block address.

C.RBTFL (2)

C.RBTPRU (3) Byte in first RBT word assigned to each file which defines last + 1 PRU assigned to that file.

C.RBTRBR (1) Byte in first word of RBT word pair containing RBR ordinal for the file.

C.RBTWPL (0) Byte in each RBT word pair containing linkage to next RBT word pair for that file.

C.RCL (3)

C.RFL (2) Reserved field length (Scheduler).

C.RMS (2) Byte of word P.RMS in CMR containing the starting address (divided by 8) of the RMS table.

C.RMSL (3) Byte of word P.RMS in CMR containing the length of the RMS table.

C.RPTCP (2) Control point number byte in removable pack table (RPT) entry.

C.RPTEST (0) EST ordinal byte in RPT entry.

C.RPTFST (1) FNT address byte in RPT entry.

C.RPTPKN (3) Removable pack device code (AM or AP) in display code.

C.RPTVID (0) Visual identifier of current pack.

C.RQSCS (0) Byte in stack pointer word (P.RQS) containing a count of the stack entry word pairs.

C.RQSFS (2) Byte in stack pointer word (P.RQS) containing address of first stack entry.

C.RSTA (2) Byte in M.EREQS Monitor function containing PP available flag.

C.RSTRA (1) Byte in M.EREQS Monitor function containing request accepted field which is used for Monitor-PP Resident communication.

C.RSTU (4) Byte in M.EREQS Monitor function containing unit number (DST ordinal).

C.RSTWP (3) Byte in M.EREQS Monitor function containing request stack word pair address.

C.RWPPCC (3) Byte in READP/WRITEP control word into which stack processor for control phase 1 places channel number to be used in data transmission.

C.RWPPCF (0) Byte containing phase control flag in control word for READP/WRITEP.

Phases:

0 = Request in stack
1 = Set channel
2 = Channel set, await transmission
3 = Transmission in progress
4 = Order completed.

C.RWPPLW (2) Byte in READP/WRITEP control word containing LWA + 1 of data transmitted. It is updated by PP Resident during order each time it completes phase 3.

C.RWPPST (3) In READP/WRITEP control word, operation status available in phase 4 is contained in this byte.

C.RWPPWC (4) In READP/WRITEP control word, word count for transmission during phase 3 is contained in this byte.

C.RWPPWT (1) In READP/WRITEP control word, total number of words transmitted during all phase 3s is cumulated by PP Resident in this byte.

C.SCHPT (3) FWA/10 of CMR statistics table.

C.SDL (0) Byte in word P.PFM2 containing the number of entries per subdirectory.

C.SDTL (0) Byte in word P.PFM1 containing the number of permanent file subdirectories.

C.SEQ (2) Byte in word P.SEQ containing the FWA/100 of the sequencer table.

C.SEQL (3) Byte in word P.SEQ containing the length of the sequencer table.

C.SRS (1) Byte in word P.SCH containing FWA/10 of scheduler exchange package.

C.STCPU (4) Byte in word 1 of stack request containing control point and unit number.

C.STFB (3) Flag byte in word 2 of a PP request stack.

C.STG (4) Input staging pointer to tape staging table (T.STG).

C.STGFLG (2) Tape staging scheduled by operator (on/off option).

C.STGMT (0) 7-track tape byte for T.STG.

C.STGNT (1) 9-track tape byte for T.STG.

C.STMF (3) Scheduler.

C.STO (3) Specific order in word 1 of a stack request: high order 6 bits are a record level number when relevant (order = O.SKF).

C.STPFW (2) Next address in PP memory for data in word 2 of READP/WRITEP stack request. Used by PP Resident to compute byte count and as FWA for data transmission in a call to R.READP or WRITEP.

C.STPLW (4) Last address in PP memory for data in word 2 of a READP/WRITEP stack request. Used by PP Resident to compute byte count in call to R.READP or R.WRITEP.

C.STPMS (1) Location of message buffer of PP in word 2 of a READP/WRITEP stack request. First 3 words are used for communication with stack processor.

C.STPPRU (2) PRU number at which to begin data transmission in word 1 of a stack request with flag set for no FNT.

C.STPRBA (0) RBT address of word pair containing record block at which to begin data transmission in word 1 of stack request with flag set for no FNT.

C.STPRBN (1) RBT ordinal of record block at which to begin data transmission in word 1 of stack request with flag set for no FNT.

C.STPWC (0) Count of bytes to be transmitted in word 1 of READP/WRITEP stack request.

C.SWPECS (2) ECS swap flags.

C.TFLGS (0) Tapes table flag bits.

C.VRNFIN (1) RA within VRN (VSN) buffer.

C.VRNFNT (4) FNT address within VRN buffer.

C.VRNFUL (4) VRN buffer full flag.

C.VRNFWA (0) FWA/10 of VRN buffer pointer in CMR word P.VRNBUF.

C.VRNINT (3) VRN buffer interlock byte in CMR word P.VRNBUF.

C.VRNNXT (3) Next link number within VRN buffer.

CH.CPA (34) Pseudo-channel for control point area interlock.

CH.DMP (37) Pseudo-channel used by DMP.

CH.EST (40) Equipment status table pseudo-channel (see CH.TAPE).

CH.FNT (15) File Name Table pseudo-channel; possession of this channel interlocks word one of FNT.

CH.FST (14)	File Status Table pseudo-channel; possession of this channel provides an interlock of words two and three of the FNT (alternately called FST).
CH.ICOM (41)	INTERCOM/SCOPE interlock channel.
CH.IEMBF (42)	INTERCOM low speed empty chain channel.
CH.IHSMT (46)	INTERCOM high speed empty chain channel.
CH.IHUSR (45)	INTERCOM high speed user tables channel.
CH.INS (36)	Pseudo-channel reserved for installation use.
CH.IUSER (43)	INTERCOM user pseudo-channel.
CH.LIB (16)	Library directory pseudo-channel; only used by GP250 software package (NOVA).
CH.PFM (35)	Pseudo-channel used by permanent file manager.
CH.RBT (17)	Record Block Table pseudo-channel.
CH.SCH (44)	Scheduler pseudo-channel.
CH.TAPE (40)	Tapes table pseudo-channel.
D.BA (40)	D.BA through D.BA+4 contain first word of file environment table (FET) located by relative address in low order 18 bits of input register.
D.CPAD (76)	Typically contains address of control point area currently in use by PP. A primary overlay usually stores the address as part of its initialization.
D.DTS (37)	High order 6 bits of D.DTS contain device type found in high order portion of byte 0 of second word of FNT. Low order 6 bits of D.DTS contain allocation type found in low order portion of byte 0 of second word of FET.
D.EST (32)	D.EST through D.EST+4 contain EST entry in process.
D.FA (57)	Contains address of second word of FNT entry in process.
D.FIRST (60)	D.FIRST and D.FIRST+1 contain 18-bit CM address specifying beginning of a circular buffer (contents of first pointer (word 2) from a FET).
D.FL (56)	Central memory field length/100 of control point to which PP is currently attached. Primary overlay usually stores field length as part of initialization.
D.FNT (20)	D.FNT through D.FNT+9 contain word 2 and 3 of FNT entry for file in process. Word 2 and 3 are alternately referred to as the FST entry.

D.HN (71) Constant value of 100 (octal). Generally, D.HN is preset by a primary overlay for use by a secondary overlay.

D.IN (62) D.IN and D.IN + 1 contain IN pointer (word 3) from FET.

D.JECS (45) Used by 2TJ to return ECS field length requirement to calling program.

D.JFL (37) Used by 2TJ to return CM field length requirement to calling program.

D.JPAR (32) PP word containing the first word address of the job parameter table.

D.JPR (46) Used by 2TJ to return computed priority to calling program.

D.LIMIT (66) D.LIMIT and D.LIMIT + 1 contain 18-bit address specifying LWA + 1 of circular buffer (contents of LIMIT pointer (word 5) from FET).

D.OUT (64) D.OUT and D.OUT + 1 contain OUT pointer (word 4) from FET.

D.PPIR (74) Contains CM address of PP input register. Initialized at deadstart time and must never be altered.

D.PPIRB (50) D.PPIRB through D.PPIRB + 4 hold the contents of PP input register. Primary overlay usually stores input register contents as part of initialization.

D.PPMES1 (75) Contains CM address of first word of PP message buffer. Initialized at deadstart time and must never be altered.

D.PPONE (70) Contains constant 1. Generally, D.PPONE is preset by a primary overlay for use by a secondary overlay.

D.PPSTAT (77) PP word containing the CM address of the PP status word for this PP.

D.RA (55) Contains central memory reference address/100 of control point to which PP is attached. Primary overlay usually stores address as part of initialization.

D.TH (72) Contains constant 1000 (octal). Generally, D.TH is preset by a primary overlay for use by a secondary overlay.

D.TR (73) Contains constant 3. Generally, D.TR is preset by a primary overlay for use by a secondary overlay.

EX.CMSM (0) CM storage move.

EX.ECOVL (2) Load ECS resident routine.

EX.ECSM (1) ECS storage move.

EX.FLHB (15) Flush ECS buffer.

EX.MVIN (13)	Move into ECS.
EX.MVOUT (14)	Move out of ECS.
EX.RELEB (10)	Release ECS file buffer.
EX.RELSB (12)	Release system buffer.
EX.REQEB (7)	Request ECS file buffer.
EX.REQSB (11)	Request system buffer.
EX.SCH (5)	Standard scheduler call.
EX.SCH1 (6)	Scheduler storage request.
EX.SPM (3)	SPM call.
F.ERAR (2)	Value of error flag set for CP arithmetic error abort. Sensed by MTR; error message is written by 1EJ.
F.ERCC (11)	Error flag value set for a control card error.
F.ERCP (4)	Value of error flag set for CP abort. F.ERCP used if CP program aborts execution; program must write a message to dayfile.
F.ERECP (12)	Value of error flag set when MTR detects permanent parity in ECS during ECS storage move. 1EJ writes a message to dayfile.
F.EREX (11)	Value of error flag set by 1AJ when it detects a control card error, or by MTR when it gets an ABT request with the exit(s) bit (bit 36) set.
F.ERHANG (16)	Error flag for job hung in automatic recall.
F.ERJC (13)	Error flag for a job card error.
F.ERK (7)	Value of error flag in control point area set when operator types "n.KILL". There will be no output from this job.
F.ERMSL (17)	Value of the error flag when the mass storage limit has been exceeded.
F.EROD (6)	Value of error flag set for operator drop type-in. 1EJ writes a message to dayfile.
F.ERPA (14)	Error flag for a job which has been pre-aborted, i.e., before it comes to control point.
F.ERPARF (61)	Parity error occurred when swapping-in a job.

F.ERPCE (5) Value of error flag set for PP call error abort. Sensed by MTR; error message is written by 1EJ. Used when central program requests PP program with name that does not begin with a letter.

F.ERPP (3) Value of error flag set for PP abort. PP requesting abort is responsible for writing message.

F.ERRCL (15) Error flag for a bad PP call with the auto-recall bit set.

F.ERRN (10) Value of error flag in control point area set when the operator types "n.RERUN". This job will be put back into the input queue.

F.ERTL (1) Value of error flag set for CP time limit abort. Sensed by MTR; error message is written by 1EJ.

F.ESABS (0) Event stack word address is absolute.

F.ESCPA (2000) Event stack word address is relative to control point address.

F.ESOFF (0) Event stack job assigned when bit is 0.

F.ESON (4000) Event stack job assigned when bit is 1.

F.ESREL (1000) Event stack word address is relative to RA.

F.JDACT (2) Job active at control point (status flag in C.JDJST of JDT entry).

F.JDBAT (1) Job class is batch.

F.JDBNA (2) Job class is batch with non-allocatable device.

F.JDDROP (2) Job dropped – error code.

F.JDEXP (5) Job class is express.

F.JDINP (0) Job in input queue.

F.JDINT (3) Job class is INTERCOM.

F.JDKILL (1) Job killed – error code.

F.JDLMB (0) Job status – in limbo.

F.JDMUJ (4) Multi-user job class.

F.JDRRNP (4) Rerun job with priority – specified code.

F.JDRRUN (3) Job rerun – error code.

F.JDWCC (3) Job waiting for control point cleanup – status.

F.JDWCM (10) Job waiting for CM – status.

F.JDWDA (30) Job waiting for device assignment – status.

F.JDWIA (50) Job waiting for INTERCOM action – status.

F.JDWOA (40) Job waiting for operator action – status.

F.JDWPF (20) Job waiting for permanent file action – status.

F.JDWSI (1) Job waiting for swap in – status.

F.MVFG (2) Storage move flag for real time programs.

LE.APF (2) Length of APF entry.

LE.DFB (100) Length of dayfile buffer.

LE.DPT (12) Length of device pool table.

LE.FNT (3) Length of FNT entry.

LE.MAIL (6) Length of mailbox entry.

LE.TAPES (10) Length of tapes table entry.

L.CPNUM (17) Mask of ones equal to the length in bits of highest numbered control point.

L.IBUFF (20) Length of INTERCOM low speed buffer.

L.MTRRS (4) Length of CPMTR to MTR request stack.

L.PPHDR (5) Number of PP words comprising header information appended by assembler. Loading of all PP overlays begins at C.PPxWA minus L.PPHDR.

L.SCHRS (4) Length of scheduler request stack.

N.STG (2)

O.BPRU (16) Backspace n PRUs. Number of PRUs to be backspaced is given in third byte of second word of the order. O.BPRU requests repositioning defined by physical rather than logical units. No data is transmitted.

O.RCHN (17) Release chain. All record blocks assigned to a file and the RBT word pairs containing them are released. FNT is reset to an empty condition if its address is supplied in the order. Requests 16 and 17 required no communication with the device and, therefore, are given highest priority in the search for the next order to be executed. All other requests are assigned priority based on repositioning requirements.

O.RCMPR (2) Read into central memory after dropping first three CM words of first PRU. Used for loading program from system library eliminating the three word header added to system programs by EDITLIB.

O.RCTU (20) Read continuous order code.

O.RCTNU (20) Read continuous order code.

O.RDNS (3) Value of the stack processor order code for non-stop read.

O.RDP (10) Read into requesting PP's memory until a short PRU is encountered or until input area is full.

O.RDPNP (11) Read into requesting PP after dropping first three CM words of first PRU. Used for all PP system program calls.

O.RDSK (1) Read into central memory until a short PRU is encountered or until buffer is full. Set FNT to reference first PRU following first end-of-record of level X or greater. Level is given in high-order 6 bits of the order byte.

O.READ (0) Read into central memory until a short PRU is encountered or buffer is full (IN = OUT).

O.RMR (6) Read multiple records order code.

O.SKB (13)
O.SKF (12) Skip backward/forward n records of level X or greater. Level is specified in high 6 bits of the order byte; number of records to be skipped is given in third byte of second word of the order. No data is transmitted.

O.WCTU (24)
O.WCTNU (24) Write continuous order code.

O.WRP (14) Write from requesting PP, full PRUs only.

O.WRPR (15) Write from requesting PP, ending with a short PRU of level specified in high order 6 bits of order byte.

If EOF flag bit is set in this order, a zero length PRU of level 17 is written following short PRU terminating record.

O.WRT (4) Write full PRUs from central memory.

O.WRTR (5) Write from central memory, ending with a short PRU of level specified in high order 6 bits of the order byte. If EOF flag bit is found in this order, a zero length PRU of level 17 is written following short PRU terminating record.

OV.ACE	Control card reader
OV.ALJ	ECS over-commitment algorithm
OV.APR	Automatic program sequencer
OV.CEM	CIO error message processor
OV.CIO	Circular I/O processor
OV.CKP	Tape checkpoint dump
OV.CLO	File CLOSE processor
OV.CON	Connect file to terminal
OV.CTS	Common file processor
OV.DIS	Display processor (DSD)
OV.DMP	CM dump
OV.DPF	Permanent file dump utility
OV.ECI	ECS diagnostics
OV.EFL	
OV.EPF	Permanent file tables entry manager
OV.ETL	Enter time limit
OV.FNT	
OV.GEJ	
OV.IAP	
OV.LDR	Loader control card processor
OV.LOD	Loader call processor
OV.LPF	Permanent file loader
OV.MDI	Move system directory (EDITLIB)
OV.MEM	MEMORY function processor
OV.MES	Message processor (DSD)
OV.MSG	Add message to dayfile
OV.OPE	File OPEN processor
OV.PFA	Permanent file ATTACH processor
OV.PFC	Permanent file CATALOG processor
OV.PFE	Permanent file EXTEND and ALTER processor
OV.PFG	
OV.PFM	Permanent file manager (see 1PF)
OV.PFP	Permanent file PURGE processor
OV.PFR	Permanent file RENAME processor
OV.PFS	Permanent file SETP processor
OV.PRM	PERM macro processor
OV.REQ	REQUEST function processor
OV.RFL	Field length request processor
OV.RPV	REPRIEVE function processor
OV.RST	Restore control point area for restart
OV.RWE	
OV.SRB	Enter expanded disk address into directory (EDITLIB)
OV.TBL	INTERCOM – get table in low core
OV.TDS	Peripheral job delay stack processor
OV.VSN	Process volume sequence number (control card parameter)

OV.XDQ	PP counterpart of XXXDMPQ
OV.XRQ	PP counterpart of XXXRESQ
OV.1AJ	Advance job
OV.1AU	
OV.1BT	Blank label processor – tape and disk
OV.1CT	INTERCOM queue manager-Scheduler
OV.1CL	Close file – trailer label processing
OV.1DA	Process family disk pack releases
OV.1DF	Dump dayfile
OV.1DL	
OV.1DM	Device queue manager-Scheduler
OV.1DS	
OV.1EJ	End-of-job and error processor
OV.1EP	ECS stack processor
OV.1FC	
OV.1FM	JANUS – film driver
OV.1IB	Initiate batch job from input queue
OV.1ID	
OV.1IM	
OV.1IQ	JANUS – input queue processor
OV.1IR	JANUS – input register processor
OV.1IS	JANUS – initialize control point
OV.1IU	JANUS – print file backspacer
OV.1I1	
OV.1LT	Load jobs from tape
OV.1MF	Multi-file positioning
OV.1MH	Mother's helper for DSD
OV.1MT	Read/write other 7-track tapes
OV.1M1	
OV.1NR	Read 9-track tapes
OV.1NW	Write 9-track tapes
OV.1OP	File open routine
OV.1PC	Scheduler – drop permanent file mass storage (queue manager)
OV.1PF	Permanent file job queue manager (PFM)
OV.1PK	
OV.1PL	JANUS – plotter driver (dummy)
OV.1PT	
OV.1RC	Reload CM for checkpoint/restart
OV.1RN	Scheduler – age jobs in I/O queues
OV.1RP	End-of-reel processing
OV.1RS	Read stranger (S) tapes
OV.1RT	Read standard (E) SCOPE tapes
OV.1SI	Swap or roll in a job at a control point
OV.1SJ	
OV.1SO	Swap or roll out a job at a control point
OV.1SP	Stack processor
OV.1SX	Mass storage I/O error processor
OV.1S5	DST ordinal checker (PP input register); Stack processor loader

OV.1TD	Tape dump routine
OV.1TF	Move tape forward (except long record (L) tapes)
OV.1TO	Tape open; move tape backward (rewind)
OV.1TS	Scheduler – device queue manager
OV.1WI	Write standard (E) SCOPE tapes
OV.1WS	Write stranger (S) tapes
OV.2EI	
OV.2LP	On-line printer driver
OV.2PC	On-line card punch driver
OV.2RC	On-line card reader driver
OV.2RQ	
OV.2TA	
OV.2TB	Write trailer label – backward tape positioning (except long record (L) tapes)
OV.2TC	
OV.2TJ	Translate JOB card
OV.3DO	Allocatable device file open routine
OV.3PK	
OV.3TT	INTERCOM – teletype I/O processor
OV.4ES	Enter stack request for mass storage I/O
OV.4LB	Tape label processor
OV.4LC	
OV.5DA	Family disk pack processor
OV.6BR	
OV.6BW	
OV.6CR	
OV.6CW	
OV.6LC	
OV.6LM	
OV.6MD	Controls CMR directory changes
OV.6SI	
OV.6WM	Write error messages
OV.7EC	
OV.7SB	
OV.7T1	ASCII conversion table
OV.7T2	EBCDIC conversion table
OV.8AC	
OV.8M1	
OV.8M2	
OV.8PT	
OV.8SF	
OV.8SJ	
OV.8T3	Load MMTC with EBCDIC conversion table for INR
OV.9PT	
OV.9SJ	

P.AVTAPE (15) Word in CMR containing available tapes for a given configuration.

P.CHRQ (41) Pointer to channel request queue.

P.CMFL (24) Control memory field length/100.

P.CMLWA (2) CMR last word address pointer.

P.CPSTAT (54) CPU A/B status.

P.CST (5) Word in CMR containing channel status table pointers.

P.DFB (3) Address of dayfile buffer pointer word. Only the first byte (byte 0) is used; it contains CM address/10 of dayfile buffer.

P.DPT (14) Device port table pointer.

P.ECSFL (27) Machine ECS field length pointer.

P.EIRPR (11) ICC pointer for ECS.

P.ELBST (12) Pointer to ECS empty page stack.

P.EST (5) Address of EST pointer word. Byte 0 contains 12-bit first word address; byte 1 contains 12-bit last word address + 1.

P.FNT (4) Address of FNT pointer word. Byte 0 contains 12-bit first word address; byte 1 contains 12-bit last word address + 1.

P.HEC (4) CMR location of a word containing the hardware error count.

P.INS (10) Address of a pointer word to an installation area; content is unspecified.

P.INT (16) Word in CMR pointing to INTERCOM communication areas.

P.LIB (1) Address of library directory pointer word. Bytes 0 and 1 contain right justified 18-bit first word address of library directory. Bytes 2 and 3 contain right justified 18-bit last word address plus one. Byte 4 contains a deadstart load flag; it must always be zero when a disk or recovery deadstart is attempted.

P.MAIL (65) Message mailbox pointer.

P.NCP (3) CMR location of a word containing the number of control points.

P.NOVA (3) CMR location of the NOVA pointer word.

P.NPP (3) CMR location of a word containing the number of peripheral processor units.

P.PCOM (5) PP communications area pointer word.

P.PFM1 (6) See P.PFM3.

P.PFM2 (7) See P.PFM3.

P.PFM3 (17) CMR locations of three pointer words used by the permanent file manager.

P.PJT (26) Peripheral job table pointer.

P.PPLIB (42) PP library program name table pointer.

P.RBR (2) Address of RBR pointer word (also serves as RBT pointer word – see P.RBT). Bytes 0 and 1 contain right justified 18-bit first word address of RBR table area.

P.RBT (2) Address of RBT pointer word (also serves as RBR pointer word – see P.RBR). Byte 2 contains first word address/2 of RBT empty chain. Byte 3 contains current length/100B entire RBT area. Byte 4 contains (LWA+1)/100 of central memory.

P.RMS (14) Word in CMR containing the pointer and length of the Rotating Mass Storage Diagnostic Table.

P.RPT (64) Removable pack table pointer word.

P.RQS (13) Address of request stack area pointer word. Byte 0 contains stack entry word pair count. Byte 2 contains first word address/2 of actual request stack. Byte 3 contains number of allocatable devices (N.DEVICE). Byte 4 contains FWA/10 of DST entries. All DST entries appear at beginning of request stack area followed immediately by actual request stack.

P.SCH (60) Scheduler pointer word.

P.SCHPT (65) CMR statistics table pointer.

P.SEQ (4) Word in CMR containing diagnostic sequencer pointers.

P.STG (15) Tape scheduler pointer word.

P.STR (61) Scheduler/MTR communications word.

P.SWPECS (65) Swap to ECS flag word.

P.TAPES (14) Word in CMR containing the pointer and length of the Tapes Table.

P.VRNBUF (43) Pointer to VRN buffer table.

P.ZERO (0) Address of central memory word containing all zeros. Used by most PP routines as a quick means of zeroing five successive PP locations. The system is destroyed by setting contents of this location to non-zero.

Q.IUSRST	User tape identifier in INTERCOM user table.
R.DCH (627)	PP resident routine to drop a channel.
R.DFM (656)	PP resident routine to transmit a dayfile message.
R.EREQS (431)	PP resident routine to access a request stack.
R.FAF (100)	PP Resident cell which contains the field access flag.
R.IDLE (103)	PP resident idle loop.
R.MTR (544)	PP resident routine to process a request for MTR.
R.OVL (144)	PP resident routine to load a PP overlay.
R.OVLJ (135)	PP resident routine to load a PP overlay and transfer control to it.
R.PAUSE (464)	PP resident routine to test control point storage move flag and to pause until completion of the move.
R.PROCES (544)	Identical with R.MTR.
R.RAFL (464)	PP resident routine to process requests to access control point field length.
R.RCH (617)	PP resident routine to process requests to reserve a channel.
R.READP (316)	PP resident routine to read data via channel from Stack Processor.
R.RWP (342)	Special entry point to R.READP used by LDR.
R.RWPP (356)	Word in R.READP modified by LDR.
R.STB (650)	PP resident routine to perform a masking operation using specified words.
R.STBMSK (641)	Address within PP Resident of a mask used by R.STB routine.
R.TAFL (517)	PP resident routine used to terminate access of a control point field length.
R.TFL (533)	PP resident routine to insure that a relative address is within the control point field length.
R.WAIT (555)	PP resident routine to idle until output register clears.
R.WRITEP (325)	PP resident routine to write data via channel to Stack Processor.
S.ACTION (6)	Action (DS parameter byte).

S.APFIL (1)	APF table interlock (C.PFMCH)
S.CPA (2)	Right offset of rerun bit in byte C.CPFP of the control point area.
S.CPCLR (66)	Control point clear request.
S.CPDP (6)	Right offset of private pack bit in byte C.CPFP of the control point area.
S.CPE (13)	Right offset of the EXPORT/IMPORT bit in byte C.CPFP of word W.CPERT of the control point area.
S.CPEOJ (65)	End of job flag.
S.CPF (0)	Right offset of the reprocess control card bit in byte C.CPFP of word W.CPERT of the control point area.
S.CPEVNT (6)	Swapout event bit in W.CPSWP.
S.CPFFL (64)	Flags FNTs in positive FL (bit in W.CPSCH of JCA).
S.CPFLG (7)	Right offset of the CP loader flag. If set, the CP loader is being used.
S.CPG (1)	Right offset of the abort bit in byte C.CPFP of the control point area. This bit is set by 1AJ if a control card error occurs or if the user tries to load the binary output of an assembly or compilation which had errors.
S.CPJ (11)	JANUS bit for DSD.
S.CPJFL (10)	Job card field length assigned bit.
S.CPLDAF (0)	Directory access flag offset.
S.CPL1 (4)	Incomplete load flag.
S.CPLL (2)	Library load bits.
S.CPLM (6)	Map flag bits.
S.CPLP (1)	System routine protection bits.
S.CPLR (5)	Reduce flag.
S.CPLT (4)	Trap debugging and flag.
S.CPLW (12)	Loader type flag. (Which loader to be used.)
S.CPN (4)	Right offset of the checkpoint bit in byte C.CPFP of the control point area. This bit is set if a valid checkpoint has been taken.

S.CPNFNT (3) 1AJ should skip FNT search.

S.CPNO (0) Flag set by DSD on NO type-in.

S.CPR (12) Right offset of the INTERCOM bit in byte C.CPFP of the control point area. This bit is set for all INTERCOM jobs.

S.CPRFL (67) Storage request in progress flag.

S.CPROP (70) Rollout is in progress flag.

S.CPS (30) Right offset of the sequencer bit in byte C.CPFP of the control point area. Bit is set to indicate input file is not to be purged.

S.CPSIP (71) Swap in is in progress.

S.CPSOP (72) Swap out is in progress.

S.CPSWC (73) Swap out is complete.

S.CPUSTA (2) Run in CPU-A only status.

S.CPUSTB (3) Run in CPU-B only status.

S.CPUSTC (7) Currently assigned to CPU-A status.

S.CPUSTD (10) Currently assigned to CPU-B status.

S.CPUSTM (0) Suspended for storage move status.

S.CPUSTP (12) Suspended for checkpoint status.

S.CPUSTR (6) Suspended for real time status.

S.CPUSTS (11) Suspended for swapout status.

S.CPUSTW (5) Waiting for CPU (unless suspended) status.

S.CPUSTX (4) Recall status.

S.CPUSTY (1) Automatic recall status.

S.CPX (5) Right offset of the exit bit in byte C.CPFP of the control point area. This bit is set whenever 1AJ encounters an exit card.

S.CPYES (1) Set by DSD when YES type-in received.

S.CP1IB (63)	1IB operating at control point.
S.DIRPR (10)	Number of bit positions to right shift in PP word to right justify program residence to bit zero.
S.DIRPT (4)	Number of bit positions to right shift in a PP word to right justify the program type to bit zero.
S.ECIOT (11)	ECS buffer type in C.ECFLG.
S.ECSB (13)	ECS buffer active flag in C.ECFLG.
S.EDTRUN (1)	Set in C.DSFLAG every time EDITLIB runs.
S.ESTBSY (4)	Private device busy bit in EST.
S.ESTFR (5)	Private device free bit in EST.
S.ESTPF (10)	Public device/permanent file device bit in EST.
S.ESTPFD (7)	Public device/permanent file directory device bit in EST.
S.ESTRMS (13)	RMS device bit in EST.
S.ESTSYS (6)	Public device/system device bit in EST.
S.EVJST (57)	Job status modified flag in word 4 of PP message buffer.
S.FECS (5)	Uppermost of 2 ESS flags in FST.
S.FEXNEW (6)	E/N bit in FNT.
S.FINDXW (6)	Index write bit in FNT.
S.FLINK (4)	Link bit in FNT first word.
S.FLOCK (5)	Lock bit in FNT first word.
S.FMUJ (3)	Multi-user job bit in FNT.
S.FNTEQP (6)	Right offset of equipment code field in FNT. Equipment code field is positioned in bits 6-11 of byte zero.
S.FNTWRT (7)	Write bit in FNT.
S.FPFREW (2)	Permanent file rewrite bit in FNT.
S.IEDIT (13)	INTERCOM/EDITLIB flag.

S.ILKOUT (13) INTERCOM lockout flag.

S.ILOGO (5) INTERCOM login/logout flag.

S.I.MUJ (13) INTERCOM user attached to MUJ.

S.INTC (2) INTERCOM LCC up/drop flag.

S.INTH (1) INTERCOM high speed up/drop flag.

S.INTL (0) INTERCOM low speed up/drop flag.

S.ISTATE (7) INTERCOM user status.

S.IUCCP (6) INTERCOM control cards processed flag (C.IUCCA).

S.IUDMP (7) INTERCOM dump flag (C.IUFILE).

S.IUECS (11) INTERCOM swap file on ECS flag (C.IUFILE).

S.IUEOE (6) INTERCOM end of job flag (C.IUFILE).

S.IUFNT (12) INTERCOM FNTs to be gathered flag (C.IUFILE).

S.IUPS (10) INTERCOM pause flag (C.IUFILE).

S.IURED (13) INTERCOM value for S.CPLR in CP area (C.IUFILE).

S.IIII (12) INTERCOM initialization flag.

S.JDBCB (43) Job recovery occurred (C.JDFLG).

S.JDECS (36) Swap file on ECS (C.JDFLG).

S.JDEXP (22) Job is for express queue (C.JDOPF).

S.JDGO (21) Operator typed GO (C.JDOPF).

S.JDGR (3) Job origin (Graphics bit).

S.JDINT (5) Job origin (INTERCOM bit).

S.JDLGI (41) Job LOGIN command (C.JDFLG).

S.JDLGO (40) No swap file needed (C.JDFLG).

S.JDLOK (17) Job not to be brought to control point (C.JDOPF).

S.JDMUJ (4) Job origin (multi-user job bit).

S.JDNFNT (33) FNT search inhibited (C.JDFLG).

S.JDNJ (37) New job in INTERCOM area (C.JDFLG).

S.JDNRR (42) Job cannot be rerun (C.JDFLG).

S.JDNS (20) Job cannot be swapped or rolled (C.JDOPF).

S.JDROLL (34) Job cannot be swapped out (C.JDFLG).

S.JDRT (2) Real time job (C.JDORG).

S.JDSKFL (35) Skip fl on swap file (C.JDFLG).

S.LBLLVL (12) RMS label level (DS parameter byte).

S.NXJSR (42) Storage request flag.

S.PFDEU (3) PFD entry in use (C.PDFLAG).

S.PFDIL (2) Permanent file directory interlock (C.PFMCH).

S.PFLVL (11) Permanent file level (DS parameter byte).

S.PFUTIL (4) Utility interlock (C.PFMCH).

S.RBRUNT (6) Right offset of DST ordinal field in RBR header.

S.RBTCIL (3) RBTC I/L offset in byte C.PFMCH in word P.PFM1.

S.RBTCW (0) RBTC wraparound offset in byte C.PFMCH in word P.PFM1.

S.RBTDEV (13) Right offset of the bit that indicates whether the file was assigned by device type in byte 2 of the first word of the first RBT word pair in a chain.

S.RBTEST (12) Right offset of the bit that indicates whether the file was assigned by EST ordinal in byte 2 of the first word of the first RBT word pair in a chain.

S.RBTNEW (10) Right offset of new RBT flag bit in flag field of RBT word pair.

S.RBTOVF (7) Right offset of the overflow bit in byte 2 of the first word of the first RBT word pair in a chain.

S.RBTPFD (11) Right offset of the permanent file flag bit in byte 2 of the first word of the first RBT word pair in a chain.

S.RBTRBR (3) Right offset of RBR ordinal field in RBT word pair.

S.RBTREL (7) Right offset of release bit in flag field of RBT word pair.

S.RBTRND (6) File organization is random (C.RBTBIT).

S.RBTSAM (5) File organization is index-sequential or direct access (C.RBTBIT).

S.STECS (0) ECS flag bit in stack request.

S.STF (6) Right offset of flag field in a stack request.

S.STFA (0) Right offset (in the flag field) of PP-available bit in a stack request.

S.STFEOF (4) Right offset (in flag field) of end-of-file bit in stack request.

S.STFETP (2) Right offset (in flag field) of FET-present bit in stack request.

S.STFNTP (3) Right offset (in flag field) of FNT-present bit in stack request.

S.STFPRI (5) Right offset (in flag field) which specifies high priority for this stack request.

S.STFREL (4) Right offset (in flag field) of release bit in stack request.

S.STFXCT (1) Right offset (within the flag files) of the exaxt bit in byte C.STFB of word W.STFB of a request stack entry. Offset is relative to S.STF.

S.SYSEDT (0) Bypass MDI GO/DROP message (C.DSFLAG).

S.SYSLVL (7) System level flag (DS param byte).

S.TRANIL (6) TRANSPF interlock.

T.CLK (30) Location in which clock is kept, in form HH.MM.SS. T.CLK is updated by MTR and displayed on top line of left scope. Time will be time of day, if a time entry to DSD was made; otherwise, it will be the time since deadstart.

T.CPA1 (200) Control point area address of control point 1.

T.CPJOBN (26) CMR location of a pointer word containing the job sequence number and job count.

T.CPSTA (44) CPU-A control word.

T.CPSTB (45) CPU-B control word.

T.CPT1 (56) Location of the CPU activity status. CPU A status is in byte 4, CPU B status is in byte 3. The activity status is the control point area address of the program currently using the CPU (0000 for idle).

T.DATE (31) Location of today's date in the form entered by operator at deadstart time. Date is displayed on top line of left scope.

T.ECSPAR (57) This word contains in bytes

0,1	Zero.
2	An ECS flaw table full flag. It is set when a flaw count overflow is detected during as ECS transfer (by ICEBOX).
3	An ECS parity flag which can assume values of 2 or 4. It is set when a parity error is detected during an ECS storage move, and indicates whether 1 or 2 control points shall have their error flags set to F.ERECP. The control point requesting the ECS will always have its error flag set. In the case of storage overlap and the occurrence of the parity error in the part of ECS which belongs to another control point, the flag will have the value 4. After the flag is set, DSD will display a message stating that an ECS parity error has occurred and MTR will not assign ECS anymore. These conditions shall prevail until the next deadstart.
4	This byte contains the address of the block in ECS/1000 in which the parity error occurred. DSD will display this value as above.

T.EPBL (27) ECS page and buffer length.

T.JDATE (20) Location of today's date in Julian format. Value is computed from date entered by operator at deadstart time.

T.MSC (40) Contains the time to the millisecond since deadstart.

Byte 1	Time in seconds (reset each 2**12 seconds)
Byte 2	Milliseconds since updating the seconds
Byte 3-4	Time in msec (reset each 2**24 msec).

T.MSP (37) Location of Monitor step flag used for communication between DSD and MTR while system is in step mode.

T.MXNCTL (46) MXN control word.

T.PPID (47) PP identification for MXN to CPMTR.

T.PPIP (50) PPMTR function pointer.

T.PPS1 (154) Location of the PP status word. Byte 0 contains the control point area address of the control point to which the PP is assigned. T.PPS0 (52) is used by MTR as a scratch memory word.

T.PPSO (52) Pointer to control point area to which PP is assinged.

T.RCHN (55) Location of the first RBT word pair to be released.

T.SCHCP (62) Scheduler interlock word.

T.SCHPP (63) PP interlock word for scheduler use.

T.SLABx (31-36)
(x = 1-6) Locations containing system label displayed on top line of left scope.

T.UAS (56) Location of unassigned storage length. MTR keeps tally of size of the holes between control points in T.UAS; 1RA uses this size to determine whether or not there is adequate central memory to bring a given job to a control point.

Byte 0	Central Memory UAS
Byte 1	ECS UAS

W.CHTIM (65) Relative word in control point area giving channel I/O time.

W.CKP (52) Relative word in control point area contain checkpoint parameters for a job.

W.CPAR (60) Relative word in a control point area which contains the auto-recall register.

W.CPCAF (70) Relative first word address in a control point area of the 100B word buffer containing current control cards.

W.CPCAL (167) Relative LWA in a control point area of a 100-word buffer containing current control cards.

W.CPCC (26) Relative word in control point area containing reprieve function address.

W.CPCKP (52) Checkpoint parameter word. Same as W.CKP.

W.CPDFM (30) First of eight words in a control point area containing dayfile message currently on the B display.

W.CPDFMC (62) Word in the control point area containing the dayfile message count.

W.CPDPV (62) Relative word in a control point area containing the job dependency count and ID.

W.CPECS (27) Relative word in a control point area containing ECS field length/1000 and ECS RA/1000 assigned to a job.

W.CPEF (24) Relative word in a control point area containing error flags for job.

W.CPERT (63) Relative word in a control point area containing internal flags used by job processing routines.

W.CPFACT (50) Control point word containing permanent file accounting parameter. (Same as W.ID).

W.CPFL (24) Relative word in a control point area containing memory field length/100 assigned to the job.

W.CPFLAG (63) Relative word in a control point area containing various flags.

W.CPFP (63) Job processing flags. See W.CPFLAG.

W.CPFST (51) FNT positioning information for job input file. (Same as W.FSTCC).

W.CPINS (170) Relative first word address in a control point area of an 8-word zone reserved for installation use.

W.CPINT (41) Relative word in a control point area containing INTERCOM status and communication information.

W.CPIRB (62) Control point area word containing INTERCOM user ID.

W.CPJCP (40) Word in the control point area containing job card parameters.

W.CPJNAM (25) Relative word in control point area containing seven-character job name.

W.CPLDR1 (55) Same as W.CPLDR.

W.CPLDR (55) Relative word in control point area containing various loader flags.

W.CPLDR2 (56) Library set information (global set) for loader use.

W.CPLDR3 (57) Loader global set information word 2.

W.CPLINK (20) Active control point link word.

W.CPMSI (66) Relative word in a control point area containing the INTERCOM PP recall register.

W.CPMSLM (64) Relative word in a control point area containing mass storage limit information.

W.CPOAE (53) Relative word in control point area containing byte used to communicate operator-assigned equipment. MTR sets EST ordinal of equipment requested by DSD in this byte for subsequent testing by REQ.

W.CPOUT (63) Relative word in control point area which contains an output file flag in byte C.CPOUT.

W.CPPRI (40) Relative word in control point area containing current job priority.

W.CPPTM (23) PP time word in control point area. See C.PPTIME.

W.CPRO (42) Relative word in a control point area containing the roll-out flags.

W.CPSCH (42) Scheduler information for current job at control point.

W.CPSLIC (20) Maximum CPU time slice allowed for job.

W.CPSR (67) Relative word in a control point area containing outstanding stack requests.

W.CPSSW (43) Word contains sense switch bits.

W.CPSTAT (24) Relative word in control point area containing status byte.

W.CPSTG (61) Tape staging parameters are in this word. (Same as W.CPTAPE).

W.CPSWP (41) Job swapping control word.

W.CPTAPE (61) Relative word in a control point area containing the available tape status.

W.CPTIMB (22) Relative word in control point area containing CPU-B time accumulated by job.

W.CPTIME (21) Relative word in control point area containing CPU-A time accumulated by job.

W.CPTIML (40) Relative word in control point area containing CP time limit imposed on the job.

W.CPUST (20) CPU status word in control point area.

W.CPVRNO (54) Relative word in control point area which transmits private pack visual reel number typed by operator to tape labelling routine.

W.FAPF (0) FNT word containing APF pointer for file.

W.FCS (2) Code and status word offset in FNT.

W.FEQP (1) Equipment code word offset in FNT.

W.FSTCC (51) Relative word in control point area containing FST entry (FNT word 2) for job input file. Contents of this word designate position on device at which next PRU of control cards may be found. (Same as W.CPFST).

W.ICOM (0) INTERCOM/SCOPE communications word.

W.ID (50) Relative word in control point area containing the owner ID. (Same as W.CPFACT).

W.IENDLS (1) INTERCOM low speed word.

W.IHUSR (44) INTERCOM high speed user table pointer.

W.IINS (17) INTERCOM installation word.

W.IMNAME (1) Multi-user job name word for INTERCOM.

W.IUACP (15) CP time field for INTERCOM user.

W.IUAPP (14) PP time field (INTERCOM).

W.IUEQP (10) INTERCOM user equipment word.

W.IUFST (3) INTERCOM communication word with swapper.

W.IUMUJ (2) INTERCOM MUJ word.

W.IUSER (27) INTERCOM user ID word.

W.IUSTA (0) INTERCOM user table header word.

W.JDDSD (2) Third word of job descriptor entry. Communication with DSD.

W.JDINT (4) Fifth word of job descriptor entry when IP.INTCM is not zero.

W.JDLNK (0) First word of job descriptor entry (also W.IDNAM). Link to next JDT entry.

W.JDMGR (3) Fourth word of job descriptor entry. Manager/scheduler communications.

W.JDNAM (0) First word of job descriptor entry (also W.JDLNK). Job name.

W.JDSWP (1) Second word of job descriptor entry. Swapper word.

W.LBCK (7) Relative location in the device label of the word containing the label checksum.

W.LBDATE (0) Relative location in the device label of the word containing the date.

W.LBFLAW (10) Relative location in the device label of word containing first word of RBR header.

W.LBID (0) Relative location in the device label of the word containing the label ID.

W.LBPFD (2) Relative location in the device label of the word containing the PFD pointer.

W.LBPRIV (3) Relative location in the device label of the word containing private pack information.

W.LBRBR (4) Relative location in the device label of the word containing RBR ordinal.

W.LBRBTC (2) Relative location in the device label of the word containing the RBTC pointer.

W.LBVID (1) Relative location in the device label of the word containing the visual ID.

W.PPIR (0)	Relative word in a PP communication area containing PP input register.
W.PPMESx (2-7) (x = 1-6)	Relative word in PP communication area containing six-word PP message buffer.
W.PPOR (1)	Relative word in PP communication area containing PP output register.
W.PPTIME (23)	Relative word in control point area containing total PP time accumulated by job.
W.RBRLAV (1)	Relative word in RBR header word containing count of record blocks logically available.
W.RBRTPA (0)	Relative word in RBR header containing equipment type and allocation style.
W.RBRUNT (0)	Relative word in RBR header containing unit number (DST ordinal).
W.RWPPCW (2)	Relative position of READP/WRITEP communication word in message buffer of PP communication area.
W.SCHIR (0)	Scheduler input register displacement.
W.SCHOR (1)	Scheduler output register displacement.
W.SCHRS (2)	Scheduler request stack displacement.
W.SSW (43)	Relative word in control point area containing sense switch settings for job. (Same as W.CPSSW).
W.STCPU (0)	Relative word in stack request containing control point and unit number (DST ordinal) of request.
W.STEI (0)	Relative word in stack request containing empty indicator. If this field is 0, the entry is not in use.
W.STFB (1)	Word in stack request which contains flag byte.
W.STGFRE (1)	Indicates units not assigned and on.
W.STGMAX (0)	Maximum units defined in CMR.
W.STGSAT (3)	Units held by satisfied jobs.
W.STGUFD (2)	Indicates unfilled demands for units.
W.STO (0)	Word in stack request which contains stack processor order.
W.STPFW (1)	Word in stack request which contains next address in PP memory for data transmission. Field is used in this manner only on calls to R.READP or R.WRITEP.

W.STPLW (1) Word in stack request which contains LWA+1 in PP memory for data transmission. Field is used in this manner only on calls to R.READP or R.WRITEP.

W.STPMS (1) Word in stack request which contains PP message buffer address.

W.STPPRU (0) Word in stack request which contains PRU number at which to begin data transmission if no FNT is specified.

W.STPRBA (0) Word in stack request which contains address of RBT word pair containing record block at which to begin data transmission if no FNT is specified.

W.STPRBN (0) Word in stack request which contains RBT ordinal of record block at which to begin data transmission if no FNT is specified.

W.STPWC (1) Word in stack request which contains number of PP words (bytes) to be transmitted during a READP or WRITEP request.

W.TFLGS (5) Relative word in a Tapes Table entry containing flags used by tape routines.

W.TFLN1 (1) Relative word in a Tapes Table entry containing the first 10 characters of the label name.

W.TREEL (4) Relative word in a Tapes Table entry containing the reel number.

W.TVRN (6) Relative word in a Tapes Table entry containing the visual reel number of the current reel.

W.TVRN1 (7) Relative word in a Tapes Table entry containing the visual reel number of the first reel.

W.XPIR (24) Exchange package input register.

W.XPOR (25) Exchange package output register.

W.XPRS (26) Request stack for scheduler exchange packages.

X.ECOVL (4) Executive processor ECS code.

X.ICE (2) Executive processor ICEBOX code.

X.SCH (5) Central executive code for scheduler used when storage request is rejected.

X.SCH2 (6) See X.SCH.

X.SPM (3) Executive SPM code.

PPSYS MACROS

LDK MACRO

Generates LDN, LDC, or LCN instruction depending on size of its argument, which may be any valid address expression. Recommended for referencing SCPTEXT symbols for CM pointer words.

ADK MACRO

Generates ADN, ADC, SBN, or no instruction depending on size of its argument, which may be an address expression. Recommended for referencing SCPTEXT symbols for relative positions in control point tables (W.x symbols).

SBK MACRO

Generates SBN or ADC instruction, depending on size of its argument. Any symbols used in its argument must be defined.

UJK MACRO

Generates UJN or LJM instruction, depending on length of jump. In general, the jump must be backward, since symbols used in address field must have been previously defined. Macro is useful for exiting from small subroutines subject to expansion.

BIT MACRO

Generates no code; merely defines the symbol in the location field. Value assigned to symbol is a 1-bit mask where the bit is positioned according to the value of address field. Bits are counted in the standard manner, from right to left, beginning with zero. Thus, the statement MASK BIT 2 would set MASK equal to 4. Macro is useful for generating 1-bit flag values with the S.x SCPTEXT symbols.

ENM MACRO

Generates standard PP subroutine entry and exit lines. The subroutine name is that declared in location field of ENM; subroutine may be entered by an RJM to that name. If address field of ENM is blank, no exit symbol is defined; otherwise, address field contents are appended to location symbol to generate subroutine exit symbol. (Typically, address field contains only an X) An exit from subroutine may be made by jumping directly to the generated symbol.

PPENTRY MACRO

Used as first instruction following ORG in a primary level overlay. PPENTRY generates code to set up low core parameter as follows:

D.PPIRB through D.PPIRB+4	Input register contents
D.RA	Reference address/100_8
D.FL	Field length/100_8

The macro also sets the field access flag in the control point area. Address field of the PPENTRY macro must always contain: D.PPIRB, D.TO or an error will result.

LDCA MACRO

The macro loads PP A register with absolute 18-bit central address. Relative CM address is obtained from two consecutive PP low core locations; the first is specified in address field of LDCA macro. CM address is assumed to be right justified within these two words. Contents of D.RA are added to CM address. Macro is useful for loading many different CM addresses. Space may be conserved by using a subroutine rather than a macro if the same address is to be loaded three or more times.

CRI MACRO

Reads contents of a CM word whose address is contained in a central memory pointer. Address field of CRI macro contains X, Y, and Z subfields, in that order.

X	6-bit CM pointer word address
Y	First of five PP locations into which the desired CM word will be read.
Z	Byte within CM pointer word containing 12-bit CM address of desired word, counting from zero to 4.

CEQU MACRO

If symbol in label field has not been previously defined, sets the symbol equal to the value in operand field; otherwise the macro is a no-operation.

CMICRO MACRO

Acts as a MICRO instruction to define the symbol in label field if label has not been previously defined; otherwise the macro is a no-operator.

JOBCARD MACRO

SCPTEXT contains a definition of a macro called JOBCARD. The release version is empty, consisting of a macro definition header and a terminator. System characteristics may be altered by an installation by inserting (between header and terminator) one or more cards as described below.

If the symbol SCOPE 2.0 is defined in JOBCARD, SCOPE 3.4 is altered to accept only SCOPE 2.0 job cards. The value to which the symbol SCOPE 2.0 is equated is irrelevant

SCOPE 3.4 may be altered to accept a decimal value on one or more of the job card parameters by inserting a card in the following form:

DECIMAL *field*

where *field* is one of the terms EC, CM, T, P, D, TP, MT, or NT. Currently, all values are assumed to be octal; however, it may be declared specifically that a parameter is to be interpreted as octal by inserting a card in the following form:

OCTAL *field*

INDEX

COMMENT SHEET

CONTROL DATA

TITLE: SCOPE System Programmer's Reference Manual Version 3.4

PUBLICATION NO. 60306500 REVISION A

Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

CUT ON THIS LINE

General comments:

FROM NAME: _____ POSITION: _____

BUSINESS ADDRESS: _____

_____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

:OLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Software Documentation
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ON THIS LINE

:OLD

FOLD

STAPLE

STAPLE

► ►CUT OUT FOR USE AS LOOSE-LEAF BINDER TITLE TAB

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

Pub. No. 60305600

Litho in U.S.A.