



**NOS/BE VERSION 1
SYSTEM PROGRAMMER'S
REFERENCE MANUAL**

Volume 1 of 2

CDC® COMPUTER SYSTEMS:

CYBER 180

CYBER 170

CYBER 70

MODELS 71, 72, 73, 74

6000

PAGE INDEX TO MACROS

<u>Macro</u>	<u>Page</u>	<u>Macro</u>	<u>Page</u>
ACCSF	7-14	READLS	5-19, 25
ACQUIRE	4-7	READNS	5-24
BKSP	5-17	READSKP	5-9, 24
CALL	2-16	RECOVR	7-18
CALLSS	2-24	REPRIEVE	7-18
CESAB	7-28	RETURN	2-16
CEVAL	7-20	ROUTE	4-3
DAYFILE	4-3	SEGDEF	2-16
DISPOSE	4-3	SETLC	7-32
ENCSF	7-15	SETLOF	7-18
ENDSEG	2-16	SETMFL	2-39
GETLC	7-32	SFCALL	2-28
GETLOF	7-18	SKIPB	5-17, 26
GOTO	2-16; 10-3	SKIPF	5-16, 25
GOTOTAB	2-16	SSCT	2-39
READ	5-6, 24	SYSTEM	6-15; 7-3
READC	5-18, 25	WRITE	5-11, 25
		WRITEC	5-20, 25
		WRITEF	5-13, 25
		WRITER	5-12, 25



**NOS/BE VERSION 1
SYSTEM PROGRAMMER'S
REFERENCE MANUAL**

Volume 1 of 2

CDC® COMPUTER SYSTEMS:

CYBER 180

CYBER 170

CYBER 70

MODELS 71, 72, 73, 74

6000

REVISION RECORD

REVISION	DESCRIPTION
A (11-01-75)	Manual released.
B (07-16-76)	Revised to reflect features and PSRs added with the release of NOS/BE Version 1.1. Features included are Factory Format Support (83), Stack Processor Enhancements (133), INTERCOM Restart, 844-41 Support (145), 844 Expander Support (169), Job Management and System Control Point Enhancements (159, 163), 2X PPU Support, and FORMAT/FDP Version 2 Enhancements.
C (04-22-77)	Manual revised to support NOS/BE Version 1.2 at PSR level 447 and to make editorial and technical corrections. The following features are documented: CYBER Control Language (CCL), Fast Dynamic Loader (FDL), Reliability Feature Utilization (System Idle), CE Validation, Product Set Support, INTERCOM Enhancements, Advanced Tape Subsystem (ATS), 580 Line Printer Programmable Format Control (PFC), 844 Full Tracking, and support of the CYBER 170 Model 176 and 819 disk drive. References to 604 and 607 tape units, and the 501 line printer are removed.
D (08-19-77)	Updated to support NOS/BE Version 1.2 at PSR level 454 and to make editorial and technical corrections. Support of CYBER 170 Model 171 is included.
E (06-13-78)	Manual revised to support NOS/BE Version 1.3 at PSR level 473 and to make editorial and technical corrections. New features documented are permanent file utilities PFLOG GENLDPF; user capability to assign universal password and permissions to private sets; user relieve processing; support of all 677/679 tape units; option to schedule tapes by density; support of INTERCOM 5; SF.RERN function; SETMFL, GETLOF, SETLOF, GETLC, and SETLC macros. This edition obsoletes all previous editions.
F (10-20-78)	Manual revised to support NOS/BE Version 1.3 at PSR level 481. Information is added in the INTERCOM 5 Pointer and Buffer Area table to support the capability to turn communication lines on and off. Miscellaneous technical corrections and clarifications are made.
G (02-19-79)	Manual revised to support NOS/BE Version 1.3 at PSR summary level 488. New features documented are FNT space threshold; direct access user ECS swapping; sequencer jobs; and VSN parameter on GETPF, SAVEPF, and PURGE. Miscellaneous technical corrections and clarifications are included.
H (07-02-79)	Manual revised to support NOS/BE Version 1.3, at PSR summary level 499. New features documented are EXPORT High Speed (PP routines and EXPORT Multiplexer Subtable) and Improved Load Leveling (STF routine). Miscellaneous technical corrections and clarifications are included.
J (12-21-79)	Manual revised to support NOS/BE Version 1.4, at PSR summary level 508. New features documented are Common Test and Initialization (CTI), deadstart dump analyzer, 885 disk drive support, enhancements to the CEVAL macro, and reorganization of monitor functions. Information duplicated in the NOS/BE Reference Manual on relieve processing and permanent file utilities is removed. Miscellaneous technical corrections and clarifications are included. This edition obsoletes all previous editions.
K (05-19-80)	Manual revised to support NOS/BE Version 1.4 at PSR summary level 518. New features documented are down-line load utility, remote batch accounting for INTERCOM 5, and user capability to log error information for ECS errors. Information duplicated in the NOS/BE Reference Manual on permanent file macros is removed. Miscellaneous technical corrections and clarifications are included.
L (11-17-80)	Manual revised to support NOS/BE Version 1.4 at PSR summary level 530. New features documented include fixed assignment of user/terminal IDs and a cumulative status for 67x tape drives. In addition, the INTERCOM 5 tables have been moved from central memory resident to the managed buffer area. Miscellaneous technical corrections and clarifications have also been made.
Publication No. 60494100	

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

Address comments concerning this manual to:

Control Data Corporation
 Publications and Graphics Division
 4201 North Lexington Avenue
 St. Paul, Minnesota 55126-6198

©1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1984, 1985, 1986
 by Control Data Corporation
 All rights reserved
 Printed in the United States of America

or use Comment Sheet in the back of this manual.

REVISION RECORD

REVISION	DESCRIPTION
M (04-20-81)	Manual revised. Manual divided into two volumes as of this revision. This revision supports NOS/BE Version 1.5 at PSR level 538. New features documented include support for CTI binary release and improved SECDED error logging. All material concerning INTERCOM 4 is removed to coincide with withdrawal of support for INTERCOM 4. This edition obsoletes all previous editions.
N (05-17-82)	Updated to reflect NOS/BE Version 1.5 at PSR level 564 and to make editorial and technical changes. This revision includes documentation of the Remote Host Facility (RHF) and support for the CYBER 170 Computer Systems Models 825, 835, and 855. This edition obsoletes all previous editions.
P (11-22-82)	Manual updated to reflect NOS/BE Version 1.5 at PSR level 577 and to make editorial and technical changes. This revision includes support for the CYBER 170 Computer Systems Models 865 and 875.
R (02-20-84)	Manual updated to support NOS/BE Version 6.5 at PSR level 604 and to make editorial and technical corrections.
T (02-22-85)	Manual updated to support NOS/BE Version 1.5 at PSR level 627 and to make editorial and technical corrections.
U (02-21-86)	Manual updated to support NOS/BE Version 1.5 at PSR level 650 and to make editorial and technical corrections.

Publication No.
60494100

LIST OF EFFECTIVE PAGES

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.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	2-36	N	4-18	J	5-37	M	7-22	N
Inside Front Cover	T	2-37	N	4-19 through		5-38	M	7-23	N
Title Page	-	2-38	N	4-21	T	5-39	M	7-24	N
ii	U	2-39	P	4-22	N	5-40	M	7-25	L
ii-a/ii-b	U	2-40	M	4-23	N	5-41	N	7-26	N
iii	U	3-1	T	4-24	J	5-42	T	7-27	N
iv	U	3-2	N	4-25	N	5-43	N	7-28	M
v	U	3-3	T	4-26	T	5-44	M	7-29	U
vi	U	3-4	T	4-27	T	5-45	N	7-30	T
vii	U	3-5	T	4-28	T	5-46	M	7-31	U
viii	U	3-6	T	4-29	J	5-47	M	7-32	M
ix	T	3-7	J	4-30	J	5-48	M	7-33	M
x	U	3-8	T	4-31	T	5-49	M	8-1	M
xi	T	3-9	T	4-32	T	5-50	M	8-2	K
xii	U	3-10	U	4-33	T	5-51	M	8-3	N
xiii	T	3-11	T	4-34	J	5-52	M	8-4	J
1-1	T	3-12	U	4-35	J	5-53	M	8-5	U
1-2	T	3-13	U	4-36	J	5-54	P	8-6	N
1-3	T	3-14	T	4-37	J	5-55	P	8-7	T
1-4	T	3-15	T	4-38	J	5-56	M	8-8	J
1-5	U	3-16	U	4-39	J	5-57	M	8-9	J
1-6	M	3-17	T	4-40	J	5-58	M	8-10	T
2-1	T	3-18	T	4-41	P	6-1	M	8-11	T
2-2	K	3-19	T	5-1	T	6-2	K	8-12	T
2-3	J	3-20	M	5-2	T	6-3	K	8-13	T
2-4	T	3-21	T	5-3	T	6-4	U	8-14	T
2-5	T	3-22	U	5-4	U	6-4.1/6-4.2	U	8-15	T
2-6	K	3-23	T	5-5	J	6-5	T	8-16	T
2-7	N	3-24	T	5-6	J	6-6	K	8-17	T
2-8	T	3-25	U	5-7	J	6-7	K	8-18	T
2-9	U	3-26	M	5-8	J	6-8	K	8-19	T
2-10	T	3-27	T	5-9	J	6-9	M	8-20	M
2-11	T	3-28	T	5-10	J	6-10	N	8-21	M
2-12	U	3-29	T	5-11	J	6-11	U	8-22	M
2-12.1/2-12.2	T	3-30	U	5-12	J	6-12	K	8-23	M
2-13	R	3-31	U	5-13	J	6-13	K	8-24	M
2-14	R	3-32	U	5-14	J	6-14	K	8-25	M
2-15	N	3-33	T	5-15	J	6-15	T	8-26	N
2-16	N	3-34	T	5-16	J	7-1	N	8-27	N
2-17	J	3-35	N	5-17	J	7-2	L	8-28	N
2-18	N	3-36	U	5-18	T	7-3	R	8-29	N
2-19	N	4-1	T	5-19	T	7-4	T	8-30	N
2-20	N	4-2	T	5-20	T	7-5	J	8-31	N
2-21	N	4-3	T	5-21	T	7-6	T	8-32	N
2-22	N	4-4	U	5-22	N	7-7	T	8-33	M
2-23	U	4-5	N	5-23	T	7-8	U	8-34	N
2-24	T	4-6	N	5-24	J	7-9	T	8-35	M
2-25	M	4-7	U	5-25	J	7-10	T	8-36	M
2-26	M	4-8	J	5-26	J	7-11	J	9-1	J
2-27	M	4-9	U	5-27	J	7-12	J	9-2	M
2-28	M	4-10	U	5-28	L	7-13	T	9-3	J
2-29	M	4-11	U	5-29	M	7-14	U	9-4	J
2-30	P	4-12	U	5-30	M	7-15	J	10-1	N
2-31	U	4-12.1/4-12.2	U	5-31	M	7-16	K	10-2	J
2-32	N	4-13	J	5-32	N	7-17	J	10-3	N
2-33	M	4-14	U	5-33	U	7-18	J	10-4	N
2-34	M	4-15	U	5-34	N	7-19	J	10-5	N
2-35	N	4-16	J	5-35	N	7-20	T	10-6	J
		4-17	J	5-36	M	7-21	N	11-1	U

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
11-2	U								
11-3	U								
11-4	U								
11-5	U								
11-6	U								
11-7	U								
11-8	R								
11-9	N								
11-10	U								
11-11	J								
12-1	U								
12-2	U								
12-3	T								
12-4	T								
12-5	T								
12-6	K								
12-7	K								
12-8	U								
12-9	U								
13-1	U								
13-2	T								
Index-1	T								
Index-2	U								
Index-3	U								
Index-4	U								
Index-5	T								
Index-6	T								
Index-7	U								
Index-8	U								
Index-9	T								
Index-10	T								
Index-11	T								
Index-12	T								
Index-13	T								
Index-14	T								
Comment	U								
Sheet/Mailer	U								
Back Cover	-								

PREFACE

This manual describes the Control Data[®] Network Operating System/ Batch Environment (NOS/BE) Version 1.5. NOS/BE can operate on the following computer systems:

CDC[®] CYBER 180 Computer Systems
Models 810, 830, 835, 840, 845, 850, 855, 860, and 990

CDC CYBER 170 Computer Systems
Models 171, 172, 173, 174, 175, 176, 720, 730, 740, 750, 760, 815, 825, 835, 845, 855, 865, and 875

CDC CYBER 70 Computer Systems
Models 71, 72, 73, and 74

CDC 6000 Computer Systems

ORGANIZATION

This manual is the first volume of the NOS/BE System Programmer's Reference Manual which comprises two volumes. It is written for systems programmers who perform system evaluation or program modification.

Volume 1 describes the system interface with the central processor and peripheral processors, input/output, job processing, permanent file manipulation, and various system utilities.

Volume 2 contains system tables and file formats divided into the following general areas: character sets, central memory, job control point, disk and files, extended core storage, and symbol definitions. In general, the central memory tables, extended core storage tables, disk tables, and file formats are of interest only to system programmers. The character set tables and job control point tables are of interest to all users of the product set. Job control point tables can be used by central processor programs running at any control point. The tables in the appendixes serve as reference material for those familiar with the system and its product set. More detailed information is available in the various reference manuals and internal maintenance specifications.

CONVENTIONS

CENTRAL MEMORY WORD FORMATS

Conventions for central memory word formats are as follows:

- Crosshatching indicates a field is not used by or is not applicable to a function processor. However, Control Data reserves the right to assign these fields to system use in the future.
- Fields reserved for system use are so labeled.
- Fields with numeric labels indicate the actual value that is used or returned for a particular function. Numeric identifiers are octal unless otherwise noted.

CYBER 70 COMPUTER SYSTEMS

The term CYBER 70 Computer Systems refers to models 71, 72, 73, and 74 only.

170-CLASS MAINFRAMES

The term 170-class mainframes refers to CYBER 170 models 171, 172, 173, 174, 175, 176, 720, 730, 740, 750, 760, 865, and 875.

180-CLASS MAINFRAMES

The term 180-class mainframes refers to CYBER 180 models as well as CYBER 170 models 815, 825, 835, 845, and 855, which have most of the functional and architectural attributes of the CYBER 180 mainframes.

EXTENDED MEMORY

Extended memory for model 176 is large central memory extended (LCME). Extended memory for the 180-class mainframes is unified extended memory (UEM).

Extended memory for models 865 and 875 is UEM, extended core storage (ECS), or extended semiconductor memory (ESM). Extended memory for all other NOS/BE computer systems is ECS or ESM. ECS and ESM in 24-bit format standard addressing mode (sometimes called ECS mode) are the only forms of extended memory that can be shared in a multiframe complex, can be accessed by a distributive data path (DDP), and can be used as a station link medium. In this manual, ECS refers to both ECS and ESM, and extended memory (EM) refers to all forms of extended memory.

CONTROL STATEMENT AND MACRO FORMATS

In the sections that describe the control statements or macros, the control statement or macro formats must be followed carefully. All uppercase letters must be entered exactly as shown and all lowercase letters must be replaced with the appropriate characters as described after the format.

Unless otherwise indicated, bit and byte numbers are given in decimal; word addresses, field and table lengths, and block and page sizes are given in octal. Unless reserved for a specific purpose or group, all currently unused fields, names, codes, and so on are reserved for future development.

SUBMITTING COMMENTS

The last page of this manual is a comment sheet. Please use this comment sheet to give us your opinion of the manual, to suggest specific improvements, and to report technical or typographical errors. If the comment sheet has already been used, you can mail your comments to:

Control Data Corporation
Publications and Graphics Division ARH219
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198

Please include the manual title, publication number, and revision level with each inquiry, and indicate whether or not you would like a reply.

Additionally, if you have access to SOLVER, an online facility for reporting problems, you can use it to submit comments about the manual. Use NB0 as the product identifier.

RELATED PUBLICATIONS

Programming information for the various forms of extended memory can be found in the COMPASS Reference Manual and in the appropriate computer system hardware reference manual. Hardware descriptions of the various forms of extended memory can be found in the following manuals.

<u>Control Data Publication</u>	<u>Publication Number</u>
Extended Semiconductor Memory Hardware Reference Manual	60455990
Extended Core Storage Reference Manual	60347100
Extended Core Storage II and Distributive Data Path Reference Manual	60430000

The NOS/BE Manual Abstracts is a pocket-sized manual containing brief descriptions of the contents and intended audience of all NOS/BE and product manuals. The abstracts can be useful in determining which manuals are of greatest interest to a particular user.

The following is a list of NOS/BE operating system manuals, NOS/BE product set reference manuals, and other manuals of interest.

Manuals, except for the NOS/BE System Programmer's Reference Manual, Volumes 1 and 2, are available through Control Data sales offices or Control Data Literature Distribution Services (308 North Dale, St. Paul, Minnesota 55103). The NOS/BE System Programmer's Reference Manual, Volumes 1 and 2, is available through Software Manufacturing Distribution (ARH230, 4201 North Lexington Avenue, St. Paul, Minnesota 55126-6198).

<u>Control Data Publication</u>	<u>Publication Number</u>
CYBER Loader Reference Manual	60429800
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
CYBER 170 Computer System Model 825 Hardware Reference Manual	60469350
CYBER 170 Computer Systems Models 835 and 855 Hardware Reference Manual	60469290
CYBER 170 Computer Systems Models 865 and 875 Hardware Reference Manual	60468920
EXPORT High Speed Reference Manual	60456880
Export/Import Reference Manual	60436200

<u>Control Data Publication</u>	<u>Publication Number</u>
Extended Core Storage Reference Manual	60347100
Extended Semiconductor Memory Hardware Reference Manual	60455990
INTERCOM Version 5 Reference Manual	60455010
NOS/BE Version 1 Diagnostic Handbook	60494400
NOS/BE Version 1 Diagnostic Index	60456490
NOS/BE Version 1 Installation Handbook	60494300
NOS/BE Version 1 Operator's Guide	60493900
NOS/BE Version 1 Reference Manual	60493800
NOS/BE Version 1 System Programmer's Reference Manual, Volume 2 of 2	60457370
NOS/BE Version 1 Manual Abstracts	84000470
RHF Access Method Reference Manual	60459990
SCOPE Version 2 Operator's Guide	60455090
Update Reference Manual	60449900

DISCLAIMER

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

M.CLRST - Clear Status	3-21	Maintenance Channel Error Processing	3-36
M.CPJ - Capture Peripheral Job	3-21	Software Access to Maintenance Channel	3-36
M.CPUST - Change CPU Status	3-21		
M.DCP - Drop Central Processor Job	3-21		
M.DFM - Process Dayfile Message	3-22		
M.DPP - Drop PP	3-23	4. FILES AND FILE TABLES	4-1
M.EES - Enter Event Stack	3-23	Files	4-1
M.EESD - Enter Event Stack and Drop PP	3-23	System Files	4-1
M.ICE - Initiate Central Executive	3-24	Permanent Files	4-2
M.ISP - Initiate Stack Processor	3-25	Local Files	4-2
M.KILL - Bad Function Request	3-25	Queue Files	4-2
M.MCH - Read Maintenance Channel	3-25	Input Queue	4-3
M.MFLA - Monitor Field Length Access at Control Point	3-26	Output Queue	4-3
M.NOTE - Null Function	3-26	DAYFILE Macro	4-3
M.NTIME - Enter New Time Limit	3-26	ROUTE Macro - Additional Capabilities	4-3
M.PASS - PPMTR Ignores Function Request	3-26	ACQUIRE Macro	4-7
M.PATCH - Insert a Patch in PPMTR	3-26	I/O Tables	4-17 through 4-21
M.PPLIB - PP Library Search Function	3-27	File Tables	4-17 through 4-21
M.RACT - Request Control Point Activity	3-27	File Environment Table (FET)	4-22
M.RBTSTO - Request RBT Storage	3-28	File Name Table (FNT)	4-26
M.RCH - Request Channel Reservation	3-28	Device Tables	4-26
M.RCLCP - Recall Central Program	3-29	Equipment Status Table (EST)	4-26
M.RCP - Request Central Processor	3-29	Dismountable Device Table (DDT)	4-26
M.RPJ - Request Peripheral Job	3-29	Mounted Set Table (MST)	4-27
M.RPJD - Request Peripheral Job and Drop PP	3-30	Device Status Table (DST)	4-27
M.RSTOR - Request Storage	3-30	Device Activity Table (DAT)	4-27
M.SCB - System Circular Buffer Surveillance	3-31	Channel Status Table (CST)	4-27
M.SCH - Initiate Integrated Scheduler	3-31	Tapes Staging Table (STG)	4-28
M.SEF - Set Error Flag	3-31	Tape Drive Scheduling	4-29
M.SEQ - Assign Job Sequence Number	3-32	Automatic Tape Drive Assignment	4-29
M.SETST - Set Status Bits	3-32	Tape Job Prescheduling	4-29
M.SLICE - Terminate Time Slice Period	3-33	Job Scheduling with Tape Drive Overcommitment	4-30
M.SLPER - Initiate Central Monitor in Other CPU	3-33	Dynamic Tape Drive Status Checking	4-30
M.SPM - SPM Call from 1SP/1SQ	3-33	RMS Set Terminology	4-31
M.SPRCL - Stack Processor Recall	3-34	Public Sets	4-31
M.TRACE - Enter Monitor Trace Mode	3-35	Private Sets	4-32
M.TSR - Terminate Storage Request	3-35	Shared Device Sets	4-33
CYBER 180-Class Mainframe Maintenance Channel	3-36	RMS Tables	4-34
		Record Block Reservation Table (RBR)	4-35
		Record Block Table (RBT)	4-35
		INTERCOM/RDF Tables	4-41
		5. INPUT/OUTPUT	5-1
		CIO Codes	5-1
		Circular Buffer	5-2
		CIO Operation	5-4
		CPCIO	5-4
		Allocatable Device I/O	5-18
		Stack Processor	5-21

Dismountable Pack		Device Queue	7-10
Processing - I/O Detail	5-36	Permanent File Queue	7-10
Extended Memory-Buffered I/O	5-41	Permanent Pack Queue	7-10
819 Disk I/O Processing	5-43	Operator Action Queue	7-11
Logical I/O Processing	5-43	INTERCOM Queue	7-11
General Description	5-43	Job Advancing	7-11
Logical I/O Segments	5-46	Control Statement Processing	7-13
Physical I/O Processing	5-47	Job Control Statement Source	
General Description	5-47	File	7-14
Physical I/O Segments	5-48	Job Termination	7-16
PPIO Processing	5-50	Normal Termination	7-16
LCM Buffer Management	5-52	Permanent Files	7-17
Tables	5-52	Local Files	7-17
Transfer Buffer Table	5-52	Input File	7-17
TBT Address Table	5-53	Output File	7-17
Unit Queue Table	5-53	Dayfile	7-17
Channel Table	5-53	Abnormal Termination	7-17
CE Error File	5-54	Job Post-Processing Utilities	7-18
819 Subsystem Flush Function	5-58	List-of-Files Address - GETLOF	
		and SETLOF Macros	7-18
		Job Control with Logical Identifiers	7-19
6. PERMANENT FILES	6-1	CEVAL Macro	7-20
Permanent Files - System Interface	6-1	CESAB Macro	7-28
Permanent File Interlocks	6-1	Access to Loader Word - GETLC	
Permanent File Tables	6-2	and SETLC Macros	7-32
Permanent File Accounting	6-4		
Permanent File Utility Routines	6-5	8. EDITLIB	8-1
DUMPF Utility	6-6	Introduction	8-1
PFLOG Utility	6-8	Character Set	8-1
RMS Set Processing	6-10	Syntax and Semantics	8-3
LABELMS Control Statement	6-10	System EDITLIB	8-5
ADDSET Control Statement	6-10	EDITLIB Files	8-5
MOUNT Control Statement	6-11	Special Handling of Local	
RELABEL Control Statement	6-11	File Name System	8-7
RECOVER Control Statement	6-12	Control Statement	8-7
RECOVER Notes and Cautions	6-15	Directives	8-10
Addressing Public Sets by		Optional Directive Parameters	8-10
Set Attribute	6-15	Directive Restrictions	8-12
		Directive Formats	8-13
7. JOB PROCESSING	7-1	ADD Directive	8-12
Job Flow	7-1	CHANGE Directive	8-13
Job Input Queue	7-1	COMPLETE Directive	8-13
Tape Job Scheduling	7-1	CONTENT Directive	8-14
Loading Jobs from Tape	7-2	DELETE Directive	8-14
Sequencer Jobs	7-2	ENDRUN Directive	8-14
JANUS	7-3	FINISH Directive	8-14
Integrated Scheduler	7-4	INCLUDE Directive	8-15
Job Scheduling	7-6	LIBRARY Directive	8-15
Rolling	7-6	LISTLIB Directive	8-16
Swapping	7-6	LISTLNT Directive	8-16
Job Control Area	7-8	MOVE Directive	8-16
Job Descriptor Table (JDT)	7-9	RANTOSEQ Directive	8-17
Job Scheduling Queues	7-9	READY Directive	8-17
Central Memory Queues	7-9	REMOVE Directive	8-17
		REPLACE Directive	8-18

REWIND Directive	8-18	System Security	10-4
SEQTORAN Directive	8-18	Reserved Names	10-4
SETAL Directive	8-18	LDCMR Interlock	10-5
SETFL Directive	8-19	LDC	10-5
SETFLO Directive	8-19	Examples	10-5
SKIPF Directive	8-19		
TRANSFER Directive	8-20		
TRANS77 Directive	8-20		
Library Directory Access	8-20	11. SYSTEM DUMPS	11-1
PP Routines	8-20	Deadstart Dump Analyzer	11-1
CP Routines	8-21	DSDUMP Control Statement	11-1
EDITLIB	8-21	Analysis Directives	11-2
Library Directory Pointer		Display Directives	11-2
Format	8-21	Dump Directives	11-2
PP Library Pointer Format	8-22	Special Dump Directives	11-6
Files	8-22	Default Directives	11-7
System Files	8-22	DSDUMP Messages	11-7
User Files	8-22	System Dynamic Dump	11-8
File and Library Positioning	8-22	Interface	11-8
System Security	8-23	Dump File Format	11-8
MDI (Move System Directory)	8-24	Listing Dump Files	11-10
Directory/Library/Program Limits	8-25	Error Messages	11-11
Table Formats	8-25		
Examples	8-35		
9. SYSTEM BULLETIN UTILITY	9-1	12. DOWNLINE LOADING UTILITY	12-1
BULLUP Statement	9-1	BCLOAD	12-1
BULLUP Data Statements	9-1	DLEB	12-6
Name Statements	9-2	WPPF	12-6
Contents Statements	9-2	ANSWER Buffer	12-7
Processing Data Statements	9-2	DLL	12-7
Creating a System Bulletin File	9-3	Building a Controlware File	12-8
Updating a System Bulletin File	9-3	Examples	12-8
Reducing a System Bulletin File	9-4		
NOS/BE-INTERCOM Considerations	9-4		
10. LDCMR	10-1	13. DSD L-DISPLAY PROGRAMMING	13-1
Introduction	10-1	Using the L-Display	13-1
LDCMR Control Statement	10-1	Display Data	13-1
LDCMR Files	10-3	Inputting Data	13-2

INDEX

FIGURES

2-1 Allocation of CM	2-1	5-2 FET - Circular Buffer Interface	5-3
2-2 Sample Control Point Storage	2-2	5-3 Device Set I/O Processing	5-37
2-3 Call Formats	2-6	5-4 Output Flow to RMS File	5-42
2-4 Typical CMR Assignments	2-7	5-5 CYBER 176 Computer System with	
2-5 Extended Memory System Areas	2-15	819 Disks	5-43
2-6 Extended Memory System Image	2-19	5-6 Logical I/O Processing	5-44
2-7 Extended Memory Segment	2-20	5-7 Physical I/O Processing	5-48
3-1 Pool PP Layout	3-4	5-8 PPIO Processing	5-51
3-2 PP Input Register	3-5	5-9 CE Error File	5-54
3-3 PP Resident Routines	3-7	6-1 APF Table Entry	6-4
3-4 PP Chain	3-17	6-2 Permanent File Dump Tape Format	6-6
3-5 PP Job Queue (MTR Memory)	3-19	6-3 File Header	6-7
4-1 System Dayfile Area	4-24	6-4 PFLOG Dump Tape Format	6-8
4-2 System File Entries	4-25	7-1 JANUS Interfaces	7-4
4-3 Nonallocatable Device File		7-2 Integrated Scheduler Interfaces	7-5
Processing	4-28	7-3 Scheduler Request Stack in	
4-4 RMS Tables	4-34	System Exchange Package Area	7-7
4-5 Record Block Table	4-36	Control Statement Processing	
4-6 File Table Interfaces - FNT		Flowchart	7-12
Points to RBT Chain	4-37	8-1 Basic Usage of System EDITLIB	8-2
4-7 File Table Interfaces - RBT		8-2 Library Table Interfaces	8-34
Points to RBR Via DDT	4-39	13-1 L-Display Communication Buffer	13-1
4-8 File Table Interfaces - RB		13-2 L-Display Lines	13-2
Byte Points to RB	4-40	13-3 CPU Program Interface to DSD	13-2
5-1 Circular Buffer Interface	5-3		

TABLES

2-1 Example of Storage Moves	2-3	5-8 WRITEN Macro Logical Sequence	5-15
2-2 SFCALL Return Codes	2-30	5-9 WPHR Macro Logical Sequence	5-16
3-1 PP Direct Cell Assignment	3-1	5-10 SKIPF Macro Logical Sequence	5-16
4-1 ACQUIRE Macro Parameters	4-13	5-11 SKIPB Macro and BKSP Macro	
4-2 Default RB Size	4-33	Logical Sequence	5-17
5-1 READ Macro Logical Sequence	5-6	5-12 BKSPRU Macro Logical Sequence	5-17
5-2 READN Macro Logical Sequence	5-8	5-13 Stack Processor Orders	5-26
5-3 READSKP Macro Logical Sequence	5-9	5-14 Ranges of Cylinders Used	5-34
5-4 RPHR Macro Logical Sequence	5-10	6-1 Header Fields	6-7
5-5 WRITE Macro Logical Sequence	5-11	8-1 Directive Interpretation and	
5-6 WRITER Macro Logical Sequence	5-12	Execution	8-8
5-7 WRITEF Macro Logical Sequence	5-13		

VOLUME 2

APPENDIXES

A. CHARACTER SETS	A-1	D. DISK TABLES AND FILE FORMATS	D-1
B. CENTRAL MEMORY RESIDENT		E. EXTENDED MEMORY TABLES	E-1
TABLES	B-1	F. SYMBOL DEFINITION	F-1
C. JOB CONTROL POINT TABLES	C-1	G. ABBREVIATIONS AND ACRONYMS	G-1
		H. RHF TABLES	H-1

The Batch Environment Network Operating System (NOS/BE) is an operating system for the CYBER 180; CYBER 170; the CYBER 70, Models 71, 72, 73, and 74; and the 6000 computers. NOS/BE accepts input in the form of jobs submitted by users, processes jobs as directed by the accompanying job control statements, and provides operations control in accordance with command instructions that are input at the console's keyboard. This section describes the inherent hardware characteristics, the basic software elements, and how they work together to accomplish efficient processing of users' jobs.

HARDWARE CHARACTERISTICS

The operating system uses peripheral processor units (PPs) for system and input/output (I/O) tasks and a central processor unit (CPU) to execute user and system jobs. Central memory (CM) contains the user programs. System software areas are located at the upper and lower ends of CM. Extended memory may be included and may contain system libraries and other items (such as file buffers for rotating mass storage and swap files), to conserve CM space or to decrease access time on mass storage devices.

CENTRAL PROCESSOR (CPU)

The CPU performs tasks of a computational nature. It has no I/O capability. It communicates with other system components through CM. The CPU is used almost exclusively for program compilations, assemblies, and executions. The CPU makes system requests through a CPU request word located at the reference address plus one (RA+1) of the current program in execution. The CPU is discussed in section 2.

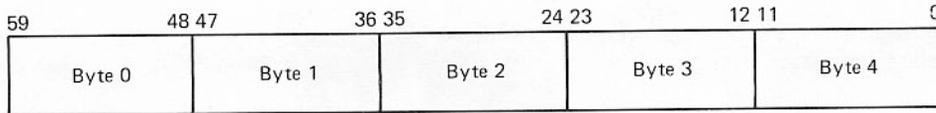
PERIPHERAL PROCESSORS (PPs)

The PPs, of which there may be up to 20 (identified as PP0,PP1,...,PPn), are identical. They perform many tasks for requesting programs in CM. A PP can be assigned to control, I/O, job scheduling, control statement interpreting, 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 CM through which communications with the system are conducted. Each block contains an input register, an output register, and a message buffer. PPs are discussed in section 3.

CENTRAL MEMORY (CM)

CM 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 as follows:



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

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, and the RBT is discussed in section 4.

EXTENDED MEMORY

Features

Extended memory is divided into the system area, the paged area, and the direct access area. These areas function as follows:

<u>Area</u>	<u>Function</u>
System	Contains system pointers and tables required by extended memory software.
Paged	Contains buffers, library programs, swap files, and other files assigned to extended memory.
Direct access	Assigned to user as result of job statement request; used and managed by the user; contains the CMR segments in extended memory control point 0 field length.

This division of extended memory allows the following features to be provided.

- I/O buffering to speed up RMS sequential transfer rates.
- Library residence in extended memory.
- Job swapping to/from extended memory by the integrated scheduler.
- User access to extended memory.
- Segmentation of system central processor code.

Extended Memory Paging

The major uses of the paged area of extended memory are for swap files, I/O buffering, and library residence. The basic element of extended memory paging is a PRU made up of 1008 CM words. A group of eight consecutive PRUs (1008 CM words) forms a page. The parameter IP.EBUF determines the default buffer size for I/O buffering and has a default value of 1610 pages.

Extended memory paging provides the following advantages:

- More efficient use of extended memory through dynamic allocation/release of space.
- Availability of extended memory to more users by allowing the use of an over-commitment algorithm for extended memory.

SOFTWARE ELEMENTS

Files and control points are basic to the operating system.

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 is terminated by an end-of-file indicator. The jobs processed by the operating system 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. Control points are discussed in section 2.

SYSTEM ORGANIZATION

The operating system consists of PP programs, CP programs, macro definitions, and symbol definitions. The entire system is contained on program library files produced by the library maintenance program Update. Programs on these library files are in source language form. Installation options are provided to permit flexible selection of system features during extraction of programs and texts from the libraries. These software components can be assembled for subsequent creation of a deadstart file on tape by the system maintenance program EDITLIB. EDITLIB is discussed in section 8.

A system monitor controls the operating system. The system monitor consists of PP overlay MTR (operates in PP0) and CPMTR. In a disk system, CPMTR is assembled as part of CMR; in an extended memory system, it consists of a number of separate segments.

SYSTEM LOADING

To load the operating system, the deadstart tape or disk is mounted on the appropriate unit, and a small bootstrap loader program is set up on the hardware deadstart panel switches. When the deadstart button on the operator's console or deadstart panel is pressed, the bootstrap program is transferred to and executed in PP0. The bootstrap loader reads the first record on the deadstart file and executes the routine contained there. This routine reads in the remainder of the common test and initialization (CTI) routines. The CTI routines determine the machine attributes and run a confidence test to ensure that the mainframe is operating properly. When confidence testing is complete, the first operating system routine (OSB) is read into PP0. It 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 and 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, PP2, and PP3. CMR is read from the deadstart file into CM, and a display shows all deadstart functions and options that may be selected. The functions include the following:

- Library reload option.
- Queue file recovery level.
- User device set processing.
- Equipment configuration changes to the system.
- Initialization level of extended memory.
- First mainframe to deadstart.

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

After deadstart options are processed, control is passed to MTR and the display routine, DSD. If a deadstart tape was used rather than disk, the tape is rewound to its load point and is not referenced again during normal system operation. The tape can be removed and the tape unit can be cleared for other operations. At this point, the system can process jobs (refer to section 7).

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

- System libraries stored on one mass storage device (more if the first device overflows) in the system set. Programs can be loaded from any system library into PPs or CM as needed.
- CMR loaded into the low end of CM. A set of tables in CMR contains information about the system. Some tables are used by the system PP programs to communicate with each other. An RBT is built into the upper end of CM.
- Some system programs stored in CM in an area immediately following 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. Only the most frequently used library programs are stored in CM. For the same reason, CMR is kept as small as possible.
- The system monitor programs (MTR in PP0 and CPMTR in CMR) which control the system. They control allocation of system physical resources (CM, extended memory, channels, equipments, PPs, and CPUs), handle all communications between user programs and the system, and coordinate activities of the other PPs.

- PP1 which contains the operator console display driver program DSD. DSD provides a communication path between the system and the operator. Current system status is displayed by DSD on the operator console display(s). The operator can control system operation by typing commands on the console keyboard.
- Remaining PPs which contain 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 program is responsible for reading the input register, loading PP overlay programs into its assigned PP, and providing communications between the overlays and MTR.

SYSTEM TAPE

The released system deadstart tape consists of the following programs.

<u>Name</u>	<u>Description</u>
ZZZ	Operating system header record (CTI interface).
OSB	Operating system bootstrap routine; resides in PP0.
CED-CEF	Deadstart PP control programs; reside in PP0.
DDR	Read driver for 844 or 885 disk subsystem; resides in PP1.
MDR	Read driver for 66x and 67x tape drives; resides in PP1 and PP3.
CMR	Central memory resident (up to 64 copies).
COM	Deadstart option matrix generator; resides in PP2.
IRP	Deadstart RMS driver control program; resides in PP2.
5CY	Deadstart driver for 844-21, 844-41, and 885 disk subsystems.
LFP	Deadstart 7000 type peripheral processor unit (PPU) loader (CYBER 176 only).
HCD	PPU driver for 819 disk subsystem (CYBER 176 only).
0SY	844 buffer controlware for the 7054 controller.
0SZ	844 buffer controlware for the 7154 controller.
0SJ	885 buffer controlware for the 7155 controller.
0MT	66x magnetic tape subsystem controlware.
IRCP	Deadstart main CP program.
STL	Deadstart PP initiation program (PP resident).
MTR	System monitor program.
DSD	Display control program.

<u>Name</u>	<u>Description</u>
Directory	Library name table.
	PP name table.
	PP programs; the first must be stack processor's segment.
System Libraries	Entry point table.
	External reference list.
	External reference table.
	Program number table.
	Program name table.
	CP routines.

An installation may expand the preceding records by placing up to 63 additional CMR records on the system tape for different equipment configurations.

CM ORGANIZATION

Figure 2-1 shows the allocation of CM.

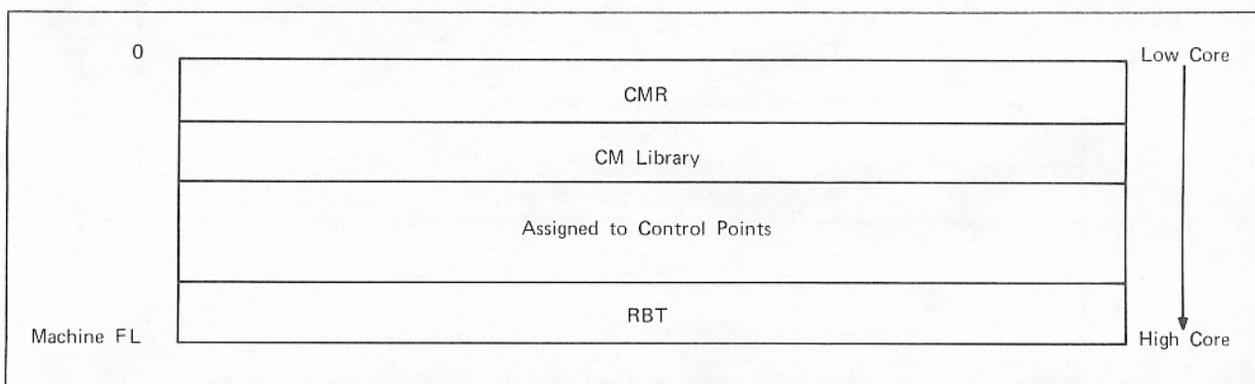


Figure 2-1. Allocation of CM

Low core is allocated to the central memory resident (CMR) portion of the operating system, 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 POINTS

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

Any number of control points up to 15 decimal are possible. In the released system, the default value of N.CP is 15. In an installation with n control points for user jobs, they are numbered from 1 to n . Only one job can be assigned to a control point at any time. Once a job is assigned to a control point, system resources such as CM, extended memory, 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 storage block assigned to the job at control point 2 is higher than the block for the job at control point 1, storage for control point 3 is always higher than that for control point 2, and so on.

In figure 2-2, no storage is assigned to control points 3 and 5; unassigned storage appears between assigned storage.

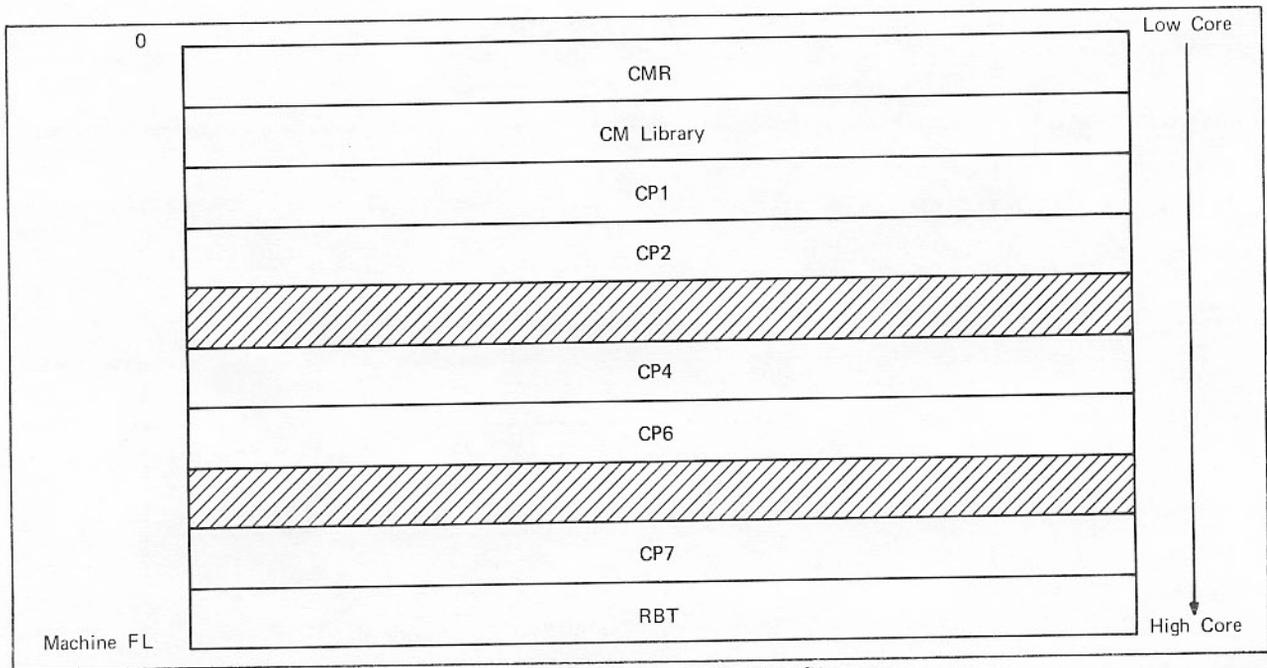


Figure 2-2. Sample Control Point Storage

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

Control point 0 identifies 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 control point number is entered into the system table entry for that equipment. If not assigned to a job at a control point, the equipment is assigned to control point 0 and is available to be assigned to a job. All active system files are attached to control point 0. 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 and remote batch processors.

Control point $n+1$ is used by CPMTR for executing system jobs such as the integrated scheduler or storage move routines. Control point $n+1$ has an abbreviated control point area that consists primarily of an exchange package. The field length of control point $n+1$ is all available memory.

JOB DESCRIPTOR NUMBER

During 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 in, it may be associated with a different control point.

While 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 identifies the job during its entire execution.

To clarify the difference between job descriptor number and control point number, JDT numbers start at $n+1$ (n is the number of control points).

STORAGE MOVES

CM storage must be reallocated and jobs must be moved as jobs finish processing and new jobs begin or as jobs are swapped in and out. If a job at a control point requests additional storage, it may be necessary to move jobs to obtain the required storage. CPMTR keeps a tally of unassigned CM in CMR word T.UAS.

Storage associated with each control point is allocated or unallocated. Either storage 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 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, results in a transfer of unallocated storage to allocated storage; no storage move takes 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 are scanned. This scan locates the combination of unallocated storage blocks that result in a move of the least amount of storage.

If control point 1 in figure 2-2 needs more storage, it is 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 are moved downward to provide the storage. Added storage always extends the field length upward.

Example:

Control point 5 requests an FL of 300 (refer to table 2-1). All values are increments of 100 octal. If CPMTR takes the UAS from control point 7, the 150 units of CM at control point 6 must be moved. However, taking UAS from control points 3 and 4 requires moving 120 units of CM at control points 4 and 5 (20 units are moved from 4 to 3; 100 units are moved from 5 and added onto the 20 units moved to 3).

TABLE 2-1. EXAMPLE OF STORAGE MOVES

Control Point	Before			After		
	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

CP – SYSTEM COMMUNICATION

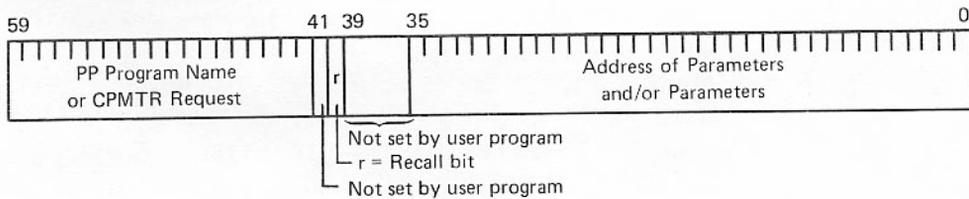
A running CP program must communicate with the system in the following situations.

- When a CP program is loaded and executed as a result of a control statement call. The system must place any parameters specified on the control statement in an area where they can be read by the CP program.
- When a CP program needs to perform input/output. No CP instructions allow a CP program to perform I/O. The CP program must send a request to the system to load a PP program to execute the I/O.
- When a CP program terminates. The program must advise the system that the system may process the next control statement.

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 101g locations of each job's field length are reserved for this purpose. The following 10g words are reserved for the loader table. The first program loaded into a user field length is always loaded at location RA+111g (for the user, this is location 111g because the reserved words are RA+0 through RA+110g).

The RA communication area is shown in appendix C.

The first word of a user field length (location RA+0) is reserved for use of hardware and software flags. Location RA+1 is used by the user program to request action by CPMTR or a PP program. The form of the request is as follows:



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

CP – PP COMMUNICATION

If a user's program places a call for a PP program in RA+1 and performs an XJ instruction, CPMTR will pick up the RA+1 call, insert the control point number of the caller into bits 39 through 36 of the word, and clear bit 41. If CPMTR determines that the RA+1 call should be assigned to a PP, it passes the call on to MTR (T.MTRRS).

When a PP is available, MTR writes the word into the input register of the available PP in CMR (refer to figure 3-2 for the format of a PP input register of a transient program called from a CP program). The name, the auto recall bit, and any parameters in bits 35 through 0 appear in the input register exactly as they did in RA+1. Parameters are passed from a CP program to a PP program through this parameter field.

For example, if the PP program CIO is called, CIO finds the relative address of the file environment table (FET) 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. In extended memory systems, CPMTR traps all CIO calls and sends them to CPCIO where the device type is checked. If the device is RMS or extended memory, the request is processed; otherwise, the request is sent to CIO.

MTR continually scans RA+1 in the event that the user's program does not use the XJ instruction. When an 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 always set to indicate that the XJ instruction is available to the user program. This bit is primarily needed for old software that tested bit 59 to determine if the XJ instruction could be used.

PROGRAM RECALL

The recall program status enables efficient use of the CP and capitalizes on the multiprogramming capability of the operating system. 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 is placed in a loop to await I/O completion, a CP program can request that the control point be put into recall status until a later time so that the CPU can be assigned to execute a program at another control point. The job 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. The program is not eligible for execution until the specific request has been satisfied. Periodic recall can be used when the program is waiting for one of several requests to be completed. The program is activated periodically so that it can determine which request has been satisfied and whether or not it can proceed.

To enter periodic recall, a CP program inserts the characters RCL left-justified into RA+1. Upon encountering the RCL request, CPMTR examines the auto recall bit (bit 40). If set, the request is considered to be an auto recall request. If it is not set, CPMTR checks bits 10 through 0 (decimal) for a delay count. The delay count is specified in units of 0.244 millisecond (the same as the real-time clock). The largest delay time that can be specified is 2047 (decimal) or approximately 0.5 second. If this delay count is not set, CPMTR specifies a default value for it. The current default, as defined by symbol RCLPER in the CMR internal configuration parameters, is 25 milliseconds. The delay count of the control point in periodic recall is examined regularly by the advance control point routine (ACP) of MTR. When the delay count expires, the control point loses its recall status, and the CPU is again assigned to execute the program at the control point. At this time, the CP program can check the completion bit in the FET to see if the I/O is finished. If so, the CP program can proceed with computations. If I/O is not complete, the CP program can go into recall.

To enter auto recall, a CP program makes a request in RA+1 with bit 40 of RA+1 set to one. The control point is put into auto recall after the request has been initiated. The CPU is assigned to another control point. The program in recall is 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 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 is put into auto recall. If bit 40 is set, bits 17 through 0 of RA+1 must contain the address of a word in the program's field length called a reply word. When the PP has completed its function, it sets the completion bit (low order bit) in the reply word. When the completion bit is set, MTR restarts the program.

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

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 go into auto recall without calling a PP program by putting RCL left-justified into RA+1 and setting bit 40 of RA+1 to one. Bits 17 through 0 of RA+1 must contain the address of a reply word. A program which has already initiated one or more I/O operations might go into auto recall in this way, using the first word of the FET associated with one of the I/O operations as the reply word. 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 are shown in figure 2-3. 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. All the requests are issued without recall (using a separate FET for each request). Then the user requests 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 files are incomplete.

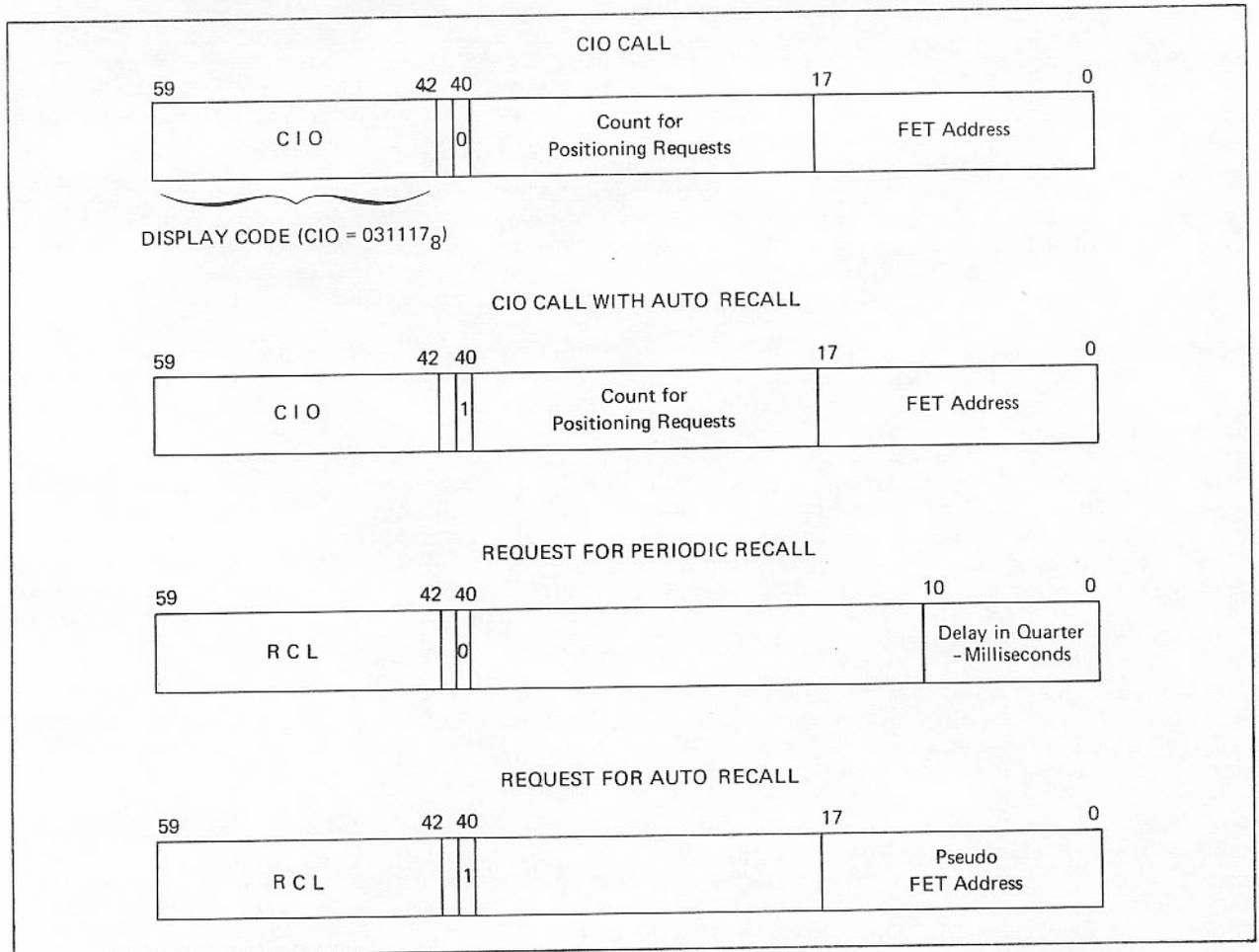


Figure 2-3. Call Formats

Periodic recall can also be used when a CP program can initiate an I/O request and perform computation. In some cases, the I/O would be completed before the computation; in others, the computation would complete first. The user would go into recall only after computation was completed 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 PRUs have been processed.

CENTRAL MEMORY RESIDENT (CMR)

The low end of central memory is reserved for the CMR portion of the operating system and the system library portions which reside in CM. CMR contains pointers, tables, and programs. Its length depends upon several factors, including the number of PPs and the number of control points, which determine the number of tables and the length of certain tables in CMR. Some CMR tables are optional and appear only by installation parameters. Figure 2-4 illustrates a typical CMR.

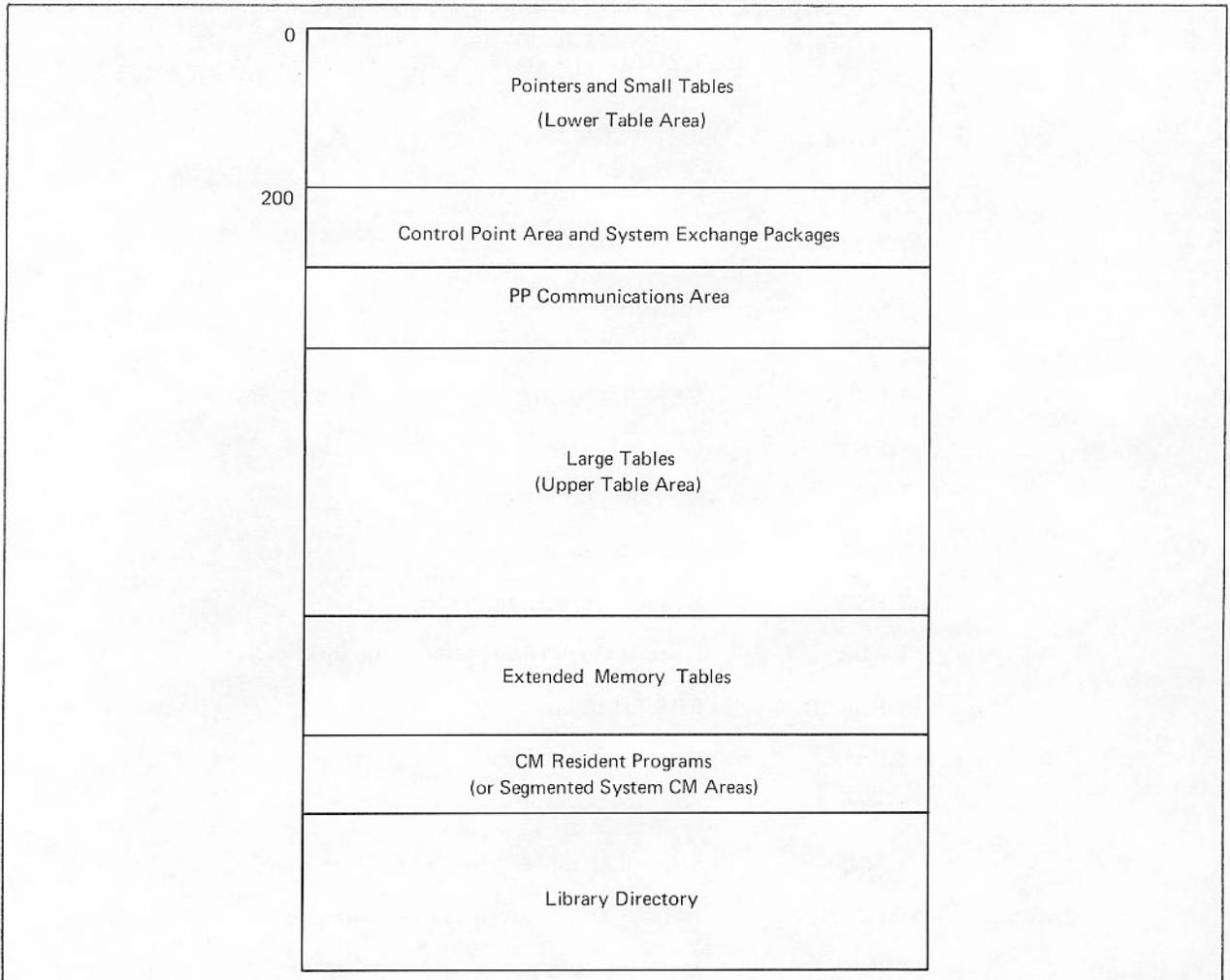


Figure 2-4. Typical CMR Assignments

The CMR area contains the following tables and information.

<u>First Word Address</u>	<u>Table Name</u>	<u>Description</u>
0		CMR pointer area.
100	T.CST	Channel status table.
154	T.PPS1	PP status words.
200	T.CPA _n	Control point areas.
	T.XPSCH	System job exchange package area.
	T.PPCI	PP communication areas.
	T.EST†	Equipment status table.
	T.FNT†	File name table. CIO-CPCIO special file name tables. Permanent file name tables. T.ELIBD - Extended memory resident library descriptor word.
	T.ITABL†	INTERCOM table.
	T.DAT†	Device activity table.
	T.RMSBUF†	RMS buffer.
	T.STG†	Tapes staging table.
	T.APF	Attached permanent file table.
	T.EXPIO	CYBER 176 exchange package and I/O buffers.
	T.CHT	Channel table.
	T.UQT	Unit queue table.
	T.RQS††	Request stack.
	T.RST	Request scheduling table.
	T.RBR	Record block reservation table (headers).
	T.RBRBIT	RBR bit table.
	T.DST	Device status table.
	T.DOT	Device overflow table.
	T.SEQ	Sequencer table.
	T.ITMSG	INTERCOM 5 terminal dayfile message.
	T.INS	Installation area.

† Table must begin before 10000g.

†† Table must begin before 20000g.

<u>First Word Address</u>	<u>Table Name</u>	<u>Description</u>
	T.MST	Mounted set table.
	T.DDT	Dismountable device table.
	T.TRB	Trace buffer.
	T.VRNBUF	VSN buffer.
	T.TAPES	Tapes table.
	T.URT	Tape unit recovery table.
	T.MAIL	Scheduler mailbox buffer.
	T.IDT	Logical ID table.
	T.DFB	Dayfile buffers.
	T.PJT	Parameter storage for delayed PP jobs.
	T.MAB	Mainframe attribute block.
	T.QLBMP	Logfile bitmap table.
	T.SSCT	Subsystem control table.
	T.SCHPT	(Optional) scheduler performance table.
	T.SCHJCA	Scheduler job control area.
	T.SCHJDT	Scheduler job descriptor table.
	T.EICB	Environment interface communication table.
	T.EPIB	Error Processing Interface Block.
	T.ELST	Error logging status table.
	T.PPOVL	PP resident overlay save buffer.
	T.BRKPT	Breakpoint table (extended memory system).
	T.AREA	Area table (extended memory system).
	T.ENTRY	Entry table (extended memory system).
	T.BCFAP	CEFAP buffer.
	T.EPAGE	Empty page stack (extended memory system).
	T.ECSPRM	Extended memory parameters.
	T.SCBHDR	System circular buffer.
	T.SUBPG	Subpage buffer.

<u>First Word Address</u>	<u>Table Name</u>	<u>Description</u>
		CM resident programs (disk system).
		Segmented system areas (extended memory system).
	T.LIB	Library directory.
		INTERCOM pointer area. INTERCOM buffer and user tables.
		Job control point user field length.
	T.RBT	RBT chains.

SUMMARY OF CMR AREAS

<u>Area</u>	<u>Description</u>										
Lower table	Contains pointers to larger tables in the upper table area of CMR, various flags, constants, and installation options 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 CM.										
Control point	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	Contains eight words for each PP 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 six-word message buffer.										
Upper table	Contains major tables pertinent to system and job operation.										
Extended memory table	Contains buffers for transferring PP overlays and RMS files.										
CMR program	Contains five resident programs: <table border="1" data-bbox="616 1371 1330 1689"> <tbody> <tr> <td>CP.MTR</td> <td>Central processor monitor.</td> </tr> <tr> <td>CP.SM</td> <td>Central processor storage move.</td> </tr> <tr> <td>CP.SPM</td> <td>Central processor stack processor manager.</td> </tr> <tr> <td>CP.SCH</td> <td>Central processor memory manager (scheduler).</td> </tr> <tr> <td>CP.CIO</td> <td>Central processor CIO processor.</td> </tr> </tbody> </table>	CP.MTR	Central processor monitor.	CP.SM	Central processor storage move.	CP.SPM	Central processor stack processor manager.	CP.SCH	Central processor memory manager (scheduler).	CP.CIO	Central processor CIO processor.
CP.MTR	Central processor monitor.										
CP.SM	Central processor storage move.										
CP.SPM	Central processor stack processor manager.										
CP.SCH	Central processor memory manager (scheduler).										
CP.CIO	Central processor CIO processor.										
	In a segmented system, the CMR program area is overlaid by the extended memory system resident and the CP code overlay segments.										
Library directory	Contains tables related to the system libraries, including library name table, PP program name table, and CMR library programs.										

SUMMARY OF TABLES IN UPPER TABLE AREA

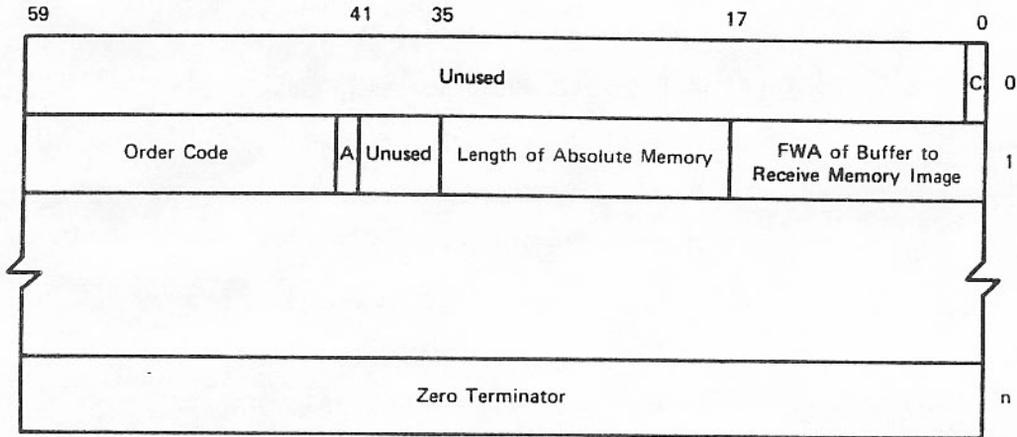
<u>Table</u>	<u>Description</u>
Equipment status	Contains one entry for each device in the system configuration. Nonallocatable devices can be assigned to one control point at a time; allocatable devices can be assigned to many control points simultaneously.
File name	Contains an entry for each file 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	Provides control information for INTERCOM use.
Device activity	Contains a four-word entry for each RMS device in system. Each entry provides dynamic information related to current activity of the RMS device.
Rotating mass storage buffer	Holds a message to be flashed on the bottom line of the B display. The message reports an error on an RMS device and asks the operator to idle down the device.
Tape staging	Defines availability, assignment, and demand for tape devices.
Attached permanent file	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 has at least one three-word entry in this table when a request for its use is active.
Record block reservation	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	Directly related to the request stack; contains a two-word entry for each allocatable device in the system, plus an additional pseudo entry for unassigned file processing.
Sequencer	Contains 30-word entry for each preallocated RMS device for use for CE diagnostic programs.
INTERCOM 5 terminal dayfile message	Contains a dayfile message (maximum six words) to be sent to a terminal.
Installation	Reserved for specific needs of installation. Tables are generated in the area only by installation.
Mounted set	Contains one entry for each mounted device set, including the public sets.

<u>Table</u>	<u>Description</u>
Dismountable device set	Contains entries for each RMS device, plus entries for each queueing device needed by jobs.
Tapes	Contains one entry (108 words per entry) for each tape unit defined.
Tape unit recovery	Contains one entry (five words per entry) for each 66x/67x tape unit.
Mailbox	Used for communications between system and swapped out jobs.
ID	Contains host ID, logical IDs, and physical (link) IDs. The ID table can be zero-length.
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 0 buffer is at the end of this area.
Peripheral job	Contains parameters saved for delayed PP jobs.
Mainframe attribute block	Contains attributes of a mainframe, such as number of PPs/PPUs and the presence of ILR/SCR, CMU, CEJ/MEJ, and CPU-1/CPU-0.
Subsystem control	Contains names of defined subsystems.
Scheduler performance	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	Contains linked entries for each class of job. Entries describe job requirements, current status, accumulated use time of system components, and so on (refer to section 7).
Environment interface	Used to establish communications with the 180 state processes in order to effectively share the CPU and properly process hardware errors.
Error logging status	Table that controls the processing of single-bit CM errors (also LCME errors for CYBER 176 systems). These errors are recorded in the maintenance registers of the CYBER 180-class mainframes; they are recorded in the status/control register for other CYBER 170 systems.
Error processing interface	Contains pointers and flags for 1MR to interface with the DFT error processing routine.
Breakpoint	Contains the breakpoint code exchange package, the breakpoint wait loop, flags and data used by DSD, and the breakpoint entries. It is used by DSD and breakpoint processing.
Area	Contains an eight-word buffer that receives the extended memory area table from a segmented system. It describes the system extended memory and CM structure.

<u>Table</u>	<u>Description</u>										
Entry	Contains the date-time stamp for this CMR, the associated segment library name, and a list of the entry points defined in the tables with their addresses.										
CM resident programs (in a disk system)	<table border="0"> <thead> <tr> <th style="text-align: left;"><u>Symbol</u></th> <th style="text-align: left;"><u>Name/Function</u></th> </tr> </thead> <tbody> <tr> <td>CP.MTR</td> <td>Central processor monitor.</td> </tr> <tr> <td>CP.SM</td> <td>Central memory storage move.</td> </tr> <tr> <td>CP.SPM</td> <td>Stack processor manager.</td> </tr> <tr> <td>CP.SCH</td> <td>Memory manager scheduler.</td> </tr> </tbody> </table>	<u>Symbol</u>	<u>Name/Function</u>	CP.MTR	Central processor monitor.	CP.SM	Central memory storage move.	CP.SPM	Stack processor manager.	CP.SCH	Memory manager scheduler.
<u>Symbol</u>	<u>Name/Function</u>										
CP.MTR	Central processor monitor.										
CP.SM	Central memory storage move.										
CP.SPM	Stack processor manager.										
CP.SCH	Memory manager scheduler.										
Library directory	Falls at the end of the CMR upper table area following the CM resident programs and extended memory tables. It contains two-word entries in the program name table section and one-word entries in the entry point table. The directory length can expand or contract as programs are added and deleted or as program residence is changed.										

STF — SYSTEM TABLE FIND

The PP routine STF copies a specified portion of central memory to the system user buffer area. This allows functions to be performed by CP programs without the need for a special PP program. Bits 17 through 0 of the input register point to a table of commands that the user specifies in the following format.



<u>Word</u>	<u>Bit</u>	<u>Description</u>
0	0	Complete bit. The complete bit must be cleared before STF can be called.* The bit is set on completion of any function.
1	59-42	Order code. The order code is either an absolute CM address, or a three-character table name or copy directive as described below. If the order code specifies an absolute CM address of a block of memory to be copied, the absolute memory flag (bit 41) must be set. If the order code specifies one of the following table names or copy directives (CPS and CPx), the absolute memory flag must not be set. The following names must be specified in display code.

<u>Name</u>	<u>Description</u>
APF	Attached permanent file table.
AUT	Auxiliary user table.
CPS	Copy a specific word of the user's control point into the buffer. The word address (relative to the start of the control point) is specified in the field length (bits 35 through 18).
CPx	Copy control point area x into the buffer. Normally, x is the binary value of the desired control point. However, when x is 0 (in either binary or display code), the user's own control point is copied.
CST	Channel status table.

<u>Word</u>	<u>Bit</u>	<u>Description</u>
	<u>Name</u>	<u>Description</u>
	DAT	Device activity table.
	DDT	Dismountable device table.
	DST	Device status table.
	EST	Equipment status table.
	FDT	DDT (fixed section).
	FNT	File name table.
	IDT	ID table.
	IUT	INTERCOM user table.
	JCA	Job control area.
	JDT	Job descriptor table.
	MST	Mounted set table.
	MUX	Multiplexer table.
	RBR	Copy RBRs.
	RBT	Copy RBTs.
	SEQ	Sequencer table.
	SPT	Scheduler performance table.
	STG	Tapes staging table.
	TPS	Tapes table.
	URT	Unit recovery table.
	VDT	DDT (variable section).
	41	Absolute memory flag. This flag must be set if the order code is an absolute memory address.
	40-36	Unused.
	35-18	Length of absolute memory if the absolute memory flag is set; unused if the absolute memory flag is clear.
	17-0	FWA of buffer to receive memory image.
n	59-0	Zero terminator word.

When the order code is a table name, the length of the table in memory is used as the length of the user buffer. The information is read from memory without any system interlocks set; therefore, the data returned may be inconsistent.

CMR SEGMENTATION FOR EXTENDED MEMORY SYSTEMS

CMR segmentation is implemented for installations with extended memory. A segmented system is intended to have most of its CP code extended memory resident. Sections of code (segments) are loaded as needed into CM overlay areas. A segmented system is started from an extended memory system image by a bootstrap monitor function, which overlays an existing system (disk or extended memory) with a new system. The extended memory system image for this new system must have been created in a previous step using utility program LDCMR (refer to section 10). For an extended memory system, the equivalent of a disk system's system resident programs section in CMR is a number of specialized areas as shown in figure 2-5.

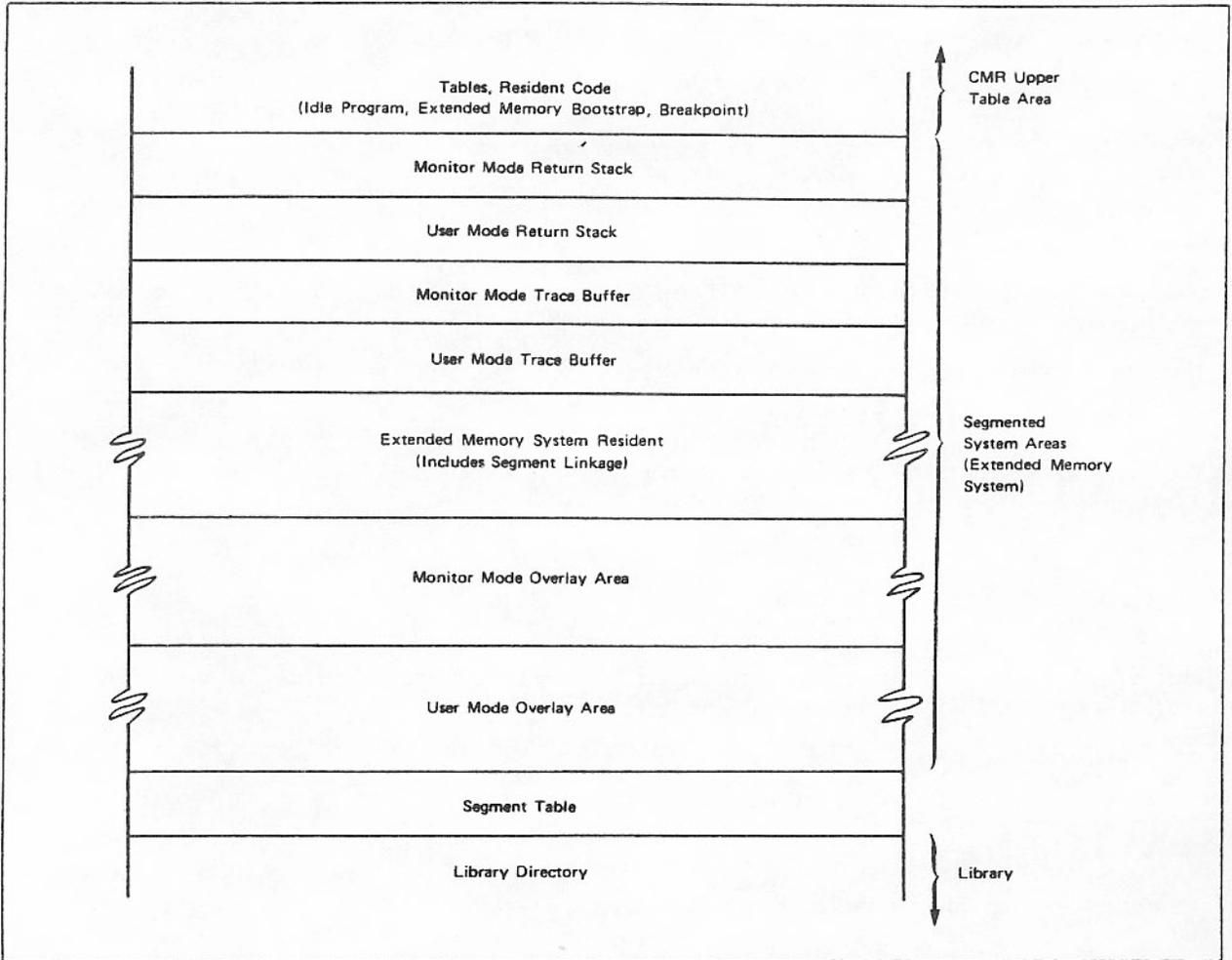


Figure 2-5. Extended Memory System Areas

These areas are set up by the initialization segment INIT, loaded by the bootstrap in the monitor mode overlay area. The position of the library directory is adjusted if necessary by LDCMR. The extended memory system resident contains the segment linkage program which loads new segments and passes control to them, the extended memory parity error recovery routine, and some heavily used code (CPMTR start, CPMTR return to user). The trace buffer and return stacks are used by segment linkage.

The breakpoint table is initially empty. The breakpoint (N) display and commands allow breakpoints (temporary halts) to be set and released in the operating system during system execution. CM, extended memory, and the operating system exchange package can be observed while the operating system is at a breakpoint.

The area table completely describes an extended memory system. It is read in from the extended memory image of a segmented system by the bootstrap monitor function.

The entry table contains the entry points for tables in CMR which are not directly accessible by a text symbol of the T. type. This table is used by utility LDCMR to load the extended memory system and read the date-time stamp.

The breakpoint table, area table, and entry table are detailed in appendix B.

SEGMENT LOADING

Segment Linkage

Linkage is done through the GOTO (GOTOTAB), CALL, and RETURN macros. These macros can be used only in a segment defined through the SEGDEF and ENDSEG macros.

The following macro transfers control to the entry point EPTNAME.

```
GOTO          EPTNAME
```

The following macros transfer control to the entry point referenced in a GOTOTAB macro in the position indexed by register in TABLE.

```
GOTO          TABLE,Register  
  
TABLE GOTOTAB EPTNAM2  
GOTOTAB      EPTNAM2
```

The following macro returns to the last address and segment saved by a CALL macro. The previous address and segment saved will be used by the next RETURN.

```
RETURN
```

Segment Definition

The following macro must be called immediately after the segment IDENT (first group).

```
SEGDEF          SEGMENT,mode,CM  
  
SEGMENT      Segment name.  
  
Mode         USER indicates user mode segment; otherwise, monitor mode is assumed.  
  
CM           Segment must be CM resident.
```

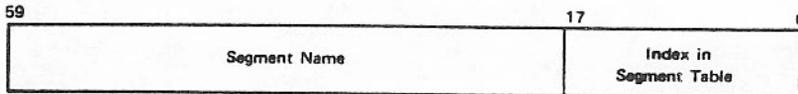
The following macro must be called immediately before the segment END.

```
ENDSEG
```

The name of the segment can also be the name of an entry point in the segment but not a tag. If it is not an entry point, the ENDSEG macro defines an entry point by that name referencing the second word of the segment.

The SEGDEF macro generates a segment header word with the tag ...REUSE.

Header format after processing by LDCMR is

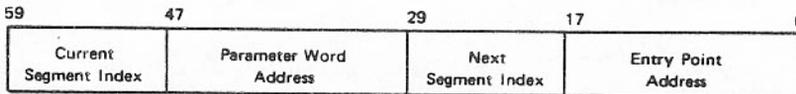


A segment should be serially reusable. If not, the segment must set its header word to zero before relinquishing control to prevent its reuse.

Parameter Word

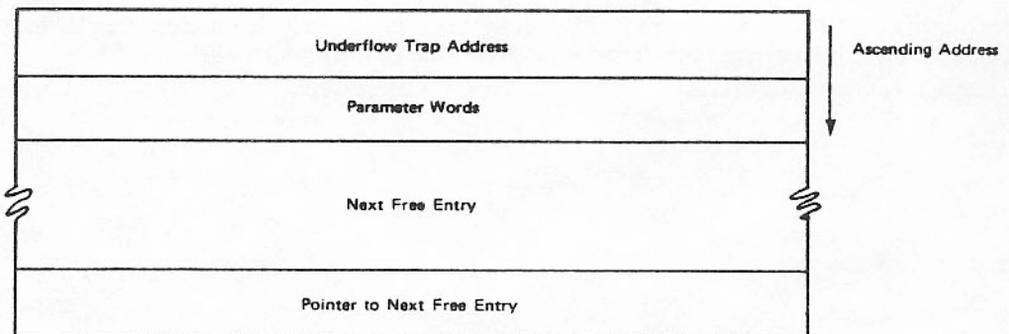
Linkage is done through a parameter word generated by the linkage macros and filled in by LDCMR. For a GOTO or a CALL, A1 is set to the address of the proper parameter word and a jump is made to one of the linkage processor entry points.

Parameter word format:

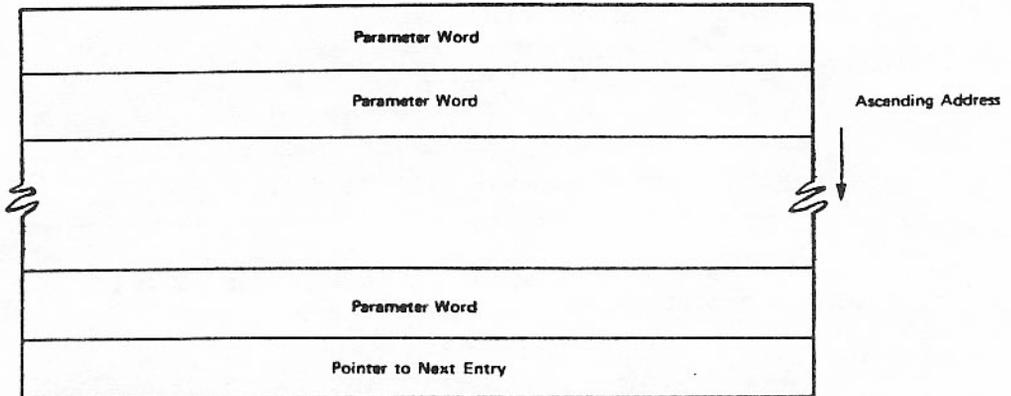


The addresses are absolute. The indexes are in the segment table. The parameter word is transformed into a return descriptor on a CALL by shifting it 30 positions and adding 1; it is stored in the proper return stack. A RETURN loads the last stored descriptor and performs a GOTO on it. If trace buffers are defined (refer to section 10), all parameter words processed are stored in the trace buffer for the current mode.

Return stack format:



Trace buffer format:



Segment loading is done by using the segment descriptor in the second word of the segment table entry for the segment. The address of the segment table entry for a segment is

$$\text{Segment table base address} + 2 * \text{index}$$

The segment table is described in appendix B.

EXTENDED MEMORY SYSTEM IMAGE

The system is written to the extended memory direct access area as an extension of control point 0 extended memory field length. Two systems, named the old system (lower system) and the new system (upper system), can coexist in extended memory. LDCMR creates extended memory system images as shown in figure 2-6.

The operating system requires approximately 40K of extended memory in the direct access area. For example, if 40K is to be available for user direct access, 80K must be allocated to the direct access partition. If LDCMR is to be used after deadstart, the direct access area must be large enough to contain two operating system CP code versions, approximately 80K, minimum.

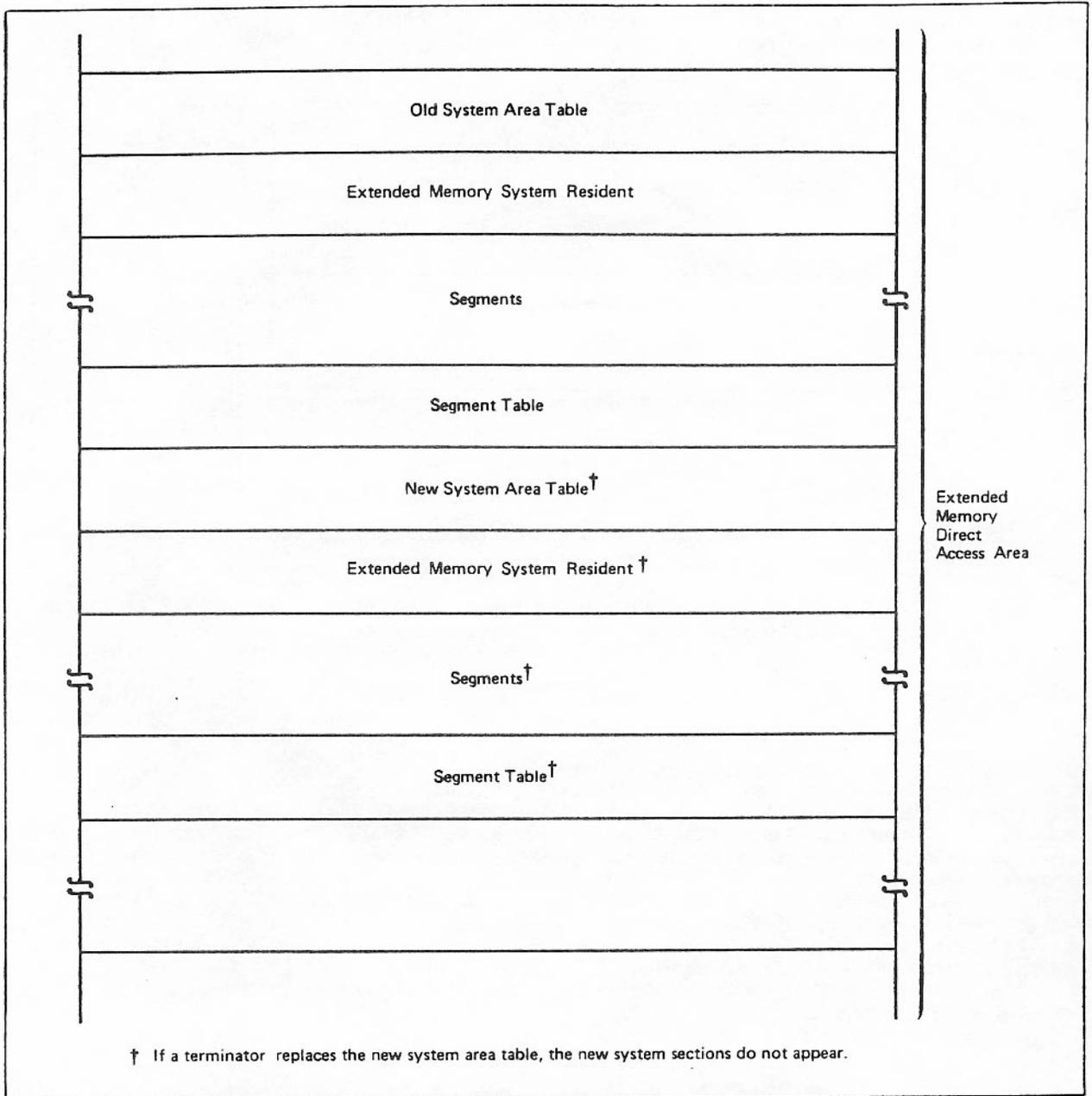


Figure 2-6. Extended Memory System Image

EXTENDED MEMORY ERROR RECOVERY

Segments are protected against extended memory errors in transmission or storage. An autocorrective code is applied to obtain correction vectors. The correction vectors and two checksums are placed at the end of the segment as shown in figure 2-7.

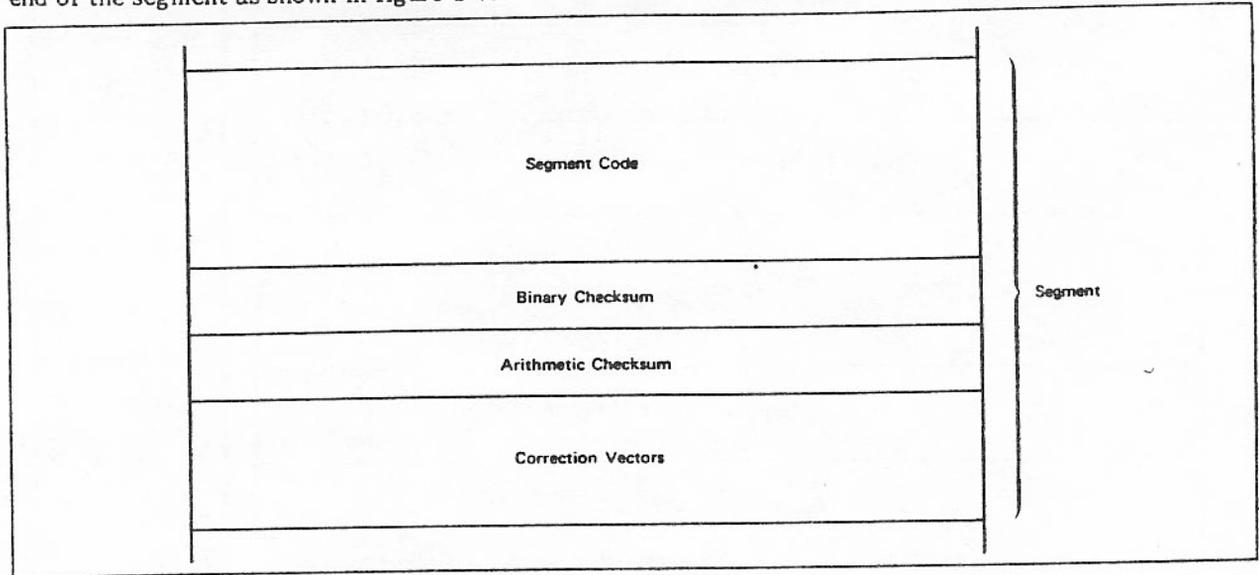
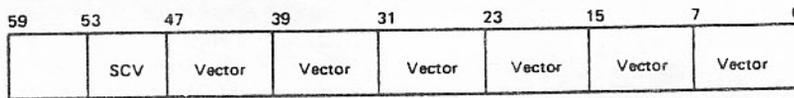


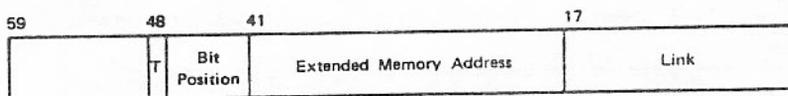
Figure 2-7. Extended Memory Segment

The code is capable of correcting a single bit error every four words. Vectors correct each four words, and a 6-bit secondary correction vector makes vector words self-correcting.



Errors are corrected only if they are detected by the extended memory parity error mechanism.† Detected errors are recorded in an error directory, then displayed in the dayfile (by a call to CEM, function 10). Error directory entries for a segment are linked to that segment's segment table entry. When an extended memory error is encountered on a segment that has error directory entries, a first correction attempt uses the information in these entries. If this fails, the error entries are released and the full correction process is reapplied. The test of a successful recovery is the comparison of the existing checksums with the checksums for the recovered segment. The system is killed if a segment cannot be satisfactorily recovered. A recovered segment is written back to extended memory, as it is possible the extended memory error is transient.

Error directory entry format:

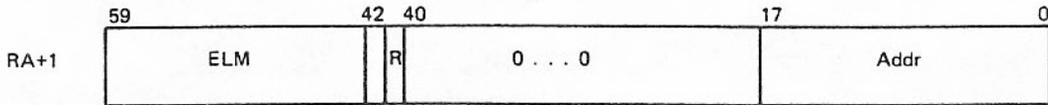


<u>T</u>	<u>Meaning</u>
0	Bit dropped.
1	Bit picked.

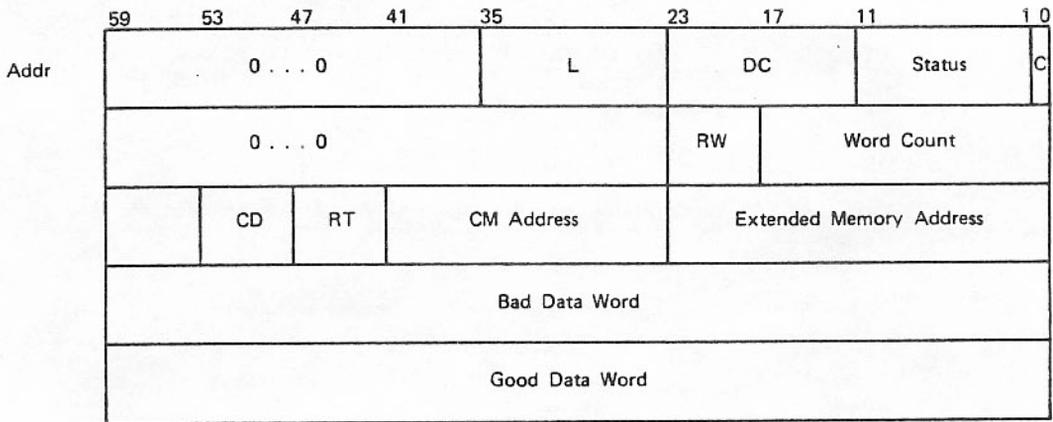
† The system extended memory read parity error recovery routine executes before the correction vectors are used. This routine attempts a standard recovery algorithm and records the results in the CERFILE.

ELM - ERROR LOG MESSAGES

The ELM RA+1 request allows a user program to log error information related to the processing of direct user access to extended memory errors. The format of the ELM RA+1 request is as follows:



The format of the parameter block is:



<u>Word</u>	<u>Bit</u>	<u>Field</u>	<u>Description</u>
0	35-24	L	Length minus one of parameter block.
	23-12	DC	Device code.
			EC Extended memory/Coupler Error.
	11-1	Status	Status flag.
			0 No errors.
			1 CERFILE message limit reached.
0		C	Complete bit. This bit must be set to zero when the request is issued.

<u>Word</u>	<u>Bit</u>	<u>Field</u>	<u>Description</u>
1	23-18	RW	Read/write flag. 1 Read. 2 Write.
	17-0	Word Count	Number of CM words transferred.
2	53-48	CD	Error packet description code. Bits 53-49 are unused. Bit 48 equals 0 if error is recovered. Bit 48 equals 1 if error is unrecovered.
	47-42	RT	Retry count. This count is the total number of recovery retry operations associated with the error being reported in this request.
	41-24	CM Address	CM First Word Address of transfer.
	23-0	Extended Memory Address	Extended Memory First Word Address of transfer.

SYSTEM CONTROL POINT

A module or group of modules that performs a specific set of functions is known as a subsystem. A subsystem has the ability to make privileged requests (reserved by subsystems) in addition to any requests that a standard control point is allowed. Typical subsystems are a data base management system or the Remote Host Facility. Each subsystem has a unique four-digit ordinal by which it is referenced.

A system control point (SCP) is any control point occupied by one of the subsystems. The term SCP can describe both the control point and the subsystem at the control point. An SCP provides a centralized location for a subsystem, allowing it to perform functions for one or more jobs at other control points. This facility provides overall reduction of CM usage. Instead of several jobs having duplicate copies of these specially privileged modules in their field lengths, only one set of these modules occupies CM. This feature also improves coordination of control and access.

A user control point (UCP) is any job or module at a control point that makes a request to an SCP. A UCP can be a batch job, INTERCOM job, multiuser job, or another SCP.

MANAGING SUBSYSTEM RESOURCES

A subsystem at a control point may receive requests irregularly, leaving it idle for long periods. Since a system control point cannot be swapped out, some other action must be taken to reduce its memory requirement while it is idle.

The idle subsystem should be organized so that it can reduce its field length as much as possible. It should specify a 1/2-second periodic recall by using RCL with a 3777₈ delay period indicated by bits 10 through 0. When it receives a call, the subsystem is started immediately.

When a call is received, the following steps should be taken.

1. Issue a MEM call to acquire the field length necessary to process the requested information.
2. Acknowledge the request by resetting RA.SSC.
3. Load the subsystem overlay(s) required to process the request.
4. Process the request.

Take these steps in the order specified. If steps 1 and 2 are taken out of sequence, a memory deadlock could occur. If the request is acknowledged before the memory is requested for the SCP, it is possible for a second UCP to make a request and have it successfully passed to the SCP. Both UCPs would then have outstanding SSC calls. If the UCPs cannot be swapped out, and the sum of the field lengths of both UCPs and the required field length of the SCP is greater than what is available, the system is deadlocked. If this occurs, one of the UCPs must be dropped to resolve the conflict.

CALLSS MACRO

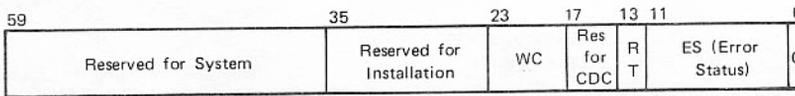
The CALLSS macro is issued by a UCP to request a particular function from a subsystem. A UCP can call more than one subsystem, either serially or concurrently. Also a UCP can make more than one call to an individual subsystem. Registers X1, X2, A1, and A6 are destroyed during execution of the macro and should not be used as parameters. The format of the macro is

```
label CALLSS ssid,addr,recall
```

label	An optional statement label.
ssid	A required subsystem code. This parameter can be a register name.
addr	Address of the parameter block for this request. This parameter is required and must be nonzero. (The parameter can be a register name.) If the address is outside the UCP field length, the UCP is aborted.
recall	If nonblank, the request is made with auto recall, and processing at the UCP is suspended until completion of the request.

The parameter block pointed to by the addr parameter of the CALLSS macro is used by the UCP to pass parameter information to the SCP. The parameter block must be at least one word long.

The first word of the parameter block is used for calling and status information. The second and subsequent words of the parameter block are used for data which is passed to the SCP by the operating system. The format of the first word is



C This bit indicates whether or not the current request has been completed. The user program must set this bit to 0 prior to executing the CALLSS macro. If the request has been completed, the operating system sets the bit to 1; otherwise, it remains set to 0.

ES The operating system sets this field to indicate the presence of an error or unavailable system condition. This field is not set (all zeros) if the RT field is 0.

Bit 1	0	Subsystem currently running.
	1	Subsystem not initiated.
Bit 2	0	Subsystem not busy (that is, it has acknowledged receipt of the last request).
	1	Subsystem busy.
Bit 3	0	Subsystem defined.
	1	Subsystem not defined.
Bits 4 and 5		Reserved.

Bits 6 through 11 Error condition other than bits 1, 2, and 3. Bits 6 through 11 can assume the following octal values.

<u>Value</u>	<u>Significance</u>
00	No other error.
01-17	Reserved for system errors.
20-67	Reserved for subsystem errors.
70-77	Reserved for installation.

RT This field is set by the user prior to making a subsystem call.

Bit 12 0 Operating system holds the current request until the subsystem is able to accept it. The UCP is placed in periodic recall and is not assigned the CPU until the request is accepted. The C bit is not set until processing of this request is complete.

1 Operating system returns control to the user if the subsystem is present but is unable to accept the user's call. In this event, bit 2 in the ES field is set to 1.

Bit 13 0 ES field is not set (with the possible exception of bit 2), and subsequent errors cause the UCP to abort.

1 Operating system sets the ES to indicate which error condition occurred and returns control to the user on nonfatal error conditions (fatal errors cause a UCP abort).

A message is issued to the UCP dayfile indicating the error condition.

When either of the RT bits is set and a condition is encountered that causes any of the bits in the ES field to be set, the operating system sets the C field and considers the operation complete. Therefore, it is necessary to reissue the CALLSS macro.

WC Some subsystems require the user to specify the length of the parameter list. WC is the number of words (excluding the first word) to be passed with the request. The maximum is determined by the subsystem but may not exceed 77g.

The remaining bits in this word are reserved.

If a call is issued with auto recall, the user's program is not restarted until the C field has been set.

A subsystem can call another subsystem as long as this does not result in a circular chain reaction. An SCP cannot make a call to itself, unless bit 12 of the RT field is set. An attempt to call itself without bit 12 set can result in a subsystem hang.

SYSTEM CONTROL POINT INTERFACES

Three types of system communication interfaces are unique to subsystems.

- Subsystem notification to the operating system that it is entering active status.
- Subsystem request acknowledgement.

- Special requests to the operating system allowing the subsystem to have access to the UCP field length, control and forward accounting data, obtain UCP swap and error status, transmit dayfile messages and error conditions to the UCP, and exit from active status.

Subsystems should use a symbolic reference for SCP locations and operations (for example, refer to RA.SSID rather than RA+50 and SF.READ rather than 10).

Requesting Active Status

The operating system does not recognize an SCP until a subsystem is loaded and ready to enter SCP status. To notify the operating system that it is entering active status, the subsystem puts its name and code in word RA.SSID (RA+50). RA.SSID is used by the operating system to identify the subsystem and must be maintained at all times. A CALLSS macro using automatic recall is then executed with the ssid field equal to SS.SYS.

Prior to assigning SCP status to the requesting control point, the operating system ensures that the following conditions are true:

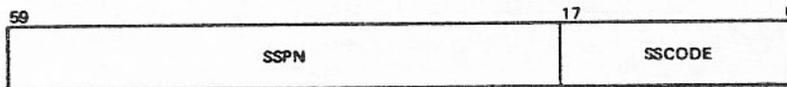
- Request is made with recall.
- Program is called by a system origin job.
- Subsystem name in RA.SSID matches that in the corresponding subsystem control table entry.
- The same subsystem is not in SCP status at another control point and the maximum number of SCPs has not been reached.

If any of the preceding conditions are not met, an appropriate diagnostic message is entered in the dayfile, and the job is aborted. Once the operating system has assigned SCP status to the subsystem, the C field is set to 1.

If RT bit 13 is set, failure to meet either of the last two tests will not abort the job but will set the following codes in bits 11 through 6 of the ES field.

- 04 Another control point has SCP status for this subsystem.
- 05 Nine control points have SCP status.

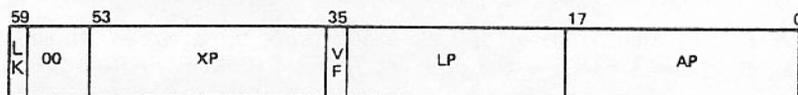
The format of SCP word RA.SSID is



- SSPN Subsystem program name.
- SSCODE Identification code (SS.XXX).

Subsystem Request Acknowledgement

A word in the SCP field length, RA.SSC (RA+51), is set by the subsystem and used as a pointer to indicate where incoming requests from the UCP are put. The format of RA.SSC is



- AP Address of the UCP parameter block.
- LP Length of the request parameter block.
- VF Variable move flag.
- XP UCP exchange package address.
- LK Interlock bit.

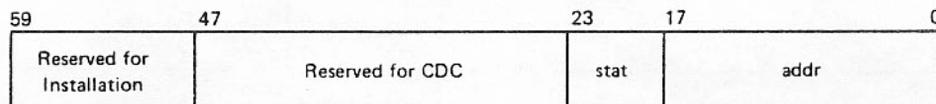
LK is set by the operating system when a request has been placed in the parameter area. It is cleared by the subsystem to acknowledge that the request has been received. When the bit is cleared, AP points to a parameter area in which the subsystem is prepared to receive the next request. After clearing LK, the subsystem should not attempt to rewrite word RA.SSC until the next request has been received.

If it is necessary to force a request, it is possible for a subsystem to call itself if bit 12 of the RT field in the first word of the parameter block is set. Attempting a call to itself without this bit being set can result in a subsystem hang.

If LK is set at the time of the request, one of the following actions is performed, depending on bit 12 of the RT field.

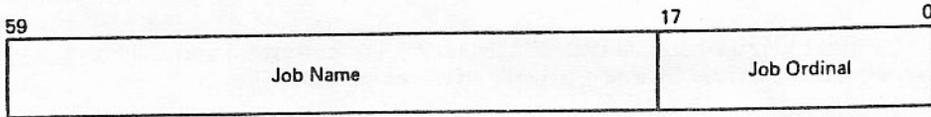
- Return control to the UCP and indicate a busy status in the ES field (RT=1).
- Hold the request and periodically attempt to give it to the SCP (RT=0).

The word at the address AP has the following format:



- addr Same as the address in the CALLSS macro. This is a relative address within the UCP field length.
- stat Status values.
 - 0 Call is from a user.
 - 1 Normal termination.
 - 2 Error termination of UCP.

AP+1 is the job identifier whose format is



AP+2 through AP+LP-1 (minimum value of LP is 2) contains LP-2 words which are from the UCP parameter list starting at addr. The UCP parameter list address is taken from the CALLSS macro call.

If the VF bit is set, the length of the move is determined by the WC field in the first word of the UCP parameter block. If WC+3 is greater than LP, only LP words are moved in.

If XP is nonzero, the UCP exchange package is stored in the 16 words starting at XP.

Special Subsystem Requests to the Operating System

A subsystem at the SCP can make special requests of the operating system. These special requests, called subsystem functions (SFCALLs), are allowable from an SCP only. If an SFCALL is issued from a nonsystem control point, the control point is aborted with the error message PP CALL ERROR.

The SFCALL macro call has the following format:

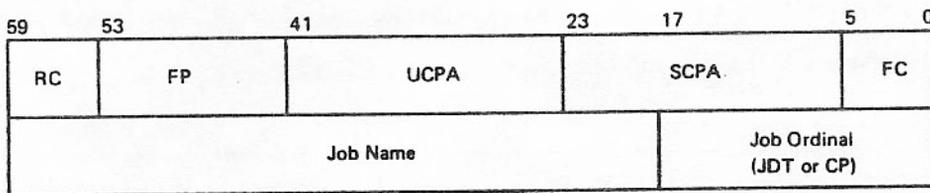
label SFCALL addr,recall

label An optional statement label.

addr Address of an SFCALL parameter word pair (refer to following SFCALL format). This parameter can be a register.

recall If nonblank, the job is put in automatic recall. Although this option is allowed, its use by a subsystem is discouraged.

A typical format of the SFCALL parameter word pair is:



RC Return code.

FP Function parameter.

UCPA Relative address within the UCP.

SCPA Relative address within the SCP.

FC Function code (an even number, incremented by one when the function has been completed).

If a parameter error prevents the processing of the function, RC will be set to a value in the range 40 through 77.

The following list of SFCALL return codes (RC) gives all codes which have been defined. Not all of these codes apply to this operating system. Codes 40 through 77 indicate that the function was not processed.

<u>Return Code</u>	<u>Meaning</u>
00	No error encountered.
01 through 33	Trivial errors (reserved for Control Data).
34 through 37	Trivial errors (reserved for installations).
40	At least one error encountered in list.
41	Job identifier invalid (NOS/BE considers all IDs to be valid; returns 45 if ID is not found).
42	SCPA not within the subsystem FL (code not returned by NOS/BE; the system aborts the SCP and issues a message to the dayfile).
43	UCPA not within the UCP FL.
44	User job swapped out.
45	User job not in system.
46	Reserved for Control Data.
47	Unknown function code.
50 through 56	Reserved for Control Data.
57	Connection previously established.
60	Connection rejected.
61	Connection not previously established.
62	Word transfer too long. [†]
63	UCP not established with subsystem. [†]
64	Subsystem not established with receiver. [†]
65	Attempt to set illegal error flag. [†]
66	Illegal dayfile processing flag. [†]
67 through 73	Reserved for Control Data.
74 through 77	Reserved for installations.

Possible return codes for each SFCALL function are shown in table 2-2.

[†] Not checked under NOS/BE.

TABLE 2-2. SFCALL RETURN CODES

Function	Return Code													
	40	41	42	43	44	45	57	60	61	62	63	64	65	66
SF.REGR			X		X	X								
SF.TIME			X			†								
SF.ENDT			X	X	X	X								
SF.READ			X	X	X	X								
SF.STAT					X	X								
SF.WRIT			X	X	X	X								
SF.EXIT						†								
SF.SLTC					X	X	X	X						
SF.CLTC					X	X			X					
SF.SWPO					X	X								
SF.SWPI					††	X								
SF.LIST	X		X		X	X								
SF.RERN					X	X								

† Return code of 45 may be returned if the user ID word is not zero.
 †† Return code of 44 is returned if job is locked out.

The function codes (octal) used with SFCALL are as follows:

<u>Function</u>	<u>Code</u>	<u>Description</u>
SF.REGR	02	Place message into the UCP dayfile and/or abort the UCP.
SF.TIME	04	Obtain accounting data for SCP.
SF.ENDT	06	Indicate end of task to UCP.
SF.READ	10	Read from UCP field length.
SF.WRIT	14	Write to UCP field length.
SF.STAT	12	Request status of UCP.
SF.EXIT	16	Exit from SCP status.
SF.SWPO	24	Indicate UCP as candidate for swap-out.
SF.SWPI	26	Request swap-in of UCP.
SF.SLTC	30	Set the long-term connection indicator or set the end-of-job connection indicator.

<u>Function</u>	<u>Code</u>	<u>Description</u>
SF.CLTC	32	Clear the long-term connection indicator or clear the end-of-job connection indicator.
SF.LIST	34	Process of list of SF.xxxx functions.
SF.RERN	36	Set/clear rerun status of UCP.
SF.INS1-4	70,72,74,76	Reserved for installations.

SF.REGR - Regrets

The SF.REGR function code places a message of up to 80 characters into the dayfile of the UCP and/or aborts the UCP. It has the following format:

59	53	41	23	17	5	0
RC	FP	UCPA	SCPA		02	
Job Name					Job Ordinal	

UCPA = 0 Do not abort the UCP.

UCPA = 0 Abort the UCP. The F.ERPP error flag is set at the UCP.

SCPA = 0 No message.

SCPA = 0 Address of a message that is to be sent to the UCP dayfile.

FP Dayfile processing flags. The following is a list of the symbolic values and their meanings as defined for the FP field.

<u>Value</u>	<u>Significance</u>
F.SYCP	Send message to system dayfile and control point dayfile.
F.NMSN	Do not send message to control point dayfile.
F.JNMN	Do not send message to control point dayfile; job name is in message.
F.CPON	Send message to control point dayfile only.
F.ACFN	Accounting message to system dayfile only.
F.AJNN	Accounting message to system dayfile only; job name is in message.
F.ERLN	Send message to error file only.
F.EJNN	Send message to error file only; job name is in message.

SF.TIME - Accounting

The SF.TIME function code allows the SCP to obtain the accumulated accounting totals at its own control point. The format is:

59	23	17	5	0
Unused		SCPA		04
Job Name			Job Ordinal	

SCPA Relative address within the SCP of a word block for accounting data.

Job name Set equal to zero.

Job ordinal Set equal to zero.

The accounting totals are returned to SCPA through SCPA+5.

SCPA+0	CPA time.
SCPA+1	CPB time.
SCPA+2	I/O time.
SCPA+3	CM field length.
SCPA+4	Extended memory field length.
SCPA+5	PP time.

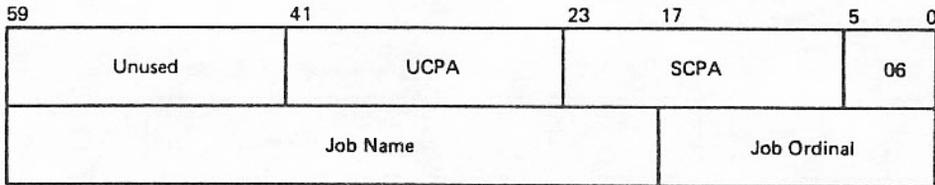
The symbol L.SACT must be used by all subsystems to define the work block lengths (for example, BSS L.SACT). It allows an installation to add a specially defined area to the word block by modifying the symbol and reassembling the subsystems using this symbol. The operating system is not responsible for setting or clearing the installation area.

When an installation defines one or more words for installation usage, these words must be located by using the last address of each word block (first + L.SACT-1) and referencing installation words backwards from this address.

Since the data delivered to this area varies among operating systems, a module must be provided for each operating system to process this area. Multitask users can use the data in this area to charge a particular UCP for the SCP resources used in processing the UCP's task. The resource data sent to the SCP can be the accumulated totals. The subsystem at the SCP has the responsibility for storing the previous totals of used resources and for calculating the differences.

SF.ENDT - Subsystem Task Complete

The SF.ENDT function informs the operating system and the UCP that the subsystem task has been completed, and allows the SCP to distribute the accumulated resource costs back to the UCP. Its format is



UCPA Relative address within the UCP of the request status word of the task being performed.

- > 0 Set bit 0 of the word at UCPA. (Restart UCP if auto recall was selected.) Reduce the request count by one.
- = 0 Do not set complete bit (bit 0). Reduce the request count by one.
- = -1 Do not set complete bit. Reduce activity count to zero, no matter how many requests are outstanding. (This activity count is the number of requests from the UCP JOBID to this subsystem.) Clear the long-term connection bit if it is set.
- < -1 Return error 43.

SCPA Relative address within the SCP of a word block of accounting data. The content is the same as specified by the SF.TIME function.

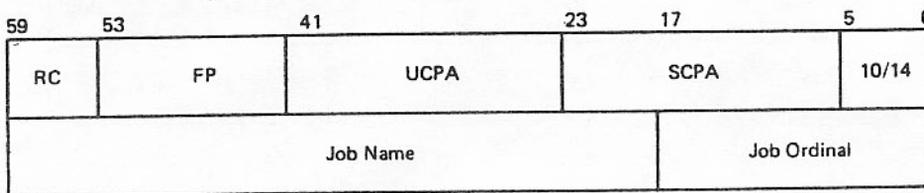
If SCPA is not 0, computer resource usage is based upon the data provided at SCPA. The resource accumulators at the UCP are incremented with this computed resource data. The core seconds that are computed and added to the UCP are based on the SCP field length at SCPA+3. An SCP that is multitasking should not charge a single user for the full SCP field length, but use a somewhat smaller value in this field. Each of these fields can be passed to the SF.ENDT function exactly as returned to the SCP by the SF.TIME function, or they can be adjusted as required by the SCP. At a later time, the normal computation of the core seconds at the UCP includes the CP and I/O time of the SCP. The result is that the effective field length of CP and I/O time used by the SCP is the sum of the UCP and SCP field length.

SF.READ - Read from the UCP Field Length

SF.WRIT - Write to UCP Field Length

SF.READ moves FP words from the UCPA to the SCPA, whereas SF.WRIT moves FP words from the SCPA to the UCPA.

The format of SF.READ and SF.WRIT is as follows:



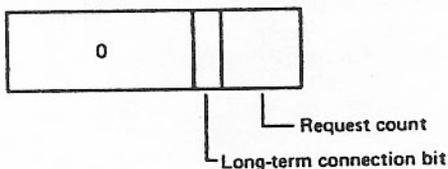
If RC is 428, the SCPA or SCPA+FP is outside the SCP field length. The SCP is aborted. If RC is 438, the UCPA or UCPA+FP is outside the UCP field length.

Transfers of large blocks of data between the UCP and SCP should be avoided because they may cause significant system response delays. Block transfers take place when the UCP calls an SCP or when an SCP makes an SF.READ or SF.WRIT request to read or write data to the UCP field length. Generally, no block transfer should attempt to transfer more than 64 words in one request or call.

SF.STAT - Status

The SCP uses the SF.STAT function to request the current status of the user job. The RC and FP fields are used for reply.

If RC is 0, the UCP is not swapped out or being swapped out. The state of the connection indicator is returned in FP as follows:



The UCPA and SCPA fields are not used but should be set to 0 in case optional uses are assigned in the future.

SF.EXIT - Exit from SCP Status

SF.EXIT removes the SCP from SCP status. The following actions should be taken in the order specified.

1. Make an SF.EXIT call.
2. Clear RA.SSID.
3. Issue ENDRUN or ABORT.

SF.SWPO - Swap Out

SF.SWPI - Swap In

SF.SLTC - Set Long-Term Connection or Set End-of-Job Connection

SF.CLTC - Clear Long-Term Connection or Clear End-of-Job Connection

These functions, with the SF.ENDT function described previously, control the setting and clearing of bits in the inter-job connection table (T.IJCT). The T.IJCT is a part of the subsystem control table (T.SSCT) used to define and control SCP status.

Each user job has a corresponding word in the T.IJCT that contains nine connection control fields. When a control point is assigned SCP status, it is assigned one of the nine fields in each T.IJCT word. The connection control field contains the following connection indicators.

WAIT RESPONSE COUNT (alternately called request count)

This 3-bit field contains the number of unanswered requests submitted to the SCP by the UCP. The count is incremented each time a CALLSS request is passed to the SCP and decremented by an SF.ENDT function with UCPA=0.

LONG TERM CONNECTION

This bit is set by SF.SLTC and cleared by SF.CLTC when they are called with FP=0. An SCP sets this bit to be notified when the UCP is terminating because of either an ENDRUN or any abnormal termination. The method of notification is described under End Processing for UCPs in this section.

END-OF-JOB CONNECTION

This bit is set by SF.SLTC and cleared by SF.CLTC when they are called with FP=1. An SCP sets this bit to be notified when the UCP is at end-of-job. The method of notification is described under End Processing for UCPs in this section.

SWAP OUT REQUESTED BIT

This bit is set by the SF.SWPO and cleared by the SF.SWPI. If the swap out bit is set in any one of the connection control fields for a user control point, that job is treated as if its CM time quantum has expired. This causes it to be swapped out unless it is locked in or there are no other jobs in the CM queue that could use the memory that would be released by swapping it out.

While a job is swapped out, it is not aged as long as any of its connection control field swap-out bits remain set. When an SF.SWPI function causes the swap-out bit to be cleared, the UCP is artificially aged so that it can be swapped in again promptly.

Subsystems should use the SF.SWPO and SF.SWPI carefully, because misuse can cause unnecessary and inefficient swapping among user jobs.

Neither the wait response count nor the long term connection is considered when a job is selected to be swapped out. If, however, any of the connection control fields has a nonzero wait response count and the swap-out bit is not set, the swapping of that job will be delayed for over 1 second, if necessary, to give the subsystem a chance to complete the request before the user job is swapped out.

An SF.SWPI function is not set until the swap-in has been completed. If the swap-out request bit is set for a different SCP, the UCP remains a good candidate to be swapped out again. The existence of a wait response can, however, guarantee that it will be held at a control point for at least 1 second.

SF.LIST - Presents a List of SF.xxxx Functions

The multiple request capability is invoked through the use of the function code SF.LIST. The format of the SFCALL parameter word pair in this case is

59	53	41	23	5	0
RC	FP	0	SCPA	34	
Job Identifier					

- RC Return code.
- FP Number of entries in the list.
- SCPA First word address of the contiguous parameter list.

The entries created by the SF.LIST function are each one word in length. The entry consists of the first word as described for each function. Only one UCP may be addressed for each list processed and this is the UCP indicated in the SF.LIST word pair. An SF.LIST function may not be included as a member of a list.

When the FC field is set complete by the operating system, the RC and FP fields must be examined to determine the action to be taken. If FP is 0, the entire list has been processed by the operating system. If FP is not 0, processing of the list was abandoned, and FP contains the number of entries remaining in the list. SCPA is set to the address of the first entry in the remaining list. The subsystem reissues the SF.LIST call by resetting the FC field and executing the SFCALL macro until FP equals 0.

The operating system will set the RC field only if an error is detected. Multiple issues of the same SF.LIST request (until FC is set complete and FP is 0) accumulate error returns whether or not the entire list is processed on one SFCALL.

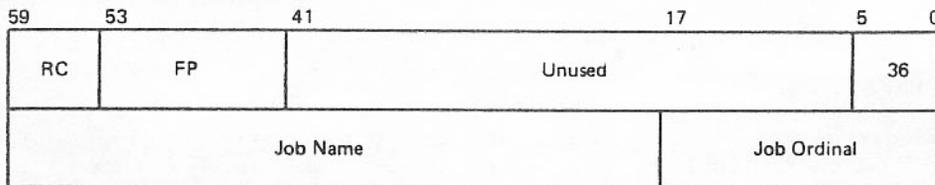
The user of SF.LIST should consider the following notes and special conditions.

- The operating system aborts the SCP during SF.LIST processing if a fatal error occurs in the SF.LIST or in any member of the list.
- The detailed error conditions must be determined by examining the individual list entries whenever the SF.LIST RC field equals 40g.
- The individual functions are handled in the same way whether or not the list mode is enabled.
- Error status 42g when SCPA is illegal means none of the list entries has been processed. This check is made prior to initiating the list process. Illegal SCPA is a fatal error.
- Error status 42g when FP is 0 means that none of the list entries has been processed on this call. If the subsystem handles the FP equal to 0 condition improperly, the entire list may have been processed prior to the subsystem abort. Illegal SCPA is a fatal error.
- List entries are processed sequentially by the operating system and entries detected as erroneous for any reason are considered completed. It is expected that in most cases the entire list will be processed on one SFCALL. The option of abandoning the list allows the operating system to take corrective action if it decides that either the length of the list, the complexity of the processing, or other reasons have possibly caused a long uninterruptable interval.

- If the SCP is aborted due to an error in one of the list entries other than the SF.LIST, RC equals 408, SCPA and FP are updated, and FC is set complete. The proper return status is also placed in the offending list entry.
- The functions SF.REGR, SF.RERN, and SF.EXIT are performed by a PP program instead of CPMTR. When any of these are included in a list, processing is transferred to the PP for that function. When it is completed, the list is abandoned rather than attempting to transfer processing back to CPMTR. To process the remainder of the list, it is necessary to reissue the SF.LIST function.

SF.RERN - Set/Clear Rerun Status

The SF.RERN function code sets or clears the status bit in the UCP area and input file's permanent file catalog (PFC) entry, which determines whether the job can be rerun. This bit is sometimes referred to as the no-rerun status bit. SF.RERN has the following format.



RC **Return code.**

FP = 0 **Sets status such that job can be rerun (clears no-rerun status).**

FP = 1 **Sets status such that job cannot be rerun (sets no-rerun status).**

A user may wish to set the no-rerun status bit if he is performing operations that could destroy the validity of his files if done more than once. An example is the updating of a data base. If an error occurs and the job is stopped before the data base changes are complete, the user may not want to risk rerunning the job and making incorrect changes to a data base that is already partially modified.

END PROCESSING FOR UCPS

If a program running at a UCP is terminated while there is a long-term connection or a nonzero wait response count, or if the program is at end-of-job and the end-of-job connection bit is set, the SCP is notified in the form of a two-word call to the SCP. The stat field in the word at AP+0 identifies the termination notification. If the UCP was terminated by an ENDRUN, the stat field contains the value 1. In the event of an error condition, the field contains the value 2. All normal calls from the user contain a 0. If the UCP aborts and reprieves, the SCP is not notified. If the UCP is aborted by an SF.REGR function, the SCP is notified as for any other error condition.

When a subsystem receives termination notification, it should complete all requests as quickly as possible, issuing an SF.ENDT for each wait response and an SF.CLTC if the long-term connection is set. Until this is done, the termination notification is repeated every 2 seconds and the message CONNECTED TO jobname is flashed at the UCP on the B display (jobname is the SCP job name).

If the subsystem is unable to complete the outstanding requests, it should issue an SF.ENDT to that UCP with a -1 (777776g) in the UCPA field to unconditionally release the UCP.

If the CONNECTED TO jobname message continues to flash for an extended period of time, it can be assumed that the SCP is not functioning correctly. The SCP and all of its connected UCPs can be terminated by an operator drop on the SCP.

A UCP must not call SWT when there is a long-term connection or a nonzero wait count.

NORMAL SCP TERMINATION

The following steps should be taken to terminate execution of a subsystem.

1. Stop accepting any requests.
2. Complete processing any requests already received.
3. Issue an SF.EXIT.
4. Issue an ENDRUN.

To force a subsystem to stop accepting requests, the user can have the subsystem send a dummy request to itself. The RT bit (bit 12) must be set whenever a CALLSS is issued. If bit 2 of the ES field is set, the operating system has delivered a request from another UCP, which must be processed along with any other uncompleted requests before the SF.EXIT is issued. The operating system sets the LK bit in RA.SSC and does not send any more requests as long as LK is not cleared.

An SCP must not call SWT.

ABNORMAL SCP TERMINATION

A subsystem should make use of the RECOVR capability (refer to NOS/BE Reference Manual) so that the subsystem will be reprieved if an error condition occurs. During reprieve processing, it can attempt to complete all outstanding requests, so as to cause as little user interruption as possible. The subsystem retains SCP status during reprieve processing.

If a subsystem has completed reprieve processing, or was not reprieved and attempts to terminate without issuing the SF.EXIT, the operating system performs the SF.EXIT and issues the message SYS CTL PT STATUS CANCELLED.

When performing the SF.EXIT function, the operating system determines if the SCP still has any long-term connection or active wait-response counts with a UCP. If there are any, the following actions will be performed by the operating system.

1. SCP is aborted with the message EXIT - WITH CONNECTIONS.
2. The wait-response counts and long-term connections are nullified.
3. The message subsystem name ENDED BY SYSTEM is sent to the UCP dayfile.
4. UCP is rerun or, if rerun is not allowed, aborted.
5. UCP can perform reprieve processing.

HOW TO DEFINE A SUBSYSTEM

The CMR symbol N.SBSYS determines if T.SSCT is assembled in CMR. The default value is 4. N.SBSYS is the maximum number of subsystems that may be defined; it must be at least 4. Installations that want to define their own subsystems should use position ordinals 10g through 17g.

The SSCT macro is provided for defining subsystems. The macro has the following parameters:

SSPN	Subsystem program name.
SSCODE	Position ordinal for the subsystem.
PUF	Permit user files.

SSCODE is the unique code that identifies the subsystem and determines the position within T.SSCT at which the defining entry is assembled. Its value may not be greater than N.SBSYS. This is the same code that the subsystem uses in word RA.SSID when it requests system control point status. It is also used as the ssid for the CALLSS macro to identify which subsystem is to be called.

If the PUF parameter is omitted, the subsystem program must be called by a system origin job or it will not be granted system control point status. The PUF parameter should only be used during subsystem development.

System control point macros are contained in SSYTEXT.

SETMFL MACRO

Several programs, such as CDCS, that execute as system origin jobs with SCP status, require a maximum field length similar to that specified by the CM parameter on a batch origin job statement. The SETMFL macro sets the maximum field length for system origin jobs. The format of the macro is

label	SETMFL	maxfl
label		An optional statement label.
maxfl		Maximum field length; maxfl can be a constant, symbol, or register name.

The SETMFL macro is defined in the system text SSYTEXT. When SETMFL executes, it makes an RA+1 system request with auto recall set.

Under certain conditions, the maximum field length (MFL) established may differ from the field length requested. If the requested field length is 0, less than 0, or larger than 400000g, the system default field length (set by IP.SFL) is used. If the requested field length exceeds the system MFL (set by IP.MFL), the system MFL is used.

The SETMFL macro should be executed only once in a system origin job. Subsequent calls to SETMFL or any calls to SETMFL from a nonsystem origin job are ignored unless the requested field length exceeds the current MFL. In that case, the job is aborted.

PROGRAMMING TIPS

A system control point runs at a high CPU priority level. When it receives a request from a user, the CPU is immediately assigned to process the request. In most cases, the CPU is not reassigned to any of the lower priority jobs until the SCP releases it by issuing an RCL. If the SCP uses the CPU inefficiently, monopolizing it for long periods of time, the throughput of the whole system will suffer. When subsystem action is blocked, waiting for actions from other parts of the system, the CPU must be relinquished.

It is also important that the subsystem be ready to accept and process requests from other users. Before issuing an RCL, it should be certain that the LK bit is not set, indicating that it is ready to receive a new request. The auto recall bit should not be used in any RA+1 call from an SCP because an auto recall status would prevent it from responding to new requests as they come along.

PERIPHERAL PROCESSOR ORGANIZATION

When the operating system is loaded into the computer at deadstart time, MTR and DSD are loaded into PP0 and PP1, respectively, where they reside permanently. All PPs in the system contain a group of permanently assigned storage locations called the PP direct cells. Refer to table 3-1 for PP direct cell assignment. The contents of the direct cells are not guaranteed from one PP overlay to the next. Each PP, except PP0 and PP1, contains a copy of the PP resident program, which handles common service functions for the PP programs that are loaded into the unassigned pool processors.

TABLE 3-1. PP DIRECT CELL ASSIGNMENT

Location (Octal)	Identifier	Function
0	D.Z0	Temporary storage.
1	D.Z1	Temporary storage.
2	D.Z2	Temporary storage.
3	D.Z3	Temporary storage.
4	D.Z4	Temporary storage.
5	D.Z5	Temporary storage.
6	D.Z6	Temporary storage.
7	D.Z7	Temporary storage.
10	D.T0	Temporary storage.
11	D.T1	Temporary storage.
12	D.T2	} PP output register buffer
13	D.T3	
14	D.T4	
15	D.T5	
16	D.T6	Temporary storage.
17	D.T7	Temporary storage.

NOTE

Most temporary storage cells are used and often changed by PP resident routines. Therefore, stored data may be different after a PP resident call.

TABLE 3-1. PP DIRECT CELL ASSIGNMENT (Contd)

Location (Octal)	Identifier	Function
20	D.FNT/D.TW0	D.FNT through D.FNT+11 (octal) contains 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 contains the EST entry in process. D.JPAR contains a job parameter word.
37	D.DTS/D.JFL, D.TH7	D.DTS contains the device type code in the left 6 bits and the allocation type code in the right 6 bits. D.JFL contains the CM field length requirement returned to the caller by 2TJ.
40	D.BA/D.FR0	D.BA through D.BA+4 (buffer address) contains the first word of the FET.
45	D.JECS/D.FR5	Extended memory field length returned by 2TJ to caller.
46	D.JPR/D.FR6	Computed job priority returned to caller by 2TJ.
47	D.JTL/D.FR7	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) for the control point to which the PP is attached.
56	D.FL/D.FF6	CM field length divided by 100 (octal) for the job at the control point to which the PP is attached.
57	D.FA/D.FF7	Address of the second word of the 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 of word OUT in circular I/O buffer.
66	D.LIMIT/D.SX6	This and next cell contain 18-bit address of word LIMIT in circular I/O buffer.

TABLE 3-1. PP DIRECT CELL ASSIGNMENT (Contd)

Location (Octal)	Identifier	Function
70	D.PPONE/D.SV0	Can be set to constant value +1.
71	D.HN/D.SV1	Can be set to constant value +100 (octal).
72	D.TH/D.SV2	Can be set to constant value +1000 (octal).
73	D.TR/D.SV3	Can be set 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	R.FAF	Field access flag.
101		Reserved.
102		Reserved.

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 can load additional overlays to help complete its function; on completion, the program can be overlaid by another transient program loaded by PP resident to perform another, often unrelated, function. A typical layout of a pool PP loaded with a transient PP program and an overlay is shown in figure 3-1. The PP direct cells occupy locations 0 to 77; the PP resident is loaded starting at location 103 to approximately 773. The remainder of the PP is occupied by a PP transient program.

† These locations must always contain the indicated data whenever a PP resident routine is called.

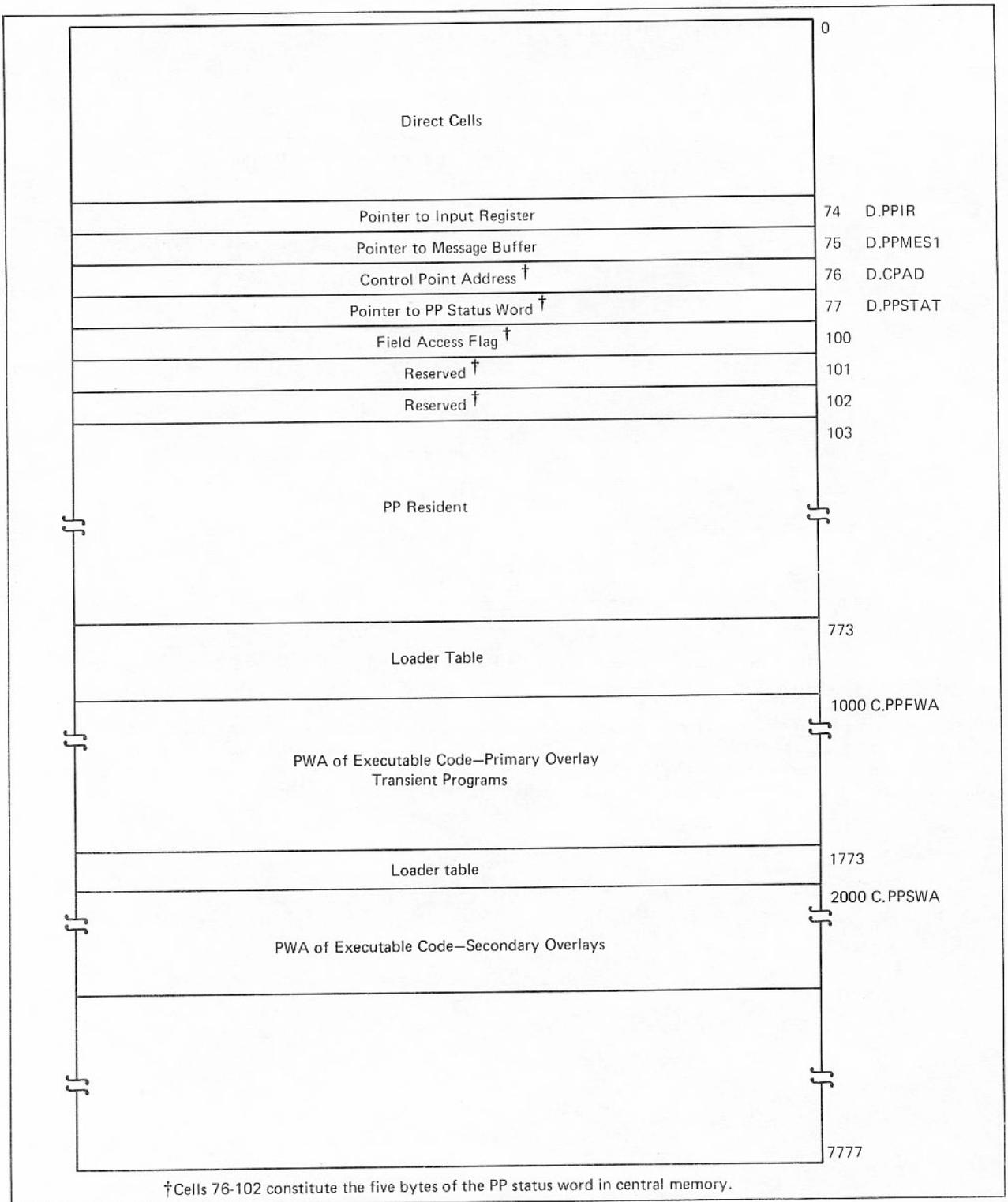


Figure 3-1. Pool PP Layout

PP COMMUNICATIONS

For each pool PP, CMR has an area used for communication between the PP monitor (PPMTR) and the PP (refer to figure 2-4). Each PP communications area contains a PP input register and a PP output register (each one-CM-word long), plus a six-CM-word message buffer.

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 loads 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 inserts the requesting program's control point number into bits 39 through 36 of the word and clears bit 41. Bits 35 through 0 appear in the input register exactly as they did in RA+1 of the requesting CP program.

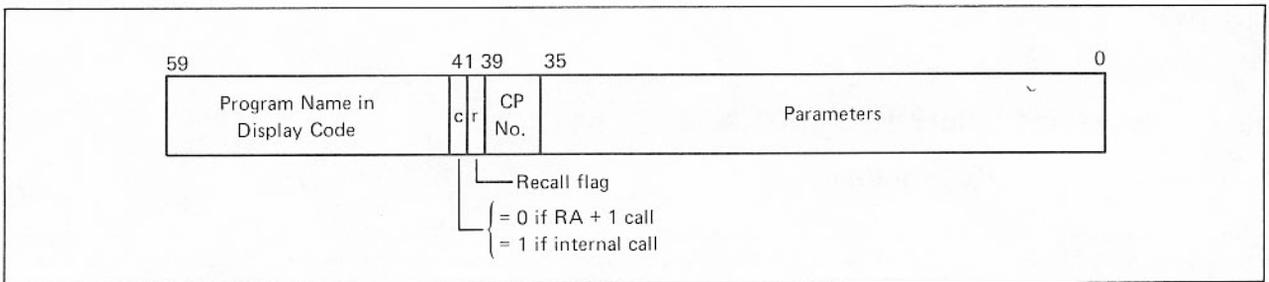


Figure 3-2. PP Input Register

The PP resident in each PP constantly scans its own input register. When it becomes nonzero, the PP resident issues the M.PPLIB CPMTR request with the program name and load address. CPMTR searches for the program in the PP library and establishes the mechanism with which the PP will load the code. The PP loads the transient program at location 773, stores the address of the control point area to which the PP is assigned in direct cell D.CPAD, and transfers control to location 1000 to start execution. If the transient program 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 MTR 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 MTR to drop the PP. MTR clears the PP input register and records 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.

To make a monitor request, a PP program places a message into its 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, signaling that the monitor has processed the request.

PP RESIDENT

The PP resident program performs the following 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.
- Controls PP access to the assigned control point's field length.

The PP resident consists of a series of routines, each of which performs a specific function (figure 3-3). These resident routines are used by the transient and overlay programs as required. R.OVLJ and all other PP resident routines, except R.IDLE, destroy temporary cells 0 through 17. R.IDLE destroys direct cells 20 through 22 and some of the temporary storage cells. The names, locations, calling sequences, and functions of the routines follow.

R.IDLE — PP RESIDENT IDLE LOOP

Calling sequence: LJM R.IDLE

In the idle loop, PP resident first clears the field access flag (R.FAF), then continually scans its input register for an assigned task. When R.IDLE finds an assigned task, the control point address (D.CPAD) is set, and R.OVLJ is entered to load the overlay.

R.OVLJ — PRIMARY OVERLAY (TRANSIENT PROGRAM) LOADER

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

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

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.

R.TAFL — TERMINATE CONTROL POINT FIELD 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.

59)	0
	Field Access Flag	R.FAF
	PP Resident Ldle Loop	R.IDLE
	Load Primary PP Overlay	R.OVLJ
	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
	Load PP Overlay	R.OVL
	Access Request Stack Entry	R.EREQS
	Transmit Dayfile Message	R.DFM
	Transmit Data To/From PP	R.WRITEP R.READP
	Read/Write Logic (Disk)	R.RWP
	Read/Write Segments (Non-disk I/O)	

Figure 3-3. PP Resident Routines

R.TFL — TEST FIELD LENGTH

Calling sequence: Load A register with 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 zero in the A register; otherwise, R.TFL exits with the resultant absolute CM address (RA + relative address) in the A register. A call to R.RAFL sets a flag which enables R.TFL to return a reliable result. R.TAFL clears the flag. Therefore, the transient and its overlays must not call R.TFL until R.RAFL has been called.

R.MTR — PROCESS MONITOR FUNCTION

Calling sequence: Store function parameters in D.T1 to D.T4.
Load A register with function code.
RJM R.MTR

This routine 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.

R.PROCES is the same as R.MTR.

R.WAIT — PP WAIT LOOP

Calling sequence: RJM R.WAIT

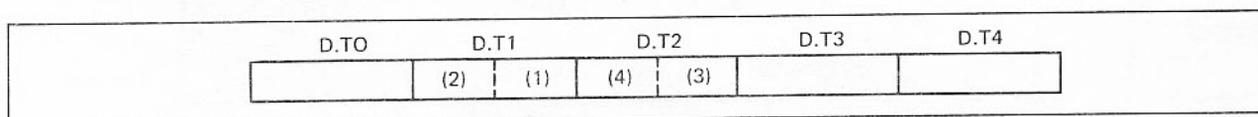
This routine determines if the monitor function is for CPMTR or MTR. If the function is for CPMTR, the PP input register address is written into T.PPID and a monitor exchange jump is executed. If the function is for MTR, the input register address is written in T.PPIP.

After either action, R.WAIT idles until byte zero of the output register is cleared. If the field access flag (R.FAF) is set, R.WAIT pauses for relocation via calls to R.RAFL.

R.RCH — REQUEST CHANNEL

Calling sequence: Load A register with the first and second choice channel numbers.
Store the third and fourth choice channel numbers in D.T2.
RJM R.RCH

This routine stores the channel numbers loaded in the A register in D.T1, inserts the monitor function M.RCH into D.T4, and writes D.T0 through D.T4 to the output register for that PP. Channel numbers in D.T1 and D.T2 are 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, and so on.

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

LDN	0
STD	D.T2
LDC	1312B
RJM	R.RCH

Normally, CPMTR stops looking for alternate channels after four have been investigated. In the preceding example, only two channels are investigated.

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

DSD and a few other programs need an immediate reply on a channel request, even if the channel is already reserved by another PP. These programs do not use R.RCH; they issue an M.RCH function through R.MTR. These requests are made with a zero in D.T4 and, if the channel is available, the monitor assigns it and sets D.T4 to nonzero in the reply. If the channel is not assigned to the requesting PP, D.T4 remains zero in the reply.

R.DCH — DROP CHANNEL

Calling sequence: Load A register with channel number.
RJM R.DCH

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

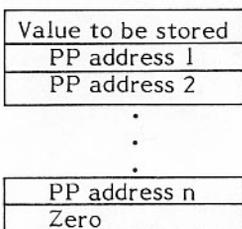
R.STBMSK

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.

R.STB — STORE BYTE

Calling sequence: Load A register with the address of a list.
RJM R.STB

List has the following form.



A logical AND is performed with the mask at location R.STBMSK on each word specified in the list before an exclusive OR is performed with the value to be stored. R.STB is used primarily to substitute channel numbers in driver overlays.

R.OVL — OVERLAY LOADER

Calling sequence: Store name of overlay left-justified in D.T6,D.T7.
Load A register with load address.
RJM R.OVL

An M.PPLIB monitor function is issued along with the overlay name and load address via an R.MTR call. CPMTR determines overlay residence (disk, extended memory, or CM) and sets up the load accordingly. For disk resident overlays, a stack request is set up in the PP message buffer and the M.PPLIB function is changed to an M.ICE/EX.SPM function. For extended memory resident overlays, a system circular buffer (DDP or non-DDP) is reserved and the buffer parameters are passed in words 1 and 2 of the PP message buffer. For CM resident overlays, the CM address and length of the overlay is passed in word 3 of the PP message buffer.

After the M.PPLIB request is acknowledged, R.OVL calls R.READP to load the overlay. If an illegal program name has been requested, R.READP returns an error status in D.T4+C.RWPPST and R.OVL issues an M.ABORT function and exits to R.IDLE. Otherwise, R.OVL returns to the caller.

R.EREQS — ENTER STACK REQUEST

Calling sequence: Store address of stack request in D.TO.
RJM R.EREQS

This routine adds the control point number to the already formatted request, writes it in words 1 and 2 of the message buffer, clears word 3, and issues the request via the M.ICE/EX.SPM function.

R.DFM — ENTER DAYFILE MESSAGE

Calling sequence: Load A register with the address of message + flag bits.
RJM R.DFM

A message is written to the dayfile and/or displayed on the console. The flag bits in the high-order 6 bits of the A register are used to determine message destinations. In the following flag bit values, one or more bits may be on; all are optional (refer to M.DFM). When a bit is set, the corresponding action occurs.

<u>Bit Set</u>	<u>Description</u>
0	Do not send message to B display.
1	Do not send message to control point dayfile.
2	Do not send message to system dayfile (A display).
3	Flag the message as an accounting message.
4†	Send the message to the hardware error file.
5††	Do not put the job name in the message.

†If bits 1 and 4 are set, the message is not sent to an INTERCOM terminal but is sent to an INTERCOM control point dayfile.

††If bits 2 and 5 are set, the time is omitted and replaced with blanks in the control point dayfile. This option identifies messages from a task that was executed on a different mainframe.

R.READP — TRANSMIT DATA VIA CHANNEL FROM STACK PROCESSOR

R.WRITEP — TRANSMIT DATA VIA CHANNEL TO STACK PROCESSOR

Calling sequence: Load A register with address of stack request.
 Load A register with 0 if request already issued.

RJM R.READP
 or
RJM R.WRITEP

These routines control transmission of data to/from the PP via a channel-stack processor interface or a CM system circular buffer interface. R.READP or R.WRITEP calls R.RWP to perform the read/write logic after which the following is returned to the caller.

D.T3+C.RWPPPLW	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.

R.RWP — PERFORMS READ/WRITE LOGIC FOR R.READP/R.WRITEP

Calling sequence: Store address of stack request in D.T0.
 Load A register with IAM/OAM instruction.
RJM R.RWP

This routine is called by R.READP/R.WRITEP to perform the read/write logic. If D.T0=0, read/write logic is entered directly. Otherwise, the PP message buffer address is added to the stack request which is issued by R.EREQS.

Read/write logic idles on byte 0, word 3 of the PP message buffer. If the field access flag is set, R.RWP pauses via R.RAFL while waiting. The following values may be found in byte 0.

<u>Value</u>	<u>Significance</u>
1 through 4	Indicates channel-stack processor transfers.
5	Indicates that word 3 contains parameters to load a PP overlay from CM.
6 or greater	Indicates a system circular buffer interface.

A system circular buffer interface requires the load of a segment into the area between the end of R.RWP and C.PPFWA. Unless the information required to load the segment is already loaded, the information is contained in word 3 of the message buffer.

Segments exist for loading extended memory-resident PP overlays either through CM buffers or directly from extended memory via a DDP. Segments also exist for communicating with the 819 disk stack processor, HSP (applicable to CYBER 176 only). Segments are loaded from T.PPOVL in CMR.

R.RWP returns to R.READP/R.WRITEP after the request is complete.

FIELD ACCESS FLAG USAGE

The control point field access flag (R.FAF) is found at location 100g in PP resident. A copy of the flag is kept in CM in the PP status table entry (T.PPS1) for each PP (except PP0). It is used by the PP to prevent storage moves at a control point while the PP is accessing the control point's field length. R.RAFL and R.TAFL (refer to descriptions in this section) obtain and release field access.

The field access flag must be set whenever data is read or written within a control point's field length. If a PP program is looping, waiting for an external event to occur, the loop must be performed while the field access flag is not set, or the loop must include a call to R.RAFL. When no field access is required for a major operation (such as searching a CMR table), it is advisable to call R.TAFL before the process.

Execution of the R.MTR subroutine or any resident routine that calls R.MTR (that is, R.RCH, R.OVL, R.EREQS, R.DFM, R.READP, R.WRITEP, or R.RWP) may result in a call to R.RAFL. If an absolute CM address within a control point's field length has been computed and saved, the address will be invalidated because the control point may have been moved.

SYSTEM MONITOR

The system monitor consists of MTR, which runs continuously in PP0, and CPMTR, which resides in CM and uses the CP intermittently for short bursts. The monitoring tasks are divided between MTR and CPMTR on a functional basis to distribute the work load in the most efficient manner.

MTR STRUCTURE

Unlike CPMTR, MTR is not initiated to perform a specific function. It runs continuously and must keep searching for requests directed to it. The frequency with which it scans for each type of request can have major impact on system efficiency. The following major responsibilities are listed in an order which corresponds approximately to the frequency with which they should be performed.

- Advance system clocks.

The accuracy of the system clocks in T.CLK and T.MSC is directly related to the frequency with which MTR accesses the real-time clock on channel 14g.

- Check T.PPIP.

This is the word into which PP resident writes its input register address when it has a monitor function for MTR. Frequent checking reduces the MTR response time and reduces chances of conflict between two PPs in the use of this word.

- Check T.MTRRS.

This is the short buffer through which CPMTR passes PP calls taken from RA+1. It is also used for some PP monitor functions called by CPMTR or scheduler (M.MFLA or M.ISP). MTR should keep this buffer clear so that it will not inhibit the efficient execution of CPMTR.

- Check individual PP output registers.

T.PPIP is used for quick attention from MTR on monitor functions. It is still necessary for MTR to scan the output registers because if two PPs make requests in near unison, one will be lost from T.PPIP. Also several monitor functions, such as M.BUFPTR, M.DFM, and M.MFLA may not be completed on their first processing. These functions will be processed more quickly if the output registers are scanned more frequently.

- Advance control point.

Examine each control point in turn to see if a PP program should be initiated from the event stack, if the CPU should be restarted from recall status, or if 1AJ should be called to advance to the next control statement.

- Check RA+1.

Examine RA+1; if it is nonzero, initiate CPMTR. This function is intended only to initiate CPMTR for a program that does not use the exchange jump capability.

- Check for hardware errors (CYBER 180-class mainframes only).

Test the maintenance registers or a flag in T.EPIB (if DFT is in dedicated mode) for hardware errors. If any error is found, call 1MR to process the error.

- Check status and control register (CYBER 70 and 170-class mainframes).

Test the status and control register for hardware errors. If any error is found other than a corrected memory error that has occurred previously, call ISC to process the error.

CPMTR ORGANIZATION

CPMTR is the one central program that has no exchange package area of its own. When CPMTR is not running, its exchange package is stored in the exchange package area of the control point for the job it selected to execute. Whenever a program that is part of the currently executing job makes a system request, it places the system call in the word at RA+1 and performs a central exchange jump (XJ) instruction. This reinitiates the execution of CPMTR and saves the register contents of the user mode program in its own exchange package area.

When CPMTR begins, it checks RA+1 of the control point that was running. If RA+1 is nonzero, its content is picked up by CPMTR and RA+1 is cleared. This call is compared against a list of system calls to be performed immediately by CPMTR. This list of system calls includes ABT, CIO, END, RCL, TIM, XJR, and others.

If not one of these, the call is placed into the small buffer at T.MTRRS so that MTR can assign it to a PP. If the RA+1 call does not have the auto recall bit set, CPMTR immediately returns control to the user program. If the auto recall bit is set, CPMTR sets that control point into auto recall status and reassigns the CPU to another user program.

PP programs also can direct system requests to CPMTR. Typically, they use the R.MTR routine of PP resident for such requests. R.MTR places the monitor function in its output register and then determines if the function should be directed to CPMTR or to MTR. When calling CPMTR, the PP input register address is written in T.PPID so CPMTR can identify the calling PP without scanning all the output registers.

If CPMTR finds RA+1 is empty, T.PPID is checked. If an input register address is in T.PPID, CPMTR clears it and checks the corresponding output register for a function to be performed. The CPMTR functions are described with the other monitor functions.

CPMTR also checks MTR's output register. This extra check is made because MTR does not use T.PPID when it issues a CPMTR function. Otherwise, a function from MTR is handled just like any other PP.

If CPMTR cannot determine why it was called, it returns control to the interrupted user program.

OPERATIONS

MTR and CPMTR perform operations that, once started, must be completed without interruption. MTR and CPMTR perform other tasks that are best managed by a single centralized routine. Among these operations are

- CPU scheduling.
- Assignment of the PPs.
- Channel reservations.
- Time accounting.
- Storage requests.
- Other operations easily done by a single centralized routine.

CPU SCHEDULING

CPMTR assigns the CPU to jobs at control points or to scheduler or storage move, which are system tasks that can be interrupted. The CPU status of jobs is controlled by PPs or by the job. CPMTR accepts requests for a change of status and records the current status as a set of bits in the control word (W.CPUST) associated with each exchange package.

The most significant of these bits are

<u>Bit</u>	<u>Description</u>
W	Set by an M.RCP or M.SETST monitor function when a central program is loaded for execution. It remains set until the program posts END in RA+1 or is aborted for any reason.
C and D	Set when the job is being executed in CPU A or B, respectively.
X	The job is in periodic recall status because of an RCL request from the program.
Y	The job is in auto recall status and is not restarted until the requested system function is completed.
Z	The job is suspended because it threatened to saturate the system with PP calls.
M, P, and S	Job execution has been temporarily suspended by storage move, checkpoint, or the job swapper, respectively.

An exchange package exists for each control point, in addition to one for storage move and scheduler, and one for each CPU idle program. An exchange package is ready to use the CPU if W is on and M, P, S, X, Y, and Z are all off. CPMTR assigns the exchange package to the CPU on a priority basis. The priority used is the CPU priority.

The CPU priority is a 6-bit field in the CPU status word. Priority levels, in ascending order, are listed below.

<u>Priority Level</u>	<u>Description</u>
PR.IDLE	Zero level default CPU job to be used only in the absence of any other.
PR.BATCH	Batch jobs initiated by IIB.
PR.INT	INTERCOM jobs initiated by ISI.
PR.SCP	System control point jobs initiated by DSD.
PR.SYS	Storage move and scheduler.

When more than one job is at the highest active priority level, the CPU is shared on a round-robin basis. Each uses the CPU for BASESLIC milliseconds before control is passed to the next job.

This combination of priority and round-robin scheduling is overridden during the time that an RMS driver is transferring data to or from a user's buffer. If the user's program is not in auto recall, it is given a slice of CPU time so that it can process the data as it is transferred. In this way, a low priority job may temporarily preempt the CPU away from a high priority job.

When CPMTR assigns the CPU to a job, BASESLIC is added to the current time to produce the projected end-of-slice time. The end-of-slice time is posted in T.CPSTA (or T.CPSTB). When the time arrives, MTR issues an M.SLICE function which causes CPMTR to select the next job for the CPU.

ASSIGNMENT OF THE PPs

PP scheduling is done by MTR. A PP can be requested by a CPU program through a call in RA+1 or by another PP through an M.RPJ or M.EES monitor function.

MTR always reserves at least one PP for the RMS stack processors to ensure that the stack processor is not locked out while all programs in the PPs are waiting for the stack processor to access RMS. More than one PP can be reserved for the stack processor (refer to N.SPRPP in CMR). MTR maintains the following lists of PP jobs that are not currently assigned to a PP.

<u>Call</u>	<u>Description</u>
PP job queue	This is where PP jobs are placed when no PP is available. It is a first in/first out, ordered queue except for stack processors. A PP call that MTR identifies as a stack processor will be added at the front of the list, pushing down all the members already in the list.
Delay stack	These are PP jobs generated by the M.RPJ function with a nonzero time delay. The jobs are ordered in sequence of their time delay. When the delay is expired, the jobs are removed from the delay stack and assigned to a PP or added to the PP job queue if no PP is available.
Event stack	These are PP jobs generated by the M.EES function. The stack is searched periodically to find any entries whose event has occurred. When one is found, it is removed from the event stack and assigned to a PP or added to the PP job queue if no PP is available.

MTR keeps the control values for these lists in PP0. The PP jobs are kept in the peripheral job table (T.PJT) in CM. Each job consists of the input register and three words for the PP message buffer. When a PJT entry is not in use, the input register word is set to zero.

To control PP assignments, MTR keeps (in PP0) an 8-byte status word for the PP entries which form the PP status table. Each status word has one of two formats, depending upon whether the PP is assigned or unassigned and available.

1	2	3	4	5	6	7	8
CPAD	PPLIST	BUFPTR	APLINK	JUMPAD	PPFLAG	PPSEC	PPMSEC

CPAD	Base address of control point area to which PP is assigned.
PPLIST	Address of list pointer.
BUFPTR	The low order 12 bits of the buffer pointer the last time that MTR checked it (refer to M.BUFPTR).
APLINK	Active PP link. This is a pointer to the next member in a chain of active PPs. The chain always starts with MTR (PP0) and ends with DSD (PP1). The next PP 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 (PP assigned). When idle, PPFLAG contains the link to the next idle PP.
PPSEC	PP starting time in seconds.
PPMSEC	PP starting time in milliseconds.

When the PP is unassigned and available, the PP status word is linked in a chain of unassigned and available PPs, using byte 6 of the status word (PPFLAG). There can be a maximum of three linked lists, one for each bank of PPs and one for PPs reserved for stack processors. PP direct cells SPLIST, PPIL, and PPIU each contain a pointer to the status word at the head of their own chain. Byte 6 contains a pointer to the status word of the next available PP. If no more PPs are available, byte 6 contains zero. A chain of three available PPs is illustrated in figure 3-4.

MTR assigns a PP by writing a peripheral job name and a control point number into the PP input register to perform one of the following actions.

- Satisfy a PP program call issued as an RA+1 request.
- Answer a PP request for another PP job (M.EES, M.EESD, M.RPJ, or M.RPJD 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 IAJ to a control point when all control point activity has ceased.

MTR maintains a PP queue table containing a maximum of 40 entries, each 4 bytes long. Each entry corresponds to a four-word entry in the peripheral job table (PJT) in CMR. In the queue, the following chains are kept.

- A queue of PP jobs that cannot be initiated currently because PPs are not available. This PP job queue is a chain of PP input register images.
- A queue of PP jobs that must be initiated after a given time delay. This queue is a time-ordered chain of PP register images, called the delay stack.

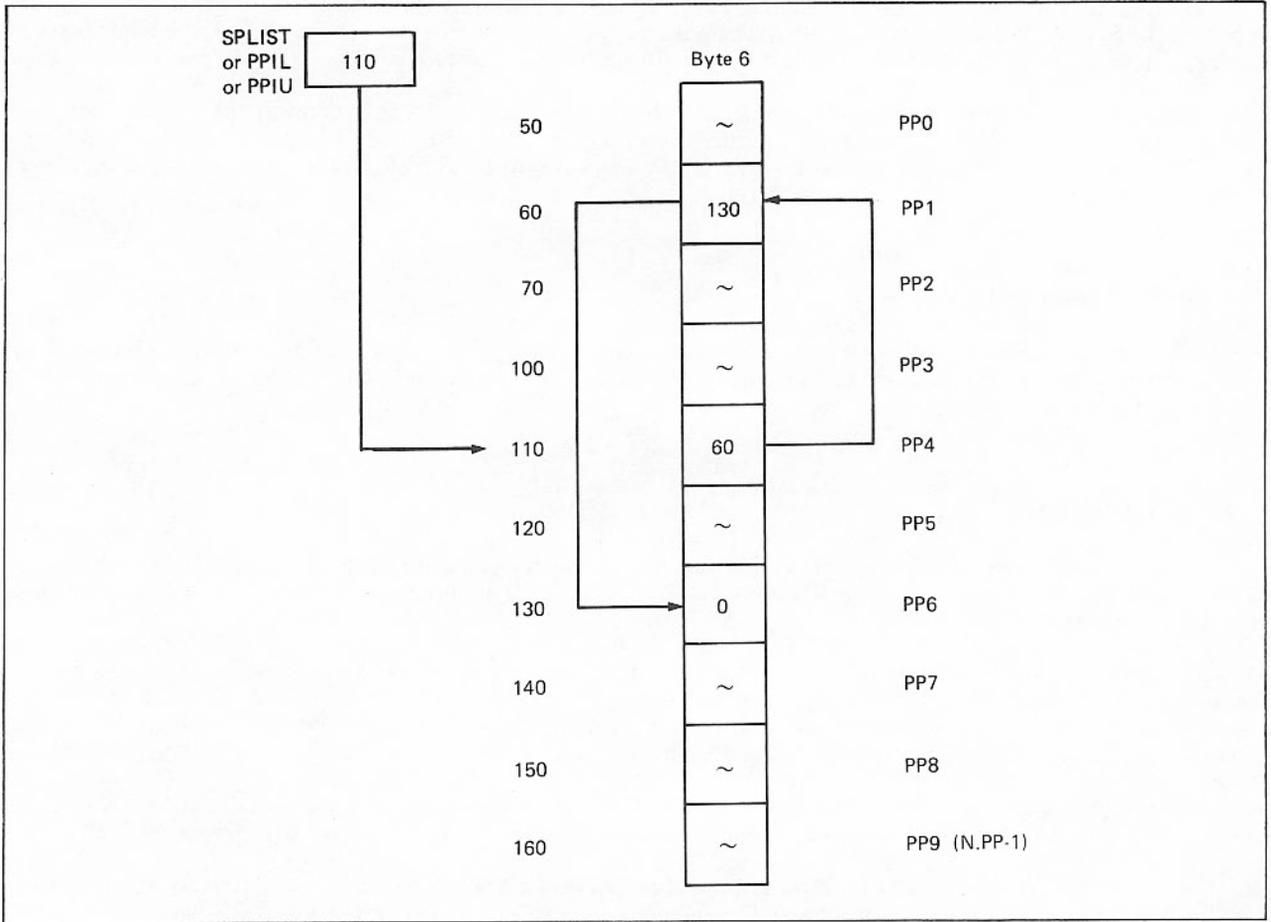


Figure 3-4. PP Chain

- A separate queue of PP jobs for each control point which must be initiated after a specified bit has been set or cleared in CM. This queue is called the event stack.
- An empty queue of all unused entries in the PP queue.

Each type of chain has a chaining mechanism within MTR. The direct cell area of MTR has a pointer that defines the beginning of each chain, except event stack entries.

<u>Chain</u>	<u>Pointer</u>
PP job queue	NPPQ
PP delay queue	NACT
Empty PJT entries	EMPTY

The pointers to the event stack are in the MTR control point table EVST. Each control point has a separate EVST. A fourth pointer, LPPQ, defines the end of the PP job queue. When the time delay expires for an entry in the delay stack, that entry will be transferred to the end of the job 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. A PP job queue containing a two-entry overload queue, a one-entry delay stack, the empty chain, and two one-entry event stacks is illustrated in figure 3-5.

CHANNEL RESERVATIONS

CPMTR processes channel reservations. The CPU is used for this frequently used function because it is easily accessed from PP resident. If the requested channel is already busy, MTR periodically reissues the request to CPMTR.

TIME ACCOUNTING

MTR uses the real-time clock on channel 148 as the source for its time keeping duties. MTR maintains the two basic system time clocks T.CLK and T.MSC. T.CLK is a 24-hour clock that gives the time of day in hours, minutes, and seconds.

Bits 35 through 12 of T.MSC contain the total number of seconds since the last deadstart, expressed as a 24-bit binary number. Bits 11 through 0 are the binary fractional parts of a second. Bits 35 through 0 contain a continuous binary number recording the time as seconds times 4096, which is used as the accounting basis for CPU time, I/O time, and PP time.

CPU time is compiled by CPMTR. Each time the CPU is rescheduled, the current value of T.MSC is recorded so that the elapsed time can be computed the next time the CPU is rescheduled.

MTR accumulates I/O time and PP time, except that portion for RMS devices. This part is accumulated by CPMTR from a PRU count that is passed to it in the M.SPRCL function from the stack processor.

STORAGE REQUESTS

CPMTR processes storage requests for CM and EM. If storage is not available, CPMTR turns control over to the integrated scheduler which determines if storage can be made available through swap and/or rollout. Storage allocation and storage moves are described under Control Points, section 2.

When a control point RA and/or FL is to be changed, CPMTR suspends the control point by setting the M bit in the CP status byte and setting the storage move flag. PPMTR is then called. It waits for each PP accessing the control point's field length to clear its field access flag. When all field access flags are clear for the control point, the system job control point is restarted to perform the move. When the move is complete, CPMTR clears the M status bit and restarts the control point that was moved.

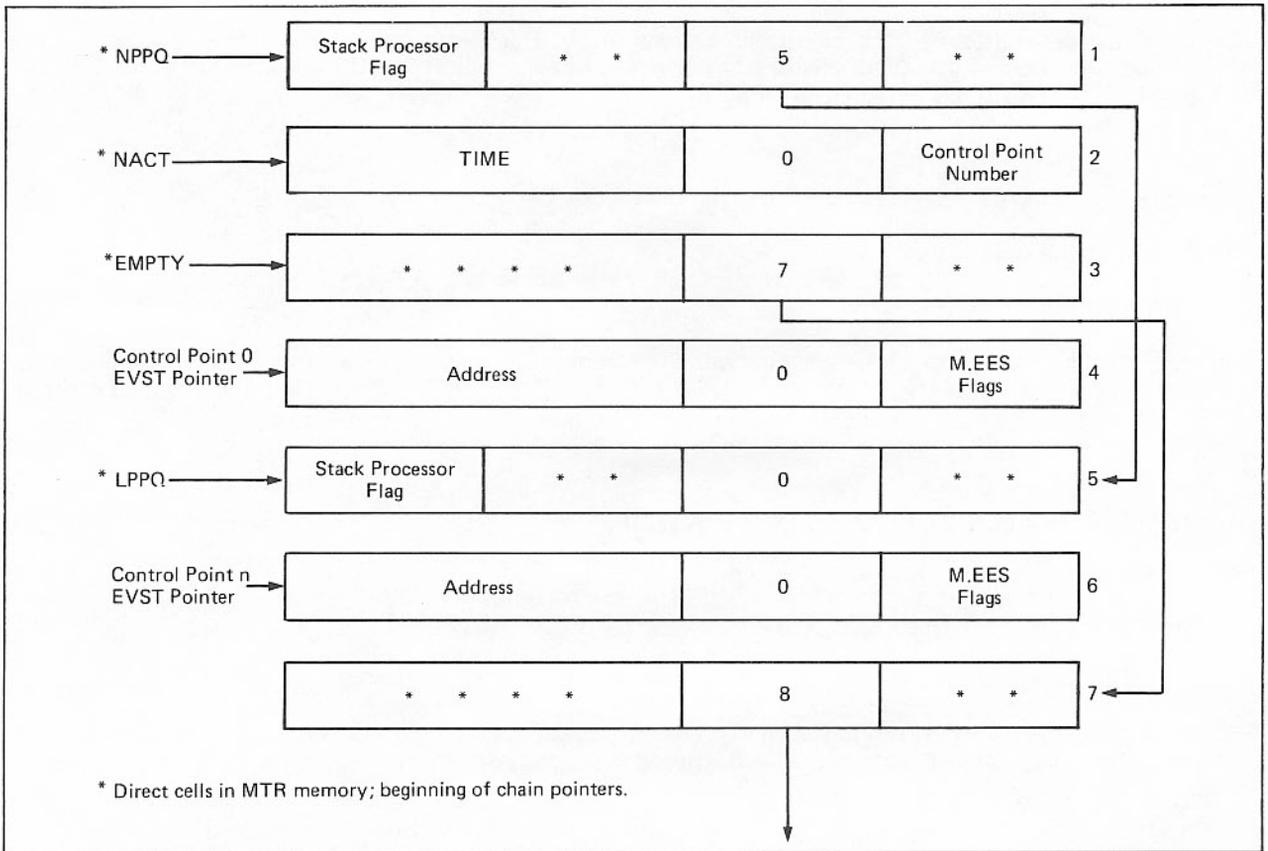


Figure 3-5. PP Job Queue (MTR Memory)

MONITOR FUNCTIONS

The following descriptions of the monitor functions are in alphabetic order. The tables in appendix B of this manual list the monitor functions in numerical sequence. Functions with a code number of less than or equal to M.MTRCPU are assigned to CPMTR.

M.ABORT — ABORT CONTROL POINT AND DROP PP

The format of the function is

```
M.ABORT,****,****,****,****
```

The job at the requesting PP 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 to note the abort function.

M.BUFPTR — WATCH BUFFER POINTER WORD

The format of the function is

```
M.BUFPTR,****,****,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

The format of the function is

```
M.CCPA,****,****,****,**NN
```

The requesting PP is released from its current control point assignment as if it had issued an M.DPP function, but its input register is not cleared. The PP is assigned to control point NN, and the new control point number inserted in its input register. It is the responsibility of the requesting PP to alter D.CPAD. R.TAFL should also be used to clear the field access flag at the old control point.

M.CLRST†— CLEAR STATUS

The format of the function is

M.CLRST,BBBB,****,****,00NN

BBBB Pattern of bits to be cleared.

NN Control point number (used only by MTR).

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

M.CPJ†— CAPTURE PERIPHERAL JOB

The format of the function is

M.CPJ,00XX,XXXX,****,****

XXXXXX Buffer address, relative to RA.

This request is issued to find all PP jobs for a control point that are 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 completed.

M.CPUST†— CHANGE CPU STATUS

The format of the function is

M.CPUST,000X,****,****,****

<u>X</u>	<u>Description</u>
0	If either CPU is off, it is turned on. If the CPU was locked off at deadstart time, it remains off.
1	If both CPUs are on, CPU-A is turned off.
2	If both CPUs are on, CPU-B is turned off.

When the requested function cannot be performed, no action is taken.

M.DCP†— DROP CENTRAL PROCESSOR JOB

The format of the function is

M.DCP,****,****,****,****

†CPMTR function.

Execution of the CP job at control point is stopped. The control point status bits W, X, Y, and Z are cleared. The control point status bits set prior to M.DCP are returned in byte 1 of the output register of the requesting PP.

M.DFM — PROCESS DAYFILE MESSAGE

The format of the function is

M.DFM,FFFF,MMMM.****,****

Dayfile flag bits FFFF determine message handling.

<u>FFFF</u>	<u>Description</u>
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 MMMM is larger than the address of PPOR, MMMM is taken to be the LWA+1 of the message in the PP message buffer. FWA of the message is the FWA of the PP's message buffer (limit of 60 characters). When MMMM is smaller than the address of PPOR, it is taken to be an index for a dayfile buffer dump.

Value of dayfile dump index:

<u>Value</u>	<u>Description</u>
0	System dayfile dump.†††
1 through N.CP	Control point dayfile dump.
N.CP+1	Hardware error file dump.

†If bits 1 and 4 are set, the message is not sent to an INTERCOM terminal but is sent to an INTERCOM control point dayfile.

††If bits 2 and 5 are set, the time is omitted and replaced with blanks in the control point dayfile. This option identifies messages from a task that was executed on a different mainframe.

†††Bits 0 and 1 of FFFF must be set to flush the dayfile buffer for a system dayfile dump.

M.DPP — DROP PP

The format of the function is

M.DPP,FFFF,****,****,****

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

M.EES — ENTER EVENT STACK

The format of the function is

M.EES,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.

<u>Value</u>	<u>Description</u>
F is 0	Event stack job assigned when bit is off (F.ESOFF).
F is 4	Event stack job assigned when bit is on (F.ESON).
B is 0	AAAAAA is an absolute address (F.ESABS).
B is 1	AAAAAA is relative to RA (F.ESREL).
B is 2	AAAAAA is a control point area address (F.ESCPA).

This function is used to start a PP program after a specified event has occurred. That event must be defined as a specific bit in CM being on or off. The bit is defined by the parameters in the output registers. W.PPMES1 contains the PP input register image of the program that is to be assigned when the event occurs. The contents of W.PPMES4, W.PPMES5, and W.PPMES6 are also saved and set in the message buffer of the PP program when it is started. This function will not complete if the peripheral job table is full. If possible, use M.EESD instead of M.EES.

M.EESD — ENTER EVENT STACK AND DROP PP

The format of the function is

M.EESD,00AA,AAAA,FFFF,SYTT

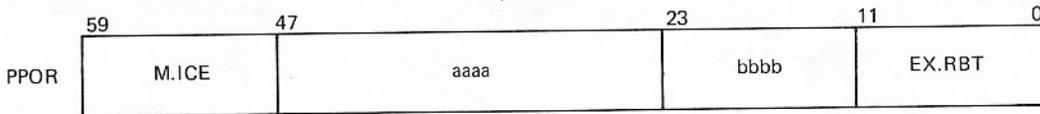
M.EESD combines the functions of M.EES and M.DPP. This function is completed even if the peripheral job table is full. If FFFF=M.EESD, the control point is not charged for the PP time. All other parameters are identical to M.EES.

M.ICE†— INITIATE CENTRAL EXECUTIVE

The format of the function is

M.ICE,PPPP,PPPP,PPPP,EX.xxx

EX.xxx is a subfunction to be performed. These subfunctions are listed under Monitor Functions in appendix B. The PPPP fields can be used as parameters to the subfunction. An example of the call format for these subfunctions is the following call to EX.RBT for PRU conversion.

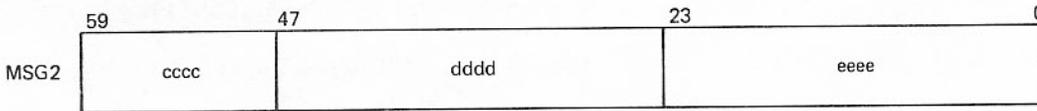


aaaa PRU number
 bbbb FST(1) address

If bbbb=0, then the call includes the following words:

MSG1 Contents of FST(1)
 MSG2 Contents of word one of the FNT supplement (extended memory resident files only); the address of the supplement being in bits 59-48.

Upon completion of the function, PPOR will contain zeros, MSG1 will contain either a new value for FST(1) or, if the file is extended memory resident, a new value for word two of the FNT supplement. MSG2 will be as follows:



cccc PRUs divided by record blocks (PRUs/RB)
 dddd PRU number
 eeee Remainder

If aaaa is zero, dddd is the length of the file's current position and eeee is zero.

If aaaa is not zero, dddd equals aaaa and eeee is zero. If aaaa exceeds the length of the file, dddd is equal to the length of the file in PRUs and eeee = aaaa - dddd.

When aaaa is not zero, the file is repositioned to the value indicated by aaaa and a new value for FST(1) (word two of the FNT supplement if it is an extended memory resident file) is placed in MSG1.

Refer to the section on INPUT/OUTPUT for a detailed description of the 819 subsystem flush function (EX.SUB).

†CPMTR function.

M.ISP — INITIATE STACK PROCESSOR

The format of the function is

M.ISP,000X,000Y,****,CCCC

CCCC	DST ordinal of stack processor to be initiated.
X is 0	Initiate 1S5 only if PP active flag is zero.
X is not 0	Initiate 1S5 regardless of PP active flag setting.
Y is 0	Initial assignment of 1SP to the DST entry.
Y is 1 (or 2)	A partner call.

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 set, the output register is cleared and an exit made; if set, 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. When Y is 1 or 2 and X is not 0, this is a call from 1SP for a partner to work together for a dual access device.

M.KILL — BAD FUNCTION REQUEST

The format of the function is

M.KILL,****,****,****,****

MTR flags the function request as bad and automatically enters STEP 0 mode. The output register of the requesting PP is not cleared. MTR sets bit 11 of byte 0 in the output register to indicate the BAD MONITOR REQUEST message was issued, and sets bit 10 to indicate the STEP request was issued.

M.MCH — READ MAINTENANCE CHANNEL

The format of the function is

M.MCH,000F,****,00PP,PPPP

MTR reads the contents of the P register of a primary processor for a CYBER 180-class mainframe over the maintenance channel, provided F is set to 1 when the request is issued. This routine returns the following values:

F is 0	The maintenance channel is busy; M.MCH has not yet executed.
F is 1	M.MCH has executed.
PPPPPP	Contents of the P register.

This routine must not be issued more than once every 500 milliseconds. M.MCH operates only on the CYBER 180-class mainframes; issuing M.MCH on any other machine constitutes an illegal monitor request.

M.MFLA — MONITOR FIELD LENGTH ACCESS AT CONTROL POINT

The format of the function is

M.MFLA,000M,****,****,CCCC

M	Suspend/Resume flag.
CCCC	Address of control point being moved.

This function is used by the memory management routines to suspend execution of a control point that is to be moved and to resume execution of the control point after the move. Setting M to zero places the control point in move status. If there are no field access flags set, CPMTR restarts the system job control point to perform the move. If there are field access flags set, PPMTR waits until the flags are clear, then restarts the system job control point to perform the move. Setting M to one takes the control point out of move status. The control point then becomes eligible for CPU execution.

M.NOTE — NULL FUNCTION

The format of the function is

M.NOTE,****,****,****,****

This is a null function used with STEP mode as a breakpoint. During normal execution, MTR clears the output register of the requesting PP.

M.NTIME — ENTER NEW TIME LIMIT

The format of the function is

M.NTIME,TTTT,T***,****,**NN

A CP job time limit of TTTTTT seconds is entered at the control point. Any previous time limit is superseded. If the requesting PP is assigned to control point 0, the parameter NN gives the number of the control point to be considered; in any other case, this parameter is irrelevant.

M.PASS — PPMTR IGNORES FUNCTION REQUEST

The format of the function is

M.PASS,****,****,****,****

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

M.PATCH — INSERT A PATCH IN PPMTR

The format of the function is

M.PATCH,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.PPLIB†— PP LIBRARY SEARCH FUNCTION

The formats of the function are

M.PPLIB,AAAA,AA**,LLLL,****
M.PPLIB,000X,****,****,****

This function is used by PP resident routines only. Its initial use is to locate and set up the loading of a PP overlay.

AAAAAA Name of overlay.
LLLL PP load address.

Subsequent use of the function is to control the loading process.

<u>X</u>	<u>Description</u>
0	Release system circular buffer and DDP.
1	Continue loading SCB.
2	No DDP code; default to disk copy.
3	No extended memory code; default to disk copy.
4	DDP block transfer error; start error recovery process.
5	Report DDP recovery error.
6	Report completion of error recovery process.
7	Load next overlay to continue error processing.
10	Obtain event number for use by ISP/ISQ in logging DDP errors.
11	Report ISP/ISQ DDP error.
12	Report STL channel error.

M.RACT — REQUEST CONTROL POINT ACTIVITY

The format of the function is

M.RACT,**NN,IIII,****,****

†CPMTR function.

This request provides the various activity counts of control point NN at a given time (NN cannot be zero). If IIII is nonzero, the pseudo activity count is 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†— REQUEST RBT STORAGE

The format of the function is

M.RBTSTO,SSSS,****,****,****

CPMTR sets SSSS*100 as the new RBT starting address. If the request cannot be honored, the old RBT starting address is returned in SSSS.

M.RCH†— REQUEST CHANNEL RESERVATION

The format of the function is

M.RCH,BBAA,DDCC,***G,RRRR

AA	First choice channel number.
BB	Second choice channel number.
CC	Third choice channel number.
DD	Fourth choice channel number.
G is 0	Normal charge for channel time.
G is 4	Do not charge control point for channel time.
RRRR is 0000	Request immediate reply.
RRRR is M.RCH	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 more choices. If none of the requested channels is available, PPMTR sets byte 0 of the PP output register to 0. When a channel is granted, its number is returned in the PP output register byte 1 (location of AA) and byte 4 is set to a value not 0. Thus, programs that request an immediate reply must check that byte 4 is not 0 before using the channel.

†CPMTR function.

On exit, if a channel has been reserved, the output register appears.

0000 XXXX TTTG TTTT YYYY

XXXX Channel number.

TTTG TTTT Information from the channel status word, where G is the charge/no charge bit for channel time.

YYYY PP input register address.

M.RCLCP†— RECALL CENTRAL PROGRAM

The format of the function is

M.RCLCP,****,****,****,****

This request is effective only if the central program associated with the requesting PP 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.RCP†— REQUEST CENTRAL PROCESSOR

The format of the function is

M.RCP,****,****,****,****

CPMTR puts the job in waiting status (W), provided none of the following conditions exists:

- Requesting PP is assigned to control point 0.
- Error flag is set for the control point.
- Job is already in waiting status.

If any of these conditions exists, the request is ignored.

M.RPJ — REQUEST PERIPHERAL JOB

The format of the function is

M.RPJ,SSSS,FFFF,****,****

This function requests that another PP program be initiated after a specified time delay. The first word of the requesting PP message buffer contains the input register image of the new PP program. The time delay is SSSS seconds plus FFFF/10 000_g seconds. If the time delay is zero and no PP 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.RPJD should be used in preference to M.RPJ whenever possible.

†CPMTR function.

M.RPJD — REQUEST PERIPHERAL JOB AND DROP PP

The format of the function is

M.RPJD,DDDD,DDDD,FFFF,****

Combines the functions of M.RPJ and M.DPP. This function will be completed even if the peripheral job table is full. If FFFF=M.RPJ, the control point is not charged for the PP time. The use of the time delay is the same as for M.RPJ.

M.RSTOR†— REQUEST STORAGE

The format of the function is

M.RSTOR,CCCC,XXXX,00TT,****

CCCC	Request CM/100 (octal).
XXXX	Request extended memory/1000 (octal).
TT	00 CM request only.
	01 Extended memory request only.
	02 CM and extended memory request.
	04 Request CM - awaits response.
	06 Request CM and extended memory - awaits response.
	07 IP.POSFL requested by swapper.

This function assigns CCCC central memory and/or XXXX extended memory to the control point of the requesting PP. 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 is always honored. If TT is 02 or 04, CPMTR can honor part of the request without honoring the remainder.

†CPMTR function.

M.SCB — SYSTEM CIRCULAR BUFFER SURVEILLANCE

The format of the function is

M.SCB,****,00BB,BBBB,EX.CBM

BBBBBB System circular buffer address.

The system circular buffer is an FET-like table that has a trigger and a direction flag in addition to FIRST, IN, OUT, and LIMIT. MTR uses IN, OUT, and the trigger and direction (RMS-to-extended memory or extended memory-to-RMS) to determine if a threshold has been reached. If not, no action is taken. If so, MTR issues an M.ICE/EX.CBM function to start CBM for processing of the system circular buffer.

M.SCH†— INITIATE INTEGRATED SCHEDULER

The format of the function is

M.SCH,00YX,00CC,00JJ,JJJJ

When X is 2, the contents of the output register are placed into a buffer at T.SCHRS for the integrated scheduler. When the integrated scheduler processes this request, it links the JDT at location JJJJJJ to the job queue for JCA ordinal CC. If CC is zero, the job class is taken from the JDT.

When X is not 2, the contents of the output register are not passed to the scheduler.

In both cases, CPMTR initiates the scheduler immediately.

If Y is set to 2, CPMTR will set location RJBBI= -1, causing the scheduler to call BBJ.

M.SEF†— SET ERROR FLAG

The format of the function is

M.SEF,**NN,EEEE,****,****

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

The M.SEF function recognizes two special control values in the error flag field that are used to initiate and terminate the memo mode.

When a control point is in memo mode, error flags are not set in byte C.CPEF(1), but are recorded in bits 5 through 0 of byte C.CPMEMO(0). The high order bits 11 through 6 of C.CPMEMO are set on when the control point is in memo mode, and are used by MTR to recognize the mode.

F.ERMEMO (-77g) initiates memo mode. Bits 11 through 6 of C.CPMEMO are set on. Bits 5 through 0 of C.CPEF are moved to bits 5 through 0 of C.CPMEMO and C.CPEF is cleared. If the control point is already in memo mode, the effect of the F.ERMEMO is to clear an error flag memo without terminating the memo mode.

†CPMTR function.

F.ERTMM (-0) terminates memo mode. Bits 5 through 0 of C.CPMEMO are moved to C.CPEF and C.CPMEMO is cleared. Error flag zero can also terminate memo mode; it clears both the error memo and the error flag fields. If an error memo is already recorded when the F.ERTMM is issued, the memo is made an error flag causing the normal error flag processing to take place.

When a control point is in memo mode, the M.SEF function with error flag values 1 through 77g does not cause the CPU to be dropped as when in normal error flag mode; however, there are some exceptions. The following error codes are caused by errors in the central program and render the CPU useless.

2	F.ERAR	Arithmetic error.
4	F.ERCPC	CPU abort (ABT in RA+1).
5	F.ERPCE	PP call error (garbage in RA+1).
15B	F.ERRCL	Auto recall error.
23B	F.ERPE	CPU parity error.

Any of these codes cause the control point to revert to normal error flag mode.

When entering memo mode, it is possible that an error flag had been set just prior to the processing of the F.ERMEMO. The PP program that initiates memo mode should immediately check the error memo field after completion of the F.ERMEMO. If an error memo is set, it should be assumed that it occurred as an error flag prior to the F.ERMEMO. Usually the best action at this point is an F.ERTMM. Since the program is not yet committed to its critical stage, it is best to allow the error flag processing to continue.

Memo mode is restricted to use during single control statement executions only. It is the responsibility of the program that initiates memo mode to terminate it. If 1AJ finds a control point in memo mode, it is processed as an error flag.

M.SEQ†— ASSIGN JOB SEQUENCE NUMBER

The format of the function is

M.SEQ,****,****,****,****

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

M.SETST†— SET STATUS BITS

The format of the function is

M.SETST,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; can cause linkage to or delinkage from chain of control points actively waiting for a CPU.

†CPMTR function.

M.SLICE†— TERMINATE TIME SLICE PERIOD

The format of the function is

M.SLICE,****,****,****,****

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.SLPER — INITIATE CENTRAL MONITOR IN OTHER CPU

The format of the function is

M.SLPER,****,****,****,***C

C CPU in which CPMTR is to execute next instruction.

0 Initiate CPMTR in CPU-A.

1 Initiate CPMTR in CPU-B.

This request is used by CPMTR to cause the next execution to take place in the opposite CPU of a dual CPU system. If CPMTR is executing in CPU-A and this request is made with C set, the next execution of CPMTR takes place in CPU-B. A possible use would be to terminate a job executing in CPU-B while CPMTR is executing in CPU-A.

M.SPM†— SPM CALL FROM ISP/ISQ

The format of the function is

M.SPM,PPPP,PPPP,PPPP,EX.xxx

The PPPP fields are subfunction parameters. EX.xxx is the subfunction to be performed as follows:

EX.SPRCL Stack processor recall. SPM is called to terminate the actual I/O portion of the current stack request. SPM will terminate, reissue, or otherwise further process the stack request and issue a new stack request or special order (O.IDLE, O.DROP, O.SEEK) to ISP/ISQ and complete the function. Call format is

M.SPM,****,****,****,EX.SPRCL

EX.STAT Changes status. SPM is called to change ISP/ISQ status in the DST and take appropriate action. DST status is

0 No PP assigned.

1 PP assigned but not ready; PPIR - PP assigned and ready.

† CPMTR function.

When PPMTR assigns ISP/ISQ to a PP, it changes the DST status from zero to one. After initialization, ISP/ISQ issues an EX.STAT with PPIR status. This call may be issued again later to activate SPM if ISP/ISQ is idle and the PPMTR stack processor drop flag is set. A pending EDITLIB or LDCMR will set a wait flag for ISP/ISQ. When ISP/ISQ is idle and encounters the flag, ISP/ISQ issues a status of one to SPM and reinitializes itself. When ISP/ISQ is idle and detects an outstanding channel request for its channel, it issues a status to zero to SPM to request a drop. SPM will then issue an O.DROP to ISP/ISQ so that ISP/ISQ can give up the PP. Call format is

M.SPM,0000,0000,SSSS,EX.STAT

<u>SSSS</u>	<u>Status</u>
0	Request drop.
1	Request inactive status (reinitializing).
PPIR	Request active status (initialized).

EX.NXTPB

Get next PB/PRU. This is a time-critical call made during the I/O transfer. This call is entered by ISP/ISQ into its PPOR just prior to starting transfer of the current PB/PRU chunk of data. While the current chunk is being transferred, PPMTR sees the PPOR call and initiates SPM. If the current transfer is a write, SPM will allocate more RBs, if needed. In any case, SPM then converts the current RB position in the RBT chain to a PB position and stores this in PPMES4, bytes 0 through 2 of the ISP/ISQ communication area. When ISP/ISQ completes the current PB/PRU transfer, it updates PPMES6 (current PB/PRU) from PPMES4 (next PB/PRU), issues the next EX.NXTPB call to the PPOR, and continues the transfer. Call format is

M.SPM,2,SSPP,PPPP,EX.NXTPB

<u>SSPP,PPPP</u>	<u>Successor Call Type</u>
PPPP is 0	No successor call (clear PPOR).
PPPP is not 0	} Set up M.BUFPTR call.
SS is 0	
PPPP is not 0	} Set up M.SCB call PPPP; successor call parameters.
SS is not 0	

M.SPRCL†— STACK PROCESSOR RECALL

The format of the function is

M.SPRCL,****,****,000F,CCCC

† CPMTR function.

CCCC Control point area address.

F Modification of the count of outstanding stack requests in W.CPSR.

<u>F</u>	<u>Description</u>
0	No adjustment.
1	Subtract one.
2	Add one.

M.TRACE — ENTER MONITOR TRACE MODE

The format of the function is

M.TRACE,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 provided for system debugging.

M.TSR — TERMINATE STORAGE REQUEST

The format of the function is

M.TSR,****,****,****,****

Request is valid if real-time monitor is installed (IP.RTMTR is nonzero); it terminates wait period involving an M.RSTOR request.

CYBER 180-CLASS MAINFRAME MAINTENANCE CHANNEL

Channel 17₈ of the CYBER 180-class mainframe provides access to the maintenance registers in order to sense the status of the system hardware. Each hardware subsystem element has a set of maintenance registers that contain error and status information for that element.

MAINTENANCE CHANNEL ERROR PROCESSING

Error processing can operate in either dedicated or nondedicated mode. In dedicated mode, DFT (the routine that accesses the maintenance registers) is loaded at the start of system operation and remains in that PP, continually testing for errors. When DFT encounters an error, it places the error data in a buffer and sets a bit in T.EPIB. As soon as MTR sees this bit set it loads 1MR to reformat the data, write it in the CERFILE, and initiate the appropriate system response.

In nondedicated error processing mode, MTR periodically tests the status summary bit (bit 59 of the IOU status summary register) which is set by the hardware whenever any error occurs. As soon as MTR sees the bit set, it loads 1MR which, in turn, loads DFT to get the data from the maintenance registers. After DFT puts the data in a buffer, it reloads 1MR to put the data into the CERFILE and initiate the appropriate system response.

SOFTWARE ACCESS TO MAINTENANCE CHANNEL

Operating system functions requiring access to the maintenance channel are distributed between PPMTR and 1MR. Routines using the channel must conform to the following rules:

- Maintenance Channel Reservation

The MCH flag must be reserved with the set channel flag (SCF) instruction before any maintenance channel I/O is permitted. If this flag is set when an SCF instruction is issued, a subsequent reservation attempt must be made. A routine should not set the channel flag more than once every 500 milliseconds.

- Maintenance Channel Release

The maintenance channel is released with the clear channel flag (CCF) instruction. All maintenance channel I/O must be completed, the channel must be empty and inactive, and the channel error flag must be cleared prior to releasing the channel. The reservation period must not exceed 125 microseconds.

FILES

A name associated with each file identifies it to the system and to the user. Files are uniquely known by the file name, source or destination ID, and the terminal ID (if applicable). Files are stored on either allocatable or nonallocatable devices. Rotating mass storage (RMS) units, such as disks, 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, and line printers are nonallocatable because files on these devices (and in fact the devices themselves) can be associated with only one control point 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, local, permanent, and queue (input and output). The following paragraphs describe briefly each of the file groups.

SYSTEM FILES

The following files are always in the system. They are always assigned to control point 0 and they reside on allocatable devices. These files, except for the job dayfiles, are maintained on system devices as permanent files. Each job dayfile exists only for the duration of the job.

<u>File</u>	<u>Description</u>
System	The file has the name of ZZZZZ04 and contains a copy of the deadstart system tape. After a system EDITLIB, the file ZZZZZ03 that contains the latest version of a library entry is also part of the system file.
System 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.
Job dayfile (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 0 and a user cannot access them directly. When a user job terminates, the content of the job dayfile is copied to the end of the job output file. A job dayfile contains images of all control statements 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.
ZZZZCMR	This file contains the absolute segments of CMR in an extended memory system. Depending on the options selected, this file is used by LDCMR at deadstart or when the system is reloaded.

<u>File</u>	<u>Description</u>
ZZZZZ06	If the installation parameter IP.ELIB is one, this file is created to contain the extended memory library.
ZZZZZ23	This file contains a copy of the CM resident library area. It is created at deadstart and is updated by EDITLIB. It is not permanently attached to control point 0 and is used only for deadstart recovery.

PERMANENT FILES

Permanent files are saved across deadstarts and are therefore considered 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.

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 nonallocatable 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 control point, disposition code, and device type on which it resides. For local files on nonallocatable 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. Files with the special names OUTPUT, PUNCH, PUNCHB, P80C, FILMPR, FILMPL, HARDPL, HARDPR, and PLOT are assigned nonzero disposition codes when created. All other files are assigned a zero disposition code when created. For local files with other disposition codes the file name is changed to the job name and the file is assigned to control point 0. A local file with the name PUNCH or PUNCHB is output to a card punch; the file OUTPUT is printed.

QUEUE FILES

To provide for the recovery of input and output files from disk tables on nonrecovery deadstarts, the input and output queues are kept as subsets of the permanent file system. All input files that enter the system via JANUS, INTERCOM, Remote Batch processor, or load tape (n.X TLOAD) are automatically cataloged. They are not purged until the job has completed execution and all output files of the job have been cataloged. When the permanent file catalog (PFC) becomes full, a message is issued to the operator and input halts until there is more space for files to be cataloged.

The file name table entry of queue files contains permanent file information and the file description parameters. 1TJ is the common routine for entering a file into the input queue called by JANUS, INTERCOM Import processors, and DSP for ROUTE(filename,DC=IN). 2VJ verifies the job statement parameters. 1QF is the PP routine which catalogs and purges queue files.

Input Queue

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 to indicate an input disposition code; the file is cataloged and released to control point 0 as an input file. The input queue consists of all files assigned to control point 0 with an input disposition code.

The system packages that read in batched local jobs make an effort to ensure that each job file in the system has a unique seven-character name. The job name from the job statement is truncated to the first five characters (or extended with zeros to five characters) and two unique sequence characters are added. All numerals and letters can be used as sequence characters; therefore, 1296 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. There are some circumstances which increase the possibility of duplicating names, such as: RESQ, level 0 or 1 deadstarts or system programs which control all seven characters of jobs placed in the input queue.

Output Queue

Output files originate from local files on allocatable devices; they have nonzero disposition codes. When a job terminates, such files are cataloged, assigned to control point 0, and given either the job name or the name in the file ID field of the FNT file routing supplement. These files then form a system output queue which is, essentially, a list of files waiting to be output to unit record equipment.

In each output queue file name table entry, fields define the destination of the file. The characteristics that are defined are device type, terminal ID, destination ID, external and internal characteristics codes, disposition codes, and forms codes.

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

- The user gives the file a special name. When a file with a special name is created, it receives a nonzero disposition code. These files are sent to the corresponding destination when the file is released for output processing. For example, the file named OUTPUT receives a print disposition code. A file named PUNCH receives a punch disposition code.
- The user can specify file disposition with a DISPOSE or ROUTE control statement or macro. The file can have any name. Files must reside on allocatable devices that are members of the queue set.

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 CLOSE,RETURN is performed. Since the name of an output queue file can be the name of the job which created it, and since a job can create several files which go into the output queue, names in the output queue often are not unique.

DAYFILE Macro

A system job using the DAYFILE macro has an additional capability that a user job does not. A system job can specify SYSTEM as the dayfile type, making the SYSTEM dayfile available as a local file. The DAYFILE control statement and macro are described for user jobs in the NOS/BE Reference Manual.

ROUTE Macro — Additional Capabilities

A system job using the ROUTE macro has additional capabilities that a user job does not. A system job can specify a source ID, a seven-character job name, routing information for queue files, and a predayfile file name. The other ROUTE capabilities are the same as for user jobs. The ROUTE control statement

and macro are described for user jobs in the NOS/BE Reference Manual. The remainder of this discussion describes only the differences for system jobs.

The parameter block that must be set up to use the additional capabilities for system jobs has the following format.

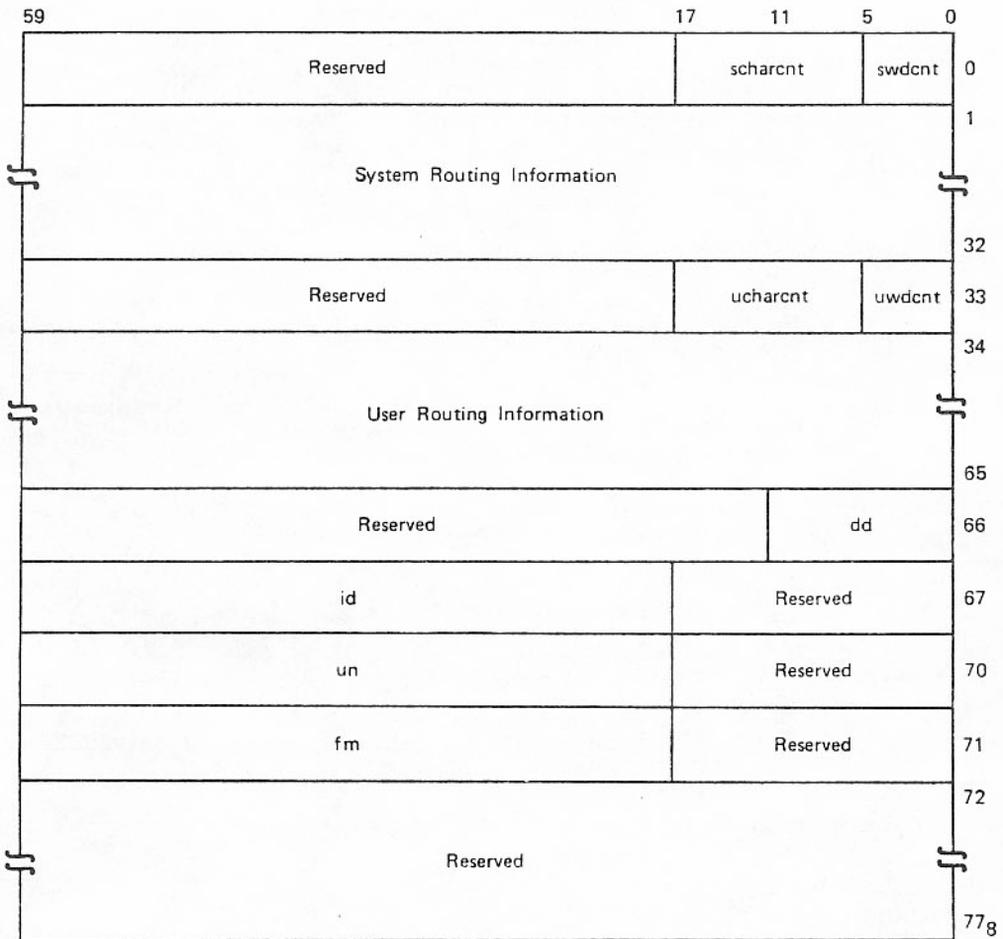
	59	53	47	41	35	23	21	17	11	0	
tag	File Name								Error Code	Unused	A
tag+1	0	0	0	0	Forms Code/ Input Flags	Disposition Code	E C	I C	Flags		
tag+2	Station ID- Source				Station ID- Destination			Unused		TID	
tag+3	File Identifier (FID)							Un- used	B	Priority	
tag+4	Spacing Code (Output Only)	Pre-dayfile File Name (Input File Only)						Repeat Count	Unused		
tag+5	Reserved							FWA of Routing Information Packet			

The fields in the parameter block are identical to those described for the ROUTE macro in the NOS/BE Reference Manual with the following exceptions.

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
tag+1	47-36	Forms code/input flags	Bits 44 and 46 are meaningful for files that are to be routed to an input queue. Setting bit 44 indicates that a file identifier (FID) of seven characters is specified in tag+3. Setting bit 46 indicates that the system is to preabort the job. These settings are ignored if the job is not a system job.
	17-0	Flag bits	Indicates which parameters are specified (in addition to the values given in the NOS/BE Reference Manual). <ul style="list-style-type: none"> 16 Routing information is specified. 13 Dayfile is attached for immediate routing to output. 11 Predayfile file name is specified. 5 System uses FID specified in tag+3, bits 59-18. Only system jobs can specify seven characters; user jobs specify five characters.
tag+2	59-42	Station ID-source	Three display code characters used as the source ID for a job routed to an input queue. When this field is binary 0, the routed file has no source ID. When DC=IN, the job's source ID is used as the setting of this field. A job's source ID is found in the control point area.

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
tag+4	59-18	Predayfile file name	Name of the file which contains the predayfile. This parameter is meaningful only for DC=IN.
tag+5	17-0	Routing information packet	FWA of the buffer that contains the extended parameter packet of routing information to be associated with the file named in tag.

The extended parameter packet must be in the following format.



<u>Word</u>	<u>Bits</u>	<u>Description</u>
0	17-6	Scharent is a count of the number of characters used in the system routing information block.
0	5-0	Swdent is a count of the number of words used in the system routing information block.
1-32		System routing information block. It contains information generated by the system that sent the file. The receiving system stores the information. When the receiving system returns any output that might result from a job file, it also returns the information in the system routing information block. The generating system uses the information to direct the output back to the user who submitted the job file. If the job running on the receiving system submits another job (regardless of destination), the information in the block is also sent along with that job. Then when the output from that second job is returned to the generating system, the information in the block is sent with it. Thus, the output from the second job is also returned to the user. The information in the system routing information block is also referred to as implicit text.
33	17-6	Ucharent is a count of the number of characters used in the user routing information block.
	5-0	Uwdent is a count of the number of words used in the user routing information block.
34-65		User routing information block. It contains information generated by the user who sent the file. The receiving system uses the information to place the file into the proper queue. The information in the user routing information block is also referred to as explicit text or as the user routing directive.
66	11-0	Dd is a two-character data description of the file. Dd indicates the character set and the structure of the file. Dd can have the values defined for it with the MFQUEUE command in the NOS/BE Reference Manual.
67-71	59-18	Id, un, and fm are supplied for compatibility with the NOS operating system. For documentation on their use refer to the MFQUEUE command in the NOS Reference Set, Volume 3.

ACQUIRE Macro

The ACQUIRE macro calls the PP routine QAF to search the input, print, punch, special (nonstandard) output, and execution queues looking for entries that satisfy given selection criteria. The user specifies one of four functions specified by a function code in bits 3 through 1 of word 0 in the ACQUIRE parameter list: ALTER, modify queue entries; GET, attach a file to the caller's control point; PEEK, return information about the queue entries; or COUNT, count the entries in the specified queue(s). QAF can be called only by a routine resident in the system library. The format of the ACQUIRE macro is

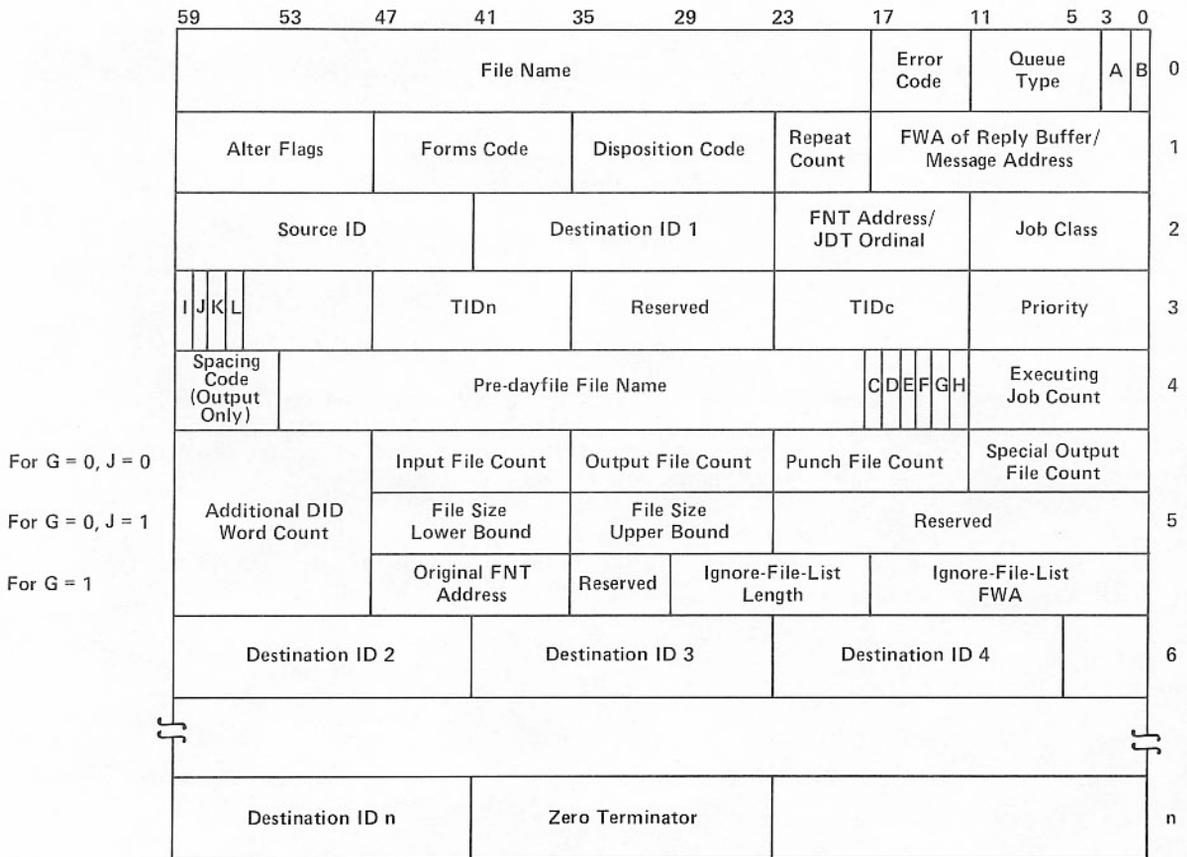
ACQUIRE addr,recall,N.

addr The address of the first word of the parameter list.

recall Optional parameter specifying auto recall.

N Required parameter to distinguish this new macro from an older version.

QAF requires the parameter list to be at least six words in length. The list can be longer if there are additional destination IDs. The additional length is specified in the additional destination ID word count field.



<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
0	59-18		File name. If the file name of a particular file is specified on any function, each FNT entry is examined until the specified file name is found. This file must also meet the specified criteria to qualify as the selected file.
	17-12		Error code (in octal).

<u>Code</u>	<u>Significance</u>
01	Invalid queue type.
02	No queue entry found with specified parameters.
03	Function prohibits 7777g priority.
04	No FNT space.
05	Invalid reply entry buffer address.
06	Internal QAF error on FNT address.
07	Illegal request.
10	Too many extra DID words.
11	PEEK requires single queue type.
12	Duplicate file name on GET.
13	Count of 0 is invalid.
14	LFN needed for file having predayfile.
15	Invalid FNT address/JDT ordinal.

11-4 Queue type. The queue type must be supplied on all functions except the special PEEK function when it must be 0. The binary values are: 00000001 (INPUT), 00000010 (OUTPUT), 00000100 (PUNCH) or 00001000 (special output), or 00010000 (execution).

INPUT file	Has a valid FNT entry, is unlocked at control point 0, and has a disposition code of 04g (INPUT job), 05g (INPUT tape job), or 06g (INPUT tape job on P display).
OUTPUT file	Has a valid FNT entry, is unlocked at control point 0, and has a disposition code of 40g (any 512 or 580 line printer), 42g (any 512), 43g (any 580-12), 44g (any 580-16), or 45g (any 580-20).
PUNCH file	Has a valid FNT entry, is unlocked at control point 0, and has a disposition code of 10g (PUNCH 026 set from display code).

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
			Special output file
			Has a valid FNT entry, is unlocked at control point 0, and has a disposition code of 20 _g (film print), 22 _g (film plot), 24 _g (hardcopy print), 26 _g (hardcopy plot), or 30 _g (plot).
			Executing job
			Has a valid JDT entry.
3-1	A	Function code (in octal).	
		0	ALTER
		1	GET
		2	PEEK
		3	COUNT
0	B	Complete bit. The complete bit must be cleared before any call to QAF. The bit is set on completion of any function.	
1	59-48	ALTER flags.	

<u>Bit Set</u>	<u>Significance</u>
53	Abort and/or evict job and/or file.
52	Change repeat count.
51	Change forms code; for other functions, means compare forms codes.
50	Change priority.
49	Change terminal ID to TIDn.
48	Send to central site.

47-36 Forms code.

35-24 Disposition code.

23 Not used.

22-18 Repeat count.

17-0 FWA of reply buffer or a message to be issued when aborting a job. This buffer is limited to 30 characters (ALTER).

or

FWA of buffer to which routing information packet is to be returned (GET). The format of the information returned is the same as that documented for the routing information packet with the ROUTE macro.

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>														
2	59-48		Source ID (SID).														
	41-24		Destination ID 1 (DID).														
	23-12		FNT address/JDT ordinal. This field in the parameter list is an absolute FNT address whenever a queue type other than job queue is specified. It is a JDT ordinal whenever only the job queue is specified. The field is an FNT address and is required on a special PEEK function and optional on all other functions.														
	11-0		Job class.														
			<table border="1"> <thead> <tr> <th><u>Bit Set</u></th> <th><u>Significance</u></th> </tr> </thead> <tbody> <tr> <td>5</td> <td>Graphics job.</td> </tr> <tr> <td>4</td> <td>Express job.</td> </tr> <tr> <td>3</td> <td>Multiuser job.</td> </tr> <tr> <td>2</td> <td>INTERCOM job.</td> </tr> <tr> <td>1</td> <td>Batch job with nonallocatable device requirement.</td> </tr> <tr> <td>0</td> <td>Batch job without nonallocatable device requirement.</td> </tr> </tbody> </table>	<u>Bit Set</u>	<u>Significance</u>	5	Graphics job.	4	Express job.	3	Multiuser job.	2	INTERCOM job.	1	Batch job with nonallocatable device requirement.	0	Batch job without nonallocatable device requirement.
<u>Bit Set</u>	<u>Significance</u>																
5	Graphics job.																
4	Express job.																
3	Multiuser job.																
2	INTERCOM job.																
1	Batch job with nonallocatable device requirement.																
0	Batch job without nonallocatable device requirement.																
3	59	I	Return the routing information associated with the specified file (meaningful only for the GET function). If you set this flag, but there is no routing information associated with the file, the system resets the flag to zero.														
	58	J	File size lower and upper bound selection specified for output, punch, and special output queues only. File size bounds may be specified only when ignore-file-list is not specified (G=0).														
	57	K	Dayfile attached to output file.														
	56	L	Ignore queued files with invalid local file names.														
	47-36		TIDn. New TID (used by ALTER only).														
	23-12		TIDc. Current Terminal ID (used for search).														
	11-0		Priority. If 0 priority is specified and the other criteria are satisfied, GET attaches the first file found; PEEK writes a reply entry for each file; and COUNT increments the file type count. If the priority is greater than 0 and less than 7777g and the other criteria are satisfied, GET attaches the first file having a priority greater than or equal to the specification; PEEK writes a reply entry for each file that has a priority greater than or equal to the specification; and COUNT increments the file type count when a file has a priority greater than or equal to that specified. If the priority is equal to 7777g, GET attaches the file having the highest priority among those that satisfy all requirements; PEEK and COUNT do not allow this priority and return an error code of 03g. ALTER does not use the priority as a search criterion. This value replaces whatever the entry had before if bit 50 in word 1 is set.														

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
4	59-18		Predayfile file name. The predayfile file name is required only if the qualifying INPUT file has a predayfile. If the file has a predayfile, a separate FNT entry is created to describe the predayfile entry. If the predayfile file name is not given, the complete bit is set and the error code 14 ₈ is returned. If a predayfile file name is always in the field, the predayfile flag must be checked after each call to prevent duplicating the file name in the FNT entries.
	59-54		Spacing code (output file only). The spacing code associated with the output file is returned on a GET function. It cannot be specified as a parameter but will be returned.
	17	C	Predayfile flag bit.
	16	D	Class 2 INPUT file inhibit flag. The class 1 and class 2 INPUT file inhibit flags are used to achieve selectivity in terms of file classes for GET, PEEK, or COUNT functions. A class 1 file has no nonallocatable files associated with it. A class 2 INPUT file has at least one nonallocatable file associated with it. If the job statement specifies MTxx or NTxx, a nonallocatable file is associated with the INPUT file. If the caller sets the class 2 inhibit flag on a GET call and also sets the INPUT file type bit, only class 1 INPUT files are returned.
	15	E	Class 1 INPUT file inhibit flag.
	14	F	Inhibit duplicate file name search flag. If the flag is 1, no search is made for a duplicate file name.
	13	G	Ignore-file-list specified. The calling program sets this bit to 1 to indicate an ignore-file-list has been specified in Word 5. For a GET function, this bit also causes QAF to return the acquired file's original FNT address in Word 5.
	12	H	When the caller sets this bit to 1, QAF ignores files with DIDs that match the host's physical or logical identifier (defined in the IDT).
	11-0		Executing job count.
5	59-48		Additional DID word count. The additional word count is used whenever more than one is needed in the parameter list. Additional DIDs are packed three per word and terminated by a byte of zeros. A maximum of 64 (decimal) is allowed. The count is the number of CM words required to hold the additional DIDs. It is not necessary to allocate an additional CM word simply to hold a terminating byte of zeros. To hold six additional DIDs, for a total of seven DIDs in the parameter list, the DID word count is two to three DIDs in the first word and three in the second.

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
5	47-0		The remainder of word 5 is interpreted in one of three ways, depending on the setting of the G selection flag (word 4, bit 13; ignore-file-list) and the J selection flag (word 3, bit 58; file size bounds).

For G=0 and J=0:

<u>Bits</u>	<u>Description</u>
47-36	INPUT file count.
35-24	OUTPUT file count.
23-12	PUNCH file count.
11-0	Special output file count.

Only one of the four file count fields can be specified on a PEEK function; the other fields are not used. File type determines which file count field is to be used.

For G=0 and J=1:

<u>Bits</u>	<u>Description</u>
47-36	File size lower bound (FSLB)
35-24	File size upper bound (FSUB)

The file size bounds specify an inclusive range of file sizes, in units of 64 (decimal) PRUs, for output, punch, and special output queue files. For a file in one of these queues to be selected, its size must satisfy the following:

$$\text{FSLB} \quad (\text{file size}/64) \quad \text{FSUB}$$

Results are undefined if FSLB is greater than FSUB.

For G=1:

<u>Bits</u>	<u>Description</u>
47-36	Original FNT address. For a GET function, QAF will return the acquired file's original FNT address in this field. (The original FNT address identifies the file if it is returned to the I/O queue).
29-18	Ignore-file-list length. This field contains the length (in CM words) of the ignore-file-list established by the calling program. The length may be zero, and must not exceed 203 (decimal).
17-0	Ignore-file-list FWA. This field contains the first word address of the ignore-file-list established by the calling program. The list consists of continuous 12-bit FNT addresses (5 per CM word, scanned left to right) that QAF is to ignore when it is searching the FNT for any function. An address of zero ends the list, overriding specified length. QAF does not validate the FNT addresses.

<u>Word</u>	<u>Bits</u>	<u>Field</u>	<u>Description</u>
6	59-42		Destination ID 2.
	41-24		Destination ID 3.
	23-6		Destination ID 4.
	5-0		Not used.
n	59-42		Destination ID n.
	41-24		Zero terminator.
	23-0		Not used.

Summary of Parameter List Usage

The required and optional parameter list field usage for each QAF function and the corresponding returned parameters are shown in table 4-1.

TABLE 4-1. ACQUIRE MACRO PARAMETERS

Field Name	Field Usage							
	ALTER		GET		PEEK		COUNT	
	Call	Return	Call	Return	Call	Return	Call	Return
File/job name	O	X	O	X	-	X	-	X
Error code	-	X	-	X	-	X	-	X
Queue type	R	X	R	X	R	X	R	X
QAF function code	R	-	R	-	R	-	R	-
Complete bit	R	X	R	X	R	X	R	X
ALTER flags	R	-	-	-	-	-	-	-
Forms code	O	-	O	-	O	-	O	-
DISP code	-	-	O	X	O	X	O	X
Repeat count	O	-	-	X	-	X	-	-
FWA reply buffer	O	-	-	-	R	-	-	-
Source ID	O	-	O	X	O	X	O	X
DID 1	O	-	O	X	O	X	O	X
FNT add/JDT ord	O	X	O	X	O/R††	X	O	X
Terminal IDn	O	-	-	-	-	-	-	-
Terminal IDc	O	-	O	-	O	-	O	-
Priority	O	-	R	X	R	X	R	X
Spacing code (SC)	-	-	-	X	-	-	-	-
Predayfile file name	-	-	-/R†	-	-	-	-	-
Predayfile flag	-	-	-	X	-	X	-	X
Class 2 inhibit	O	-	O	-	O	-	O	-

TABLE 4-1. ACQUIRE MACRO PARAMETERS (Contd)

Field Name	Field Usage							
	ALTER		GET		PEEK		COUNT	
	Call	Return	Call	Return	Call	Return	Call	Return
Class 1 inhibit	O	-	O	-	O	-	O	-
Ignore-file-list specified	O	-	O	-	O	-	O	-
Ignore local DID	O	-	O	-	O	-	O	-
Executing job count	-	-	-	-	R/-††	X	-	X
Add DID word count	O	-	O	-	O	-	O	-
Input file count	-	-	-	-	R/-††	X	-	X
Output file count	-	-	-	-	R/-††	X	-	X
Original FNT address	-	-	-	X	-	-	-	-
Punch file count	-	-	-	-	R/-††	X	-	X
Special out file count	-	-	-	-	R/-††	X	-	X
Ignore-file-list length	O	-	O	-	O	-	O	-
Ignore-file-list FWA	O	-	O	-	O	-	O	-
DID 2	O	-	O	-	O	-	O	-
DID n	O	-	O	-	O	-	O	-
Routing information	-	-	O	X	-	-	-	-
File size lower bound	-	-	O	-	-	-	-	-
File size upper bound	-	-	O	-	-	-	-	-
Dayfile attached	-	-	-	X	-	-	-	-
Skip invalid file name	-	-	O	-	-	-	-	-

Explanation of symbols:

- O Optional parameter.
- X Parameter returned by QAF.
- Parameter not used.
- R Required parameter.

†Required only if predayfile is present.
 ††Second symbol is for the special PEEK function.

ALTER

ALTER, function code 0, gives the user the ability to change various fields within the queue entries that match the selection criteria specified in the parameter list. Required parameters are queue type and the ALTER flag bits which indicate the actions to be performed. Optional parameters are queue entry name, address of an abort message, source ID, destination ID(s), and terminal ID. The forms code and priority fields contain the new values and, thus, cannot be used as a search criteria.

The actions that can be performed are:

- Change routing of INPUT and/or OUTPUT queue files to the central site.
- Change routing of INPUT and/or OUTPUT queue files to another terminal.
- Change priority of OUTPUT queue files.
- Change forms code of OUTPUT queue files except for nonstandard output (PLOT, FILM, and so forth).
- Change repeat count of OUTPUT queue files except for nonstandard output.
- Abort/evict queue entries and issue supplied error message.

The bits that indicate these actions may be set in any combination, but certain combinations are mutually exclusive. For example, if the first two actions are specified, the result is as if only the first action had been specified. Similarly, the last action overrides all other actions.

The user may also set the queue type bits in any combination but the combinations used when aborting a job can make a difference. For example, if all queue types are specified, the job is killed rather than dropped.

GET

GET, function code 1, selects the file that best meets selection criteria and attaches it to the control point of the calling routine. Required fields are function code, priority, file type, and a zeroed completion bit.

Before a file is attached, a search is made to ensure that no file having the same name is already attached. If a duplicate file is found, an error code of 12g is returned and the completion bit is set. The search for a duplicate file name can be suppressed by setting the inhibit search flag.

If the L bit is set in word 3 of the parameter list, QAF skips over a queue file if its name is not a valid local file name. If file size selection is specified (the J bit), queue files in the output, punch, and special output queues must satisfy the size criteria to be eligible for selection.

When the selected file is attached, an FNT supplement of type 0101g (if an input file is attached) or 0102g (if the attached file is output) is created and linked to the base FNT. The control point number of the job is written into the FNT. When the file is returned by the calling job, the FNT supplement is erased.

After the file is attached, the complete bit is set to one, the file name and FNT address are inserted, and the source ID and the destination ID are entered. Should no file satisfy all the selection criteria, the complete bit is set to one, the FNT address is zeroed, and an error code of 02g is returned.

Using the I flag in the parameter list, the user can specify that any queue file routing information associated with the file is to be returned to the reply buffer specified in word 1 of the parameter list.

PEEK

PEEK, function code 2, creates a list of three-word reply entries built from the queue entries matching the selection criteria. Required fields are function code, priority, zeroed completion bit, the first word address of the reply buffer in the user's field length where the reply entries are returned, the queue type count of the number of reply entries to be returned, and the queue type. Only one queue may be specified in the queue type. Optional queue entry selection criteria also include the starting FNT address or JDT ordinal from an earlier PEEK request for the same queue.

PEEK begins examining the FNT entries at the point specified by the FNT address or at the start of the FNT if no address is provided. For each file that matches the file selection criteria, a three-word reply entry is built from the file's FNT. The reply entry is placed in the reply buffer, and the file type count is incremented by one. PEEK continues searching until the requested number of reply entries is found or the end of the FNT is encountered. The function works in a similar manner for the execution queue using an optional starting JDT ordinal.

On return to the calling routine, the reply buffer, beginning at the first word address specified, contains the three-word reply entries. The count field for the queue type requested contains the number of reply entries built. The count is either the number requested or the number of entries built upon reaching the end of the FNT or JDT. For example:

A user calls the QAF PEEK function to obtain 20 input queue reply entries for files having a destination ID of ABC. The search is to begin at FNT address 4420g, with reply entries stored in the user field length, beginning at REPBUF. QAF begins searching the FNT at FNT address 4420g looking for input queue having a destination ID of ABC. Assuming that only 15 entries are found before reaching the end of the FNT, the file count is set to 15, the FNT address is set to 0, and REPBUF contains 15 three-word input queue file reply entries built from the FNT of the 15 qualifying files. If 20 entries are found with the last qualified at FNT address 4730g, the FNT address is set to 4733g, ready to begin the next search, the input queue file count remains 20, and REPBUF contains 20 three-word reply entries.

A special PEEK function is defined with the file type field zero. The caller may check a particular FNT entry at the address specified to determine whether or not the entry matches the file selection criteria. Required fields are function code, a zeroed completion bit, the queue type field cleared, the first word address of the reply buffer, and the FNT address of the file. Optional parameters are any of the file selection criteria.

If the file at the specified address qualifies, the queue type, the complete bit, priority, and so forth are inserted into the fields and a single three-word reply entry, built from the FNT entry, is placed in the reply buffer. If the file does not qualify, the complete bit is set, an error code of 02g is inserted, and the queue type field remains clear.

Format of the three-word reply entry for an input queue file is

59					47		41		35		29		23		11		0	
File Name														Priority				0
Source ID					Destination ID					Reserved			FNT Ordinal				1	
Job Dependency ID				Dependency Count			Maximum MT Drives		Maximum NT Drives		Reserved			Terminal ID				2

Format of the three-word reply entry for an output queue (print, PUNCH, or special output) file is

59					53		41		23		11		0					
File Name														Priority				0
Source ID					Destination ID					Forms Code			FNT ordinal				1	
Repeat Count		Disposition Code			Size of File (Words/1000g)				Reserved			Terminal ID				2		
└ File Interrupt Bit																		

Format of the three-word reply entry for an execution queue entry is

59					41		36		34		29		23		17		11		0	
Job Name														Not Used		Priority				0
Source ID					Time					FL/100g			Job Ordinal				1			
Operator Action Codes				Error Flags		Type	CP	Job Status		Reserved			Terminal ID				2			

<u>Word</u>	<u>Bits</u>	<u>Description</u>
0	59-18	File name/job name.
	17-12	Not used.
	11-0	Priority.

<u>Word</u>	<u>Bits</u>	<u>Description</u>
1	59-42	Source ID.
	41-24	Destination ID/time left for execution in seconds.
	23-12	Forms code for output; not used for input; job field length/100g for execution.
	11-0	FNT ordinal/JDT ordinal.
2 INPUT	59-48	Job dependency ID.
	47-36	Dependency count.
	35-30	Maximum number of seven-track drives to be assigned at one time.
	29-24	Maximum number of nine-track drives to be assigned at one time.
	23-12	Reserved.
	11-0	Terminal ID.
2 OUTPUT	59	One means file interrupted.
	58-54	Repeat count.
	53-42	Disposition code.
	41-24	Size of file (words/1000g).
	23-12	Reserved.
	11-0	Terminal ID.
	59-42	Operator action codes (SCOPE 2 only).
2 EXECUTION	41-36	Error flag values (in octal):
	10	Kill.
	4	Drop.
	2	Rerun.
	35	Type of job (0 = 7600 or CYBER 70 Model 76; 1 = all others).
34-30	Control point number the job currently occupies.	

<u>Word</u>	<u>Bits</u>	<u>Description</u>
	29-24	Job status (in octal):
	70	Waiting for MMF action.
	60	Waiting for pack mount.
	40	Waiting for operator action.
	30	Waiting for tape/device assignment.
	20	Waiting for permanent file.
	10	Waiting for time/event.
	02	Executing.
	23-12	Reserved.
	11-0	Terminal ID.

COUNT

COUNT, function code 3, counts the number of queue entries of a specified type satisfying the selection criteria. Multiple queue type bits can be set on a single call giving the caller the count of each queue type desired. Required fields are queue type, function code, a zeroed completion bit, and priority. Optional fields are the rest of the file selection criteria.

The counts of the queue types specified are returned to caller, and the complete bit is set. The file name, file type, disposition code, source ID, destination ID, FNT address, and priority are returned for the first file that satisfies the selection criteria.

I/O TABLES

The input/output file requirements are coordinated 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 are kept in the following tables: file environment table (FET), file name table (FNT), file information table (FIT), and record block table (RBT). The FET and FIT are created within the job field length; the other tables are CM resident. The CM resident tables are in the upper table area of CMR, except the RBT which resides at the highest address of CM. Detailed descriptions of CM resident tables appear in appendix B. The FET is detailed in appendix C. The FIT is described in the CYBER Record Manager reference manuals listed in the preface.

PAGES 4-20 AND 4-21 HAVE BEEN REMOVED

File Environment Table (FET)

Every file for which I/O is to be performed must have a FET. Each FET consists of a basic five-word entry followed by additional words; the form depends on the type of I/O to be performed.

The basic five-word FET entry is as follows:

	59	47	35	32	23	17	13	8	1	0	
FET 0	File Name in Display Code						Record Levels	Error Flags	Code/Status	M	C
FET 1	Device Type	Flag Bits	Disposition Code	LFET -5	FIRST						
FET 2	0						IN				
FET 3	0						OUT				
FET 4	FNT Pointer	RB Size	PRU Size	LIMIT							

M File parity (mode); meaningful only for tape files:
 0 Coded (even)
 1 Binary (odd)

C Completion bit.

LFET -5 Length of FET minus five words for basic entry.

Buffer parameters
 $LIMIT \leq FL$
 $OUT < LIMIT$
 $IN < LIMIT$
 $OUT \geq FIRST$
 $IN \geq FIRST$

	59	47	41	35	29	23	17	0
FET 5	fwa Working Storage			lwa+1 Working Storage				
FET 6	UBC				MLRS			
	Record Request/Return Information							
FET 7	Record Number		Index Length			Index fwa		

Random/Indexed Files (Mass Storage) ———→
 Sequential Files (tape) ———→

FET 5 is used for input/output blocking/deblocking by CPC.

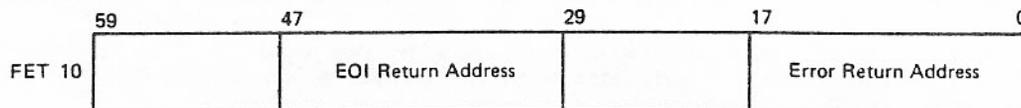
FET 5 and FET 6 are used for S and L tape file processing.

FET 6 and FET 7 are used for indexed file processing by CPC; FET 6 is used to pass RMS address between CP programs and system PP input/output routines.

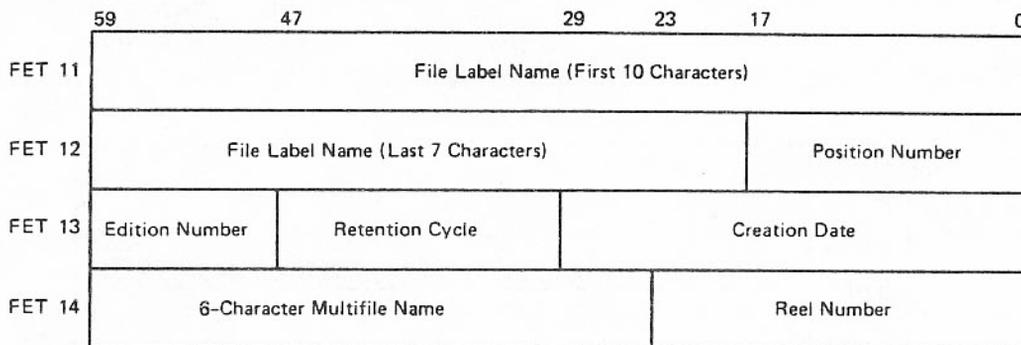
UBC Unused bit count.

MLRS Maximum logical record size (S/L tapes only).

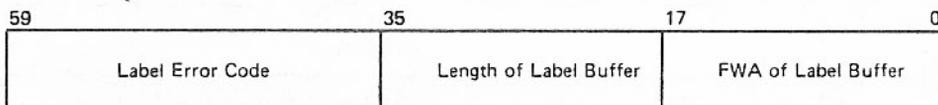
When the UP and/or EP flags are set in FET 1, then FET 10 contains



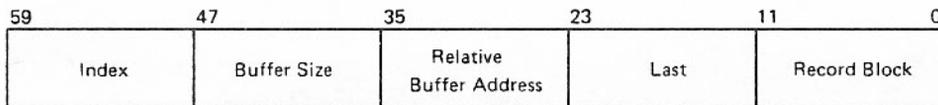
When standard file labels are to be written, the following FET words are filled with information from the LABEL control statement or macro. When a labeled file is read, the fields contain data read from the label.



When LFET-5 flag in FET 1 is set to 1 for extended label processing, FET 11 has the following format:



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 0 dayfile buffer. The format of the one-word dayfile FET entries is



Buffer sizes are set by the operating system assembly configuration parameters internal to CMR. The origin address of each buffer is calculated by adding the relative buffer address to the T.DFB origin address of the dayfile buffer area in CMR. The current position within the buffer is determined by adding the index value to the buffer origin address. The field labeled last (byte 3) contains the value of the index when the buffer was last flushed to disk. For the system dayfile and hardware error file, the record block field contains the end-of-information record block when the file was last extended. For control point dayfiles, the record block field is not used.

The first five words of the control point 0 dayfile buffer are preset as follows (b represents a blank):

```

b D A Y F I L E ::
b b b N O R M A L b
( b b b b b b b b )
D E A D b S T A R T
: : : : : : : :

```

The system dayfile area in CMR is diagrammed as shown in figure 4-1.

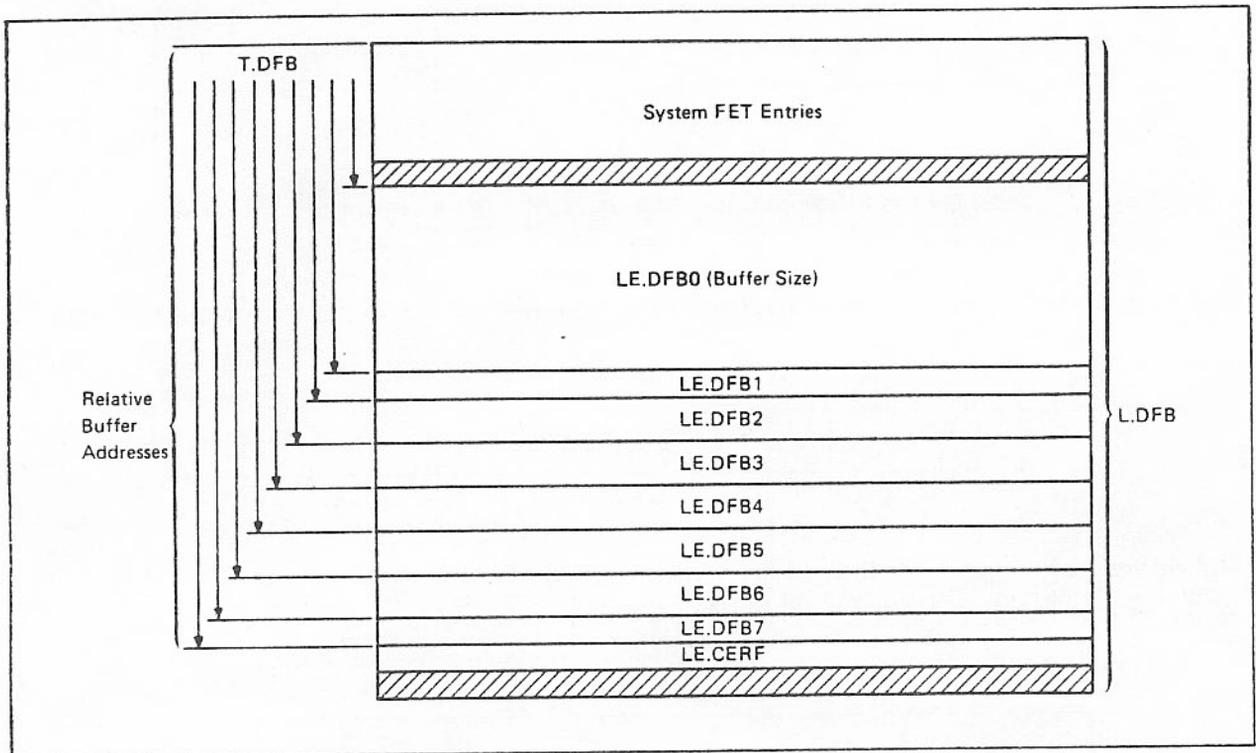


Figure 4-1. System Dayfile Area

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 figure 4-2.

T.FNT		59	53	47	35	23	17	11	0			
		Z	Z	Z	Z	Z	0	4	000000	0	Priority	
Device Type	0	FWA of RBT Word Pair				Current RBT Word Pair		Current RBT Ordinal		Cur. Byte	Current PRU	Status
0		0				7 4		0 0		0	0	01
		D	A	Y	F	I	L	E	000000	0	0	01 = Inactive
0		0				7 4		0 0		0	0	01
		D	F	I	L	E	0	1	000000	0	0	
0		0				7 4		0 0		0	0	01
		D	F	I	L	E	0	2	000000	0	0	
0		0				7 4		0 0		0	0	01
		D	F	I	L	E	0	3	000000	0	0	
0		0				7 4		0 0		0	0	01
		D	F	I	L	E	0	4	000000	0	0	
0		0				7 4		0 0		0	0	01
		D	F	I	L	E	1	5	000000	0	0	
0		0				7 4		0 0		0	0	01
		C	E	R	F	I	L	E	000000	0	0	
0		0				7 4		0 0		0	0	01
		Z	Z	Z	Z	Z	0	3	000000	0	0	
0		0				7 4		0 0		0	0 0 0 1	
		Z	Z	Z	Z	Z	0	6	010000	100100100100	LE.FNT-1 Link Address Extended Memory Library Entry If IP.ELIB ≠ 0	
Device Type AX	20	0				7 4		4 0		0	0 0 0 1	
Link Address	0	0	0	7	7	4	4	4	1	0	0	

Figure 4-2. System File Entries

File Name Table (FNT)

To provide linkage between user programs and all I/O processing routines, the FNT is maintained 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 six or nine words in length. The first word contains the 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 information; the format differs depending upon the type of file and where it resides. The various forms of the FNT entry are detailed in appendix B. The second and third words of the FNT entry are often called the file status table (FST) entry.

DEVICE TABLES

Tables in CMR that provide information on input/output equipment and channels are used by the operating system to make file assignments. Tables included in this section are the 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 on each record block in a mass storage device; the dismountable device table (DDT) and mounted set table (MST) containing information related to the recording surfaces. 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 device pool table (DPT), and the INTERCOM table (ITABL). These tables are detailed in appendix B.

Equipment Status Table (EST)

The EST resides in the upper table area of CMR and is pointed to by P. EST in the CMR pointer area. Table length depends on installation parameters. 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, and so on.

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 xxx in the operator command n.ASSIGNxxx. Only RMS devices may have EST ordinals greater than 77g.

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 RBR, linking that table to the allocatable device it describes.

Dismountable Device Table (DDT)

The DDT is used to maintain the status of all rotating mass storage devices. The fixed section of the DDT is used to relate the status of an RMS drive to the status of the pack mounted on that drive. The variable section of the DDT is used to store pack requests that have not been satisfied. The second word of a fixed section entry has a pointer to the EST entry for the drive. Whenever the physical status of the drive changes, the EST is updated. IRN compares the status bits in the EST with the status bits in the DDT and calls IPK when a difference is detected. IPK updates the DDT to reflect the new status of the drive and checks the variable section of the DDT to see if any pack requests can be satisfied. If a requested pack

has been mounted, 1PK updates the fixed section to include the DAM and MST ordinals, deletes the variable section entry, and recalls the job that had requested the pack. When a new pack request is made, 1PK checks the DDT to see if the device is already mounted. If it is mounted, 1PK satisfies the request. If the pack is not mounted, 1PK makes an entry in the variable section of the DDT and swaps the job out.

Mounted Set Table (MST)

The MST is used to keep pointers for each mounted device set in the system. Each MST entry has a corresponding set subdirectory table entry in the FNT. Entries are made by MNT when a master device is mounted and deleted by DSM when a master device is dismounted.

Device Status Table (DST)

The stack processor uses the DST 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 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 shown in appendix B.

The DST format reflects the new SPM/ISP working relationship and the multiple access approach. The first word contains multiple access information and the request stack chain pointer. The second word is the stack processor input register. It contains the DST ordinal used by SPM when ISP calls, equipment and channel numbers, and PPIR activity pointer (used only as part of the DST).

The DST is a key table in the processing of allocatable storage files. DST ordinals are found in the DAT, RBR table header, and 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.

Multiple access uses a DST master entry with DST multiple access memory entries. SPM assigns and tracks each ISP independently at each monitor call. Each ISP works on only one stack request at a time. All ISPs operate independently of each other and are unaware of any other ISP activity.

Device Activity Table (DAT)

The DAT 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 device to assign to a new or overflowing file.

The format for DAT entries are shown in appendix B.

Channel Status Table (CST)

The CST 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.

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.

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.

Details of CST are given in appendix B.

Refer to figure 4-3 for tables related to file processing on nonallocatable devices.

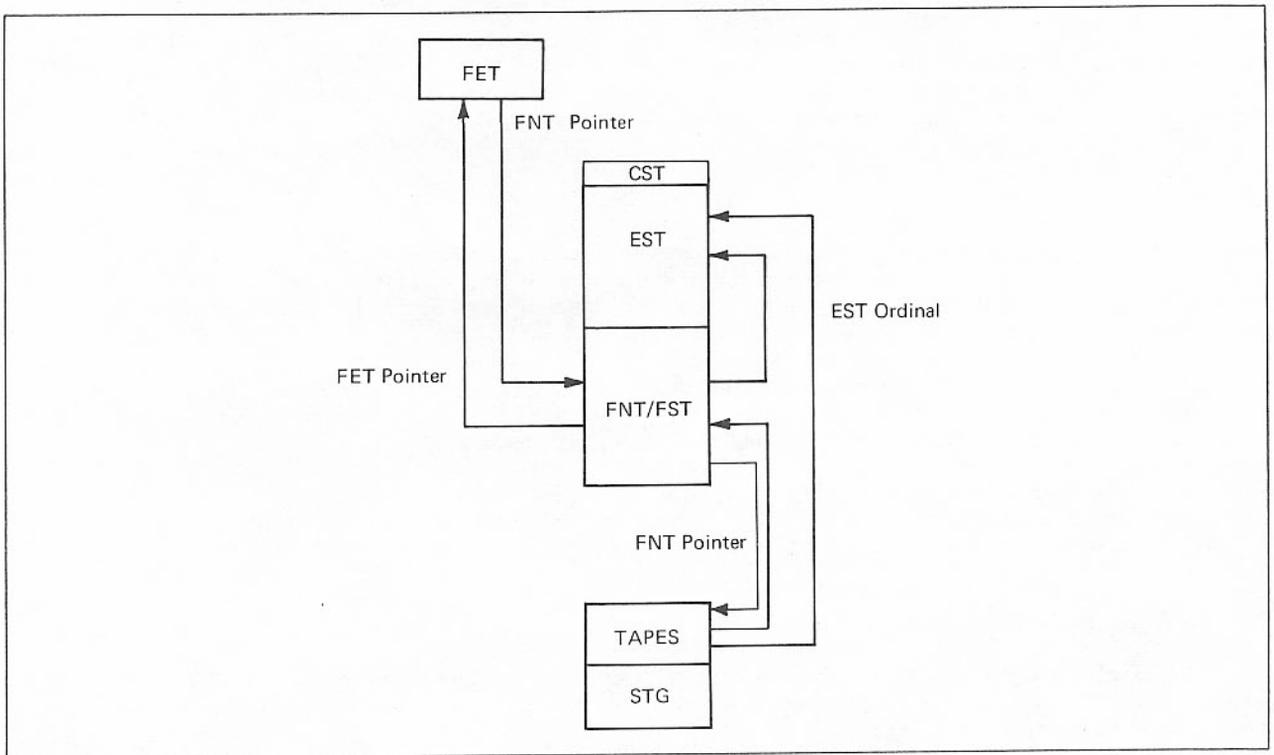


Figure 4-3. Nonallocatable Device File Processing

Tapes Staging Table (STG)

A satisfied job has all the tapes requested on its job statement. Unfilled demand is the sum of the job statement reservations of active jobs, less the tapes assigned to them.

The NO TAPE STATUS FLAG makes it possible to issue tape channel functions through DSD without interference from tape status processing. Status processing is not performed if the byte is nonzero. Normal system operation resumes when the byte is zeroed.

Three clocks are used to make event triggers for automatic assignment.

The STG is described in appendix B.

TAPE DRIVE SCHEDULING

Tape drive scheduling improves overall system throughput, particularly as it relates to tape job setup and execution. Automatic assignment, prescheduling, and overcommitment options are controlled by the value of IP.TSG.

AUTOMATIC TAPE DRIVE ASSIGNMENT

ANSI tape labels include a volume serial number (VSN) field. The user can have tape drives assigned automatically to his ANSI-labeled tapes by specifying the VSN on a VSN statement, in the REQUEST function, or as a parameter on the REQUEST or LABEL control statement. The VSN statement relates the external sticker or VSN to the file name and also provides information required for the tape job prescheduling display. When used with the REQUEST or LABEL control statements or the REQUEST function, it relates a VSN to a file name, which is relevant to equipment assignment. By itself, however, the VSN serves no purpose. When a VSN control statement provides the first reference to a file, a dummy FNT entry is set up using equipment code 64. If no subsequent REQUEST or LABEL control statement or REQUEST function provides additional information about the file, CIO finds the 64 equipment code in the FNT entry, releases that entry, and creates a default disk file. This feature does not encroach upon automatic assignment by label. The VSN parameter declares the tape label as either type U (full ANSI-standard label) or type Z (SCOPE 3 nonstandard label). The Z labels are not ANSI standard because the recording density field (character 12 of the volume header label) is not standard.

For automatic assignment of unlabeled tapes, the VSN must be entered by the operator. The tape is then assigned automatically to all jobs naming its VSN. Y-labeled tapes do not contain VSN information; however, to achieve automatic tape assignment, the operator can enter a VSN for a Y tape through the console. No automatic assignment is provided for 2MT or 2NT parameters.

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 any job which contains one or more of the tape parameters MT, NT, HD, PE, or GE on its job statement. All incoming tape jobs are entered in a prescheduling queue, a subset of the input queue. The purpose of a prescheduling queue is to advise the operator of tape reel requirements and to keep jobs from being processed until the required tapes 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 commands 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 command. Once released, the job is considered for assignment to a control point and execution; it no longer appears 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 and that most processing uses fewer than the maximum number of drives requested. Therefore, a job is assigned only the drives it needs to continue execution at any instant in time; excess drives, at that instant, are made available to run other jobs. Such a job scheduling algorithm permits the total tape requirements of all active jobs to exceed the total number of drives in the installation. However, a system deadlock could occur if two or more jobs have unfilled tape demands, such that every tape drive is assigned but no job has enough tapes to run to completion. Such a deadlock could be broken only by rerunning or killing one of the competing jobs. Although the job scheduling algorithm includes some built-in deadlock prevention features, preventing deadlocks is a function of tape assignment, not job scheduling.

As part of REQUEST processing, a deadlock prevention algorithm is provided. A potential deadlock exists if at least two jobs have unsatisfied tape requirements and the number of free tapes is less than the maximum required to satisfy any one job. The deadlock prevention algorithm refuses any tape assignment (manual or automatic) if such assignment would create a potential deadlock. Tape jobs could be scheduled at random without regard to tape drive availability and the deadlock algorithm would prevent deadlocks, but the resulting refusal of tape assignments would cause operator confusion and loss of efficiency.

Depending on the installation option to enable or disable scheduling by density (IP.SCHDE), deadlocks involving 679 tape units may occur. If the option to schedule by density is disabled (IP.SCHDE = 0), the system assumes all nine-track tape units have the same recording capabilities. However, models 679-2/3/4 tape drives are capable of 800/1600-cpi density operations while models 679-5/6/7 tape drives are capable of 1600/6250-cpi density operations. Without careful scheduling, a single job can cause a deadlock. For example, assume an installation has two 800/1600-cpi 679 tape units and two 1600/6250-cpi 679 tape units. A job requires two tapes recording at 1600 cpi and one tape at 800 cpi. If the two tapes requiring 1600 cpi are assigned to the 800/1600-cpi units, the tape requiring 800 cpi cannot be assigned. Procedures to resolve schedule deadlocks involving a mixture of 679 tape units and other nine-track tape units are described in part I of the NOS/BE Installation Handbook.

If the option to schedule by density is enabled (IP.SCHDE = 1), nine-track tape units are scheduled by the system according to the density parameters specified on the job statement.

DYNAMIC TAPE DRIVE STATUS CHECKING

Information concerning the physical status of tape drive units is entered into the TAPES table and updated by periodic checks of unassigned units for a ready/not ready status. This information is displayed in the top half of the P display. The period for status checking is set by the installation; it must be short enough to preclude the possibility of an operator dismounting a tape from a tape drive and mounting another without detection. Such periodic checking of unassigned tape drives makes automatic assignment more efficient and flexible.

Initially a tape drive is set to not-ready status as noted in the TAPES table. When a drive is made ready, the TAPES table is updated with information from the tape label. (If the tape is unlabeled, this fact is noted in the table.) A search is 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 applies to both labeled tapes and tapes qualifying as scratch.

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 is automatically assigned to the requesting job and the job is rolled back into CM to continue processing. While the job is rolled out, the operator can make a manual tape assignment which causes the job to be rolled in automatically.

Dynamic tape drive status checking permits the automatic assignment of unlabeled tapes by VSN. 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 can be assigned without operator intervention by any job requesting that VSN.

RMS SET TERMINOLOGY

Every RMS device (disk) is a member of a group of devices known as a device set. Such sets can be either public sets or private (user) sets. All disks used in the operating system are divided into sets. The term disk includes fixed disks and removable packs and is distinct from a drive which can hold different disk packs at different times. A set is an independent group of disks; a disk belongs to only one set, and files do not overflow to another set.

PUBLIC SETS

Public sets remain mounted at all times and have either permanent file default, system, queue, or scratch attributes, or a combination. These can all be combined in one set. Also, the members individually have SYS, PF, and Q attributes to further delimit file allocation.

Members of public sets cannot be dismounted; however, empty members can be deleted by DELSET, and new members added by ADDSET.

The system set is used for the system file and its related files created by post-deadstart use of EDITLIB and LDCMR, and the dayfile and CERFILE. There is no parameter on the REQUEST statement to specify system. The user can request the system set and VSN by name.

The PF default set is assigned when a file requests PF and no setname; only the PF default set is consulted on an ATTACH when no SN (setname) is supplied.

The Q set is assigned for special name files such as OUTPUT, PUNCH, and so forth; these files cannot be assigned to another set. Deadstart consults only the Q set to retrieve the queues. If a file is to be moved via DISPOSE or ROUTE, it must first be assigned to the Q set with a REQUEST(filename,Q) request.

Scratch sets are unlike the other sets as several sets may have the scratch attribute. Files not assigned by REQUEST and not special-named (OUTPUT, and so on) are assigned to a scratch set.

Each public set is assigned one or more of the following set attributes:

System set	This set contains system files such as ZZZZZ04, ZZZZZ23, the system dayfile, and the C.E. diagnostic file.
Permanent file default set	This set contains permanent files for which an alternate device set is not explicitly assigned.

Queue set	This set contains the INPUT, OUTPUT, and PUNCH queue files.
Scratch set	This set contains nonpermanent files for which a device set residence is not explicitly assigned.

Set attributes are assigned at disk initialization time. All four set attributes must be assigned for each mainframe. Only the scratch set attribute can be assigned to more than one public set on a mainframe.

Every device in a public set can (but need not) be assigned one or more of the device attributes listed below.

System device	This device can contain the system files given previously for the system set attribute. The system device attribute can be assigned only to public devices that are members of the system set.
Master device	The master device contains system tables relating to its device set. These tables include the device label, the PFD, the PFC, the SMT, the DAM, the PFT, and the LFT. Every device set must have a master device.
Permanent file device	This device can contain files for which the REQUEST control statement specifies PF.
Queue device	This device can contain files with names such as INPUT, OUTPUT, and PUNCH, file with nonzero disposition codes, and files for which the REQUEST control statement specifies Q. This device attribute cannot be assigned to devices within a private set.

The system device attribute is assigned at deadstart; the master, permanent file, and queue device attributes are assigned when the device set is created. Attributes assigned to devices (except for the system device attribute) need not match the attributes assigned to the sets of which they are members.

PRIVATE SETS

Any user may own a set of removable disk packs (a private set). Private sets are removable and mountable by job requests and operator action. Each member is mounted as needed, and members (other than the master) may be dismounted by operator command at any point in processing; masters may be dismounted when no jobs reference them. All members of a private set must be the same device type such as all 844-21 disks or all 844-41 disks; a combination of device types is not allowed.

A private device set is a group of RMS devices that can contain permanent files and be logically and physically removed from a running system. Permanent files stored on a private device set can therefore be transferred from one computer to another without moving the entire system. Set attributes cannot be assigned to private device sets. Devices within a private set can be assigned master device and permanent file device attributes. System device and queue device attributes are prohibited in private sets.

SHARED DEVICE SETS

In a dual-mainframe system, certain sets can be shared between mainframes. Such sets must consist entirely of 844-21, 844-41, or 885 devices. They cannot have the system set attribute. When a set is shared, all devices within that set are shared. Devices can be shared at either the unit or the controller level. The system uses the hardware reserve feature to reserve access to critical tables during an update; consequently, only one mainframe can access a device during an update.

A pool of free space is maintained in the RBR of each mainframe sharing the device. Additional space is maintained in the DAM on the master device. The pool is replenished when it gets low, and excess is returned to DAM. If a stack request is outstanding but all local space is used, the request is chained into the device overflow table (DOT) contained in CMR. Permanent file access between mainframes is coordinated through the PFC.

To comprehend the functions of the various tables described in figure 4-4, the terms used in 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 The smallest amount of data a user can access; it is 64_{10} CM words and is usually equal to 1 sector.
- RB The smallest amount of mass storage that can be allocated. An RB, defined in the RBR header, is several PRUs in length. The number of PRUs in a default RB is given in table 4-2.
- RBR A bit-coded table which indicates those RBs on a device which are assigned to file, flawed (defective), or available for assignment. A zero bit indicates that the specific RB is available for assignment.

TABLE 4-2. DEFAULT RB SIZE

Device	RMS Type	Device Type		Default PRU/RB (Decimal)
		Mnemonic	Code	
844-21	Disk pack	AY	13	57
844-41	Disk pack	AZ	14	57†
819	Disk pack (fixed)	AH	15	160
885	Disk pack (fixed)	AJ	17	160†

†Using the default PRU/RB size, the 885 and 844-41 devices require two RBRs to fully describe the available disk space.

RMS TABLES

The record block reservation table (RBR) and the record block table (RBT) contain information about each record block in an RMS device. Figure 4-4 shows the interface between the RMS tables and the file/device tables described earlier in this section.

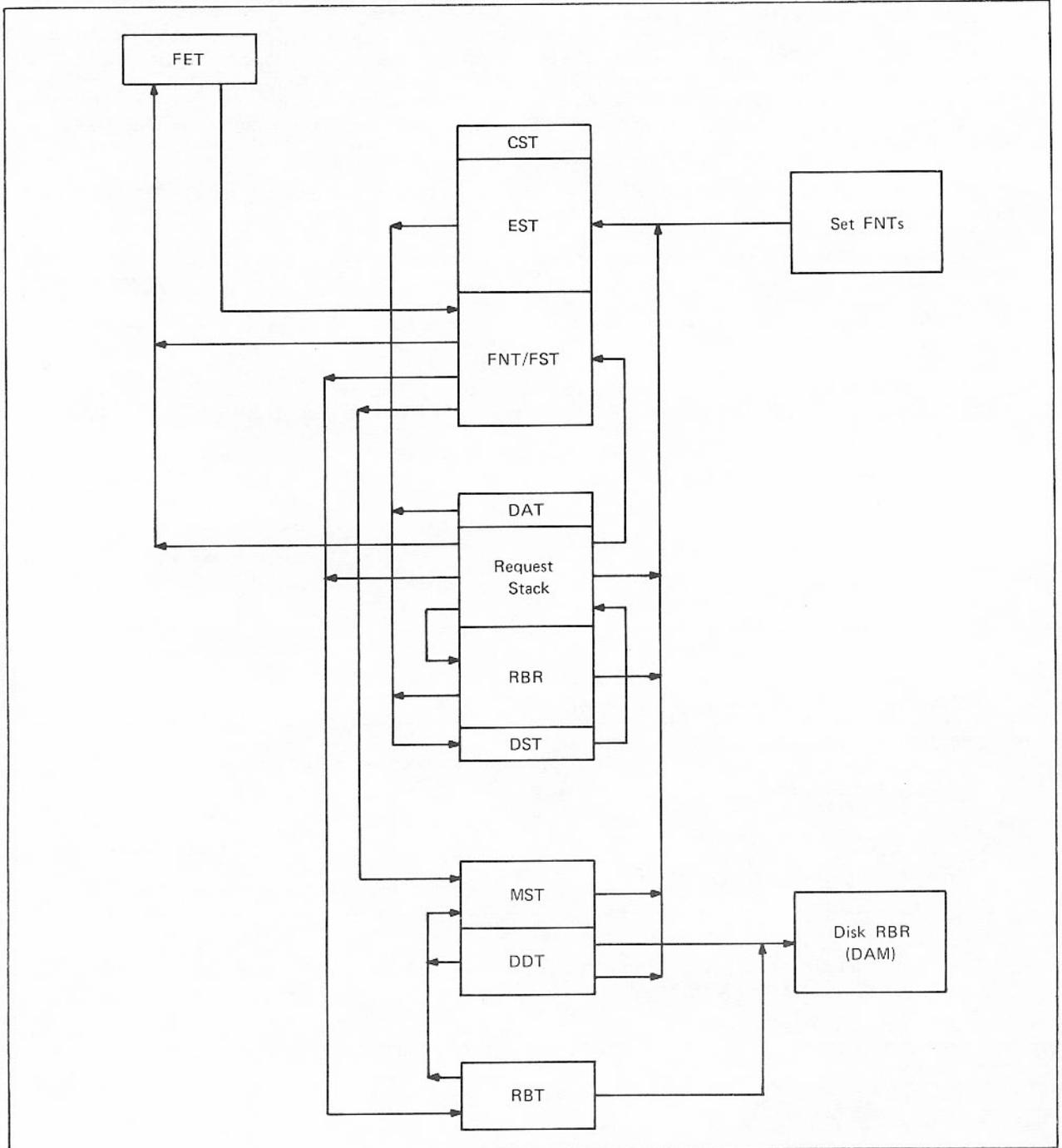


Figure 4-4. RMS Tables

Record Block Reservation Table (RBR)

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.

The first word of each RBR header contains a 6-bit allocation style code supplied as a parameter to the RBR macro when the CMR is assembled at an installation. Unique allocation style codes for each RBR can be set by the installation; this code can be used to direct a file to the RBR with a specific RB size and/or recording technique.

Record Block Table (RBT)

The RBT is file oriented. Each mass storage file in the system has an associated RBT chain. The RBT, located in the high address end of CM, consists of word pairs which are linked to form an RBT chain for each file that exists on an allocatable device currently recognized by the system (refer to figure 4-5). The RBT expands and contracts by 100 (octal) word blocks as files are allocated and released. A maximum of 8192 (decimal) CM 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 numbers of record blocks 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 and DAM bit tables 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 CM.

The CMR pointer word P.RBT contains the current size of central memory divided by 100 (octal), as well as the current length of the RBT in 100 (octal) 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 to form the empty chain. As record blocks are released from an evicted file, the dropped word pairs are linked into the empty chain. Word pairs are assigned to files from the head of the RBT empty chain, and the new first-member word pair ordinal is entered into the CMR pointer word. The RBT channel is requested as an interlock before a word pair is removed from the empty chain.

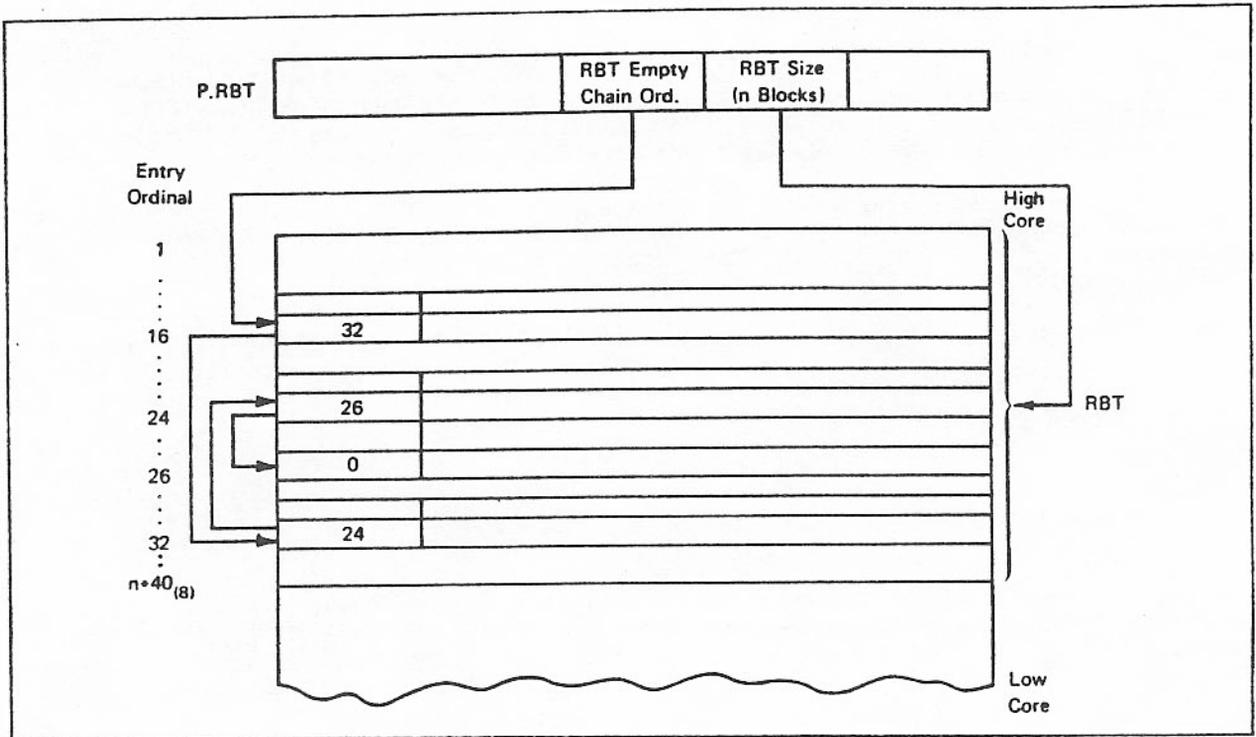


Figure 4-5. Record Block Table

Word 1 of the first word pair assigned to a file contains ordinals, flags, and so forth. The RB bytes 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 RB numbers which indicate the physical address on the device and a corresponding bit in the bit tables. As a file expands, additional RBs are entered into the RB byte fields until the word pair is filled; in this case, another word pair is assigned to the file and linked to the current word pair. If no more record blocks are assignable from the RBR/DAM table, an overflow condition occurs; in this case, a word pair in overflow format is linked into the chain, another word pair is linked to the overflow word pair, and processing continues with the remaining RB byte fields in the last link on the overflowed device set to zero.

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 into the RBT empty chain.

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.

When a file is established for a job, an entry is made in the FNT table. The FST part of the FNT entry for a mass storage file contains the ordinal of the first RBT word pair (WP) of the RBT chain that describes the file. The same FST word also contains the current file position.

Figure 4-6 shows the FNT pointing to the RBT chain for the file. The RBT chain is made up of word pairs that are forward linked (byte 0). That is why the first word pair must be known. In the example, the file is described by the contents of two word pairs. The first word pair is 27 (octal) and the second is 52 (octal) to represent 27 and 52 word pairs from the end of CM, respectively. The first word pair contains some additional information besides RBs, such as the EOI PRU. This is the last PRU+1 of the last RB of the last word pair in the RBT chain that describes the file. The first word pair also contains the MST ordinal of the set that the file resides on. Each word pair contains the DAM ordinal of that part of the set that contains the RBs in that word pair.

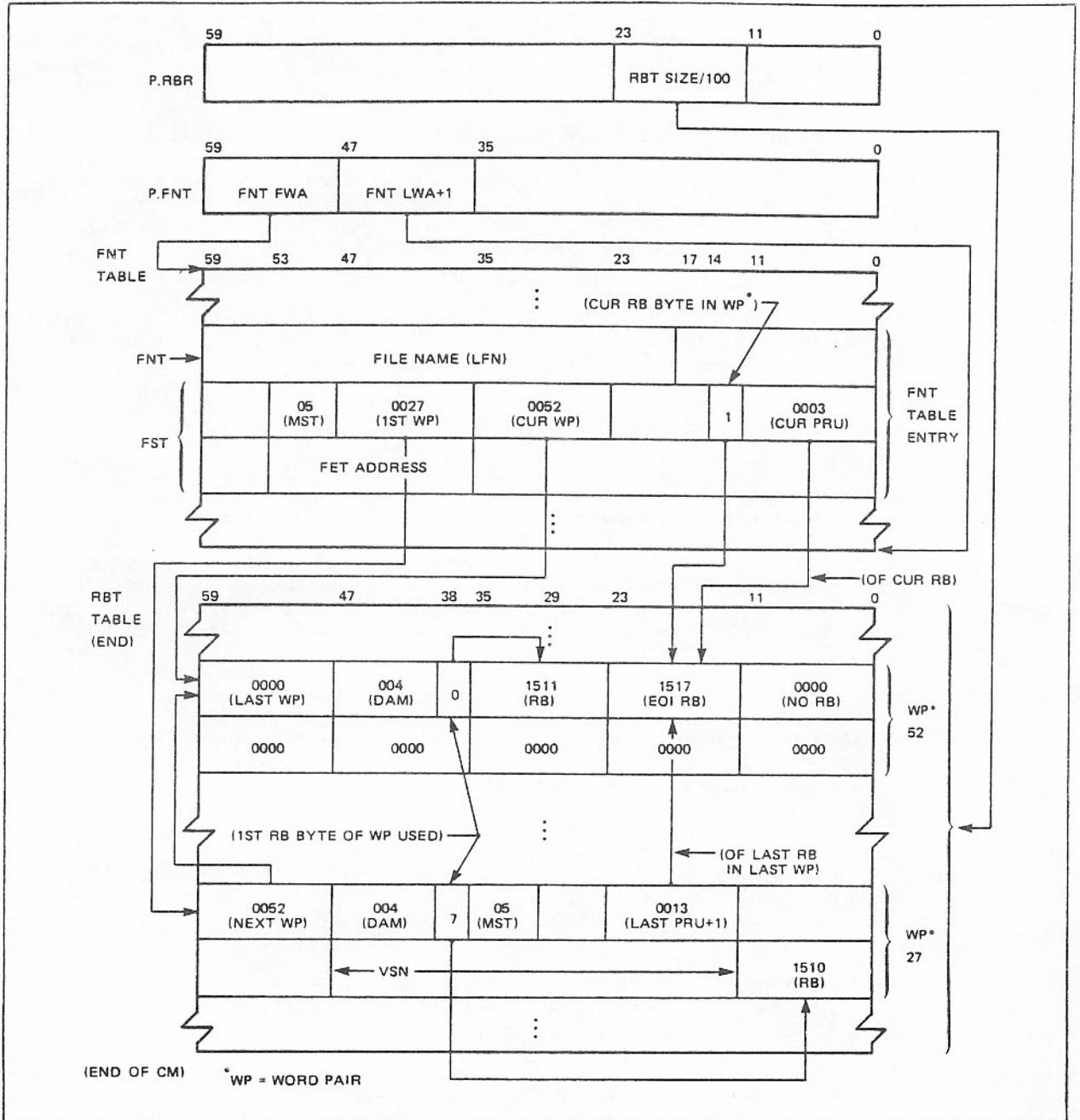


Figure 4-6. File Table Interfaces - FNT Points to RBT Chain

The FST in the example points to the first RBT word pair (0027) and the current position, which is specified as the current RBT word pair (0052), the current RB byte of that word pair (1), and the current PRU (0003) within the RB specified by that RB byte. The current position of the file in the example is seven PRUs from EOI.

Figure 4-7 shows how information in the RBT first word pair is used to find the correct RBR via the DDT. The DAM ordinal, rather than the RBR ordinal, is used to identify files because it is associated with the device (pack) in the set, while the RBR ordinal is associated with the drive (unit). When a pack is moved from one unit to another, its RBR ordinal changes but its DAM ordinal does not. The DAM ordinal is actually the relative PRU within the set DAM table on the master device. This table contains the DAMs for all members of the set. A DAM starts on a PRU boundary and may take one or more PRUs. Only those DAM (PRU) ordinals corresponding to the start of a DAM are valid as DAM ordinals.

The RBT first word pair in the example specifies DAM ordinal 004 and MST ordinal 05. A linear search is first made of the DDT table for an entry with the corresponding MST ordinal and a DAM ordinal range that includes the one specified in the RBT first word pair. (If no entry is found, the device is not mounted.) If only one DAM is associated with this DDT entry, the first and last DAM ordinals designated there will be the same number. Once the DDT entry is located, the DAM ordinal range (003 to 006 in the example) and the EST ordinal (10 in the example) found in that entry are saved.

A linear search of the RBR table headers is now made for the first entry with the same EST ordinal that was found in the DDT (10 in the example). When found, this RBR corresponds to the first DAM ordinal from the DDT (003 in the example). If this does not match the DAM ordinal from the RBT first word pair, the DAM ordinal corresponding to the next RBR is determined. This is done by taking the length of the bit map from the RBR header plus 3 for the DAM header (on disk), adding 77 (octal) and dividing by 100 (octal) to determine the number of PRUs this DAM takes, and adding that to the DAM ordinal this RBR represents to give the DAM ordinal for the next RBR. The DAM ordinal for this RBR is now compared to the DAM ordinal from the RBT first word pair and the search continues until the correct RBR is found. In the example, the search concludes at the second RBR.

The DAM ordinal range from the DDT was 003 to 006 and the DAM ordinal from the RBT first word pair was 004. The last DAM ordinal of the range 006 means either there are two more RBRs corresponding to DAM ordinals 005 and 006, or that the DAM with ordinal 004 requires two PRUs and only one RBR follows corresponding to DAM ordinal 006. As above, the correct situation is determined by examining the length field in the RBR header corresponding to DAM ordinal 004.

Figure 4-8 shows how the RB bytes in the RBT word pairs point to the RBs in the RBR bit table. Bits in the RBR bit table (pointed to from the RBR header) are allocated contiguously, 60 bits per word for the entire allocation space represented by this RBR. When an RB is allocated, the bit in the RBR bit table is set and its position is converted to an RB ordinal and placed in an RB byte in an RBT word pair. When an RB is deallocated, the RB ordinal from the RB byte is converted back to a bit position. That bit in the RBR bit table and the RB byte in the RBT word pair are both cleared.

To convert the current file position of the example to its equivalent RB bit position, check the FST current position from figure 4-6 to find word pair 0052 and RB byte 1. Figure 4-6 shows that RB (1517 in the example) pointing to the RBR bit table. The RB ordinal of 847 decimal (1517 octal) is first converted to an RB number by subtracting 1. Dividing 846 by 60 gives word 14 with a remainder of 6. Since RBR bit table bits are allocated left to right, this corresponds to bit 53 of word 14 of the RBR bit table. To find the corresponding physical address on the device, use the RB number and apply the appropriate formula given under Record Block Table Entry in appendix B.

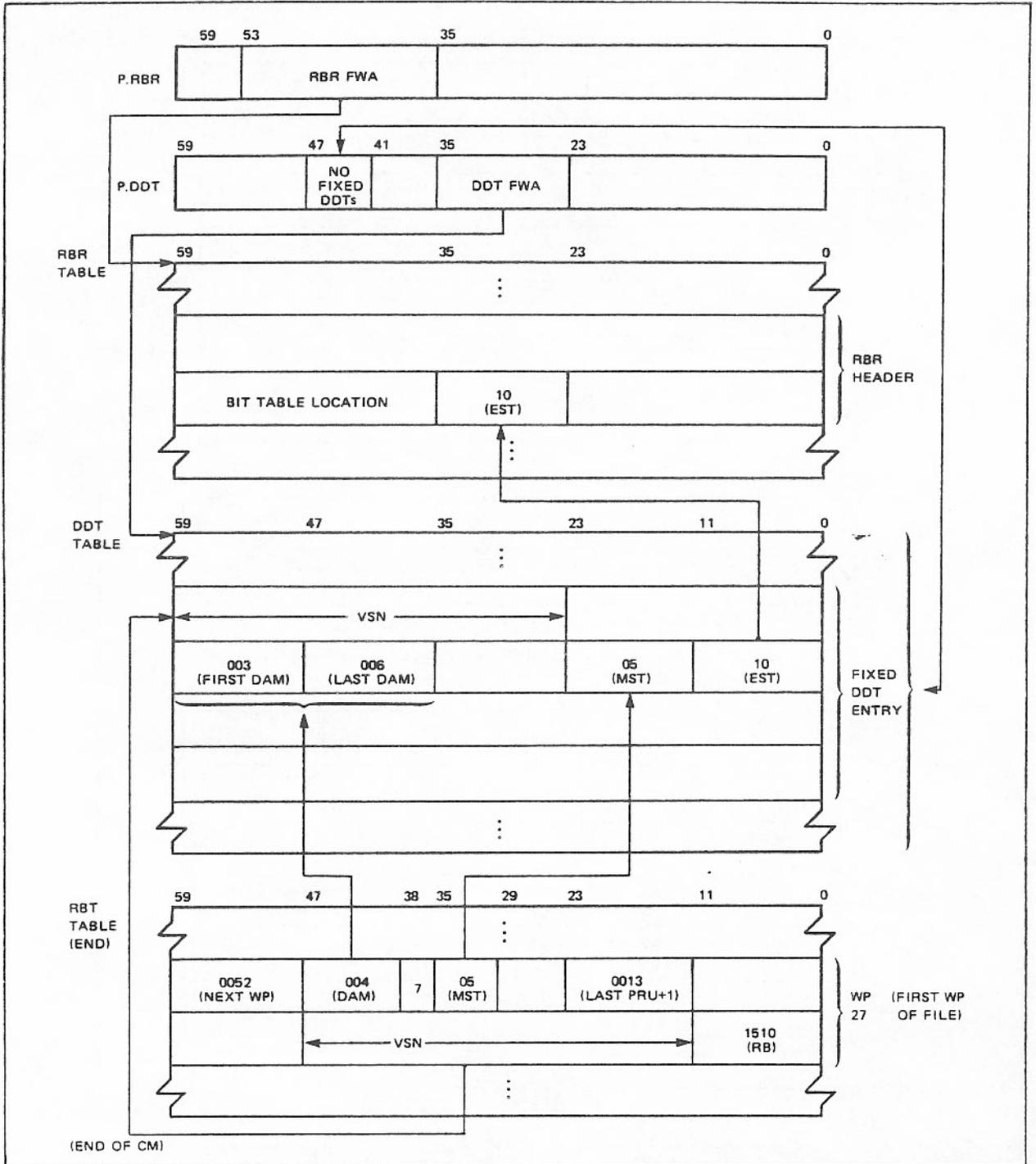


Figure 4-7. File Table Interfaces - RBT Points to RBR Via DDT

INTERCOM/RDF TABLES

INTERCOM and RDF use word 16g of the CMR pointer area as the pointer to the INTERCOM/RDF multiplexer table header and the INTERCOM/RDF pointer area. The multiplexer table header is central memory resident.

The INTERCOM/RDF pointer and buffer area is generated when INTERCOM or RDF is initialized and is not resident when neither INTERCOM nor RDF is running in the system. This area contains pointers to the various chains in the INTERCOM/RDF buffer area, and the multiplexer subtable entries.

The INTERCOM/RDF tables are detailed in appendix B.

Input and output request processing depends upon the source/destination of each request. There are four source/destination devices:

1. disk
2. Extended memory (EM)
3. tape
4. user terminal

The two sources for I/O requests are:

1. user programs
2. PP routines

A user program always requests I/O with the same protocol and with no change for I/O destination. This allows the user to change the source/destination device without changing his program.

All user program I/O requests are passed to CIO. CIO checks the FET for correctness, and establishes the proper linkages with the FNT before turning the I/O function over to the proper driver for the actual I/O operation.

If the I/O device is disk or EM, CIO changes it to a stack request and transfers it to the stack processor manager (SPM) for processing. SPM is a part of CPMTR that stores the requests for disk I/O in the request stack table (T.RSQ/T.RST) to await action by a disk driver (ISP/ISQ/HSP). SPM transfers stack requests for EM I/O to the EM driver, which is also part of CPMTR.

If the I/O device is tape, CIO calls a tape driver to transfer data to or from the user's buffer.

If the I/O device is the user terminal, CIO loads overlay 3TT to move the data from/to the user's field length to/from a buffer in INTERCOM'S field length. An INTERCOM driver then moves the data from/to the INTERCOM buffer to/from the user's terminal.

PP routines that desire to perform I/O usually make stack requests, or call a tape driver directly.

CIO CODES

The code and status field is the rightmost 18 bits of the first word in the FET. When CIO is called, this field contains a code that designates the type of operation CIO is to perform (for example, read, write, skip forward).

Bits 0 and 1 of this field have the following significance.

Bit 0

- 0 Indicates the I/O operation is not complete. This bit must be zero when CIO is called.
- 1 Indicates the I/O operation is complete. Set by appropriate driver.

Bit 1

- 0 Indicates I/O operation is to be performed in coded mode instead of binary code. Coded mode has significance only for the data transfer macros READIN and WRITOUT, and for tape I/O where coded means display code characters are to be converted to some external code such as BCD, ASCII, or EBCDIC, or vice versa. Otherwise, coded mode is ignored.
- 1 Indicates I/O operation is to be performed in binary mode.

All codes indicated by * are illegal; all reserved codes are illegal. All codes are octal for coded mode operations; 2 is added for binary mode. For 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	RETURN	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	*		

The 200 series is for special read or write (reverse, skip, nonstop, rewrite, and so on).

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				

The 300 series is 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	CLOSER,RETURN

The 400 series is reserved for Control Data.

The 500 series is reserved for installations.

The 600 series is as follows:

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

The 700 series is reserved for Control Data.

CIRCULAR BUFFER

A circular buffer is a temporary storage area in CM through which data passes during I/O operations (figures 5-1 and 5-2). 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.

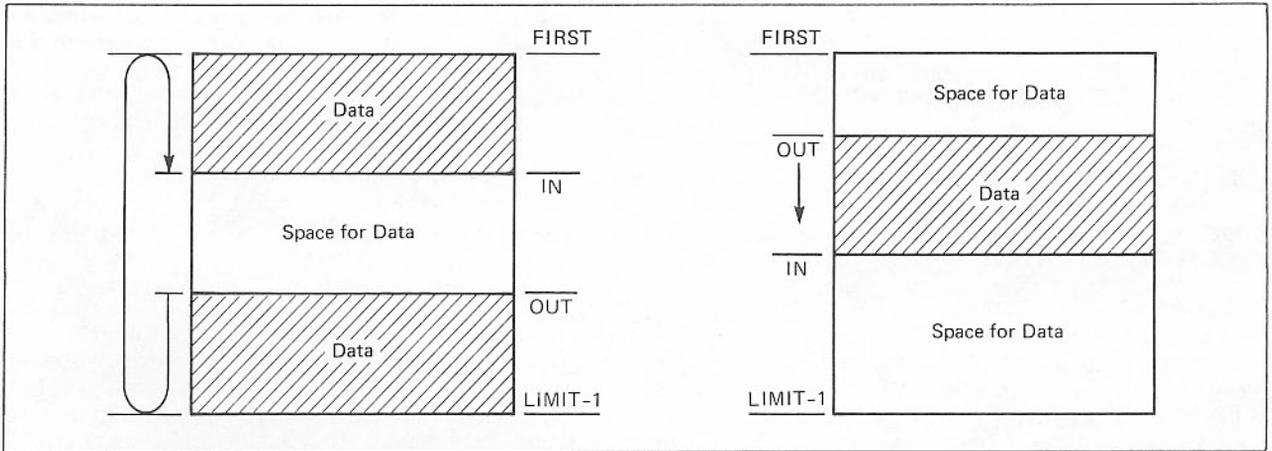


Figure 5-1. Circular Buffer

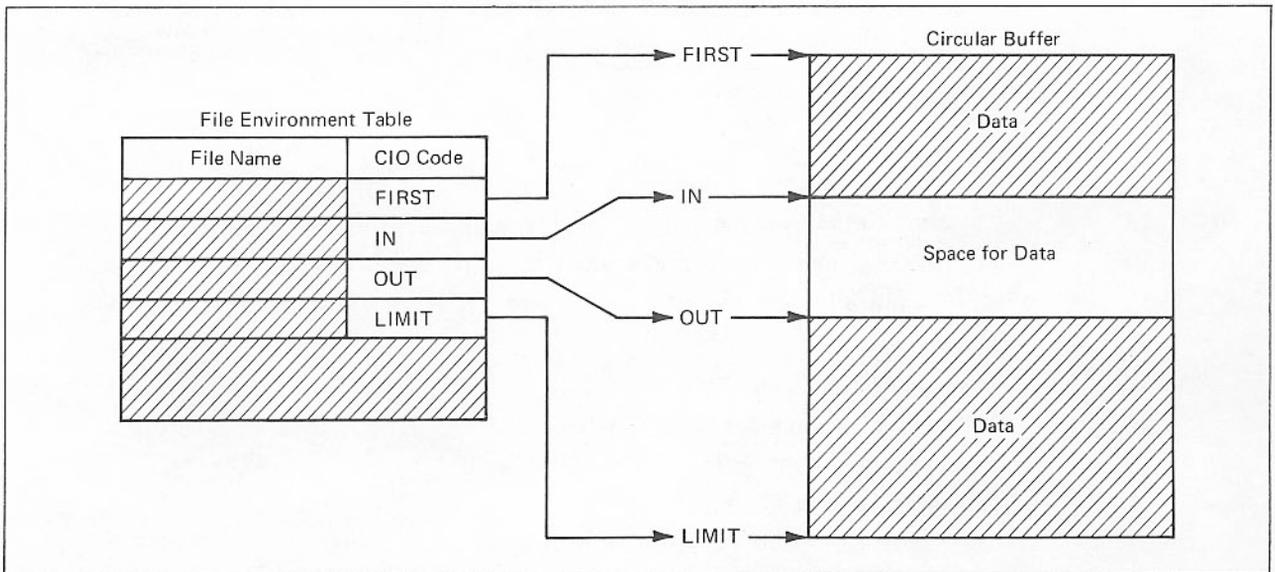


Figure 5-2. FET - Circular Buffer Interface

CIO OPERATION

When CPMTR detects a request for CIO, the request is processed according to the availability of EM. If the system does not have EM, control is turned over to the PP routine CIO. If the I/O request is for disk or EM, control is turned over to the CPU program CPCIO, which is a CPU version of the PP routine CIO.

CPCIO

CPCIO acts the same as CIO except for the following differences.

CPCIO checks many fields of the FET for validity, including buffer pointers, CIO code, and the file name. If the FNT pointer field contains a pointer to an FNT entry, CPCIO checks that the file names match, and that the FNT entry has the proper control point number. If the FNT entry is not proper, CPCIO searches the entire FNT for the proper entry. If no proper entry is found, CPCIO creates one with only default set attributes. Once the proper FNT entry is found or created, CPCIO puts the address of that FNT entry into the FET to bypass the FNT search if another request is made on the file using the same FET.

Next, CPCIO checks the complete bit in the FNT entry. If the bit is zero, another request has not completed on the file, and the new request is delayed. CPCIO puts the control point into periodic recall causing the RA+1 request to be retried after the recall period expires. CIO loops while waiting for the complete bit to be set.

Once the FNT is determined not to be busy (the complete bit is set), the previous operation from the code/status field of the FNT entry is checked to ensure the current operation can legally follow. If the sequence is illegal (such as a sequential read after write), or if any of the other checks described above fail, the request is aborted. CPCIO replaces the RA+1 CIO request with a CEM request, and returns it to CPMTR for processing. When CEM is loaded, it calls 6WM to post the dayfile messages to document the error. After the dayfile messages have been sent, the EP bit is checked. If it is set and an OWNCODE routine address is given in the FET, control is given to the OWNCODE routine. Otherwise, the job is aborted and CIO calls 6WM directly.

Once all the checks have been made, CPCIO determines if the file has been assigned to a device. If it hasn't, CPCIO calls CIO to handle the request. Otherwise, CPCIO enters CP4ES that generates a stack request stack for the I/O transfer. The stack request is put in the proper request stack, and CPCIO exits. CIO calls in the appropriate overlay listed below.

1OP	File open (nontape files).
3DO	Mass storage device file open.
3IC	File close for 66x/67x tapes.
3IF	Multifile positioning for 66x/67x tapes.
3II	66x/67x initialization and setup.
3IJ	System calls to 1IT.
3IL	Slave for 3IO, 3IC, and 3IV.
3IM	Write error message for 66x/67x tapes.
3IN	VSN message processor for 66x/67x tapes.
3IO	Tape open for 66x/67x tapes.
3IV	Reel close; EOR processor for 66x/67x tapes.
3TT	Connected I/O read/write.
4ES	Enter stack request (mass storage I/O).

Tape drivers:

- 1IT Main 66x/67x tape driver, calls the nix overlays.
- 1LC Load conversion tables into 66x/67x controller.
- 1TS Tape sampler.
- 2IA L tape read for 66x/67x tapes.
- 2IB L tape write for 66x/67x tapes.
- 2IC Coded read (seven-track) for 667/677 tapes.
- 2ID Coded write (seven-track) for 667/677 tapes.
- 2IL Label read/write for 66x/67x tapes.
- 2IP 66x/67x tape positioning.
- 2IR 66x read driver.
- 2IT 67x read driver.
- 2IW 66x write driver.
- 2IX 67x write driver.

Tape error recovery drivers:

- 3IE Error diagnosis for 66x/67x tapes.
- 3IR Read recovery for 66x/67x tapes.
- 3IW Write recovery for 66x/67x tapes.

If the file device code is for a nonallocatable device, CIO 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 standard format seven-track tape from a 667 tape unit, CIO calls the overlays 1IT, 2IR, and 3II into its PP. 2IR reserves one of the hardware channels connected to the equipment. It then issues the function codes to connect the controller and tape drive. 2IR issues functions to transmit one PRU of data from the tape drive over the data channel.

2IR accumulates the PRU of data in a PP buffer. When the entire PRU is transmitted or an end-of-record (short PRU) is encountered, 2IR picks up the pointers to the circular buffer in CM from the FET. 2IR 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. 2IR and 1IT update the PRU count in the file FNT, release the channel, set completion bits in the FNT and FET, and drop out.

Tables 5-1 through 5-12 list the logical sequence of events during various CIO tape operations.

TABLE 5-1. READ MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x					1. Exit if not enough room in buffer for one maximum size physical record.
		x	x	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	4. Read one physical record into CM.
x	x					5. If physical record exceeds maximum allowable size, return error status DEVICE CAPACITY EXCEEDED and perform error procedures.
		x	x	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	x	x	7. If end-of-file mark is read, perform end-of-file mark procedures.
x	x	x	x	x	x	8. If noise records are encountered, go to 3.
x	x	x	x	x	x	9. If parity error, perform parity procedures.
		x	x	x	x	10. If end-of-tape reflective spot is encountered and tape is unlabeled, perform end-of-reel procedures.
x	x					11. If short PRU is read, strip level number.
x	x					12. If zero length PRU is read, go to 21.

TABLE 5-1. READ MACRO LOGICAL SEQUENCE (Contd)

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
	x		x			13. When 6681 is present, convert data in PP from BCD to display code.
					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	x	x	x			16. Transmit data to CM.
x	x	x	x	x	x	17. Update IN.
x	x					18. Fetch OUT from CM.
		x	x	x	x	19. Place in word 7 of FET the number of unused bits in the last data word.
x	x					20. If full PRU, go to 1.
x	x	x	x	x	x	21. If last record was level 17 or 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.

TABLE 5-2. READN MACRO LOGICAL SEQUENCE

S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x	x	x	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 greater than record length plus one (header word) to avoid OUT equal to IN when buffer is full.
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 size, 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	x	x	7. If end-of-file (tape mark) is read, perform end-of-file mark procedures. Go to 18.
x	x	x	x	8. If noise records are encountered, go to 3.
x	x	x	x	9. If parity error is encountered, perform parity procedures.
x	x	x	x	10. If end-of-tape reflective spot is encountered on unlabeled tape, perform end-of-reel procedures.
	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	x			13. Transmit data to CM.
x	x	x	x	14. Update IN in PP memory.
x	x	x	x	15. Place length of record and number of unused bits in last data word in buffer header word.
x	x	x	x	16. Update IN.
x	x	x	x	17. Fetch OUT.
x	x	x	x	18. If last record is tape mark, set end-of-file status and exit.
x	x	x	x	19. Go to 2.

TABLE 5-3. READSKP MACRO LOGICAL SEQUENCE

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

TABLE 5-3. READSKP MACRO LOGICAL SEQUENCE (Contd)

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

TABLE 5-4. RPHR MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	Sequence of Events
x	x	1. Set OUT to IN.
x	x	2. Read one physical record.
x	x	3. If end-of-file mark is read, perform end-of-file procedures.
x	x	4. If noise records are encountered, go to 2.
x	x	5. If parity error is encountered, perform parity procedures.
x	x	6. If zero length PRU is read, go to 10.
x	x	7. Transmit data to CM.
x	x	8. Update IN.
x	x	9. If last record is level 17 or tape mark, set end-of-file status.
x	x	10. Exit.

TABLE 5-5. WRITE MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x					1. Exit if not full PRU.
		x	x	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					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 is present, convert from display code to BCD in PP memory.
					x	7. When 6681 is present, convert from display code to BCD in CM.
	x					8. Convert zero byte line terminator to 1632.
x	x	x	x			9. Write record to tape.
				x	x	10. Write, from CM to tape, data contained between OUT and IN, adjusted by unused bit count.
					x	11. When 6681 is present, convert data in CM buffer back to display code.
x	x	x	x	x	x	12. If parity error is encountered, perform parity procedures.
x	x	x	x	x	x	13. If end-of-tape reflective spot is encountered, perform end-of-reel procedures.
x	x	x	x	x	x	14. Update OUT.
		x	x	x	x	15. Exit.
x	x					16. Fetch IN from CM.
x	x					17. Go to 1.

TABLE 5-6. WRITER MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x	x	x	x	x	1. If IN equals OUT, exit.
		x	x	x	x	2. If PRU is not full, insert level number in PP buffer.
		x	x	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					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 from display code to BCD in PP memory.
					x	8. When 6681 is present, convert from display code to BCD in CM.
	x					9. Convert zero byte line terminator to 1632.
x	x					10. If IN equals OUT, write zero length record. Go to 12.
x	x	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.

TABLE 5-6. WRITER MACRO LOGICAL SEQUENCE (Contd)

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
					x	13. When 6681 is present, convert data in CM buffer to display code.
x	x	x	x	x	x	14. If parity error is encountered, perform parity procedure.
x	x	x	x	x	x	15. If end-of-tape reflective spot is encountered, perform end-of-reel procedures.
x	x	x	x	x	x	16. Update OUT.
		x	x	x	x	17. Exit.
x	x					18. If full PRU is not written, exit.
x	x					19. Go to 1.

TABLE 5-7. WRITEF MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x					1. If no data from OUT to IN, go to 20.
		x	x	x	x	2. If no data from OUT to IN, go to 17.
x	x					3. If not full PRU, insert level number 0.
		x	x	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.

TABLE 5-7. WRITEF MACRO LOGICAL SEQUENCE (Contd)

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x					6. Fetch one PRU of data starting at OUT or fetch 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 from display code to BCD in PP memory.
					x	9. When 6681 is present, convert from display code to BCD in CM.
	x					10. Convert zero byte line terminator to 1632.
x	x	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.
					x	13. When 6681 is present, convert data in CM buffer to display code.
x	x	x	x	x	x	14. If parity error is encountered, perform parity procedures.
x	x	x	x	x	x	15. If end-of-tape reflective spot is encountered, perform end-of-reel procedures.
x	x	x	x	x	x	16. Update OUT.
		x	x	x	x	17. Write end-of-file mark and exit.
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 is WRITE, write zero length PRU.
x	x					21. Go to 17.

TABLE 5-8. WRITEN MACRO LOGICAL SEQUENCE

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

TABLE 5-9. WPHR MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	Sequence of Events
x	x	1. If IN equals OUT, exit.
x	x	2. Fetch data from OUT to IN.
x	x	3. Write record to tape.
x	x	4. If parity error is encountered, perform parity procedures.
x	x	5. If end-of-tape reflective spot is encountered, perform end-of-reel procedures.
x	x	6. Update OUT and exit.

TABLE 5-10. SKIPF MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x	x	x	x	x	1. If n is 0, set n to 1.
		x	x	x	x	2. If L is less than 17, interpret as L equals 0.
x	x	x	x	x	x	3. Read a physical record.
x	x	x	x	x	x	4. If noise record is encountered, go to 3.
		x	x	x	x	5. If end-of-tape reflective spot is encountered on unlabeled tape, perform end-of-reel procedures.
x	x					6. If record is full PRU, go to 3.
		x	x	x	x	7. If end-of-file mark is 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	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 is not equal to 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.

TABLE 5-11. SKIPB MACRO AND BKSP MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x	x	x	x	x	1. If n is 0, set n to 1.
		x	x	x	x	2. If L is less than 17, interpret as L equals 0.
x	x	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 is encountered, go to 4.
x	x					6. If record is 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	x	x	9. If end-of-file mark is encountered, assume level number equals 17. Otherwise, assume level number equals 0.
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 is not equal to zero, go to 3.
x	x	x	x			12. Exit.

TABLE 5-12. BKSPRU MACRO LOGICAL SEQUENCE

Standard Binary	Standard Coded	S Binary	S Coded	L Binary	L Coded	Sequence of Events
x	x	x	x	x	x	1. If at load point or if PRU count equals 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 is not equal to 0, go to 1.
x	x	x	x	x	x	4. Exit.

Allocatable Device I/O

Most files in the system are stored on allocatable devices. The system library ZZZZZ04 is stored on an allocatable device known as the system device. Each time a PP overlay or a CP program not resident on CM is to be loaded from the library, I/O must be performed on the system device. The job and system dayfiles and the CE error files are stored on allocatable devices, as are all input and output queue files and all files created by CIO.

A request for I/O on a mass storage allocatable device must be placed in a table, called the request stack. The stack is searched, and the request which requires 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 devices 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 devices 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. It applies to all mass storage devices. Since READC uses intersector time to the maximum while reading high-speed mass storage devices, it does not include checks for erroneous programming and control words. READC should be used by system programmers only. The format is:

READC fet,recall

READC transmits PRUs continuously to the circular buffer, with a control word preceding each PRU. Reading continues until one of the following occurs.

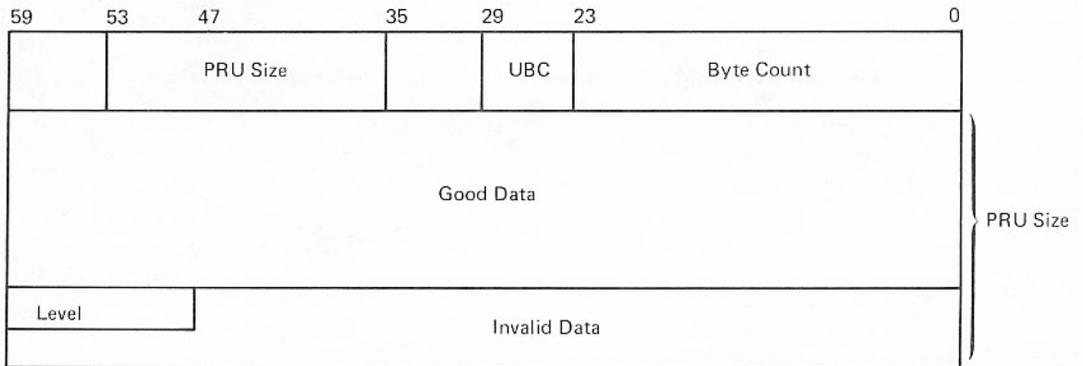
- The buffer does not have enough room for the next PRU and its control word.
- An error condition occurs.
- End-of-information is encountered.

Code and status on completion (x depends on file mode):

00020x	Normal completion.
0ee20x	Error code ee.
74123x	EOI.

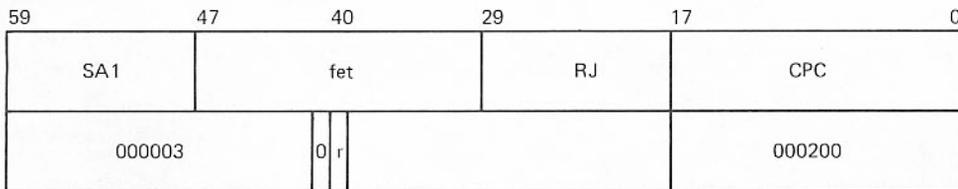
On mass storage, the same amount of data is transmitted for every PRU: the control word and one device standard PRU. The last 12 bits of the control word and the entire standard PRU length are exactly the physical data recorded on the device, including system control information.

The following diagram shows the format of the PRU.



- PRU size** 64 CM 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 CM words occupied by the data. The value is recorded on disk as 12 bits, but expanded here to 24 bits.
- Level** System logical record level number. If byte count divided by 5 is a full PRU, level does not exist.

The READC macro generates the following code.



READLS

The READLS function applies only to mass storage files. READLS reads several random records into the file circular buffer according to the list of relative PRU numbers provided by the user. No information in the buffer reveals boundaries. READLS should be used by system programmers only.

The format is

READLS fet,recall

Before READLS is called, bits 17 through 0 of FET+6 should be set to the address of the list of addresses to be read. Reading continues until one of the following occurs.

- The 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 discontinued for device repositioning.

Code and status on completion:

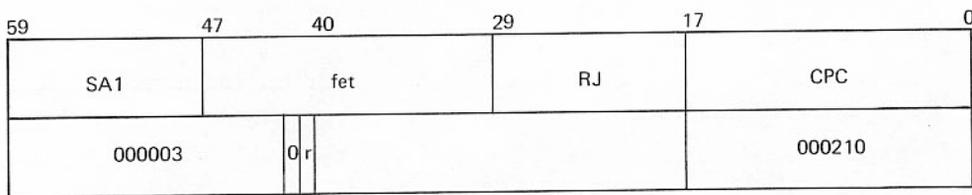
Bits 3 and 4 contain 01, 10, or 11, giving the status at the point where the operation terminated (10 is end-of-record, 11 is end-of-file). The operation terminates with EOI status if the last PRU of the file is read and no EOR or EOF occurs. The contents of bits 8 through 5 do not pertain to this description.

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

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

- Bits 59 through 36 contain a PRU number, the same as used in the system indexes. These are the numbers the system returns to the record request/return information fields (bits 29 through 0 of word 7) of the FET when records are written on a mass storage device. Bits 35 through 0 are zero. A user list in this format is converted by the system to the next format.
- Bits 35 through 0 contain the internal direct access address (RBTA/RBB/PRU) address RBT.

The READLS macro generates the following code.



WRITEC (Continuous Write)

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

WRITEC fet,recall

WRITEC transmits PRUs from the circular buffer to a mass storage device. Each PRU in the buffer must be preceded by a control word. Writing continues until one of the following occurs.

- The buffer is empty.
- An error occurs.

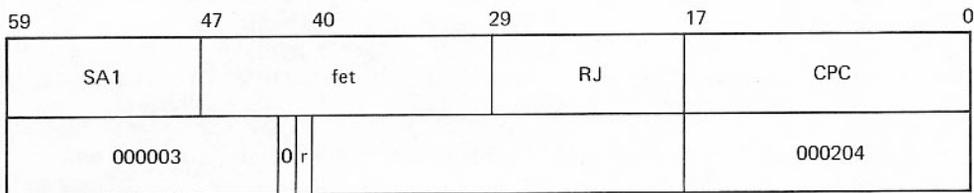
The diagram of the PRU and control word appears with the discussion of READC. PRU size must be standard 64 CM words for mass storage; if not, serious errors result. The 24 low-order bits of the control word and the full CM words are written to the device.

Byte count is the count in 12-bit bytes of good data in the PRU and must be a multiple of 5. If byte count/5 is less than the device PRU size, the next 12-bit byte after the good data is the system level number in binary. Level must be in the range 0 level 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 system logical record format, data resolution is to the nearest full CM word so that the UBC field is always zero. This field is reserved for future expansion.

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

The WRITEC macro generates the following code.



Stack Processor

The stack processor consists of the CPU monitor mode routine SPM and the following PP routines:

- 1S5 Examines DST ordinal and loads 3DO if the DST ordinal is 1; otherwise, loads 4DO.
- 1SP/1SQ The stack processor driver supervisor.
- 1RN Requests/releases RBT storage; merges released chains to the empty chain.
- 3DO Assigns a device and an RBT word pair to a new or overflowing file.
- 4DO Processes stack requests which require no device access, specifically O.SKPF/O.SKPB with skip counts of 777777g and O.BPRU.

If SPM finds a request for an 819 disk, it calls HSP, the 819 disk stack processor (refer to 819 Disk I/O Processing). The remainder of this discussion applies to only 844 and 885 disk drives.

SPM is called to enter, terminate, and reissue stack requests. SPM performs request scheduling, device optimization, and all RMS I/O functions except file assignment (performed by 3DO), non-I/O skipping (performed by 4DO), and physical I/O (performed by 1SP/1SQ). 1SP and 1SQ are RMS device drivers. 1SP is used with 7054 and 7154 controllers; 1SQ is used with 7155 controllers. Each program handles one request at a time.

A PP or CP system routine initiates I/O by placing a stack request (first two words of the format) in the first two words of its associated communication area (T.PPCn for PPn) and calling the CP monitor function SPM (M.ICE/EX.SPM). SPM picks up the stack request from the communication area, generates the third word, puts the three-word stack request into the request stack area, and links it to the proper DST chain. If the priority bit in the stack request is set, that stack request becomes the first stack request in the proper DST chain. The priority bit should be used with discretion; otherwise, priority stack requests could be pushed down in the DST chain.

To perform RMS I/O, SPM selects a stack request and assigns it to an RMS device driver. The driver must be assigned to a DST ordinal that represents an access to the specified RMS device. If a stack request is received and no device driver is operational, the request is entered by SPM and an M.ISP request (initiate stack processor driver) is sent to MTR. If the MTR communication path is busy, no request is posted and the device driver is assigned later during the normal MTR DST scan for work outstanding with no PP assigned. The PP is assigned by MTR by using the second word of the DST as the PPIR entry to call the proper PP routine.

When a device driver comes up, it initializes itself for the proper device (844 or 885). Channel and equipment numbers are in the PPIR. Initialization is completed and SPM is called for a stack request. (A special procedure is used during EDITLIB operations.) The device driver performs the I/O requested, obtaining field access as necessary, and at I/O completion, returns the stack request to SPM for termination processing. If there is another stack request outstanding for this driver, SPM assigns the request to the driver. Otherwise, it assigns the driver to idle. A subsequent stack request for this device need only be placed in this device driver's PP communication area to start I/O.

Device dependent code for 844 devices is in overlay 3SY which is contained in common deck RMSY. Device dependent code for 885 devices is in overlay 3SJ which is contained in common deck RMSJ. RMSY and RMSJ are in PL1A.

In addition to the overlay area for device dependent code, the stack processor has an overlay area for the executive routine that performs the other side of the data transfer or a nondata transfer related function. There are currently four such executive routines.

- CM input/output.
- PP input/output.
- Positioning.
- Extended memory I/O via CM buffers or DDP.

As each stack request is processed, the appropriate executive routine is loaded if necessary. No load occurs if the proper routine is already present.

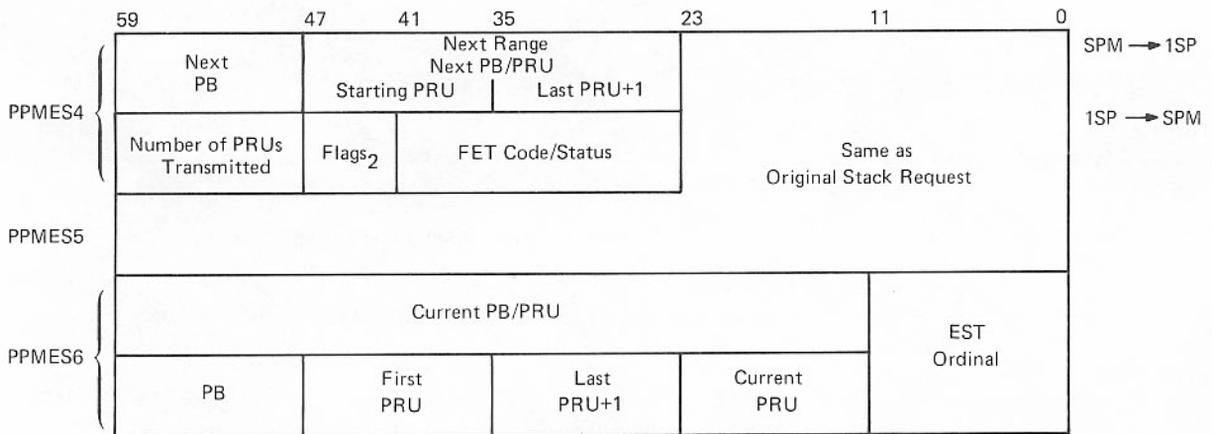
MTR ensures that at least one PP contains a stack processor at all times, or that one is reserved for that purpose. This requirement is necessary so that it is always possible to load a PP overlay that resides on mass storage. To avoid unnecessary loading and dropping of stack processors, a stack processor remains in the PP until MTR needs a free PP. At that point, MTR sets a PP request flag for SPM. Idle stack processors check this flag in their idle loop and call SPM if it is set. SPM then issues a drop order to the first idle stack processor it finds. If all stack processor PPs are busy, no action is taken until one is about to become idle. When the monitor drop flag is set, SPM issues a drop order to the PP instead of an idle order.

Stack Request Formats

There are two kinds of stack request formats, external and internal. The external formats are used by PPs to submit disk I/O requests to SPM through the communication area of the caller. The internal format is used for SPM/stack processor communication through the stack processor communication area. It is different from the external format because the stack processor does not work with record blocks (RBs) but with physical blocks (PBs), currently 114PRUs/PB for 844 disks and 320 PRUs/PB for 885 disks. SPM converts RBs to PBs for ISP and RBs to physical addresses for ISQ. This is done so that stack processors do not have to access the RBT chains.

The external stack request format consists of three words. The first two words are supplied by the calling PP; the third word is added by SPM. The first word specifies the type of stack request (order code, interlock flag, direct) and source of RMS file or device information (FST pointer, RBT chain pointer, equipment pointer). The second word specifies the other side of the request (PP, CM, extended memory). The third word is SPM internal information. External stack request formats are summarized in the Request Stack Entry illustration in appendix B.

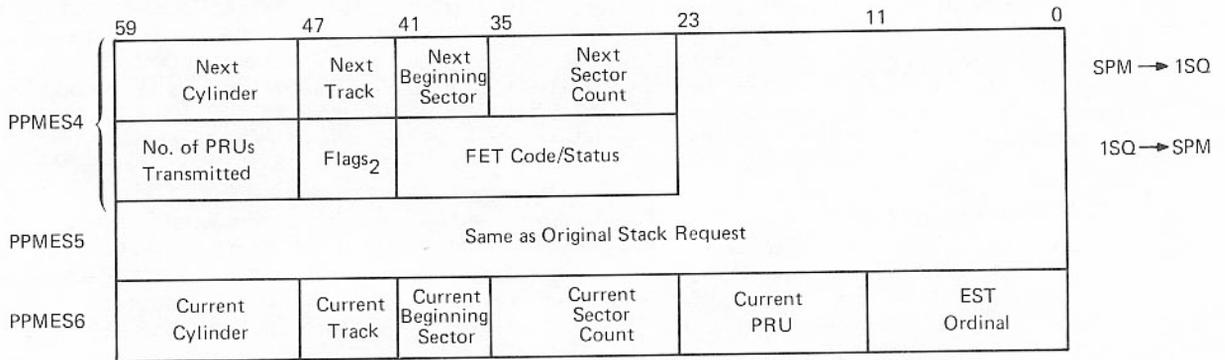
The internal stack request format consists of three words. It is part of the SPM/stack processor interface at communication area locations PPMES4 through PPMES6. The format varies depending on whether ISP or ISQ is used. The following diagram shows the format of the internal stack request for the SPM-ISP interface.



The first and third words contain the next PB/PRU and current PB/PRU transfer information. The next PB/PRU is the next one to be processed by ISP and, in the case of skipping backward, is the PB/PRU which logically precedes the current PB/PRU being processed. The NXPB function of SPM updates the next PB/PRU in an overlap synchronous manner so that ISP can continue to do I/O and have the next PB/PRU information available when needed. ISP uses the next PB/PRU information to update the current PB/PRU in word 3 and thus determine the current operation. The process is then repeated.

The second word and rightmost two bytes of the first word are the same as the original stack request. The first PRU and last PRU fields specify the range of the request within the specified PB; the current PRU is the one currently being processed. The flags₂ field (bits 47 through 42) is described under SPM-ISQ Interface in this section.

The following diagram shows the format of the internal stack request for the SPM-1SQ interface.



The SPM-1SQ interface is the same as the SPM-1SP interface except physical addresses are passed instead of PBs. The flags₂ field (bits 47 through 42 of the first word) is described under SPM-1SQ Interface in this section.

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, standard system symbols, and functions are as follows.

<u>CM Read/Write Order</u>	<u>Description</u>
00 O.READ	Corresponds to READ macro, CIO codes 010 and 012. Read data from device to CM until end-of-information is reached, a short PRU is read, or next PRU does not fit into the buffer. (Refer to table 5-1.)
01 O.RDSK	Corresponds to READSKP macro, CIO codes 020 and 022. Read as for O.READ until end-of-information is reached, a short PRU is read, or the CM buffer is completely full. Change to O.SKF with n=1 unless reading was stopped by a short PRU with record level greater than or equal to request level. (Refer to table 5-3.)
02 O.RCMPR	No corresponding CPC macro or CIO code. Read as for O.READ, but do not transmit the first n CM words of the PRU. n is the number of CM words in the PRFX (77) table. Used for loading programs from a system library in which the first n words represent the PRFX (77) table in each record and contain information of interest only to EDITLIB and deadstart.
03 O.RDNS	Corresponds to the READNS macro, CIO codes 250 and 252. Read data from device into CM buffer until end-of-information is reached, a short PRU with record level 16 or 17 has been read, or next PRU does 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.

CM Read/Write OrderDescription

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. (Refer to table 5-5.)
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, this corresponds to the WRITEF system macro, CIO codes 034 and 036. Same action as for WRITER macro, but short PRU is followed by zero-length level 17 record (logical end of file mark). (Refer to table 5-6.)
06 O.RMR	Corresponds to READLS macro, CIO codes 210 and 212. 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.
20 O.RCTNU, O.RCTU	Corresponds to READC macro, CIO codes 200 and 202. Provides nonstop reading from device to CM without releasing/reloading PP between logical records. Buffer must provide space for at least two records and their header words.
24 O.WCTNU, O.WCTU	Corresponds to WRITEC macro, CIO codes 204 and 206. Provides nonstop writing from CM to device without releasing/reloading PP between logical records. User's buffer must contain at least two records. Writing stops when buffer is empty.

PP Read/Write OrderDescription

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 OrderDescription

12 O.SKPF	Corresponds to SKIPF macro, CIO codes 240 and 242. Skip forward until n short PRUs, with level greater than or equal to the level specified in the request, have been read or until end-of-information is reached. With n=777777, the file is positioned at end-of-information. (Refer to table 5-10.)
-----------	--

Positioning Order

Description

- 13 O.SKB Corresponds to SKIPB macro, CIO codes 640 and 642. Skip backward one or more PRUs until n short PRUs, with level greater than or equal to 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). (Refer to table 5-11.)
- 16 O.BPRU Corresponds to the BKSPRU macro, CIO codes 044 and 046. Skip backward n PRUs. This repositioning is by PRUs rather than logical records. (Refer to table 5-12.)
- 17 O.RCHN Release allocatable storage and RBTs (processed by SPM). For permanent file set RBT chains, the first word pair to be evicted must be a first word pair or an overflow word pair.

SPM-Stack Processor
Communication Order †

Description

- 35 O.IDLE SPM has determined that there is no work for this stack processor and that MTR is not requesting a PP. The stack processor checks the order code in its idle loop, waiting for SPM to assign a stack request. When another order code appears, the stack processor processes it.
- 36 O.DROP SPM has determined that this stack processor is either idle or going to become idle, and that MTR is requesting a PP. The stack processor drops out.
- 37 O.SEEK SPM has stack requests for this stack processor but none are on-cylinder. The stack processor issues overlap seeks (up to five), monitors the units, and reserves the first one to come on-cylinder. Overlap seeks can also be issued with other stack requests. In this case, no monitoring is done. The seeks are issued before stack request execution. Status is taken after stack request execution and returned to SPM with the stack request.

Table 5-13 is a summary of stack processor orders.

TABLE 5-13. STACK PROCESSOR ORDERS

Octal Code	System Symbol	Order Function
00	O.READ	Read into CM.
01	O.RDSK	Read-skip into CM.
02	O.RCMPR	Read into CM, drop first three CM words.

† Internal format only.

TABLE 5-13. STACK PROCESSOR ORDERS (Contd)

Octal Code	System Symbol	Order Function
03	O.RDNS	Read nonstop.
04	O.WRT	Write from CM.
05	O.WRTR	Write EOF/EOR from CM.
06	O.RMR	Read multiple records to CM.
10	O.RDP	Read into PP memory.
11	O.RDPNP	Read into PP, drop first three CM words.
12	O.SKf	Skip forward.
13	O.SKB	Skip backward.
14	O.WRP	Write from PP memory.
15	O.WRPR	Write EOF/EOR from PP 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).
35	O.IDLE	Wait for stack request.
36	O.DROP	Drop PP.
37	O.SEEK	Issue overlap seek.

Stack Processor-System Interface

MTR functions used by the stack processor are described in this section.

The following system tables are used by the stack processor.

<u>Tables</u>	<u>Description</u>
Control Point Areas	Contain control point error flag, storage move flag, RA, and FL fields. The stack processor accesses but never changes these fields.
DST	All fields of the DST entry whose ordinal is placed in the stack processor input register by MTR are used. SPM makes all DST changes except one made by MTR during stack processor assignment to a PP.

<u>Tables</u>	<u>Description</u>
EST	Mass storage flag, unloaded flag, off flag, and DST ordinal are checked but not altered.
FET	Code and status field in the first word and error processing flag in the second word are accessed by SPM. IN and OUT pointers in the third and fourth words are accessed by the stack processor. Code/status is marked busy (even value) before a request enters the stack and is marked complete (odd value) when the request is executed.
FST	RBT/RB/PRU position pointers in the first word and code/status field in the second word are accessed by SPM. The code/status field has been processed the same as for the FET.
DAT	If there are no outstanding I/L stack requests, the stack processor clears the interlock word (W.DATIL) of the Device Activity Table upon termination of a stack request. The stack processor updates W.DATIL when it terminates in a PP with an outstanding I/L stack request.
RMSBUF	Stack processor (1SP/1SQ) formats and stores RMS hardware error diagnostics in the RMSBUF three-word area for DSD to display.
RST	Request scheduling table is parallel to the request stack area and is used by SPM to hold request scheduling parameters.
DDT	First and last DAM ordinals, MST ordinal, and EST ordinal are used by SPM in determining RBR ordinal of a permanent file set member and whether or not it is mounted.
RBR Area	All of the first header word, the EST ordinal, and available RB count bytes in the second header word are used by SPM. SPM assigns record blocks for a write request by searching the RBR table for available bits. When the request is terminated or reissued, 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 are used. The pseudo channel CH.RBT is reserved only when RBT word pairs are being removed from the RBT empty chain.
SCB	FIRST, IN, OUT, and LIMIT are accessed in the same manner as FET when transferring data between RMS and ECS.
R.STBMSK (PP resident)	Contains appropriate mask when calling R.STB; always returned to 7700 (octal), its normal value.

The following system routines and programs are used by the stack processor.

<u>PP Program</u>	<u>Description</u>
1SX	Stack processor auxiliary program called by MTR request for tasks that the stack processor cannot handle or does not have time to do. For example, the stack processor does not issue dayfile messages, because if the dayfile buffer is full, 1SP/1SQ and MTR could loop endlessly waiting for each other.
CEM	Auxiliary program called by MTR request to handle CERFILE logging and dayfile messages for DDP errors encountered by 1SP/1SQ.
MTR	System monitor initially calls the stack processor (via 1S5), when a request has been made for an inactive DST entry, and performs various functions for the stack processor while it is processing the request.
7ID	Stack processor auxiliary program called by 1SX. 7ID informs the operator (via a flashing message at the bottom of the B display) that the job associated with control point x has an outstanding request for an idle device.

<u>PP Resident Routine</u>	<u>Description</u>
R.DCH	Releases a channel reservation.
R.IDLE	Entered when a stack processor releases its PP.
R.MTR	Used for all MTR functions other than to reserve or drop a channel.
R.TAFL	Terminates access to the control point field length. When necessary, it interlocks storage moves during execution of a request and, when a request is terminating (except at control point 0), switches the stack processor back to control point 0.
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.

<u>Monitor Function</u>	<u>Description</u>
M.DPP	Releases PP assignment.
M.SPM	Used by the stack processor to call SPM. The three executives are: <ul style="list-style-type: none"> ● EX.SPRCL Stack processor recall; terminate a stack request. ● EX.STAT Change status; get next stack request. ● EX.NXTPB Get next PB/PRU.
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.

SPM-1SP Interface

The PP communication area through which SPM and 1SP communicate consists of a PP input register (PPIR), PP output register (PPOR), and a six-word PP message buffer (PPMES1-6). The PPIR, used to call 1SP, comes directly from the second word of the DST and contains initialization information such as drive, name, equipment, and channel number. The PPOR is used by 1SP to initiate SPM via the M.SPM monitor call in addition to other monitor calls. PPMES1-2 are used for up to five byte pairs (corresponding bytes of PPMES1-2) of overlap seek information and for the 1SX error information interface. PPMES3 is for SPM internal information although 1SP supplies the device PB/PRU count in the rightmost byte during initialization. PPMES4-6 is the internal format stack request that SPM gives to 1SP. The order field in PPMES4 is monitored by 1SP to determine what work is to be done. The status₁, flags₁, and flags₂ fields are described under SPM-1SQ Interface in this section.

The format of the 1SP communication area is:

	59	50	47	41	38	35	29	23	20	17	11	0	
PPIR	1 S		P O		Driver Name	DST Ordinal	F	O	CH				
PPOR	M.xxx or 0		Dependent on M.xxx Call										
PPMES1†	DDT Ord. ₁	Status 1	DDT Ordinal _i and Status _i (Up to Five); 0 if No Unit Specified										
	1SX Error Information												
PPMES2†	PB ₁	PB _i (Up to Five); 0 if No PB											
	1SX Error Information												
PPMES3	RBT Ordinal	Release EST	Byte	PRU		Flags ₁		PB/PRU					
PPMES4	Next PB	Next Range Next PB/PRU				Starting PRU		Last PRU + 1		Same as Original Stack Request			
	Number of PRUs Transmitted	Flags ₂	FET Code/Status										
PPMES5													
PPMES6	Current PB/PRU								EST Ordinal				
	PB	First PRU		Last PRU + 1		Current PRU							

† Example shows three units specified for overlap seek with zero terminator.

SPM-1SQ Interface

SPM and 1SQ use the PP communication area similar to SPM and 1SP. The difference is that SPM sends and receives physical disk addresses to and from 1SQ in cylinder, track, sector, and number-of-sectors format rather than PB, first, and last PRU format. This moves the conversion of physical addresses to the CPU saving time and space in the PP.

The format of the 1SQ communication area is:

	59	50	47	41	35	29	23	20	17	11	0	
PPIR	1 S		Q 0		Driver Name	DST Ordinal	E	0	CH			
PPOR	M.xxx or 0		Dependent on M.xxx Call									
PPMES1	DDT Ordinal ₁	Status ₁	DDT Ordinal _i and Status _i (Up to Five); 0 if No Unit Specified									
PPMES2	PB Number											
PPMES3	RBT Ordinal	Release EST	Byte	PRU		Flags ₁		PB/PRU				
PPMES4	Next Cylinder	Next Track	Next Beginning Sector	P	Next Sector Count		SPM → 1SQ 1SQ → SPM					
	No. of PRUs Transmitted	Flags ₂	FET Code/Status									
PPMES5	Same as Original Stack Request											
PPMES6	Current Cylinder	Current Track	Current Beginning Sector	Current Sector Count		Current PRU		EST Ordinal				

The following coded values are contained in the status₁ field of PPMES1 (bits 50 through 48).

Code	Significance
7	Controller/unit reject.
6	Unit busy.
2	On cylinder.

The following flags are contained in the flags₁ field (PPMES3, bits 23 through 12).

Bits Set	Significance
23	End of RB flag; indicates end of RB.
22	Recording mode flag.
21	Gap sector flag; if set, the pack was written with no gap sectors.
20	DST flag.

<u>Bits Set</u>	<u>Significance</u>
19-18	Device type. <ul style="list-style-type: none"> 0 All devices other than 844 or 885 disk drives. 1 844-21 disk drive. 2 885 disk drive. 3 844-4x disk drive.
17	Release flag; if set, 1SP releases the interlock unit.
16	Interlock flag; if set, stack request is from interlock user.
15	Access flag; if set, CM access is set for the CM I/O request.
14	EOF flag; if set, the current cylinder is at EOI (for forward) or BOI (for backward).
13-12	Type of stack request; set by SPM. <ul style="list-style-type: none"> 0 Normal write. 1 Skip backward. 2 Direct I/O. 3 Others.

The following flag is contained in the P field (PPMES4, bit 35).

<u>Bit Set</u>	<u>Significance</u>
35	New PB flag; indicates new PB.

The following flags are contained in the flags₂ field (PPMES4, bits 47 through 42).

<u>Bits Set</u>	<u>Significance</u>
47	If set, 1SQ desires to drop. SPM does not assign a stack request if it recognizes this flag. Otherwise, SPM automatically assigns a stack request following an EX.SPRCL request from 1SQ.
46	If set, no PB was assigned at the last NXTPB request.
45	If set, interlock is broken.
44	If set, the unit is idle.
43	If set, a fatal error was detected.
42	If set, EOR/EOF was read.

RMS Device Capacity Limitations

The stack processor does not use all the space physically available on 844 and 885 devices. The innermost cylinders are unconditionally reserved for use by on-line disk diagnostics.

The ranges of cylinders used by the stack processor on each type of disk device are shown in table 5-14. These limitations must be observed by any installation that wishes to use CDC on-line disk maintenance software.

TABLE 5-14. RANGES OF CYLINDERS USED

Device	Cylinders Used		Reserved for CTI	Cylinders Physically Available
	Octal	Decimal	Decimal	Decimal
844-21	0-623	0-403	407	0-410
844-4x	0-1447	0-807	819	0-822
885	0-1506	0-838	840	0-842

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. If IP.DEBUG is zero, these conditions are treated similarly to other errors. An error code is placed in the code and status field of the FET and FST entries and the control point is aborted if the error processing (EP) bit in the FET is zero. If IP.DEBUG is not zero, an invalid MTR function is issued with the stack processor output register having 77₈ in byte 0 and an error code in byte 1.

END OF INFORMATION - The error code 01g is inserted in bits 13 through 9 of the code/status field, but no message is issued and the control point is not aborted (nonfatal 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, and mispositioning. The PRU is reread or rewritten up to 63 times. 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 03g (dayfile message RECOVERED PARITY ERROR), but this condition does not affect code/status or abort the control point. If all 63 attempts fail, 1SX is called with code 04g (dayfile message UNCORRECTABLE PARITY ERROR), 04g is put into bits 13 through 9 of code/status, and the control point is aborted if the EP bit is zero. A request at control point 0 is not aborted.

When an uncorrectable parity error occurs for a READ request and the EP bit is not zero, request execution terminates with the bad PRU being the last one read. When an uncorrectable parity error occurs for a WRITE request and the EP bit is not zero, request execution terminates with the FET pointer positioned after the last good write operation and the file positioned after the bad PRU.

BUFFER PARAMETER ERROR - This error occurs when a request is processed that references an FET when not all of the following conditions are satisfied.

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

1SX is called with code 11g (dayfile message BUFFER ARGUMENT ERROR), error code 22g is put in bits 13 through 9 of the code/status field, and the control point aborts if the EP bit is zero.

UNDEFINED ORDER CODE - A request contains order code 07g. 1SX is called with code 22g (dayfile message INVALID STACK ENTRY), error code 22g is put in bits 13 through 9 of the code/status field, and the control point aborts if the EP bit is zero.

NO FET FOR O.RMR - A request contains order code 06g (O.RMR), but no FET is specified. 1SX is called with code 11g (dayfile message BUFFER ARGUMENT ERROR), error code 22g is put in bits 13 through 9 of the code/status field, and the control point aborts if the EP bit is zero.

ADDRESS OUT OF FL FOR O.RMR - The address for a table of disk addresses for O.RMR is out of field length. 1SX is called with code 22g (dayfile message INVALID STACK ENTRY), error code 22g is put in bits 13 through 9 of the code/status field, and the control point aborts if the EP bit is zero.

INTERLOCK BROKEN - A group of interlocked stack requests are interrupted by a malfunction of a controller and/or unit. The stack processor puts 24g into bits 13 through 9 of code/status field.

RMS HARDWARE ERROR - An RMS hardware error is reported when any of the following errors occur on a device.

- Unit not ready.
- Positioner not ready.
- 6681 internal/external reject.
- Unit busy too long.
- Channel stays active after connect or function.
- Unable to connect.
- No status returned.
- Address byte not accepted.
- No disconnect on status request.
- Abnormal on seek.
- Channel not active after ACN.
- Irrecoverable write error between 7154 coupler buffer and disk.

These error conditions are reported by the stack processor via a flashing message at the bottom of the B display. The function on which the error condition occurred is retried until it is recovered or until the device is idled down by the operator. In either case, the error diagnostic is cleared. If the device is idled, 22g is returned to bits 13 through 9 of code/status and 7ID is called. The operator is notified of the job name associated with the idled equipment via a flashing message at the bottom of the B display. 1SX is called when the operator acknowledges the message (the message is also sent to the dayfile), and the control point is aborted if the EP bit is zero.

Dismountable Pack Processing — I/O Detail

Figure 5-3 shows the flow of control of disk I/O, including the processing of dismountable devices.

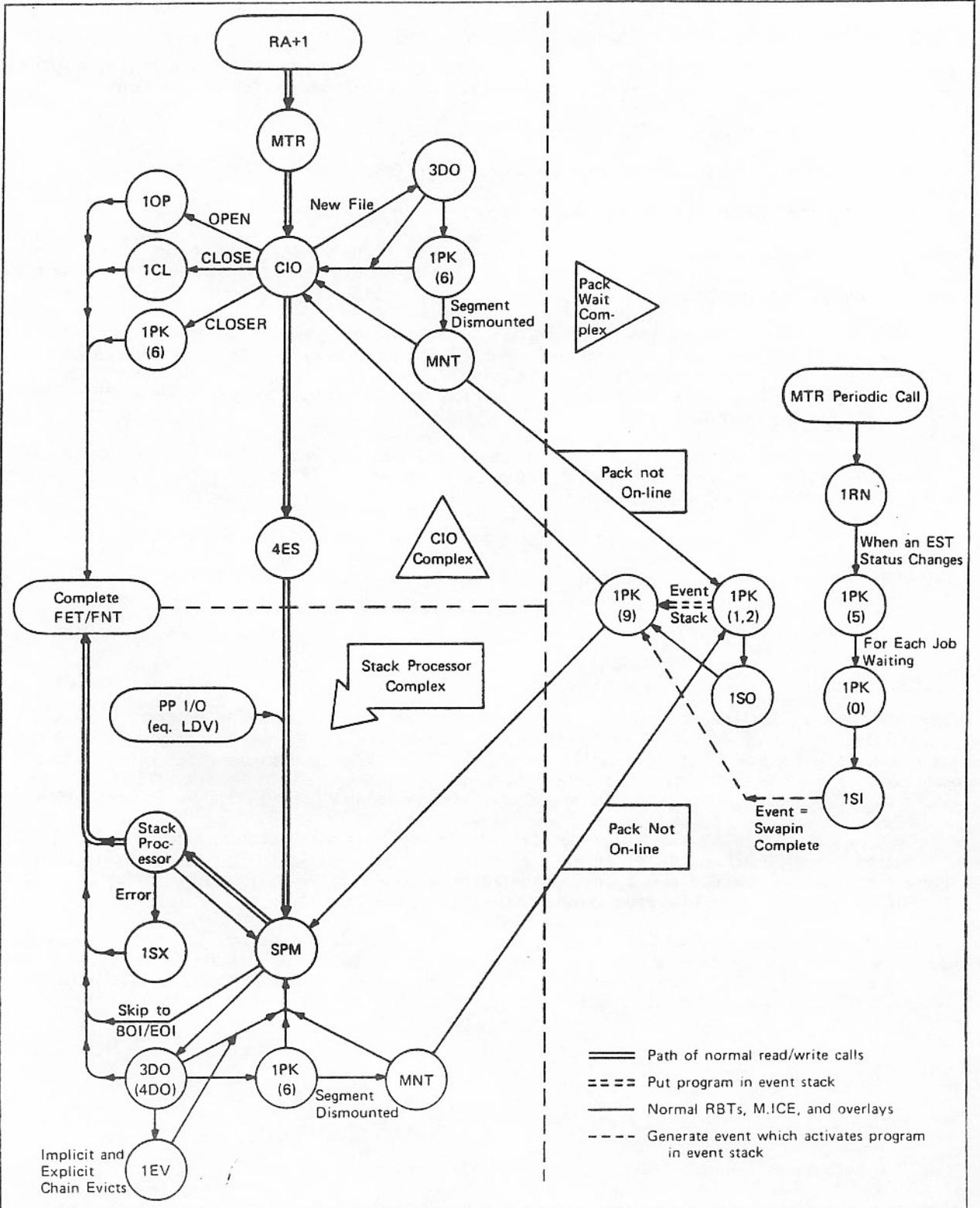


Figure 5-3: Device Set I/O Processing

Normal Calls for Read/Write

The user requests I/O by calling CIO in RA+1. If the file is new, CIO calls 3DO to assign it to a specific pack. If it is an existing assigned disk file, CIO loads 4ES which performs the following actions.

- Performs random positioning.
- Generates the stack request which accomplishes the function.
- Issues the function to SPM via the PP resident routine R.EREQS.

SPM processes all stack requests. Some system routines (JANUS, overlay loads, batch terminals) send requests directly to SPM rather than use CIO. SPM performs release-chain functions except for those file segments which are not on line.

SPM first determines if the function is a read or write function and whether the current segment of the file is on line. If the function is a normal write (not REWRITE), SPM assigns RBs to the file based on the amount of data in the buffer. If there is no space in the file and no free space in the current RBR, SPM sends the request to 3DO. When the write completes and too much space has been assigned, SPM is again called to remove any extra RBs.

SPM next determines which DST this disk belongs to and adds the request to that DST's chain. MTR ensures that a stack processor or a DST is active by checking the DST chain pointers. If there is no activity, MTR activates a stack processor.

The stack processor processes I/O to an arbitrary point, for example, to a cylinder boundary, and then may reissue the request to SPM. The stack processor always returns the request to SPM if the DAM ordinal in the RBT changes; thus, the ordinal 777 in overflow word pairs always causes a request to return to SPM and then to 3DO.

SPM-3DO Interface

SPM forwards to 3DO all requests which fail at any point in the preceding description: BKSPRU, release-chain not mounted, SKIPF or SKIPB with count=777777g, file is new and has no RBTs (has not been assigned), position is at an overflow word pair (DAM=777g), and so on. 3DO acts as the stack processor for the first DST, which is reserved; actual disk controllers begin at the second DST. Thus, SPM forwards 3DO requests via the same mechanism as normal requests by putting them on the first DST chain.

The only task of 3DO is to select a disk for files requiring space to write, such as new files. Other tasks are passed on. BKSPRU and SKIPF/SKIPB with count=777777g are sent to 4DO for completion. If the current segment of the file is not on a mounted pack, the function is not a write (as opposed to REWRITE), or if the file still has space available to write in, 1PK is called.

If the FNT indicates a new file (no RBTs), 3DO obtains a word pair and changes the format from a new file to an existing file by moving the flags from the FNT to the RBT. Next, the set for the file is chosen. If SN was specified, no selection is made. If SN is not specified, PF, Q, or SYS (SYS is available only by macro) is selected as specified. If none is specified, a flag is set to indicate that a scratch set is required.

Order of device selection is:

- The device must belong to the required set.
- If PF, Q, or SYS, the device must also have that attribute.
- If a device type, allocation style, or VSN is specified, attempt to match it; if no match, but DV was specified, repeat the attempt ignoring the requirements of VSN, device type, and allocation style.

Selection is limited to the allowed devices. Controller and unit activity from the DAT and available space in the RBR are factors in selection of the optimal device. 3DO creates an overflow word pair for all but new files, creates an empty word pair for the selected device, and reissues the request to SPM. 3DO considers only mounted devices. If 3DO finds no space available and this is not a public set request, 3DO calls 1PK to consider selection of an unmounted member of the set.

Once the file is written, its device set attribute is fixed.

1PK may be called by CIO, MNT, ADS, or independently via M.RPJ. 3DO cannot load 1PK via R.OVL because 3DO is considered a stack processor and, therefore, not allowed to use M.DFM or other I/O because it could lead to a deadlock condition. 1PK issues dayfile messages and can be accessed by 3DO only via a call to M.RPJ. The calling routine places one of the following function numbers in the PP input register byte 2 to specify the 1PK function desired.

Function 6 - Calls from 3DO are always made using function 6, and 1PK is called via M.RPJ. The call from CIO for CLOSER is also done in this way, so 1PK must determine if CIO is called and if the function is CLOSER. Space must be assigned if the function is not CLOSER, the function is a write (not REWRITE), and input register byte 4 is zero indicating a write at EOI. Assuming 3DO checked the mounted devices, the SMT is read and the file assigned to a pack with available space which is not mounted, considering first the on-line devices, then those not present. If there is no space and the user has not set UP, the user is aborted with error 10 (device capacity exceeded). 1PK selects a device and calls MNT as an overlay. When the mount is successful, MNT reloads 1PK which reissues the stack request. If the pack is not on line, 1PK function 9 is put on the event stack with the stack request in its message buffer (refer to functions 5 and 9).

If UP is on and there is no space, the FET is completed with error 10 (device capacity exceeded). If the current position is not on line on a public set, the job is aborted.

If not CLOSER and not a write, the first step is a mount. First, advance position past any overflow word pair; check if required disk is mounted. If it is, reinitiate the function by calling CIO, or SPM if CIO is not in input register. If not CIO, call MNT, which calls 1PK back and the process repeats if MNT found the pack on line and mounted it. If not, MNT calls 1PK function 2 to swap the job out while waiting for the pack. When the operator puts the pack on and turns on the EST, 1PK function 5 is called by 1RN, which periodically checks on/off changes. Function 5 causes the removal of all waiting 1PKs from the event stack and the stack request is reissued. MNT finds the pack on line and I/O proceeds.

CLOSER is processed by CIO calling 1PK function 6. When 1PK detects a CLOSER call, 1PK processing depends on whether the position is EOI. If EOI, a dummy overflow word pair is attached on the end of the RBT. The DAM and VSN fields are zero. Both EOI and EOV status are set in the FET.

If the current position is not EOI, the file is positioned to the next EOV word pair or to EOI, if it comes first. If the current position is an EOV word pair, nothing is done. EOV status is returned to the FET if the new position is an EOV word pair or EOI is encountered.

If MNT finds the operator dismount flag set in the SMT, it loads 1PK mode 7, which calls 1PK mode 8 with a delay, which reissues the stack request. This continues until the pack goes off line or is remounted.

Functions 1 and 2 - These functions are called by MNT, when a required pack is not on line, to delay the job until the pack becomes available. The SN/VSN are put into the variable area of the DDT if not there. The JDT enters the queue with others that may be waiting on the DDT. 1PK function 9 is put in the event stack on the JDT swap-in, and the job is swapped out by macro C1SO. The initial PP input register from the caller is sent to the message buffer. Stack requests are added to the message buffer.

Function 5 - Function 5 is called by 1RN when an EST free, busy, or off status changes (becomes different from the DDT). The fixed DDT for this EST is updated. If a pack has come on line and appears in the variable DDT area, 1PK function 0 is called for each job queued on the DDT and the job is cleared.

Functions 8 and 9 - Function 8 swaps the job in. Function 9 reinitiates the I/O function. If the input register was CIO, function 9 reissues it to CIO. If not CIO, the stack request is in the message buffer and is reissued to SPM.

Removing a Pack - DELSET and DSMOUNT

Both DELSET and DSMOUNT can be used to make a disk unavailable to the system. DELSET removes a disk from set membership and requires that there be no files on it. DSMOUNT makes a disk unavailable to a job. The operator dismount command, DMNT, permits the operator to remove a pack from a drive. DELSET uses PP routine DLM; DSMOUNT and DMNT use PP routine DSM.

DLM and DSM must halt all stack request activity before a disk can be removed from mounted status. To do this, they set the request idle bit in the EST. 1PK checks the stack request area for requests for this unit, and if it finds none, sets the EST status to FB=11 to indicate there is no activity. Any subsequent requests for the pack cause the requesting job to be swapped, and DLM or DSM can put the pack into unavailable status.

DELSET removes a disk from a set by zeroing its entry in the SMT (which defines set membership). Before this is done, DLM searches the RBT for local files and the PFC for permanent files resident on this disk and checks that the SMT usable count matches the total DAM available counts (if the disk is in dismounted status). If any test fails, DLM aborts the job.

A DSMOUNT call from a job (including the automatic one at the end of job) decrements the set's activity and resets it to dismounted status if no jobs reference it. The operator command DMNT resets the disk to dismounted status as soon as all I/O clears and sets the operator dismount flag in the SMT so that no other mainframe can mount the disk. Only an RMNT command or a RECOVER clears this flag.

If the pack is mounted on this mainframe, an operator dismount proceeds as follows:

1. The EST request idle bit is set, which locks out all I/O.
2. The routine waits until all activity stops on the disk (signaled by FB being set to 11); then it clears the request idle bit.
3. If the device is not shared, the RBR is translated into a DAM and written on the disk, the mounted flag is cleared from this mainframe's bit in the SMT entry, and the DDT is written in dismounted format (S.DDSN=1).
4. If the device is shared, the RBR is not written to the disk. The PP reads the DAM, clears all bits in the DAM which are clear in the RBRs, and then writes the DAM back. Other processing is the same as in step 3.

If the pack is not mounted but is on line on this mainframe, DSM sets the EST to FB=11.

If another mainframe has the pack mounted, DSM terminates with an operator message that the pack cannot yet be removed. It sets the EST to FB=10 (free). The pack cannot now be mounted unless the operator enters the RMNT (remount) command. The operator can attempt DMNT again later.

If no other mainframe has the member mounted, DSM sets the EST to off and FB=0 (no pack on line). The operator can then remove the pack.

Extended Memory-Buffered I/O

Reading and writing of large sequential RMS files is greatly enhanced by the use of extended memory buffers. Such operations involve the use of a small CM buffer in the user's field length and a large user's buffer in extended memory. The data is transferred between extended memory and the RMS device either through a system circular buffer (SCB) in CM or through a distributive data path (DDP). Extended memory buffering is not available on a CYBER 176. Refer to the section on 819 Disk I/O Processing for a description of 819 buffered input/output. The following describes a write sequence involving an extended memory buffer; a read sequence is essentially the reverse.

The user requests extended memory buffering on a file-by-file basis through the REQUEST control statement or macro. On the control statement, the user includes an EC parameter in addition to the normal parameters in one of the following forms.

<u>Parameter</u>	<u>Use</u>
EC	For a default (IP.BUF) size buffer.
ECxxxx ECxxxxK	For a buffer of xxxx-thousand (octal) words.
ECxxxxP	For a buffer of xxxx (octal) pages.

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

In a write sequence, the user first puts data into a CM buffer, which need be only about 200₈ words long, then issues a CIO call through RA+1. If an XJ instruction follows the request in RA+1, the job is exchange-jumped out of execution, and CP.MTR begins processing the request.

CP.MTR recognizes the CIO call and passes it to CP.CIO for processing. Extended memory-buffered file I/O causes CP.CIO to perform a validity check on the FET and activates the proper extended memory driver. The data is then written directly from the user's CM buffer to the user's buffer in extended memory.

The preceding process continues until the user's extended memory buffer is full; then a stack request is generated by CP.CIO, requesting that the extended memory buffer be written to an RMS device. In processing the request, the stack processor loads the appropriate extended memory executive routines into an area of stack processor memory.

When the stack processor requests an SCB, CBM (system circular buffer manager) assigns the first one available from the list of SCBs. Each SCB has an integral number of PRUs, so that PRUs are not split across the end of the buffer. When a CM buffer path is selected, the stack processor uses the SCB in the normal circular I/O mode, with the following modification. In addition to FIRST, IN, OUT, and LIMIT, the FET-like SCB control table also contains a TRIGGER and a DIRECTION field. Before the transfer, the stack processor puts an M.SCB in its PPOR. During the transfer, MTR checks to see if the trigger has been reached. If so, it calls CBM to process the SCB; if not, no action is taken. This circular I/O continues until the extended memory buffer has been emptied and all data has been written out to the RMS file. The processing is designed to prevent the stack processor from missing disk revolutions during an I/O buffer transfer.

When the DDP is selected, the processing is similar. CP routines do the same bookkeeping as in CM, but only point to the data in extended memory instead of transferring it to CM. As a result, less CM is used for the transfer, n words instead of $65_{10} * n$ words for an SCB containing n PRUs.

Figure 5-4 illustrates the general flow of output to an extended memory-buffered RMS file. Either data path may be assigned dynamically, depending on availability.

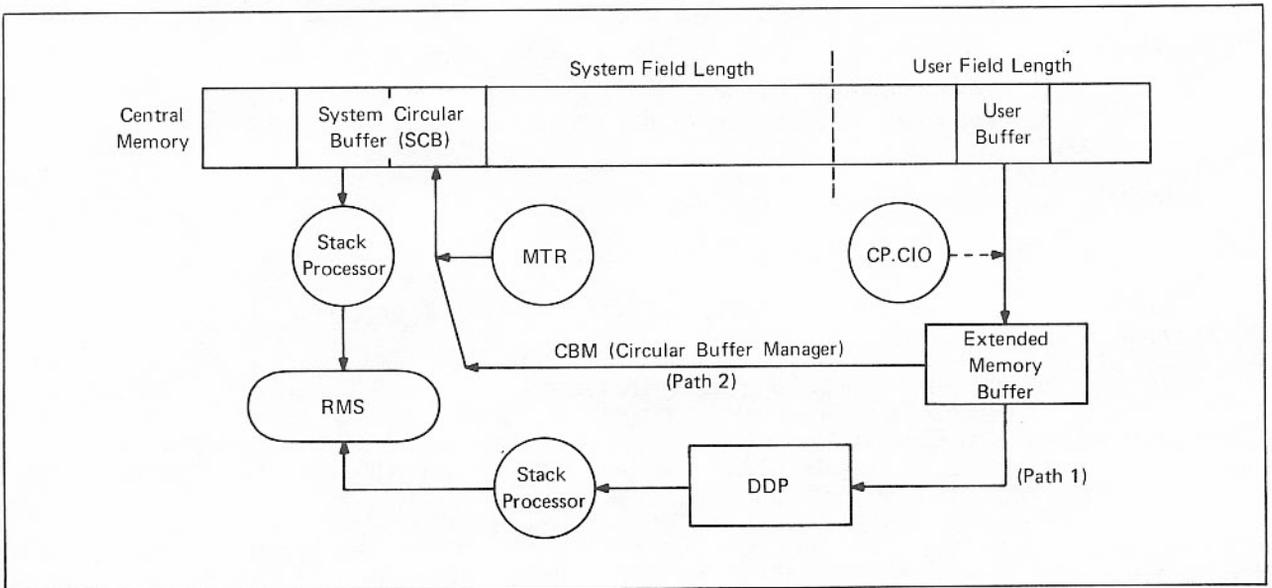


Figure 5-4. Output Flow to RMS File (Two Paths)

819 DISK I/O PROCESSING

The CYBER 170 Model 176 computer uses a peripheral processor subsystem consisting of 10 or 20 peripheral processors (PPs) and up to six first level peripheral processors (PPUs) to perform I/O tasks (figure 5-5). The peripheral processor subsystem communicates with all standard NOS/BE equipment such as card readers, line printers, magnetic tape units, and so on. The PPU can communicate only with an 819 disk drive. Because a PP cannot access the 819 disk directly, a special method of I/O processing is necessary when a request involving the 819 disk is encountered. This method uses LCM as an intermediate buffer area. Data is transferred between the 819 disk and LCM through the PPU and a hardware buffer in CM and between LCM and the requesting control point or PP. The remainder of this section describes the logical and physical I/O processing involved in the data transfers, PPIO processing for the 819 disk, LCM buffer management and tables, and the CE error files.

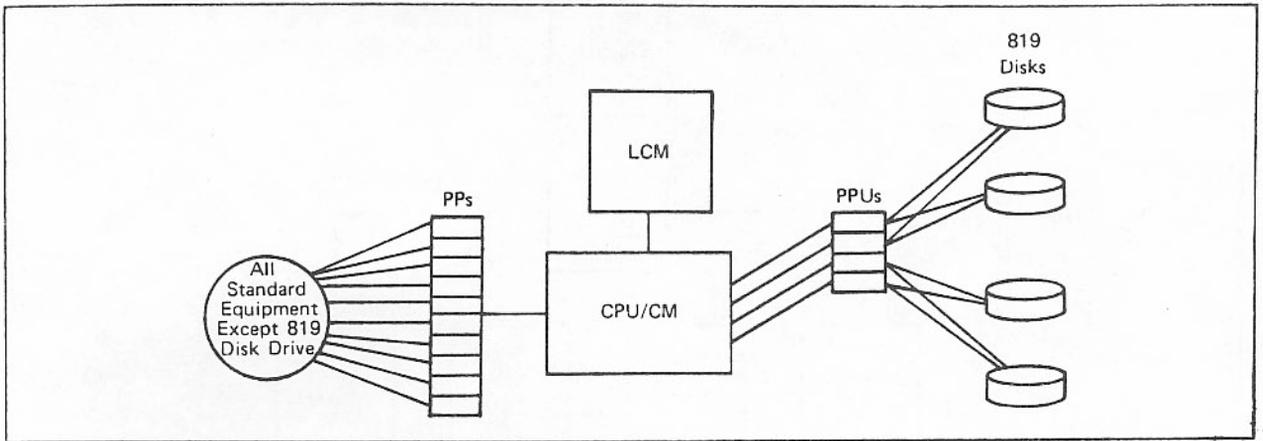


Figure 5-5. CYBER 176 Computer System with 819 Disks

LOGICAL I/O PROCESSING

General Description

The following discussion describes the flow of 819 disk logical I/O processing (figure 5-6). The segments labeled HDRV, IH, and HDC are part of a sequence of routines which handle the physical I/O. Physical I/O is discussed in detail in this section.

When a user requests that an 819 disk file be read, written, or repositioned, CPCIO/CP4ES makes a stack entry and assigns the request to SPM. SPM calls the segment BFM which allocates a buffer area and a transfer buffer table (TBT) in LCM. (Refer to LCM Buffer Management in this section.) The TBT contains the current stack request and information about the data in LCM for each 819 disk file. If the required number of buffers in LCM are not immediately available for allocation, the TBT is put on an empty TBT chain where it can be processed at a later time. CPMTR recalls BFM periodically to process the empty chain.

Segment BFM adds the TBT address to the TBT address table (TAT). BFM also initially creates TAT in the paged area of LCM.

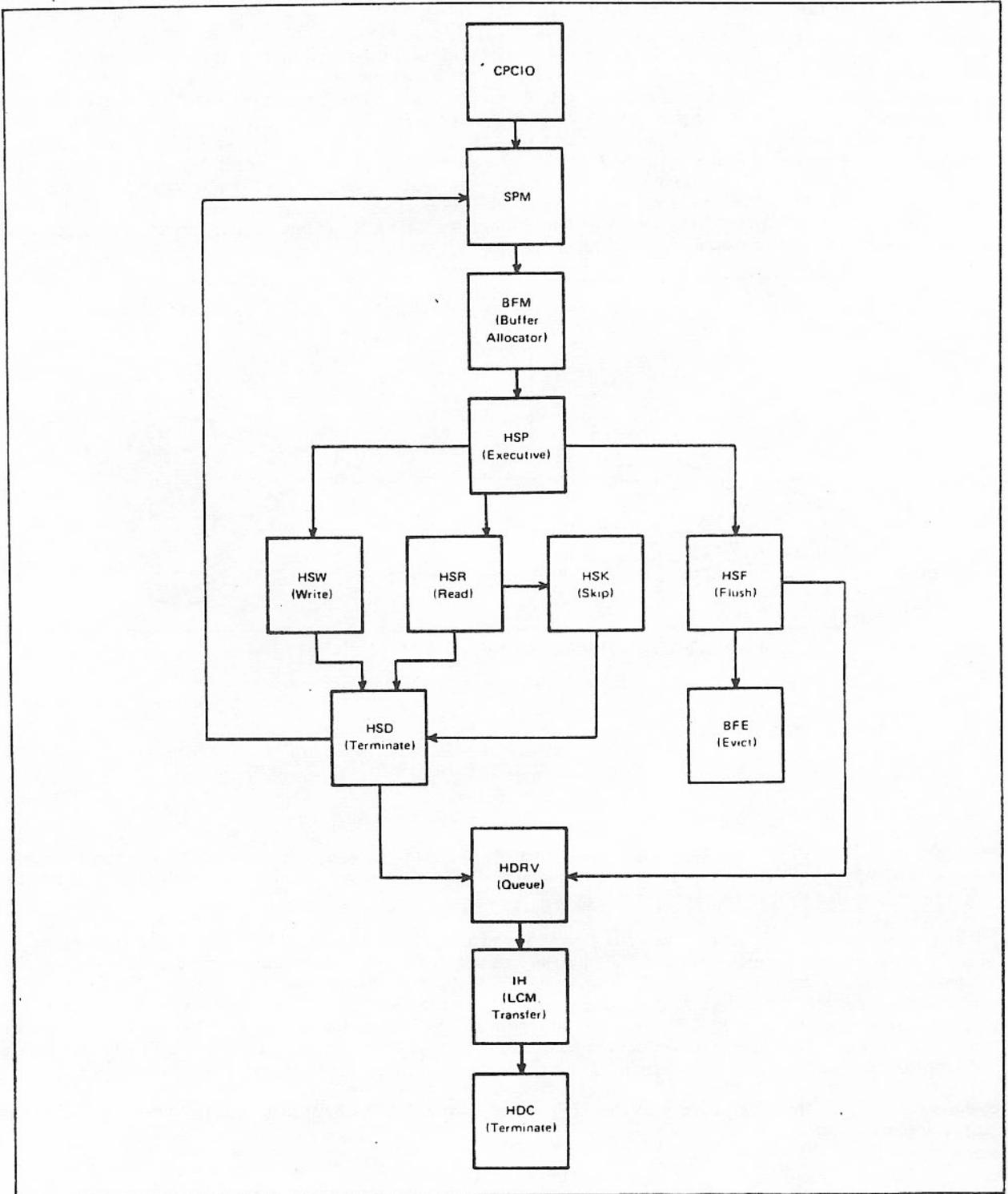


Figure 5-6. Logical I/O Processing

BFM then calls HSP, the 819 disk executive. Depending on the operation requested, HSP calls one of the following segments.

<u>Segment</u>	<u>Description</u>
HSW	Write operation.
HSR	Read operation.
HSK	Skip operation (called through HSR).
HSF	Flush operation.

If, for example, the user wants to write a file on an 819 disk, HSP calls the segment HSW. HSW gets the starting PRU from the stack processor and begins transferring data from CM to the buffer in LCM. HSW saves short PRU headers, builds the sector header PRU flags, and updates IN† and INW† in the TBT. When the end of the 819 sector is reached, one of the following occurs.

- If the limit PRU is equal to the current PRU and the next PB† is zero, SPM is called to allocate another PB. If another PB is not available, the request is terminated for reissue.
- The next PB is moved to current PB and data transfer continues until the user buffer is empty and/or the LCM buffer is full.

When the LCM buffer threshold is reached (buffer is half full), HSD calls HDRV with a request to initiate transfer of data to the 819 disk. HDRV places the request in the unit queue table (UQT). The UQT holds all 819 disk requests received from HDRV for each unit. When the proper channel is available, HDRV assigns the channel, formats the request for the PPU driver, and calls the interrupt handler. The interrupt handler (IH) moves data between LCM and a hardware buffer in CM and sends a request to transfer data to the PPU. The PPU disk driver (HCD) writes the data from the hardware buffer to the 819 disk. Segment HDC is then called to complete the request. HDC is discussed further under Physical I/O Processing in this section.

If the write request was not completed when the LCM buffer was filled, HSD sets the wait-on-disk flag before calling HDRV. When the interrupt handler reads or writes a sector from a TBT which has wait-on-disk status set, it calls HSP which recalls HSW to continue processing the request. HSW transfers data until the request is satisfied or the CM buffer is empty. This sequence of calls allows the LCM buffers to be circular buffered. When the request is satisfied or the CM buffer is empty, HSW sets the stack request completion flag and returns control to HSD. HSD calls SPM which drops the stack request and sets the FST and FET completion bits.

If the user wants to read a file on an 819 disk, HSP calls the segment HSR and essentially the write operation is reversed. The request to read the disk is given to HDRV, which places the request in the UQT. The PPU driver, HCD, reads data from the 819 disk to the hardware buffer and IH transfers the data to LCM. HSR then moves the data from LCM to CM.

HSR gets the starting PRU number from the stack request and checks the beginning PRU and PB values in the TBT to determine if data has been read from the disk. It stores the level number to be transferred to the FET by SPM and transfers the buffer, updates OUT† and OUTW† in the TBT, and processes sector error flags. When the end of a PB is reached (the next PB in the TBT is zero), HSR gets the next PB from SPM.

† Refer to the transfer buffer table in appendix E for an explanation of fields.

HSR transfers data to the user's field length in CM until the next PRU will not fit in the area or the request is satisfied. When the request is satisfied, HSR sets the stack request completion flag and returns control to HSD. HSD calls SPM which drops the stack request and sets the FST and FET completion bits.

If the file was found to have a request currently outstanding before the read or write operation, HSR checks the outstanding operation. If it is a request to read, and the PRU being read is not the requested PRU, HSR calls HSF, which calls HDRV to terminate the unit queue entry and set the flag to stop the transfer.

The HSK segment is called by HSR to skip logical records forward or backward. HSR gets the correct PRU from the stack request and searches for the end-of-record. If the desired PRU position is found in the LCM buffers, HSK performs the repositioning forward or backward depending on the stack request code. If the end-of-record is not found in the buffers, HSK returns control to HSR which repeats the process until EOI or BOI is found.

HSP calls the HSF segment when it receives a request which indicates a mode change (read to write operation or write to read operation). HSF is called directly from CPMTR for a monitor flush command or an evict request. A flush command is made when a file must be returned (such as before swap-out or roll-out) between job steps, and at the end of the job. When the job is swapped or rolled back in, the first reference to the file reinitializes the TBT.

For empty buffers and inactive read buffers (last operation performed was a read), HSF calls BFE to clear the TBT and release the LCM buffers. For buffers currently active and write buffers (last operation performed was a write), HSF queues the TBT to the physical I/O segments (HDRV, HDC, and so on), which later recalls HSF. If I/O is complete, HSF flushes the inactive write buffers. The TBT is cleared.

Logical I/O Segments

The following segments are involved in 819 disk logical I/O processing.

<u>Segment</u>	<u>Function</u>
BFM	Allocates a TBT and LCM buffer areas for each file.
HSP	819 executive; calls routines to perform allocation, reading and writing data between LCM and user's field length in CM, repositioning within files, and flushing of data. HSP performs the following sequence of actions. <ol style="list-style-type: none">1. Checks whether storage access is allowed.2. Checks FET for valid parameters.3. Compares new request to last request. If there is a mode change (read to write or write to read) or evict request, calls HSF.4. If read request, calls HSR.5. If write request, calls HSW.
HSF	Flushes and evicts files. Performs the following actions. <ol style="list-style-type: none">1. Drops inactive read buffers and associated TBT.2. Flushes write buffers.3. Drops inactive write buffers and associated TBT.

<u>Segment</u>	<u>Function</u>
HSR	<p>Initiates read operation, or if skip request, calls HSK. If read request, performs following actions.</p> <ol style="list-style-type: none"> 1. Compares buffer position to requested position. 2. If positions are the same, calls CHR to transfer data from LCM to CM. 3. If positions are incorrect, HSR sets up a disk request to read data from disk.
HSW	<p>Initiates write operation. Performs following actions.</p> <ol style="list-style-type: none"> 1. Compares buffer position to requested position. 2. If positions are correct, calls CHW to transfer data from CM to LCM. 3. If positions are incorrect, calls HSF to flush the buffer.
HSK	<p>Skips logical records forward or backward. HSR fills several LCM buffers and calls HSK when the requested position is in the buffers. Based on the stack request code, HSK calls SKF to skip forward or SKB to skip backward. For a skip backward, the LCM buffer area is subtracted from the file position so that the maximum amount of data behind the current empty buffer position is read.</p>
HSD	<p>Terminates stack request if the request is completed. If the requested operation has caused a buffer threshold to be reached, HSD puts the request on the unit queue (in the UQT) to be processed.</p>

PHYSICAL I/O PROCESSING

General Description

The preceding section on logical I/O described the sequence of segment calls leading to HDRV and IH. This section describes, in detail, the segments involved in actual physical I/O processing for the 819 disk (figure 5-7).

HDRV is called to queue I/O requests for the 819 disk. It calls HDIN, only once, to initialize the UQT and channel table (CHT) after a deadstart. HDRV enters the request in the unit queue (UQT) and, if it is the only request in the queue, calls HDSL. HDSL selects the best request for a unit to perform I/O, selects the best channel, and calls the interrupt handler (IH). IH puts the request in the hardware I/O buffer and contacts the PPU through the disk driver HCD. When the PPU acknowledges the request, IH calls HDRQ. HDRQ checks for a request on another unit and, if it finds one, calls HDSL. HDSL sends the request to the PPU as described previously.

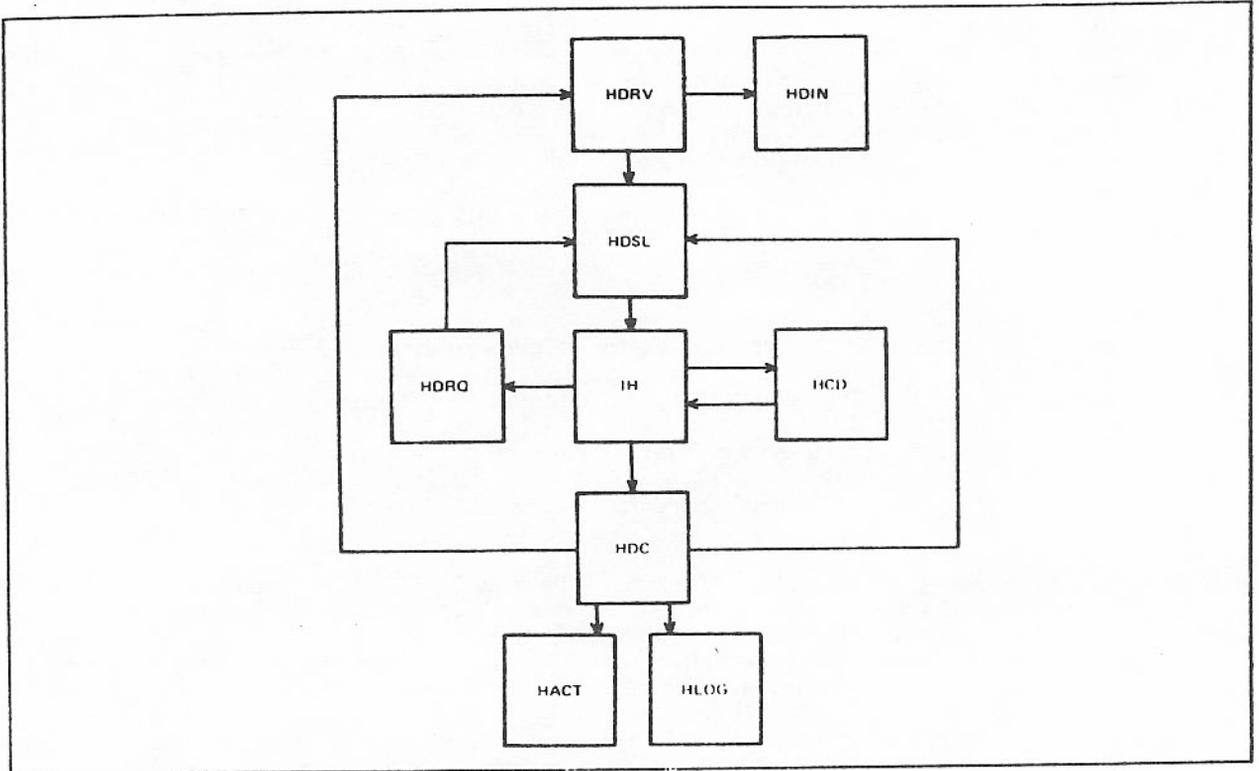


Figure 5-7. Physical I/O Processing

When the PPU finds one of the units on the proper cylinder and sector, it communicates this to the IH, and data is transferred to or from the disk. At the end of the data transfer, IH calls HDC to terminate the request. HDC may call HDRV to requeue the request, HACT to record activity status, or HLOG to record disk errors. (Refer to CE Error File in this section.) If there are any other requests on any unit, HDC returns to HDSL.

Physical I/O Segments

The following segments are involved in 819 disk physical I/O processing.

<u>Segment</u>	<u>Function</u>
HDRV	Queues 819 disk requests. Enters the TBT in the UQT according to ascending order of cylinder addresses. Within the cylinder, TBTs are positioned in the order in which they were received. If there are no other TBTs in the queue, I/O is initiated for that request.
HDIN	Initializes the UQT and CHT based on the EST and DST. HDIN is called only once after a deadstart.
HDSL	Selects best TBT for I/O transfer and best channel. HDSL bases its selection on the cylinder address of the forward TBT, backward TBT, and the current position of the unit. It assigns the unit to the best channel, sets up the request for HDC, and calls the interrupt handler.

Segment

Function

IH

Transfers data between LCM and the PPU I/O buffers. The following sequence of events occurs.

1. Master output IH sends disk request to master PPU.
2. Master PPU receives request and initiates master input IH.
3. Slave PPU calls slave input IH to start I/O link-up.
4. Master and slave PPUs call output IH to transfer data from LCM to the hardware buffer or input IH to transfer data from the hardware buffer to LCM.
5. Slave PPU calls input IH to terminate all input and output disk requests.

The interrupt handler can perform the following actions.

- Handle 513-word transfer size, picking up first word from the TBT and remainder from LCM buffer.
- Transfer to LCM buffer based on TBT pointers; update the buffer flags, IN and OUT fields.
- Recognize a switch to a new PB and send the next head/sector to the PPU in the middle of a request.
- Decide one sector in advance whether to do continuous transfer based on the following conditions.

If the next TBT in the UQT is on the same cylinder and is the same type of request, IH automatically switches to that request. HDC is notified to terminate the previous request.

If next PB on the current file is on the same cylinder. IH continues on the current TBT. IH sets last PB equal to current PB and current PB equal to next PB.

- Calls HSP after a sector transfer if the wait-on-disk flag is set.

HDRQ

Searches for requests. HDRQ is called by IH when the PPU acknowledges the request. HDRQ searches for an unassigned unit with a TBT entry in the queue. If one is found, HDRQ calls HDSL to initiate I/O on the unit.

HDC

Completes request. When a request is completed, HDC performs the following actions.

1. Processes termination status from the PPU.
2. Sets the channel to idle mode.
3. Releases PPUs by calling IH.
4. Updates the current PB, cylinder, headgroup, and sector.
5. Removes TBT from UQT.

<u>Segment</u>	<u>Function</u>
	6. Sets unassigned status for the unit.
	7. Checks conditions for calling HSP.
	8. If threshold is reached or flush process was not complete, requeues the TBT in UQT.
	9. Calls HDSL to select the next TBT for I/O processing.
HACT	Logs activity status for CE file. HACT records the number of sectors transferred on a unit and sets up a CE message in the CEFAP buffer, which contains the number of sectors transferred since the last recorded message. It calls CEM through CALCEM to complete logging of the CE message.
HLOG	Logs 819 disk errors. HLOG retrieves recovered and unrecovered errors recorded in the CM hardware I/O buffer. It reformats the errors to the CE message format and puts them in the CEFAP buffer. CEM is called through CALCEM to complete logging of the CE messages.
HDAY	Calls CEM to send message to job and system dayfile and conditionally abort the job.

PPIO PROCESSING

PPIO stack requests control data transfers directly to and from PP memory. Figure 5-8 shows the logical I/O segments (as shown in figure 5-6) plus the segments used in PPIO processing. The calling PP program builds a stack request and passes its location to the PP resident routines R.READP or R.WRITEP. These routines call the subroutine R.RWP to perform the data transfer. R.RWP returns control to the calling PP program when the stack request is complete.

Although the type of mass storage involved in the transfer is transparent to the calling PP program, a special interface with the 819 stack processor HSP has been defined for PPIO. This interface consists of reserving a CM system circular buffer (SCB) for the PPIO request and then emulating CM FET I/O.

The following segments are used in PPIO processing.

<u>Segment</u>	<u>Function</u>
RWH (PP resident)	Interfaces with HSP and performs data transfers to and from the SCB.
BFP	Executes during 819 stack request initialization. BFP reserves the SCB, linking it to a TBT, and sets the W.RWPPCW in the calling PP's communication area to load and execute the segment RWH.
HPO	Initiates data transfer operation when called by RWH via an M.ICE/EX.PPIO function. HPO determines the type of request and calls HSP to transfer data between the SCB and the PP. If a read operation is being terminated or reissued, HPO exits to HPP.

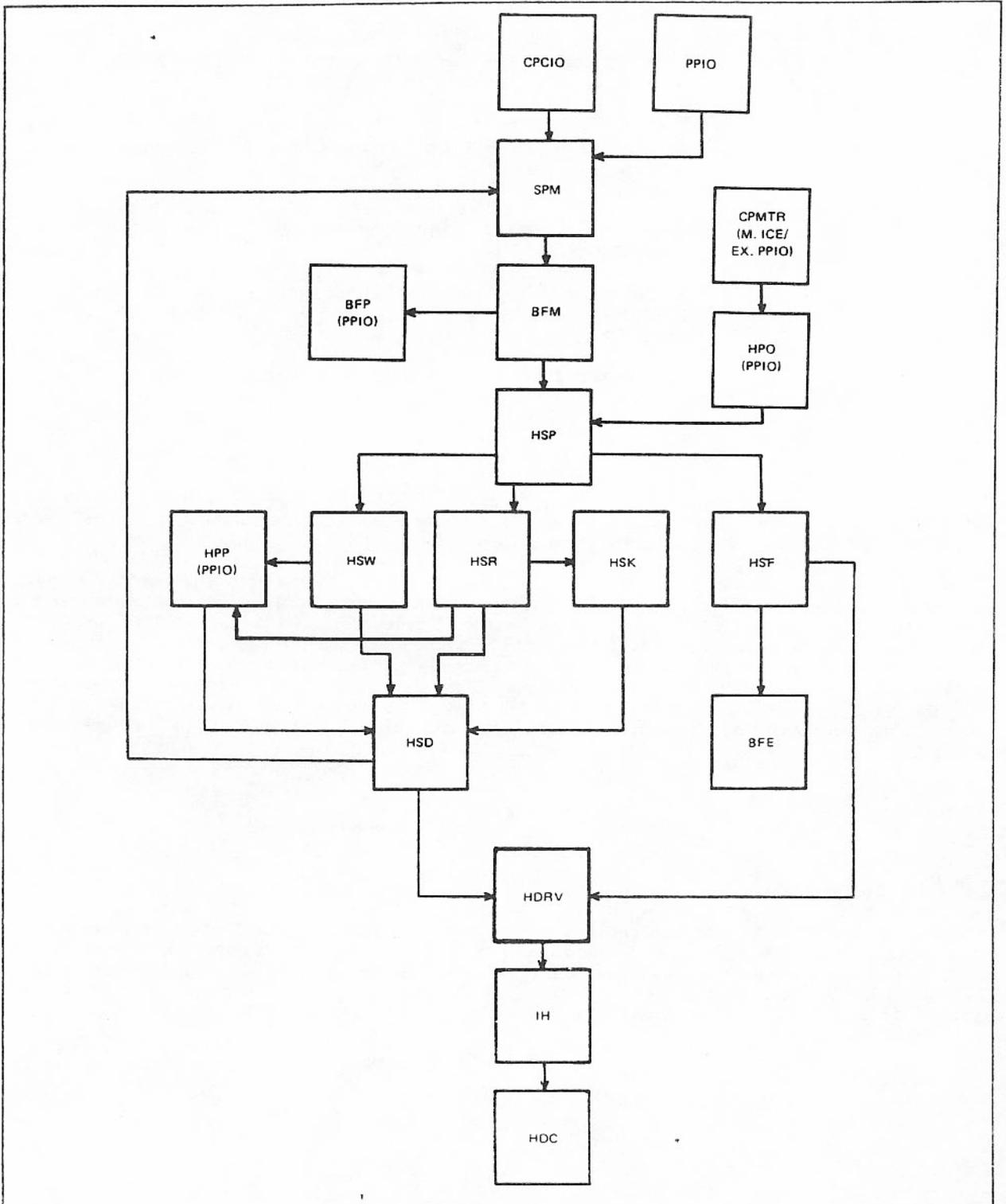


Figure 5-8. PPIO Processing

Segment

Function

HPP

Called by the 819 read and write routines HSR and HSW after a data transfer.

If there is data to be transferred between the SCB and the PP, HPP sets W.RWPPCW to execute RWH in the calling PP and then returns to CPMTR.

If the request must be reissued, HPP sets up W.RWPPCW to cause PP resident to loop in R.RWP waiting for communication to be reestablished.

If the request is to be terminated, HPP sets complete status in W.RWPPCW. R.RWP exits to R.READP or R.WRITEP which returns to the caller.

HPP releases the SCB and calls HSD to terminate or reissue the request.

LCM BUFFER MANAGEMENT

The LCM partition, part of the dynamic paged area, consists of 512-word buffers, each of which can hold an 819 disk sector. From 2 to 24 buffers can be allocated for each file. The contents of the TBT fields FIRST, IN, OUT, and LIMIT are pointers to these buffers.

When a request for a new file is made, BFM must initialize a TBT. If an empty TBT cannot be found, another system page of 512 words is allocated. If there are not enough LCM buffers to allocate area for the new file, BFM must call BFE to evict an inactive file. Selection of a file is based on FNT order. The 819 file evict code keeps track of the position in the FNT where the last file was evicted. The search for the next file to be evicted takes place ahead of this entry in an end-around search of the FNT. When the file is selected, BFE clears the TBT and releases the LCM buffers. If the buffer is empty or a read buffer, the new file is assigned to the released area immediately. If the buffer is a write buffer, the new file request must wait on the recall chain until I/O on the write buffer is complete and the buffer has been evicted.

TABLES

Transfer Buffer Table

The transfer buffer table (TBT) contains information necessary to move data through the LCM buffers. One TBT is allocated for each file and remains until the file is evicted. It contains the addresses of the LCM buffers and the latest stack request for the file. TBTs are allocated until a 1000₈-word page is full. A CM location contains a pointer to the first TBT page. Subsequent TBTs are pointed to by the first word of the page. That word contains a zero if it is the end of the chain. The format of the TBT is given in appendix E.

TBT Address Table

The TBT address table (TAT) contains the addresses of TBTs for all active files. Segment BFM creates the TAT which resides in the paged area of LCM. The TAT contains a one-word-long entry corresponding to each entry in the FNT. When initially assigning a TBT to a file, segment BFM adds the address of the TBT to the appropriate entry in the TAT. Segment BFE clears this address when the TBT is released.

During processing of the monitor flush function, the system saves in the TAT the address of the output register of the PP, issuing the function or the address of the third word of the FNT entry for the file being flushed. The format of the TAT is described in appendix E.

Unit Queue Table

The unit queue table (UQT) is used by HDRV to queue disk requests for each unit. The UQT contains a header and a four-word entry for each unit. It points to the first and current TBT in the unit queue and is also linked to a particular channel table entry (primary/secondary channels). The format of the UQT is given in appendix B.

Channel Table

The channel table (CHT) describes the activity of each data path. There is one entry for each data path. The CHT is used primarily by the interrupt handler. It points to the UQT for the current unit. The format of the CHT is given in appendix B.

CE ERROR FILE

After each disk request, the PPU sends status to the CPU. If the status shows any error, the segment HLOG logs the status information, along with other information it has gathered, in the CE error file. Generally, one error file entry describes an attempt to recovery one error, but if other errors occurred during the recovery procedure, the additional errors are also recorded in the entry. At least 1 bit in the first or second group of flags in the error file is set to describe the type of error. All statuses in the entry reflect the status of the first failure for the error. Figure 5-9 shows the format of the CE error file.

59	53	47	35	29	23	20	11	5	0
24 ₈	Error Code	EST Ordinal		PPU No.	Function	Previous Cylinder	Request Head	Sector	
Flags (Group 1)		Function	Unit No.	Error Messages This Request	Cylinder		Starting Head	Sector	
Flags (Group 2)		Subsystem Status		Controller Status	Retry Count		Expected Head	Sector	
Status				Error Code 3	Error Code 1		Error Code 2		
Error Code 3		Error Code 1		Error Code 2	Error Code 3		Error Code 1		
Error Code 2		Error Code 3					Activity Count Blocks/512		

Figure 5-9. CE Error File

<u>Word</u>	<u>Bits</u>	<u>Description</u>
0	59-54	24 ₈ ; type of error.
	53-48	Error code (in octal).
	<u>Code</u>	<u>Description</u>
	06	Status message.
	17	RMS address.
	20	RMS checkword error.
	71	RMS abnormal error.

<u>Word</u>	<u>Bits</u>	<u>Description</u>
	47-36	EST ordinal.
	35-30	Unused.
	29-24	Number of PPU's which encountered the error.
	23-0	Previous request on this unit.
	23-21	Previous function.
	20-12	Cylinder address.
	11-6	Head group.
	5-0	Starting sector.
1	59-48	Flags (group 1).

<u>Bit Set</u>	<u>Significance</u>
59	Request aborted; error unrecovered.
58	Last sector transferred with unrecovered checkword error.
57	Request completed successfully.
56	No resumption on control channel.
55	No resumption to RF sent to disk.
54	Slave aborted the request.
53	Unit never on cylinder.
52	Unit down.
51	Channel down.
50	CPU request error.
49	Hardware error.
48	Partner PP down.

<u>Word</u>	<u>Bits</u>	<u>Description</u>						
	47-42	Function code.						
		<table border="1"> <thead> <tr> <th><u>Code</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Read.</td> </tr> <tr> <td>1</td> <td>Write.</td> </tr> </tbody> </table>	<u>Code</u>	<u>Description</u>	0	Read.	1	Write.
<u>Code</u>	<u>Description</u>							
0	Read.							
1	Write.							
	41	Verify sector header address on write request.						
	40-36	Physical unit number.						
	35-24	Number of error messages for this request.						
	23-12	Cylinder address.						
	11-6	Starting head group.						
	5-0	Starting sector.						

2 59-48 Flags (group 2).

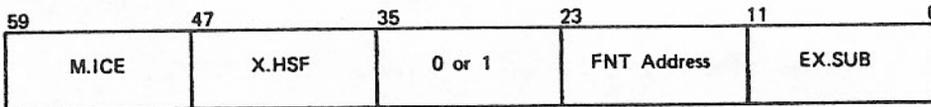
<u>Bit Set</u>	<u>Significance</u>
59	Slave encountered error.
58	Error during read or write of data; refer to controller status for type of error.
57	Not-on-cylinder status occurred without previous position function.
56	Unused.
55	Subsystem busy.
54	Error correction attempted (includes error code statuses).
53	Unit not ready (maximum includes unit fault and interlock statuses).
52	Unit, track, head, or sector in header not in desired position.
51	Cylinder address in the cylinder status is not the desired cylinder.
50	Head address in the head status is not the desired head.
49	Either the unit number in the controller status is not the desired unit or a controller error occurred during a seek.
48	Seek error.

<u>Word</u>	<u>Bits</u>	<u>Description</u>
	47-36	Subsystem status.
	35-24	Controller status if controller error detected.
	23-12	Retry count.
	11-6	Expected head group.
	5-0	Sector address expected.
3	59-48	Contents vary depending on error. <ul style="list-style-type: none"> ● Cylinder status if status is incorrect. ● Head status if status is incorrect. ● Cylinder address recorded on sector if address is incorrect. ● Unit fault status if not ready. ● Error code 1 if error correction was attempted. †
	47-39	Contents vary depending on error. <ul style="list-style-type: none"> ● Head and sector recorded on sector if incorrect. ● Interlock status if not ready. ● Error code 2 if error correction.
	35-24	Error code 3 if error correction.
4	23-0	Error codes 1, 2, and 3 for up to three more channels of error correction (multiple checkword errors). †
5	59-36	
	35-12	Not used.
	11-0	Activity count; number of blocks/1000 since last activity count was reported.

† The controller status tells which channels had checkword errors. The order of error correction for checkword errors on multiple channels is 3, 2, 1, 0. If one channel cannot be corrected, the message does not include any further error code statuses.

819 SUBSYSTEM FLUSH FUNCTION

A PP routine can use the M.ICE function with the subfunction EX.SUB to flush the LCM buffers associated with a file residing on an 819 device. A PP invokes this function by setting up the PP output register as follows:



The FNT address field contains the FNT address of the file to be flushed. The PP sets byte 2 of the output register equal to 0 or 1 with the following implications.

Setting

Significance

- | | |
|---|--|
| 0 | The system does not clear the PP output register until the flush is complete. Thus, it is not necessary for the PP to check for a completion bit. |
| 1 | The system clears the PP output register after segment HSF receives the flush function. Thus, the PP must check bit 0 (completion bit) of the third word of the FNT to determine flush completion. Only PP routine 1S0 issues the flush function with byte 2 equal to 1. |

PERMANENT FILES-SYSTEM INTERFACE

Two pointer words in the CMR pointer area contain pertinent system information about files. Word 6 contains the number of attached permanent file table entries, the starting address of the attached permanent file table, and a byte of flags. Word 7 contains the mounted set table (MST) ordinal of the system set and PF default set; the starting address and number of entries in the dismountable device table, and the starting address and number of entries in the mounted set table.

PERMANENT FILE INTERLOCKS

Several types of interlocks are used to prevent conflicts during the accessing of permanent files on RMS devices:

- **Hardware interlocks.**
- **Set interlocks in the label.**
- **Set interlocks in the MST.**
- **Interlocks in the set member table (SMT).**
- **Permanent file manager (PFM) interlocks in the MST.**
- **Utility interlocks in the MST.**
- **Attached permanent file (APF) interlocks.**

Hardware interlocks are used by RECOVER, TRANSPF, SPM, and PFM to test and set interlocks residing on a device and to prevent any changes to critical tables (SMT, DAM, PFC, and PFD) on the master device, if the set is already mounted on any mainframe. They are set by issuing an interlocked stack request and released by issuing a stack request without interlock. They are retained when interlocked stack requests continue from the same PP or to the same FNT as with the first stack request.

The set interlock in the label is used to prevent device set routines (such as ADS,MNT) on other mainframes from simultaneously updating critical tables (SMT, DAM). When this interlock is set, no other mainframe is allowed to run a MOUNT on the master device until the interlock is cleared. Thus, a program can ensure that no other mainframe accesses a particular set by first checking that no other mainframe has yet mounted the master device and then setting the set interlock in the label. To set this interlock, first the label should be read with hardware interlock, then the set interlock should be tested and set, and finally the label should be rewritten without hardware interlock. To release this interlock, first the label should be read, then the interlock should be tested and cleared if still set, and finally the label should be rewritten.

Set interlocks in the MST are used to prevent two or more device set routines from running on the same mainframe. (This function can also be performed by the set interlock in the label.) The purpose of keeping the central memory set interlock is to allow jobs to swap out while waiting for the set interlock in central memory to clear; thus, the action, if the interlock in central memory is not set, is different from the action required if the interlock on the device is not set. The set interlocks in the MST can be set by reserving the MST channel, setting the interlock, and then releasing the MST channel. The interlocks can be cleared by reading the MST entry, testing, and clearing the interlock if not already cleared.

Interlocks in the SMT are used to interlock a member device, preventing other mainframes from mounting the device so that it can eventually be physically dismounted. They are also used by RELABEL to ensure exclusive access to the device's SMT, DAM, and so on. There are five such interlock bits in the SMT: four correspond to the mounted/unmounted flag on the four possible mainframes; the fifth is a request dismount flag. This last flag, when set, prevents any other mainframe from mounting the device; thus, it can be used to interlock a member which is not yet mounted. These bits can be set by reading the SMT with hardware interlock, setting the request dismount flag, and then rewriting the SMT releasing the hardware interlock. They can be released by reading the SMT with hardware interlock, testing, and clearing the interlock if not already cleared, and then rewriting the SMT.

PFM interlocks in the MST are used to prevent two or more PFM routines from simultaneously accessing the PFD or PFC within a single mainframe. This can also be achieved with interlocked stack requests; however, the central memory interlock allows a PP to enter the event stack to wait for the PFM interlock in the MST to clear. Furthermore, it also prohibits TRANSPF's table transfer mode from running in the same mainframe at the same time.

In a multimainframe environment, permanent file disk tables PFD and PFC are accessed with interlocked stack requests, so as to provide interlocking between mainframes; however, within a single mainframe, the same PFM interlock is used to interlock both the PFD and also the PFC. The PFM interlock is set based on the device set (that is, in the MST) and is always cleared along with the device interlock whenever the PP takes an abnormal exit. When a broken connect occurs during the process of reading or writing the PFD or PFC, new sequences are executed to perform extensive checking and clean-ups for recovery attempts.

For a complete permanent file cycle, the type of access permission currently granted to each mainframe is recorded in the PFC interlock word in the PFC entry. To access an individual permanent file, it must be ensured that there are no access permission conflicts with the other mainframes. For an incomplete permanent file cycle, only exclusive access permission is granted to one mainframe by recording its mainframe ordinal in the PFD entry.

The PFM interlock in the MST is set by reserving the MST channel, setting the interlock, and then releasing the MST channel. It is released by reserving the MST channel, reading the MST entry, testing and clearing the interlock if still set or allowing error exit for the interlock if not still set, and releasing the MST channel.

The utility interlock in the MST is used by TRANSPF in conjunction with the set interlock in the master device label, after testing that no other mainframe has the set mounted and that the permanent file manager activity is quiet on the set. This ensures that the host mainframe has exclusive access to the set while transferring disk tables and also prevents CATALOG and ATTACH functions on the same mainframe. The utility interlock is set by reserving the MST channel, setting the interlock bit, and then releasing the MST channel. The interlock is cleared by reserving the MST channel, clearing the interlock if set or allowing error exit for the interlock if not set, and releasing the MST channel.

The APF interlock is used for interlocking an APF entry, preventing two or more routines from simultaneously accessing or updating the APF entry. The APF interlock is set by reserving the APF channel, setting the APF flag in the APF entry, and then releasing the APF channel; it is cleared by clearing the flag in the APF entry.

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 permanent file catalog (PFC). The PFC was called the RBT catalog (RBTC) and these names can be used interchangeably.

PFD and PFC are of fixed length, and reside on the same mass storage device. A user device set has its own PFD and PFC on the master device of the set. Pointers to these tables are kept in the device label.

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 has 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 RBT chain for use in recovery. Each subdirectory consists of PFD entries.

The PFD is a file whose first allocation unit is recorded in the device label of a master device. This file has two fixed length parts. The first part, called the preamble, is always 512 words long. It contains a complete list of record blocks assigned to the PFD. The second part, called the PFD body, contains the individual entries. The size of the PFD body must always be an integral multiple of pages. Also, this multiple must be a power of 2. The size of the PFD body must be less than 511 pages. The PFD body is initialized to binary zeros.

The size of a PFD entry is always 16 words, 4 entries per PRU.

The PFD body has n ID hash points. These hash points are equally spaced within the body and the space between any two ID hash points is an integral multiple of pages. This multiple is a power of 2. The number (1 to n) of the hash point is the ID hash.

Since space between two ID hash points is a multiple of pages, the start of each of these pages is a PFN hash point.

A subdirectory is a subset of PFD entries which have a common ID hash.

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 PFC, each corresponding to an entry for a cycle. Each cycle entry contains the cycle number, the mainframe ordinal (for incomplete cycles only), four flags, and the PFC pointer. The mainframe ordinal indicates the mainframe currently accessing the cycle, so that in a multiframe environment, exclusive access can be granted. The first flag is not used; the second signifies that the entry is incomplete, since no PFC pointer exists; the third, that the cycle is archived; and the fourth, that the cycle has a parity error.

When cataloging, the user can specify the number for the cycle or permit a cycle number to be generated automatically. Unless the user specifies the cycle when a file is attached, the one with the highest cycle number is attached. 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 ID.

The entry-in-use flag in word 0 of the PFD is set when an entry is built. Detailed formats of the PFC and PFD entry appear in appendix D.

Detailed formats of other tables relating to device sets are included in appendixes B and D of this manual. The set member table (SMT) is a directory of the members of a device set. The device allocation map (DAM) functions like an RBR for a device. The DAM is copied to central memory when the device is mounted, maintained in central memory while the device is being used, and copied back to disk when the device is dismounted. The subdirectory table is used to keep track of the number of permanent files in each subdirectory.

The logical flaw table is a bit map of space reserved for system tables on the device and space unusable because of hardware flaws on the device. The physical flaw table is a list of hardware flaws for a device. PFC entries are always allocated as one or more PRUs. One PFC entry may occupy several PRUs which must be contiguous.

Figure 6-1 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 FNT entry of an attached permanent file relates that entry to the corresponding APF entry. The NOS/BE permanent file management subsystem allows more than one user to have read-only access to the same file at the same time. The number of these users is kept in the count 1 field. This subsystem also allows one user to have write-append access, but not control access, while other users have read-only access to the file. This user has a separate RBT chain. When the write-append user makes a new (longer) RBT chain and uses the extend function to make the new chain permanent, the count 2 field is set to the number of users reading the file with the old chain ($CNT2 = CNT1 - 1$), and the count 1 field is set to 1 to indicate the number of users with the new chain. Users subsequently attaching the file with read-only permission will gain access to the new chain and count 1 is incremented.

After the original write-append user returns the file, one more extend is allowed while others are reading the file with the first or second RBT chain. A new user may attach the file with write-append permission ($RW=1$), create a third RBT chain, and extend the file. Users subsequently attaching the file with read-only permission will be swapped out to the permanent file queue and will not gain access to the file until either count 1 or count 2 is zero. Use of the third RBT chain for this file is not allowed until all users of either the first or second chain have finished.

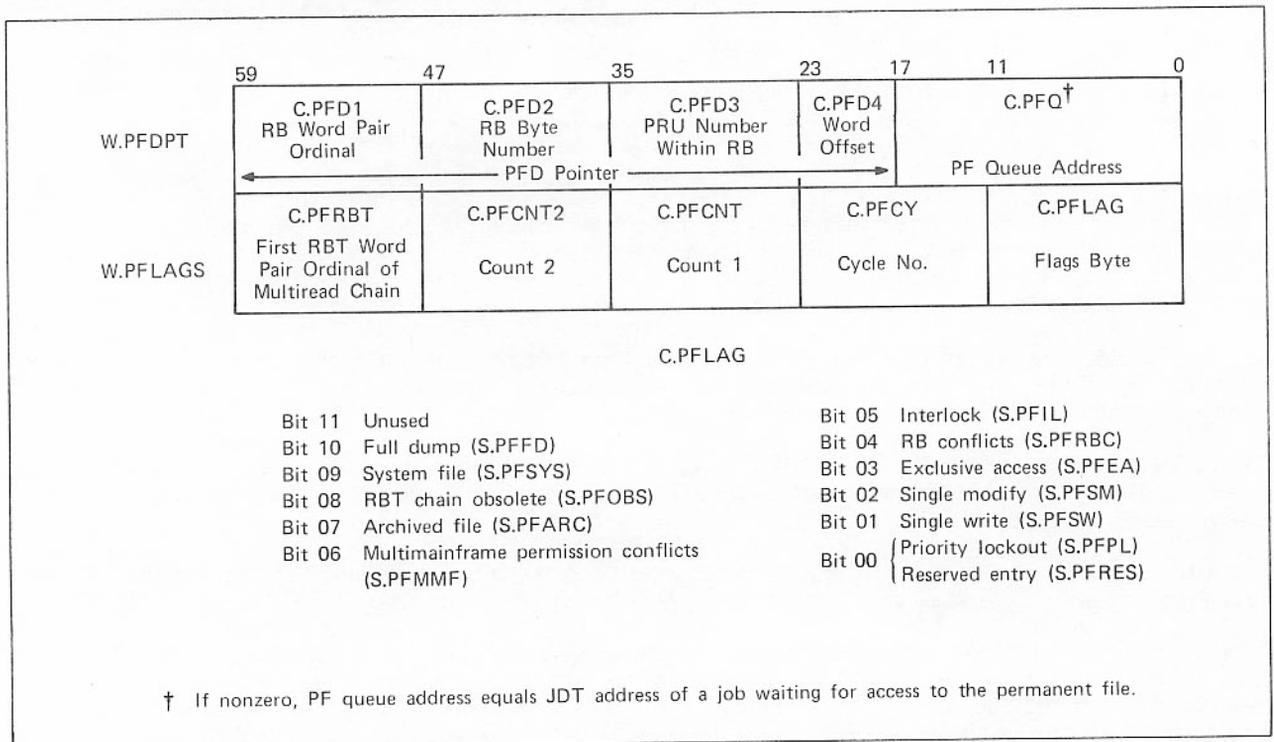


Figure 6-1. APF Table Entry

An incomplete cycle is a file that has a PFD entry but no PFC entry. It can result from either a CATALOG or LOADPF job that does not terminate normally. An incomplete cycle can be attached and purged if the CY parameter is used to specify the cycle and control permission is established.

The PFC, like the PFD, is a file whose first allocation unit is recorded in the device label of a master device. This file has two fixed length parts. The first part, called the preamble, is always 512 words long. It contains a complete list of allocation units assigned to the PFC. The second part, called the PFC body, must always be an integral multiple of pages. This multiple need not be a power of 2. The PFC body is initialized to binary zeros.

PFC entries are always allocated in blocks of 64 words. One PFC entry can occupy several blocks which must be contiguous.

PERMANENT FILE ACCOUNTING

When W.CPFACT in the control point area is nonzero, accounting messages will be sent to both the system and user dayfiles whenever the status of a permanent file changes in the job. These messages are given in the NOS/BE Reference Manual. The ACCOUNT statement will set W.CPFACT nonzero.

PERMANENT FILE UTILITY ROUTINES

Permanent file utility routines provide capabilities of 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. The permanent file utilities are:

<u>Utility</u>	<u>Description</u>
DUMPF	Copies selected permanent files from mass storage to tape, either to clear all permanent files from mass storage devices or to provide periodic backup files for an installation. DUMPF must be on a system library. No files should be attached to a job containing DUMPF operations during DUMPF execution.
LOADPF	Reloads permanent files from the tape created by DUMPF to mass storage. LOADPF allows automatic retrieval of an archived file. The user is unaware of this activity if loading is from tape. When an attempt is made to attach an archived permanent file, the operator can type n.GO, which causes the file to be loaded from the appropriate dump tape. An archived file can be purged without loading the file.
PFLOG	Writes an image of the PFC to tape.
GENLDPF	Reads the PFLOG tape, creates a permanent file directory (PFD) entry for each archived file entry, and calls the routine LPF to recatalog the file. Remaining unarchived file entries are sorted by dump tape VSN in ascending order. GENLDPF creates a LOADPF job for each VSN and routes it to the input queue.
TRANSPF	Transfers permanent files and/or related tables within a set or from one set to another. TRANSPF cannot move system files.
AUDIT	Produces a formatted output file containing statistics on permanent files on a set.

The following permanent file utility function codes (octal) are for system interface between permanent file utilities and PFM routines.

<u>Utility</u>	<u>Function Code</u>
AUDIT	166
DUMPF	172
LOADPF	174
TRANSPF	176

The control statements which call the permanent file utilities are described in the NOS/BE Reference Manual. The remainder of this discussion describes the dump tape formats for DUMPF and PFLOG.

DUMPF UTILITY

DUMPF writes the permanent file dump to a multivolume magnetic tape file in the format shown in figure 6-2.

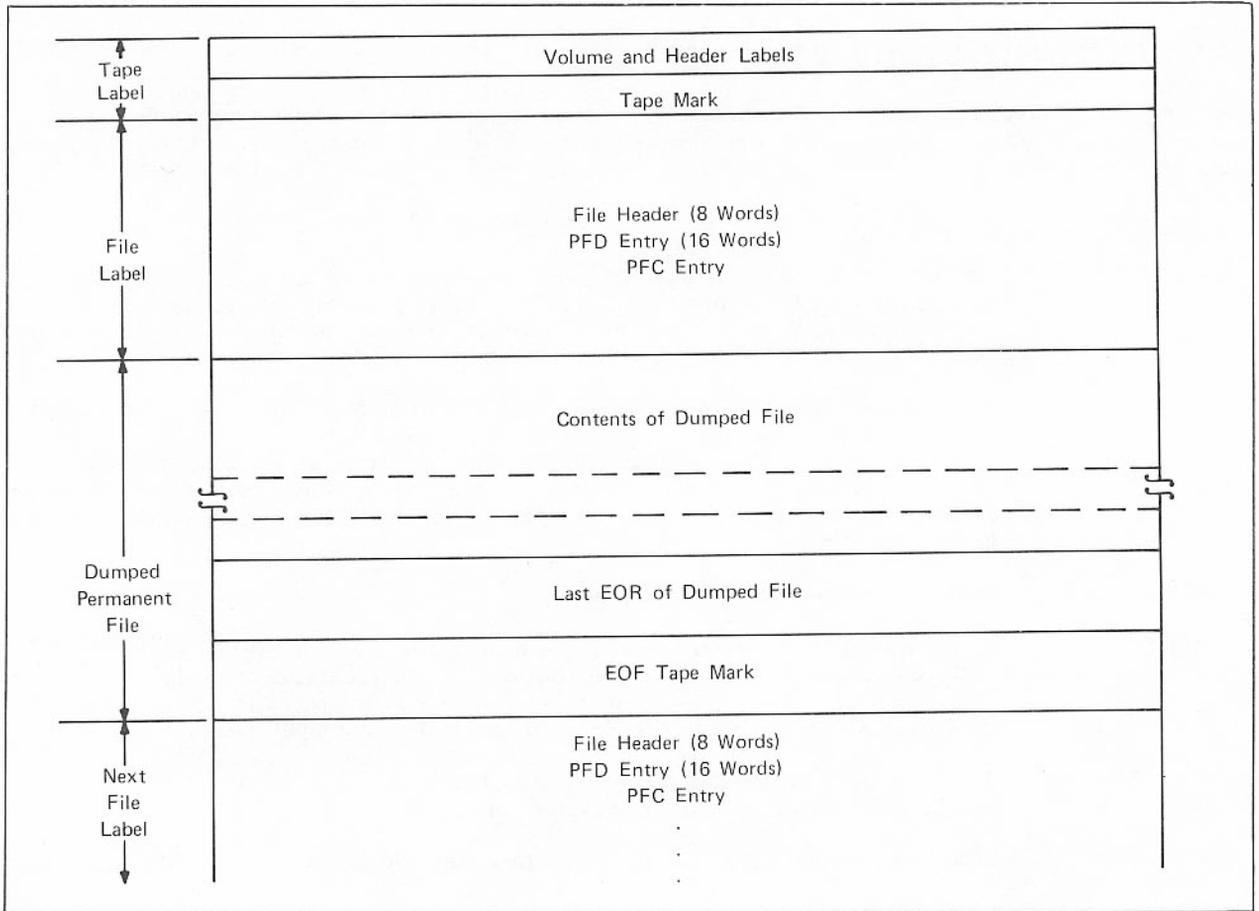


Figure 6-2. Permanent File Dump Tape Format

The first 20 characters of the tape label are as follows:

HDR1DUMPF TAPE- NEW

VOL1DUMPF TAPE-UNIT

Figure 6-3 shows the file header. The fields in the header are in table 6-1.

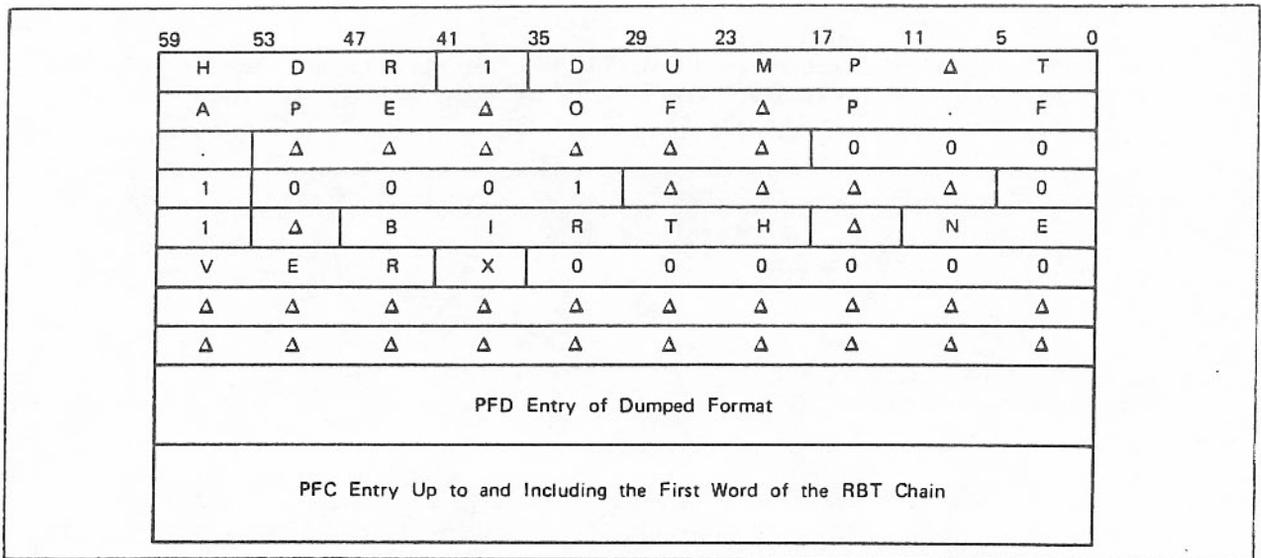


Figure 6-3. File Header

TABLE 6-1. HEADER FIELDS

First Character Position Within Header	Field Length (Characters)	Field Contents	Field Description
1	3	HDR	Label identifier.
4	1	1	Label number.
5	17	DUMP TAPE OF P.F.	File label name.
22	6	Blank	Multifile identification.
28	4	0001	Reel number.
32	4	0001	Multifile position number.
36	4	Blank	Reserved for tape compatibility.
40	2	01	Edition number.
42	1	Blank	Reserved for tape compatibility.
43	5	BIRTH	Creation date.
48	1	Blank	Reserved for tape compatibility.
49	5	NEVER	Expiration date.
54	1	X	Security.
55	6	000000	Block count.
61	20	Blank	Reserved for tape compatibility.

PFLOG UTILITY

PFLOG calls the PP routine OUX with function code 10_8 to create an FNT entry for the PFC. This makes the PFC a local file at the control point where the PFLOG job is running. PFLOG reads the PFC and writes it to a multivolume magnetic tape file in the format shown in figure 6-4.

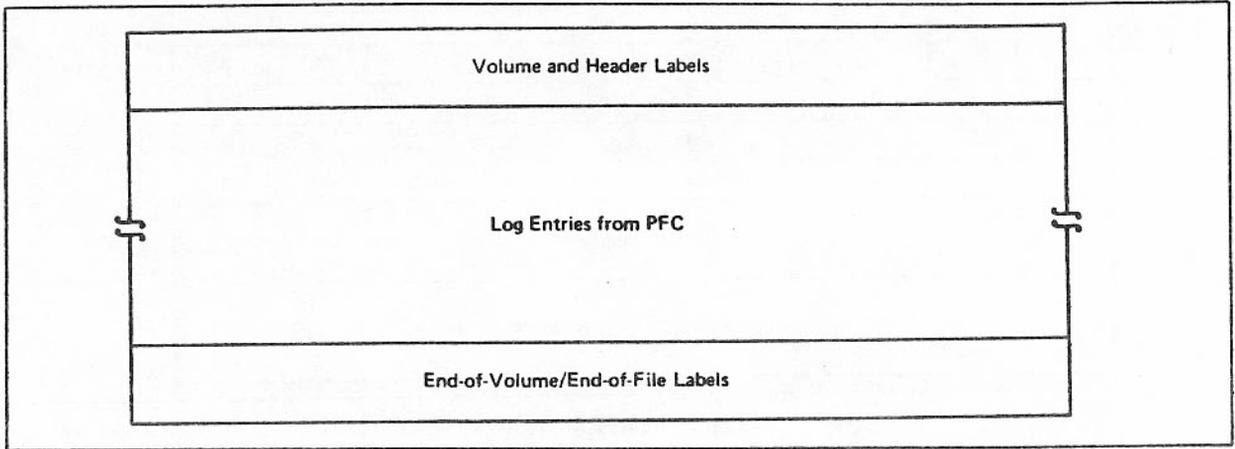
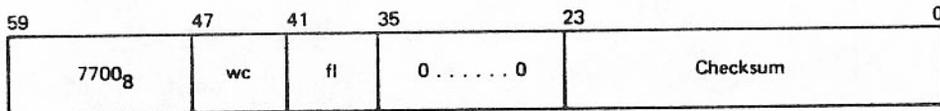


Figure 6-4. PFLOG Dump Tape Format

The first 20 characters of the tape label are as follows:

HDR1 LOG-FILE-TAPE

The log entry for an archived file is identical to the PFC entry for permanent files (refer to PFC entry in appendix D) with the exception of the first word (word 0) and the RBT chain. Instead of the entire RBT chain, the log entry contains only the first RBT word pair; this word pair is the last word of the log entry. The format of word 0 of the log entry for an archived file is as follows:



The log entry for a file which is not archived contains only eight words as follows:

59	47	41	35	23	11	5	0	
7700 _g		wc	fl	0 0		Checksum		0
Unused		Dump Tape VSN				Cycle No.		1
Account Parameter								2
Owner ID						14 _g		3
								4
								5
Permanent File Name								6
								7

The wc and fl fields are the same for each entry.

Field

Significance

wc

Word count of log entry.

fl

Flags.

Bit

Significance

41 If set, indicates the log entry is for an archived file.

40 If set, indicates nine-track tape.

39-38 Tape density, seven-track.

00 556 bpi.

01 200 bpi.

10 800 bpi.

11 Default.

Tape density, nine-track.

00 Default.

01 6250 cpi.

10 800 cpi.

11 1600 cpi.

37 If set, indicates the log tape is in consolidated format.

RMS SET PROCESSING

Device set creation and usage are described in the NOS/BE Reference Manual. The remainder of this section describes additional capabilities of the LABELMS, ADDSET, MOUNT, RELABEL, and RECOVER statements and the FSN program to address public sets.

LABELMS CONTROL STATEMENT

In addition to the parameters described for the LABELMS control statement in the NOS/BE Reference Manual, system-generated jobs at postdeadstart time can also specify the NPR, DS, and EST=est parameters.

<u>Parameter</u>	<u>Description</u>
NPR	Allows the system to use the entire range of cylinders normally usable on the device. If CTI is on the device, the system does not specify the NPR parameter when it generates the LABELMS statement. If NPR is not specified, and if CTI or MSL is on the disk, LABELMS preallocates space to prevent CTI or MSL being overwritten by the system. Refer to RMS Device Capacity Limitations in section 5 for the usable cylinders on each disk device. The NPR parameter has meaning only when the DS parameter is specified.
DS	Deadstart parameter; used only by the operating system to call LABELMS during postdeadstart.
EST=est	EST ordinal of device; used only when the DS parameter is specified.

If DT=AH (819 disk drive) is specified, the system ignores the NPR, DS, mode, and EST=est parameters.

The deadstart routine GENDJ uses the LABELMS utility during an initial deadstart to blank label RMS devices and to determine flaws from factory format information.

ADDSET CONTROL STATEMENT

In addition to the parameters described for the ADDSET control statement in the NOS/BE Reference Manual, system-generated jobs at postdeadstart time can also specify the *Q, DS, and EST=est parameters.

<u>Parameter</u>	<u>Description</u>
*Q	Queue files may reside on the member added to this device set. This parameter is allowed only if the set is the public queue set.
DS	Deadstart parameter.
EST=est	EST ordinal specified at deadstart.

ADDSET will not add a new member to a set if any of the following conditions exist.

- The number of members in the set is already equal to NM.
- Use of nonstandard allocation styles has caused the DAM to be filled, even though NM has not been exceeded.
- The DAM slot, where the new member is to be added, has a corresponding SMT entry that would extend beyond the SMT length. This could occur when the user is replacing a member device with a device that has more DAMs (RBRs).

MOUNT CONTROL STATEMENT

In addition to the parameters described for the MOUNT control statement in the NOS/BE Reference Manual, system jobs can also specify the DS and EST=est parameters.

<u>Parameter</u>	<u>Description</u>
DS	Deadstart flag; used by the operating system at deadstart to mount public devices.
EST=est	EST ordinal used at deadstart.

For the 844-21 disk drive (AY), 844-41 disk drive (AZ), and 885 disk drive (AJ), MNT issues a stack request with the factory data flag set to read the pack serial number and manufacture date. It then issues a CE error file message containing the serial number and date. If a parity error occurs or the factory data does not exist on the pack, the error file entry contains a pack number of *NO S/N* and a date of 000000. The CE error file entry is described in detail in the On-Line Maintenance Software Reference Manual.

MNT processes the serial number and date only during the initial mount of the pack, not when the pack is mounted by subsequent jobs at other control points.

RELABEL CONTROL STATEMENT

RELABEL is used only by deadstart to rewrite the label on RMS devices to add or delete flaws, or to re-create a label that has been destroyed. The format of the RELABEL control statement is

```
RELABEL(VSN=vsname,SN=setname,MP=mp,I=0,L=0,DS,mode,mpmode)
```

<u>Parameter</u>	<u>Description</u>						
VSN=vsname	Volume serial number of the device to be relabeled.						
SN=setname	Name of the device set containing the device.						
MP=mp	Volume serial number of the master device of the device set.						
I=0	No input file is used.						
L=0	No output file is used.						
DS	RELABEL is to be run during deadstart.						
mode	Recording mode (full-track or half-track) of the disk pack to be relabeled.						
	<table><thead><tr><th><u>mode</u></th><th><u>Recording Mode</u></th></tr></thead><tbody><tr><td>FT</td><td>Full-track; read/write contiguous sequential sectors.</td></tr><tr><td>HT</td><td>Half-track; read/write alternate sectors.</td></tr></tbody></table>	<u>mode</u>	<u>Recording Mode</u>	FT	Full-track; read/write contiguous sequential sectors.	HT	Half-track; read/write alternate sectors.
<u>mode</u>	<u>Recording Mode</u>						
FT	Full-track; read/write contiguous sequential sectors.						
HT	Half-track; read/write alternate sectors.						

Refer to the description of mpmode for restrictions on using full-track mode.

<u>Parameter</u>	<u>Description</u>
mpmode	Recording mode of the master pack of the device set.

<u>mpmode</u>	<u>Recording Mode</u>
MF	Full-track; same effect as FT.
MH	Half-track; same effect as HT.

If the master pack of a device set is to be relabeled, it is not necessary to specify mpmode.

NOTE

Full-track recording mode (FT or MF) can be specified only if the system is using 2XPP speed, and if the device to be relabeled is one of the following:

- 844-21 disk pack accessible by a 7154 controller.
- 844-41 disk pack accessible by a 7154 or 7155 controller.
- 885 disk pack accessible by a 7155 controller.

RELABEL run under deadstart obtains a complete new set of flaws from IRCP through CMR, rewrites the PFT and the device label, and rewrites the DAM and LFT for the device on the master device. It detects any conflicts between the new flaws and already existing files and system tables, and advises the operator when such conflicts exist.

RECOVER CONTROL STATEMENT

The RECOVER control statement is used to validate a device set and reconstruct critical tables. In a dual-mainframe system, it can also be used to recover space on a device set when one of the mainframes becomes inoperative.

RECOVER can have up to six continuation cards or lines. The last nonblank character of the card or line to be continued must be a separator; parameter fields (keyword=nnn) cannot be split between cards or lines.

The format of the RECOVER control statement is

RECOVER(SN=setname,V=vsn,L=filename,MF=mfid,T=term,MO=mode)

<u>Parameter</u>	<u>Description</u>
SN=setname	Name of the device set to be processed.
V=vsn	Volume serial number of the master device of the device set.

<u>Parameter</u>	<u>Description</u>
L=filename	Name of the file on which any errors encountered in the validation process are listed; default is file OUTPUT. When MO=1 or 2, this parameter need not be specified and is ignored.
MF=mfid	If MO=1, ID of this mainframe. If MO=4, primary logical ID of the mainframe that has become inoperative. If MO=0 or 5, this parameter need not be specified and is ignored.
T=term	Sequence number assigned to RECOVER by GENDJ for identification when RECOVER runs under the control of deadstart. When MO=0 or 4, this parameter need not be specified and is ignored.
MO=mode	Specifies mode of RECOVER processing.

<u>Mode</u>	<u>Description</u>
0	<p>Default mode. It is called by control statement or, if the device set is shared, by operator command. Mode 0 runs only on an unmounted device set. It performs the following actions.</p> <ul style="list-style-type: none"> ● Performs a complete interlock cleanup. ● Rebuilds the DAM using the LFT and the PFC. ● Cross checks the PFD and the PFC. ● Detects the RB conflicts. ● Resets the available RB counts in the SMT. ● Flags the PFC entries for which RB conflicts have been detected. ● Gives a comprehensive listing of all errors.
1	<p>Mode for first mainframe to deadstart. It is called at deadstart by GENDJ which provides the term parameter. Mode 1 runs only on an unmounted public device set. If the device set is being shared, the mainframe initiating the RECOVER must be the first mainframe in the multimainframe system to be brought up. It performs the following actions.</p> <ul style="list-style-type: none"> ● Performs a complete interlock cleanup. ● Rebuilds the DAM using the LFT and the PFC. ● Detects the RB conflicts. ● Resets the available RB counts in the SMT. ● Flashes a message on the B display when an error is detected; either gives a GO option to the operator (nonfatal errors) or aborts (fatal errors). ● Calls HDS at the end of processing; transmits the term parameter value and the number of RB conflicts.

Parameter

Description

mode

Description

2 Mode for mainframe other than the first to deadstart. It is called at deadstart by GENDJ, which provides the mfid and term parameters. Mode 2 runs on a shared device set. It performs the following actions.

- Performs interlock cleanup for the local mainframe.
- Rebuilds the DAM using the LFT and the PFC if the device set has not been mounted by another mainframe; otherwise, validates the DAM against the LFT and PFC.
- Detects the RB conflicts.
- Flashes a message on the B display when an error is detected; either gives a GO option to the operator (nonfatal errors) or aborts (fatal errors).
- Calls HDS at the end of processing; transmits the term parameter value and the number of RB conflicts.

4 Resource recovery mode. It is called by control statement or, if the device set is shared, by operator command. Mode 4 runs on either mounted or unmounted device sets. It performs the following actions.

- Performs a cleanup of the interlocks left by the inoperative mainframe.
- Rebuilds (if two mainframes) or validates (if more than two mainframes) the DAM using the LFT, the PFC, and the RBRs and RBTs of the host mainframe.
- Detects the RB conflicts.
- Gives a comprehensive listing of all errors.

5 Called by GENDJ to get a full listing of the errors detected during the preceding call to RECOVER in mode 1 or 2. Mode 5 runs on either mounted or unmounted device sets. It performs the following actions.

- Validates the DAM against the LFT and PFC.
- Cross checks the PFD and the PFC.
- Detects the RB conflicts.
- Flags the PFC entries for which RB conflicts have been detected.
- Gives a comprehensive listing of all errors.

RECOVER NOTES AND CAUTIONS

The frequent use of the RECOVER utility in a shared disk environment is essential for the following two reasons.

1. Whenever a level zero or level one deadstart is performed in a shared-disk environment, some disk space is lost unless RECOVER is run before each deadstart.
2. When one mainframe becomes inoperable, it has probably left interlocks set that prevent the other mainframes in the system from accessing certain disks or files.

Improper use of RECOVER can damage a system's files by allowing the mainframes to allocate the same record blocks (RB) to two different files. The result would be crossed RB's.

For public devices a site must strictly adhere to the following two rules.

1. If RECOVER is used to recover disk space on a public set, only the mainframe that is running the RECOVER utility can be active. No other active mainframe can have the set mounted. If the mainframe running RECOVER is functioning properly, RECOVER mode 4 must be called for each failed mainframe to return space and release interlocks. If the mainframe running RECOVER must be deadstarted, the first-mainframe-to-deadstart option must be selected to cause RECOVER mode 1 to execute.
2. All other mainframes sharing the public device sets must be deadstarted with a level zero or level one deadstart. If a level three deadstart was successful, the RBR tables would no longer reflect the rebuilt disk DAM tables. RECOVER cannot be called on a level three deadstart.

ADDRESSING PUBLIC SETS BY SET ATTRIBUTE

The PP program FSN places the name of a specified default set in the SN parameter slot of an FDB and sets the FDB complete bit. The SYSTEM macro can be used to call FSN from an executing program using the following format.

SYSTEM FSN,R,fbaddr,code

FSN	Program name.
R	Recall (mandatory).
fbaddr	Address of FDB into which the correct set name should be placed. A dummy set name should be used when creating the FDB.
code	Device set to find.
	0 System set.
	100g Permanent file default set.
	200g Queue set.
	300g Reserved.

FSN aborts if no set exists unless the code 300 was specified. In that case, the name of a scratch set is returned.

JOB FLOW

Jobs in the system are processed in three sequential, independent stages: input, scheduling, and 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 or from a user terminal through INTERCOM.

A job comes under scheduler control after all system resources specified on the job statement have been assigned to it. When a job is under scheduler control, it has an assigned job descriptor table (JDT) entry and is either in one of the scheduling queues or assigned to a control point. When it is assigned to a control point, the control point status values indicate the job's activity. With the job swapping/rolling features of the operating system, several jobs can be assigned alternately 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 the job is executed as directed by the control statement record of the job's input file.

When the job has terminated, JANUS is assigned to process the OUTPUT, PUNCH, and PUNCHB files produced by the job.

JOB INPUT QUEUE

As each job is read into the computer by JANUS, the tape loader, 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 and INTERCOM call PP overlay 2VJ to translate the job statement. 2VJ scans the statement, 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. TLOAD, the tape loader, and other CP programs call DSP to enter a job into the input queue.

The PP program 1IB scans the job input queue, looking for jobs eligible for execution scheduling. To be eligible, a job must have all resources requested on the job statement, except for central memory. A job remains in the input queue if it is dependent on a job which has not run, if it is waiting for operator tape staging, if the job is not to be brought to a control point, or if the amount of extended memory it requested is not available.

Tape Job Scheduling

All incoming jobs requiring a specific number of seven-/nine-track tapes (as defined by parameters MT, NT, HD, PE, and GE on the job statement) are entered in the preinput queue, a subset of the input queue. The operator communicates with the preinput queue through console entries and the prescheduling (P) display. Each time the operator requests a P display, the highest priority tape jobs in the queue are displayed. A job requiring tapes is not placed in the normal job input queue until the operator releases it with a command. Once released, the job is considered for assignment to a control point and execution. It no longer appears in the P display. Tape drive scheduling is described in section 4.

Loading Jobs from Tape

The CP routine TLOAD is used to enter jobs into the input queue via magnetic tape. TLOAD is called to a control point by DSD when the operator enters the following command.

n.X TLOAD,x,y.

- n Number of the control point to be used.
- x Optional parameter specifying the number of the file group to be loaded (decimal digit; numbering begins with 1, not 0).
- y Optional parameter specifying the number of the job in file x to be loaded (decimal digit; numbering begins with 1, not 0).

A file group is a collection of jobs on tape, terminated by a double end-of-file.

A standard circular buffer technique is used to process the job files. The first statement of the first record in each file is presumed to be a job statement, 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.

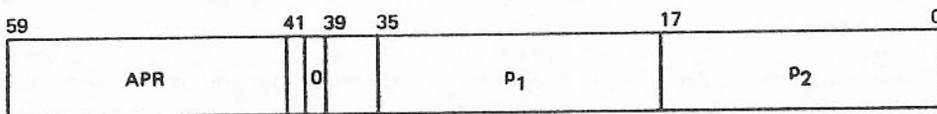
The number of control statements that can appear in a given job is not limited. If the user fails to place an end-of-record (7/8/9) statement behind the control statement record of a job, the omission is not detected by TLOAD, JANUS, or INTERCOM until the job runs.

Sequencer Jobs

Jobs running under the control of the sequencer remain in the input queue with a locked FNT. When the clock interval for the job has expired, the FNT is unlocked and the job becomes eligible for execution under the same constraints as other jobs in the input queue.

When the job completes execution, it returns to the input queue as a locked FNT until its clock interval expires again.

The sequencer is called to a control point using the following RA+1 request format.



Bit 40		Auto recall; must be zero.	
<u>P1</u>	<u>P2</u>	<u>Effect</u>	<u>Called by</u>
0	0	Place sequencer program (APR) in delay request stack.	System job
0	1	Turn sequencer on.	Operator
0	2	Turn sequencer off.	Operator
1	xxxxnn	Add job to sequencer table at ordinal nn, with interval xxxx minutes.	Control statement

<u>P1</u>	<u>P2</u>	<u>Effect</u>	<u>Called by</u>
2	nn	Run job at ordinal nn.	Operator
3	nn	Drop job at ordinal nn in sequencer table.	Operator or Control Statement
4	xxxxnn	Set interval xxxx minutes for job at ordinal nn in sequencer table.	Operator
5	addr	Write real-time clock value in RA+addr.	System job
10	addr	Write RA, RA extended memory, and FL extended memory in RA+addr.	System job
11		Evict OUTPUT file for the job.	Control statement

Range of p₂ parameters (in octal):

0 < nn < L.SEQ/2

0 < xxxx < 3777₈

addr < RA+FL

For p₁ parameters (functions) 1, 3, and 11, APR can be called from a control statement as follows:

APR(p₁,p₂)

Functions 5, 10, and 0,0 can be called only by jobs in the system libraries using the SYSTEM macro as follows:

SYSTEM APR,,p₂,p₁

All other functions can be called only from the operator's console.

JANUS

The unit record I/O package, JANUS, consists of the following PP programs (refer to figure 7-1 for JANUS interfaces).

<u>Program</u>	<u>Description</u>
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 commands 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, or find proper spacing code array.
1PL	Dummy output file driver for plotter.
1FM	Dummy output file driver for film/hardcopy devices.

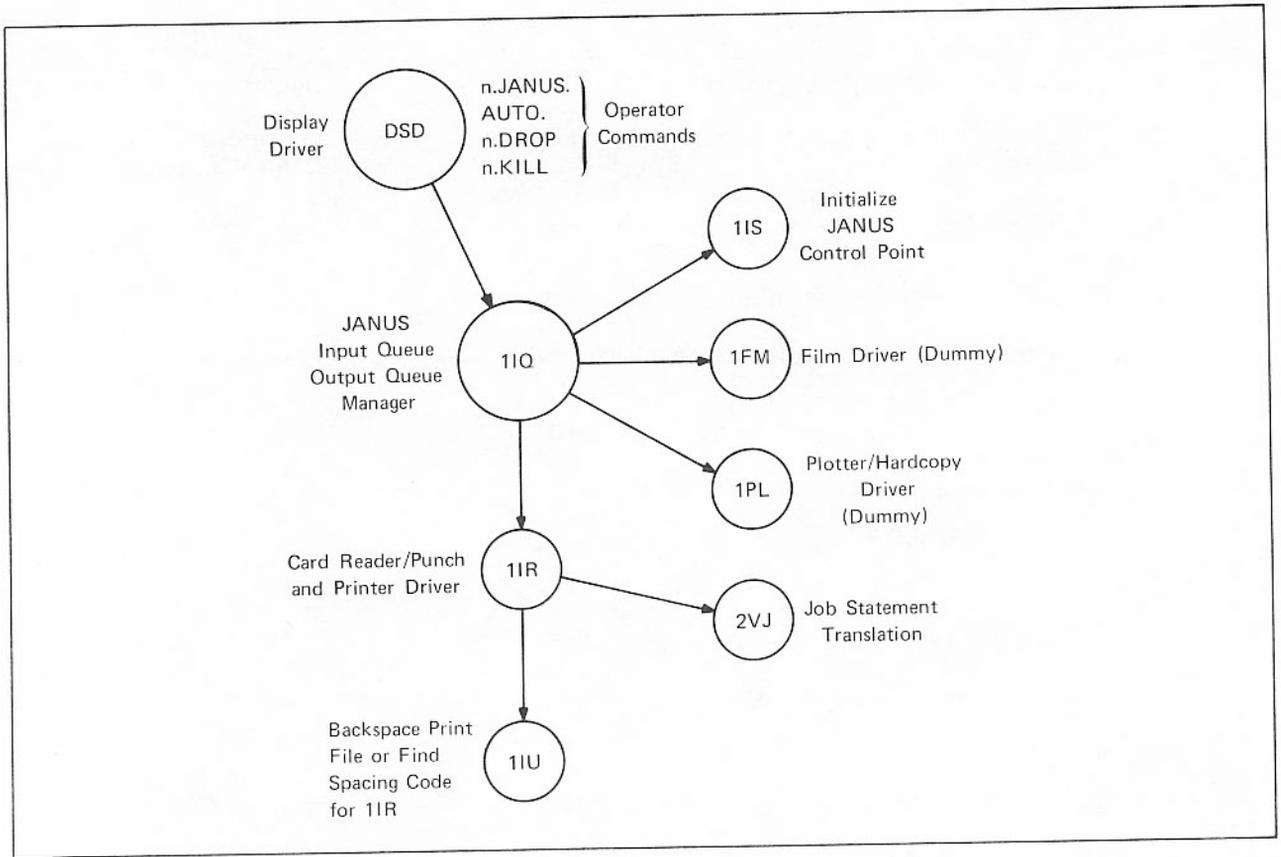


Figure 7-1. JANUS Interfaces

INTEGRATED SCHEDULER

The job scheduling scheme is made up of the following segments (refer to figure 7-2 for scheduler interfaces).

<u>Segment</u>	<u>Description</u>
Scheduler (MMGR)	CPU program operating at control point N.CP+1 in 300 to 1000 millisecond cyclical intervals. It allocates control points and central memory to jobs and determines which jobs should be swapped or rolled in and/or out.
IIB	Brings jobs from input queue to CM queue; builds JDT entries for jobs.
IRN	PP program operating at control point 0 at about 1-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 (ISI) or out (ISO) and to initiate new INTERCOM jobs.

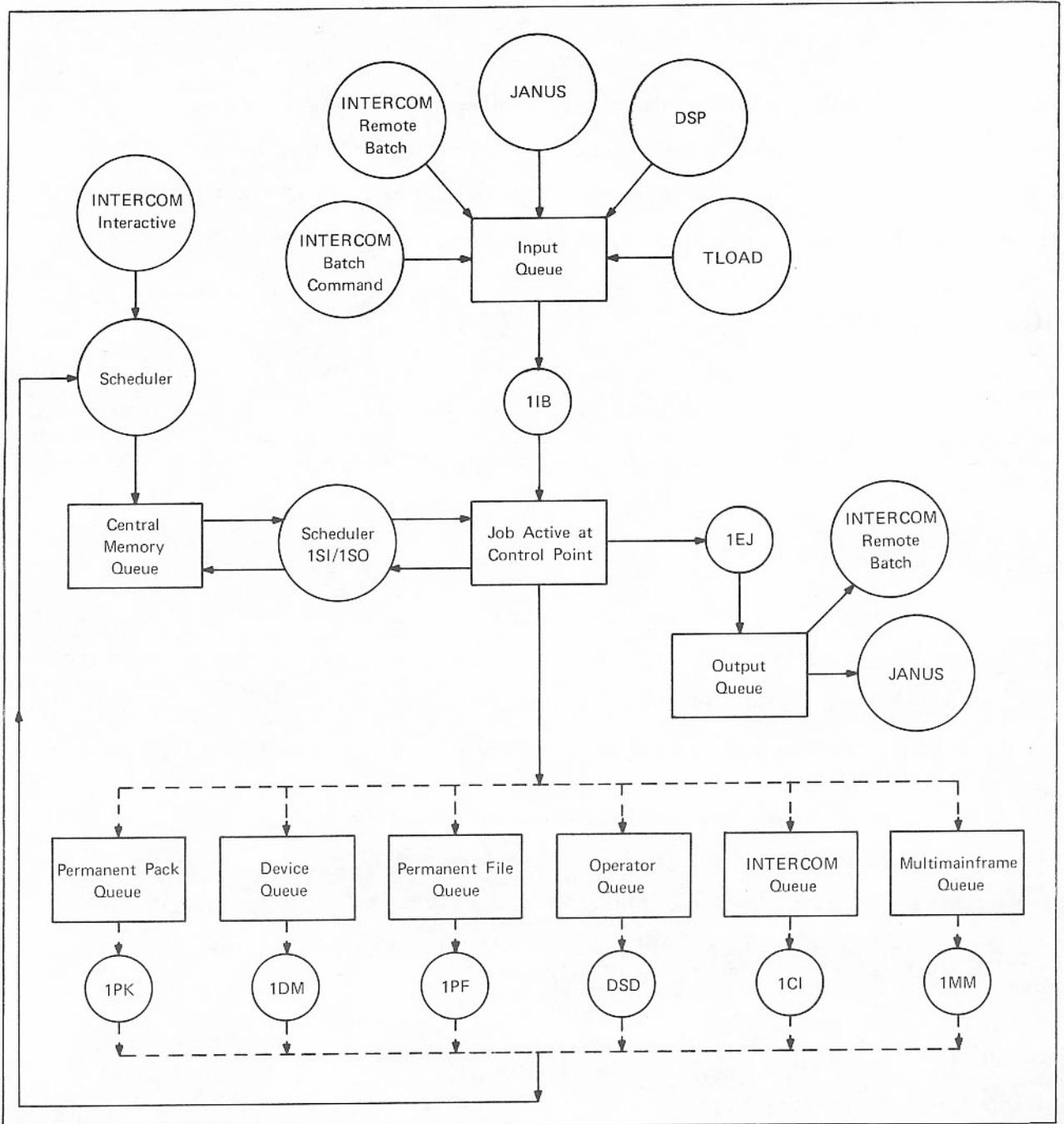


Figure 7-2. Integrated Scheduler Interfaces

<u>Segment</u>	<u>Description</u>
IDM	PP program which processes job entries in the device queue.
IPF	PP program which processes job entries in the permanent file queue.
ICI	PP program which processes job entries in the INTERCOM queue.
IQP	PP program which processes multiuser job entries in the INTERCOM queue.

Figure 7-3 shows the scheduler request stack in the system exchange package area.

JOB SCHEDULING

The 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. A system job such as JANUS cannot be swapped or rolled.

ROLLING

When a job is rolled out, its entire field length is written to an allocatable file, and all of its central 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 resumes use of its control point.

A job is rolled out instead of swapped out under the following conditions.

- Job has direct-access extended memory assigned and IP.ECSW is zero (user extended memory swapping disabled).
- Job has nonallocatable equipment assigned (magnetic tape unit).
- Job is in device queue (waiting for magnetic tape assignment).
- Job is in operator action queue (waiting for n.GO or n.DROP).
- Job field length plus its required positive field length for swap-out exceeds IP.MFL (maximum field length a job can request) plus IP.POSFL or 131K.

SWAPPING

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. The control point, CM, and extended memory are released and made available for reassignment. The JDT entry is placed in the appropriate queue; it contains a pointer to the swap file containing the job. The job, when swapped in, can be assigned a different control point and a different area in CM and extended memory.

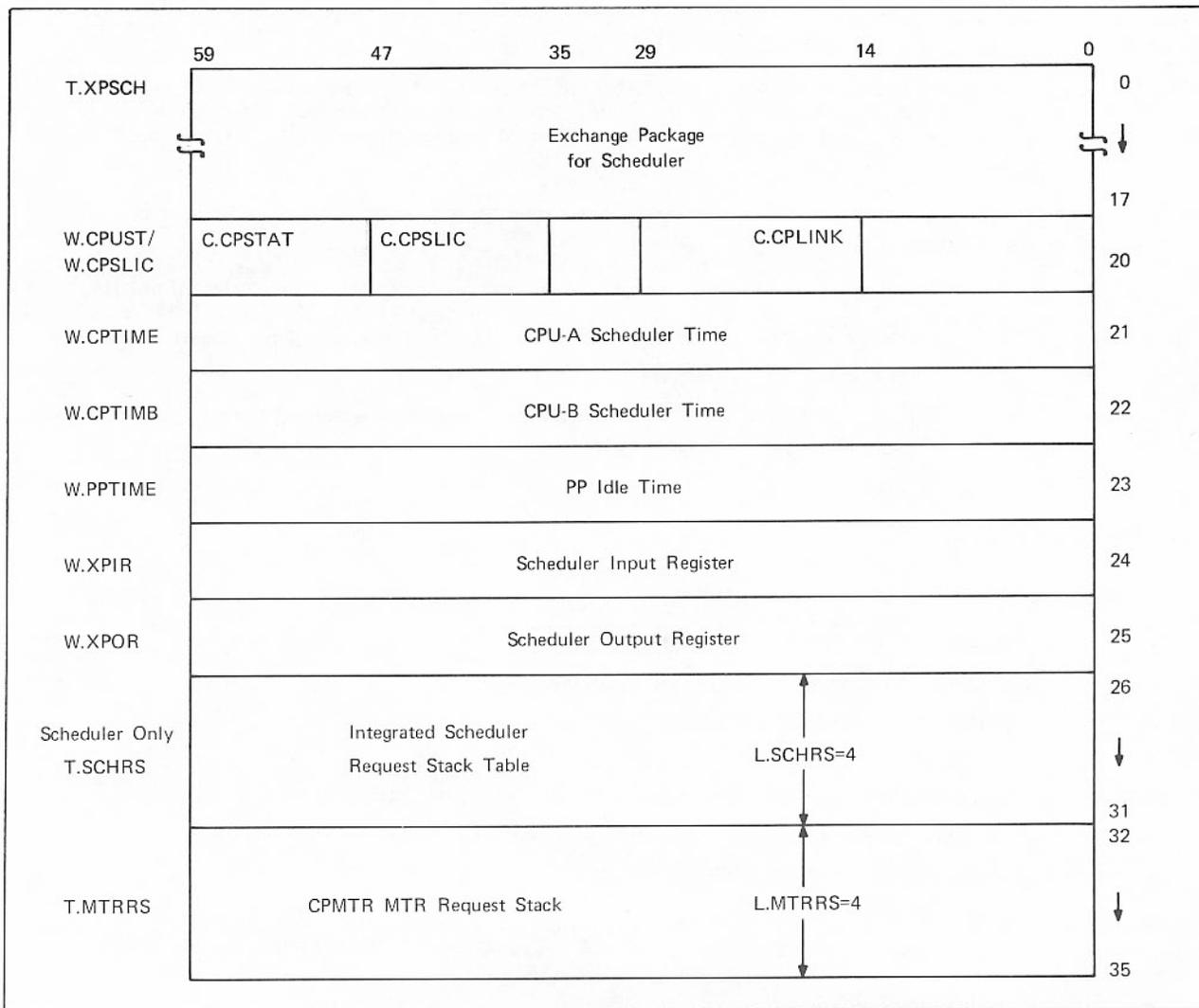


Figure 7-3. Scheduler Request Stack in System Exchange Package Area

A job is swapped out under the following conditions.

- Job is waiting for CM or extended memory.
- Job is in permanent file queue (waiting for access to a permanent file).
- Job is waiting for permanent pack.
- Job is waiting for INTERCOM or graphics terminal I/O (job is in INTERCOM queue).
- Job is waiting for multimainframe action (via station control point).
- Job is connected to a system control point, and SCP has requested job swapout with SFCALL SF.SWPO request.

The scheduler can be initiated by a PP program using an M.SCH request. The scheduler is called in the following cases.

- When a PP program has a job to be added to the CM queue, the scheduler determines if the job has a priority high enough to cause any other jobs 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 CM queue.
- When the field length of a job executing at a control point is reduced, the scheduler is called to search the CM queue for a new job to be scheduled.
- The scheduler is called periodically to determine if any job assigned to a control point has used up its quantum time period for CP and/or PP, and if any jobs in the CM queue have been aged enough to cause a swap-out. A job is swapped out during its quantum time period only if a job with a higher quantum priority has entered the CM queue.
- When the job requests CM or EM that is not available, the scheduler determines whether to swap or roll out the job, or other jobs with less priority.

JOB CONTROL AREA

Jobs entering the system for execution are placed into one of seven classes.

- Batch jobs using no nonallocatable resources.
- Batch jobs using one or more nonallocatable resources.
- INTERCOM (interactive) jobs.
- Multiuser jobs.
- Express jobs (for which operator has entered DROP, KILL, or RERUN).
- Graphics jobs (jobs interfacing with graphics terminals under INTERCOM 4).
- Batch jobs using direct access extended memory.

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 queue priority.
- Maximum queue priority.
- Aging rate (time) for incrementing job priority.
- Quantum priority value.
- Quantum time length value.
- Pointer of 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 a display 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 format of the job control area appears in appendix B.

Scheduling parameters are further described in the NOS/BE Installation Handbook.

JOB DESCRIPTOR TABLE (JDT)

Each job to be scheduled for execution has a 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 scheduling queues. Scheduling queues are subsets of the job descriptor table.

Each JDT in the CM queue is linked into one of the seven 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 each 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 device set 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 that it is in one of the following job scheduling queues.

- Central memory queue.
- Device queue.
- Permanent file queue.
- Device set queue.
- Operator action queue.
- INTERCOM queue.
- Multimainframe queue.

JOB SCHEDULING QUEUES

Central Memory Queues

The CM queue consists of rolled-out and swapped-out jobs waiting for CM, extended memory, and/or control point assignment.

The PP routine IIB scans the job input queue for jobs to be assigned to a control point. When such a job is found, IIB creates the JDT entry and requests that the scheduler put the entry in the CM queue. Jobs in scheduling queues, such as the permanent file queue, for which requested action has been completed, also are eligible to re-enter the CM queue.

When a system control point program issues a SFCALL SF.SWPO function for a user control point (UCP), the UCP is swapped out, if possible, and is held in the CM queue until swapout encouraged flags are clear in the jobs entry in the inter-job connection table.

Device Queue

If a job executing at a control point requests a tape that must be mounted, the REQ routine puts the JDT into the device queue and calls a swapper to roll out the job. The PP program ITS is responsible for detecting when the requested device is ready and for calling PP program IDM to put the JDT entry back into the CM queue. When the job is rolled back in, REQ is 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 appears on the B display.

Permanent File Queue

When a job issues an ATTACH request for a permanent file, it may be denied access to the file because of permission conflicts with jobs already having the file attached on the same or any sharing mainframes. When this occurs, PFA calls IPF to link the job's JDT entry to the APF entry of the file and swap the job out. The permissions requested by this job are stored by IPF in the job's JDT entry. If exclusive access or control permission are requested, IPF sets the priority lockout bit (S.PFPL) in the APF entry and in the PFC entry. This causes queuing of all subsequent attaches that do not require exclusive access or control permission.

Whenever a permanent file is returned, IPC checks for jobs queued on the file. If any exist, IPC calls IPF to swap in as many as possible. IPF checks the priority lockout bit and, if set in either the APF or PFC entries, selects the first job in the queue requesting control permission or exclusive access. If no other jobs in the queue require exclusive access or control permission, the priority lockout bit is cleared in the APF entry and in the PFC interlock byte for this mainframe in the PFC entry.

If the priority lockout bit is clear in both the APF entry and the PFC interlock byte of all sharing mainframes, IPF scans the entire queue, selecting as many candidates for swap-in as possible. These candidates cannot conflict with each other nor with any jobs having the file attached on this or any sharing mainframe.

If PFA detects permission conflicts with jobs having the file attached on other mainframes, it calls IPF to queue the job on the APF entry and to set bit S.PFMMF in the APF entry. IRN scans the APF table periodically, looking for entries having bit S.PFMMF set and calling IPF when such an entry is found. IPF then reads the PFC entry for any file having the bit set. If IPF determines that any jobs can be swapped in, it does so, clearing S.PFMMF if no jobs remaining in the queue conflict with jobs on other mainframes.

Permanent Pack Queue

If a job attempts to mount a member of a device set and the device is not on line, the job is swapped out or rolled out. Operator-initiated jobs abort. When the device being requested is on and free, all jobs waiting for the pack are sent to the CM queue.

Routine IPK is responsible for processing this queue. If the request for the device is generated by an I/O request, the I/O function continues from the point of interruption.

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 appears 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 that the job is to be put into some other queue. For example, REQ sets the pause bit and then requests 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 is rolled in and resumes execution at its control point.

INTERCOM Queue

Jobs waiting for input from an interactive INTERCOM or graphics 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 assigned a job to a control point, the job advancing routine 1AJ is called into a PP. This section describes how the 1AJ routine processes the control statements in a batch origin job.

1AJ reads one PRU (100g CM words) from the control statement record of the job's input file into the control statement buffer of the control point area (locations W.CPAF to W.CPCAL). 1AJ stores the input file's FST entry for the next PRU of the control statement record in word W.CPFST of the control point area. The source file from which the control statements are read can be changed from the job's input file to another file using CYBER Control Language (CCL). For further information on specifying a different control statement file, refer to Job Control Statement Source File in this section.

The job statement of a batch origin job is processed by routine 1TJ and is skipped by 1AJ. 1AJ reads the next statement from the control point area buffer and sets an index to the buffer in word W.CPCC. The statement just read is processed as described in Control Statement Processing in this section (the sequence of operations is shown in figure 7-4). After that statement has been processed, 1AJ is reloaded and, using the buffer index in W.CPCC, repeats the procedure. When all statements in the buffer have been read and processed, 1AJ reads another PRU from the input file's control statement record into the buffer and resets W.CPFST to the next PRU. The 1AJ routine continues either until all statements in the input file's control statement record have been processed, the job is terminated because of an error, or an EXIT statement is encountered.

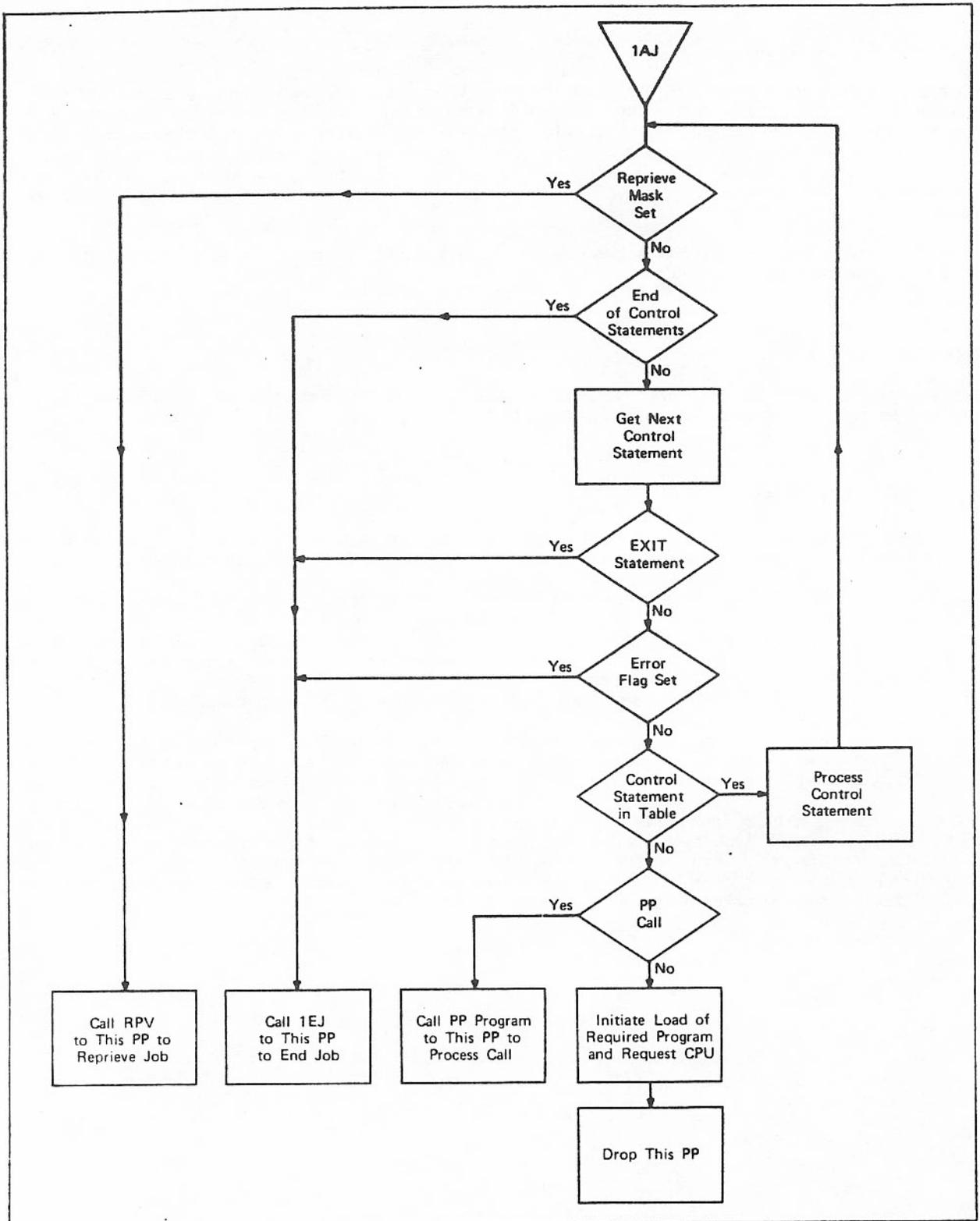


Figure 7-4. Control Statement Processing Flowchart

CONTROL STATEMENT PROCESSING

When IAJ processes a control statement, it performs the following sequence of operations.

1. IAJ compares the verb with a list of verb names (kept internally by IAJ) that are treated as special cases. If there is a match, IAJ completes the processing of the statement and continues with the next statement in the buffer.
2. If a match is not found, IAJ attempts to find a local file of the same name as the verb. If a file is found, the statement is issued to the dayfile, copied to RA+70g (RA.CCD) through RA+77g, and the parameters are separated and stored in RA+2 (RA.ARG) through RA+52g. Then the loader is called to load the file.
3. If the verb is not a file name, IAJ looks for a PP routine that can be called by a control statement and that has the same name as the verb. If such a PP routine is found, the statement is issued to the dayfile, and the parameters are separated and stored in RA.ARG. For all routines except the routine VSN, the first two parameters are converted to binary and stored in bits 35 through 0 of the input register. Then the PP routine is loaded on top of IAJ.
4. If the verb is not a PP routine name, IAJ searches in order the global library (as long as the libraries are resident in CM) for an entry point that can be called by a control statement and that has the same name as the verb. If such an entry point is found and the program is an absolute overlay, IAJ separates and stores the statement, loads the program at RA+100 (RA.ORG), and starts it.
5. If an entry point is not found or the program is not an absolute overlay, IAJ loads the system loader, which completes processing of the control statement.

The following special case names are recognized and processed by IAJ.

ACCOUNT

Routine ACCOUNT is loaded from the system library, and execution is initiated. IAJ does not issue a dayfile message.

COMMENT

The control statement, without the keyword COMMENT, is issued as a dayfile message. If the message is longer than 40 characters, it is issued in 2 parts.

MODE

The exit mode is set as specified on the MODE statement. If the mode value is greater than 7, a control statement error message is issued.

LIMIT

This statement 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 statement, an installation-defined value (IP.SMS) 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 the corresponding bit to be toggled in the C.CPSSW byte of word W.SSW in the control point area.

RFL,f1 or RFL,CM=f1

The new CM field length value (f1) is set in byte C.CPNFL of word W.CPCC, and the CM reduce bit in W.CPLDR1 is cleared. The new field length must not exceed the field length given on the job statement or the current value in C.MFL in P.MFL.

RFL,EC=fle

The new extended memory field length (fle) is requested immediately, and the extended memory reduce bit in W.CPLDR1 is cleared. The new extended memory field length must not exceed the field length given on the job statement or the current value of C.MFLE in P.MFL.

Permanent file control statements (CATALOG, ATTACH, EXTEND, RENAME, PURGE, and so on)

Each permanent file control statement is copied into the job field length, starting at RA+70. No dayfile message for the statements is issued, since the password must be removed from the statement for security. CM program PFCCP is loaded to process the statement.

EXIT

The exit flag bit is in byte C.CPFP of word W.CPFP in the control point area. If a control statement 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 statement contains a parameter, the exit flag bit is set when the error flag in byte C.CPEF of word W.CPEF on the control point area is set. If the EXIT statement does not contain a parameter, the exit flag bit is set only when the error flag is set and the abort bit is not set. PP routine 1EJ is called to do end-of-job processing.

SETNAME

This statement sets the control point default set either to the name specified or, if no name is specified, to the system default set.

SUMMARY

PP overlay 6BM is loaded to issue job statistics to the dayfile.

LOGIN

Routine LOGIN is loaded and execution is initiated. 1AJ does not issue a dayfile message.

MACHINE

Stack purging is set according to the PB option on the control statement.

JOB CONTROL STATEMENT SOURCE FILE

Control statements for a job are usually read from the first record of the job's input file. However, CCL can change the source file so that control statements (usually CCL procedure files) can be read from a different file.

The ACCSF macro retrieves the name and position of the current control statement file. If a different control statement is used, the current file can be restored later.

The format of the macro is

ACCSF **lfn,save**

- lfn** Address of word to receive name of the current control statement file.
- save** Address of word to receive current position of the control statement file.

The ENCSF macro activates a different control statement file at the position specified by the calling program.

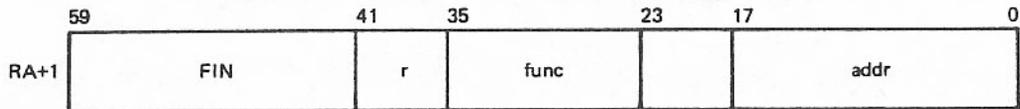
The format of the macro is

ENCSF **lfn,restore**

- lfn** Address of word containing the name of the new control statement file.
- restore** Address of word containing new position of control statement file.

The ACCSF and ENCSF macros are defined in the common deck ACTCOM.

Execution of either macro results in the following RA+1 request for PP routine FIN.



r Recall; set to 20g by macros.

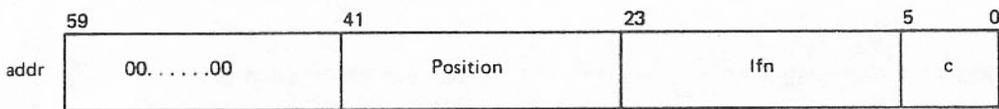
func Function code.

2 ACCSF

3 ENCSF

addr Address of a pointer word generated by ACCSF or ENCSF. The pointer word contains the address of the words containing lfn and the position of the control statement file.

The pointer word addr has the following format.

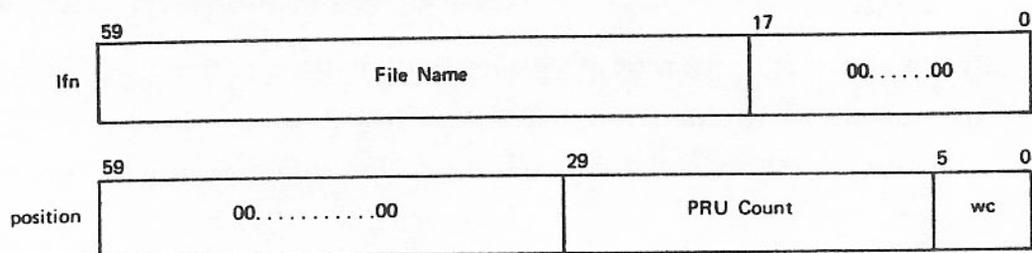


position For ACCSF, address of word to receive current control statement file position. For ENCSF, address of word containing new control statement file position.

lfn For ACCSF, address of word to receive name of current control statement file. For ENCSF, address of word containing name of new control statement file.

c Completion flag; must be zero before macro execution. Set to one when function completes.

The words containing lfn and control statement file position have the following format.



file name A legal file name, one to seven characters. ACCSF sets file name to zero if the current control statement file is invalid or missing.

Both ACCSF and ENCSF ensure that the next control statement is null (end-of-job) if the file name is invalid or missing.

PRU count Position of control statement file in PRUs counting from the beginning of information (BOI=1).

wc Word count; position of word within current PRU (0-77g).

JOB TERMINATION

NORMAL TERMINATION

Normal job termination occurs when the error flag value is zero and all control statements for the job have been processed (an end-of-record mark for the control statement record on the job input file has been read) or when an EXIT or EXIT,S statement has been encountered. The end-of-job processor, 1EJ, is called by 1AJ to terminate the job by performing the following steps.

1. Dispose of all files associated with the job as follows:
 - a. Drop local files on system devices and all equipment or storage associated with them.
 - b. Release nonzero disposition files to control point 0 for further processing of output files. Scratch files are zero disposition files and are dropped.
 - c. Update the TAPES table; rewind and unload tapes as appropriate.
 - d. When tape scheduling is used, return reserved tapes to the system pool.
 - e. Call overlay 1PC for disposal of permanent files and local files on user device sets. 1PC calls DSM if private sets are associated with the job.
2. Transfer the job dayfile to the OUTPUT file associated with the job.
3. Release storage for the job and set the job name to NEXT. The control point clear bit will be honored by the scheduler when it clears the control point area.

The following sections describe the disposal of specific files.

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 deletes the FNT/FST entry for the file. The file still exists on a mass storage device. The attached permanent file (APF) entry for the file is updated or deleted, as appropriate. A purged file is no longer permanent and is processed like any other mass storage file.

Local Files

If the file is on a nonallocatable 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. 1PC disposes of local files residing on private device sets.

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 control point 0. Then the disk space for the job input file is evicted and its FNT/FST is cleared.

Output File

After the dayfile has been copied to the job output file, output file FNT is modified so that the file name is replaced by the job name, and the file is assigned to control point 0 and rewound. The file is then ready for processing by JANUS.

If the output file is on a nonallocatable device or pack, the equipment is dropped and the FNT is zeroed because JANUS cannot process files on nonallocatable devices.

If the file is on an allocatable device, the FNT is rewritten so that the file name is replaced by the job name, and the file is assigned to control point 0 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.

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 and its FST is cleared.

ABNORMAL TERMINATION

Abnormal termination of a job occurs when the error flag value is nonzero. If the error flag value is other than KILL (value 7) or RERUN (value 10), 1EJ performs the following steps.

1. Attempts to flush each local file (that is, to send unwritten data from the file's CM buffer to mass storage) that has a disposition code other than scratch (print, punch, and so on), or has both the disposition code of scratch and the file flush flag set in its FET.
2. Dumps the contents of the exchange package and the contents of memory.
3. Performs normal termination processing as described in the previous section.

The dump includes 100g words before the error stop through 100g words after the error stop (P-100g to P+100g). It is written to the output file immediately preceding the job's dayfile. A dayfile message describes the error.

If an error flag value other than KILL or RERUN occurs and an EXIT, EXIT(S), or EXIT(U) statement follows in the control statement record, 1AJ resumes job processing after that statement is encountered. If EXIT(C) is encountered prior to EXIT, EXIT(S), or EXIT(U) and the conditions mentioned above are true, the job terminates at that point.

If the operator has killed the job, the error flag value is set to 7. Permanent 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 nonzero disposition files. The message 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 entered 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 predayfile is created with its location contained in a supplement to the input file FNT. The predayfile becomes part of the job's dayfile when the job is rerun.

If abnormal termination occurs as the result of an ABORT macro call, job processing proceeds in accordance with the ABORT call parameters as discussed in the NOS/BE Reference Manual.

JOB POST-PROCESSING UTILITIES

During execution of a user program, various errors (such as an arithmetic mode error or exceeded time limit error) could cause the program and job to terminate prematurely. Normally, the operating system places a message in the dayfile and dumps the exchange package and 100g words of CM preceding and following the error location. The system then resumes job processing at an EXIT statement (if one is included) or terminates the job.

However, at the request of the user, the operating system returns control to the user program when an error condition is detected or the program terminates normally. Control is given to a routine supplied by the user which specifies procedures to be performed after abnormal or normal program termination. The NOS/BE Reference Manual describes the RECOVER and REPRIEVE macros and direct calls to the PP program RPV, which performs program recovery.

LIST-OF-FILES ADDRESS — GETLOF AND SETLOF MACROS

The list-of-files address is an 18-bit field in word 43 of the control point area. Currently it is used only by system programs (such as CYBER Record Manager) to access files which must be processed by a recovery routine if a job step ends with an error. (The system clears the list-of-files address when each job step begins.) The GETLOF and SETLOF macros provide access to the field containing the list-of-files address.

The format of the GETLOF macro is

```
label  GETLOF  addr,recall
```

label An optional symbolic address.

addr Address of the word to receive the current value in the list-of-files address field.

recall Optional recall parameter.

The format of the SETLOF macro is

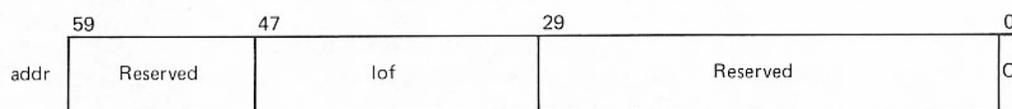
label SETLOF addr,recall

label An optional symbolic address.

addr Address of the word which contains the new value to be placed in the list-of-files address field.

recall Optional recall parameter.

For both GETLOF and SETLOF, addr has the following format



lof An 18-bit field containing current value in list-of-files address field (returned by GETLOF) or new value to be placed in list-of-files address field (set by user with SETLOF).

C Completion flag; cleared by macro initially and set to one when operation completes.

GETLOF and SETLOF are defined in the common deck ACTCOM.

JOB CONTROL WITH LOGICAL IDENTIFIERS

Logical identifiers (IDs) specify logical groups or functions. They allow extended control over job initiation and job/file processing. For example, logical IDs can be used in a single operating system to group all the jobs for a special printer. Those jobs must enter the system with the same logical ID.

The ID table, resident in CM, contains the information needed to process logical IDs. When the ID table has a length of zero, logical IDs are not allowed by the system. A job specifying a logical ID must wait to be processed until the operator enters that logical ID in the ID table. In the preceding example of grouping jobs for a special printer, the operator should wait until the printer is available, then enter the logical ID in the system's ID table and thus allow the jobs to execute.

A mainframe, in this discussion, is a hardware configuration controlled by a dedicated operating system. A linked mainframe communicates with a mainframe other than itself, and a multiframe system consists of linked mainframes. Each linked mainframe in a multiframe system communicates with the other mainframes. The ID table contains the name of the host mainframe (host ID), the logical IDs associated with the host mainframe, and the names of the other linked mainframes (physical or link IDs). A physical ID stored in the ID table is identical to the host ID of one of the other linked mainframes. For example, in a multiframe system consisting of two linked mainframes, one could have an ID table listing a host ID of MFA and a physical ID of MFB. The other mainframe would then have an ID table listing a host ID of MFB and a physical ID of MFA.

The host IDs and logical IDs consist of three letters or digits. The host ID in a multiframe system is unique because the third character of each host ID must be unique and must be a letter. As many as 58 decimal logical IDs can be associated with the host mainframe.

In a single mainframe system operating with an ID table of zero length, a job/file that does not specify a logical ID is processed on the system. A job/file that does specify a logical ID causes the system to flag the control statement as an error and terminate processing.

In a single mainframe system operating with an ID table containing the host PID and the list of associated logical IDs, a job/file that does not specify a logical ID is processed on that system. A job/file that specifies a logical ID causes the system to inspect the ID table for a matching logical ID. If a matching logical ID is found, the job/file is processed. If no match is found, processing waits until the operator places the specified logical ID in the LID table and the match is found.

In a multimainframe system, a LID table exists for each linked mainframe. A job/file that does not specify a logical ID is processed by the host system. A job/file that specifies a logical ID causes the host system to inspect its ID table. If a match is found in the list of logical IDs, the job/file is processed on the host system. If a match is not found, the host system checks its LID tables for a matching linked mainframe. If a match is found, the job/file is transferred to the appropriate linked mainframe. If a match is still not found, the job/file waits in queue on the host system until a match for its logical ID is found, either for the host system or for a linked mainframe system.

CEVAL MACRO

The CEVAL macro enables a hardware diagnostic routine to determine the status of equipment it is going to use. Before the user calls the macro, the diagnostic routine must set up a ten-word parameter block describing the equipment to be used and the type of operations (read-only or read/write) that will be performed. The system returns a code in the parameter block granting or denying the request. The routine should use this response code to generate an appropriate message for the user.

When the system grants a CEVAL request for a device that is nonallocatable and not an operator-assigned magnetic tape unit, it reserves the device by setting the maintenance flag and control point number in the device's EST entry. The system cannot access the device until the job releases it with the CEVAL request to return the unit to the system. Up to five nonallocatable devices can be reserved at a time. All reserved devices are released automatically at the end of the job.

When the system grants a CEVAL request for a Network Access Device (NAD), if the requesting diagnostic routine is granted exclusive access, or if the EST entry of the NAD is turned off, other programs cannot access the NAD. If the routine is granted shared access and the EST entry of the NAD is turned on, other programs can continue to access the NAD. The reserving and releasing is done in the same manner as documented for other nonallocatable devices in the preceding paragraph.

The CEVAL macro can also be used to request access to a remote NAD; that is, a NAD connected to the local NAD by the Loosely Coupled Network (LCN). The parameter block must specify both of the NADs. The system first validates and reserves the local NAD. Then, if the remote NAD is to be shared, the system grants the CEVAL request, leaving all further communication up to the requesting diagnostic routine. Otherwise, if the request is for dedicated access to the remote NAD, the system obtains the status of the two NADs and of the LCN. If the remote NAD is not in use, the system grants the CEVAL request with a response code of 34g, indicating situation indeterminate. The diagnostic routine should then request authorization from the system operator before attempting to communicate with the remote NAD. If the remote NAD is in use, or if a hardware error is discovered during the status check, the system releases the local NAD reservation and denies the CEVAL request with a response code of 16g, indicating the remote NAD is in use, or 33g, indicating a device error.

When the system grants a CEVAL request for a magnetic tape unit that was assigned previously by the operator, the system does not set the maintenance flag in the tape unit's EST entry. Therefore, it cannot be released with a CEVAL request (return unit to system request). If necessary, the unit can be released with the system control statements or macros RETURN or UNLOAD. When requesting an operator-assigned tape unit, only the mnemonics MT or NT should be specified in the parameter block; the EST ordinal, channel, equipment, and unit numbers must not be specified or the request is denied. CEVAL can request only one seven-track and one nine-track operator-assigned tape unit at a time.

The diagnostic routine must be allowed system access; that is, either it must reside in the system library and be called when the system is in engineering mode (enabled by the ENGR console command), or the job calling the routine must be initiated from the console. If the routine is not allowed system access, or if the parameter block is outside the job's field length, or if the routine improperly requests or releases a device, the system sends an error message to the dayfile and aborts the job without setting the complete flag.

The CEVAL macro is defined in the common deck ACTCOM.

The format of the macro is

CEVAL **addr,vo**

addr Location of the ten-word parameter block.

vo Optional parameter ignored unless vo=VO. When VO is specified, the system aborts the job if it is not allowed system access; otherwise, the system completes the macro call by setting the complete flag in the first word of the parameter block.

The format of the parameter block is:

	59	54	50	47	35	23	17	11	5	0	
addr + 0									Response Code		C
addr + 1	Error Bits			D					EST Ordinal		
addr + 2	Error Bits	S		D	Channel 1	Channel 2	Channel 3	Channel 4			
addr + 3	Error Bits	S		D				Local Device Code	Equipment Number		
addr + 4	Error Bits	S		D					Unit Number		
addr + 5	Error Bits			D				Device Mnemonic	Device Code		
addr + 6									Function Bits		
addr + 7	Error Bits			D	Pack Serial Number				Remote NAD Access Code		
addr + 10	Error Bits	S		D					B S I	TCU	
addr + 11	Error Bits	S		D				Remote NAD Device Code	Logical TCU Addr		

<u>Word</u>	<u>Bits</u>	<u>Description</u>																						
addr+0	11-6	Response code.																						
		<table border="1"> <thead> <tr> <th><u>Code (octal)</u></th> <th><u>Significance</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Validation granted.</td> </tr> <tr> <td>2</td> <td>Validation granted; element might be shared with another mainframe.</td> </tr> <tr> <td>4</td> <td>Validation denied; could not find element.</td> </tr> <tr> <td>5</td> <td>Validation denied; critical information may be destroyed.</td> </tr> <tr> <td>6</td> <td>Validation denied; element in use.</td> </tr> <tr> <td>7</td> <td>Validation denied; share byte not set.</td> </tr> <tr> <td>16</td> <td>Validation denied; NAD connected to host; that is, the remote NAD is in use.</td> </tr> <tr> <td>17</td> <td>Validation denied; no TCU enables specified.</td> </tr> <tr> <td>33</td> <td>Validation denied; device status error.</td> </tr> <tr> <td>34</td> <td>Situation indeterminate; no response.</td> </tr> </tbody> </table>	<u>Code (octal)</u>	<u>Significance</u>	0	Validation granted.	2	Validation granted; element might be shared with another mainframe.	4	Validation denied; could not find element.	5	Validation denied; critical information may be destroyed.	6	Validation denied; element in use.	7	Validation denied; share byte not set.	16	Validation denied; NAD connected to host; that is, the remote NAD is in use.	17	Validation denied; no TCU enables specified.	33	Validation denied; device status error.	34	Situation indeterminate; no response.
<u>Code (octal)</u>	<u>Significance</u>																							
0	Validation granted.																							
2	Validation granted; element might be shared with another mainframe.																							
4	Validation denied; could not find element.																							
5	Validation denied; critical information may be destroyed.																							
6	Validation denied; element in use.																							
7	Validation denied; share byte not set.																							
16	Validation denied; NAD connected to host; that is, the remote NAD is in use.																							
17	Validation denied; no TCU enables specified.																							
33	Validation denied; device status error.																							
34	Situation indeterminate; no response.																							
	0	Completion bit.																						

<u>Word</u>	<u>Bits</u>	<u>Description</u>	
addr+1 through addr+5, addr+7 through addr+11	59-55	Validation error bits.	
	<u>Bit Set</u>	<u>Significance</u>	
	59	Error in word.	
	58	Error in byte 1.	
	57	Error in byte 2.	
	56	Error in byte 3.	
	55	Error in byte 4.	
	48	Data-in-word flag.	
addr+2 through addr+4, addr+10, addr+11	54-51	Share bits; requests exclusive access if bits are set.	
	<u>Bit Set</u>	<u>Significance</u>	
	54	Element in byte 1.	
	53	Element in byte 2.	
	52	Element in byte 3.	
	51	Element in byte 4.	
addr+1	11-0	EST ordinal of element.	
addr+2	47-36 35-24 23-12 11-0	} Channel numbers.	
addr+3	23-12		Local device code (for NAD validation only).
	11-0		Equipment (controller) number.
addr+4	11-0		Unit number.
addr+5	23-12	Device mnemonic as shown under device code description.	
	11-0	Device code.	
	<u>Code</u>	<u>Mnemonic</u>	<u>Device</u>
	0001	AM	Reserved for installation.
	0002	AY	7054/844-2x disk.
	0003	AZ	7054/844-4x disk.
	0004	AYF	7154/844-2x disk.

<u>Word</u>	<u>Bits</u>	<u>Description</u>	
		<u>Code</u>	<u>Mnemonic</u> <u>Device</u>
		0005	AZF 7154/7155/844-4x disk.
		0006	AH 819 disk.
		0007	AJ 885 disk.
		0010 thru 0017	Reserved.
		0020	CR 405 card reader.
		0021	CP 415 card punch.
		0022	LQ Reserved for installation.
		0023	LR 580-12 line printer.
		0024	LS 580-16 line printer.
		0025	LT 580-20 line printer.
		0026	LRP 580-12 PFC line printer.
		0027	LSP 580-16 PFC line printer.
		0030	LTP 580-20 PFC line printer.
		0031 thru 0037	Reserved.
		0041	MTM Reserved for installation.
		0042	MTS 667 seven-track tape.
		0043	MTB 667 seven-track tape with block ID.
		0044	MTA 677 seven-track tape.
		0045	
		0046	Reserved.
		0047	
		0051	NTM Reserved for installation.
		0052	NTS 669 nine-track tape.
		0053	NTB 669 nine-track tape with block ID.

<u>Word</u>	<u>Bits</u>	<u>Description</u>	
		<u>Code</u>	<u>Mnemonic</u> <u>Device</u>
		0054	NTA 679 nine-track tape.
		0055	NTG 679 GCR nine-track tape.
		0056	} Reserved.
		0057	
		0060	DC Reserved for installation.
		0061	YC Reserved for installation.
		0062	DCX Reserved for installation.
		0063	YCX Reserved for installation.
		0064	FE 255x host communications processor.
		0065	CS 7077-1 LCC.
		0066	SC 6673 data set.
		0067	CC 6683 coupler.
		0070	ED DDP.
		0071	AX Extended memory coupler.
		0072 thru 0277	} Reserved.
		0300	NC Network Access Device (on local mainframe).
		0301	NC Network Access Device (on remote mainframe).
		0302 thru 7677	} Reserved.
		7700 thru 7777	} Reserved for QSEs.

<u>Word</u>	<u>Bits</u>	<u>Description</u>												
addr+6	6-0	Function bits. <table border="1" data-bbox="672 397 1302 755"> <thead> <tr> <th><u>Bit</u></th> <th><u>Significance</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Load controller memory (controller share byte must be set).</td> </tr> <tr> <td>1</td> <td>Will write on unit.</td> </tr> <tr> <td>2</td> <td>Will not use preallocation area.</td> </tr> <tr> <td>3</td> <td>Return unit to system (overrides other function bits; EST ordinal is required).</td> </tr> <tr> <td>6</td> <td>Designated element is not in system EST.</td> </tr> </tbody> </table>	<u>Bit</u>	<u>Significance</u>	0	Load controller memory (controller share byte must be set).	1	Will write on unit.	2	Will not use preallocation area.	3	Return unit to system (overrides other function bits; EST ordinal is required).	6	Designated element is not in system EST.
<u>Bit</u>	<u>Significance</u>													
0	Load controller memory (controller share byte must be set).													
1	Will write on unit.													
2	Will not use preallocation area.													
3	Return unit to system (overrides other function bits; EST ordinal is required).													
6	Designated element is not in system EST.													
addr+7	47-0	Pack serial number (binary).												
	15-0	Remote NAD access code.												
addr+10	5-4	Buffer size index (for remote NAD validation). <table border="1" data-bbox="672 914 840 1033"> <tbody> <tr> <td>0</td> <td>516.</td> </tr> <tr> <td>1</td> <td>2064.</td> </tr> <tr> <td>2</td> <td>4128.</td> </tr> <tr> <td>3</td> <td>Reserved.</td> </tr> </tbody> </table>	0	516.	1	2064.	2	4128.	3	Reserved.				
0	516.													
1	2064.													
2	4128.													
3	Reserved.													
	3-0	Trunk Control Unit Enable flags (for remote NAD validation, specifies which of the TCUs on the local NAD can be used to communicate with the remote NAD). <table border="1" data-bbox="672 1172 1050 1252"> <tbody> <tr> <td>Bit 0</td> <td>- TCU 3</td> <td>Bit 1</td> <td>- TCU 2</td> </tr> <tr> <td>Bit 2</td> <td>- TCU 1</td> <td>Bit 3</td> <td>- TCU 0</td> </tr> </tbody> </table>	Bit 0	- TCU 3	Bit 1	- TCU 2	Bit 2	- TCU 1	Bit 3	- TCU 0				
Bit 0	- TCU 3	Bit 1	- TCU 2											
Bit 2	- TCU 1	Bit 3	- TCU 0											
addr+11	23-12	Remote NAD device code (for remote NAD validation).												
	7-0	Logical TCU address of remote NAD.												

CESAB MACRO

The CESAB macro returns a parameter block that contains information about the current hardware environment. This information includes the following:

- PP speed
- Amount of physically available CM
- Mainframe model
- Mainframe characteristics
- Physical and logical PP/PPU bit maps

The format of the macro is

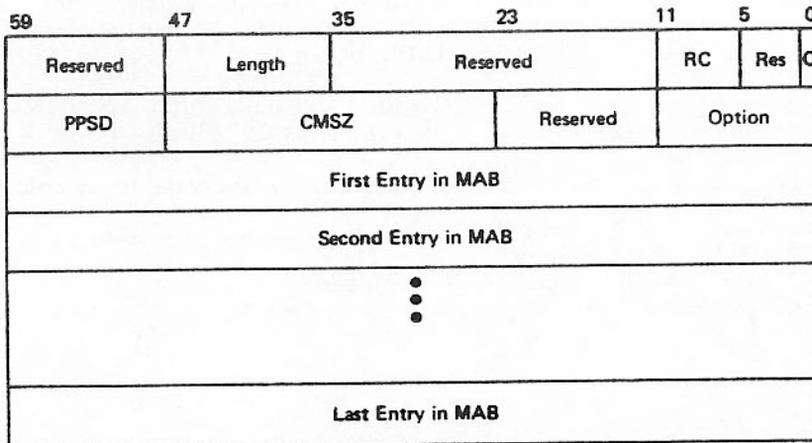
CESAB addr

addr Location of the parameter block.

Word 0 of the parameter block must be created at location addr before the CESAB call is made. This involves setting bits 47 through 36 to the maximum allowable buffer size for the parameter block (in words) and clearing bit 0.

The CESAB macro is defined in the common deck ACTCOM.

The format of the CESAB parameter block is shown in the following diagram.



<u>Field</u>	<u>Bits</u>	<u>Description</u>
LNGTH	47-36	The caller initially sets this field to the size of the buffer to which the parameter block will be written. If the buffer is not large enough to contain the parameter block, the CESAB call is aborted. The system resets the value of this field to the actual size of the parameter block.
RC	11-6	Return code. If 0, no errors occurred.
C	0	Complete bit. The caller must clear this bit before executing the macro. The system sets this bit after data has been written to the buffer.
PPSD	59-48	PP speed flag.

<u>Value</u>	<u>Meaning</u>
0	1x speed (PP major cycle = 1 us).
1	2x speed (PP major cycle = 0.5 us).
3	4x speed (PP major cycle = 0.25 us).

CMSZ	47-24	Physically available CM/100g.
OPTN	11-0	Mainframe characteristics.

<u>Bits</u>	<u>Meaning (when bit is set)</u>
11	Mainframe is not a CYBER 180-class, 865, or 875 mainframe.
10-9	Mainframe is a CYBER 170 Model 176 if either bit is set or if both bits are set.
8	Mainframe is not a CYBER 170 Model 720, 730, 740, 750, 760, or CYBER 176 Model B.
7	ILR is not present.
6	SCR is not present.
5	Not used.
4	CPU 0 has no instruction stack.
3	CMU is not present or is not used.
2	CEJ/MEJ are not present or are not used.
1	CPU 1 is not present or is not used.
0	CPU 0 is not used.

The variable section for all models except the CYBER 180-class mainframes contains at least one entry of each of the following types:

CPU Entry

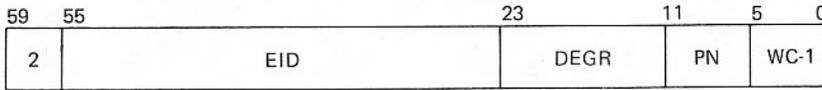
59	55	47	39	5	0
1	Reserved	MDL	Reserved	WC-1	

Memory Entry

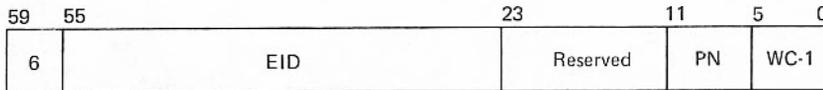
59	55	47	39	5	0
5	Reserved	MDL	Reserved	WC-1	

The variable entry section for CYBER 180-class mainframes contains at least one entry of each of the following types:

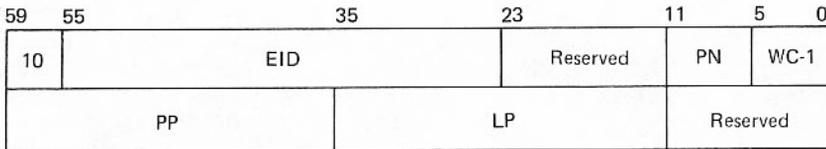
CPU Entry



MEMORY Entry



IOU Entry



<u>Field</u>	<u>Bits</u>	<u>Description</u>
EID	55-24	Element identifier. The contents of the corresponding EID register.
DEGR	23-12	Map and cache degrade flags.
	<u>Bits</u>	<u>Meaning</u>
	23-20	Page map element unavailable.
	19-18	Segment map element unavailable.
	17-16	Not used.
	15-12	Cache element unavailable.
PN	11-6	Port number. The maintenance channel port used to access the maintenance registers for the element.
WC-1	5-0	An octal number that is one less than the word count of the entry.
PP	59-36	Physical PP bit map.
	<u>Bits</u>	<u>Meaning</u>
	59-58	Not used.
	57-48	Each bit that is set indicates that the corresponding PP (9-0) is not physically present.
	47-46	Not used.
	45-36	Each bit that is set indicates that the corresponding PP (19-10) is not physically present.

<u>Field</u>	<u>Bits</u>	<u>Description</u>
--------------	-------------	--------------------

or

Physical PPU bit map.

<u>Bits</u>	<u>Meaning</u>
-------------	----------------

59-49	Not used.
48-36	Each bit that is set indicates that the corresponding PPU (12-0) is not physically present.

LP	35-12	Logical PP bit map.
----	-------	---------------------

<u>Bits</u>	<u>Meaning</u>
-------------	----------------

35-34	Not used.
33-24	Each bit that is set indicates that the corresponding PP (9-0) is turned off or is not physically present.
23-22	Not used.
21-12	Each bit that is set indicates that the corresponding PP (19-10) is turned off or is not physically present.

or

Logical PPU bit map.

<u>Bits</u>	<u>Meaning</u>
-------------	----------------

35-25	Not used.
34-12	Each bit that is set indicates that the corresponding PPU (12-0) is turned off or is not physically present.

ACCESS TO LOADER WORD — GETLC AND SETLC MACROS

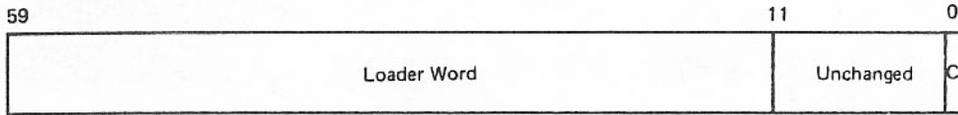
Certain system programs, such as the Loader and CYBER Interactive Debug, use information contained in W.CPLDR1, the loader word, in the control point area. The GETLC and SETLC macros provide access to this word. Although any program can call these macros, they are intended for use only by system programs.

The GETLC macro format is:

```
GETLC  addr
```

addr Address of the word to receive the current loader word.

The GETLC request is made with auto recall set. The initial contents of addr is ignored. When the operation is complete, addr has the following format.



Loader word Bytes 0 through 3 of W.CPLDR1. Refer to word 55 (W.CPLDR1) of the control point area table in appendix A for the format of the loader word.

C Completion flag; initially cleared by macro; set to one when the function completes.

The SETLC macro format is

SETLC addr

addr Address of the word which contains the new value of the loader word.

The SETLC request is made with auto recall set. The user sets bits 59 through 24 of addr to correspond to the desired loader word settings. The remainder of addr is ignored except for bit 0. Bit 0 is the completion flag, similar to the flag for GETLC. It is cleared by the macro initially and set to one when the function completes.

Before setting bits in the loader word, the user should read the current contents of the word via the GETLC macro. In this way, the user can avoid setting bits unnecessarily and changing bits unintentionally. Bit 49 (S.CPLP), which indicates a nonsystem library program, can be set but never cleared.

GETLC and SETLC are defined in the common deck ACTCOM, which is included in CPCTEXT, CPUTEXT, and SCPTEXT.

INTRODUCTION

EDITLIB is a utility for creating, maintaining, and manipulating files that contain binary program code. EDITLIB can be used in either of two forms. User EDITLIB, which is described in the NOS/BE Reference Manual, provides the user with the means to create and maintain user libraries. System EDITLIB provides the systems programmer with the following capabilities:

- Creating deadstart tapes.
- Creating and maintaining system libraries.
- Manipulating the running system to effect temporary changes.
- Modifying the running system or the deadstart tape by merging in permanent changes and creating a new deadstart tape.

To create a new deadstart tape and/or running system for a specific configuration, the user obtains an unconfigured deadstart tape from Software Manufacturing and Distribution. This tape contains the binary decks of the entire operating system to be used to build the new system. After appropriate assemblies and compilations have been performed, EDITLIB is called by selected control statements and directives to modify the current system. Thus, a new deadstart tape and/or running system can be created, which reflects the system as it is physically configured (figure 8-1).

System EDITLIB can function like user EDITLIB. Directive statements before the first READY, after the last COMPLETE, or between a COMPLETE and the next following READY are interpreted as they would be in user EDITLIB. However, user EDITLIB does not allow the addition of PP programs.

EDITLIB can handle object code only if it is the type processed by the CYBER loader; 7000-type object code cannot be processed. All object code for EDITLIB input must be in system logical format. All procedure records must be in zero-byte terminator format. S, L, or other nonsystem format tapes cannot be processed by EDITLIB.

CHARACTER SET

EDITLIB interprets \$ () = + - * / , and . as delimiters. Blanks are free characters and are removed unless in a literal delimited by dollar signs (\$). From the character set, the user can create:

Symbol	Single character or a string of characters.
Vocabulary word	Symbol that has meaning as defined by the semantics.
Name	Symbol that is defined by syntax and semantics.

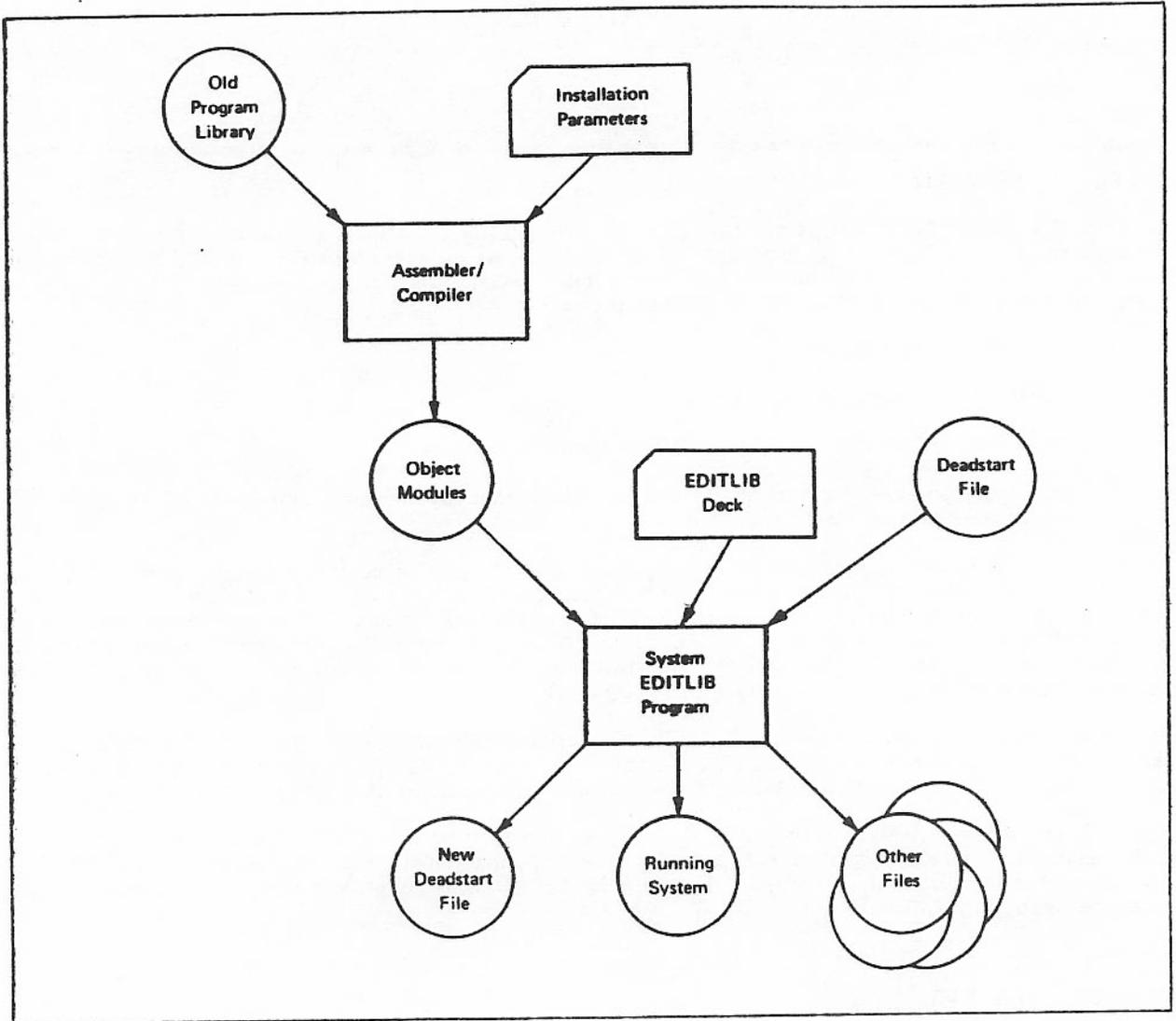


Figure 8-1. Basic Usage of System EDITLIB

When any delimiter and/or a blank is used to form a symbol, the symbol must be enclosed between \$ signs (refer to example). When a \$ sign is to be used as a character within a symbol, an additional \$ sign must be written immediately adjacent to each \$ sign desired. The only character that is removed is the first \$ sign of each pair of \$ signs encountered (refer to example).

Whenever a name could be interpreted as a number or a vocabulary word, that name should be enclosed between \$ signs.

Examples:

\$-DSD-\$	Symbol is -DSD-.
\$\$\$CIO\$	Symbol is \$CIO.
\$D\$I\$\$\$\$\$-\$	Symbol is D\$I\$\$\$\$-.

The input to EDITLIB is the first 72 columns of an 80- or 90-column card image. A directive must be completely contained in one statement.

SYNTAX AND SEMANTICS

The formats of the various directives and the meanings associated with the symbols are presented here. 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 sets the abort flag. After EDITLIB has completed a given task, such as checking all directive formats, it checks the abort flag. When set, EDITLIB aborts. EDITLIB continues to check a directive, even if it has located an error in that directive. EDITLIB proceeds 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.

The following are the symbols used in the syntax.

<u>Symbol</u>	<u>Significance</u>
[]	Enclosed items are optional.
⊥	Or.
()	Enclosed item is the actual symbol.
p	Name of a routine.
lfn	File name.
n	Positive or negative integer.
pn	Library name.
r	Residence:
	CM Central memory.
	DS Disk.
	ECS Extended memory; if no extended memory available, then default to disk.
	EM Extended memory; if no extended memory available, then default to central memory.
	Default Disk.

Interval Program name or range of program names in one of the following forms.

P_1

P_1+P_2

P_1-P_2

$P_1/P_2/P_3$

P_1 is a single program name. $P_1/P_2/P_3$ is valid only for ADD and REPLACE directives. It is acceptable to replace either P_1 or P_2 , but not both, with * . EDITLIB ignores a directive under 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.

* Replaces p in the syntax. When * is encountered, EDITLIB considers 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.

+ Designates 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.

- Designates 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.

/ Delimits routine names.

Notes:

- EDITLIB assigns a number to each directive it processes.
- EDITLIB lists each input statement it receives on the file designated for listable output by the L parameter on the EDITLIB statement. The directive, as interpreted by EDITLIB, appears directly below the input statement. The interpreted directive is prefaced by the number assigned to this directive. All comments about the directive follow the interpreted directive. EDITLIB is a two-pass program. EDITLIB interprets and checks all directives during the first pass; it then proceeds to completion, unless a fatal error is encountered. EDITLIB may stop then, however, in the second pass. Error messages are generated by referencing the number that is assigned to each directive. Error messages are issued whenever EDITLIB finds it cannot comply with the user's request. EDITLIB also issues messages when it is unable to determine what the user is trying to accomplish.

- The directives ADD and REPLACE produce a CONTENT directive on the routines involved.
- Any interval specified in a directive is interpreted to mean:

In a reference to a file that is a library, the interval is formed by the entries in the order they appear in the program name table (PNT). The interval begins when P₁ is encountered and ends when either P₂ or the last entry in the PNT is encountered. The table is not handled circularly.

In a reference to a file that is not a library, the interval is formed by the order of the routines on the file. To find P₁, EDITLIB searches from current file position to end-of-file, and then if necessary, rewinds and searches the file from the beginning. The interval ends at whichever occurs first, P₂ or end-of-file. The file is handled circularly only when searching for P₁.

- In a system EDITLIB, all directives that do not specifically name a particular library and are not contained between a LIBRARY and FINISH directive refer to the PP library. This means that the PP library is the default library when no library is specified. In the user EDITLIB, no PP programs are allowed.
- EDITLIB does not support TEXT, SEGLOAD records and/or DATA input (7000 LIBEDT). If TEXT/SEGLOAD record/DATA input is found, a message is issued and the unrecognized records within the interval are ignored.
- CYBER EDITLIB ignores miscellaneous 7000 LIBEDT directives TYPE, ERROR, PCOPY, PMOVE, MOVEC, COPYC, and DELETEC, and prints a warning message on each directive.
- System EDITLIB considers entry points which are four characters long and end with an equal sign (for example, MFL= and RFL=) to be special entry points. These entry points are either used by EDITLIB as FL control or they are ignored.

SYSTEM EDITLIB

EDITLIB FILES

EDITLIB uses the following files.

<u>File Name</u>	<u>Cataloged</u>	<u>File Use</u>
ZZZZZ01	Yes	RESET.
ZZZZZ02	Yes	RESTORE.
ZZZZZ03	Yes	System extension.
ZZZZZ04	Yes	SYSTEM.
ZZZZZ05	No	Interpreted directives.
ZZZZZ06	No	Extended memory resident routines library file (special case system file).
ZZZZZ07	No	Entry point name table spill file.
ZZZZZ08	No	Program number table spill file.
ZZZZZ10	No	Program name table spill file.

<u>File Name</u>	<u>Cataloged</u>	<u>File Use</u>
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 produces the following type of files.

<u>File Use</u>	<u>lfn</u>	<u>pfm</u>
RESET	ZZZZZ01	ZZZZZ01
RESTORE	ZZZZZ02	ZZZZZ02
System extension	ZZZZZ03	ZZZZZ03
SYSTEM	ZZZZZ04	ZZZZZ04
Extended memory library	ZZZZZ06	
Current directory	ZZZZZ23	ZZZZZ23

The files that are made permanent are cataloged using the following values.

<u>File Name</u>	<u>ID</u>	<u>RP</u>	<u>TK</u>	<u>CN</u>	<u>MD</u>	<u>EX</u>
ZZZZZ01	SYSTEM	999	SSSSSSSS	XNOX	XNOX	---
ZZZZZ02	SYSTEM	999	SSSSSSST	---	---	---
ZZZZZ03	SYSTEM	999	SSSSSSSU	XNOX	XNOX	---
ZZZZZ04	SYSTEM	999	SSSSSSSV	XNOX	XNOX	XNOX
ZZZZZ23	SYSTEM	999	DIRECTORY	XNOX	XNOX	---

The allocation of extended memory is controlled by Monitor. Since the allocation of extended memory involves the creation of tables and pointers by Monitor, the deadstart loader does not duplicate this effort to set up extended memory files. An EDITLIB job is required to place routines into extended memory after deadstart.

EDITLIB maintains a permanent file (ZZZZZ23) for the deadstart loader. This file contains the current system directory as it appears in CMR.

When EDITLIB modifies the running system, it adds the new routines to the end of file ZZZZZ03. Using this method, EDITLIB does not prevent others from accessing the file, except during the short period of time while the new directory is written to CMR. Bit S.EDTRUN in byte C.DSFLAG or word P.LIB is set when the directory is changed.

SPECIAL HANDLING OF LOCAL FILE NAME SYSTEM

The system file is named ZZZZZ04. To allow the use of the local file name SYSTEM and to clarify the handling of the libraries on a system-type file, the user must understand that the use of the file name SYSTEM depends on the context in which it is used.

If the file name SYSTEM does not appear between the directives READY and COMPLETE, it is treated as a local file whose name just happens to be SYSTEM.

If the file name SYSTEM does occur between a READY and COMPLETE directive, the CONTENT directive treats it as a local file name. Also, ADD and REPLACE treat it as a local file name whenever the LIB parameter is absent from the optional parameter list on those directives.

Examples:

```
ADD(interval,SYSTEM,AL=1)
```

Add the interval from the local file named SYSTEM.

```
ADD(interval,SYSTEM,LIB)
```

Add the interval from the running system file (ZZZZZ04 or ZZZZZ03), specifically the PP library.

```
ADD(interval,SYSTEM,LIB=NUCLEUS,FLO=1)
```

Add the interval from the running system file (ZZZZZ04 or ZZZZZ03), specifically the CP library called NUCLEUS, with the field length override (FLO) option.

CONTROL STATEMENT

The program call statement is:

```
EDITLIB(SYSTEM,t1,t2,...,tn)
```

<u>Parameter</u>	<u>Description</u>
SYSTEM	This parameter is a password defined in EDITLIB. It is used by EDITLIB to determine what kind of EDITLIB run is to be performed. This key is a program parameter, which cannot exceed nine characters; it allows restricted access to the running system directory.

t_i One of the following:

<u>t_i</u>	<u>Description</u>
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 uses the directory from the RESET permanent file to replace the directory presently in CMR; it changes this directory as prescribed by directives in the input file. EDITLIB replaces the current directory with the directory just modified. A copy of the current directory is saved on the RESTORE permanent file before the change takes place.

ParameterDescription

<u>t_i</u>	<u>Description</u>
RESTORE	This parameter informs EDITLIB to replace the current system directory with the directory as it appeared before the last EDITLIB took place. When the RESTORE parameter is encountered, EDITLIB uses the directory on the RESTORE permanent file to replace the directory presently in CMR. The rest of the procedure EDITLIB follows is the same as for an EDITLIB (SYSTEM, RESET), except that the current directory is not saved on the RESTORE file.
I=lfm	This symbol designates the file containing EDITLIB directives. If it is not present, lfm INPUT is used. If INPUT is used, any binary input to EDITLIB on this file must be in the order in which it is requested. This file is not searched. I, appearing alone, is equivalent to I=INPUT.
L=lfm	This symbol designates the file of listable output from EDITLIB. If it is not present, lfm OUTPUT is used. L, appearing alone, is equivalent to L=OUTPUT.
MSGL=n	This parameter determines the type of list or dayfile output generated during execution; n equals 0 to 8. The default value is 0. Refer to table 8-1.

TABLE 8-1. DIRECTIVE INTERPRETATION AND EXECUTION

n	Directive Interpretation		Directive Execution	
	Message to Dayfile	List Input Statement	List Prefix Table	List Optional Parameters
0	No	Yes	Yes	Yes
1	Yes	Yes	Yes	Yes
2	No	No	No	Yes
3	Yes	No	No	Yes
4	No	Yes	Yes	No
5	Yes	Yes	Yes	No
6	No	No	No	No
7	Yes	No	No	No
8	No listing produced		No listing produced	

Parameter

Description

<u>t_i</u>	<u>Description</u>
ERROR=n	This parameter determines when an abort occurs. The degree of severity of each error is checked against the error flag set by the ERROR parameter. Bit 3 set indicates a critical error, bit 2 set indicates a serious error, bit 1 set indicates a minor error, and bit 0 set indicates a warning error. Checking is performed only when the COMPLETE directive is encountered in system mode.

<u>n</u>	<u>Fatal</u>	<u>Critical</u>	<u>Serious</u>	<u>Minor</u>	<u>Warning</u>
0	Yes	No	No	No	No
1	Yes	Yes	Yes	Yes	No
2	Yes	Yes	Yes	No	Yes
3	Yes	Yes	Yes	No	No
4	Yes	Yes	No	Yes	Yes
5	Yes	Yes	No	Yes	No
6	Yes	Yes	No	No	Yes
7	Yes	Yes	No	No	No
8	Yes	No	Yes	Yes	Yes

When ERROR=n is not specified, any error or warning causes job termination.

EDITLIB errors are explained in the Diagnostic Handbook.

Examples:

- EDITLIB(SYSTEM)
- EDITLIB(SYSTEM,RESET)
- EDITLIB(SYSTEM,MSG1=1,RESTORE)
- EDITLIB(SYSTEM,I=INFILE,MSG1=0)
- EDITLIB(SYSTEM,I=INFILE,L=OUTFILE)
- EDITLIB(SYSTEM,I,L)
- EDITLIB(SYSTEM,MSG1=7,I,L=INFILE)

DIRECTIVES

OPTIONAL DIRECTIVE PARAMETERS

Optional parameters are positionally independent.

<u>Parameter</u>	<u>Description</u>
AL=al	al is one to four octal digits from 0 to 7777. Default is 0. AL stands for access level and is a parameter of the ADD and REPLACE directives. Of the 12 bits, 11 are meaningful to INTERCOM only. The rightmost bit of this field is used by the system to regulate access to the program through control statement requests. When this bit is set to 0, the routine or entry cannot be called from a control statement. 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 three sections. The first section is 1 bit in size and is the rightmost bit of the field. The remaining 11 bits are divided into two sections according to the installation parameter, IP.IACES. IP.IACES=n defines the number of bits to be reserved for the access level. The remaining bits (11-n) are used as permission bits.

When the use of a command is requested, the user's 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 bits 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	k is from 0 to 377777 octal. Default is 0. FL is a parameter of the ADD and REPLACE directives and stands for field length. It is meaningful for CM programs only. The value is used in allocating memory when the program is called. This field length must take into account the CM needed by the loader plus all programs, overlays, and segments that can be loaded by the specified program. (Refer to FLO parameter description.)
------	---

The two cases in which the value of FL will be overridden are as follows.

1. The existence of HHA (absolute overlays with 5400 tables). If HHA has a value higher than that specified on the FL parameter, the value of HHA is used.
2. The existence of either an RFL= or an MFL= entry point. If the low order 16 bits of the value of these entry points is greater than the FL specification and greater than HHA (if present), the value of these entry points is used. These special entry parts are not entered in the entry point table, and are interpreted as follows.

RFL= Specifies the FL and implies FLO=0.

MFL= Specifies the FL and implies FLO=1.

NOTE

Only one of RFL=/MFL= should be specified. If both are given, only the higher value is considered for field length. If both values are identical, and are higher than the FL specification and HHA (if it exists), the last one encountered determines the setting of FLO.

FLO=m m is 0 or 1. Default is 0. This parameter can only appear on an ADD or REPLACE directive. It sets or clears the field length override bit, which determines the execution field length. If set, the user's field length can override the FL value set in the library.

The value used by the system to assign memory for execution of a library resident program is either the field length specified by the EDITLIB FL parameter (PFL) or the nominal field length (NFL). NFL is the last RFL request (if one was made), the field length specified on the job statement, or the default value IP.SFL. The system determines which field length to assign, depending on the setting of the FLO parameter and the automatic field length flag (reduce bit).

<u>Conditions</u>	<u>Field Length Assigned</u>
PFL=0, reduce bit is off.	NFL
PFL=0, reduce bit is on.	IP.SFL
PFL=0, FLO=0.	PFL
PFL=0, FLO=1, reduce bit is off.	NFL
PFL=0, FLO=1, reduce bit is on.	PFL

Parameter

Description

The field length assigned cannot exceed that specified on the job statement. Any job that does not have sufficient field length for the library program is terminated abnormally.

LIB=lib lib is the name of a library. This parameter 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. LIB specifies which library on the deadstart file is to 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 contains 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 memory; if no extended memory available, then default to disk.††
DS	Disk (default).
EM	Extended memory; if no extended memory available, then central memory.

†Either NEW or OLD is mandatory, not both.

††Not allowed for procedure files.

DIRECTIVE RESTRICTIONS

The following list indicates where directives must occur relative to LIBRARY-FINISH and READY-COMPLETE sequences. Any violations of these restrictions are diagnosed by EDITLIB before any directives are executed and can cause an abort accompanied by diagnostic error messages.

<u>Directive</u>	<u>Restrictions</u>
ADD	Inside LIBRARY-FINISH or inside READY-COMPLETE.
CHANGE	Inside READY-COMPLETE.
COMPLETE	Outside LIBRARY-FINISH and after READY.
CONTENT	None.
DELETE	Inside LIBRARY-FINISH or inside READY-COMPLETE.
ENDRUN	Outside LIBRARY-FINISH and outside READY-COMPLETE.
FINISH	After LIBRARY.
INCLUDE	Inside READY-COMPLETE.
INCLUDEP	Inside READY-COMPLETE.
LIBRARY	Not inside LIBRARY-FINISH.
LISTLIB	Outside LIBRARY-FINISH.
LISTLNT	Inside READY-COMPLETE.
MOVE	Inside READY-COMPLETE.
RANTOSEQ	Outside LIBRARY-FINISH and outside READY-COMPLETE.
READY	Not inside READY-COMPLETE.
REMOVE	Outside LIBRARY-FINISH and inside READY-COMPLETE.
REPLACE	Inside LIBRARY-FINISH or inside READY-COMPLETE.
REWIND	None.
SEQTORAN	Outside LIBRARY-FINISH and outside READY-COMPLETE.
SETAL	Inside LIBRARY-FINISH or inside READY-COMPLETE.
SETFL	Inside LIBRARY-FINISH or inside READY-COMPLETE.
SETFLO	Inside LIBRARY-FINISH or inside READY-COMPLETE.
SKIPF	None.
SKIPB	None.
TRANSFER	Inside READY-COMPLETE.
TRANS77	Inside READY-COMPLETE.

DIRECTIVE FORMATS

The following are the system EDITLIB directive formats. Refer to optional directive parameters for additional information.

ADD Directive

This directive adds the specified program(s) from the lfn to the library under construction or modification.

```
ADD(interval,lfn[,r][,(AL=)a1][,(FL=)k][,(LIB=)lib][,(FLO=)m])
```

If the lfn contains a library, the program name table of the library is searched to locate the specified programs to be added. If the file is not a library, the search for the specified programs begins at the current record position on the file and proceeds to end-of-file. If the programs are not found, the file is rewound and the search continues until all programs have been searched. If the programs are not found in the library or on the file, an error message is issued. If the programs to be added already exist in the library, an error message is issued. The ADD directive should not be used to replace the current program with the new program. This function is accomplished by a REPLACE directive.

Examples:

```
ADD(HIGH,FILEH,ECS,AL=71,FL=1000)
ADD(HIGH,FILEH,FLO=1)
ADD(HIGH,SYSTEM,LIB=A,ECS,AL=123)
```

CHANGE Directive

This directive is used to change the residence of the library tables in the LNT.

```
CHANGE(lib[,r])
```

If the library already has the specified residence or if the library is not part of the system, EDITLIB issues an appropriate comment.

Example:

```
CHANGE(SYSLIB,CM)
```

COMPLETE Directive

This directive informs EDITLIB that the directive list for the directory whose name appeared in the READY directive has been exhausted and there are no more modifications or additions to be performed on the directory.

```
COMPLETE.
```

This directive must be preceded by a READY directive.

CONTENT Directive

This directive is used to obtain information about a program or series of programs on a specified file (lfn).

```
CONTENT(interval,lfn)
```

The information is obtained from the program-expanded prefix table and the program itself. The CONTENT directive applies to all files other than a deadstart file or a LIBRARY file. If information is desired on PP programs, the SYSTEM parameter must be entered on the EDITLIB call statement. In a multifile situation, CONTENT must be specified for each file.

Example:

```
CONTENT(ONE,FILE1)
```

DELETE Directive

This directive removes all entries from the library table that refer to the specified programs.

```
DELETE(interval)
```

Examples:

```
DELETE(FEAR)
```

```
DELETE(FEAR+HATE)
```

ENDRUN Directive

This directive indicates that execution of directives is to stop.

```
ENDRUN.
```

All directives following ENDRUN are checked, but not processed. This is an optional directive; if it is not found, one is generated. If this directive is found and EDITLIB finds an error in a directive after the ENDRUN card, it is flagged but does not stop EDITLIB from executing directives which precede ENDRUN.

FINISH Directive

This directive designates the end of the list of directives that affects the library defined by the last LIBRARY directive.

```
FINISH.
```

Each LIBRARY directive must have a FINISH directive. It is permitted to have multiple (LIBRARY, FINISH) sequences within a single directive sequence.

INCLUDE Directive

This directive adds the library on the specified file to a specified system.†

```
INCLUDE(lib,lfn[,r])
```

This allows a user library to be made part of the running system. This directive is also used to add libraries from a system to a new system on a deadstart tape.

Examples:

```
INCLUDE(MYLIB, MYLIB,ECS)
```

```
INCLUDE(FTN5LIB, SYSTEM)
```

The following directive allows an old PP program library to be added to the current directory model.

```
INCLUDEP(lfn)
```

The PP program library in the current model must be empty when the directive is encountered. Residency and access levels of each entry are that of the entries in the source PP program library.†

Example:

```
INCLUDEP (SYSTEM)
```

LIBRARY Directive

This directive performs two functions during a system EDITLIB run.

```
LIBRARY(lib,(OLD) ↓ (NEW),(r))
```

When encountered outside of a READY and COMPLETE directive, the action taken is equivalent to a user's EDITLIB. When the directive is between a READY and COMPLETE, the action is also equivalent to that of a user's EDITLIB, but the LIBRARY is found using the library name table (OLD) or is added to the library name table (NEW).

Examples:

```
LIBRARY(FTN5LIB,OLD)
```

```
LIBRARY(MYLIB,NEW,ECS)
```

```
LIBRARY(COBOL,NEW)
```

† When creating a new deadstart tape from the running system, programs that were extended memory resident become disk resident.

LISTLIB Directive

This directive is used to list a library which can be a user's library, running system, new system, and/or deadstart file.

```
LISTLIB(interval,lfn[,lib])
```

When between READY and COMPLETE, the lfn listed is the running system and/or the new system under construction. When LISTLIB 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. LISTLIB must be called for each library on a file.

These directives send information about the specified program(s) on the lfn or on the lfn and in the lib to the output file. The information comes from the object deck being processed.

Examples:

```
LISTLIB(*,SYSTEM,FTN5LIB)
```

```
LISTLIB(COPY,FILE1,A)
```

```
LISTLIB(1AJ,SYSTEM)
```

LISTLNT Directive

This directive must appear between a READY and a COMPLETE directive.

```
LISTLNT.
```

It lists the library name table of the file referenced on the READY directive. It can be used to list either the old system or new system LNTs. It does not list the LNT on an old deadstart file. There are no parameters.

MOVE Directive

This directive changes the residence of CP and PP programs within a system library.

```
MOVE(interval,r)
```

EDITLIB moves PP programs to disk (specified by DS), extended memory (specified by ECS or EM), or CM. Movement of CP programs depends on the residence of the system library of which the CP program is a part. If the library is in CM, movement is unrestricted; if the library is on disk or in extended memory, the new residence can be specified only as DS, ECS, or EM. The library directory residence must be at least as fast as the new residence.

After a library program is requested to be extended memory resident and a GO response is received, the user should request LISTLIB to determine if the program is resident in extended memory. This is advisable because output is generated before the actual move is made and, if extended memory is not available, the program may reside on disk (if ECS was specified) or in CM (if EM was specified).

Examples:

```
MOVE(IAJ,CM)
MOVE(MDD,DS)
MOVE(FILE1+FILE2,CM)
```

RANTOSEQ Directive

This directive places a random user library on a sequential file in library format.

```
RANTOSEQ(lfn1,lfn2)
```

lfn₁ is the source file name; lfn₂ is the destination file name. All records not referenced by the PNT are lost.

READY Directive

This directive is the first directive of any directive list pertaining to the lfn specified.

```
READY(lfn [(,OLD) (NEW)] )
```

Only those directives between a READY and a COMPLETE affect the lfn. READY is used in conjunction with a system EDITLIB run.

READY specifies that the operation is performed on a directory. The lfn is either the running system or the name of the file which contains the new system. If the name SYSTEM appears as the lfn, it means the running system whether the OLD parameter is specified or not; if OLD is specified, lfn must be SYSTEM. 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), and those that do not change any part of the system directory.

Examples:

<u>Directive</u>	<u>Identical to</u>
READY(FUTURE,NEW)	READY(FUTURE)
READY(SYSTEM,OLD)	READY(SYSTEM)

REMOVE Directive

This directive removes the library from the running system.

```
REMOVE(lib)
```

The library's entry in the library name table is removed.

Example:

```
REMOVE(FTN5LIB)
```

REPLACE Directive

This directive replaces the program(s) specified if it is already in the library, or adds the program(s) if it is not in the library.

```
REPLACE(interval, lfn[,r] [, (AL=)al] [, (FL=)k] [, (LIB=)lib] [, (FLO=)m])
```

The REPLACE directive in a system EDITLIB retains the AL, FL, FLO, and r values of the replaced program, unless otherwise specified.

Examples:

```
REPLACE(AUDITZ, LGO, AL=7777)
```

```
REPLACE(HPA1+HPA9, CIULBIN)
```

In the first example, the previous FL, FLO, and r values are retained. In the second, the previous AL, FL, FLO, and r values are retained.

REWIND Directive

This directive positions the specified file to the first record on the file.

```
REWIND(lfn [ /lfn ] ...)
```

To accomplish this, EDITLIB issues a CIO request to do a REWIND. In system mode, the file name SYSTEM refers to the running system.

SECTORAN Directive

This directive places a sequential user library on a random file in library format.

```
SECTORAN(lfn1, lfn2)
```

lfn₁ is the source file name; lfn₂ is the destination file name.

SETAL Directive

This directive permits the user to change the access level of a program already in a library.

```
SETAL(interval, al)
```

al = 0 to 7777 (octal).

Examples:

```
SETAL(HPA1+HPA9, 0)
```

```
SETAL(MALET, 3)
```

SETFL Directive

This directive permits the user to change the field length required to run the specified program already in the directory.

```
SETFL(interval,k)
```

k = 0 to 377777 (octal).

Examples:

```
SETFL(COPY,16000)
```

```
SETFL(MYPROG1+MYPROG4,40000)
```

SETFLO Directive

This directive permits the user to change the field length override bit of programs already in the library.

```
SETFLO(interval,k)
```

k = 0 or 1.

The default value is 0 (override is not permitted).

Examples:

```
SETFLO(AUDITZ,0)
```

```
SETFLO(MYPROG1+MYPROG4,1)
```

SKIPF Directive

This directive is the skip-forward and skip-backward directive.

```
SKIPF/SKIPB(N,Ifn,[F]L[P])
```

If N is numeric, it represents a number of records to be skipped, unless the third parameter is present, in which case N indicates a number of files. If N is alphanumeric, the function is skip-by-name and the direction implied by the directive is ignored. The file is searched in the forward direction; if the name N is not found, the file is rewound and the search continues in the forward direction. When found, the Ifn is positioned before the record of that name (N). If N is not found, the original position is maintained. The alphanumeric form is applicable to sequential files only. While in system mode, the file name SYSTEM means the running system.

TRANSFER Directive

This directive is used to create deadstart tapes.

```
TRANSFER(interval  $\perp$  n,lfn)
```

It moves records whose prefix table has been removed or is to be removed. A check is 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; for example, program name is 026, the directive is TRANSFER(\$026\$,SYSTEM).

Examples:

```
TRANSFER(*,SYSTEM)
```

```
TRANSFER(10,SYSTEM)
```

TRANS77 Directive

This directive is used to create deadstart tape records with prefix tables.

```
TRANS77(interval  $\perp$  n,lfn)
```

It performs the same functions that the TRANSFER directive does, with the exception that it does not remove prefix tables.

The following symbol is used to denote a comment field; whenever encountered in columns 1 and 2, EDITLIB treats it as a comment and does not check the statement.

```
*/
```

Example:

```
*/THIS ADDS A PROGRAM THAT WILL DO A/B*C
```

LIBRARY DIRECTORY ACCESS

PP Routines

To access the library directory, a PP routine first checks bit 59 of P.LIB. When bit 59 is set to 1, EDITLIB either 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.

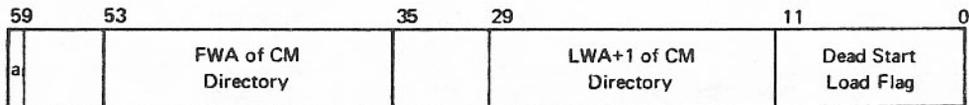
CP Routines

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

To access the directory, EDITLIB issues a monitor request to have PP routine MDI assigned to its control point. By passing MDI the appropriate code, MDI tries to change the directory. MDI sets 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 goes into the event stack until the bit is cleared. After MDI has completed its task, it adjusts the FWA and the LWA+1 pointers in P.LIB and sets bit 59 to 0.

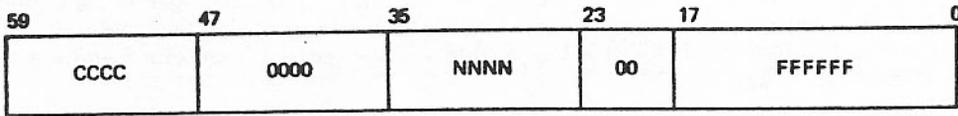
Library Directory Pointer Format



a Library change flag.

P.LIB The first word holds a library change flag (1 bit). This flag tells the PP and any CP program using the directory that EDITLIB either wants to change the directory or is changing the directory. Only when the flag in the control point areas is zero does EDITLIB proceed to change the directory.

PP Library Pointer Format



P.PPLIB FFFFFFFF 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 sets this word to zero and waits 1 second before changing the directory.

FILES

SYSTEM FILES

A copy of all the routines referenced by the directory resides on a permanent file called ZZZZZ04 after initial deadstart. If a program or library subsequently is to be included into the directory, it is placed in the permanent file called ZZZZZ03. EDITLIB never writes, modifies, or extends permanent file ZZZZZ04. The new version of replaced routines resides on the file ZZZZZ03. Since EDITLIB does not alter any record currently on the file ZZZZZ04 or the file ZZZZZ03, routines can be read from either file while EDITLIB is adding new records to ZZZZZ03.

USER FILES

The user is responsible for library contents. EDITLIB does not overwrite any record on the file. It only adds records at EOF/EOI. This means that the user has a file with unused records when library programs are deleted or replaced. It is up to the user to remove this dead space by building a new library using the old one as a source. No permanent file requests are issued by EDITLIB. The user's file must be attached, cataloged, and extended. 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.

FILE AND LIBRARY POSITIONING

EDITLIB rewinds all files except INPUT before executing any directives. After a random library is written, it is rewound. When a new sequential library is written, it is left positioned after the end-of-file. New directories are rewound when completed.

Index to user's libraries on a random file (six words):

<u>Word</u>	<u>Address</u>
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:

<u>Record</u>	<u>Contents</u>
1	This record is three words long. <ol style="list-style-type: none">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 end-of-file	Routines.

SYSTEM SECURITY

EDITLIB uses the reprieve function to protect the system files and its scratch files. Under any normal or abnormal termination condition, EDITLIB control is turned over to a program that returns or releases 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 performs the following tasks.

1. Requests field length for directory in CMR.
2. Sets directory change flag.
3. Gets length of directory.
4. Places CM address into LNT for CM-resident libraries.
5. Copies directory to control point.
6. Copies directory to CMR.
7. Checks CP activity flags at each control point.
8. Writes current directory file.
9. Attaches FNT to ZZZZZ06 to EDITLIB control point.
10. Monitors EDITLIB error flag after GO.
11. Catalogs ZZZZZ01 and ZZZZZ02 at control point 0 with an ID of SYSTEM.

A parameter list is provided in EDITLIB for MDI. This list contains MDI request code and all necessary information. MDI returns information to EDITLIB in this list. This list is the MDI INPUT/OUTPUT register.

An EDITLIB change to the CMR library directory is incompatible with INTERCOM operation. If INTERCOM is running, MDI issues the appropriate message to the operator when any of the following EDITLIB directives are encountered for the reasons noted.

<u>EDITLIB Directive</u>	<u>Reason for Incompatibility</u>
ADD	Library is CM resident.
DELETE	Library is CM resident.
REPLACE	Library is CM resident.
CHANGE	CM is involved in the residency change.
INCLUDE	Unconditionally incompatible.
LIBRARY	Library is new or CM is involved in a residency change.
MOVE	CM is involved in the residency change.
REMOVE	Unconditionally incompatible.
INCLUDEP	Unconditionally incompatible.

DIRECTORY/LIBRARY/PROGRAM LIMITS

The LNT allows a maximum of 27 libraries. A library can contain a maximum of 2047 programs, 2047 entry points, and 2047 internal references.

A particular program in the library can contain a maximum of 250 entry points and 500 external references. The PP program name table allows a maximum of 450 PP programs.

TABLE FORMATS

CMR DIRECTORY					
59	41	35	23	17	0
T.LIB	MDI Count Since Last Level 0 or 1 Deadstart				LWA+1 of LNT
Library Name Table (LNT) (5 Words per Entry)					
		LWA+1 of PP Program Bodies			LWA+1 of PPNT
PP Program Name Table (PPNT) (2 Words per Program)					
PP Program Bodies					
CM Resident Libraries					

Fields not being used contain zero.

Word 1:

Bits 59-18	Library name; can contain up to seven alphabetic and/or numeric characters of which the first must be alphabetic.
Bit 17 †	0 Residency addresses are valid. 1 Use disk addresses only.
Bit 16 †	Library is modified.
Bit 15 †	Library is new.
Bit 3	0 Library on file ZZZZ04. 1 Library on file ZZZZ03.
Bit 1 ††	0 Library is not extended memory resident. 1 Library is extended memory resident.
Bit 0 ††	0 Library is not CM resident. 1 Library is CM resident.

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.

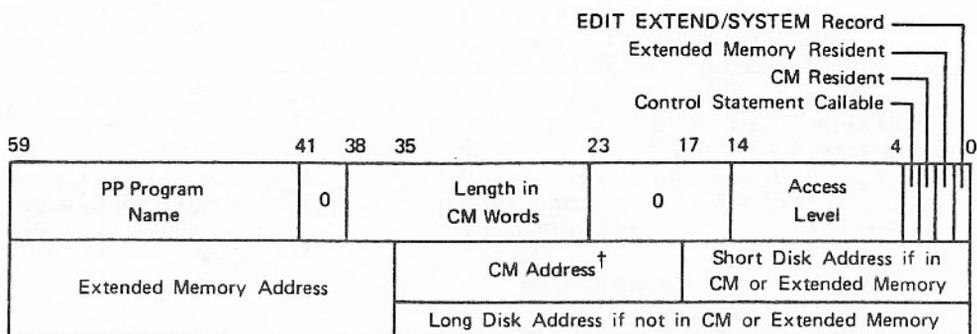
Word 5:

Address word for the program name table.

† 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 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 extended memory and the nature of the new directory.

†† If bits 0 and 1 are both zero, the directory is on disk.

PP PROGRAM NAME TABLE ENTRY



Word 1:

- Bits 14-4 Access level bits for INTERCOM.
- Bit 3 0 Program cannot be called from a control statement.
- 1 Program can be called from a control statement.
- Bit 2 0 Record/routine is not CM resident.
- 1 Record/routine is CM resident.
- Bit 1 0 Record/routine is not extended memory resident.
- 1 Record/routine is extended memory resident.
- Bit 0 0 Record is on (pfn) ZZZZ04.
- 1 Record is on (pfn) ZZZZ03.

Word 2:

If the routine is disk resident, there is a special disk address in bits 35 through 0 for use by PP resident.

If the routine is central memory or extended memory resident, bits 17 through 0 contain the short disk address, which is the PRU offset from the start of the chain. The central memory address is absolute.

Fields not being used contain zero.

[†]For extended memory resident overlay, this field contains the last consecutive access failure count. The residence is moved to RMS when the failure count exceeds an established threshold.

ERT ENTRY FORMAT

59	47	35	23	11	0
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 0 or by the fifth 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 are shown in this table.

PNT ENTRY FORMAT

59	47	35	23	11	0
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 five 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 program's entry in the program name table. Address is relative to the start of the program name table.

If the library contains any capsules, the program name table begins with the following capsule header.

CAPSULE HEADER FORMAT

59	47	35	17	0	
0	7 6 0 4 _g	K	0		
1	Group Name ₁			Number of Members ₁	
•					
•					
•					
•					
K-1	Group Name _{K-1}			Number of Members _{K-1}	
K	All Ones (Negative Zero)				

Word 0:

- Bits 59-48 7604₈ is the capsule header code. It and the next K words following it constitute a capsule header.
- Bits 47-36 K contains the number of words (octal) in the capsule header minus one.
- Bits 35-0 Zero.

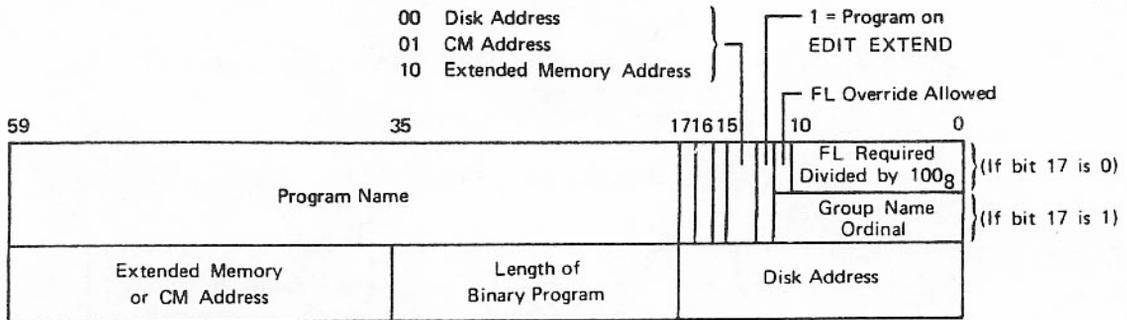
Words 1 through K-1 or 1 through K:

- Bits 59-18 Group name. Name of group of capsules. Display code, left justified, zero fill.
- Bits 17-0 Number of member capsules in the group.

Word K:

The capsule header must be an even number of words in length. If the number of group names in the capsule header is K-1, a word of all ones is added to the end of the capsule header.

PNT ENTRY FORMAT



Word 1:

- Bits 59-18 Program name in display code.
- Bit 17
 - 0 Normal program.
 - 1 Capsule.
- Bit 16
 - 0 Binary program.
 - 1 CCL procedure.

	Bit 15	Not used.
	Bits 14-13	00 Disk address.
		01 CM address.
		10 Extended memory address.
	Bit 12	1 Program is on (pfn) ZZZZZ03.
		0 Program is on (pfn) ZZZZZ04.
If bit 17 is 0	Bit 11	INTERCOM field length override bit.
		0 Do not override.
		1 May be overridden.
	Bits 10-0	Contain the amount of field length divided by 100 (octal) to execute this program. If not specified during an EDITLIB run, it is set to DFAULTFL/100 (octal) where DFAULTFL is a program parameter.
If bit 17 is 1	Bits 11-0	Group name ordinal. Specifies the ordinal of the group name to which this capsule belongs. The ordinal is derived from the order in which the group names appear in the capsule header.

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

59	17	0
		Program Number
Entry Point		
Word of Zeros		
External Reference		
Word of Zeros		

The entry point and external reference list has an entry for each program that is a part of the library. This list is maintained by EDITLIB. The entry point and external references for each routine are in alphabetical order and are left-justified with zero fill.

The index to this record, when the library is not part of the running system, is in the index record for the random file. 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 is placed following the entry point name table record. By using the index to the entry point table, it is possible to access the external reference list. A CIO read skip request is used with the index to the entry point name table. Then a read with a zero index is used to obtain the entry point and external reference list.

Figure 8-2 shows library table interfaces.

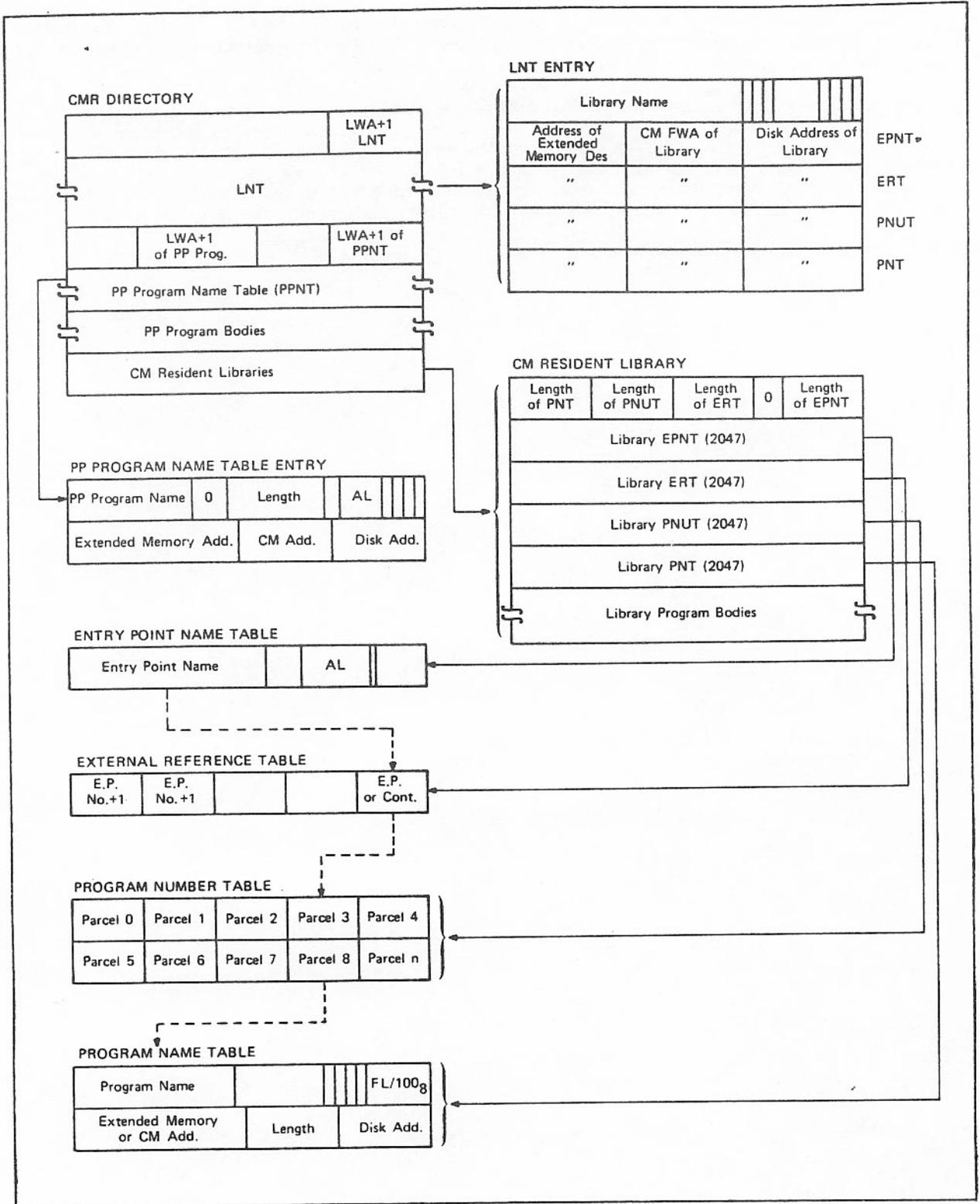


Figure 8-2. Library Table Interfaces

EXAMPLES

Deadstart creation from the running system with changes:

```
READY(NEWSYS,NEW)
TRANSFER(IPL+OSB,SYSTEM)
TRANSFER(CED+MDR,SYSTEM)
TRANSFER(CMR,NEWCMR)
SKIPF(1,SYSTEM)
TRANSFER(COM+LFP,SYSTEM)
TRANSFER(OSY,SYSTEM)
TRANSFER(OSZ,SYSTEM)
TRANSFER(OMT,SYSTEM)
TRANSFER(OSJ,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,LIB)
REPLACE(1SP,LGO,AL=0,CM)
LISTLIB(*,SYSTEM)
COMPLETE.
LISTLIB(*,SYSTEM,NUCLEUS)
LISTLIB(*,SYSTEM,FORTRAN)
ENDRUN.
```

Deadstart creation from the running system with no changes:

```
READY(NEWSYS)
REWIND(NEWSYS/SYSTEM)
TRANS77(IPL+OSB,SYSTEM)
TRANSFER(CED+MDR,SYSTEM)
TRANSFER(1,SYSTEM)CMR
TRANSFER(COM+LFP,SYSTEM)
TRANSFER(OSY,SYSTEM)
TRANSFER(OSZ,SYSTEM)
TRANSFER(OMT,SYSTEM)
TRANSFER(OSJ,SYSTEM)
TRANSFER(*,SYSTEM)
INCLUDEP(SYSTEM)
INCLUDE(NUCLEUS,SYSTEM,CM)
INCLUDE(SYSOVL,SYSTEM,CM)
INCLUDE(BAMLIB,SYSTEM,DS)
INCLUDE(SYMLIB,SYSTEM,DS)
INCLUDE(FORTRAN,SYSTEM,DS)
.
.   INCLUDE directive for all other libraries that are to
.   be a part of the new system.
.
INCLUDE(SYSMISC,SYSTEM)
.
.   Directives that modify any of the libraries
.   just included into the new system, or directives
.   that create new libraries.
.
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,LOG,AL=0)
LIBRARY(NUCLEUS,OLD)
DELETE(HOHUM)
FINISH.
COMPLETE.
ENDRUN.
```

Installation system bulletins can be created and stored for user access. Bulletins are created and updated with the system utility BULLUP, and are accessed by the user with the control statement SYSBULL. BULLUP has the capability to create a system bulletin file if none exists, update an existing bulletin file, and reduce the size of an existing bulletin file. BULLUP creates and updates the bulletin file using information from data statements. Updating the bulletin file consists of adding new bulletins to the file and/or updating the information in an existing bulletin. When BULLUP creates or updates a bulletin file, it also builds an index so that bulletins can be accessed by name. Reducing the size of the bulletin file consists of reading the active information from the bulletin file and writing it to a new file.

BULLUP STATEMENT

The control statement for calling the BULLUP utility is as follows:

BULLUP.

or

BULLUP(input,output)

Default input and output files are INPUT and OUTPUT. Alternate files can be used by specifying alternate file names. The SYSBULL control statement does not have this capability; output from SYSBULL always goes to the file OUTPUT.

BULLUP DATA STATEMENTS

BULLUP accepts data statements in the following format.

Name statement	}	First bulletin
Contents statement		
.		
.	}	Second bulletin
Contents statement		
Name statement		
Contents statement	}	
.		
.		
7/8/9		

NAME STATEMENTS

Name statements have the following format.

Column 1 2 3 through 9 36 through 80

* \$ bulname bulletin description

bulname Bulletin name consisting of one through seven letters and digits; it must begin in column 3. The first blank terminates the field. The reserved names INDEX and ALL are not allowed. Up to 100 names can be specified for the system bulletin file.

bulletin description Bulletin description consisting of up to 45 characters (generally, a brief description of the bulletin). The description, which is listed with the index, is the second line of the bulletin. If the bulletin description is *EMPTY*, SYSBULL does not pass the bulletin to the file OUTPUT.

CONTENTS STATEMENTS

Contents statements are any statements that do not have * \$ in columns 1 and 2. Column 1 is used either for printer carriage control or as a continuation indicator. If column 1 is not a comma, it is considered a printer carriage control character, and columns 2 through 80 are printed in print line positions 1 through 79. A statement with a comma in column 1 is considered to be a continuation of the preceding statement, and columns 2 through 58 are printed in print line positions 80 through 136. Multiple continuation statements are allowed.

PROCESSING DATA STATEMENTS

The most significant difference between a creation run and an update run is the fact that a bulletin file does not exist before a creation run. Data statements for a creation run are processed in the same way as for an update run.

A name statement is read and validated. If an error occurs in the name statement, succeeding bulletin contents statements are skipped up to the next name statement, and a diagnostic is printed. If the name statement is valid, the bulletin file index is searched for a bulname match. If none is found, the name is added to the end of the index, and the index is changed to reflect the new bulletin. If a matching name is found, the index is changed to reflect the update to this bulletin. BULLUP then fills a buffer with the name statement, contents statement, and other information, including the current date and whether the bulletin was created or updated. The buffer is written to the end of the bulletin file when it becomes full and when a new name statement is encountered. After all data statements have been read, the bulletin file is closed and the bulletin file index is appended to the end of the file.

CREATING A SYSTEM BULLETIN FILE

BULLUP creates a bulletin with the name specified in the bulname position of the name card. An index entry for each bulletin is stored in the bulletin file index. The following control statements are required to create a system bulletin file.

```
job statement
REQUEST(ZZZZZIN,PF)
BULLUP.
CATALOG(ZZZZZIN,SYSTEMBULLETINFILE,ID=SYSBULL,...)
7/8/9
Name statement
Contents statement
.
.
.
6/7/8/9
```

Only the permanent file name and the ID are defined in SYSBULL. Both can be changed; however, if they are changed, the attach function in SYSBULL must also be changed. A turnkey (TK=) and/or a read (RD=) password for the system bulletin file may be added. If so, the attach function in SYSBULL also must be changed to specify the addition. Extend, modify, and control passwords can be specified if desired, without any changes.

UPDATING A SYSTEM BULLETIN FILE

BULLUP updates a bulletin file by adding new bulletins and/or updating information in existing bulletins. When a new bulletin is added to the file, a new index entry is added to the file index. When the information in an existing bulletin is updated, the file index is changed to reflect the fact that the bulletin has been updated.

The following control statements are required to update a system bulletin file.

```
job statement
ATTACH(ZZZZZIN,SYSTEMBULLETINFILE,ID=SYSBULL,...)
BULLUP.
EXTEND(ZZZZZIN)
7/8/9
Name statement
Contents statement
.
.
.
6/7/8/9
```

Permanent file considerations for an update run are the same as for a creation run. Data statement requirements are the same as for creating a new bulletin. The old contents of the bulletin are not carried forward to the updated bulletin. If desired, the contents must be re-entered via contents statements. Information is deleted from an existing bulletin by using a name statement and omitting succeeding contents statements. Since disk space for this bulletin still exists, even though it contains no information, a single contents statement stating that the bulletin contains no information is suggested.

REDUCING A SYSTEM BULLETIN FILE

An index entry and its associated bulletin become inactive when that bulletin is updated. BULLUP does not rewrite in place, since the new information in an updated bulletin may exceed the old information. To maintain minimum disk space for the system bulletin file, BULLUP copies the active bulletins and the current index to a new file, which can then be cataloged.

The following control statements are required to reduce the size of the system bulletin file.

```
job statement
REQUEST(ZZZZZOU,PF)
ATTACH(ZZZZZIN,SYSTEMBULLETINFILE,ID=SYSBULL,...)
BULLUP.
CATALOG(ZZZZZOU,SYSTEMBULLETINFILE,ID=SYSBULL,...)
PURGE(ZZZZZIN)
6/7/8/9
```

ZZZZZOU is the new bulletin file. It must be requested for permanent file device residence before BULLUP is called and it must be an empty file. ZZZZZIN is the current bulletin file with active and inactive information. Data statements should not be used when the active bulletins and the current index are to be copied to a new file. If a system bulletin file backup is desired, PURGE(ZZZZZIN) need not be used.

NOS/BE-INTERCOM CONSIDERATIONS

Under INTERCOM, a call to SYSBULL is initiated when a user logs in. SYSBULL attempts to find the specific bulletin named LOGIN (if SUP was not specified during login procedures) or SUP. If SYSBULL finds the appropriate bulletin, information is immediately displayed at the terminal. Bulletins named LOGIN and SUP are not mandatory on the system bulletin file. If SYSBULL does not find the system bulletin file or the specific bulletin LOGIN or SUP, processing continues. However, installations with INTERCOM users can keep their users informed via these named bulletins since INTERCOM automatically initiates an attempt to find them.

A call to SYSBULL is initiated automatically when a batch job enters the system. SYSBULL attempts to find the specific bulletin named BATCH. If found, the bulletin information is sent to OUTPUT immediately, and processing continues. The bulletin named BATCH is not mandatory on the system bulletin file. If SYSBULL does not find the system bulletin file or the specific bulletin BATCH, processing continues. However, installations can keep their users informed via BATCH, since the system automatically attempts to find the BATCH bulletin.

INTRODUCTION

The purpose of segmenting CMR is to move operating system central processor code from central memory to extended memory in systems having extended memory. The code is loaded back to CM and executed as needed. This reduces the amount of CM required for the operating system and releases that CM to user jobs running at control points.

Because of CMR segmentation, the system CP code executes in two different configurations, depending on extended memory availability. When extended memory is not available, the system CP code is all located in CM. When extended memory is available, most of the CP code is extended memory resident and loaded as needed in overlay areas. The CMR loader, LDCMR, handles the segment relocation and the creation of the extended memory segment library. LDCMR is called by deadstart and can also run as a utility. If P.AREA is changed, LDCMR must be reassembled.

The default library used by LDCMR is CMRLIB. An alternate library can be specified in T.ENTRY+W.SGLIB. This library should contain all CMR segments and routines used by these segments. Two other programs are used by LDCMR. Program LDCMR= is loaded by CYBER Loader to reload LDCMR; this program must be a relocatable binary. Program CMRDIR describes the system to be generated and contains the trace and stack buffer lengths, as well as segment names needed for a load. CMRDIR also designates areas where the segments are to be loaded.

LDCMR uses direct access extended memory by requesting control point 0 extended memory FL. Therefore, the length of the direct access area changes if LDCMR is used to add or replace extended memory segment libraries.

LDCMR catalogs a file called ZZZZCMR with ID=SYSTEM, TK=ECSSYSTEM, MD=XNOX, and EX=XNOX. ZZZZCMR has two records: the first record contains a local area table followed by a segment table; the second record contains all the completed segments. On a level 1 deadstart, LDCMR uses the lowest cycle of ZZZZCMR to rebuild an extended memory system if the date-time stamp matches the one in T.ENTRY+W.DTIME.

If the library directory is moved, LDCMR will attach ZZZZZ02 to lock out EDITLIB operations. On completion of the move, ZZZZZ23, ZZZZZ01, and ZZZZZ02 are modified to reflect the current directory.

If S.SYSEDIT is set in P.LIB, LDCMR does not need operator permission to add a system to extended memory or to bootstrap an extended memory system. The bit is cleared by a successful bootstrap.

LDCMR CONTROL STATEMENT

The LDCMR control statement has the following format.

LDCMR(p₁,p₂,...,p_n)

Parameters are order independent.

<u>Pi</u>	<u>Description</u>
F=lfn ₁	If PO=P is not selected, lfn ₁ contains CMR segment relocatable binaries and/or CMRDIR. When segment binaries needed to build a CMR are not on lfn ₁ , they are taken from library CMRLIB. If two or more copies of the same segment exist on lfn ₁ , the last one encountered is used. If a deck called CMR is present in the input, all information needed by LDCMR is taken from this CMR. If P.AREA is changed, LDCMR must be reassembled. If PO=P is selected, lfn ₁ contains the same information as ZZZZCMR (area table, segment table, and CMR without the low core tables).
F	Use file LGO for input file.
F=0	No load. The B, L, and LO parameters are ignored. PO options A and T are ignored.
F omitted	No input file.
B=lfn ₂	LDCMR writes the system image to lfn ₂ . This file can be used later to create a new system (refer to PO options).
B	LDCMR writes the system image to file CMR.
B=0 or B omitted	No system image file is generated.
L=lfn ₃	All output goes to lfn ₃ .
L or L omitted	All output goes to file OUTPUT.
LO=option	The letter or letters selected specify the contents of the map. Any of the options S, B, E, X, and L can be combined; for example, LDCMR(LO=SBL). The N option can be combined only with the L option; for example, LDCMR(LO=LN). If none of the CYBER Loader map options (N, S, B, E, or X) are selected or an error is detected in processing the parameter, the current job default is used.

<u>Option</u>	<u>Map Contents</u>
S	CYBER Loader statistics.
B	CYBER Loader block map.
E	CYBER Loader entry point map.
X	CYBER Loader entry point cross references.
L	LDCMR map.
N	No CYBER Loader maps are generated.

LO Equivalent to LO=SBEXL.

LO omitted Equivalent to LO=L.

Pi

Description

PO=option

The letter or letters selected specify actions to be performed; order of execution is IPTRDCAB.

Option

Action

- A Add a new CMR to extended memory. If the old system is not running, LDCMR bootstraps the old system first.
- B Bootstrap alternate system and move library if necessary. This option requires operator permission.
- C Catalog a new cycle of ZZZZCMR with the lowest cycle.
- D Delete new system from extended memory. If the new system is running, LDCMR bootstraps the old system before releasing extended memory. Option D is completed before option A, so PO=DA has the same effect as PO=A.
- I Initialize system. This option is used only at deadstart time to examine deadstart options and perform the indicated action. This option overrides all other PO options.
- P Input file specified by F parameter has been preprocessed by LDCMR and is in the same format as the ZZZZCMR file.
- R Replace old system in extended memory with the new system. If the old system is running, LDCMR bootstraps the new system before releasing extended memory space taken by the old system. The bootstrap moves the extended memory copy.
- T Trace all GOTOS. Do not change a GOTO to an entry point in CM resident to a direct jump.

PO Options ABC are used.

PO omitted No options are used.

LDCMR FILES

<u>File Name</u>	<u>Cataloged</u>	<u>Use</u>
ZZZZZ03	No	File from which system library binaries are read. CMR library directory may be on this file.
ZZZZZ04	No	Same as ZZZZZ03.
ZZZZZL0	No	CYBER Loader input file containing CM resident binaries.
ZZZZZL1	No	CYBER Loader input file containing the monitor area binaries.
ZZZZZL2	No	CYBER Loader input file containing the user area binaries.

<u>File Name</u>	<u>Cataloged</u>	<u>Use</u>
ZZZZZL3 ZZZZZL4 ZZZZZL5 ZZZZZL6	No	Reserved.
ZZZZZL7	No	File containing LDCMR= and a dummy (0,0) overlay. This forces CYBER Loader to load at the correct address and return to LDCMR when loading completes.
ZZZZZL8	No	CYBER Loader output file where all the loaded segments are written.
ZZZZZL9	No	LDCMR scratch file used to save its tables while CYBER Loader is running.
ZZZZZ01	Yes	Modify reset file in case directory moved.
ZZZZZ02	Yes	Interlock file to lock EDITLIB out. Modify restore file if directory is moved.
ZZZZZ23	Yes	Modify directory file if LDCMR has moved T.LIB.
ZZZZCMR	Yes	File on which the area table, segment table, and segments are saved so that LDCMR does not need to reload them.

The files that are cataloged have the following parameter values set.

<u>File Name</u>	<u>ID</u>	<u>RP</u>	<u>TK</u>	<u>CN</u>	<u>MD</u>	<u>EX</u>
ZZZZZ01	SYSTEM	999	SSSSSSSS	XNOX	XNOX	XNOX
ZZZZZ02	SYSTEM	999	SSSSSSST	---	---	XNOX
ZZZZZ23	SYSTEM	999	DIRECTORY	XNOX	XNOX	---
ZZZZCMR	SYSTEM	999	ECSSYSTEM	---	XNOX	XNOX

SYSTEM SECURITY

LDCMR uses the reprieve function to protect the system files and its scratch files. Under normal or abnormal termination, LDCMR returns or releases all LDCMR files.

RESERVED NAMES

LDCMR uses the following names which should not be used by the operating system:

DUMMYnn

nn A number from 00 to 77g.

The names DUMMYnn are entry point names which must not be defined in any segment.

LDCMR also uses the common block name of CMRES to relocate the load address for each area. At the present time, common blocks are not permitted within segments.

LDCMR INTERLOCK

When LDCMR changes the extended memory system, LDC is called to assign the extended memory. LDC assigns the block of extended memory under control point zero in the direct access area.

LDC uses a bit in the control point area to keep CPMTR from changing the extended memory FL and extended memory RA. This prevents the scheduler from changing LDCMR's exchange package while LDCMR is running. This bit is cleared when LDCMR signals LDC that it has completed the change. The method used to set/clear this bit is as follows:

1. Set/clear bit in control point area.
2. Wait before reading the word.
3. If MTR has changed the word, try again; otherwise, continue.

This interlock is also used to prevent two LDCMRs from bootstrapping or changing the extended memory system at the same time.

LDC

This PP overlay is used to help LDCMR. It can perform the following tasks.

- Read absolute CM.
- Assign or release control point 0 extended memory from the direct access area.
- Catalog permanent files with ID=SYSTEM at control point 0.
- Reload LDCMR after CYBER Loader completes loading each stage.
- Bootstrap a new or old extended memory system.
- Move the directory.
- Issue all operator GO/DROP messages.

A parameter list is provided in LDCMR for LDC. This list contains the LDC function code and all necessary information. LDC returns information and sets the completion bit within this list.

For an explanation of the directory move process, refer to EDITLIB in section 8. LDC uses the same procedure.

EXAMPLES

To add a new system to extended memory and bootstrap the new system, use the following LDCMR statement. The new binaries are on file X. A map will also be generated.

```
LDCMR (PO=AB,F=X)
```

To replace the old system with the new and release the extended memory space, use the following statement.

```
LDCMR (F=0,PO=R)
```

To get a load map of the deadstart system, use the following statement.

LDCMR.

To save the preloaded system for later use, use the following sequence of statements.

**REQUEST(X,PF)
LDCMR(B=X)
CATALOG(X,PERM,ID=USERA)**

To load and bootstrap a new system from the preloaded file, use the following statements.

**ATTACH(X,PERM,ID=USERA)
LDCMR(F=X,PO=PAB)**

This section describes the deadstart dump analyzer and the system dynamic dump.

DEADSTART DUMP ANALYZER

The deadstart dump analyzer processes an express deadstart dump (refer to the NOS/BE Operator's Guide for procedures to take an express deadstart dump) and produces as output the dump in readable format. Analysis directives allow the user to specify portions of CM, PPs/PPUs, and/or extended memory to be reformatted and printed. The tape file is in stranger format.

DSDUMP CONTROL STATEMENT

The format of the analyzer control statement, DSDUMP, is

DSDUMP(I=lf_{n1},L=lf_{n2},T=lf_{n3},C=lf_{n4},ZB=lf_{n5},EM,BC)

- | | |
|--------------------|--|
| I=lf _{n1} | File containing analysis directives. Directives are read starting at the current position of the file. If I=0 is specified, the directives A, B, IT,† CM, SR, RBTC, M, PP, H, JDT, XP, and WO are assumed. Default is INPUT. |
| L=lf _{n2} | File to which output is written. Default is OUTPUT. |
| T=lf _{n3} | File containing express deadstart dump (EDD). This file cannot be processed in its original format; the analyzer must reformat it. To save time in subsequent runs using the same file, the resultant file (C=lf _{n4}) can be saved. If the T=lf _{n3} parameter is omitted, it is assumed that the EDD file has already been reformatted. The reformatting phase is skipped and the file specified by C=lf _{n4} is used as input. If the keyword T is specified alone, lf _{n3} is assumed to be TAPE. |
| C=lf _{n4} | Reformatted dump file (T=lf _{n3}). Default is CRASH. |
| Z | Directives are on the DSDUMP control statement. If I=lf _n is also specified, directives on the control statement take precedence. Directives can start after the first right parenthesis or period on the control statement; they terminate at the next period encountered. Directives can be continued; if a period is not found on a control statement and column 80 is not blank, the next statement is used as input starting in column 1. If an end-of-record is encountered, reading of directives ends and all legal directives that have been read are processed. |
| B=lf _{n5} | Maintenance register data found in either the maintenance registers or the Error Processing Interface Buffers (EPIB) are to be written in CERFILE format on lf _{n5} for processing by NORMS/HPA. |
| EM | The EM (Extended Memory) record will be copied to disk only if EM is specified. |
| BC | The controlware record(s) will be copied to disk only if BC is specified. |

†If INTERCOM is in the system.

ANALYSIS DIRECTIVES

Analysis directives can appear on an input file (specified by the I=lf_{n1} parameter) or on the DSDUMP control statement (specified by the Z parameter). Directives are separated by commas and terminated by a period or an end-of-record. In the following description, directives are separated into three groups: display directives, dump directives, and special dump directives (such as those used for pattern matching).

After the reformatted dump file is created or opened by DSDUMP, a page of information is printed on the output file. This information includes the system label from words 32 through 26 in low core, the date and time of the dump from words 31 through 30 in low core, and the data cards that were read by DSDUMP.

Display Directives

The display directives generate tables that are expanded forms of console displays.

<u>Directive</u>	<u>Display</u>	<u>Description of Table Generated</u>
A	A	Contains the most recent system dayfile and dayfiles for all active control points.
B	B	Indicates activity at each control point, channel and PP allocation, JDT address, and so on.
H	F, H	Contains the file name table. The octal representations of the FNT entries are printed but supplements are not.
JDT	R	Contains the job descriptor table showing all jobs in the system.
M	M	Contains the current status of all PPs.
TAPES	P	Contains the entire octal presentation of the TAPES table for every drive in the EST.

Dump Directives

The dump directives generate dumps of specified area. Dumps are in octal, unless otherwise noted in the description.

<u>Directive</u>	<u>Description</u>
BC	Dumps (in hexadecimal) all buffer controllers that were dumped to tape by EDD. Dumps are identified by channel number. Octal dumps can be obtained by specifying BC=OCT.
CMR	Interprets low-core pointer area. The first 100 words of CMR are formatted and labelled by word and field.

<u>Directive</u>	<u>Description</u>
CPA	Interprets active control point areas. The exchange package and control card buffer is formatted in readable format. All other fields are formatted and labelled by word and field.
CPA=cp _i /.../cp _n	Interprets control point area of specified control points. cp _i Specifies control point or range of control points (in the form cp _i - cp _n) for which the control point is to be dumped. For example, CPA = 2/5 - 7/10 dumps control points 2, 5, 6, 7, and 10.
Cy CX††	Dumps central memory from address 0 to the first word address of CM resident library tables. Central Memory is printed four words per line.†
Cy=memstring ₁ /.../memstring _n CX=memstring ₁ /.../memstring _n	Dumps specified CM and optionally prints a message at the beginning of the dump. y Specifies number of octal words printed per line; 0 through 4 or M. The values 0, 4, and M specify four words per line. memstring _i Specifies either the range of CM to be dumped for the range plus a delimited character string. The range of memory to be dumped is specified in the form fwa-lwa where fwa is the first word address of the dump and lwa is the last word address of the dump. A message of up to 50 characters can immediately follow the range. It is printed at the beginning of the dump. The character string must be delimited by one of the special characters + * = ; - \$ or %. Both delimiters must be the same; the delimiter cannot appear in the string.

†In CM and extended memory dumps, lines of all zeros are omitted. The next nonzero line contains the symbol Γ after the address. Also omitted are duplicate lines and lines which are the same, word for word, as the last word of the previous line. The next nonduplicate line contains an equal sign after the address.

††CX produces a hexadecimal dump with ASCII interpretation. With computers other than CYBER 180-class mainframes, the 60-bit word is right-justified in a 64-bit field.

Directive

Description

Examples:

C2=200-1000
C3=3700-4700*DUMP NO. 1*
CM=200-600\$FIRST
DUMP\$/3000-
4000\$SECOND DUMP\$

The values of fwa and lwa are assumed to be octal, unless followed by D specifying decimal.

If lwa is less than fwa, lwa is reset to fwa + 100g. If lwa is greater than the memory size, it is reset to the memory size. Memory is read in blocks of 100g and, therefore, some shifting of fwa and lwa may occur. If shifting causes lwa to be greater than memory size, the nonexistent words are printed with the display code equivalent of 10 asterisks.

Ey=memstring₁/.../memstring_n

Dumps specified extended memory and optionally prints a message at the beginning of the dump.† The parameters for this directive are identical to those for the Cy=memstring directive. Refer to that directive for a description of parameters and their usage.

PJT

Dumps the peripheral job table area.

PP
PPX††

Dumps all PPs in a format of 16 bytes per line.

PP=pp₁/.../pp_n
PPX=pp₁/.../pp_n††

Dumps specified PPs or PPU's in a format of 16 bytes per line.

PP_i Specifies PP or PPU to be dumped or a range of PPs or PPU's (in the form PP_m-PP_n) to be dumped. For example,

PP=3-7/10/17

dumps PPs 3, 4, 5, 6, 7, 10, and 17. To specify a first level PPU, the number is preceded by F. For example,

†In CM and extended memory dumps, lines of all zeros are omitted. The next nonzero line contains the symbol □ after the address. Also omitted are duplicate lines and lines which are the same, word for word, as the last word of the previous line. The next nonduplicate line contains an equal sign after the address.

††The PPX directives produce a hexadecimal dump with ASCII interpretation. With computers other than CYBER 180-class mainframes, the 12-bit word is right-justified in a 16-bit field.

Directive

Description

PP=F5

dumps PPU 5. PP numbers are assumed to be decimal; PPU numbers are assumed to be octal. Either can have a post-radix of B or D.

PPs/PPUs are dumped and printed in the order they were specified. When PPs other than PP0 or PP1 are printed, three lines of PP resident entry point information are printed in addition. The first line is the entry point address. The second line is the entry point name. The third line is the contents of the memory cell at entry point+1, which reflects the return address of the last return jump to this entry point. No attempt is made to verify that the return address is valid.

For CYBER 180-class mainframes, after DSDUMP prints out each PP memory, it prints out (in both hexadecimal and octal) the address and contents of each word that has a nonzero value in the upper four bits unless the PP was assigned to NOS/VE.

If a specified PP/PPU is not on the dump file, an error message is generated and the analyzer continues. PP memory cells destroyed by the deadstart process are printed as asterisks.

RBR

Dumps all RBRs.

RBR=n

Dumps record block reservation table.

n Number of RBRs to be dumped.

RBTC

Validates RBT chains and prints chains and RBT area in a format of two words per line.

UFL

Dumps all user field lengths.

UFL=cp₁/.../cp_n

Dumps user field length at specified control point.

cp_i Specifies control point or range of control points (in the form cp_m-cp_n) for which field length is to be dumped. For example,

UFL=2/5-7/10

dumps control points 2, 5, 6, 7, and 10.

XP

Dumps the system exchange package, including the status control register. In a deadstart dump from a CYBER 180-class mainframe, the system exchange package is the processor exchange package from the hardware exchange package register file supplied by EDD.

Special Dump Directives

<u>Directive</u>	<u>Description</u>
	system exchange package is the processor exchange package from the hardware exchange package register file supplied by EDD.
IT	If INTERCOM was running at the time of the deadstart dump, this directive dumps the INTERCOM area, tables, buffers, and so on. The dump is printed in the format of four words per line.
PAT=pattern ₁ /.../pattern _n	Searches CM, PP memory, and extended memory (if it is available) for a word that matches the specified pattern. If a match is found, the type of memory and address are printed. pattern _i Specifies the word to be matched. Patterns are assumed to be octal unless they are followed by D specifying decimal. Up to 50 patterns can be tested.
SEG	If the system was using extended memory at the time of the deadstart dump, this directive dumps the segment name and entry point tables in addition to the monitor and user mode segment names. The CM resident, monitor, and user areas are also dumped in the format of four words per line.
SEQ	Dumps the sequencer tables if they are present.
PXP†	Dumps the processor exchange package in hexadecimal.
MR†	Dumps all maintenance registers in hexadecimal.
RF†	Dumps the processor register file in hexadecimal.
CS†	Dumps the processor control store (microcode) in hexadecimal.
MC†	Dumps both the processor register file and the processor control store (microcode) in hexadecimal.
PT†	Dumps the contents of the page table in hexadecimal.
EI†	Dumps the environment interface in hexadecimal.
EPIB†	Dumps the Error Processing Interface Block and its buffers.

†CYBER 180-class mainframes only.

<u>Directive</u>	<u>Description</u>
SR	Dumps stack requests. Stack requests are listed in the following order: <ol style="list-style-type: none"> 1. By DST/DAT. If all words of the DST/DAT are zero, it is not printed. 2. By DOT. If the DOT is zero, it is not printed. 3. Empty chain, if it exists. 4. Any stack requests that have not yet appeared in the preceding lists.
W0	Searches CM, PP memory, and extended memory (if it is available) for a word that matches word 0 of central memory. If word 0 is zero, no match is attempted.

Default Directives

If no directives were found on the DSDUMP control statement or input file, or if I=0 is specified, the following directives are assumed:

A, B, IT,† CM, SR, RBTC, M, PP, H, JDT, XP, W0, MR††

DSDUMP MESSAGES

The deadstart dump analyzer issues informative messages indicating the current status of the analysis. The dayfile message

DSDUMP -- ANALYZING CRASH FILE.

is issued after the dump tape is read or as the header of the CRASH file (C=lf_n) is being read. When all directives have been processed, the analyzer issues the dayfile message

DSDUMP -- NORMAL TERMINATION.

If INTERCOM analysis is requested and INTERCOM was running when the deadstart dump was produced, the message

INTRAN -- BEGINNING INTERCOM ANALYSIS.

is issued when INTERCOM analysis begins. When INTERCOM analysis is concluded, the message

INTRAN -- END INTERCOM ANALYSIS.

is issued.

†If INTERCOM is in the system.

††MR applies to CYBER 180-class mainframes only.

If one or more errors occur in the directive, the dayfile message

DSDUMP -- ERROR(S) IN DIRECTIVES.

is issued. Except for the message, errors are ignored and processing continues.

If DSDUMP detects an SCR hardware error, it issues the following dayfile message.

DSDUMP -- SCR INDICATES HARDWARE ERROR.

The dump analyzer uses CMR pointers to find many of the tables it formats. If a pointer appears to be incorrect, a message to that effect is printed in the analysis output. If the CMR tables are overwritten, the message

DSDUMP -- LOW CORE WAS OVERWRITTEN.

is issued. If the values in low core are no longer valid, the analyzer uses the following default values.

CM size	777700g
Number of PPs	20
Number of control points	17g

SYSTEM DYNAMIC DUMP

The system dynamic dump facility takes an on-line dump of PP or central memory and lists the dump on printed output. This facility is currently used by INTERCOM and EXPORT to document the PP and CM contents at the time that a detected error forces either an INTERCOM restart or an EXPORT abort.

INTERFACE

All dumps are written, directly by stack request, to the system file ZZZZZDD (the system dynamic dump file). ZZZZZDD is initialized and recovered by the operating system.

ZZZZZDD is cataloged and purged as a queue file with special disposition code 66g. When not in use, the file is unlocked at control point zero. During a dynamic dump, the file is locked by the program requesting the dump. While it is being listed, the file is attached to a control point. The file is associated with a file supplement table in the FNT which includes special link ID 602g. Current RMS position is kept in the supplement.

PP memory dumps are written directly from the PP using macros defined on system text SDDTEXT to establish the interface for access to the file. The PP service program 1DD performs CM dumps as well as the cataloging, extending, and purging of the ZZZZZDD file. Text SDDTEXT includes macros that provide a standard interface to call 1DD.

DUMP FILE FORMAT

Each memory dump request is written as a separate record on the system dynamic dump file.

PP dumps are unlabeled and are written as single records with end-of-record level 0.

CM dumps consist of a label followed by a contiguous block of CM image. The label and the CM image dump are separate records, each with end-of-record level 15g. The label format is as follows.

	59									0	
		C	M	(x	x	x	x	x	,	0
		y	y	y	y	y)	z	z	z	1
T.CLK			h	h	.	m	m	.	s	s	2
T.DATE			m	m	.	d	d	.	y	y	3
T.SLAB2		System Label									4
T.SLAB3											5
T.SLAB4											6
T.SLAB5											7
T.SLAB6											10

All entries are in display code. Words 0 and 1 of the label define the CM address bounds of the dump that follows. Words 2 through 10 are taken from CMR at the time of the dump.

<u>Field</u>	<u>Contents</u>
xxxxxx	First word address of the dump.
yyyyyy	Last word address of the dump.
zzz	If zzz is spaces, xxxxxx and yyyyyy are absolute CM addresses. If zzz is -nn, xxxxxx and yyyyyy are addresses relative to control point nn.

At the end of the dynamic dump, a trailer label is written in the following format.

	59									0	
		E	N	D	O	F	F	I	L	0	
		E		N	O	n	n	n	n	n	1
T.CLK			h	h	.	m	m	.	s	s	2
T.DATE			m	m	.	d	d	.	y	y	3
T.SLAB2		System Label									4
T.SLAB3											5
T.SLAB4											6
T.SLAB5											7
T.SLAB6											10

All entries are in display code. Word 1 contains the dump number referenced by LISTCID. Words 2 through 10 correspond to the same words in the CM dump label.

Each dynamic dump ends with an end-of-file. Multiple files may exist on a dynamic dump file. The F parameter on the LISTCID control statement allows the user to select which of these multiple files to read.

LISTING DUMP FILES

LISTCID (list core-image dump) is a system utility that lists output of all or selected portions of a core-image dump file depending on specified parameters. It can be called by a control statement in a batch job or from the central site console. When LISTCID is initiated from a batch job, it requests that the central site operator enter a LOCKIN or DROP command to provide security for access to absolute system dumps.

The LISTCID control statement has the following format.

LISTCID(p_1, p_2, \dots, p_n)

where p can be a keyword or a keyword equated to an option. Parameters can be specified in any order.

<u>P_i</u>	<u>Description</u>
I=lf n_1	Core-image input is taken from file lf n_1 .
I or I omitted	Core-image input is taken from the system dynamic dump file, ZZZZDD.
L=lf n_2	Listable output is placed on file lf n_2 . Disposition of lf n_2 must be provided by the calling job.
L or L omitted	Listable output is placed on file OUTPUT and in the output queue at end-of-job (unless otherwise disposed by the calling job).
C	List all CM dumps encountered in byte format.†
C=addr	List CM from address 0 to address addr in byte format.†
C=addr $_1$ -addr $_2$	List CM from address addr $_1$ to address addr $_2$ in byte format.†
C omitted	No specific CM dump.
CD	List all CM dumps encountered in display code format.†
CD=addr	List CM from address 0 to address addr in display code format.†
CD=addr $_1$ -addr $_2$	List CM from address addr $_1$ to address addr $_2$ in display code format.†
CD omitted	No specific CM dump.
P	List available dumped PPs in byte format.
F=n	Read file number n. (First file is number 1.)
F=n $_1$ -n $_2$	Read file numbers n $_1$ through n $_2$.
F or F omitted	Read all files.

†If no list control arguments (C or CD) are specified, all of the core-image input file will be listed in display code format. If any of these arguments are specified, each record on the core-image input file is examined and listed according to specified list control arguments.

Numeric values can be octal or decimal. The character B or D immediately following the number specifies octal or decimal. If not specified, addresses are assumed to be octal and file numbers are assumed to be octal.

When the dump input file is ZZZZZDD and it is called from a batch job, LISTCID issues the following message:

SYSTEM DUMP LIST REQUESTED/LOCKIN OR DROP.

The central site operator must enter the n.LOCKIN command to continue the LISTCID operation.

Examples:

To list the contents of the system dynamic dump file from the central site console, the operator must select a clear control point and enter

n.X LISTCID.

To list all PP dumps and any CM locations between 0 and 10000g, a batch job could contain the control statement:

LISTCID (P,C=0-10000)

ERROR MESSAGES

The error messages issued by LISTCID are explained in the NOS/BE Diagnostic Handbook.

The downline loading utility reloads controlware when the system is running in engineering mode. The CP program BCLOAD, and the PP programs DLL and NLD make up the downline loading utility. The controlware can reside on tape, on a sequential mass storage file, on file INPUT, or on the system file ZZZZZ04. Controllers that can be downline loaded are the 7021, 7054, 7152, 7154, and 7155 RMS and tape controllers and the 380-170, 380-370, and 380-110 Network Access Devices (NADs).

NOTE

Since the autoload function clears all reserved units on the channel for the access, reloading controlware during normal system operation can create broken interlocks. Units reserved by the alternate access are not affected.

BCLOAD

When the system is in engineering mode, BCLOAD can be called by the operator from the console or with the BCLOAD control statement in a batch job. If the operator is using BCLOAD to load NAD controlware, the system need not be in engineering mode. The format is as follows:

BCLOAD,cc,cw,F=ff,LT=llll,ND=nn,AC=aaaa,TY=ttt.

Parameter cc must be specified. The remaining parameters are optional and have default values.

cc One or two-digit octal channel number.

cw Source of controlware binary file. Default is SYSTEM.

<u>cw</u>	<u>Description</u>
BCFILE	Name of the local file attached by the user before calling BCLOAD. This can be a tape file or a sequential mass storage file.
INPUT	System file INPUT (valid only when BCLOAD is called from a batch job).

<u>cw</u>	<u>Description</u>
SYSTEM	System file ZZZZZ04 (for 7xxx controlware) or system library NUCLEUS (for 380-xxx controlware).

<u>Controller</u>	<u>Deckname</u>
7021	0MT
7054	0SY
7152	0MT and 0SZ
7154	0SZ
7155	0SJ
380-170	NAD-170
380-370	NAD-IBM
380-110	NAD-MIN

ff One- or two-digit octal number of files to skip in positioning to the appropriate controlware binary file on BCFILE. This parameter is ignored if cw is INPUT or SYSTEM. The default is zero.

Parameters LT, ND, AC, and TY apply only to remote NAD controlware loading. If LT=0000, BCLOAD ignores any values specified for ND, AC, and TY.

llll Local trunk control unit (TCU) enables. The llll are four binary digits that indicate which of the local TCUs the local NAD can use to pass the controlware to the network; that is, which TCUs are enabled. A 1 indicates that the corresponding TCU is enabled; a zero, disabled. The TCUs are numbered 0 through 3. A setting of LT=1010 indicates that TCUs 0 and 2 are enabled; 1 and 3 are disabled. The default is 0000, which indicates that the local NAD is the one to be loaded.

nn Two hexadecimal digits specifying the logical trunk address of the remote NAD. The default is 00.

aaaa Four hexadecimal digits specifying the access code of the remote NAD. The default is 0000.

ccc Three alphanumeric characters specifying the type of controlware to be loaded. The available values for ccc follow.

<u>ccc</u>	<u>Type of Controlware</u>
170	CYBER 170 (380-170), default
IBM	IBM (380-370)
MIN	Common minicomputer, such as DEC (380-110)

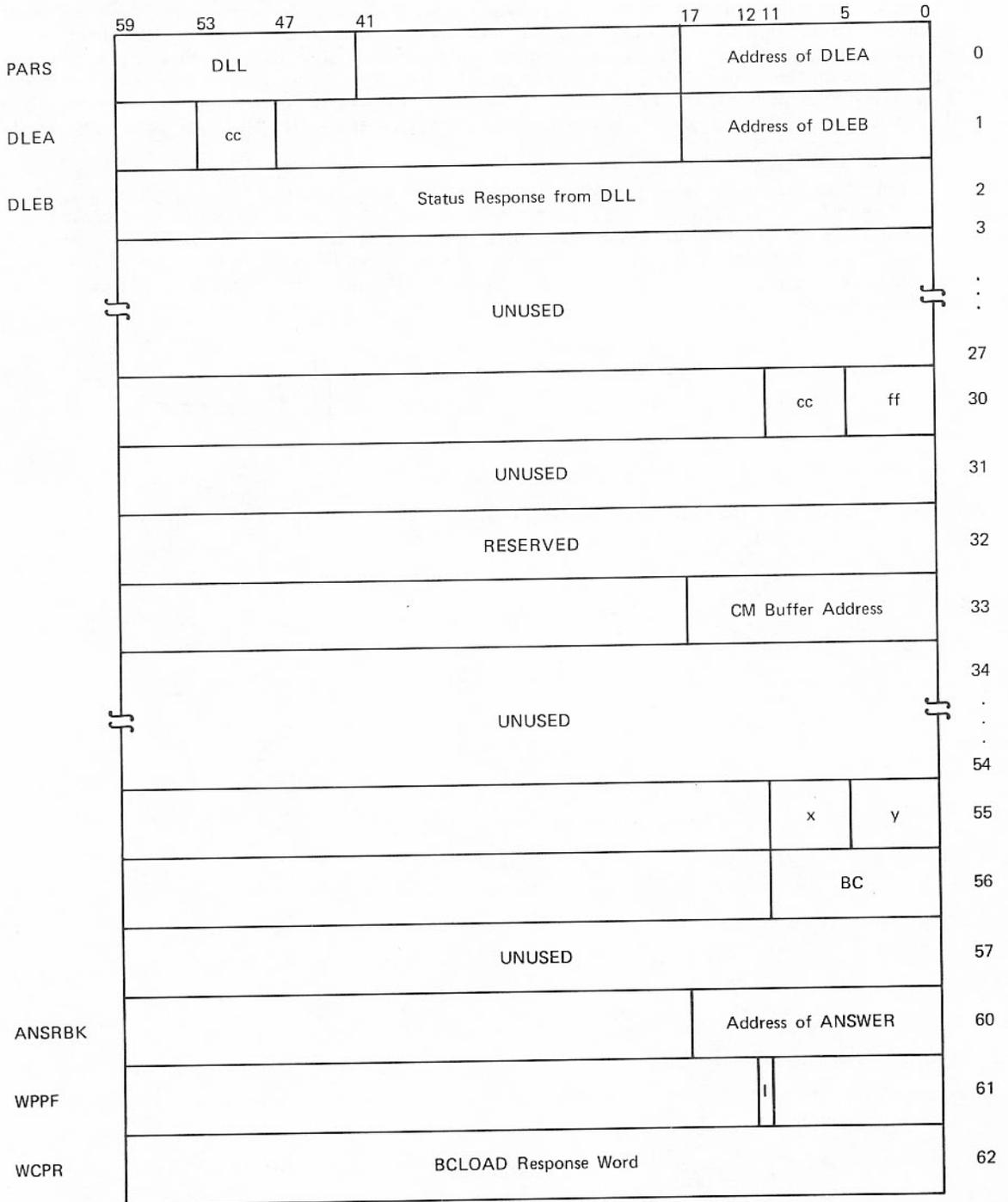
BCLOAD is a CP program which reads the controlware from the specified binary file into a buffer for use by the appropriate PP downline loading program (NLD or DLL). BCLOAD calls PP program CVL to determine whether the specified channel is associated with a NAD or with an RMS or tape controller. If the channel is associated with a NAD, CVL either grants dedicated maintenance access to the NAD, or if the NAD EST entry is not off, or the NAD is in use, CVL returns an error code. If access is granted, BCLOAD reads the NAD controlware from the specified file and calls NLD to complete the loading process.

If the channel is associated with an RMS or tape controller, BCLOAD passes the input directives information it read from the data file to PP program DLL. BCLOAD then reads the controlware from the specified binary file and verifies that the controller can be downline loaded. BCLOAD calls DLL into execution and then goes into a recall state waiting for a response from the PP. BCLOAD and DLL communicate through a parameter buffer area in BCLOAD. The address of this area is passed to DLL in the initial RA+1 request.



PARS is the address of the parameter buffer area.

The format of the parameter buffer area is:



Description of parameter buffer area:

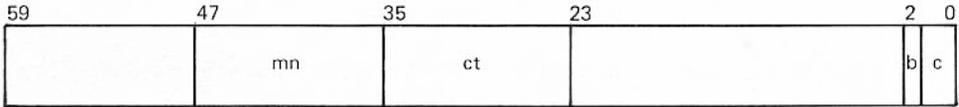
<u>Word</u>	<u>Bits</u>	<u>Field Name</u>	<u>Description</u>
0			Contains the RA+I request. Bytes 0 and 1 contain the display code for DLL left justified. Bytes 3 and 4 contain the address of DLEA right justified.
1			Bits 53 through 48 contain the channel number cc. Bytes 3 and 4 contain the address of DLEB.
2			Status response from DLL.
30	11-6	cc	Channel number.
	5-0	ff	Number of files to skip.
32			Reserved.
33	17-0	CM Buffer Address	Address of buffer containing the controlware binary file.
55	11-6	x	File containing controlware.
			1 BCFILE
			2 INPUT
			4 SYSTEM
	5-0	y	Controller type.
			1 7054
			2 7154
			3 7155
			4 7021
56	11-0	BC	Display code of characters BC.
60			Bits 17-0 contain the address pointer to PP answer area.
61			PP communication word. If bit 12 is set (I), BCLOAD is waiting for DLL to complete processing. Otherwise, bit 12 contains status codes from DLL.
62			BCLOAD response word; set to nonzero when BCLOAD processes status.

BCLOAD checks the contents of the status words returned by DLL. BCLOAD displays operator action messages and writes error data on the output file when fatal errors occur.

DLL returns status information to the following words in BCLOAD: DLEB, WPPF, and ANSWER.

DLEB

The format of DLEB is:



mn Hardware mnemonic.

(AY, AZ, AJ, MT, or NT)

ct Controller type.

1 7054

2 7154

3 7155

4 7021

b If set to 1, the specified channel number is not in EST.

c Complete bit.

WPPF

The format of WPPF is :



d Operator action required if nonzero.

1 Coupler reserved.

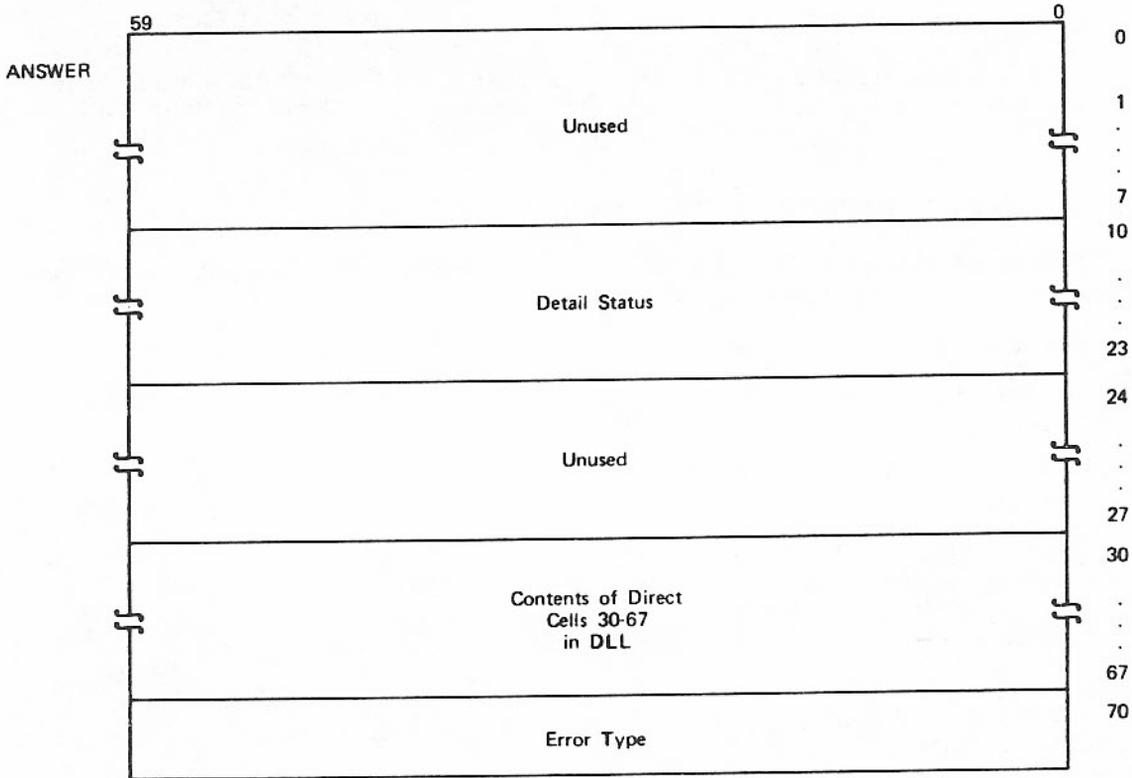
e PP error report. If set, additional status information is returned in the ANSWER buffer.

c Complete bit.

h Recall bit.

ANSWER BUFFER

The format of the ANSWER buffer is:

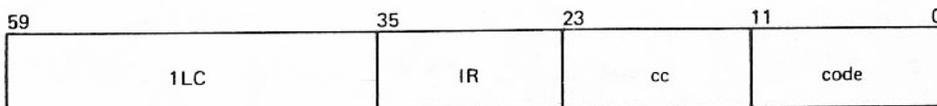


Word 70 Error type.

- 77 No error; load complete.
- Other Downline load error.

DLL

DLL is a PP program that does the actual downline loading. DLL aborts if the system is not in engineering mode. DLL verifies that the channel number specified is defined in the EST and is a 7021, 7054, 7152, or 7155 controller and returns this information to status word DLEB. DLL checks for coupler reserved errors and returns status to WPPF if such an error occurs. When reloading controlware for the 7021 controller, DLL calls 1LC to load conversion tables. The format of the request to load conversion tables is:



IR Input register address of DLL.

cc Channel number.

code If 6, load conversion tables code number.

Errors occurring in ILC or DLL during the downline load process are fatal. Status is returned to WPPF. Detail status and the contents of direct cells in DLL are returned to the ANSWER buffer in BCLOAD.

BUILDING A CONTROLWARE FILE

Use the following example to build a controlware file for use with BCLOAD. In this example, the controlware records are copied to tape and then to a mass storage file.

```

job statement
REQUEST(BCFILE,...)
COPYYBR(INPUT,DISK1)
COPYYBR(INPUT,DISK2)
COPYYBR(INPUT,DISK3)
COPYYBR(INPUT,DISK4)
REWIND(BCFILE)
REWIND(DISK1,DISK2,DISK3,DISK4)
COPYYBF(DISK1,BCFILE)
COPYYBF(DISK2,BCFILE)
COPYYBF(DISK3,BCFILE)
COPYYBF(DISK4,BCFILE)
REWIND(BCFILE)
REQUEST(CTLWARE,*PF)
COPYYBF(BCFILE,CTLWARE,4)
CATALOG(CTLWARE,...)

```

```

7/8/9
7/8/9      Coldstart binary deck      MB434 - A14      7021 controller
7/8/9      Coldstart binary deck      MA710 - A13      7054 controller
7/8/9      Coldstart binary deck      MA401 - A05      7154 controller
7/8/9      Coldstart binary deck      MA721 - A02      7155 controller
7/8/9
6/7/8/9

```

EXAMPLES

BCLOAD can be called after the operator turns on engineering mode. (Engineering mode is not required when the operator calls BCLOAD to load NAD controlware.)

1. This example shows how to bring BCLOAD to a clear control point to load controlware from the system file.

```
n.X BCLOAD,7.
```

or

```
n.X BCLOAD,7,SYSTEM.
```

2. In this example BCLOAD is called from a batch job to load controlware for a 7054 controller from any file.

a. JOB1 loads the controlware contained on the system file.

```
JOB1,....  
BCLOAD,7.  
7/8/9  
6/7/8/9
```

b. JOB2 loads the controlware contained on file INPUT.

```
JOB2,....  
BCLOAD,7,INPUT.  
7/8/9  
Coldstart binary deck          MA710  
7/8//9  
6/7/8/9
```

c. JOB3 loads the controlware from tape. The 7054 controlware is the second file.

```
JOB3,....  
REQUEST(BCFILE,...)  
BCLOAD,7,BCFILE,F=1.  
7/8/9  
6/7/8/9
```

d. JOB4 loads the controlware from a sequential mass storage file. The 7054 controlware is the second file.

```
JOB4,....  
ATTACH,BCFILE,CTLWARE,....  
BCLOAD,7,BCFILE,F=1.  
6/7/8/9
```


The DSD system console L-display is generated by a user's CPU program. If certain words within the field length of a job are formatted properly, information in a buffer within the field length can be displayed on the console when the operator brings the L-display to the left screen and assigns it to the control point of the job. The operator can also send data back to the CPU program.

USING THE L-DISPLAY

The specifications for using the L-display are as follows.

Word RA+RA.CCD must be formatted in the following manner.

12/COMMB,48/8RLDISPLAY

where COMMB is the relative address, less than 7171B, of a communications buffer. If COMMB=0, DSD assumes a default of 101B.

Keyboard entries from the console operator are transmitted to words COMB through COMMB+6. Byte 4 of COMMB+7 contains a cycle count which DSD increments each time through its main loop. Information to be displayed should start at location COMMB+10B. DSD can display a maximum of 600B words of information from the buffer. Figure 13-1 shows the format for the L-display communications buffer.

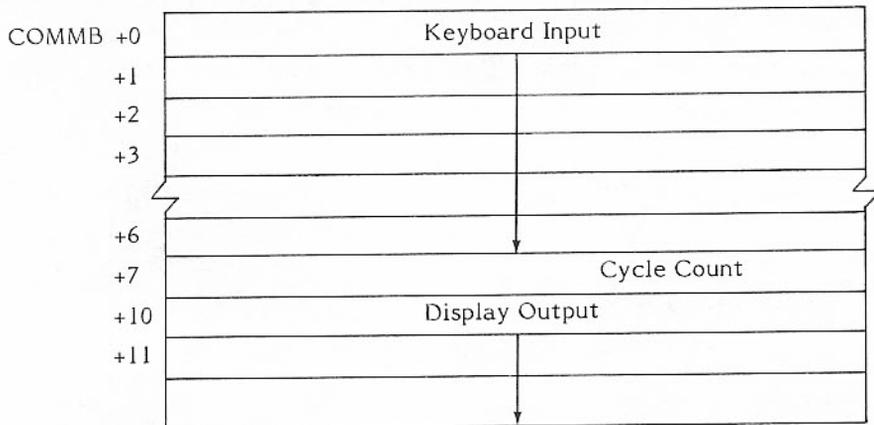


Figure 13-1. L-Display Communication Buffer

DISPLAY DATA

The information in the buffer should be organized as one or more data groups. A data group consists of a header word followed by up to 16 words containing one or more text lines. The format of the header word is as follows.

- 6/Word count (1-16 (0 or >16 acts as a buffer terminator))
- 3/Character size (0, 1 or 2 for small, medium, or large)
- 3/Repeat count (if 0, then 1 is assumed)

The L-display driver in DSD uses the information in the header word to generate the function code that is sent to the display controller to establish the character size. Therefore, all text lines in a data group

have the same size. The text line consists of coordinates that are followed by the display code text to be put on the screen. The coordinates can consist of an x coordinate, or an x and y coordinate. For further information about displaying data on the console screens, see the display station programming section in your hardware reference manual.

The display can be broken up into a grid consisting of 51 lines and 64 columns. To avoid overwriting the information that the system puts at the top and bottom of the screen, the CPU program should only use lines 5 through 47.

Two sample lines are illustrated in the following COMPASS code (figure 13-2).

```

+      VFD      6/ENDL1-*,3/0,3/0,12/6000B      INFO BYTE AND X COORDINATE
      VFD      12/7000B+47*10                    Y COORDINATE, LINE 47
      VFD      24/4LTHIS
      DIS      ,*SHOULD APPEAR ON LINE 47*
ENDL1  EQU      *
+      VFD      6/ENDL2-*,3/0,3/0,12/6000B      LINE 5
      VFD      12/7000B+50
      VFD      24/4LFORM
      DIS      ,*AT ERROR*
ENDL2  EQU      *

```

Figure 13-2. L-Display Lines

INPUTTING DATA

Figure 13-3 illustrates a processing flow chart that a CPU program should have to properly accept data from DSD.

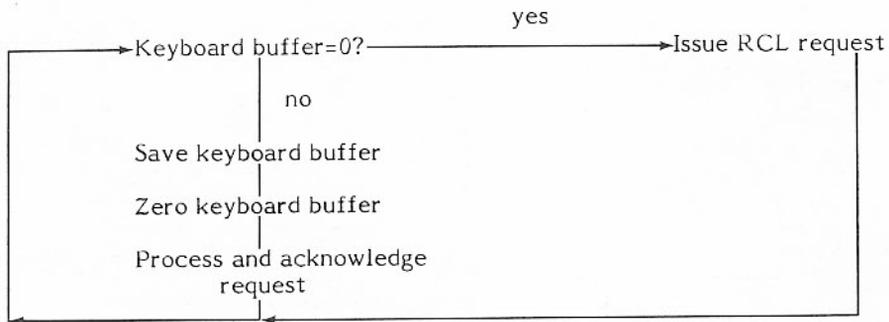


Figure 13-3. CPU Program Interface to DSD

The CPU program must go into periodic recall, not auto-recall. In auto-recall, MTR does not check for L-display activity, considers the control point hung, and aborts the CPU program. Also, DSD does not set a complete bit when it puts data into the buffer. There is no way to terminate auto-recall if a scheme was devised to recognize a no-activity condition.

DSD does not do interpretive keyboard processing while the L-display is active. Instead, the characters are only collected in a buffer. When the carriage return key is pressed, the characters in the buffer are packed 10 characters per word, left justified, and transferred to the central program starting at RA+COMMB. All characters on the keyboard (display code values from 1 to 57B) can be entered in the buffer with the exception of a \$ (display value of 53B) since that code is used by DSD to clear the keyboard buffer.

INDEX

This index contains entries for material in both volumes. Reference material for sections 1 through 12 are in volume 1.

- A directive 11-2
- Abnormal job termination 7-17
- Abort control point 3-20
- Abort flag bit 7-14
- Access, exclusive 7-10
- Access level 8-10
- Access to loader word 7-32
- ACCOUNT control statement 7-13
- ACCOUNT routine 7-13
- Accounting, permanent file 6-5
- Accounting, time 3-18
- ACCSF macro 7-14
- ACK Message (RHF) queue entry type) H-34
- ACQUIRE macro 4-7
- Active PP link 3-16
- ADD directive 8-11,13
- Address public set 6-15
- Addresses, list of 5-20
- ADDSET control statement 6-10
- Aging rate 7-8
- Allocatable device I/O 5-18
- Allocatable devices 4-1
- Allocated memory 2-3
- Allocation style 4-35; B-179
- ALTER function 4-7,15
- Analysis directives 11-1,2
- ANSWER buffer 12-7
- APF (refer to Attached permanent file)
- Application Table - Refer to RHF Application Table
- APR 7-2,3
- Archived file, retrieve 6-5
- Area table 2-16; B-138
- Area table, local 10-1
- ASCII character sets A-1,4
- Assigning PPs 3-15
- Assigning tapes 4-28,29
- Asynchronous Supervisory Message Reply (RHF queue entry type) H-31
- Attached permanent file 6-1,2
 - Interlocks 6-1,2
 - Table 6-1,3,4; B-88
- Attributes
 - Device 4-32
 - Set 4-31,32
- AUDIT utility 6-5
- Auto recall 2-5
- Automatic recall (refer to Auto recall)
- Automatic tape assignment 4-29
- B directive 11-2
- BATCH bulletin 9-4
- Batch jobs 7-8
- BC directive 11-2
- BCLOAD 12-1
- BKSP macro 5-17
- BKSPRU 5-17,26
- Block map 10-2
- Bootstrap an extended memory system 10-1,3
- Bootstrap program 1-4
- Breakpoint table 2-16; B-136.12
- Broken interlock 5-36
- Buffer argument error 5-35
- Buffer, message 1-1
- Buffer parameter error 5-35
- Buffered I/O 5-41
- Bulletin names 9-2
- Bulletin (refer to System bulletin)
- BULLUP data statements 9-1
- BULLUP utility 9-1
- Bytes 1-2
- CALL macro 2-16
- Calls for read/write 5-38
- CALLSS macro 2-24; H-3
- CALLSS macro parameter block H-5
 - NETGET H-6
 - NETGETL H-6
 - NETOFF H-6
 - NETON H-5
 - NETPUT H-7
 - NETWAIT H-6
- CBM (refer to Circular buffer manager)
- CE diagnostic file 4-31
- CE error file 5-54; 6-2
- CEFAP buffer area B-141
- CEJ (refer to Central exchange jump)
- Central exchange jump 3-13
- Central memory 1-2; 2-1
 - Allocated 2-3
 - Copy 2-13
 - Dump 11-2,3
 - Field length 7-14
 - Match 11-5
 - Online dump 11-8
 - Organization 2-1
 - Physically available 7-28
 - Queue 7-9

Read/write order codes 5-24
 Storage moves 2-3
 Unallocated 2-3
 Words 1-2
 Central memory resident 1-2; 2-1,7
 Areas 2-10
 Library directory 8-24; B-149
 Library format B-152
 Loader 10-1
 Pointer area B-3,6
 Segmentation 2-15; 10-1
 Tables 2-11; B-1
 Central processor unit 1-1
 Priority 3-15
 Scheduling 3-14
 Status 3-21
 CERFILE 4-1
 CESAB macro 7-28
 CEVAL macro 7-20
 CHANGE directive 8-13
 Change running system 8-1
 Change user library 8-1
 Channel reservations 3-18
 Channel status table 4-27; B-19
 Channel table 5-47,53; B-89
 Character sets A-1
 Checkword errors 5-55,57
 CHT (refer to Channel table)
 CIO FET H-3
 CIO (refer to Circular input/output processor)
 Circular buffer 5-2,3
 Circular buffer manager 5-42
 Circular input/output processor 5-1
 Codes 5-1
 Function routines 5-4
 Operation 5-4
 Tape operations 5-5
 CM (refer to Central memory)
 CMR (refer to Central memory resident)
 CMRDIR program 10-1
 CMRES 10-4
 CMRLIB library 10-1
 COMMENT control statement 7-13
 Comment field, EDITLIB 8-20
 Common area E-4
 Common block name 10-4
 Common test and initialization 1-4
 Communication area 1-5; 2-4; 5-22,23
 COMPLETE directive 8-13
 Connect Request Available (RHF queue entry type) H-37
 Connect Request Awaiting Assignment (RHF queue entry type) H-37
 Connect Request Reply (NDR function) H-13
 Console display dumps 11-2
 CONTENT directive 8-14
 Contents statements 9-2
 Control permission 7-10
 Control point 1-3; 2-1; 4-1
 Control point, abort 3-20
 Control point area 1-3; 2-1; 7-11; B-22
 Control point dayfile dump 11-2
 Control point field length access 3-6
 Control point, system 2-23
 Control point tables C-1
 Control point, user 2-23
 Control point zero 4-1
 Control statement buffer 7-11
 Control statement processing 7-11,13
 Control statement record 7-11
 Control statement source file 7-11,14
 Controller status 5-57
 Controlware loading 12-1
 Copy CM 2-13
 Core-image dump 11-10
 COUNT function 4-7,19 through 21
 CPCIO 5-4
 CP-PP communication 2-4
 CP routines, EDITLIB 8-21
 CP-system communication 2-4
 CPA directive 11-3
 CPMTR 2-2; 3-12,13
 CPMTR functions B-52
 CPMTR organization 3-13
 CPU (refer to Central processor unit)
 CPU request word 1-1
 Crash file 11-7
 Create deadstart tapes 8-1,20,35
 Create system libraries 8-1
 Create user library 8-1
 Creating system bulletin file 9-3
 CS directive 11-6
 CST 4-27; B-19
 CTI (refer to Common test and initialization)
 CX directive 11-3
 Cy directive 11-3
 CYBER 180-class mainframes
 CESAB parameter block variable entry section 7-29
 Mainframe attribute block element entries B-117,118
 Maintenance channel 3-25,36
 Maintenance registers 3-13,36
 Software flags B-21
 Unified extended memory v
 CYBER Loader 10-1
 Cycles, permanent file 6-23
 DAM (refer to Device allocation map)
 DAT (refer to Device activity table)
 Data Messages H-4
 Dayfile 4-1
 Buffer area B-115
 File environment table B-115
 Macro 4-3

Message 3-10,22
 Dayfile, job 4-1; 7-16,17; 11-2
 Dayfile, system 4-1,23,31; 11-2
 DDAM file format D-24
 DDP (refer to Distributive data path)
 DDT (refer to Dismountable device table)
 Deadlock 4-30
 Deadstart 1-4
 Deadstart dump analyzer 11-1
 Deadstart dump, express 11-1
 Deadstart functions and options 1-4
 Deadstart loader 8-6
 Deadstart system tape 4-1
 Deadstart tape 1-5
 Deadstart tape, create 8-1,20,35
 Deadstart tape, modify 8-1
 Debug mode 5-34
 Default DSDUMP directives 11-7
 Delay count, recall 2-5
 Delay stack 3-15
 DELETE directive 8-14
 DELSET 5-40
 Determine Job Origin (RHH function) H-22
 Device
 Master 4-32
 Permanent file 4-32
 Queue 4-32
 System 4-32; 5-18
 Device activity table 2-13; 4-27; B-85
 Device allocation map 4-35; 6-3; D-17
 Ordinal 4-36
 Device attributes 4-32
 Device codes B-75
 Device I/O 5-18
 Device overflow table 4-33; B-99
 Device pool table 4-26
 Device queue 7-10
 Device set 4-31
 Label D-1
 Validation 6-12
 Device status table 4-27; B-97
 Device tables 4-26
 Devices
 Allocatable 4-1
 Nonallocatable 4-1
 DFILE 4-1
 Diagnostic file 4-31
 Diagnostic routines, hardware 7-20
 Direct access area E-3
 Direct access area (extended memory) 1-2
 Direct access extended memory 7-8; 10-1
 Direct cells, PP 3-1
 Directives, DSDUMP analysis 11-2
 Directory limits 8-25
 Directory (refer to System directory)
 Disconnect Path (NDR function) H-16
 Disk 4-31
 Disk I/O, 819 5-43
 Disk tables D-1
 Dismountable device table 4-26; B-104
 Fixed section 4-26; B-104
 Variable section 4-26; B-106
 Dismountable pack processing 5-36
 Display directives 11-2
 DISPOSE control statement/macro 4-3
 Dispose files 7-16
 Disposition code 4-2; B-79
 Distributive data path 5-41
 DLEB 12-6
 DLL 12-7
 DOT (refer to Device overflow table)
 Downline loading utility 12-1
 DPT (refer to Device pool table)
 Drive 4-31
 Drop PP 3-20
 Dropout (NDR function) H-12
 DSD-INTERCOM communication C-29
 DSD L-Display 13-1
 Communication buffer 13-1
 CPU program interface to DSD 13-2
 Data Programming 13-1
 Inputting data lines 13-2
 DSDUMP control statement 11-1
 DSDUMP directives, default 11-7
 DSDUMP messages 11-7
 DSMOUNT 5-40
 DST (refer to Device status table)
 DUMMYnn 10-4
 Dump
 Analyzer 11-1
 Buffer controllers 11-2
 CM 11-2,3
 Control point dayfile 11-2
 Directives 11-2
 Entry point table 11-5
 Express deadstart 11-1
 Extended memory 11-3,4
 File format, dynamic 11-8
 File name table 11-2
 INTERCOM area 11-5
 INTERCOM buffers 11-5
 INTERCOM tables 11-5
 Job descriptor table 11-2
 PPs 11-4
 PPUs 11-4
 RBRs 11-5
 Request stacks 11-8
 Segment name table 11-5
 Sequencer tables 11-5
 Status control register 11-5
 System 11-1
 System dayfile 11-2
 System dynamic 11-8
 System exchange package 11-5
 TAPES table 11-2
 User field length 11-5
 DUMPF utility 6-5,6
 Dynamic dump, system 11-8

ECS (refer to Extended memory)
 EDD 11-1
 EDITLIB
 Character set 8-1
 Control statement 8-7
 CP routines 8-21
 Directive file 8-8
 Directive format 8-13
 Directive restrictions 8-12
 Errors 8-9
 Examples 8-33
 Field length 8-11
 File positioning 8-22
 Files 8-5,19
 Library positioning 8-22
 Limits 8-25
 Output 8-8
 PP routines 8-20
 Syntax 8-3
 SYSTEM 8-1,5
 Tables 8-25
 USER 8-1
 EI directive 11-6
 EM empty page stack B-145
 Empty Queue Entry (RHF queue entry type) H-30
 ENCSF macro 7-15
 ENDRUN directive 8-14
 ENDSEG macro 2-16
 Engineering mode 7-21
 Entry point 8-33
 Entry point cross reference 10-2
 Entry point map 10-2
 Entry point name table 8-28,33; B-152
 Entry point names C-16
 Entry point number 8-30
 Entry point table dump 11-5
 Entry table 2-16; B-139
 Environment Interface Communication Block B-136
 EPIB directive 11-6
 EPNT (refer to Entry point name table)
 Equipment status 7-20
 Equipment status table 4-26; B-55
 Error file, CE 5-54; 6-2
 Error logging status table 2-12; B-134
 Error recovery, extended memory 2-20
 ERT (refer to External reference table)
 ESM (refer to Extended semiconductor memory)
 EST (refer to Equipment status table)
 Event list H-10
 Event list entry H-11
 Event stack 3-15
 Event type H-11
 EX.NXTPB subfunction 3-34
 EX.SPRCL subfunction 3-33
 EX.STAT subfunction 3-33
 Examples, LDCMR 10-5
 Exchange package 3-14; 11-5; B-21
 Exclusive access 7-10
 EXIT control statement 7-14
 Exit flag bit 7-14
 EXPORT abort 11-8
 Express deadstart dump 11-1
 Express jobs 7-8
 Extended core storage (refer to Extended memory)
 Extended memory 1-2
 Buffered I/O 5-41
 Buffers 5-41
 Common area E-4
 Direct access area E-3
 Dump 11-4
 Error recovery 2-20
 Field length 7-14
 Flaw table E-7
 ICC E-5
 Intercomputer area E-5
 Label E-2
 Match 11-5
 For more than one mainframe E-1
 For one mainframe E-1
 Page assignment manager E-6
 Paged area E-3
 Paging 1-3
 Segments E-8
 System image 2-18,19
 System pages E-7
 Systems, segmentation 2-15
 Tables E-1
 Extended semiconductor memory v
 External reference list 8-33
 External reference table 8-27; B-152
 External stack request format 5-23
 Ey directive 11-4
 Facilities Interface Program (FIP) H-1,2,42
 Data formats H-45
 Data organization H-45
 Design overview H-42
 FIP-RHF interface H-5
 System flow H-42
 UCP Request (RHF queue entry type) H-33
 Factory data flag 6-11
 FDB (refer to File definition block)
 FET (refer to File environment table)
 FID (refer to File identifier)
 Field access flag 3-6,12
 Field length 2-39; 7-14
 Access 3-6,25
 EDITLIB 8-11
 Override 8-11,19
 User 2-4
 File definition block C-19
 File disposal 7-16
 File environment table 4-19 through 4-21; 5-1; C-4
 File formats D-1

File group 7-2
 File identifier 4-4
 File information table 4-19 through 4-21
 File name table 4-26; B-59
 File name table dump 11-2
 File names, special 4-2,31
 File positioning, EDITLIB 8-19
 File status information 4-26
 File status table 4-26
 File tables 4-19 through 4-21
 File transfer H-3
 File Transfer Common Area (XFRHOLD) H-47
 File Transfer Data Flow H-4
 File-transfer-in-progress H-4,42,44
 File Transfer Table (FTT) H-46
 Files 1-3; 4-1
 Input 7-17
 Local 4-1,2; 7-17
 Output 7-17
 Permanent 4-1,2; 7-17
 Queue 4-1,2
 System 2-2; 4-1
 Files, EDITLIB 8-5,19
 FILMPL 4-2
 FILMPR 4-2
 FINISH directive 8-14
 FIP - Refer to Facilities Interface Program
 FIRST 5-2
 First level peripheral processors (refer to PPU's)
 FIT (refer to File identifier table)
 Fixed section 4-26
 Flaw table E-7
 FLPP (refer to PPU's)
 Flush files 7-17
 Flush Job Dayfile (RHH function) H-21
 Flush Path (NDR function) H-16
 FNT (refer to File name table)
 FSN program 6-15
 FST (refer to File status table)
 FTT - Refer to File Transfer Table
 Function numbers 5-39

 GEMINI link file format D-27
 GENLDPF utility 6-5
 GET function 4-7,15
 GETLC macro 7-32,33
 GETLOF macro 7-18
 Global library 7-12
 GOTO macro 2-16; 10-3
 GOTOTAB macro 2-16
 Graphic character set A-1
 Graphics jobs 7-8

 H directive 11-2

 HARDPL 4-2
 HARDPR 4-2
 Hardware diagnostic routines 7-20
 Hardware error file 4-1
 Hardware interlocks 6-1
 Hash points 6-3
 Hexadecimal dump directives 11-2,3,4,6
 Host ID 2-14; 7-19
 HSP 5-44

 ICC E-5
 ID hash points 6-3
 Identifiers 7-19
 Idle (NDR function) H-10
 IDs 7-19
 IH (refer to Interrupt handler)
 Immediate return, NETXFR H-44
 IN 5-3
 INCLUDE directive 8-15
 Incoming Control Message (RHF queue entry type) H-35
 Incomplete cycle 6-3,4
 Increment Job Tape Count (RHH function) H-21
 Initialize NAD (NDR function) H-16
 Initialize system 10-3
 Initiate System Dynamic Dump of Control Point (RHH function) H-22
 INPUT file 4-2
 Input file 7-17
 Input/output 5-1,4,18; 7-3
 Extended memory buffered 5-41
 Input/output tables 4-19 through 4-21
 Input queue 4-2,3; 7-1
 Input register 1-1; 2-4; 3-5,12
 Integrated scheduler 3-31; 7-4
 INTERCOM
 Area dump 11-5
 Asynchronous terminal port entry B-162
 Auxiliary user table B-174
 Buffer area B-166
 Buffer dump 11-5
 Connection table B-164
 Driver communication area B-158
 Interlock table B-157
 Jobs 7-8
 Multiplexer subtable 4-41; B-159
 Multiplexer table 4-41; B-83
 Multiuser job table B-177
 Pointer and buffer area 4-41; B-155
 Port configuration table B-176
 PRU buffer B-165
 Queue 7-11
 Restart 11-8
 Synchronous table port entry B-160
 Tables 4-26,41
 Tables dump 11-5

- Terminal ID table B-163
- User table B-167
- INTERCOM, system bulletin file 9-4
- Intercomputer area E-5
- Interjob connection table 2-35
- Interlock broken 5-36
- Interlock, LDCMR 10-5
- Interlocked stack request 6-2
- Interlocks 6-1,2
 - APF 6-2
 - Hardware 6-1
 - MST 6-1
 - PFM 6-2
 - Set 6-1
 - SMT 6-2
 - Utility 6-2
- Internal stack request format 5-23
- Interrupt handler 5-45,47
- Interval 8-4
- Invalid stack entry 5-35
- I/O (refer to Input/output)
- IT directive 11-6
- ITABL (refer to INTERCOM table)

- JANUS 7-3
 - FET C-12.1
 - File environment table C-12
- JDT directive 11-2
- JDT (refer to Job descriptor table)
- Job advancing 7-11
- Job classes 7-8
- Job control 7-19
- Job control area 7-8
- Job control point tables C-1
- Job control statement source file 7-14
- Job dayfile 4-1; 7-16
- Job descriptor number 2-2
- Job descriptor table 2-2; 7-1,9; 11-2; B-128
- Job descriptor table dump 11-2
- Job descriptor table entry 7-1; B-128
- Job input file 7-11,14
- Job input queue 7-1
- Job killed 7-18
- Job pause bit 7-11
- Job postprocessing utilities 7-18
- Job prescheduling 4-29
- Job processing 7-1
- Job queue, PP 3-15
- Job rerun 2-37
- Job scheduling 4-30; 7-1,4,6
- Job scheduling queues 7-1,9
- Job sequencer 7-2
- Job status 2-34
- Job status code 7-9
- Job termination
 - Abnormal 7-17

- Normal 7-16
- Jobs
 - Batch 7-8
 - Express 7-8
 - Graphics 7-8
 - INTERCOM 7-8
 - Multiuser 7-8
 - Sequencer 7-2

- Kill job 7-18

- Label, rewrite RMS 6-11
- LABELMS control statement 6-10
- Large central memory v; 5-44
 - Buffer 5-49,52
- Large central memory extended v
- LCM (refer to Large central memory)
- LCME (refer to Large central memory extended)
- LCN - Refer to Loosely Coupled Network
- LDC 10-5
- LDCMR 10-1
 - Control statement 10-1
 - Examples 10-5
 - Files 10-3
 - Interlock 10-5
 - Map 10-2
 - Reserved names 10-4
 - System security 10-4
- Libraries, system 1-1,4; 8-1
- LIBRARY directive 8-15
- Library directory 8-20; 10-1
- Library files 1-3
- Library limits 8-25
- Library name 8-3,11
- Library name table 8-15,25,27; B-150
- Library positioning, EDITLIB 8-22
- Library table interfaces 8-34
- Library, user 8-1,23
- LIMIT 5-2
- LIMIT control statement 7-13
- Limits, EDITLIB 8-25
- Link ID 7-19
- List core-image dump 11-10
- List of addresses 5-20
- List-of-files address 7-18
- LISTCID error messages 11-11
- LISTCID utility 11-10
- Listing dump files 11-10
- LISTLIB directive 8-16
- LISTLNT directive 8-16
- LNT (refer to Library name table)
- Load dumped files from tape 6-5
- Loader, CYBER 10-1

- Loader map 10-2
- Loader table 2-4
- Loader word 7-32
- Loading 1-4
- Loading, controlware 12-1
- Loading, downline 12-1
- Loading jobs from tape 7-2
- LOADPF utility 6-5
- Local area table 10-1
- Local file name SYSTEM 8-7
- Local files 7-17
- Logical flaw table 6-4; D-20
- Logical groups 7-19
- Logical ID table 7-19; B-113
- Logical identifiers 7-19
- LOGIN bulletin 9-4
- LOGIN command 7-14
- Long term connection 2-35
- Loosely Coupled Network (LCN) H-1

- M.ABORT function 3-20
- M.BUFPTR function 3-20
- M.CCPA function 3-20
- M.CLRST function 3-21
- M.CPJ function 3-21
- M.CPUST function 3-21
- M.DCP function 3-21
- M.DFM function 3-22
- M directive 11-2
- M.DPP function 3-23
- M.EES function 3-23
- M.EESD function 3-23
- M.ICE function 3-24
- M.ISP function 3-25
- M.KILL function 3-25
- M.MCH function 3-25
- M.MFLA function 3-26
- M.NOTE function 3-26
- M.NTIME function 3-26
- M.PASS function 3-26
- M.PATCH function 3-26
- M.PPLIB function 3-27
- M.RACT function 3-27
- M.RBTSTO function 3-28
- M.RCH function 3-28
- M.RCLCP function 3-29
- M.RCP function 3-29
- M.RPJ function 3-29
- M.RPJD function 3-30
- M.RSTOR function 3-30
- M.SCB function 3-31
- M.SCH function 3-31

- M.SEQ function 3-31
- M.SETST function 3-32
- M.SLICE function 3-33
- M.SLPER function 3-33
- M.SPM function 3-33
- M.SPRCL function 3-34
- M.TRACE function 3-35
- M.TSR function 3-35
- MAB (refer to Mainframe attribute block)
- MACHINE control statement 7-14
- Magnetic tape (refer to Tape)
- Mailbox buffer B-112
- Mainframe 7-19
 - Attribute block B-117
 - Characteristics 7-28
 - ID 6-13
 - Model 7-28
- Map, loader 10-2
- Master device 4-32
- Match word 0 11-6
- Maximum field length 2-39
- Maximum logical record size 4-22
- Maximum queue priority 7-8
- MC directive 11-6
- MDI routine 8-24
- Memo mode 3-31
- Memory (refer to CM)
- Message buffer 1-1
- Message, dayfile 3-10,22
- Message, dump 11-3
- Message to UCP dayfile 2-31
- Minimum queue priority 7-8
- MLRS (refer to maximum logical record size)
- MODE control statement 7-13
- Mode of RECOVER 6-12
- Modify deadstart tape 8-1
- Modify running system 8-1
- Modify user library 8-1
- Monitor area 10-3
- Monitor field length access 3-26
- Monitor functions 3-20; B-52
- Monitor mode 3-13
- Monitor mode segment name 11-5
- Monitor, system 1-1; 3-12; 5-29
- Monitor trace mode 3-35
- MOUNT control statement 6-11
- Mounted set table 4-27; B-102
 - Set interlocks 6-1
- MOVE directive 8-16
- Move permanent files/tables 6-5
- Move system directory 8-24
- MR directive 11-6
- MSG table C-29

MST (refer to Mounted set table)

MTR 1-1; 3-12; 5-29

MTR structure 3-12

Multiple access 4-27

Multiplexer subtable 4-41

Multiplexer table 4-41

Multiuser job table B-177

Multiuser jobs 7-8

NAD Off (RHF queue entry type) H-36

NAD PP driver (NDR) H-1,2,4

Communication area H-37

Communication Area Queue Entry

Pointer H-40

Disconnect Path (RHF queue entry type) H-35

FET H-3,7

Address H-7

Flush Path (RHF queue entry type) H-35

Function codes H-10

Initialize NAD (RHF queue entry type) H-35

NDR-RHF communication area H-9

NDR-RHF interface H-9

Obtain Connect Request (RHF queue entry type) H-35

Request Connection (RHF queue entry type) H-34

Status (RHF queue entry type) H-35

Name, EDITLIB 8-1

Name statements 9-2

NDR - Refer to NAD PP driver

NETDBG H-42

NETGET H-5,6,42

NETGET/NETPUT H-4

Data messages H-4

Supervisory messages H-4

NETGETL H-5,6,42

NETOFF H-5,6,42

NETON H-5,42

NETPUT H-5,42

NETUXFR H-42,44

NETWAIT H-5,6,42

NETXFR H-3,5,7,42,43

Immediate return H-43

NETXFRC H-4,42,44

NFL (refer to Nominal field length)

Nominal field length 8-11

Nonallocatable devices 4-1

Normal job termination 7-16

Nsup (status word) H-5,6

Obtain Connect Request (NDR function) H-13

Online dump, PP/CM 11-8

Operator action queue 7-11

Order codes 5-24,35

CM read/write 5-24

Positioning 5-25

PP read/write 5-25

SPM communication 5-26

Stack processor communications 5-26

OUT 5-3

Outgoing Control Message (RHF queue entry type) H-34

OUTPUT file 4-2

Output file 7-17

Output queue 4-2,3; 7-1

Output register 1-1; 3-5,12; 5-1

Overcommitment of tape drives 4-30

Overflow Counter (RHF queue entry type) H-36

O26/O29 mode A-2,3

P display 7-1

P register 3-25

Pack processing 5-36

Pack removal 5-40

Page assignment management E-6

Paged area 1-2, E-3

Paging 1-3

PAM (refer to permanent file catalog allocation matrix)

Parity error 5-35

PAT directive 11-6

Pattern matching 11-5

Pause bit 7-11

PEEK function 4-7,16

Perform FET I/O (NDR function) H-17

Periodic recall 2-5

Peripheral job table B-116

Peripheral processor 3-1

Assignment 3-15

Bit map 7-28,31,32

Calls 3-15

Communication 2-4; 3-5

Communication area 1-5; 3-5; B-30

Direct cells 3-1

Directive 11-4

Drop 3-23

Dump 11-4

Input/output 5-50

Input register 3-5

Job queue 3-15

Library 3-27

Library pointer 8-22

Link, active 3-16

Match 11-5

Monitor 3-5

Online dump 11-8

Organization 3-1

Output register 3-5; 5-1

- Program name reservations B-31
- Program name table 8-6,25,29; B-150
- Program, transient 3-5
- Read/write order codes 5-25
- Resident 3-1,6
- Resident overlay save buffer B-136.11
- Routines 3-6
- Routines, EDITLIB 8-23
- Speed flag 7-29
- Status 11-2
- Status table 3-12
- Status words B-20
- Permanent file 4-2; 6-1; 7-17
 - Accounting 6-5
 - Allocation matrix D-23
 - Catalog 6-2,3,5; D-10
 - Control statements 7-14
 - Cycles 6-3
 - Default set 4-31
 - Device 4-32
 - Directory 6-2,3,5; D-6
 - Directory entry 6-5; D-7
 - Dump to tape 6-6
 - Functions 6-5
 - Incomplete cycles 6-3
 - Interlocks 6-1
 - Load 6-5
 - Name 6-3
 - Queue 7-10
 - Statistics 6-5
 - Status 6-5
 - System interface 6-1
 - Tables 6-2
 - Transfer 6-5
 - Utilities 6-5
 - Versions 6-3
- Permanent file dump tape format 6-6
- Permanent pack queue 7-10
- Permission, control 7-10
- PFC (refer to Permanent file catalog)
- PFD (refer to Permanent file directory)
- PFL 8-10
- PFLOG dump tape format 6-8
- PFLOG tape 6-8
- PFLOG utility 6-5,8
- PFM interlocks 6-2
- Physical buffer B-165
- Physical flaw tables 6-4; D-21
- Physical ID 7-19
- Physical record unit 4-33
 - Buffer B-165
- Physical status of tape drive 4-30
- PK routine 7-10
- PLOT 4-2
- PNT (refer to Program name table)
- Pointers to RHF queues H-38
- Pool PP 3-1,3
- Pool processor 3-3
- Positioning order codes 5-25
- Postprocessing utilities 7-18
- PP (refer to Peripheral processor)
- PPIO (refer to Peripheral processor input/output)
- PPMTR 3-5
- PPMTR functions B-53
- PPNT (refer to Peripheral processor program name table)
- PPS 5-43
- PPSYS identifiers F-1
- PPU dump 11-4
- PPUs 5-43
- Predayfile file name 4-3
- Prefix tables 8-20
- Preinput queue 7-1
- Prescheduling display 4-29; 7-1
- Prescheduling queue 4-29
- Prescheduling tape jobs 4-29
- Priority 7-8
- Priority lockout bit 7-10
- Private sets 4-32
- Program library file 1-3
- Program limits 8-25
- Program name 8-4
- Program name table 8-5,30; B-154
- Program recall 2-5
- PRU (refer to Physical record unit)
- PT directive 11-6
- Public device set addressing 6-15
- Public files 6-15
- Public sets 4-31
- PUNCH 4-2
- PUNCHB 4-2
- PXP directive 11-6
- P80C 4-2
- Q set (refer to Queue set)
- Quantum priority 7-8
- Quantum time length 7-8
- Queue device 4-32
- Queue files 4-2
- Queue roots (RHF) H-38
- Queue set 4-32
- Queues
 - Central memory 7-9
 - Device 7-10
 - Input 4-2,3; 7-1
 - INTERCOM 7-11
 - Operator action 7-11
 - Output 4-2,3; 7-1
 - Permanent file 7-10
 - Permanent pack 7-10
 - Prescheduling 4-29
 - Scheduling 7-1
- R.DCH routine 3-9

- R.DFM routine 3-10
- R.EREQS routine 3-10
- R.IDLE routine 3-6
- R.MTR routine 3-8
- R.OVL routine 3-10
- R.OVLJ routine 3-6
- R.RAFL routine 3-6
- R.RCH routine 3-8
- R.READP routine 3-11
- R.RWP routine 3-11
- R.STB routine 3-9
- R.STBMSK routine 3-9
- R.TAFL routine 3-6
- R.TFL routine 3-8
- R.WAIT routine 3-8
- R.WRITEP routine 3-11
- RA communication area 2-4; C-1
- Random file 8-18,23
- Random user library 8-17
- RANTOSEQ directive 8-17
- RB (refer to Record block)
- RBR directive 11-5
- RBR (refer to Record block reservation table)
- RBT chain D-16
- RBT (refer to Record block table)
- RBTC directive 11-5
- RBTC (refer to Record block table catalog)
- READ macro 5-6, 24
- Read maintenance channel 3-25
- Read Path Data and Send ACK (NDR function) H-14
- Read/write calls 5-38
- Read/write order codes 5-24
- READC macro 5-18, 25
- READLS macro 5-19, 25
- READN 5-8
- READNS macro 5-24
- READSKP macro 5-9, 24
- READY directive 8-17
- Recall
 - Auto 2-5
 - Delay count 2-5
 - Periodic 2-5
- Record block 4-33
- Record block reservation table 4-33, 35; B-97
- Record block reservation table dump 11-5
- Record block table 1-2; 2-1; 4-35; B-178
 - Area dump 11-5
 - Catalog 6-2
 - Chain 4-35; 11-5; D-16
- Record descriptors B-148
- Recording mode 6-11
- RECOVER 6-12
 - Control statement 6-12
 - Mode 6-12
 - Notes and cautions 6-15
- RECOVER macro 7-18
- Recovered parity error 5-35
- Recovery programs 7-18
- Recreate RMS label 6-11
- Reducing system bulletin file 9-4
- Register
 - Input 1-1; 2-4; 3-5,12
 - Output 1-1; 3-5,12
- RELABEL control statement 6-11
- Remote Connection Request (RHF queue entry type) H-37
- Remote Host Facility (RHF) H-1
 - Application Table H-1,24
 - Address H-5
 - Table Connection H-26
 - Table connection queue roots H-39
 - Table Header H-24
 - Table header queue roots H-39
 - Control Point Field Length Overview H-23
 - Initialize (RHH function) H-20
 - Initiation H-3
 - Local NAD Table - queue roots H-40
 - Local variables - queue roots H-38
 - Network Description Table H-1,26
 - Local NAD Entry H-28
 - Mainframe Header H-27
 - Remote NAD Entry H-28.1/H-28.2
 - Table header H-26
 - Queue entries H-28.1
 - Miscellaneous resident to overlay communication H-28.1
 - NDR communication H-28.1
 - Reserved for installations H-28.1
 - UCP communication H-28.1
 - Queue Entry Types H-29
 - RHF/FIP Control Flow H-3
 - RHF/NDR Control Flow H-3
 - Subsystem H-1
 - System control point H-1
 - Tables H-1
- Remote Host Helper (RHH) H-1,19
- REMOVE directive 8-17
- Removing a pack 5-40
- REPLACE directive 8-18
- Replace system directory 8-7
- REPRIEVE macro 7-18
- Reprieve processing 7-18
- REQ function parameter list C-22
- Request Complete (RHF queue entry type) H-30
- Request Connection (NDR function) H-13
- Request count, SCP 2-35
- Request scheduling table B-96
- Request stack 5-18
- Request stack entry B-94
- Request, system 1-1; 2-4
- Request word, CPU 1-1
- RERUN command 7-18
- Rerun job 2-37

Rerun status 2-37
 Reserved local file names C-12.6
 Reserved names, FET C-12.6
 Reserved names, LDCMR 10-4
 RESET 8-7
 Resource recovery mode 6-14
 RESTORE 8-8
 Retrieve archived file 6-5
 Return Jobname (RHH function) H-22
 RETURN macro 2-16
 Return Physical ID (PID) of Host Mainframe
 (RHH function) H-20
 REWIND directive 8-18
 Rewrite RMS label 6-11
 RF directive 11-6
 RFL control statement 7-14
 RHF - Refer to Remote Host Facility
 RHFDBG H-42
 RHFGET H-42
 RHFGETL H-42
 RHFOFF H-42
 RHFON H-42
 RHFPUT H-42
 RHFUXFR H-42
 RHFWAIT H-42
 RHFXHR H-42
 RHFXHRFC H-42
 RHH function codes H-19
 RHH - Refer also to Remote Host Helper
 RMF
 Multiplexer subtable 4-41
 Multiplexer table 4-41
 Pointer and buffer area 4-41
 Tables 4-41
 RMS (refer to Rotating mass storage)
 RMS set processing 6-10
 Rolling 7-6
 Rotating mass storage 4-1
 Device, rewrite label 6-11
 Hardware error 5-36
 Set 4-31
 Tables 4-34
 ROUTE control statement 4-3
 ROUTE macro 4-3
 RPHR 5-10
 RPV program 7-18
 Running system, modify 8-1

 SCB (refer to System circular buffer)
 Scheduler control 7-1
 Scheduler job control area B-126
 Scheduler job descriptor table B-128
 Scheduler mailbox buffer B-112
 Scheduler performance table B-124
 Scheduling jobs 7-6
 Scheduling queues 7-1,9

 Scheduling tape jobs 4-30
 Scheduling tapes 4-29
 SCHAT (refer to subchannel table)
 SCP (refer to System control point)
 SCPA (refer to System control point address)
 Scratch sets 4-32
 Sector 4-33
 SEG directive 11-6
 SEGDEF macro 2-16
 Segment 2-15
 Definition 2-16
 Descriptor 2-18
 In an extended memory system E-8
 Linkage 2-16
 Loading 2-16
 Name table dump 11-5
 Table 2-18; 10-1; B-140
 Segmentation, CMR 2-15; 10-1
 Send Control Message (NDR function) H-14
 SEQ directive 11-6
 SEQTORAN directive 8-18
 Sequencer jobs 7-2
 Sequencer table 7-2; B-100
 Sequencer table dump 11-5
 Sequential file 8-17
 Sequential user library 8-18
 Set 4-31
 Attributes 4-31,32; 6-15
 Device 4-31
 Interlocks 6-2
 Member table 6-3; D-19
 Permanent file default 4-31
 Private 4-32
 Public 4-31
 Queue 4-32
 RMS 4-31
 Scratch 4-32
 Shared 4-33
 System 4-31
 Set Force Exit Processing Bit (RHH
 function) H-22
 Set Up Local Copy of Job Dayfile FNT
 (RHH function) H-21
 SETAL directive 8-18
 SETFL directive 8-19
 SETFLO directive 8-19
 SETLC macro 7-32,33
 SETLOF macro 7-18
 SETMFL macro 2-39
 SETNAME control statement 7-14
 Sets, public 6-15
 SF.CLTC function 2-35
 SF.ENDT function 2-33
 SF.EXIT function 2-34
 SF functions list 2-36
 SF.LIST function 2-36
 SF.REGR function 2-31
 SF.RERN function 2-37
 SF.SLTC function 2-35

- SF.STAT function 2-34
- SF.SWPI function 2-35
- SF.SWPO function 2-35
- SF.TIME function 2-32
- SF.WRIT function 2-34
- SFCALL function codes 2-30
- SFCALL macro 2-28
- Shared set 4-33
- Short PRU 5-5
- Skip backward directive 8-19
- Skip forward directive 8-19
- SKIPB directive 8-19
- SKIPB macro 5-17,26
- SKIPF directive 8-19
- SKIPF macro 5-16,25
- SMT (refer to Set member table)
- SNT (refer to Spot name table)
- Source file 7-11,14
- Source ID 4-3
- Special dump directives 11-5
- Special file names 4-2,3,31
- SPM 5-21
- SPM communication order codes 5-26
- SPM-ISP interface 5-30
- SPM-ISQ interface 5-32
- SPM-3DO interface 5-38
- Spot name table C-27
- Spun-off task C-27
- SR directive 11-6
- SR.READ function 2-34
- SSCT macro 2-39
- Stack entry, invalid 5-35
- Stack processor 5-21
 - Error conditions 5-34
 - Order codes 5-24,26
 - Recall 3-33
- Stack processor-system interface 5-27
- Stack request dump 11-6
- Stack request formats
 - External 5-23
 - Internal 5-23
- Standard NDR reply format H-9
- Standard NLDR request format H-9
- Start NDR (NDR function) H-18.2
- State H-44
- Status control register dump 11-5
- Status Event (RHF queue entry type) H-35
- Status (NDR function) H-11
- Status of equipment 7-20
- Status of job 2-34
- Status of tape drive 4-30
- Status, permanent file 6-5
- Status word (Nsup) H-5,6
- STF routine 2-13
- STG 4-28
- Storage moves 2-3
- Storage requests 3-18
- Subchannel table C-25
- Subdirectory 6-3
- Subdirectory table D-22
- Subpage buffer B-147
- Subsystem
 - Control table 2-35; B-122
 - Definition 2-23,39
 - Functions 2-28,35
 - Swap-in 2-35
 - Swap-out 2-35
 - Task complete 2-33
- SUMMARY control statement 7-14
- SUP bulletin 9-4
- Supervisory Messages H-4
- Swap file format D-26
- Swap-in 2-35; 7-6
- Swap-out 2-2,34; 7-6
- SWITCH control statement 7-13
- Symbol 8-1
- Symbol definition F-1
- SYSBULL control statement 9-1
- System area (extended memory) 1-2
- System bulletin 9-1
- System bulletin file
 - Creating 9-3
 - Reducing 9-3
 - Updating 9-3
- System circular buffer 3-31; 5-41,42; B-146
- System communication 2-4
- System control point 2-23
 - Abnormal termination 2-38
 - Accounting 2-30,32
 - Address 2-28,32
 - Exit 2-34
 - Normal termination 2-38
 - Request count 2-35
 - Wait response count 2-35
- System dayfile 4-1,3,31; 11-2
- System dayfile area 4-23
- System dayfile dump 11-2
- System deadlock 4-30
- System device 4-32; 5-18
- System directory 8-6
- System directory, move 8-24
- System directory, replace 8-7
- System dumps 11-1
- System dynamic dump 11-8
- System dynamic dump file 11-8
- SYSTEM EDITLIB 8-1,5
- System exchange package dump 11-5
- SYSTEM file name 8-7
- System files 2-2; 4-1
- System files, EDITLIB 8-22
- System image 2-18,19; 10-2
- System libraries 1-1,4; 8-1
- System libraries, create 8-1
- System loading 1-4
- SYSTEM macro 6-15; 7-3
- System monitor (also refer to Monitor) 1-1,3; 3-12; 5-30

System pages E-7
 System programs 5-1
 System requests 1-1
 System security
 EDITLIB 8-23
 LDCMR 10-4
 System set 4-31
 System symbols F-1
 System table find 2-13
 System tape 1-5; 4-1

T.IJCT 2-35
 T.SSCT 2-35
 Tape drive
 Assignment 4-28
 Automatic assignment 4-29
 Overcommitment 4-30
 Physical status 4-30
 Scheduling 4-29
 Status checking 4-30
 Tape drivers 5-5
 Tape error recovery drivers 5-5
 Tape job prescheduling 4-29
 Tape job prescheduling display 4-29; 7-1
 Tape job scheduling 7-1
 Tape jobs, loading 7-2
 Tape operations 5-5
 Tape unit recovery table B-109
 Tape, unlabeled 4-29,31
 TAPES directive 11-2
 Tapes staging table 4-28; B-87
 TAPES table 11-2; B-108
 TBT (refer to Transfer buffer table)
 Time accounting 3-18
 TLOAD routine 7-2
 Trace mode 3-35
 Transfer buffer table 5-43,52,53; E-13,16
 TRANSFER directive 8-20
 Transient PP program 3-5
 TRANSPF utility 6-5
 TRANS77 directive 8-20

UCP (refer to User control point)
 UCP Asynchronous Supervisory Message Request
 (RHF queue entry type) H-31
 UCP Message Text for SSF Call (RHF queue
 entry type) H-32
 UCP NETON Reply Words (RHF queue entry
 type) H-32
 UCP nsup Reply (RHF queue entry type) H-30
 UCP SSF Call Entry (RHF queue entry
 type) H-32
 UCP Swap In Request (RHF queue entry
 type) H-30
 UCPA 2-28
 UCPA (refer to User control point address)

UEM (refer to Unified extended memory)
 UFL directive 11-5
 Unallocated memory 2-3
 Uncorrectable parity error 5-35
 Unified extended memory v
 Unit queue table 5-45,53; B-92
 Unit record I/O 7-3
 Unit recovery table B-109
 Unlabeled tapes 4-29,31
 Updating system bulletin file 9-3
 UQT (refer to Unit queue table)
 URT (refer to Unit recovery table)
 User area 10-3
 User control point 2-23
 Abort 2-31
 Address 2-28
 End processing 2-37
 Message 2-31
 Read 2-34
 Write 2-34
 USER EDITLIB 8-1
 User field length 2-4
 User field length dump 11-5
 User files, EDITLIB 8-22
 User library 8-1,23
 User library, modify 8-1
 User library, random 8-18
 User library, sequential 8-18
 User mode 3-13
 User mode segment name 11-5
 User program 5-1; 7-18
 Utility interlocks 6-2

Validate device set 6-12
 Validate RBT chains 11-5
 Validate UCP (RHH function) H-20
 Variable section 4-26
 Versions, permanent files 6-3
 Vocabulary word 8-1
 Volume serial number 4-29,31
 VSN buffer B-107
 VSN (refer to Volume serial number)

Wait response count, SCP 2-35
 WO directive 11-6
 Word 0 C-5
 Word zero, match 11-6
 Words, central memory 1-2
 WPHR 5-16
 WPPF 12-6
 Write calls 5-38
 WRITE macro 5-11,25
 Write order codes 5-24
 Write Path Data (NDR function) H-15

WRITEC macro 5-20,25
WRITEF macro 5-13,25
WRITEN 5-15
WRITER macro 5-12,25

XFRHOLD (File Transfer Common Area) H-47
XP directive 11-5

ZZZZCMR file 4-1; 10-1,2
ZZZZDD file 11-8
ZZZZ04 file 4-31; 5-18

ZZZZ06 file 4-2
ZZZZ23 file 4-2,31; 8-6

IPK functions 5-39
PK routine 7-10
ISP 5-21,30
ISQ 5-22,32
3DO 5-38
819 disk I/O processing 5-43
 Logical I/O processing 5-43
 Physical I/O processing 5-47
 Tables 5-52

COMMENT SHEET

MANUAL TITLE: CDC NOS/BE Version 1 System Programmer's
Reference Manual, Volume 1

PUBLICATION NO.: 60494100

REVISION: U

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE: _____ ZIP CODE: _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

Please Reply

No Reply Necessary

LD

FOLD



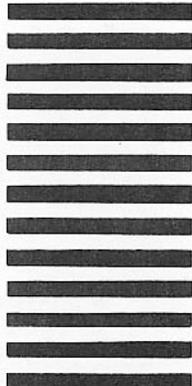
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55126-6198



CUT ALONG LINE

D

FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD