

MANUAL CHANGE NOTICE

MANUAL IDENTIFICATION

Part Number: 92060-90013

Print Date: MAY 1978

Title: Batch-Spool Monitor
Reference Manual

Library Index No. 92060.320.92060-90013

CHANGE IDENTIFICATION

Change Number: 3

Print Date: DEC 1978

Software Revision Code: 1826



THE PURPOSE OF THIS MANUAL CHANGE

is to accumulate all changes to the current edition of the manual. Earlier changes, if any, are contained herein for your convenience. *(If you have made all previous changes to this manual, you need only make the changes described under the change number indicated above.)*

CHANGED PAGES ARE IDENTIFIED

by the change number at the bottom of the page and a vertical line (change bar) in the outside margin to indicate the area of the text that has been changed..

NEW PAGES ARE IDENTIFIED

by the change number at the bottom of the page. "New" pages are those which were not present when the current edition of the manual was published.

TO UPDATE YOUR MANUAL

locate the Change Number, indicated above, on the back of this page and follow the instructions provided.



TECHNICAL MANUAL CHANGES
(92060-90013)

CHANGE 1:

Reason for Change 1: Change manual to reflect update of software to 1826 revision code.

- A. Replace changed pages with update pages. Insert any new pages in sequence. Destroy all replaced pages.
- B. Page 2-0 "Index to FMGR Operator Commands" change page reference for "TR" command:
 - From 2-58
 - To 2-59

CHANGE 2:

Reason for Change 2: Change manual to incorporate technical enhancements to the documentation.

- A. Replace changed pages with update pages.

CHANGE 3:

Reason for Change 3: Change manual to incorporate technical enhancements to the documentation.

- A. Replace changed pages with update pages.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

MANUAL CHANGE NOTICE

MANUAL IDENTIFICATION

Part Number: 92060-90013

Print Date: MAY 1978

Title: Batch-Spool Monitor
Reference Manual

Library Index No. 92060.320.92060-90013

CHANGE IDENTIFICATION

Change Number: 1

Print Date: JUNE 1978

Software Revision Code: 1826

THE PURPOSE OF THIS MANUAL CHANGE

is to accumulate all changes to the current edition of the manual. Earlier changes, if any, are contained herein for your convenience. *(If you have made all previous changes to this manual, you need only make the changes described under the change number indicated above.)*

CHANGED PAGES ARE IDENTIFIED

by the change number at the bottom of the page and a vertical line (change bar) in the outside margin to indicate the area of the text that has been changed..

NEW PAGES ARE IDENTIFIED

by the change number at the bottom of the page. "New" pages are those which were not present when the current edition of the manual was published.

TO UPDATE YOUR MANUAL

locate the Change Number, indicated above, on the back of this page and follow the instructions provided.



TECHNICAL MANUAL CHANGES
(92060-90013)

CHANGE 1:

Reason for Change 1: Change manual to reflect update of software to 1808 revision code.

- A. Replace changed pages with update pages. Insert any new pages in sequence. Destroy all replaced pages.
- B. Page 2-0 "Index to FMGR Operator Commands" change page reference for "TR" command:
 - From 2-58
 - To 2-59

Yes done
28/11/73

Batch-Spool Monitor

Reference Manual



HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

Library Index Number 92060.320.92060-90013

PUBLICATION NOTICE

Information in this manual describes the BATCH-SPOOL MONITOR software. Changes in text to document software updates subsequent to the initial release are supplied in manual update notices and/or complete revisions to the manual. The history of any changes to this edition of the manual is given below under "Publication History." The last change itemized reflects the software currently documented in the manual.

Any changed pages supplied in an update package are identified by a change number adjacent to the page number. Changed information is specifically identified by a vertical line (revision bar) on the outer margin of the page.

PUBLICATION HISTORY

Third Edition	May 78 (Software Rev. Code 1805)
Change 1	Jun 78 (Software Rev. Code 1826)
Change 2	Oct 78 (Software Rev. Code 1826)
Change 3	Dec 78 (Software Rev. Code 1826)

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

PREFACE

The Batch-Spool Monitor provides file management and batch job processing capabilities through the File Management Package; it provides spooled batch processing or spooled I/O with the Spool Monitor. The File Management Package can be used independently of the Spool Monitor, or they can be combined to extend the capabilities of the Batch-Spool Monitor.

The Batch-Spool Monitor is provided as part of the Real-Time Executive III operating system; it is an option of the Real-Time Executive II operating system.

This manual gives the reference specifications for the entire Batch-Spool Monitor. It is organized functionally so that it can be used by programmers new to the Batch-Spool Monitor. An index to each command and call is provided at the start of each section so that experienced users may quickly reference the command or call formats.

The manual is organized as follows:

- Section I System overview describing each component of the system and its functions plus a general description of files and cartridges used by the File Management Package.
- Section II Describes batch job and file control through FMGR commands by means of the command syntax and an explanation of command usage with examples.
- Section III Describes file management through FMP program calls by means of the syntax of each call and an explanation of usage with examples.
- Section IV Describes how to use the Spool Monitor, how to enter jobs for spooling and how to spool the output from jobs; command syntax with an explanation of usage and examples is provided for the FMGR spooling commands and the JOB command, XE.
- Section V Describes spool control through the GASP commands by means of command syntax, explanation of usage, and examples.
- Section VI Describes spool control through the SMP calls by means of the call syntax and an explanation of usage with examples.
- Section VII Describes how to configure the File Management Package and Spool Monitor and how to initialize each of these subsystems.
- Appendices Five appendices are provided that supply general information to the user of the Batch-Spool Monitor:
 - A HP Character Set
 - B Error Codes for FMP, FMGR, and GASP
 - C Table, Directory, and Record Formats
 - D Cartridge Formatting
 - E Making Copies of FMGR
 - F Global Equivalence Table

Anyone using the Batch-Spool Monitor should be familiar with the RTE operating system. For the RTE-II user, this system is described in the manual:

Real-Time Executive II Software System Programming and Operating Manual Part Number 92001-93001

The RTE-III operating system is described in the manual:

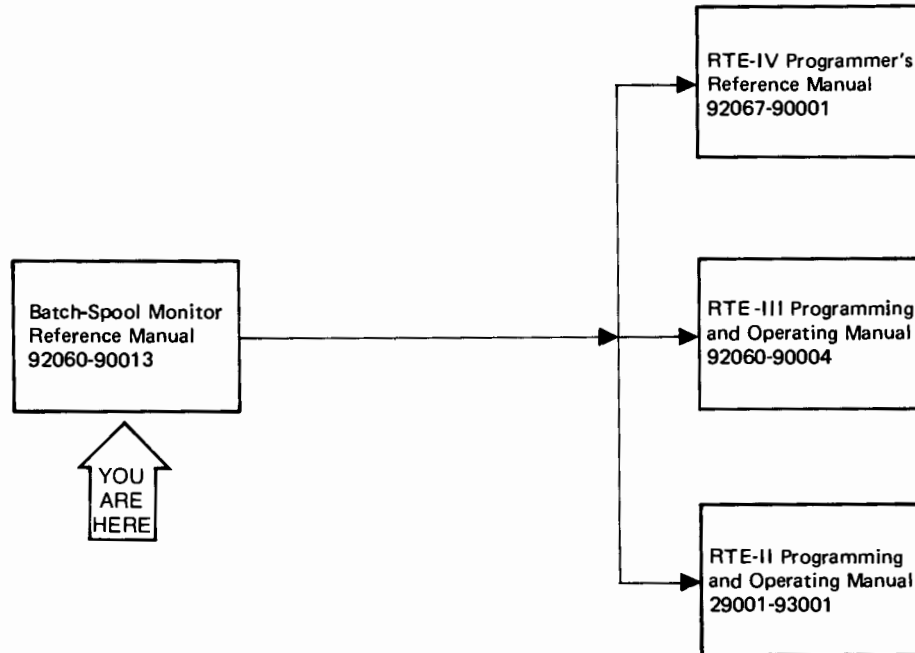
Real-Time Executive III Software System Programming and Operating Manual Part Number 92060-90004

The RTE-IV operating system is described in the manual:

Real-Time Executive IV Programmer's Reference Manual Part Number 92067-90001

Other manuals of interest may be found in the Documentation Map in the beginning of the appropriate Operating System Reference Manual.

DOCUMENTATION MAP



Refer to the Documentation Map in the appropriate Operating System Reference Manual for a complete listing of all relevant manuals.

CONTENTS

Section I	Page	Section II (Continued)	
SYSTEM OVERVIEW		RU - Run Program	2-48
INTRODUCTION	1-1	OF - Remove Program	2-51
Features	1-1	RT - Release Tracks	2-52
Batch-Spool Environment	1-1	Program Development Example	2-53
Batch-Spool Monitor Components	1-3	PROCEDURE FILE MANIPULATION	2-54
Batch-Spool Monitor Functions	1-4	Global Parameters	2-54
FILE MANAGEMENT	1-5	TR - Transfer Control	2-59
File Organization	1-5	PA - Pause and Send Message	2-62
Disc File Allocation	1-6	DP - Display Parameters	2-63
Directories	1-7	SE - Set Global Parameters	2-64
File Types	1-8	CA - Calculate Globals	2-65
File Access	1-10	IF - Conditional Skip	2-67
Security	1-10	BATCH JOB CONTROL	2-70
FMP Program Calls	1-11	JO - Start of Job	2-73
FMGR Operator Commands	1-12	EO - End of Job	2-74
BATCH PROCESSING	1-13	TL - Set Run Time Limit	2-76
SPOOLING	1-14	LU - Logical Unit Switch	2-77
GASP Operator Commands	1-15	AB - Abort Job	2-78
SMP Program Calls	1-15	FMP CARTRIDGE MANIPULATION	2-80
		Locking a Cartridge	2-81
Section II	Page	MC - Mount Cartridge	2-81
BATCH AND FILE CONTROL WITH FMGR		IN - Initialized Cartridge	2-83
OPERATOR COMMANDS		DC - Dismount Cartridge	2-88
INTRODUCTION	2-1	CL - Cartridge List	2-90
Interactive vs. Batch Operation	2-1	DL - Directory List	2-91
Multi-Terminal Environment	2-2	PK - Pack Cartridge	2-92
Interrupting FMGR	2-3	CO - Copy One Cartridge to Another	2-95
FMGR COMMANDS	2-5	MISCELLANEOUS COMMANDS	2-96
Command Structure	2-6	SY - Execute RTE System Command	2-96
Parameter Syntax Rules	2-7	** - Comments	2-97
Namr Parameter	2-8		
FMGR OPERATION	2-10	Section III	Page
Running Program FMGR	2-11	FILE MANAGEMENT THROUGH FMP CALLS	
?? - Request Error Explanation	2-13	INTRODUCTION	3-1
EX - Terminate FMGR	2-14	FMP Calls	3-1
LI - Change List Device	2-14	Data Control Block	3-1
LO - Change Log Device	2-15	PROGRAM CALLS	3-4
SV - Change Severity Code	2-16	Common Parameters	3-4
TE - Send Message to Console	2-17	Optional Parameters	3-6
AN - Send Message to List Device	2-18	FILE DEFINITION	3-8
FILE CREATION AND MANIPULATION	2-18	CREAT	3-8
CR - Create Disc File	2-19	PURGE	3-12
CR - Create Non-Disc File	2-20	OPEN	3-13
PU - Purge a File	2-22	CLOSE	3-18
ST - Transfer Data and Create File	2-23	FILE ACCESS	3-21
DU - Transfer Data to Existing File	2-28	READF	3-21
LI - List File Contents	2-31	WRITF	3-26
CN - Control Non-Disc Device	2-33	FILE POSITIONING	3-31
RN - Rename File	2-35	LOCF	3-31
PROGRAM FILE MANIPULATION	2-36	APOSN	3-34
MS - Move Source File	2-38	POSNT	3-36
LS - Set Logical Source Pointer	2-40	RWNDF	3-39
LG - Assign LG Tracks	2-41	SPECIAL PURPOSE ROUTINES	3-41
MR - Move Relocatable Program	2-42	FCONT	3-41
SA - Save Program as a File	2-43	FSTAT	3-44
SP - Save Program	2-44	IDCBS	3-46
RP - Restore Program	2-46	NAMF	3-47

CONTENTS (continued)

Section III (continued)	
POST	3-48
EXAMPLE USING FMP CALLS	3-51
Section IV	Page
USING THE SPOOL MONITOR	
INTRODUCTION	4-1
Batch Spooling	4-1
Timing Considerations	4-4
Spool Files	4-4
Logical Unit Switching	4-5
SPOOL SETUP	4-8
JO - Initiate Job For Spooling	4-9
EO - End of Spooled Job	4-10
LU - Spool Set Up and Outspool	
Control	4-10
CS - Change Spool Setup	4-15
INSPOOLING	4-17
Running Program JOB	4-17
XE - Job Input Control	4-21
Error Conditions	4-23
OUTSPOOLING	4-24
SPOOL STATUS	4-25
Inspool Status	4-25
Outspool Status	4-25
OUTSPOOL ERRORS	4-26
Section V	Page
SPOOL CONTROL WITH GASP	
OPERATOR COMMANDS	
INTRODUCTION	5-1
GASP commands	5-1
Running Program GASP	5-2
?? - GASP Error Explanation	5-3
EX - Terminate GASP	5-3
JOB MANIPULATION	5-4
DJ - Display Job Status	5-4
CJ - Change Job Status	5-6
AB - Abort Job	5-6
OUTSPOOL MANIPULATION	5-7
DS - Display Spools	5-7
CS - Change Spool Status	5-8
RS - Restart Spool	5-10
Spool File State Diagram	5-10
KS - Kill Outspool	5-11
SPOOL SYSTEM MANIPULATION	5-14
SD - Shut Down Spooling	5-14
SU - Start Up Spooling	5-15
DA - Deallocate Spooling	5-16
COMPREHENSIVE SPOOLING EXAMPLE	5-17
Section VI	Page
SPOOL CONTROL THROUGH SMP CALLS	
INTRODUCTION	6-1
SPOOL SETUP	6-1
Setup with SPOPN	6-2
SPOOL CONTROL	6-5
Change Purge to Save	6-5
Change Save to Purge	6-6
Queue for Outspooling	6-7

Section VI (continued)	
Write EOF and Queue for Outspooling	6-7
Change Spool Options	6-8
Set Buffer Flag	6-8
Clear Buffer Flag	6-9
SPOOL POSITIONING	6-10
Retrieve Record Position	6-10
Change Record Position	6-10
SPOOL CALL EXAMPLE	6-12

Section VII	Page
FMP AND SM CONFIGURATION AND INITIALIZATION	
INTRODUCTION	7-1
FMP CONFIGURATION	7-1
Parameter Input Phase	7-1
FMGR Initialization	7-3
SM CONFIGURATION	7-4
Parameter Input Phase	7-4
Table Generation Phase	7-6
Partition Definition Phase	7-8
GASP Initialization	7-9

Appendix A	Page
HP CHARACTER SET	A-1

Appendix B	Page
BATCH-SPOOL MONITOR ERROR CODES	B-1
FMP AND FMGR ERROR CODES	B-1
GASP ERROR CODES	B-10
UNNUMBERED ERROR MESSAGES	B-12
SPOOL ERRORS	B-14
Spool I/O Abort Errors	B-14
SMP Error Messages	B-15
Outspool Error Messages	B-16

Appendix C	Page
TABLES, DIRECTORIES, AND RECORD FORMATS	C-1
STANDARD LOGICAL UNITS	C-1
Standard Driver Types	C-1
DATA CONTROL BLOCK FORMAT	C-2
CARTRIDGE DIRECTORY FORMAT	C-4
FILE DIRECTORY FORMAT	C-5
JOBFIL FORMAT	C-7
SPLCON FORMAT	C-9
DISC FILE RECORD FORMATS	C-12
NON-DISC FILE RECORD FORMATS	C-14

Appendix D	Page
CARTRIDGE FORMATTING	D-1

Appendix E	Page
MAKING COPIES OF FMGR	E-1
MAKE SINGLE PROGRAM COPY	E-1
MAKE MULTIPLE COPIES AT SYSTEM START-UP	E-2

Appendix F	Page
GLOBAL EQUIVALENCE TABLE	F-1

ILLUSTRATIONS

Title	Page	Title	Page
Batch-Spool Monitor Components	1-3	Data Transfer With Type 1 Files	3-3
Relation Between Batch-Spool Monitor Components	1-4	Read Type 1 File When IL Greater Than 128	3-22
Disc Organization for the Batch-Spool Monitor	1-7	Write Type 1 File When IL Greater Than 128	3-27
Spooled Batch Processing	1-14	Batch Spooling Diagram	4-3
Interactive vs. Batch Operation	2-2	Logical Unit Switching	4-5
Relation of Files to Subfiles	2-26	Association of Logical Unit to Driver	4-6
Compile and Execute Program	2-53	Relation Between LU Switch Table, DRT, and EQT	4-7
Disc Platter With Three Cartridges	2-87	Outspool File State Diagram	5-11
Two Types of Directory List	2-93	GASP Initializes Spooling System	5-17
Packing a Cartridge	2-95	Jobs X, Y, Z	5-18
Sequential Transfer Between Disc File and Buffers	3-3	Program JOB Used to Inspool Jobs, X, Y, and Z ...	5-18
		Command Stream and Use of GASP	5-19

TABLES

Title	Page	Title	Page
Batch-Spool Monitor Programs and Routines	1-3	JOB Error Messages	4-24
File Management Terms	1-5	GASP Operator Command Summary	5-2
Categories of File Types	1-8	IBUFR Format	6-3
FMGR Operator Command Summary	2-5	Spool Control Calls	6-5
Global Equivalence	2-57	Standard Program Parameters	7-4
G and S Global Format	2-58	FMP Error Codes	B-2
FMP Call Summary	3-2	Relation Between FMP Error Codes and FMP Calls	B-4
Relation of Actual to Requested DCB Buffer Size	3-5	FMGR Error Codes	B-5
Effect of IL Parameter in READF	3-22	GASP Error Codes	B-9
Effect of IL Parameter in WRITF	3-27	Unnumbered Error Messages	B-12
Relation Between Parameters NUR and IR	3-36	Spool I/O Abort Error Codes	B-14
FCONT Function Codes	3-42	SMP Error Messages	B-15
ISTAT Format	3-45	Outspool Error Messages	B-16
		Octal Equivalence Table	F-1

SECTION I

SYSTEM OVERVIEW



1-1. INTRODUCTION

The Batch-Spool Monitor is a software package operating under control of the Real-Time Executive Operating System to provide file control, program development, job control, and job control with spooling. The Batch-Spool Monitor consists of two separate but related subsystems: the File Management Package and the Spool Monitor. The file management, program control, and batch job processing capabilities are contained in the File Management Package (FMP). The spooling function is contained in the Spool Monitor (SM).

1-2. FEATURES

With the Batch-Spool Monitor, the user of a Real-Time Executive Operating System may perform the following tasks:

- Create named disc files at a terminal or from a program
- Create and access files by name rather than by disc address
- Treat peripheral non-disc devices as files for I/O control
- Transfer data from file to file, from device to device, or between a file and a device
- Control I/O to and from files or devices with program calls
- Prepare and test programs from a terminal or in a batch job stream
- Control batch job processing from a terminal or file using an extensive job control language
- Enter jobs in the batch stream by priority and with time limits
- Output data from jobs by priority
- Automatically spool job input and output
- Control spool input and output through operator commands or program calls
- Safeguard file integrity and security in a multiprogramming environment

1-3. BATCH-SPOOL ENVIRONMENT

Software Environment

The Batch-Spool Monitor runs under control of any of the following operating systems:

- Real-Time Executive II (RTE-II)
- Real-Time Executive III (RTE-III)
- Real-Time Executive IV (RTE-IV)

System Overview

Throughout the rest of this manual, the generic term RTE will be used to refer to the above operating systems.

With RTE-II, the Batch-Spool Monitor is an option; with RTE-III and RTE-IV, it is provided with the system. If provided, the Spool Monitor need not be configured in the Batch-Spool Monitor. The File Management Package (FMP) will run equally well on systems that do not include the Spool Monitor (SM). The Spool Monitor, on the other hand, will not run without the File Management Package.

Refer to Section VII on FMP and SM Configuration and Initialization for memory requirements for these subsystems.

Hardware Environment

The Batch Spool Monitor operates within the RTE hardware environment consisting of an HP 1000-Series Computer system.

For RTE-II, the File Management Package can operate within the minimum system on a 2100 or 21MX computer. When the Spool Monitor is added, an additional 8K of memory is required. Refer to the RTE-II Programming and Operating Reference Manual for more information on the minimal system for RTE-II.

For RTE-III and RTE-IV, both the File Management Package and the Spool Monitor can operate within the minimum system on a 21MX computer. Refer to the appropriate Programming and Operating Reference Manual for information on the minimal system for RTE-III and RTE-IV.

With any operating system, a system using the Spool Monitor should also include:

- a line printer.
- a disc platter reserved for spool files.

CAUTION

To avoid disruption of data stored on disc, the disc controller and removable disc platters must not be exposed to areas with a strong magnetic field.

1-4. BATCH-SPOOL MONITOR COMPONENTS

The components of the Batch-Spool Monitor are illustrated in Figure 1-1.

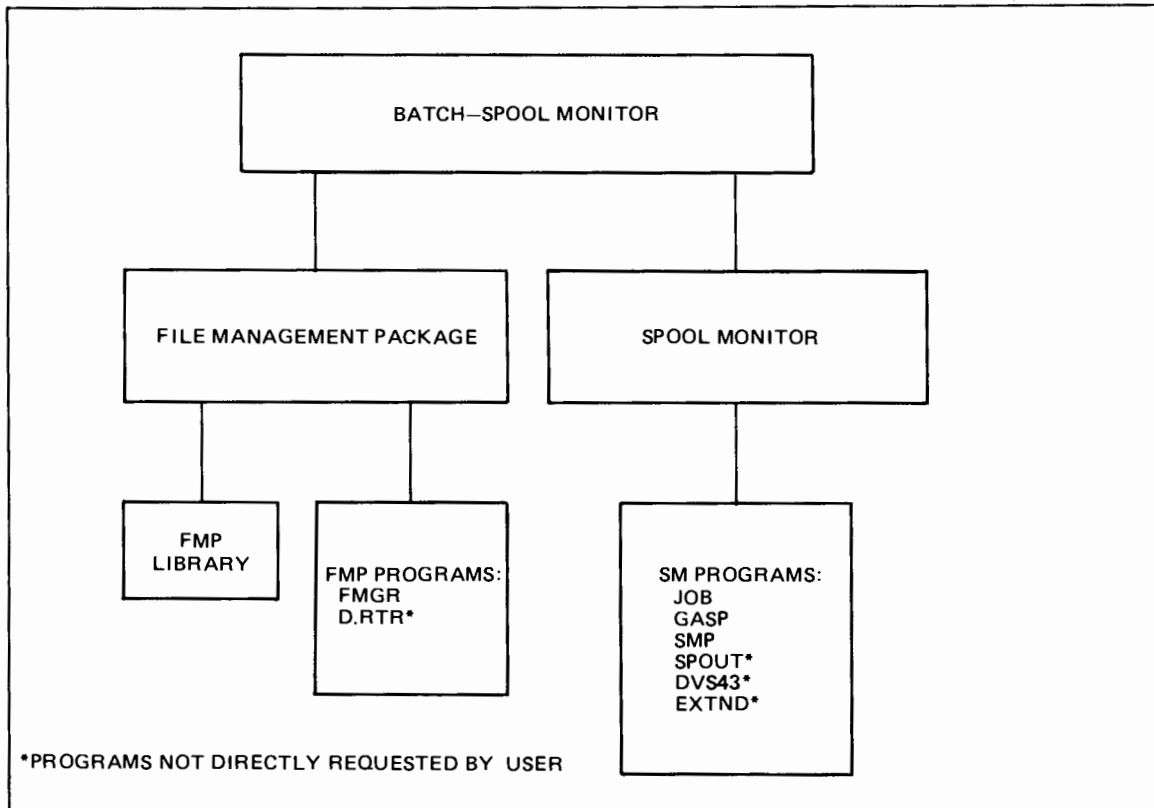


Figure 1-1. Batch-Spool Monitor Components

The File Management Package controls file input and output through program calls to the FMP library. It also manipulates files and processes batch jobs by means of operator commands to the program FMGR.

The Spool Monitor controls the entry of batch jobs (inspooling), job processing, and job output (outspooling) according to priorities assigned to each job. With minimal operator intervention, the Spool Monitor provides highly efficient use of the FMGR batch job processing capabilities. The Spool Monitor also provides spooling of program input and output apart from batch processing.

Table 1-1. Batch-Spool Monitor Programs and Routines

SUBSYSTEM	PROGRAM	FUNCTION
FMP	FMGR	is the operator interface for file management and batch processing. The FMGR commands are entered by an operator from a console or from a previously prepared procedure file.
	D.RTR	is the only program that writes on the file directories used by FMP for file control. To insure file security and integrity, D.RTR may be scheduled only by a system program or subroutine; it is never requested directly by a user.
	FMP Library	Utility routines that may be called from a user program or FMGR to perform file input and output as well as other file related operations.

Table 1-1. Batch-Spool Monitor Programs and Routines (Continued)

SUBSYSTEM	PROGRAM	FUNCTION
SMP	JOB	provides the inspool capability for batch jobs. It reads jobs from an input device and places them on a disc file or reads the jobs directly from disc. JOB places each inspoiled job in a queue of jobs to be processed.
	GASP	provides a set of commands to allow operator control of the spooling process.
	SMP	maintains information on all active spools in the system and coordinates job processing with outspooling. Program calls to SMP control spooled input and output apart from job processing and inspooling.
	SPOUT	writes the output from completed or active jobs to specified output devices. SPOUT is never directly requested by a user.
	DVS43	is the driver routine used to implement disc accesses for spooling. It is independent of user control.
	EXTND	controls the automatic assignment of extents to spool files as needed. It operates without user intervention.

The relation between the Batch-Spool Monitor programs, the FMP library, and the files they control is illustrated in Figure 1-2. Only those programs are shown that may be requested by an operator command or from a user program.

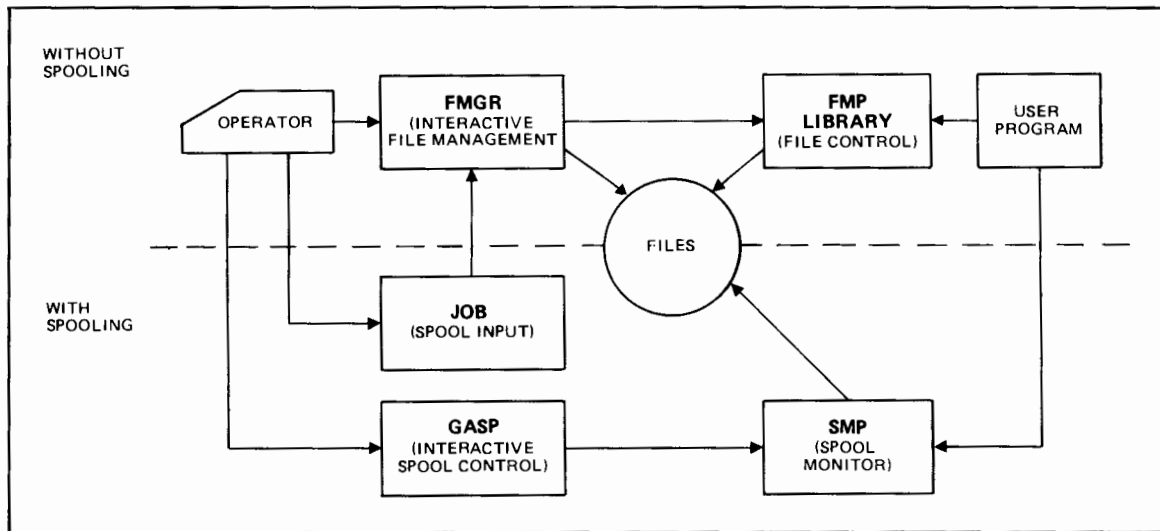


Figure 1-2. Relation Between Batch-Spool Monitor Components

1-5. BATCH-SPOOL MONITOR FUNCTIONS

The three main functions performed by the Batch-Spool Monitor are:

- File Management
- Batch Processing
- Spooling

File management is the primary function of the Batch-Spool Monitor. Batch job processing depends on the file processing capabilities of program FMGR and spooling extends these batch job processing capabilities.

1-6 FILE MANAGEMENT

File management is performed through program calls to the FMP library and by interactive operator commands to the program FMGR. The FMP calls mainly control input to and output from disc files or peripheral devices treated as files. The file management capability is increased by using FMGR for interactive program development, disc cartridge manipulation, and batch job control.

1-7. FILE ORGANIZATION

Files are a collection of information logically organized into records. The information in files may be programs or the data used by programs. Data may be binary or in ASCII code. Programs may be in the form of ASCII source code or binary code in either relocatable or absolute form. Programs may also be in memory-image form, a form used by RTE for programs ready to be executed.

Files may be stored on disc or they may refer to non-disc peripheral devices. The Batch-Spool Monitor is used to control and access files whether they are disc files or non-disc devices.

Certain terms used in discussing file organization for the Batch-Spool Monitor are defined in Table 1-2.

Table 1-2. File Management Terms

DISC	<p>A rotating random access storage device on which files may be stored and from which they may be retrieved. Discs may be physically permanent or they may be removable. The system disc on logical unit 2 contains the RTE operating system, and may contain FMP files. Peripheral discs on logical units greater than 6 are usually removable and may contain FMP files.</p> <p>The RTE Disc driver names must be DVn30, DVn31, DVn32, DVn33 with corresponding Initiator and Continuator entry points (In30,Cn30; In31,Cn31; In32,Cn32; In33,Cn33).</p> <p>Where: <i>n</i> is any character (normally, "R" for driver names and "." for entry points, e.g., driver name DVR32 and entry point names I.32 and C.32).</p>
CARTRIDGE	<p>Sometimes used as another name for disc. In FMP terms, it is a set of contiguous tracks on a disc platter. Cartridges contain disc files with a directory of the files stored on each cartridge. All files on the same FMP cartridge must have unique names.</p>
TRACK	<p>A subdivision of the disc</p>
SECTOR	<p>A further subdivision of the disc; specifically, a sector is a portion of a disc track consisting of 64 words. Two sectors make up one FMP block, the unit of information that may be physically transferred between the disc and memory.</p>
FILE	<p>A disc file is a collection of records terminated by an end-of-file mark. Non-disc devices are treated by the Batch-Spool Monitor as if the device were itself a file. The device, like a file, is a collection of records terminated by an end-of-file mark that depends on the device. Any file can have zero or more records and is designated by a name of six characters or less.</p>

Table 1-2. File Management Terms (continued)

NON-DISC DEVICE	A peripheral device, not a disc, that may be treated as a file and controlled by the File Management Package. Non-disc devices include the paper tape or card reader for input, the paper tape punch or line printer for output, magnetic tape for input or output, and terminals through which an operator can interact with the system.
LOGICAL UNIT	An integer assigned to each input-output device or disc cartridge at system generation by which the cartridge or device can be referenced.
BLOCK	A subdivision of a disc file containing 128 words (two disc sectors) that is the smallest unit that can be physically transferred between disc and central memory during file access.
RECORD	A logical collection of 16-bit words on a file or device that is terminated by an end-of-record mark. A record may have zero or more words. A record is the smallest unit that may be accessed by a call from a program.
EXTENT	An extension to a file automatically provided by FMP as needed. Each extent is the same length and type as the file and is identified by a positive integer called the extent number.
DCB BUFFER	FMP uses an array called the Data Control Buffer both as a directory to the file being accessed and as a buffer for data transfer between the file and memory. Space must be allocated for the Data Control Block in any program making an FMP file access call.
USER BUFFER	An area in the calling program to hold one record at a time during file access.
ID SEGMENT	A directory in the system area of the disc to a program that may be brought into memory for execution. The number of ID segments specified at generation directly limits the number of user programs that may be in memory at one time.

1-8. DISC FILE ALLOCATION

Disc files managed by the Batch-Spool Monitor, whether they are program files, data files, or spool files, are kept on FMP cartridges. An FMP cartridge is a logical entity that may correspond directly to a disc platter or may be a subdivision of the disc platter. On some discs, a cartridge may even cross platter boundaries although care must be taken if the cartridge is partly on a removable platter and partly on a fixed platter.

Each cartridge is defined by a beginning and an ending track, and is assigned a logical unit number and a cartridge reference number either of which may be used to reference the cartridge. Files on the same cartridge must have unique names. Duplicate file names may be used as long as the duplicates are on separate cartridges so the file can be uniquely identified by its name and a cartridge identifier.

Files are located on contiguous tracks on an FMP cartridge. User files, including spool files, begin in the lowest numbered track and work up. Directory entries for user files begin in the highest numbered track and work down. Removable cartridges containing FMP files are interchangeable between drives within a system, or between drives on different systems provided that logical track 0 refers to the same physical track on every disc unit. (Refer to Figure 1-3 for an illustration of disc organization using one cartridge on the system disc starting at the first FMP track, and one on a peripheral disc starting at track 0.)

Whenever a cartridge is mounted, the last FMP track on the disc should be known in order to assign the last track for the cartridge directory. At cartridge initialization, the number of

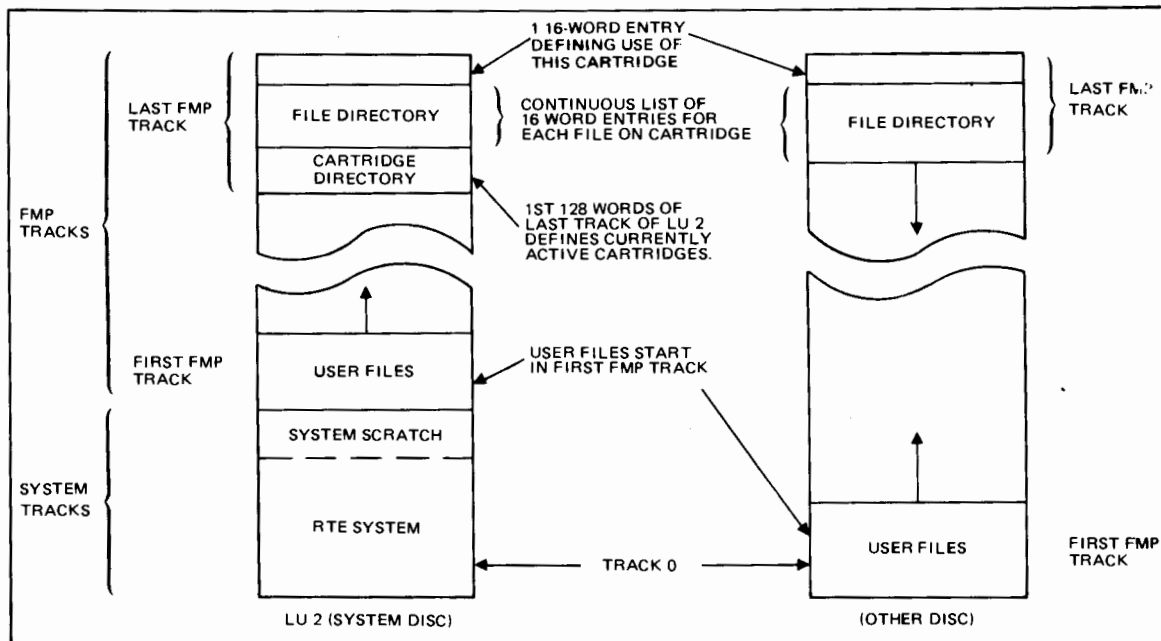


Figure 1-3. Disc Organization for Batch-Spool Monitor

directory tracks for that cartridge is specified. The first cartridge track must be assigned at initialization; the number of sectors per track may be specified at this time, but is supplied by FMP as a default if not. Otherwise, track locations are not required for cartridge specification.

Files may cross track boundaries but, in a multi-cartridge environment, no one file may cross cartridge boundaries. Files are subject to being moved whenever a cartridge is packed. This causes files to be relocatable within a cartridge and no absolute file addresses should be kept in any file or program.

Files always start on even sector boundaries and all accesses are multiples of 128 words addressed to even sectors.

Disc errors are passed back to the user for action. Error codes are printed on the system log device when using the File Manager operator commands, or passed to the user program when calling a File Management Package library routine. You may report bad tracks to the system through the FMGR Initialization command. Bad tracks discovered by the system result in an error return to the calling program.

1-9. DIRECTORIES

Two directories are created and maintained by FMP: the FMP cartridge directory on the system disc, and the file directory on each cartridge. Only program D.RTR can write to or modify the directories.

Cartridge Directory

The Cartridge directory is a master index to all active FMP cartridges. It is maintained in the first two sectors of the last track of the system disc (refer to Figure 1-3). It has an entry for all currently mounted cartridges with tracks assigned to FMP. The directory has room to describe up to 31 cartridges using four words for each. In addition, this directory keeps the master security code for the system. With this code, the security codes for the individual files in the File Management Package can be retrieved.

File Directory

A file directory, maintained on each cartridge, contains information on each file on that particular cartridge. Each directory starts in sector 0 of the last track available to FMP. The first 16-word entry in this directory contains label and track information for the cartridge itself. Each subsequent 16-word entry has information on a user file. The last entry is followed by a zero word. When a file is purged, the first word in the directory entry for the file is set to -1 to indicate that it is to be ignored. When the cartridge is packed, the directory entry for any purged file is cleared and the cartridge area where the file was located is overwritten by non-purged files wherever possible.

1-10. FILE TYPES

Eight file types are defined by the system. Additional types may be defined by the user. Only the first four types differ in format; all subsequent types differ only in the type of data FMP expects the file to contain. The file types may be divided into three categories as shown in Table 1-3. The first category contains one type, type zero. This type includes all non-disc devices defined as files and accessible by name. The second category contains two file types, types 1 and 2. These fixed-length record files are used for quick random access. The remaining file types all belong to the third category of files with variable-length records designed for sequential access. These file types may be extended automatically as needed; files in the first two categories may not be extended.

Table 1-3. Categories of File Types

CATEGORY	TYPE	DESCRIPTION
Control	0	Non-disc device files
Fixed-length, random access, non-extendable	1	Fixed-length 128-word record files
	2	Fixed user-defined record length files
Variable-length, sequential access, automatic extents	3	Variable-length records; any data type
	4	Source program file; ASCII
	5	Object program file; relocatable binary
	6	Executable program file; memory-image code
	7	Absolute binary
	8-32767	User-defined data format

Type 0 Files

Type 0 files are used to reference non-disc devices by name. They afford a measure of device independence in that the standard file commands can be used to control the device. A directory entry is made for the device as if it were a file. A type 0 file is created with a FMGR command, not with an FMP call. The File Directory entry for a file of this type contains special entries that specify logical unit number and the operations allowed on the particular device.

The record format of a type 0 file is determined by the device type. Refer to Appendix C for type 0 record formats on paper tape or card devices.

Type 1 Files

Type 1 files have fixed length records of 128 words. Because the File Management Package transfers data to and from disc in 128-word blocks, this file type allows direct access between disc and the user's buffer area in his program, thereby eliminating the need to go through a packing buffer (the Data Control Block). As a result, type 1 files have the fastest transfer rate. Any other file type, except type 0, may be opened and accessed as a type 1 file in order to take advantage of the faster transfer rate. However, if the files being transferred have less than 128-word logical records, the user must be able to recognize where his records begin and end within the 128 words, or if his records are longer, be able to work with part of a record at a time. The end-of-file is the last word of the last block. Type 1 files may cross track boundaries.

Type 2 Files

The record lengths of type 2 files are fixed, but the length is defined by the user at file creation. Like type 1 files, the end-of-file is the last word of the last block and files may cross sector or track boundaries. Only one logical record is transferred at a time, but unlike type 1 files, the transfer must go through a packing buffer (the Data Control Block). For this reason, files of type 2 and above have a slower transfer rate than type 1 files. To obtain access of the maximum number of records (32767), the record length must be 128 or a multiple of 128.

Type 3 Files

These files have variable length records, are extendable, and may contain data, source code, relocatable or absolute binary code. Only one logical record is transferred at a time and the transfer must be made through the packing buffer (Data Control Block). The first and last words of each record as written on disc always contain the number of words in the record (minus the two length words). A zero-length record consisting of two zero words can be used to separate groups of records into sub-files. The end-of-file is marked by a -1 as the first length word in the next record. Words following the end-of-file are undefined. However, FMP can write records beyond the end-of-file by replacing the end-of-file with a new record followed by an end-of-file mark.

Type 4 Files

This file type is the same as type 3, except the system expects these files to contain ASCII data. Typically, source program files are type 4.

Type 5 Files

Same as type 3 files, except the system assumes type 5 files contain relocatable binary code. Typically object program files are type 5.

Type 6 Files

This type file is the same as a type 3 file, except the system assumes it contains a program in memory-image format that is ready to run. Type 6 files are created by the Save Program (SP) command. These files are always accessed by FMP as type 1 files. The first two sectors of a type

System Overview

6 file are used to record ID segment information for the program. As a result, this file type can be used for programs that do not have a permanent ID segment.

Type 7 Files

Same as type 3 files, except the system expects type 7 files to contain absolute binary code.

Type > 7 Files

Same as type 3, but the content is user-defined. FMP does no special processing based on file type for types greater than 7. For instance, any checksums must be specifically requested. Content is also user-defined; it may be source, relocatable binary, memory-image format, and so forth.

The record formats of disc files are illustrated in Appendix C.

1-11. FILE ACCESS

Type 1 and type 2 files contain fixed length records which makes it possible to calculate the position of a desired record. On the other hand, type 3 files and above contain variable length records, so the system must access the disc at least once, and in some cases several times, in order to position to the desired record location. For this reason, access takes longer for file types greater than type 2.

File Extents

Files of type 3 and above are automatically extended whenever a write request points to a location beyond the range of the currently defined file. The extent is created by FMP with the same name and size as the main file, and access continues. FMP numbers each extent starting with 1. The extent number and location is kept in the file directory entry for the file. When a file with extents is referenced by its file name, any extents are provided automatically. At close, the extents may be truncated to provide more disc space, or they may be retained.

Files Opened for Update

Files may be opened in the update mode. This implies that the file is to be modified in some manner. Type 2 files should always be opened for update except when the file is written originally, or when adding records to the end of the file, and then only if the file is written sequentially. Reading or positioning a file is not affected by update or non-update mode.

In update mode, the entire block containing the record is read into the Data Control Block before the record is modified. After modification, the entire block is written to the disc. This is done to insure that the Data Control Block always contains the unmodified as well as the modified data, thereby guaranteeing restoration of the block to the disc.

1-12. SECURITY

FMP provides two levels of security: system security and file security.

System Security

During system setup, a master security code is assigned to the system. If the code is zero, no security is provided. If non-zero, the master code must be known in order to get directory listings that include the specific file security codes and also in order to re-initialize an FMP cartridge.

File Security

Each file has a security code. This code may be zero, positive, or negative. A zero code allows the file to be opened to any caller with no restrictions; in effect this code provides zero security. A positive code restricts writing on the file but not reading; that is, a user who does not know the code may open the file for read only, but may not write on the file. A negative code restricts all access to the file; this code must be specified in order to open a file protected by it. An attempt to open a file so protected without the correct security code results in an error message.

1-13. FMP PROGRAM CALLS

These calls to the FMP library provide programmatic control of input from, output to, and positioning of disc or non-disc files. They may also be used to create or purge disc files. The FMP calls allow you to:

- Create disc files
- Open an existing file to:
 - Add records
 - Delete records
 - Change records
 - Copy records or information
- Close an opened file
- Purge a disc file



The FMP recognizes calls from programs written in:

FORTRAN IV
 HP FORTRAN
 ALGOL
 HP Assembly Language
 Multi-User Real-Time BASIC

The FMP resolves conflicts between users who want to open the same file. If you request exclusive use of a file, FMP grants this request only if the file is not already being used. Once granted, exclusive use means that no other program may access the file until you close it. You may also request non-exclusive use of any file that you want to share with other programs. Up to seven programs may share the same non-exclusive file. If this file is a non-disc peripheral device, the file access protection provides a way to avoid contention for such devices.

The functions that will modify file directory entries are:

- Creating a file
- Opening a file
- Closing a file
- Purging a file
- Renaming a file

Since all these functions can be programmed, it is clear that there could be conflict. The FMP resolves this conflict by allowing only one program (D.RTR) to write on the directories. D.RTR is scheduled with wait whenever a directory change is required. Disc tracks on system and auxiliary discs that contain the file directories for FMP tracks on these discs are assigned to D.RTR. These directory tracks are therefore protected by FMP from all other programs.

1-14. FMGR OPERATOR COMMANDS

Program FMGR is run from the system console or from a terminal in a multi-terminal environment. It responds to a set of more than 40 commands. Any command may be entered directly from the terminal to perform a particular function, or one or more commands may be stored on a file as a procedure. Commands can also be included in jobs to be entered from peripheral devices in order to provide batch job control. Program FMGR allows an operator to perform the following basic functions:

- Control FMGR by sending messages to the console or list device, requesting error explanations, changing the log or list devices or error severity.
- Create and maintain files, both disc and non-disc, including maintenance of the file directory
- Keep track of the disc cartridge on which files are placed, including maintenance of the cartridge directory.
- Transfer data or programs between files or to and from system areas such as the load-and-go and logical source areas, creating new files as needed.
- Establish and transfer to procedure files, and manipulate the global parameters used in these files to receive and return data from other commands or procedures.
- Control execution of jobs in the batch stream, including assigning a job time limit, switching logical units, and aborting the job.

Since the files controlled by FMGR include data files, program files, non-disc devices, procedure files, and batch jobs, this program can provide full control over all input to and output from FMP. It uses many of the same FMP program calls for the actual input and output as does any user.

Multi-Terminal Monitor

When the Multi-Terminal Monitor is used, programs including FMGR may be requested from more than one terminal. Since RTE program access is not time-shared, a method has been devised to allow apparently simultaneous access to FMGR, not only from each active terminal but also for batch job control.

By making a copy of FMGR for each terminal, these copies can be run to provide the FMGR interactive capabilities to each terminal simultaneously. The program FMGR itself can then be reserved for batch job control. Methods for making copies of any program are described in this manual in Appendix E.

1-15. BATCH PROCESSING

Batch processing, the entry of one or more jobs for processing in a job stream, is controlled by FMGR commands. The jobs themselves may be stored on disc or on a peripheral input device. In either case, batch job operation is controlled through FMGR commands that delimit the job and that may be included with the job.

A batch job usually consists of FMGR commands to control system operation, a source language program to be compiled and executed, and data to be manipulated by the program. With batch processing, more than one such job can be placed in the input device so that the running of each job and the transition from job to job takes place with a minimum of human intervention.

Program FMGR performs batch processing by means of a set of commands that define the beginning and end of each job and effectively define a separate environment for the job. Any of the other FMGR commands can be included in the job itself in order to perform the full range of FMGR functions. Because the job may include programs and the commands to compile or assemble, load, and execute the programs, program development is possible in batch mode. A programmer can submit a job, return to other work and retrieve the output later.

1-16. SPOOLING

The Spool Monitor is used in conjunction with the File Management Package to provide batch job processing with "spooling". Spooling allows jobs to be processed according to assigned priorities rather than in the sequential order necessitated by standard batch processing. (Refer to Figure 1-4 for an example of spooling).

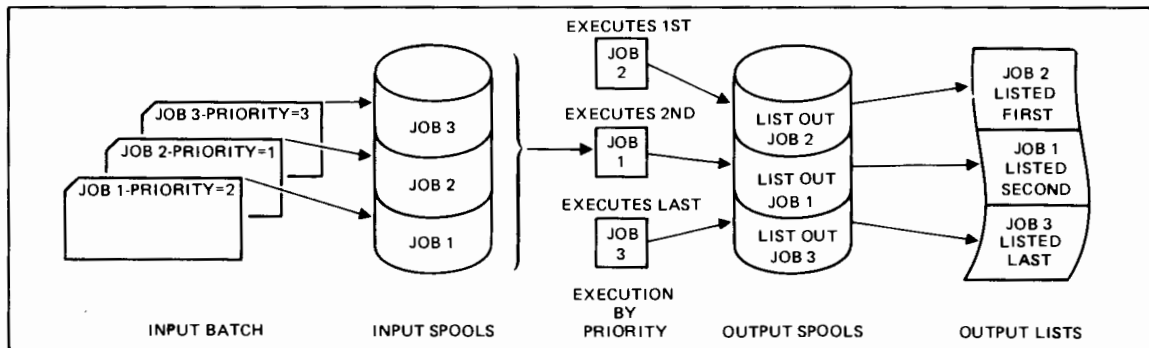


Figure 1-4. Spooled Batch Processing

To illustrate, consider that jobs processed directly from the card reader or paper tape reader must be taken in serial order as they are placed in the reader. Also, if the output goes directly to the line printer, paper tape punch, or other output device, the computer operating system must wait for the completion of one job before processing the next job. Priority of execution can be managed only by the operator manually inserting one job ahead of others, and lengthy printout from a low-priority job can delay execution of a higher priority job.

When spooling is used, jobs are read from the input device to spool files or from a user file on disc, and a directory of jobs to be executed is created. From disc, the FMGR selects each job for execution according to its priority. Similarly, the output spool file is selected from the outspool directory and is "outspooled" according to priority independently of input or processing priorities. Transition between jobs is faster as one job does not have to wait for completion of a previous job's output to the print or punch device. Also, since output files are on disc, they may be held until the operator takes action to print or punch them. This is especially convenient if, for example, the output requires a special form on the line printer.

Inspool or outspool files can be allocated automatically by the Spool Monitor or they can be specified by the user. Simplicity of use is achieved if the Spool Monitor allocates and releases all spool files. On the other hand, if a particular file is specified, that file can be retained as a permanent file.

For instance, if you define your own job command file as an inspool file, the job can be kept in that file and, at a later time, it can be re-run by command. In this case, no use is made of system spool files, and "inspooling" consists merely of creating an entry for the file in the job directory. This technique eliminates the necessity of putting a frequently run job on the non-disc input device each time it is executed.

Spooling is requested by submitting jobs to the inspool program (JOB). Output spooling to logical unit 6 is performed automatically unless specifically disabled by command in the job stream. Spooling to other devices may be specifically requested.

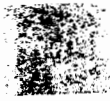
1-17. GASP OPERATOR COMMANDS

A set of operator commands is available through the spool program GASP that allow you to control various aspects of the Spool Monitor from the system console or other terminal. These commands perform such functions as:

- Display or change the status of jobs being inspoiled or outspooled.
- Hold jobs being inspoiled or outspooled and subsequently restart them.
- Remove entries from either the output or input directory.
- Restart the outspooling of a file.
- Place a system-wide hold on the spool system or de-activate it entirely.

1-18. SMP PROGRAM CALLS

The program SMP may be scheduled through RTE EXEC calls from a program in order to perform any of several spool control requests. These calls are used to spool input from or output to a program. Because spooling uses disc files, the calls provide the capability to read from or write to disc with non-disc I/O calls whether formatted or not. In addition, the calls provide spooling apart from batch job processing.



SECTION II

BATCH AND FILE CONTROL WITH FMGR OPERATOR COMMANDS

BATCH AND FILE CONTROL WITH FMGR OPERATOR COMMANDS

SECTION

II

2-1. INTRODUCTION

The FMGR commands can be entered interactively at a terminal or they can be entered in batch mode from a non-disc input device or from a procedure file on disc. The commands perform the following functions:

- Control FMGR operation
- Create and manipulate disc or non-disc files
- Develop and save program files
- Develop and save procedure (transfer) files
- Control batch jobs
- Initialize and manipulate disc cartridges



A permanent disc-resident program FMGR is used to perform the operations described in this section. You may schedule FMGR with the RTE RU command from an interactive terminal or schedule it from a program with the RTE Program Schedule EXEC call.

2-2. INTERACTIVE VS. BATCH OPERATION

In interactive mode, you enter the FMGR commands from an interactive device specified as the input parameter in the RU,FMGR command. If the input parameter is omitted or is zero, FMGR defaults its value to 1 and interacts with the system console. In a Multi-Terminal Environment, the default input device is the logical unit of your terminal. Interactive mode assumes that the FMGR input device allows two-way communication (teleprinters and display terminals).

As soon as it is scheduled interactively, FMGR issues a colon prompt (:) to signal that it is ready for you to enter a command. This prompt is issued after each command is successfully completed until the EX command terminates FMGR.

When the input parameter in the RU,FMGR command specifies a non-interactive device or is a FMGR procedure file on disc, batch mode is assumed. Batch mode is entered when FMGR, not a copy of FMGR, processes a :JO command. It is normally exited when FMGR processes an :EO command, signifying the end of the job. Batch mode is used when the input device is a card reader, paper tape reader, magnetic tape, or a FMGR file on disc. Note that an input device can be referenced as a type 0 file.

In batch mode, you must include the colon prompt (:) as the first character of the command; the system does not supply this prompt. (Refer to Figure 2-1.) When a group of commands are preceded by the JO command and terminated by EO, the commands form a batch job. Batch jobs can be entered as part of a stream of jobs; all functions are performed without operator intervention and the beginning and end of each job are logged on the list device. Batch jobs are generally entered from a device such as the card or paper tape reader, or they may be saved as FMGR procedure files to be entered from disc. Whether entered from a file or device, jobs may be spooled with the Spool Monitor (refer to Section IV).

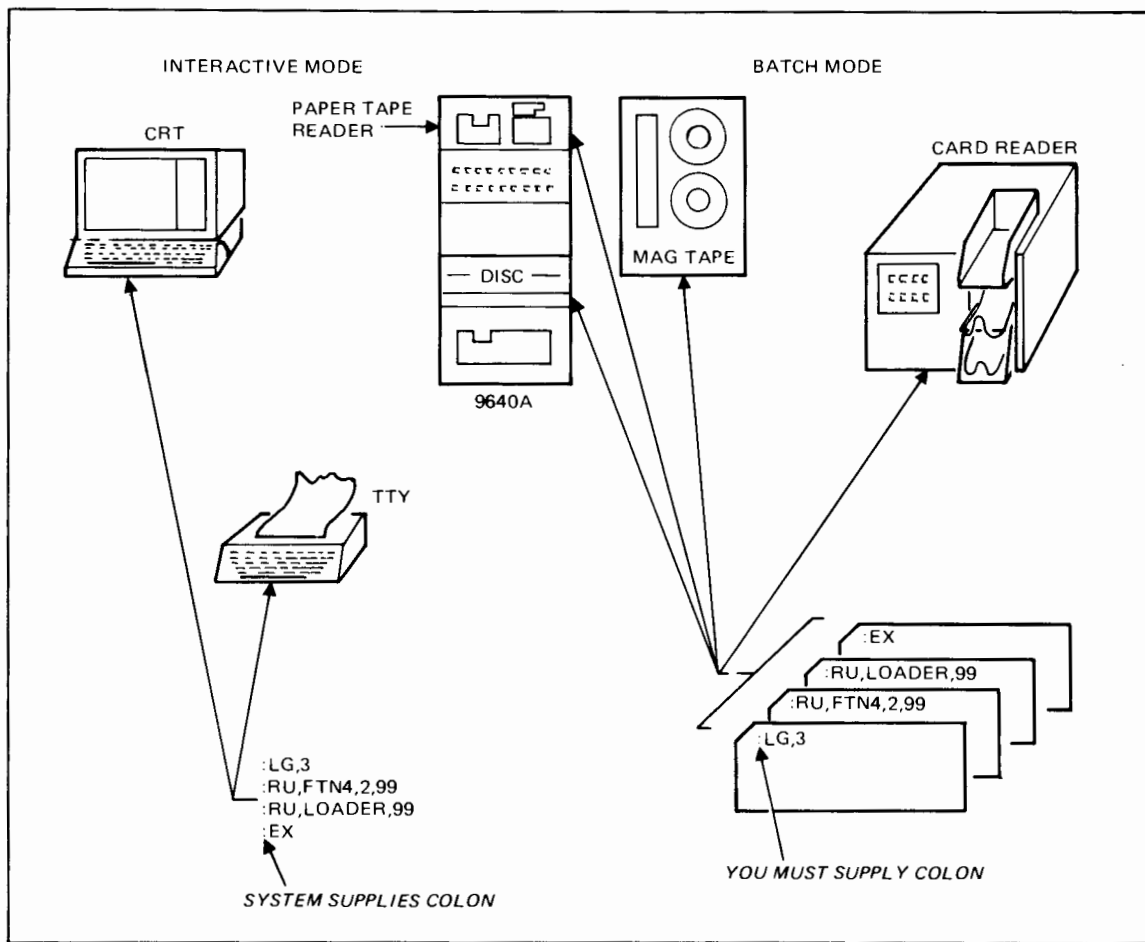


Figure 2-1. Interactive vs. Batch Operation

The program FMGR controls all batch operations. If an attempt is made to run FMGR interactively at the same time, the message **ILLEGAL STATUS** is returned to indicate that FMGR is busy. The same message is issued if FMGR is requested to run a batch job while it is being run interactively. To avoid this conflict, a copy of program FMGR can be made to be used at the interactive device. (Refer to Appendix E.)

Batch processing without spooling is explained more fully starting at "Batch Job Control".

Program JOB is used to inspool jobs in a batch environment with spooling. FMGR then processes the jobs. This capability is explained in Section IV.

2-3. MULTI-TERMINAL ENVIRONMENT

If your system operates in a multi-terminal environment under control of the Multi-Terminal Monitor, it is essential to have a copy of FMGR for each terminal that uses FMGR. Any

terminal can use any copy, but to avoid two terminals using the same copy, it is good practice to associate each copy mnemonically with a terminal number. For example, copy FMG07 could be associated with terminal LU 7. Copies of the program must begin with the letters FM.

RTE-IV Multi-Terminal Monitor Operation

In an RTE-IV Multi-Terminal Monitor (MTM) environment, the prompt for a terminal other than the system console depends on whether it has its own copy of FMGR. A terminal of logical unit number xx has its own copy of FMGR if a program exists named FMGxx.

If such a copy exists, it is automatically scheduled for execution each time the terminal interrupts the Operating System. The following prompt is issued:

```
xx>FMGxx
```

and the user is conversing with his copy of FMGR. Furthermore, the Operating System makes the interrupting terminal the log list device; a transfer file named "HI" is executed for the user; and the BReak and ABort commands have a special meaning. Refer to the MTM section of the RTE-IV Programmer's Reference Manual for further information.

If such a copy does not exist, the following prompt is issued:

```
xx>
```

and the user is conversing with the Operating System.

If copies do not exist for use with your system, you can make copies using the procedure outlined in Appendix E.

NOTE

All batch operations are performed by program FMGR, not a copy of FMGR.

2-4. INTERRUPTING FMGR

Most FMGR command processing can be interrupted with the RTE system command:

```
*BR,FMGR
```

If you are using a copy of FMGR, be sure to use the name of that copy.

FMGR Operator Commands

If FMGR is at a convenient breakpoint, command operation stops and the message:

FMGR 000

:

is printed or displayed on the log device, usually the system console or user terminal. Control is transferred to that device unless a job is being processed. If a job is being processed, the BR command terminates the job and returns control to FMGR.

In an RTE-IV Multi-Terminal Monitor (MTM) environment, the BReak and ABort commands have a special meaning. Under the following conditions, the last son of program FMGxx will be broken or aborted by the BReak or ABort commands:

1. the user is at an MTM terminal.
2. the terminal has its own copy of FMGR named FMGxx.
3. the BR or AB command is given with no program name specified.

Refer to the Multi-Terminal Monitor section of the RTE-IV Programmer's Reference Manual for more information.

Most FMGR commands recognize the BR command. For those that do not, it is recognized between commands. This insures that command processing is completed for commands where interruption could cause difficulties. For example, if FMGR is interrupted while a command is creating a file, the file is purged. Or if the command is to pack all cartridges, it will finish packing the current cartridge, but will not pack the next. Such interruptions do not occur with the BR command, but could occur if OF,FMGR is specified. For this reason, OF,FMGR is not recommended. Execution of a batch job or a program running under FMGR control can be interrupted with the RTE abort command:

*AB

The program is aborted and control returns to FMGR. If no program is executing, it acts as a break command and interrupts FMGR. Refer to the RTE Operating System manual for information on the AB options.

NOTE

AB will affect only programs executing as a result of FMGR,
not a copy of FMGR.

FMGR suspends itself when it expects further batch input through a non-interactive input device. In this case, a FMGR 006 error message is issued and you must place the input in the device and type *GO,FMGR.

If FMGR is terminated with :OF,FMGR or *OF,FMGR,8 (not recommended), it can be re-scheduled with *RU,FMGR as described in paragraph 2-10.

2-5. FMGR COMMANDS

All the FMGR commands are summarized in Table 2-1. This table presents the commands in the same functional groups in which they are described in this section.

Table 2-1. FMGR Operator Command Summary

FUNCTIONAL GROUP	COMMAND	FUNCTION	PARAGRAPH
FMGR Operation	??	Request error code explanation	2-11
	EX	Terminate FMGR; return to RTE control	2-12
	LL	Change list device	2-13
	LO	Change log device	2-14
	SV	Change severity code	2-15
	†TE	Send message to system console	2-16
	†AN	Send message to list device	2-17
File Creation and Manipulation	CR	Create disc file	2-19
	CR	Create non-disc file	2-20
	PU	Purge file	2-21
	ST	Store data in device or file; create file	2-22
	DU	Dump data to device or created file	2-23
	LI	List contents of file	2-24
	CN	Control non-disc file	2-25
	RN	Rename disc file	2-26
Program File Creation and Manipulation	MS	Move source program to LS area	2-28
	LS	Set or clear pointer to LS area	2-29
	LG	Allocate tracks to LG area	2-30
	MR	Move relocatable program to LG area	2-31
	SA	Save LS or LG area as file; create file	2-32
	SP	Save memory-image file for execution	2-33
	RP	Restore memory-image file for execution	2-34
	†RU [IH]	Execute program, program file, or procedure [inhibit command string passage]	2-35
	OF	Remove program and ID segment from memory	2-36
RT	Release tracks assigned to program	2-37	
Procedure File Manipulation	TR	Transfer control to procedure file or device	2-41
	†PA	Pause and send message to log device	2-42
	†DP	Display parameter values	2-43
	SE	Define global parameters	2-44
	CA	Calculate global parameter values	2-45
	IF	Branch on parameter values	2-46
Batch Job Control	JO	Initiate job	2-48
	EO	Indicate end-of-job	2-49
	TL	Set run-time limit within job	2-50
	LU	Switch logical units within job	2-51
	AB	Terminate job	2-52
FMP Cartridge Manipulation	MC	Mount Cartridge	2-55
	IN	Initialize cartridge	2-56
	DC	Dismount cartridge	2-57
	CL	List cartridge directory	2-58
	DL	List file directories	2-59
	PK	Pack cartridges	2-60
	CO	Copy all files from cartridge to cartridge	2-61

†Privileged command (see paragraph 2-7)

Table 2-1. FMGR Operator Command Summary (Continued)

FUNCTIONAL GROUP	COMMAND	FUNCTION	PARAGRAPH
Miscellaneous Commands	†SY †**	Execute an RTE system command from FMGR Denotes a "comment" line	2-63 2-64

†Privileged command (see paragraph 2-7)

2-6. COMMAND STRUCTURE

Each command is specified by a mnemonic code consisting of at least two letters to indicate the operation to be performed. Depending on the command, parameters may be entered to further specify the command operation. Commands allow more than two characters in the command code, but only the first two characters are significant. For example, STORE can be specified, but ST is always sufficient.

The following syntax conventions are used in this manual to specify command format.

<u>UPPER-CASE</u> BLOCK LETTERS	Literals that must be specified exactly as shown; <u>if underlined, the letters may be omitted.</u>
<i>lower-case italics</i>	Type of information to be supplied by the user; most parameters are in this form.
[,parameter]	Optional parameters are enclosed in brackets; FMGR supplies a default value if omitted.
<i>parameter 1</i> <i>parameter 2</i> <i>parameter 3</i>	One and only one of the stacked parameters may be specified.
[<i>parameter 1</i>] [<i>parameter 2</i>] [<i>parameter 3</i>]	All bracketed parameters are optional; if all are omitted, FMGR supplies default value, or only one may be specified.
[,param1 [,param2]]	Series of optional parameters; the last parameter may be omitted with no indication; embedded parameters must be indicated by a comma when omitted.
...	Ellipsis indicates that the previous parameter or series of bracketed parameters can be repeated.

To illustrate:

If the format is:

You may enter:

EOJ	EO OR EOJ
DL [,cartridge[,security]]	DL <i>default values supplied for parameters</i> DL,SC <i>position of parameter held by comma</i> DL,2 <i>last parameter omitted</i>
LIST,namr [,SOURCE ,BINARY ,DIRECTORY]	LIST,2,DIRECTORY <i>use full names</i> LI,2,D <i>use abbreviations for identical effect</i> LIST,FILA <i>default parameter supplied</i>

2-7. PARAMETER SYNTAX RULES

A parsing routine checks every parameter specified in a FMGR command according to the following syntax rules:

- The first parameter is separated from the command code by a comma (,) or a colon (:). Subsequent parameters are separated from each other by commas.
- Subparameters are legal in the first two parameters. Also, they are legal anywhere within a privileged command (see below). Subparameters are separated from each other by a colon (:). The first two subparameters may be ASCII or numeric, the rest must be numeric.
- Blanks on either side of a delimiter or the command code are deleted from the command entry; they are not transmitted or echoed back.
- Parameters are first assumed to be numeric, but if the parameter fails to convert, it is treated as ASCII unless it is a number immediately followed by B,G,P, or S or is preceded by a plus (+) or minus (-) sign.
- Numeric parameters observe the following rules:
 - A leading plus sign (+) is ignored; a number is assumed to be positive unless preceded by a minus sign (-).
 - A number followed by the letter B is octal.
 - A number followed by G, P, or S is a global reference.
- ASCII parameters are parsed to a maximum of six characters; only the first six characters are interpreted. If fewer than six, the parameter is padded with trailing blanks (octal code 40, the ASCII space). Blanks within a number are ignored. In a message command (AN,TE, or PA), a parameter may contain more than six ASCII characters since the message parameter is not interpreted.

FMGR Operator Commands

- The total number of characters in any parameter must be less than: $128 - (8 \text{ times the parameter number})$ i.e, parameter 10 must be less than 48 characters: $128 - (8 * 10) = 48$.
- The maximum number of parameters in one command is 14.
- Comments may be entered following the last parameter as long as they do not replace an omitted optional parameter. They are subject to the length and number and subparameter restriction on all parameters, but like messages are not interpreted.
- For privileged commands (see Table 2-1) minimum syntax checking occurs before the command is processed. The only syntax requirements for privileged commands are:
 1. If issued from a non-interactive device, the first character must be a colon (:).
 2. Global values specified in the command string must be within the legal range (see paragraph 2-40).
 3. The "constructed" command line length (after globals are replaced and blanks on either side of the delimiters are removed) must be less than 80 characters.

2-8. NAMR PARAMETER

A special parameter *namr* is used to identify a file or device in a FMGR command. It uses subparameters and may appear only as the first or second parameter.

Format

$namr = \text{file name} [:security[:cartridge[:file type[:file size[:record size]]]]]$
$\text{logical unit number} \qquad \qquad \qquad \text{file creation only}$

Unless specifically noted, each subparameter has a default value of zero. This value is selected so that, as closely as can be predicted, it provides the most general case. This means that in many cases, all subparameters can be omitted and the file be completely specified by name alone.

NAMR Subparameters

- file name* 6-character ASCII file name; restricted as follows:
- only printable characters, space through `_` (or `←`)
 - plus (+), minus (-), colon (:), or comma (,) not allowed
 - first character must not be blank (space) or a number
 - embedded blanks not allowed
 - must be unique to FMP cartridge
- logical unit* positive integer specifying logical unit number of non-disc device
- security* positive or negative integer or 2 ASCII characters representing a positive integer; range is from -32767 through 32767; security may be:

- zero* file is unprotected (default)
- +integer* write protected; may be read with any security or none; may be written only with correct code or negative (2's complement) of correct code.
- integer* file is fully protected; may be referenced only with correct negative code.
- cartridge* positive or negative integer or 2 ASCII characters representing a positive integer; range is from -32767 through 32767; used to identify FMP disc cartridge, it may be:
- zero* first available cartridge that satisfies the request is used; (default)
- +integer* cartridge reference number (CR) by which the cartridge is identified.
- integer* logical unit number (LU) associated with the cartridge.
- file type* positive integer in range 0 through 32767; default depends on command.
- 0 non-disc file
 - 1 fixed length 128-word record
 - 2 fixed length records; user defines length
 - 3 variable length record, sequential access, automatic extents
 - 4 ASCII code and source programs (otherwise like type 3)
 - 5 relocatable binary code (otherwise like type 3)
 - 6 memory-image format (otherwise like type 3)
 - 7 absolute binary (otherwise like type 3)
- 8 - 32767 user defined
- file size* decimal number of blocks in range 1 through 16383; a block is 128 words (two 64-word sectors); indicates space allocated to file:
- +integer* allocate specified number of blocks to file; minimum is 1.
- integer* allocate remainder of available space (up to the maximum allowed file size) to file; *integer* may be any value (valid only for files of type 3 or greater).
- record size* decimal number of words in range 1 through 32767; applies only to type 2 files; type 1 files use 128-word records, other types use variable length records.



NAMR Examples

- 10 logical unit 10
- 20B logical unit 16 (octal 20)

FMGR Operator Commands

\$XYZ:AA	file named \$XYZ is write protected by ASCII code AA (040501 octal or 16705 decimal)
ABS:-10:-3:2:40:64	file named ABS fully protected by security code -10, is located on LU 3, is a type 2 file 40 blocks long, each record has 64 words.
A23456:::3	file name A23456 is type 3; security and cartridge default to zero. Note that a colon is specified for each omitted parameter plus one colon to separate the subparameters from the file name. Thus, one more colon than default parameters must be supplied.

2-9. FMGR OPERATION

To use FMGR interactively, you simply run the program from a terminal. When it responds with the colon prompt, you may enter any command. To run in batch mode, you also run the program from a terminal but specify that command input is to be from an input device such as the paper tape reader or from a disc file. No prompt is issued. The program FMGR expects a colon to precede each command entered from a non-interactive device or disc. You may also schedule program FMGR from a program.

When you run FMGR, it assumes default values for the devices used for command input, to log errors and to list output. These values all assume interactive mode. You may specify other devices when you run FMGR and you may change the devices during operation of FMGR with FMGR commands.

When you enter each command at a terminal, it is echoed back to you at the terminal. It appears as if you were typing the command on the terminal display, but actually the command is sent to FMGR which then displays (echos) the command. Command echo can be suppressed, and it is good practice to suppress the echo when command input is from a non-interactive device.

Except for errors requiring correction, error messages can also be suppressed. If a batch job is being run, you can prevent FMGR from terminating the job in case of an error that needs correction or you can specify that the job must terminate in such a case. Normally, when an error requires operator action, it is logged on the log device and control transfers to that device.

During FMGR operation, messages can be issued to the terminal, to the log device if it is different from the terminal, or to the list output device. Each type of message uses a different command. The message to the log device also causes a pause in job processing so that you may interact with FMGR.

FMGR operation is terminated with the EX command. If spooling is operative, FMGR checks if there is another job to process before terminating. During interactive operation this may slow the actual termination, but will not affect FMGR copies in a multi-terminal environment.

2-10. RUNNING PROGRAM FMGR

To request FMGR from the RTE system console or your terminal, you specify the system command RUN. You may also schedule FMGR from a program with the RTE program schedule EXEC call.

Format

*RUN,FMGR[, <i>input</i> [, <i>log</i> [, <i>list</i> [, <i>severity</i>]]]]	<i>commands entered from device</i>
*RUN,FMGR, <i>fi,le,nm</i> [, <i>severity</i> [, <i>list</i>]]	<i>commands entered from file</i>
Parameters	
<i>input</i>	Logical unit number of input device for FMGR commands; if omitted, LU 1 is assumed; in a multi-terminal environment, default <i>input</i> is logical unit number of device where FMGR was scheduled.
<i>log</i>	Logical unit number of the log device used to log and to correct any diagnosed errors; must be an interactive device; if omitted, <i>input</i> device is assumed unless it is non-interactive, in which case, LU 1 is assumed.
<i>fi,le,nm</i>	File name of file containing command input to FMGR; specified as 3 words each consisting of 2 ASCII characters; refer to file name discussion below for restrictions.
<i>list</i>	Logical unit number of device used to list results of FMGR commands; if omitted, LU 6 is assumed.
<i>severity</i>	Severity code defines action in case of error messages; <ul style="list-style-type: none"> 0 Display error codes and echo commands on log device (default). 1 Display error codes on log device, inhibit command echo. 2 Error code displayed only if error requires transfer of control to log device for correction, in this case, active job terminates; no command echo. 3 Same as 2 except active job not terminated when an error causes transfer to log device. You can transfer back to the job. No command echo. 4 If a FMGR command error is encountered, job continues automatically; no command echo, no transfer to log device, and no job abort occurs.

NOTE

Once FMGR is running, *log* can be changed with the LO command, *list* with the LL command, and *severity* with the SV command.

Input Device

The device through which commands are sent to the FMGR program can be any device to which a logical unit has been assigned and that can be used for input. Normally, when input is from disc, a file name is used to specify the input device.

File Name

- If the name begins with NO, these characters are assumed by RTE to be the scheduling parameter NOW and the second parameter is treated as the first two characters in the file name. Either specify NO twice, or use the octal equivalent for the characters. For example, to specify input from the file NOBLE:

RU,FMGR,NO,NO,BL,E or
RU,FMGR,47117B,BL,E

- If a file name has two numbers in the last two character pairs they are treated as numeric unless they are followed by some ASCII character to indicate they are ASCII, or the octal equivalent is used. To specify input from AA1044:

RU,FMGR,AA,10X,44X or
RU,FMGR,AA,30460B,32064B

Since the parse routine deletes blanks, the name AFIL1 must be entered as:
RU,FMGR,AF,IL,30440B

Examples

1. *RU,FMGR ← *FMGR is scheduled with default values for all parameters: input from LU 1, errors logged on LU 1, list output to LU 6*
2. Ø7>RU,FMGØ7 ← *A copy of FMGR (FMGØ7) is schedule with input from LU 7 (terminal from which FMGR is scheduled), errors logged on LU 7, list output to LU 6*
3. *RU,FMGR,5,,,1 ← *Input from paper tape (LU 5), log errors on LU 1, list to LU 6, severity inhibits command echo on LU 1.*
4. *RU,FMGR,FILEA1,1,1 ← *Input from FILEA1, severity inhibits command echo on console, default list is LU 6*

Program Request For FMGR

To request FMGR from a program (optionally, a command string may be passed to FMGR), use the following call:

Format

```
CALL EXEC(ICODE,FMGR,INP,LOG,LIST,ISV,IDUM,IBUFR,IBUFL)
                                                    entered from device
```

```
CALL EXEC(ICODE,FMGR,2HFI,2HLE,2HNM,ISV,LIST,IBUFR,IBUFL)
                                                    entered from file
```

Parameters

ICODE = 23 to schedule FMGR with wait
 = 24 to schedule FMGR without wait

FMGR a 3-word array containing ASCII file name FMGR.

INP,LOG,LIST,ISV } correspond to the parameters in the RU,FMGR command.
 FI,LE,NM,ISV,LIST }

IDUM a placeholder for parameter 5 in command entered from device.

IBUFR,IBUFL buffer address and length containing command string to be passed to FMGR.

Command String Passing

The command string to be passed to FMGR via IBUFR must begin with a colon (:). The buffer length specified in IBUFL is a positive value for words, or a negative value for characters. The command string is ignored if:

- a. The input device specified is an interactive device.
- b. The command string does not start with a colon.

FMGR assumes that the string passed is a command (it may be any command) and effectively inserts it in front of the first command on the specified input file.

2-11. ?? - REQUEST ERROR EXPLANATION

Once FMGR is running, it assumes control and if a command cannot be understood (input error) or has caused a recognized problem, an error message is printed in the form:

FMGR *nnn*

where *nnn* is a three-digit number. (Refer to Appendix B for a list of all FMGR error codes, their meaning, and corrective action.) During interactive operation, a brief description of the error code can be requested with the ?? command.

Format

```
:??[,error #]
```

Parameter

error # Error code whose explanation is requested; if omitted, the explanation of the current error is issued and the value of global P6 is set to zero; if two error codes were sent, the explanation usually refers to the first number only. If *error #* is 99, a list of all FMGR error codes and their explanations is printed at the list device.

FMGR Operator Commands

In some cases, when an error code is issued, it is followed automatically by additional information. This may consist of the line in which the error occurred up to the point where the error was detected. Or it may be a second FMGR error code. Any error code explanation can be requested by entering the code number as the *error #* parameter. Be sure to include the comma separator or the current error will be explained.

2-12. EX - TERMINATE FMGR

When you have finished using the FMGR, you can terminate the program and return to RTE control with the command EX.

Format

```
:EXIT
```

In response to EX, the FMGR sends the message:

```
$END,FMGR
```

to the log device unless a non-zero severity code inhibits the message.

2-13. LL - CHANGE LIST DEVICE

You may change the list device currently assigned to FMGR with command LL.

Format

```
:LL,namr
```

Parameters

namr New list device; may be either a file or a logical unit number. (Refer to paragraph 2-8 for *namr* description).

namr may refer to any existing file or logical unit. It should be a device allowing output; an attempt to list on an input device terminates FMGR and issues the system error code I007.

Certain FMGR commands (AN, LI, CL, DL, EO, JO) direct their output to the list device. By default this is the line printer in order to provide printed copies. If you want this output saved on a file or directed to your terminal or some other device, you may request it with the LL command. If LL is specified within a batch job, the default device (LU6) is re-established at the end of the job.

Examples

1. ***RU, FMGR**
:LL, 4 ← *Change the list device from default logical unit 6 (line printer) to logical unit 4 (paper tape punch)*
 .
 .
 .
:LL, 6 ← *Subsequently change it back to logical unit 6.*

2. ***RU, FMGR**
:LL, LISTF::13 ← *Change list device to list output on existing file LISTF; Specify CR to insure that list is sent to correct file.*

2-14. LO - CHANGE LOG DEVICE

You may change the log device currently assigned to FMGR with the command :LO.

Format

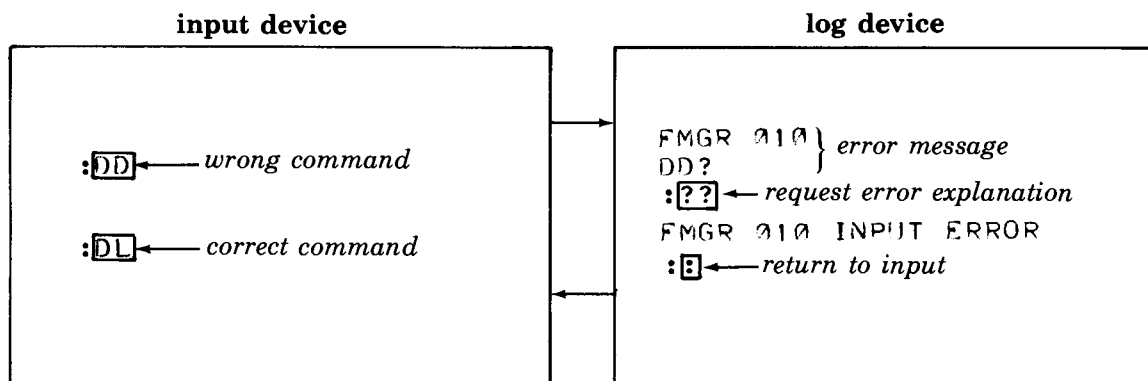
```
:LOG,lu
```

Parameters

lu Specifies the logical unit number of the new log device; note that a file name cannot be used as a log device.

lu must be a two-way device such as a teleprinter or CRT terminal since it is used both to log messages and to correct errors.

All error messages are printed or displayed on the log device. When the log device is not the input device and an error occurs that requires operator correction, control is transferred from the input to the log device and corrective action must be taken at the log device. To transfer control back to the input device, simply type a colon after the colon prompt. The second colon is interpreted as a transfer command.



If the LO command is specified within a batch job, the default device (LU1) is re-established at the end of the job.

Example

```
*RU,FMGR          log device is system console (LU 1) by default
.
.
.
:LO,7             change it to logical unit 7 (a terminal)
.
.
.
:LO,1            subsequently change it back to logical unit 1
```

2-15. SV - CHANGE SEVERITY CODE

You may change the severity code currently being used by FMGR with the command SV.

Format

`:SV,severity[,global#][,IH]`

Parameters

<i>severity</i>	New severity code; it may be:
0	Display error codes and echo commands on log device (default).
1	Display error codes on log device, inhibit command echo.
2	Error code displayed only if error requires transfer of control to log device for correction, in this case, any active batch job terminates; no command echo.
3	Same as 2 except active job not terminated when an error causes transfer to log device. You can transfer back to the job. No command echo.
4	If a FMGR command error is encountered, job continues automatically; no command echo, no transfer to log device, and no job abort occurs.
<i>global#</i>	Optional G global number in the range 1 - 9, in which the severity code is placed.
<i>IH</i>	Optional parameter to inhibit echo of command entry.

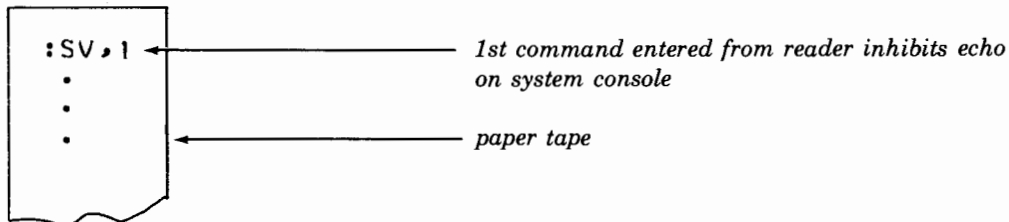
The normal default mode is to echo each command as it is entered on the input device and to log all errors on the same device. During interactive operation, the severity code should be this default value, zero. When there is no advantage to echoing commands at the console, for instance when commands are entered in a batch job, from a peripheral device, or through a file, you could set the severity code to 1. If, in addition, you want to suppress messages unless they require action, you can set the code to 2. This code terminates any currently executing job when an error occurs so that the job will not continue with errors. A severity code of 3 allows

jobs to continue in spite of errors. This code would be useful when more than one batch job must be processed so that the job containing errors does not hinder subsequent job processing. If SV is specified within a batch job, the severity is reset to zero when the job is terminated with the EO command.

Whenever command echo is suppressed, any command causing an error is displayed preceding the error code unless the error display is also inhibited.

Examples

1. *RIJ,FMGR,5 ← *command input from paper tape reader*



2. *RIJ,FMGR
 :JO
 :SV,3 ← *set severity to 3 to inhibit command echo and error messages; job will not terminate in case of error*
 .
 .
 :EO ← *end-of-job command resets severity to 0*

2-16. TE - SEND MESSAGE TO CONSOLE

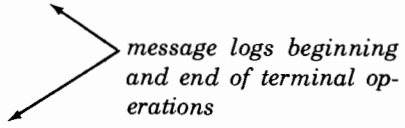
You can send a message to the system console with the command TE.

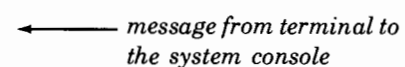
Format

<i>Privileged Command</i>	
:TELL,message	
Parameters	
message	Message to be sent to system console; must conform to parameter syntax rules for privileged commands (refer to paragraph 2-7).

Whatever you type as a message is displayed at the system console following the command code TE. The message may consist of any printable upper-case ASCII characters. It is limited by the length and number restrictions placed on any FMGR command parameters (refer to Parameter Syntax Rules, paragraph 2-7). Note that any commas divide the message into separate parameters. A one-parameter message could be as long as 60 characters.

Examples

1. *RIJ,FMGR
 :TE,START FMGR OPERATION, APRIL 22, 9AM.
 .
 .
 .
 :TE,****FINISHED**** NOON, APRIL 22
 

2. Ø7>,FMGR
 :TE, TURN ON AND READY LINE PRINTER
 

2-17. AN - SEND MESSAGE TO LIST DEVICE

You can send a message to the list device with the AN command.

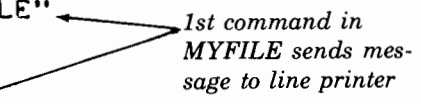
Format

<i>Privileged Command</i>		
:ANNOTATE, <i>message</i>		
Parameters		
<table style="width: 100%; border: none;"> <tr> <td style="padding-right: 20px;"><i>message</i></td> <td>Message to be sent to the list device; must conform to parameter syntax rules for privileged command (refer to paragraph 2-7).</td> </tr> </table>	<i>message</i>	Message to be sent to the list device; must conform to parameter syntax rules for privileged command (refer to paragraph 2-7).
<i>message</i>	Message to be sent to the list device; must conform to parameter syntax rules for privileged command (refer to paragraph 2-7).	

AN differs from TE in that it is sent to the list device, not the log device. Because it is printed on the list device, it is useful within batch jobs to annotate the job.

Example

```
*RU,FMGR,MY,FILE
:AN, ****FMGR OPERATION FROM FILE "MYFILE"
****FMGR OPERATION FROM FILE "MYFILE"
```



NOTE

One other command (PA) may be used to send messages. PA suspends current operation and transfers control to a specified device. Optionally, it sends a message. This command is particularly useful to request operator intervention during non-interactive operation. It is fully defined in paragraph 2-42.

2-18. FILE CREATION AND MANIPULATION

The file creation and manipulation functions include the creation of all types of files, the transfer of data between files or between files and devices, renaming files, listing file contents,

and purging files after use. A special command is available to perform control functions on non-disc files.

File creation for disc files means that the file is given a name, a file directory entry, and is assigned an area on disc. For non-disc files, creation means associating a peripheral device with a file name so that it can be treated as a file in subsequent operations. The FMP calls described in Section III that refer to non-disc files expect the files to be created by FMGR commands.

Files may be created in a variety of ways. Probably the most commonly used command, STORE, creates a file and transfers data to the file in the process. The CREATE command simply creates a disc or non-disc file with no data. Special files containing programs may also be created with FMGR commands. These commands that create files for source programs (type 4), for relocatable programs (type 5), and for memory-image programs (type 6), are described with program file manipulation beginning in paragraph 2-27.

When files or extents are created, a check is made to see if a "hole" left by a purged file is the same size as the requested file or extent size.

If such a "hole" is found, the new file or extent will be put there and the directory entry of the old file will be used for the new file or extent.

If no "hole" of the same size exists, the directory entry of the file (or extent) will be placed at the end of the directory. Disc space will be allocated for the new file at the end of all currently-allocated files on that disc. (Provided, of course, that there is enough room on the disc to accommodate the new file or extent.)

The transfer of data is accomplished either with the STORE or DUMP commands. Either of these commands allow you to append data to a file or overwrite data on a file. They may also be used to transfer data between peripheral devices or between a device and a disc file. DUMP differs from STORE primarily in that it does not create a file but assumes an existing file.

Device control is provided that allows you to perform all the functions of the RTE I/O Control EXEC call from an interactive terminal. These functions include writing an end-of-file or rewinding magnetic tape.

2-19. CR - CREATE A DISC FILE

A disc file may be created with the CR command. No data is transferred to the file.

Format

```
:CREATE,namr
```

Parameters

<i>namr</i>	File descriptor; must not be a logical unit number; omitted subparameters default to zero; file type and file size must be specified as greater than zero; record size need be specified only for type 2 files; refer to <i>namr</i> description, paragraph 2-8.
-------------	--

When a disc file is created, an entry is made in the file directory on the cartridge to which the file is allocated. If the subparameter *cartridge* is specified, the file is allocated to that cartridge; otherwise, it is sent to the first cartridge found with enough room starting at the head of the cartridge directory on the system disc. If a file with the given name already exists on the first cartridge with enough space, an error -002 is issued.

FMGR Operator Commands

If a file is type 3 or greater, an end-of-file mark is written at the beginning of the file. As data is entered serially in the file, the mark is moved to the end of the data.

The format of a file directory entry for a created file is illustrated in Appendix C. The information in the entire file directory (all files) can be listed with the DL command, information in an individual file with the LI command.

Examples

1. **:CR,MYFILE:-25:100:4:10**
MYFILE is type 4; it has a security code of -25 (read and write are restricted to users knowing the code); it is allocated to a cartridge with CR = 100; it uses 10 blocks (20 sectors) of disc space.
2. **:CR,URFILE:::2,20,72**
URFILE is type 2; uses 20 blocks; each record is 72 words. Cartridge is not specified, but a list will show the cartridge to which URFILE is allocated (see LI command).
3. **:CR,MYFILE:EJ:100:3,-1**
MYFILE is type 3; security code is EJ (only write restricted); on cartridge 100; the remaining unused portion of the cartridge up to the maximum allowed file size blocks is allocated to the file. The exact size can be determined with a list (LI).

2-20. CR - CREATE A NON-DISC (TYPE 0) FILE

A different form of the CR command is used to create non-disc files. It creates a file directory entry on the system cartridge that specifies device control information.

Format

:CREATE , <i>namr</i> , <i>lu</i>	<u>,READ</u> <u>,WRITE</u> <u>,BOTH</u>	[<u>,BSPACE</u> <u>,FSPACE</u> <u>,BOTH</u>]	[<u>,EOF</u> <u>,LEADER</u> <u>,PAGE</u> <i>control word</i>]	[<u>,BINARY</u> <u>,ASCII</u> <i>control word</i>]]]]
Parameters				
<i>namr</i>	Only the file name and, optionally, the security code are specified (refer to paragraph 2-8 for <i>namr</i> details); file type is default value 0 and other subparameters do not apply.			
<i>lu</i>	Logical unit of the non-disc device; a positive integer.			
READ, WRITE, or BOTH specify the legal input/output mode of the device; it must be specified, there is no default.				
RE	- device accepts input only; forward spacing is assumed			
WR	- device is output only; no forward spacing is supported			
BO	- device is used for input or output; backspacing and forward spacing are legal			

BSPACE, FSPACE, or BOTH specify the type of spacing the device supports; if omitted, FSPACE is assumed for READ devices and no spacing for other devices.

- BS - backspacing is supported
- FS - forward spacing is supported
- BO - both forward and backspacing are supported

EOF, LEADER, PAGE, or *control word* specify the particular type of end-of-file to be written on the device; if omitted, default depends on driver type (see Appendix C).

- EO - end of file mark for magnetic tape (default if device has driver type greater than 16 octal, magnetic tape or mass storage device).
- LE - leader on paper tape (default if driver type 02, paper tape punch).
- PA - page eject for line printer or two line feeds on teleprinter (default if not a punch or if driver type less than 17 octal).

control word - control subfunction (equivalent to function code in FCONT, see paragraph 3-21); supplied if further end-of-file definition needed; specify as octal integer of which only least 5 bits are used.

BINARY, ASCII, or *control word* specify the type of data on the device; ASCII is the default.

- BI - binary data
- AS - ASCII data

control word - subfunction (equivalent to bits 6-10 of IOPTN parameter in OPEN call, paragraph 3-10); supplied if further data definition needed; specify as decimal or octal integer of which only the lowest five bits are used.

NOTE

In general, a type 0 file can be specified with only the required parameters. FMGR needs a name, the logical unit, and whether the device is read only, write only, or both. The other parameters usually follow from this information.

Programs can use type 0 files as a means of controlling access to a device. Thus, type 0 files provide a measure of device independence in that the standard file calls (refer to Section III) can be used to control a peripheral device.

When a type 0 file is created, an entry is made in the file directory on the system disc. The entry for a type 0 file precedes all other file entries in this directory. To make the entry, the cartridge must be locked and other directory entries moved to make room for the new entry. The cartridge is also locked when a type 0 file is purged.

The type 0 file entry differs from the disc file entry in that control information replaces the track, sector, and record length information. The directory entry for type 0 files is illustrated in Appendix C.

Examples

1. `:CR,PUNCH:PP,4,WR,,LE,BI` ← create file *PUNCH* on LU 4 for write only using binary format and punching leader as end-of-file; security code is *PP*.
2. `:CR,LP,6,WR,,PA` ← create *LP* as output file on LU 6; defaults are no spacing and ASCII data.
3. `:CR,MT:32107,8,B0,B0` ← create *MT* as input/output file on LU 8; both forward and back spacing supported; security code is 32107.
4. `:CR,MAG:JT,8,RE` ← create a read-only magnetic tape file.
5. `:CR,READR,5,RE` ← create *READR* as input only file on LU 5.

2-21. PU - PURGE A FILE

A file and its extents can be removed from the system with the PU command. Type 0 files can be purged only with this command; other file types can also be purged with the PURGE call (see Section III).

Format

<code>:PURGE,namr</code>	
Parameters	
<i>namr</i>	File descriptor; enter file name and, optionally, security code and cartridge reference; (refer to <i>namr</i> syntax, paragraph 2-8).

If a file is protected by a security code it cannot be purged unless the correct code is entered. If a label is specified, that cartridge is searched for the file to be purged; otherwise, cartridges are searched and the first file found with the correct file name is purged.

When a file is purged, the first word in its file directory entry is set to -1. Tracks assigned to the file are returned to the system only if the file was the last on the cartridge. If, however, the file is a type 6 program, its tracks are not released. Disc space allocated to purged type 6 programs and all purged files except the last can be returned to the system only by packing the cartridge with the PK command (see paragraph 2-60), or by creating (in any way) a file or a file extent that is the same size as a purged file (see paragraph 2-18).

A purged file cannot be accessed and its name will not appear on the file directory list requested with the DL command. A new file with the same name can now be created on the cartridge.

Examples

1. `:PU,AA` ← *purge the first file found named AA*
2. `:PU,CC:55:100` ← *purge file CC protected by security code 55 on cartridge 100*
3. Assume files A, B, and C are the last files in the directory and are not type 6:
`:PU,A`
`:PU,B`
`:PU,C`

When C, the last file is purged, FMGR releases its disc space and then checks the next to last file B. Since B has also been purged, its disc space is released and A is checked. It too was purged and FMGR releases its disc space. This procedure continues until FMGR finds a file that has not been purged, or finds a type 6 file, purged or not.

2-22. ST - TRANSFER DATA AND CREATE FILE

The ST command transfers data from a file or logical unit to a disc file or logical unit; the receiving file is created by the command unless it is a logical unit. To transfer data to an *existing* file use the DU command, paragraph 2-23.

Format

```
:STORE,namr1,namr2 [ ,record format,eof control [ file # [ , #files ] ] ]
                   [ ,eof control
                   [ ,record format
```

NOTE

Only one place-holding comma is required when both *record format* and *eof control* are omitted

Parameters

- namr1* File name of existing file or a logical unit number; data is transferred from *namr1*. (See *namr* description, paragraph 2-8).
- namr2* File name or logical unit to which data is transferred from *namr1*; if a file name, the file is created using the last three *namr* subparameters if supplied.
- record format* Format of data in *namr1*; default is derived from file type of *namr1* or is ASCII; refer to record format description below.

<i>eof control</i>	SA to transfer embedded subfile marks in <i>namr1</i> to <i>namr2</i> and write end-of-file mark at end of data on <i>namr2</i>
	IH to inhibit end-of-file on <i>namr2</i> ; subfile marks are not transferred. If omitted, end-of-file marks are saved, subfile marks purged.
<i>file #</i>	Positive integer indicating file (or subfile) relative to beginning of <i>namr1</i> at which to start transfer; default is 1.
<i>#files</i>	Positive integer indicating number of non-disc files or disc subfiles to be transferred; default is 1 unless <i>namr1</i> is a disc file and <i>file #</i> is omitted, in which case default is 9999.

NOTE

Files are transferred record by record; records longer than 128 words are truncated.

namr1 can be a created file (disc or non-disc) or a logical unit number defined for the system;

namr2 can be a logical unit number or a disc file name; it may *not* name a type 0 file. The file type subparameter for *namr2* defaults to the file type of *namr1* if *namr1* is a disc file.

If *namr1* is not a disc file, the file type of *namr2* is based on the record format of *namr1*:

type 3 if record format is MT, MS, or AS (ASCII or System I/O) or is omitted

type 5 if record format is MSBR or BR (binary relocatable)

type 7 if record format is MSBA or BA (binary absolute).

Type 6 files are normally created with the SP command (paragraph 2-33); type 4 and 5 files with the SA command (paragraph 2-32). These files can be stored to new files with the ST command.

If a type 6 file is created or moved with the maximum file size specified for *namr2* (see paragraph 2-8 on *namr* file size) the unused disc track space will not be released automatically at the end of the store. The :PK command on the subchannel must be given to recover the unused space.

The file size subparameter for *namr2* defaults to one-half a system disc track (24 sectors) when *namr1* is a device, otherwise file size defaults to size of *namr1* without extents.

Record Format

If record format is omitted, the file type of *namr1* is used to derive the format:

if file type is 0 (non-disc files) 3 or 4 (disc files) record format is AS (ASCII).

if file type is 5, then record format is BR (binary relocatable)

if file type is 7, then record format is BA (binary absolute)

The choices for record format are:

AS	ASCII records are transferred
BA	binary absolute records are transferred; checksum is performed (see Appendix C for format)
BR	binary relocatable records are transferred; checksum is performed (see Appendix C for format)
BN	binary relocatable records are transferred; without checksum (see Appendix C for format)
MT	magnetic tape ASCII records are transferred (MT record format is identical to AS record format)
MS	magnetic tape SIO (System I/O) records are expected on <i>namr1</i> , standard records are written to <i>namr2</i> (see Appendix C for SIO record format)

Record formats can be combined as follows:

MSBR	magnetic tape SIO binary relocatable records
MSBA	magnetic tape SIO binary absolute records.

Subfiles

Before discussing the remaining parameters, it is necessary to understand subfiles. On non-disc devices, files are separated by end-of-file marks that depend on the device:

magnetic tape	EOF mark
paper tape	leader
line printer	top-of-form indicator (page eject)
terminal	two spaces (two carriage return/line feeds) on output; CTRL/D on input.

If *namr1* is a non-disc device, it may contain many such subfiles; the physical end-of-file for the particular logical unit (device) terminates the collection of files or information. A disc file is terminated by a special end-of-file, - 1, in the first length word of the last record. Disc files may be divided into *subfiles* by zero length records, two length words set to zero. These are logical divisions that usually result from transferring non-disc files to disc. (Refer to Appendix C for file formats).

Subfiles are useful when you want to save more than one tape file or relocatable program file (such as a library) on a single disc file and retain the files as separate entities. Subfiles can subsequently be stored (ST command) or dumped (DU command) to another file separately or as a single file. See Figure 2-2.

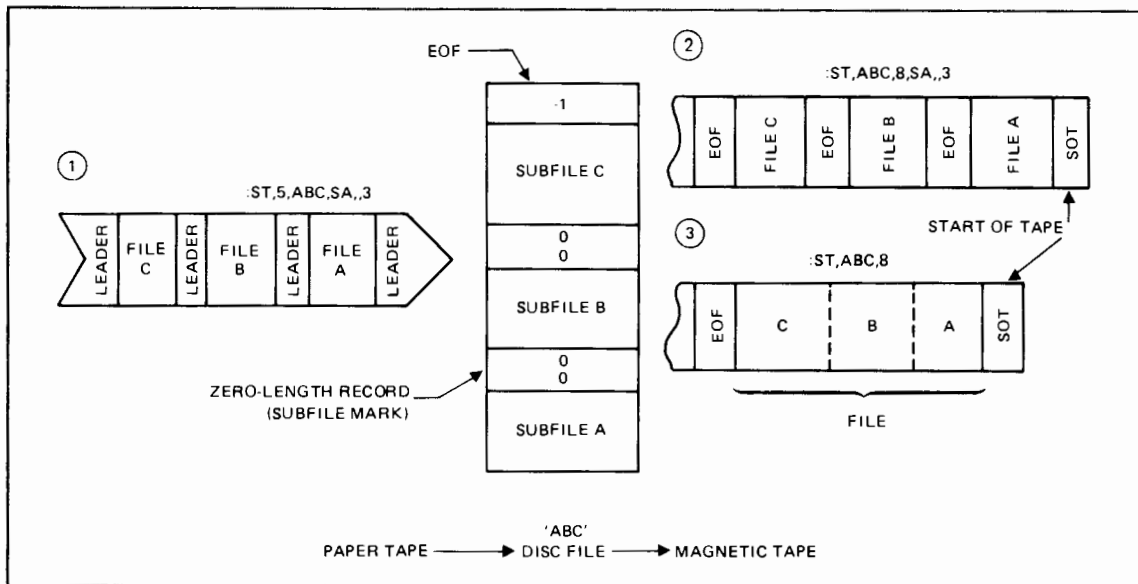


Figure 2-2. Relation of Files to Subfiles

EOF Control

If *eof control* is omitted, an end-of-file mark is written on *namr2* following the last data transferred; on paper tape, leader is punched at the beginning of the file as well as at the end. Any zero-length records on disc or embedded end-of-file marks on non-disc files are not transferred to *namr2* (see ③ in Figure 2-2).

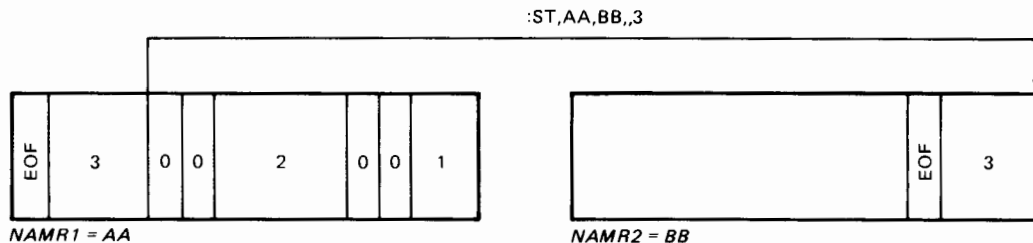
If specified, *eof control* is one of the following:

- IH inhibit the terminating end-of-file; useful only if *namr2* is a non-disc file. On paper tape punch, inhibits initial leader.
- SA transfer embedded end-of-file marks or subfile separators from *namr1* to *namr2*

When SA is specified, the embeded end-of-file marks are converted to the form used by *namr2*. For instance, if a paper tape file is being stored on disc, the leader is converted to zero-length records (① in figure 2-2) and zero-length records to EOF marks on magnetic tape (② in the same figure).

File Number

The *file #* parameter applies to files on non-disc devices or subfiles on disc. It specifies which file or subfile relative to the first with which to start the transfer. For instance, if *file #* = 3, transfer starts with the third file or subfile. If omitted, transfer starts at the beginning of *namr1*. Transfer is always to the beginning of *namr2*.



FOR ST, FILE# SPECIFIES FILE OR SUBFILE IN NAMR1.

Number of Files

The *#files* parameter applies to files on non-disc devices or subfiles on disc. It specifies the number of files or subfiles to transfer starting with *file #*. *File #* and *#files* are specified or omitted under the following circumstances:

Transfer:	<i>file #</i>	<i>#files</i>
1 particular non-disc file or the rest of a disc file	yes	no
1 non-disc file or all of a disc file	no	no
a number of non-disc files or disc subfiles from beginning of file	no	yes
a number of non-disc files or disc subfiles from particular file or subfile	yes	yes

Examples

1. Transfer contents of disc file XYZ to new file AA:

```
:ST,XYZ,AA
```

2. Copy a file from cartridge 2 to cartridge 17:

```
:ST,XYZ::2,XYZ::17    file XYZ on cartridge 2 is not affected
```

3. Duplicate a paper tape:

```
:ST,5,4    ← the contents of tape on LU 5 are punched on a new tape
```

4. Enter ASCII data on file FILEX from the system console, creating FILEX:

```
:ST,1,FILEX    record format default is ASCII, type 3 file is created
}
} ← enter lines of ASCII data; terminate each line
} ← with a carriage return
Dc ← press CTRL and D keys simultaneously to terminate input
```

5. Store a command file on magnetic tape and then execute the commands:

```
:ST,1,8
:DL } ← two commands stored on magnetic tape (LU 8)
:EX }
Dc

:CN ← rewind the tape
::8 ← transfer control to LU 8

directory list is printed on list device by :DL
$END,FMGR ← and FMGR terminates (:EX)
```

Remember that you must enter the colon prompts for all commands entered into a file for later use.

6. Store third, fourth and fifth files from a paper tape to disc as one file with an end-of-file mark at the end:

```

:ST,5,XYZ,,3,3
FMGR006 ← after storing first file, reader stops and FMGR suspends
*GO,FMGR ← type GO,FMGR to continue
FMGR006
*GO,FMGR
FMGR006
* ← when FMGR suspends the third time, the three files
   have been read to disc as one file XYZ
    
```

7. Transfer first five subfiles on disc file DFILE to magnetic tape as five separate files:

```

:ST,DFILE,8,MT,SA,,5
                    ↑
                    |
                    | default for file# is 1 (first subfile)
    
```

2-23. DU - TRANSFER DATA TO EXISTING FILE

The dump command (DU) transfers data from an existing file or logical unit to another existing file or logical unit; unlike ST, a new file is *not* created in the process.

Format

```

:DUMP,namr1,namr2 [ ,record format,eof control [ [ ,file # [ ,#files ] ] ] ]
                  ,eof control
                  ,record format
    
```

NOTE

Only one place-holding comma is required when both *record format* and *EOF control* are omitted.

Parameters

namr1 File name of existing file or non-disc logical unit number; data is transferred from *namr1*. (See *namr* description, paragraph 2-8).

namr2 Name of existing file or non-disc logical unit number to which data is transferred. (See *namr* description, paragraph 2-8).

record format Format of data being transferred; default is derived from *namr1* or is ASCII. Record format description for ST command (paragraph 2-22) applies to DU EXCEPT that MS defines format of *namr2*, and standard format is expected on *namr1*.

<i>eof control</i>	SA to transfer end-of-file or subfile marks from <i>namr1</i> to <i>namr2</i> ; IH to inhibit end-of-file on <i>namr2</i> ; subfile marks not transferred. If omitted, end-of-file is written at end of data on <i>namr2</i> (Refer to EOF Control description under ST command, paragraph 2-22).
<i>file #</i>	Positive integer indicating file (or subfiles) relative to beginning of <i>namr2</i> to which beginning of data is transferred; default is 1. Note that <i>file #</i> here applies to the file receiving data, in ST command it applies to the file from which data is transferred.
<i>#files</i>	Positive integer indicating number of non-disc files or disc subfiles to be transferred; default is 1, unless <i>namr1</i> is a disc file and <i>file #</i> is omitted in which case default is 9999.

NOTE

Files are transferred record by record; records longer than 128 words are truncated.

namr1 and *namr2* must both be existing disc files or logical units defined for the system. *namr2* may be a type 0 file.

Record Format

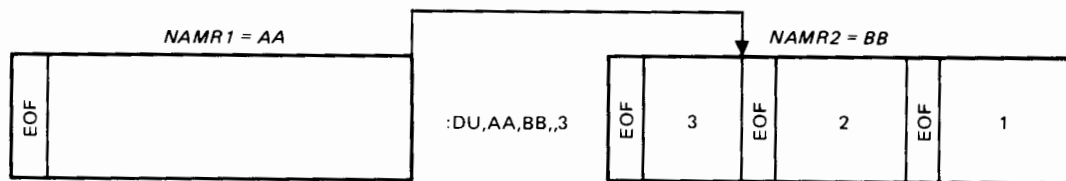
record format is identical to that described for the ST command except for MS. When data is stored (ST), MS refers to the format of *namr1*; when data is dumped (DU), it refers to the format of *namr2*.

EOF Control

The subfile concept is the same for DU as for ST; that is, by specifying *eof control* as SA, end-of-file marks on non-disc devices or on disc files are saved during the transfer. If *eof control* is IH, no end-of-file is written after the last data transfer nor are end-of-file marks saved. On paper tape, the initial leader is not punched. IH allows you to combine two files on *namr1* into one file on *namr2*.

File Number

file # specifies at which file or subfile on *namr2* transfer begins. This feature allows you to append one file to another or to replace one file with another. For example, specify *file #* as 3 to skip the first two files or subfiles on *namr2* and append a third file or subfile. Transfer is always from the start of *namr1*.



FOR DU, FILE# SPECIFIES FILE OR SUBFILE IN *NAMR2*

Number of Files

#files is exactly the same for DU as for ST; that is, it specifies the number of files or subfiles to transfer from *namr1* to *namr2*. It need be specified only when more than one file on a non-disc device or less than a full disc file is to be transferred.

Examples

1. Dump contents of MYFILE to paper tape:

```
:DU,MYFILE,4
```

2. Transfer three files from magnetic tape and one file from paper tape to disc:

```
:ST,8,A1,AS,IH,1,3  write 3 files from LU 8 as one file on A1; inhibit EOF mark
```

```
:DU,5,A1,AS,2 ←———— append 1 file from paper tape and write EOF on A1
```

3. Transfer three files from paper tape to disc file WXY as three subfiles and then transfer the first two subfiles to magnetic tape as two files following an existing file on the magnetic tape:

```
:ST,5,WXY,SA,1,3 ←———— transfer 3 files from paper tape to WXY as subfiles
```

```
:DU,WXY,8,SA,2,2  transfer first two subfiles from WXY to magnetic tape
following existing file
```

4. If an ASR35 teleprinter is used to punch tape, it must be created as a type 0 file with LE (leader) specified for EOF. This is because a teleprinter is usually considered a list device and the EOF mark is two spaces. To punch data on an ASR 35 punch:

```
:CR,TTY,1,BO,,LE  creates type 0 file TTY using leader as EOF mark
```

```
:DU,XYZ,TTY,AS ←———— punch ASCII data from file XYZ
```

To punch on a paper tape punch, simply use the logical unit number (see example 1).

5. To add the contents of file B to the end of file A when both are existing disc files:

```
:DU,B,A,,2 ←———— file B is transferred to follow file A
```

2-24. LI - LIST FILE CONTENTS

The contents of a file, file directory information, or data stored on a logical unit can be listed on the list device with the LI command.

Format

```
:LIST,namr [,format [,L1 [,L2]]]
```

Parameters

namr File name or logical unit number; (refer to *namr* description, paragraph 2-8); if file is protected by a negative security code, it must be specified; if cartridge reference number is included, that cartridge is searched for the file name, otherwise, the first file found with that name is listed.

format Specifies list format:

- S ASCII source format
- B binary format
- D directory information only

If omitted, file type determines format: S if file is type 0, 3, or 4; B for all other files types.

L1 Line numbers of file to list in format.

L2 If neither *L1* (starting line) nor *L2* (ending line) are given, the entire file is listed. If *L1* is specified, but not *L2*, one line is listed. If *L1* is greater than *L2*, no lines are listed. If *L1* is not specified, but *L2* is, *L1* defaults to line 1 of the file.

LI lists the specified file record by record. Any binary records longer than 128 words are truncated. On interactive devices, source records are truncated to 66 characters (6 characters are added to provide line numbers and spacing). On a teleprinter, the list starts in column one, on other list devices two blanks precede the list line.

Headings

If *namr* is a file, the listing is headed by:

```
file name T = file type IS ON CR cartridge USING file size BLKS R = record size
```

where the italicized words are replaced by the actual values in the file directory for the file (see examples).

If *namr* is a logical unit, then a brief heading is printed with asterisks replacing the file name:

```
***** T=00000 IS ON LU nn
```

where *nn* is the logical unit number.

Directory Format

When D is specified, one of the headings shown above is all that is listed.

Source Format

When S is specified, each line number followed by a line of text is printed. Lines may not exceed 66 characters; longer lines are truncated.

Binary Format

When B is specified, the record number is printed followed by each word of the record. Words are printed in octal followed by an ASCII equivalent if a legal ASCII character corresponds to the octal. The ASCII is separated from the octal by an asterisk. Lines are truncated after the last non-blank character (asterisks are treated as blank characters in this case). Binary format prints eight words per line, using as many lines as are needed to print the record up to the maximum of 128 words.

For zero-length records, only the record number is printed.

Examples

```
:ST,1,AA
FIRST RECORD FILE AA
Dc
:ST,1,BB
FIRST RECORD FILE BB
Dc
:DU,BB,AA,,2
```

} ← create two ASCII files, AA and BB, each with one record; dump BB to AA

1. Source Listing:

```
:LI,AA ← S is default for type 3 or 4 files
```

```
AA T=00003 IS ON CR00002 USING 00001 BLKS R=0000
```

```
0001 FIRST RECORD FILE AA
0002 FIRST RECORD FILE BB
```

2. Binary Listing:

```
:LI,AA,B
```

```
AA T=00003 IS ON CR00002 USING 00001 BLKS R=0000
```

```
REC# 00001
```

```
043111 051123 052040 051105 041517 051104 020106 044514*FIRST RECORD FIL
042440 040501 *E AA
```

```
REC# 00002
```

```
043111 051123 052040 051105 041517 051104 020106 044514*FIRST RECORD FIL
042440 041102 *E BB
```

3. Directory Listing:

```
:LI,AA,D
```

```
AA T=00003 IS ON CR00002 USING 00001 BLKS R=0000
```

2-25. CN - CONTROL NON-DISC DEVICE

A non-disc device can be controlled with the operator command CN. This command is similar to the FCONT program call (see Section III) but can be used to reference logical units directly as well as type o files.

Format



:CN[,*namr* [,*function* [,*subfunction*]]]

Parameters

namr Logical unit number of the device or its type 0 file name previously defined in a CR command; default is 8 (recommended logical unit for magnetic tape).

function Function code corresponding to FCONT function codes; two-character mnemonic may be used instead of the octal function code:

<i>mnemonic code</i>	<i>octal function code</i>	
RW (rewind)	4	(default for magnetic tape/2644 Cartridge Tape Unit/mass storage)
EO (end-of-file)	1	(magnetic tape/2644 Cartridge Tape Unit)
TO (top-of-form)	11	(default line printer, terminal)
FF (forward space file)	13	} (file spacing) } magnetic tape/2644 Cartridge Tape Unit
	14	
FR (forward space record)	3	} (record spacing)
BR (backspace record)	2	
LE (leader)	10	(default paper tape punch)

numeric code octal function code in range 0 through 37 corresponding to bits 6-10 of function code in FCONT (paragraph 3-21).

subfunction Carriage control characters for line printer or terminal; use if *function* is TO (top-of-form); it may be:

- 0 to suppress spacing on next print operation only
- +*n* to space *n* lines before next print operation
- n* to page eject on line printer or space *n* lines on terminal

The function default values are determined from the driver type of the device. For driver types greater than or equal to 17, RW is the default. For driver types less than 17, the default is the standard FMGR end-of-file. Depending on the device, this is:

TO line printer page eject or 2 spaces on a terminal
 LE leader on paper tape punch

For driver type 05 on subchannel 1 or 2 (HP 2644 Cartridge Tape Unit), the default is RW.

See Appendix C for a list of the driver types and their associated devices.

Top of Form

TO may be used to specify a particular form of line spacing other than top of form on the line printer. The desired spacing may be specified as the *subfunction* code as described under CN format.

Numeric Code

Any function code included in the FCONT description (paragraph 3-21), including those for which CN provides a mnemonic code, can be specified as an octal integer. The main purpose for the numeric code is to provide control not included as a mnemonic code.

Examples

1. To rewind magnetic tape:

:CN ← *note default to LU 8, magnetic tape*
 or
 :CN,8 ← *also, rewind is default for magnetic tape*
 or
 :CN,8,RW

2. To eject to top of new page on the line printer:

:CN,6
 or
 :CN,6,TO

3. To space two spaces on the system console and return the carriage:

:CN,1
 or
 :CN,1,TO

4. To punch leader on paper tape:

:CN,4
 or
 :CN,4,LE

5. Skip 5 spaces on the line printer (no page eject):

:CN,6,TO,5

6. To space forward one record on magnetic tape:

:CN,8,FR

7. To backspace one file on type 0 file MT assigned to LU 8 at creation:

:CN,MT,BF

8. To write end-of-file mark on magnetic tape:
:CN,,EO
9. Use numeric code to rewind magnetic tape off-line:
:CN,,5
10. To disable and then enable a terminal associated with logical unit 9:
:CN,9,21B ← disable terminal
:CN,9,20B ← enable terminal

2-26. RN - RENAME A FILE

An existing, but closed, disc file can be renamed with the RN command; none of the file characteristics except the name are changed.

Format

```
:RN,namr,name
```

Parameters

<i>namr</i>	File name and, optionally, security code and cartridge reference number of existing file; (see <i>namr</i> description, paragraph 2-8).
<i>name</i>	New file name to replace existing file name

NOTE

Subparameters of *namr* may not be changed.

The new name must be unique to the FMGR cartridge to which the existing file is allocated. If the file was created with a non-zero security code, then the *namr* in this command must include that code. If a cartridge reference number is included in the *namr*, then only that FMGR cartridge is searched. If the CR number is omitted, all mounted cartridges are searched and the first file found with the corresponding *namr* is renamed. The search starts with the first cartridge in the cartridge directory.

Examples

1. :RN,MYFILE:-25:100,MF search for MYFILE on cartridge number 100 and, if the security code is correct, change its name to MF.
2. :RN,URFILE,FILEA search cartridges for first file named URFILE and change its name to FILEA; the file is not protected by a security code.

2-27. PROGRAM FILE MANIPULATION

You can control your program compilation or assembly, loading, and execution directly through RTE, but you may prefer to save the program as a file on disc or as a non-disc device and use FMGR to control the program file. FMGR simplifies the storage and manipulation of programs by treating them as files.

Any of the commands that create and manipulate files apply equally to files that contain programs. In addition, a set of FMGR commands apply specifically to program files. Any type 3 file can be used as a source program file. Other file types greater than 3 specify particular program formats:

- source programs (type 4)
- binary relocatable programs (type 5)
- memory-image programs (type 6)
- binary absolute programs (type 7)

File types 4, 5, and 6 apply to different stages in program development. In the first stage, source programs are moved into LS (logical source) areas by EDITR or with the FMGR MS command. A program in an LS area can be edited, if necessary, and then compiled or assembled. If the relocatable binary output from compilation or assembly was moved to the LG (load-and-go) area, it can be loaded into the system preparatory to execution. A loaded program in memory-image format can be executed by the RU command in RTE or FMGR. Programs in any of these stages can be saved as disc files for later use. FMGR insures that the file is the correct type. As disc files they can be dumped to non-disc devices unless the file is type 6. The commands in this part of the manual provide control over these stages of program development.

There are no special FMGR commands dealing with binary absolute programs (type 7). Such programs can be created, stored, or dumped just like any other file; the correct format depends on your specification of file type in a CR command or of record format in the DU or ST commands.

Logical Source Areas

The logical source (LS) areas are allocated on the system or auxiliary disc for source programs to be assembled or compiled. Each LS disc area is allocated in units of whole tracks. The number of tracks specified depends on the source program size. The system maintains a pointer to the last source program moved to an LS area. You may reset or clear this pointer with the LS command. Source programs are compiled or assembled from the current LS area when the system disc is the input device for the compiler or assembler. You may move a program to an LS area with the MS command, and save a program from an LS area as a type 4 file with the SA command.

LG Area

One area on the system or auxiliary disc can be allocated for programs that have been compiled or assembled and are ready to be loaded. The number of tracks in this area must be specifically declared with the LG command before running an assembler or compiler. The relocatable binary program resulting from assembly or compilation may be placed in the LG area you have reserved. The LOADR program expects a binary relocatable program to be in the LG tracks by an LG command. Any program segments must be loaded into the LG area with the main program. If a main program or segment uses subroutines not in a resident library, these also must be loaded into the LG area.

The LG area must have enough tracks to hold the main program plus any segments or subroutines. The LG area is cleared automatically when the LOADR program terminates normally or at the end of a job by the FMGR EOJ command. You may also clear the LG area with the LG command. A file in the LG area can be saved with SA as a type 5 file. A type 5 file can be moved to the LG area with the MR command.

Memory-Image Programs

A program to be executed must be in memory-image format, the format that results from running the LOADR program. To save such a program on disc as a type 6 file, you must use the SP command. A type 6 file can be restored, executed, and removed with the RU command, or simply restored with the RP command. When restored, an ID segment is set up for the program in memory.

Program ID Segments

The number of programs RTE can accommodate at one time is limited by both disc space and the number of blank ID segments allocated at system generation. The FMGR commands give you methods to work around these limitations.

Every program must have a system-resident ID segment in which the system maintains information about the program: its name, disc location, memory location for execution, and so forth. Programs loaded at generation are permanent; their ID segments are maintained with the system on the system disc and thus are restored whenever the system is restarted with the bootstrap loader. Only the LOADR (or a new generation) can replace or delete permanent programs or add new permanent programs because only the LOADR can alter disc-defined ID segments.

Programs loaded on-line after generation are temporary (unless loaded as permanent programs with LOADR); their ID segments exist in memory but not on the system disc. As a result, ID segments of temporary programs are not restored when the system is restarted with the bootstrap loader and the program is effectively lost.

To prevent losing programs through shutdown and restart, a program can be saved as a FMGR Type 6 file using the SP command. SP creates a FMGR memory-image file containing the program, its ID segment, and its ID segment extension if it has one. The original program can then be deleted with the OF command releasing its system disc space, ID segment, and ID segment extension.

The program file saved with SP can be executed with the FMGR RU command which calls on RP to restore the program and build an ID segment for it in memory. RU then executes the program and, after execution, removes it from the system. The FMGR file containing the program will not be purged.

When RP is used to restore a Type 6 file, it simply builds an ID segment in memory that points directly to the file. The program name inserted into the ID segment is taken from the name of the Type 6 file containing the program. The file can be renamed and restored repeatedly with each resulting ID segment pointing to the original Type 6 file. This is a useful method for making copies without reserving disc tracks for each new copy (for an example, see Appendix E). RP can be used to assign an ID segment to a Type 6 file or to release an ID segment it created.

Essentially the same procedure is followed to process the RU command in an RTE-IV Multi-Terminal Monitor (MTM) environment. However, programs scheduled from a copy of FMGR will be automatically renamed by the system to allow several users to run the same program concurrently. For more information, refer to Section II in this manual on the RU command; and to the MTM section of the RTE-IV Programmer's Reference Manual.

Segmented Programs

If a program is segmented, each segment has a separate ID segment. Program segments are treated as separate programs when using the FMGR program manipulation commands; that is, they must be saved and restored separately. A short ID segment is used for a program segment wherever possible; if a short segment is not available, then a long ID segment is used.

2-28. MS - MOVE SOURCE FILE

The MS command moves a program on a FMGR file to an LS (logical source) area where it can be edited, compiled or assembled.

Format

```
:MS,namr [,program [,IH]]
```

Parameters

<i>namr</i>	File name or logical unit number of program to be transferred; (see <i>namr</i> description, paragraph 2-8 for parameter format).
<i>program</i>	Name of program to which LS tracks are assigned; if omitted, tracks are assigned to EDITR.
IH	Inhibit the pointer to the LS track from being set, which requires use of the LS command to set pointer; if omitted, pointer is set to the LS track containing <i>namr</i> .

NOTE

A program file to be moved with MS must not have records longer than 128 words.

When MS is entered, the logical unit number and first track of the LS area are reported on the log device as:

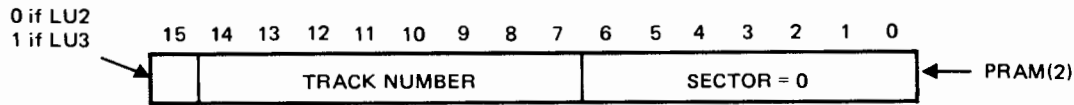
```
FMGR 015
LS LU n TRACK nnn
```

If you use the LS command to set or reset the pointer to this LS track, you will need to know these numbers.

Ordinarily, the LS tracks are assigned to the program EDITR when MS is executed. If, however, you have a reason to assign these tracks to another program, it may be specified as *program*. Care should be taken that the program specified does not release the LS tracks. FMGR, for instance, should never be specified as *program* since all tracks assigned to FMGR are released when the MS command executes and also when FMGR is terminated.

If FMGR was scheduled from a program using the RTE Program Schedule EXEC call with wait, the scheduling program can retrieve the logical unit and track number in the second parameter returned by a call to RMPAR:

```
DIMENSION PRAM(5)
CALL RMPAR(PRAM)
```



Logical source tracks are RTE system tracks so the logical unit number can be only 2 or 3. RMPAR returns the absolute track number (from system track 0, not FMP track 0).

The track number and logical unit of the LS area are important in case the LS pointer is moved and no longer points to the particular area containing your program. This may occur if another user resets the pointer. In this case, use the LS command (paragraph 2-29) to reset the pointer to the area you want.

LS tracks assigned to the EDITR are not released automatically; they may be released with the RT command (paragraph 2-37).

Examples

1. Create a source file from the terminal and then move it to a logical source area for compilation.

```
:ST,1,SPFILE
enter program statements
Dc ← end file with CTRL/D
:MS,SPFILE
FMGR 015
LS LU 2 TRACK 010 ← SPFILE is now on system track 10
```

2. Assume a program on file SOURC 1 is to be moved to an LS area to be edited by EDITR:

```
:MS,SOURC1
FMGR 015
LS LU 2 TRACK 011 ← unless a file is specified, EDITR edits source file in track 11
```

3. If IH is specified, the editor, compilers or assembler will not be able to access the particular logical source until :LS is used to set the pointer:

```
:MS,SOURC2,,IH
FMGR 015
LS LU 2 TRACK 012 ← use this LU and track number in LS command
```

2-29. LS - SET LOGICAL SOURCE POINTER

A pointer to a particular logical source track can be set with the LS command; this command will also clear the current pointer. It is identical to the RTE *LS command.

Format

:LS[, <i>lu</i> , <i>track</i>]	
Parameters	
<i>lu</i>	Logical unit number of LS area; must be 2 or 3 to set pointer, 0 to clear a current pointer.
<i>track</i>	Track number to which pointer is set.
If both <i>track</i> and <i>lu</i> are omitted, the current pointer is cleared;	

Unless the pointer is set to a particular LS area, the source program in that area cannot be referenced. Since the pointer is set whenever MS is executed, LS need be used only if the pointer has been reset to a different LS area. The LS pointer is also set by the EDITR when it is terminated with an /EL *name* command.

Example

```

:MS,AFILE,,IH ← no pointer set
FMGR 015
LS LU 2 TRACK 013
:MS,BFILE
FMGR 015
LS LU 2 TRACK 014 ← pointer set to track 14, file BFILE
.
.
.
:LS,2,13 ← pointer set to track 13,AFILE
:RU,EDITR
/Δ ← space, file in LS area can be edited
.
.
.
:LS ← pointer cleared (same as :LS,0)
    
```

2-30. LG - ASSIGN LG TRACKS

A set of tracks is allocated to the LG area with the LG command; this area must contain binary relocatable programs that are to be loaded with the LOADR.

In an RTE-IV system, the LG tracks needn't be used since the LOADR can directly access binary relocatable files. Refer to the RTE-IV Programmer's Reference Manual for more information.

Format

:LG[, #tracks]	
Parameters	
#tracks	Positive integer indicating the number of tracks to reserve on the system or auxiliary disc for the LG area; if omitted or zero the current tracks are returned to the system.

Whenever LG is specified, any binary relocatable program in the current LG area is cleared. The LG area is also cleared upon successful completion of LOADR and at the beginning or end of a batch job by the JO and EO commands.

If #tracks is not specified, the LG tracks are returned to the system and no tracks are available for a compiled or assembled program. If #tracks is specified, then the current LG area is cleared and a new area is allocated for the LG tracks.

The FMGR LG command is the same as the RTE LG command and the same error messages may be generated.

Example

```

:LG,3 ← allocate 3 tracks to LG area
:RU,ASMB,2,99 ← assemble program source in LS area; relocatable binary to LG
:SA,LG,BFILE ← save binary relocatable program as file
:LG ← return LG tracks to system
    
```

2-31. MR - MOVE RELOCATABLE PROGRAM

A program saved in relocatable binary form (type 5 file) can be moved to the LG area with the :MR command.

In an RTE-IV system, the LG tracks needn't be used since the LOADR can directly access binary relocatable files. Refer to the RTE-IV Programmer's Reference Manual for more information.

Format

:MR, <i>namr</i>	
Parameters	
<i>namr</i>	File name or logical unit number of file to be transferred; may include security code and cartridge reference number; (refer to <i>namr</i> description, paragraph 2-8).

If the file *namr* is not terminated by an END record, FMGR prints the message FMGR 006 when the end of file is reached. FMGR suspends and you should then load the next relocatable module and enter *GO,FMGR. This situation may occur when more than one relocatable binary module is entered through the paper tape reader. If it occurs within a batch job, FMGR is terminated and you must re-schedule it with the RU,FMGR command.

If no LG tracks are allocated when this command is given, the file size is checked and an LG allocation large enough to hold the file is made automatically.

Unlike the MS command, repeated MR commands add each specified file to the same LG area until it is cleared, whereas MS creates a new LS area for each specified file.

Examples

1. A relocatable binary program requiring 2 tracks has been punched on paper tape; to move it to the LG area:

```

:LG,2 ←————— clear and assign two LG tracks
:MR,5 ←————— read program from paper tape to LG tracks
  
```

2. Move type 5 file XX to LG area with 1 track:

```

:LG,1
:MR,XX
  
```

3. Put main program and two segments in LG area, then run LOADR:

```

:LG,4
:MR,PROGA
:MR,SEGA1
:MR,SEGA2
:RU,LOADR,99,,,1 ←————— the last parameter indicates a main program with
                                     segments, 99 that the program is to be found in the
                                     LG area.
  
```

Program PROGA with its segments can now be executed (RU) or saved as a type 6 program (SP).

2-32. SA - SAVE PROGRAM AS FILE

To save the contents of a logical source (LS) area or the load-and-go (LG) area as a file, use the SA command. A file is created by SA.

Format

```
:SAVE ,LS
      ,LG ,namr
```

Parameters

LS	Source program in LS area is saved.
LG	Relocatable program in LG area is saved.
<i>namr</i>	File name or logical unit number of file in which LS or LG is saved; file is created and full <i>namr</i> can be used; (see <i>namr</i> description, paragraph 2-8)

NOTE

The SA command truncates records longer than 128 words; no HP compiler or assembler generates records longer than 128 words.

If there is not enough cartridge space available when *namr* is a disc file, error message FMGR -06 is issued. Any portion of the file that was saved is purged.

Save LS

For SA,LS,*namr* the file type when omitted defaults to type 4.

The file size defaults to one half the number of blocks on a system disc track. After the final end-of-file is written on the new file, its position is checked. If there are no extents, any remaining unused disc space is returned to the system. If extents were created, the file size is not shortened.

Save LG

For SA,LG,*namr*, the file type of *namr* when omitted defaults to type 5. A checksum is provided.

Following each relocatable module (main program, program segment or library subroutine) including the last, a zero-length record is written if *namr* is a disc file or an EOF if *namr* is a non-disc device. (Refer to paragraph 2-20 for the particular device end-of-file.)

The file size, if omitted, is computed as the maximum size possible from the amount of LG area used. Extents are not created and file will be only as long as needed.

Examples

1. **:LS,2,36** ← *set LS pointer to system disc, track 36*
:SA,LS,SFILE:13 ← *save LS area as type 4 file, on cartridge 13; default file size*
2. **:SA,LG,4** ← *punch relocatable program in LG area to paper tape*
3. **:SA,LG,LFILE:JM:-14** ← *save relocatable program as type 5 file on logical unit 14 protected by security code JM*

2-33. SP - SAVE PROGRAM

A disc resident program and its ID segment is saved as a FMGR type 6 file with the SP command. SP creates a file.

Format

:SP,namr

Parameters

namr Defines the type 6 file created by :SP (refer to *namr* description in paragraph 2-8): *namr* cannot be a logical unit number; first five characters of the file name must be identical to the name of the saved program.

Subparameters default as follows:

<i>security code</i>	default to 0
<i>cartridge</i>	default to -2 (system disc logical unit)
<i>file type</i>	forced to type 6
<i>file size</i>	forced to size of program
<i>record size</i>	forced to 128

Any cartridge may be specified, but the file must be on logical unit 2 (system) or 3 (auxiliary) if it is to be restored with RP (see paragraph 2-34).

A program name is a maximum of five characters; a file name can be six characters. This means you can create a type 6 file for more than one version of the same program by adding one character to a five character program name. If, however, the program name is less than five characters, the file name must be identical.

The SP command is usually used in conjunction with the RP command. A program that has been edited, compiled or assembled, has been loaded and then run successfully can be saved in its loaded form for future use with the SP command. To run this program through FMGR, issue either the RU command alone or RP and RU. In either case, the file is restored and then executed.

One reason to save a program with SP is to release its ID segment for use by another program. After saving a temporary disc resident program (a program loaded since last start-up), issue the OF, program command (paragraph 2-36). The ID segment assigned to the program is released for use by another program. The ID segment of a permanent disc resident program (loaded at system generation) can be released only with the LOADR using the command RU,LOADR,,,4.

Examples

1. Save a temporary disc resident program APROG as a type 6 file named APROG1:

```
!SP,APROG1 ←————— save APROG and its ID segment as a file named APROG1
!OF,APROG ←————— delete the original program from disc
```

2. Save a temporary disc resident program LOCA as file LOCA and then rename file TEST01. The new name is used in all future references to the program.

```
!SP,LOCA
!RN,LOCA,TEST01 ←——— new file name is TEST01, program name is TEST0
!RU,TEST01 ←————— when running the program use either file name TEST01
                           or program name TEST0
```

Note that if the file is restored as a program with RP (paragraph 2-34), it can be run with the five character program name, in this case TEST0.

3. Save a permanent program XX, rename it as temporary program YY, then release ID segment for XX (RTE-II/III systems).

```
!SP,XX ←————— save permanent program
!RN,XX,YY ←————— rename it as temporary program YY
!RU,LOADR,,,4 ←————— run LOADR to release ID segment
/LOADR: PNAME?
XX ←————— of permanent program XX
/LOADR: SEND ←————— loader terminates
!
```


2-34. RP - RESTORE PROGRAM

A program file saved with the SP command can be restored with the RP command. RP is also used to assign an ID segment to the restored file, or to release an ID segment so assigned.

Formats

1. Restore program file *namr* :

:RP,*namr*

2. Restore program file *namr* using *program's* ID segment

:RP,*namr,program*

3. Release disc tracks, ID segment, and ID segment extension assigned by previous RP:

:RP,,*program*

Parameters

namr Identifies type 6 file on logical unit 2 or 3 that was saved with SP; security code, and cartridge reference may be specified; (see *namr* description paragraph 2-8); *namr* must not be a logical unit number. *namr* is only omitted in format 3 to release the ID segment of the named *program*.

program 1-5 character name of program whose ID segment is assigned to *namr* (format 2) or released (format 3).

Format 1

A program file, *namr*, is restored as a program that can be accessed by the RTE system commands. It is restored with the same time parameters, priority, and (in RTE-III) partition assignment it had when saved. The first five characters of *namr* define the program name. If the file *namr* has been renamed since it was saved, then the new name is used.

If a program with the same name as *namr* is already in the system, error message FMGR 023 is issued. To avoid this error, it is a good idea to delete any program saved with SP using the OF, program command (see 2-36) or to rename the type 6 file before it is restored.

All programs restored with RP are temporary; they are not recorded in the system area of disc and will not be restored automatically when the system is restarted with the bootstrap loader.

If a blank ID segment cannot be found for *namr*, FMGR 014 is issued. An ID segment may be freed by deleting a program with OF, *program* or by using the *program* option of RP.

Format 2

The restored program can be assigned the ID segment of a previously restored program by using the *program* option. *program* must identify a currently inactive program that was restored with RP. The specified program is removed from the system and an ID segment assigned to *namr*.

RP first releases the ID segment assigned to *program* and then allocates a blank ID segment to *namr*. RP looks for a blank ID segment from a temporary program, but if none are available, it uses a blank segment from a permanent program. Such ID segments occur when a permanent program is purged by RU,LOADR,,,4. (See RTE reference manual). If *namr* is a program segment, RP looks for a blank ID segment in the following order:

1. short blank ID segment from a temporary program.
2. long blank ID segment from a temporary program.
3. short blank ID segment from a permanent program.
4. long blank ID segment from a permanent program.

Format 3

The third format releases the ID segment and tracks of a restored *program*. The ID segment is returned to the system as a blank ID segment that can be used by another program.

If *program* does not exist and thus does not have an ID segment, FMGR 009 is issued but does not cause transfer to the log device. If *program* is currently active, FMGR 018 results; enter OF, *program* and try RP again. If *program's* ID segment was not set up by RP, FMGR 017 is issued; enter OF, *program* and then enter the RP command again.

The program file restored as *namr* can be purged while an ID segment points to it and while it is still running. The executing program is not affected in any way by this action. Before the file space can be recovered with the packing command PK, its ID segment must be released by OF, *program*. Failure to return the ID segment before packing results in a FMGR 011 message.

Examples

1. Restore program file APROG1 as program APROG:

```
:RP,APROG1           file APROG1 is restored as program APROG;  
                    its ID segment is stored in memory
```

2. Restore program file TEST01 as program TEST0 and assign it APROG's ID segment:

```
:RP,TEST01,APROG    APROG must be inactive
```

3. Release ID segment and tracks previously assigned to TEST0:

```
:RP,,TEST0
```

4. *program* and *namr* can be the same name since *program* is cleared from its ID segment before *namr* is restored. To illustrate, assume two type 6 files WW and XX:

```
:RP,XX              restore program XX from file  
:PU,XX              purge file XX  
:RN,WW,XX          rename file WW as XX  
:RP,XX,XX          restore XX (was WW) with ID segment released by XX
```

2-35. RU - RUN PROGRAM

The RUN command searches for and executes a named program. It will also restore, execute and release the ID segment of a program saved as a type 6 file or transfer control to a procedure file. This command can be entered in either of two forms — RU to cause the entire command string to be passed through to RTE with the EXEC schedule call, or RUIH to inhibit command string passage.

In the RTE-IV Multi-Terminal Monitor (MTM) environment, several special considerations apply. These are explained briefly below and more completely in the MTM section of the RTE-IV Programmer's Reference Manual.

Format

	<i>Privileged Command (see paragraph 2-7)</i>
RUN <u> </u> , <i>program</i> [<i>,parameters</i>] , <i>namr</i>	
passes command string, or	
RUIH <u> </u> , <i>program</i> [<i>,parameters</i>] , <i>namr</i>	
inhibits passing of command string, or	
RUN <u> </u> , <i>program</i> :IH , <i>namr</i> :IH	
inhibits renaming feature of RTE-IV Multi-Terminal Monitor environment.	
Parameters	
<i>program</i>	Program name; 5-character name of program to be executed
<i>namr</i>	File descriptor; identifies type 6 file containing program to be executed or a procedure file to which control transfers; should not be a logical unit number (see <i>namr</i> description, 2-8).
<i>parameters</i>	Parameters to be passed to program or procedure file; 1-5 program parameters depending on the program requirements; 1-9 global parameters if <i>namr</i> is procedure file; omitted if no parameters are to be passed.

When RU is executed, a search is made of the system ID segments for the named program. If found, the program is executed. If not found, a search is made for a type 6 file on which the named program is stored. If such a file is found, it is restored, its ID segment is built in memory, and the program is executed. You need not specify RP; RU will insure that RP is performed. Following execution of a program restored from a type 6 file, the program's ID segment is released and any tracks used by the program also are released.

When parameters are specified for a program to be executed, the particular parameters depend on the program. For instance, the assembler (ASMB) and the compilers (FTN and FTN4) have a standard set of parameters that are used when these programs are executed with RU. A

program may pass back up to five one-word parameters with the routine PRTN. The five program parameters are passed back to FMGR in global parameters 1P through 5P. The first three parameters are also returned to FMGR as the global parameter 10G, a three-word ASCII parameter that usually contains a file name (refer to paragraph 2-40 for a description of global parameters).

If *namr* is not a type 6 file that can be executed, then RU is interpreted as a TR command (paragraph 2-41) and control transfers to the named file. This file is assumed to be a procedure file containing commands that can be executed; if not, FMGR 019 results. Parameters in this case are set into the appropriate globals (1G through 9G), (refer to Global Parameters, paragraph 2-40). Up to nine global parameters can be specified in the parameter list for a procedure file.

When a program is scheduled using the RU command form, a section of System Available Memory is allocated for storage of a command string. A command string consists of every item following the prompt character in a scheduling command entry.

Global parameters may be used within any of the RU command parameters. The global parameters are interpreted and the correct values are passed through to the scheduled program.

Because the RU command is privileged (see paragraph 2-7) subparameters are legal with any parameter. This allows passing arbitrary strings to programs.

The string passage capability is described in detail in the appropriate RTE Operating System Reference Manual.

Scheduling a program using the RUIH command form inhibits the passage of command strings.

The following descriptions define system action when a program that was restored (implicit RP) via the RU command terminates:

1. If the running program terminates, is aborted, or terminates serially reusable:
 - a. The system releases program owned tracks, e.g., EDITR, LS, system scratch and compiler scratch tracks.
 - b. The system releases the program's ID segment.
 - c. The system releases program owned RN's, locked RN's and LU's, and program owned memory.
2. If the running program calls EXEC to place itself into the time list and then terminates saving resources:
 - a. The system does not release any resources.
 - b. The program's ID may not be released by RP (the OF command can be used). An attempt to release the program's ID (RU or RP) will result in an error (FMGR 018).
 - c. The program remains within the system.



FMGR Operator Commands

3. If the running program calls EXEC to place itself into the time list and then terminates, or terminates serially reuseable:
 - a. The system releases program owned RN's, locked RN's and LU's, and program owned memory.
 - b. The program's ID may not be released by RP (the OF command can be used). An attempt to release the program's ID (RU or RP) will result in an error (FMGR 018).
 - c. The program remains in the system.

RUN Command in RTE-IV Multi-Terminal Monitor Environment

In an RTE-IV Multi-Terminal Monitor (MTM) environment, MTM manages ID segments so that each user can have his own copy of a program. If the user wishes to run a program with FMGxx as the father (i.e., :RU,PROGX but not :SYRU,PROGX), then in certain circumstances a copy of the program will be created belonging to the particular terminal and run for the user at the terminal. MTM performs this action whenever the program to be run is a son of FMGxx, and the program is either a temporary program or in a type six FMGR file. A copy of the program will be created with the last two characters being xx and scheduled for execution to terminal xx.

For example, if the EDITR is loaded on-line as a temporary load and saved as a Type 6 file, the command.:

```
:RU,EDITR
```

will create a program named EDIxx and schedule it to terminal xx. When EDIxx is finished the ID segment is automatically returned to the system.

The advantage of processing the ID segments in this way is that all terminals can run the same program but each user gets his own copy of the program. Therefore a user does not have to wait for other users to finish with a program before using it himself.

The above procedure will still work properly even if the program to be run has been previously restored using the RP command. In fact, the program will be created more quickly since there would be no disc search time before the program could be run.

The automatic renaming feature of MTM may be circumvented by using a copy of FMGR that does not "belong" to the terminal at which the user is operating. (A copy of FMGR named FMGxx "belongs" to the terminal with logical unit number xx.) In this case, none of the features described for MTM will apply.

The program renaming feature of MTM may also be temporarily inhibited when the user runs a program. The following form of the command should be used:

```
:RU,PROGX:IH
```

The "IH" inhibits the program renaming feature, thereby ensuring that the actual program named PROGX will be run and not a copy.

This ability is especially useful when loading permanent programs. The program named LOADR is the only program that can load programs permanently into the system; a copy of the

LOADR cannot perform permanent loads. Therefore, whenever MTM would normally rename the LOADR making it impossible to do a permanent load, use the following command to load a permanent program:

:RU,LOADR:IH,.....

Examples

- 1. :RUIH,ASMB,2,99 ←————— run program ASMB stored on system disc; inhibit command string passage.
- 2. :RU,APROG1,FILE:7G:-13,FILEN2,6,99 ←————— run program APROG previously saved as type 6 file APROG1; pass command string.
- 3. :RUIH,PROG,2,4,3,99 ←————— run program PROG; inhibit command string passage.
- 4. :RU,TRANSF ←————— transfer control to file TRANSF.

2-36. OF - REMOVE PROGRAM

A temporary program and its ID segment can be removed from the system with the OF command. The command is the same as the system command OF,program,8.

Format

:OFF,program	
Parameters	
program	Names temporary program to be removed; must have been loaded or created since last system restart with bootstrap loader.

The OF command clears *program's* ID segment and ID segment extension and returns any disc tracks used by *program* to the system. The ID segment and tracks become available for use by another program. If executing, the program is terminated by OF. Any segments must be removed with separate OF commands; an OF naming the main program will not remove its segments automatically.

The message PROG ABORTED generated by OF,program,8 is also generated by OF,program.

Removal of a program with OF causes an abort message to be displayed at the system console unless the program is a background disc-resident segment.

Note that permanent programs created at generation or by the loader maintain their ID segments on the disc and thus can be removed only with the loader request: RU,LOADR,,,4.

Examples

- 1. **:OF,APROG** ← *remove APROG and its ID segment from system*
- 2. **:OF,MAIN**
:OF,SEG1
:OF,SEG2 } ← *remove program MAIN and its two segments*

2-37. RT - RELEASE TRACKS

Disc tracks currently assigned to a dormant program can be released to the system with the RT command.

Format

<i>:RT,program</i>	
Parameters	
<i>program</i>	Name of program whose tracks are to be released; must be a dormant program.

If the named program is dormant, all tracks assigned to that program are released. Any released tracks become available to the system and all programs suspended and waiting for disc track allocation are rescheduled. If the named program is not dormant, the request is illegal. Error message ILLEGAL STATUS is issued.

This command can be used to release any LS tracks assigned to the EDITR program. These tracks may accumulate through use of MS commands (paragraph 2-28) and also may be left when EDITR terminates leaving an LS area assigned.

Example

:RT,EDITR *all LS tracks assigned to EDITR are released*

2-38. PROGRAM DEVELOPMENT EXAMPLE

Figure 2-3 illustrates how a program is entered, compiled, loaded, and executed with FMGR commands.

```

*DN,FMGR
:ST,1,&FILE ← store program to file MYFILE from console
FTN4,L
    PROGRAM HELLO
    WRITE(1,10)
    10 FORMAT("HELO")
    END
    ENDS
Dc

:RU,EDITR ← edit program HELLO
SOURCE FILE?
/&FILE
    FTN4,L
/4
    10 FORMAT("HELO")
/R 10 FORMAT("HELO")
/ER ← save corrected copy of source MYFILE

END OF EDIT
:RU,FTN4,&FILE,6,-
    PAGE 0001

0001 FTN4,L
0002     PROGRAM HELLO
0003     WRITE(1,10)
0004     10 FORMAT("HELO")
0005     END

** NO ERRORS**      PROGRAM = 00021      COMMON = 00000

/FTN4: SEND

:RU,LOADR,,%FILE ← load program from file name %FILE
/LOADR:HELLO READY
/LOADR:SEND
:SP,HELLO ← save memory image program as type 6 file HELLO
:RU,HELLO ← execute program HELLO
HELLO
:OF,HELLO ← release HELLO ID segment and tracks
HELLO ABORTED
:RT,EDITR ← release EDITR tracks
:EX
SEND FMGR
    
```

Figure 2-3. Compile and Execute Program

2-39. PROCEDURE FILE MANIPULATION

Instead of entering each command at the terminal, groups of consecutive commands can be saved as a procedure file (also known as a transfer file). Using the transfer command (TR or :), you can then transfer control to the procedure file and the commands will be executed. This is useful particularly when the same set of commands, say to compile, load, and run a program, are to be used over and over.

A procedure file can be generalized with global parameters. These are a set of general purpose parameters which can replace any command parameters in a file of commands. The TR command allows you to pass particular values to the global parameters when you transfer to the file.

A set of commands that manipulate parameters may be useful within a procedure file as mini-language. These commands:

- Assign values to or clear values from global parameters
- Calculate values and assign them to global parameters
- Display the values of any parameters
- Compare the values of any two global parameters and skip according to the result

Each procedure file is terminated with TR or : in order to return control to the point of origin. Up to ten procedure files can be nested. The return is normally to the command following the TR in the preceding procedure file. If files are nested, the return may skip to any of the preceding files.

When procedure files form part of a batch job (refer to paragraph 2-47), values of global parameters are cleared upon entry to the job and again upon exit from the job (refer to JO command, paragraph 2-48). Otherwise, global values set in a procedure file are not altered upon return from the file. They may be set with the TR command upon transfer to the file in order to pass values to the file.

2-40. GLOBAL PARAMETERS

Global parameters are variables that may be set, examined, and manipulated by FMGR commands. Global parameters may replace any FMGR command parameter. When used in procedure files, they are similar to formal parameters in a procedure or subroutine to which actual values are passed through the TR command. There are three ways to access the global parameters: as G globals, P globals, or S globals. Each global is identified by a variable name that combines an integer with the letter G, P, or S. (Refer to Table 2-2 for the relation between G, S, and P globals.)

G Globals

The 11 G-type globals are named 0G through 10G. Globals 1G through 9G can be set or altered with the TR, CA, or SE commands and may be assigned any values: they may be null (no value assigned), have a numeric value, or contain up to six ASCII characters.

Globals 0G and 10G have particular values: 0G is set to the *namr* value of the input device with which FMGR is scheduled or, for batch job processing, to the job input file or logical unit. 10G is set by a PRTN call in a program executed by the FMGR RU command. Notably, it is set to the name of the loaded program following the command :RU,LOADR. 10G is always ASCII.

P Globals

There are seven P globals, 1P through 7P. The first five P globals represent integer values returned by a program executed with the RU command. Global 6P is the current error code and 7P is the current severity code. The first three P globals correspond to global 10G; that is, 1P is the integer equivalent of the first two characters in 10G, 2P of the second two characters, and 3P of the last two characters. Whenever it is necessary to reference any of these three values separately, the P globals can be used instead of 10G.

The RU command results in the first five P-type globals being set only when the program passes back parameter values in a call to PRTN. These parameter values may be used as parameters in subsequent commands. To illustrate:

```
:RU,PROG1 ← returns values to FMGR in 1P through 5P (10G, 4P and 5P)
           (PROG1 must call PRTN to set up 1P through 5P)
:RU,PROG2,1P,2P,3P,4P,5P ← PROG2 can retrieve values from 1P - 5P
           └──────────┬──────────┘
                   10G
```

The P-type globals are particularly useful when one-word parameters are expected. To illustrate:

```
:RU,XYZ,-27P,-26P,-25P ← passes global 3G to program XYZ
:RU,ABC,1P,2P,3P ← passes global 10G to program ABC
```

Global 6P can be cleared (set to zero) using either the CA or ?? command. To clear 6P, specify:

```
:CA,6:P,0 ← (see paragraph 2-45)
or
:?? ← (see paragraph 2-11)
```

An example showing one way of using global 6P follows.

Assume a program named ABLE having two segments, ABLE1 and ABLE2, is to be run from a transfer file called DOABLE.

```
:TR,DOABLE,P1,P2,P3,P4,P5 ← transfer to file DOABLE
```

The file DOABLE contains the following commands:

```
:SV,4,6,IH ← Save severity code in global 6G; inhibit
              command echo and error transfer.
```

FMGR Operator Commands

<code>:CA,6:P,0</code>	←	Set global 6P to zero.
<code>:RP,ABLE1::2</code>	←	Restore segment ABLE1.
<code>:RP,ABLE2::2</code>	←	Restore segment ABLE2.
<code>:IF,6P,NE,0,1</code>	←	If global 6P is not equal to zero, skip over next command (skip 1 command on error).
<code>:RU,ABLE,1G,2G,3G,4G,5G</code>	←	Run program ABLE; pass parameters.
<code>:RP,,ABLE1</code>	←	Purge ABLE1 ID.
<code>:RP,,ABLE2</code>	←	Purge ABLE2 ID.
<code>:IF,6P,EQ,0,1</code>	←	If global 6P equals zero, skip over next command (skip 1 command if no error).
<code>:DP,ABLE RUN ERROR FMGR,6P</code>	←	Display a message and global 6P.
<code>:??</code>	←	Set 6P to zero.
<code>:SV,6G</code>	←	Restore original severity code value from 6G

As shown in Table 2-2, 0S is identical to -2G and 7S is identical to 5G. The standard values defined above (0G - 10G, 1P - 5P, and 0S, 1S) are shown within the dark lines.

Table 2-2. Global Equivalence

S	G	P
∅	-2	-48 Type
		-47 1
		-46 2
		-45 3
1	-1	-44 Type
		-43 1
		-42 2
		-41 3
2	∅	-40 Type
		-39 1
		-38 2
		-37 3
3	1	-36 Type
		-35 1
		-34 2
		-33 3
4	2	-32 Type
		-31 1
		-30 2
		-29 3
5	3	-28 Type
		-27 1
		-26 2
		-25 3
6	4	-24 Type
		-23 1
		-22 2
		-21 3
7	5	-20 Type
		-19 1
		-18 2
		-17 3
8	6	-16 Type
		-15 1
		-14 2
		-13 3
9	7	-12 Type
		-11 1
		-10 2
		-9 3
10	8	-8 Type
		-7 1
		-6 2
		-5 3
11	9	-4 Type
		-3 1
		-2 2
		-1 3
12	10	∅ Type
		1 1
		2 2
		3 3
13	11	4 4
		5 5
		6 6
		7 7

The standard values are shown within dark lines.

S Globals

S-type globals are set by the FMGR commands JO or LU when running under control of the Spool Monitor (refer to Section IV). They may be referenced by FMGR commands within spooled jobs.

Two S globals have a particular meaning: 1S is null or is the ASCII file name of the last created spool file; 0S is null or is the spool logical unit number of the last spool file set up (see the LU Switch Table in Figure 4-4). 0S and 1S are always null when spooling is not being used.

Global Format

All globals are kept in an array in memory. G and S globals use four-words accesses; P globals use one-word accesses. Table 2-3 illustrates the format of G and S globals.

Table 2-3. G and S Global Format

word 0	global type = 0 (null)	1 (numeric)	3 (ASCII)
word 1	0	integer	characters 1,2
word 2	0	0	characters 3,4
word 3	0	0	characters 5,6

Word zero defines the global type. If the type is null, the remaining words are also 0. If the type is numeric, then word one contains the integer value and the last two words are zero. An ASCII global (type 3) can contain up to six ASCII characters. This format applies only to G and S globals. Each P global corresponds to one word of the S and G globals (refer to Table 2-2).

Since the system checks a global access by its position in the array (Table 2-2), a P global can be used to reference one word of an S or G global, and an S global can be used to reference a G global, or vice versa.

2-41. TR - TRANSFER CONTROL

The TR command transfers control to a file or logical unit, optionally passing values to the globals 1G through 9G.

Format

$:\text{TRANSFER} \left[\begin{array}{l} ,\textit{namr} \\ , -\textit{integer} \end{array} \right] [\textit{parameters}]$	
OR	
$:: \left[\begin{array}{l} \textit{namr} \\ -\textit{integer} \end{array} \right] [\textit{parameters}]$	
Parameters	
<i>namr</i>	Identifies file or logical unit to which TR transfers; (refer to <i>namr</i> description, paragraph 2-8); if omitted, TR returns control to the <i>namr</i> of a previous transfer; up to 10 transfers can be nested - a stack of return pointers is saved.
<i>-integer</i>	Indicates a transfer back the specified number of files in the nested stack.
<i>parameters</i>	The values to be set into the globals 1G through 9G; position determines to which global the value is passed; omitted globals are unchanged.
NOTE	
A comma (,) may replace TR or the colon (:) as the transfer command code.	

Whenever a TR command is executed, the *namr* associated with TR is saved in a stack that can contain up to ten file names or logical unit numbers. This allows the system to transfer back when TR or TR, *-integer* is specified. If the *namr* identifies a disc file, the current record number is also saved. The transfer returns to the point immediately following the specified TR command.

When transferring back to a control file on disc, it is possible to backspace and re-execute one or more commands within that file. To accomplish this, specify a negative integer (*-integer*) as the security code of a null *namr*. For example:

```

:TR, :-2
or,
:TR: :-2
or,
:: :-2

```

will result in a backspace of two commands while transferring back to the previous control file.

NOTE

For non-disc devices on which a backspace is legal, e.g., LU5 in spooled jobs, the backspace record function of the CN command (see paragraph 2-25) can be used (:CN,5,BR).

If TR is specified with no parameters and a previous *namr* cannot be found, FMGR terminates. In this case, it acts exactly like the EX command.

If an error requiring operator intervention occurs, the system transfers control to the log device. The TR command (TR or :) is used, in this situation, to return control to the procedure file or input device where the error occurred.

When parameters are specified, the parameter values are passed to the global parameters 1G through 9G. Position determines which globals receive values. For example:

```

:TR,PROG2A,,,ABC,3270,,XYZ      sets 4G to "ABC", 5G to 3270, 7G to "XYZ"
      ^   ^   ^   ^
      1G  2G  3G   6G
    
```

Examples

1. Transfer to generalized procedure PROC1 to compile, load, and run a FORTRAN IV program, FILEA:

```

:TR,PROC1,FILEA ← sets 1G to FILEA and transfers to PROC1
: ← TR in PROC1 returns control here
    
```

```

file "PROC1"
:RU,FTN4,1G,'TEMP,%TEMP ← 1G is source program name
:RU,LOADR,,%TEMP
:RU,10G ← LOADR sets 10G to name of
:OF,10G ← loaded program
:TR
    
```

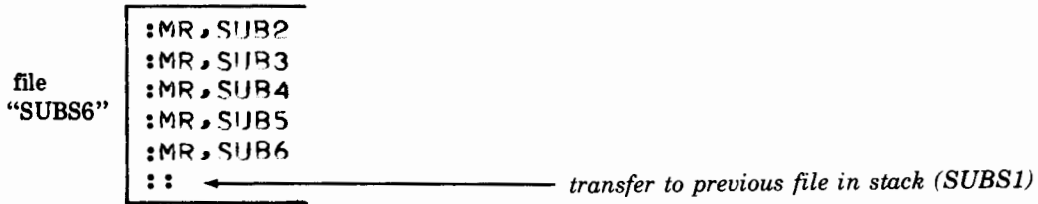
2. To transfer to one procedure that transfers to another nested procedure and returns through the first procedure to the command following the original TR. The LG area is used to build up a relocatable program to be stored in file %TEMP.

```

:LG,3 ← both procedures require LG area
:RU,FTN4,99
:TR,SUBS1 ← transfer to SUBS1; SUBS1 transfers to SUBS6
:SA,LG,%TEMP ← return here from SUBS1
:RU,LOADR,,%TEMP
:RU,10G
    
```

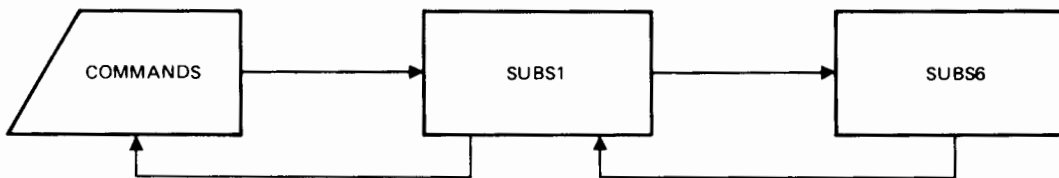
```

file "SUBS1"
:MR,SUB1
:TR,SUBS6 ← transfer to procedure file SUBS6
:MR,SUB7 ← return here from SUBS6
:MR,SUB8
:: ← transfer back to command after first TR
    
```

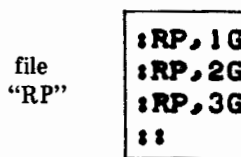


The procedure files, in this case, are both used to move subroutines to the LG area prior to loading with the main program.

The flow may be illustrated as:



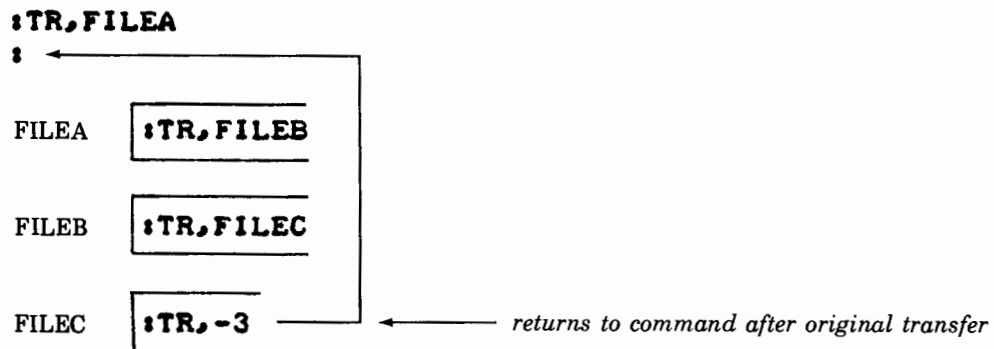
3. Set up procedure file RP to restore a main program and two segments or three main programs or two main programs and a segment:



Transfer to RP to restore program MAIN and two segments SEG1 and SEG2:

::RP,MAIN,SEG1,SEG2

4. Assume three procedure files, FILEA, FILEB, and FILEC stacked so that FILEA transfers to FILEB to FILEC:



FMGR Operator Commands

5. Assume a control file, CNTFL, containing a command to store file name ABC into file name DEF on cartridge 3. When the command is executed, the directory on cartridge 3 is full resulting in an error. A solution is to pack cartridge 3 and then backspace 1 line while transferring back to file CNTFL, as follows:

```
:TR,CNTFL ←————— transfer to control file  
  
file CNTFL  
  :ST,ABC,DEF::3 ←————— command line in control file  
  
FMGR-006 ←————— error message  
:PK,3 ←————— pack cartridge 3  
:TR,-1 ←————— transfer back one line in control file
```

2-42. PA - PAUSE AND SEND MESSAGE

The PA command suspends execution of the current job or procedure file and transfers control to the log device or to some other specified device. Optionally, PA displays a message. PA is used only to send messages during non-interactive processing.

Format

	<i>Privileged Command</i>
: <u>PAUSE</u> [, <i>lu</i> [, <i>message</i>]]	
Parameters	
<i>lu</i>	Logical unit number of device to which control transfers and where message is displayed; if omitted, log device is assumed.
<i>message</i>	Message to be displayed on <i>lu</i> ; must conform to parameter syntax rules for privileged commands (refer to paragraph 2-7).

The PA command causes a transfer to the log device or specified logical unit where the entire command line is printed. The message, if any, must conform to the syntax rules for any parameter (refer to paragraph 2-7).

When job processing is suspended with PA, it may be continued by entering a TR or : command on the device to which control transferred. Control returns to the command following PA.

FMGR Operator Commands

The P-type globals can be used to display the global type (refer to Table 2-3 for the particular P global corresponding to the type of G or S global).

The display is not inhibited by a severity code greater than 0.

Examples

1. Display non-global parameters:

```
:DP,WORD,10B,-1,-32768,50  
WORD,10B,-1,-32768,50
```

2. Display values of globals 1G and 2G passed in a TR command:

```
:TR,PROCX,PROG1,1 ← transfer to PROCX  
  
file  
"PROGX" :DP,1G,2G ← within PROCX, DP command requests display  
  
PROG1,1 ← display on log device
```

3. Assume 1G and 2G have values set in previous example and that value of 3G is null, display types of 1G, 2G, and 3G:

```
:DP,-36P,-32P,-28P  
3,1,0  
↑  
ASCII numeric null
```

2-44. SE - SET GLOBAL PARAMETERS

The SE command is used to assign values to the global parameters 1G through 9G.

Format

```
:SET[,p1 [,p2 [ . . . [,p9 ]]]]
```

Parameters

p1 through *p9* Values assigned to globals 1G through 9G; if all parameters are omitted, globals are nulled; if any one parameter is omitted, corresponding global is unchanged.

The value in *p1* is assigned to 1G, that in *p2* to 2G, and so forth. By using SE alone with no parameters, all the globals 1G through 9G can be cleared (nulled). Individual globals can be nulled or set with the CA COMMAND (paragraph 2-45). Any integer value between -32767 and 32767 or an ASCII value up to six characters can be assigned to *p1* through *p9*. If, however,

you enter a global name (0G - 10G, 1P - 5P, 0S or 1S) as a parameter, the value of the specified global is assigned to the global corresponding to the parameter position.

Examples

1. **:SE,,,,,FIVE**
:SE,256,NEWFIL,AA,Ø ← *globals 1G through 4G are assigned values;*
:DP,1G,2G,3G,4G,5G *5G through 9G are unchanged*
256,NEWFIL,AA,Ø,FIVE

2. **:SE,4G** ← *global 1G is set to the value of 4G; in this case, 0*
:DP,1G
Ø

2-45. CA - CALCULATE GLOBALS

Individual G-type global parameters can be assigned values or nulled with the CA command. The values assigned can be the result of arithmetic or logical calculations.

Format

```
:CALCULATE,global#[,p1[,op1,p2[op2 ... ,opn,pn]]]
```

Parameters

global# Integer 1 through 9 identifying the global 1G through 9G to be set to the result of the calculation.

Globals -36P through -1P and 1P through 6P also may be set to the result of the calculation using the following entry form for *global#*:

n:P

where *n* is the P type global to be set in the range -36 through +6 (excluding 0). For example, to set global 6P to zero, enter:

:CA,6:P,Ø

or

:CA,6:P *(if value or operation is omitted, zero is assumed)*

p1 - pn Values used in calculations; if omitted, *global#* is nulled.

op1 - opn Operations performed on operands; may be:

- +** add two operands
- subtract one operand from another
- /** divide one operand by another
- *** multiply one operand by another
- OR** inclusive OR two operands
- XOR** exclusive OR two operands
- AND** AND two operands

FMGR Operator Commands

Evaluation proceeds from left to right until a null operation code is detected. Any other precedence is effected by multiple CA statements.

The type of the result depends on the type of the operands. If operand types differ in any one CA statement, the highest type value is used, where type 0 = null, type 1 = numeric, and type 3 = ASCII in ascending order from 0 through 3.

Except for divide and multiply, calculations are performed separately on each word of three-word ASCII globals. For divide and multiply, all three words of the first operand are divided or multiplied by word 1 of the second operand.

In its simplest form, CA is used to null an individual global parameter or to set an individual global to the value of p1.

Examples

- ```
:CA,6,FTN4
:DP,-16P
3
:CA,6
:DP,-16P
0
```

← set global 6G to ASCII value "FTN4"  
← display type of 6G is ASCII (3)  
← clear global 6G to null value  
← display type of 6G is null (0)
- ```
:CA,2,15  
:DP,-32P  
1  
:CA,7,2G  
:DP,-12P  
1  
:DP,2G,7G  
15,15  
:CA,1,2G,*,14,+,1  
:DP,1G  
211  
:CA,7,7G,-,1  
:DP,7G  
14
```

← set global 2G to integer value 15
← display type of 2G is numeric (1)
← set global 7G to current value of 2G
← display type of 7G assumes type of 2G
← display values of 2G, 7G
← set 1G to product of 2G and 14 plus 1
← display 1G
← decrement 7G by 1
← display 7G
- ```
:CA,1,7,OR,15
:CA,2,7,XOR,15
:CA,3,7,AND,15
:DP,1G,2G,3G
15,8,7
```

← inclusive OR 7 and 15 (octal 17), assign to 1G  
← exclusive OR same values, assign to 2G  
← AND these values, assign to 3G
- ```
:CA,-1:P,-2P
```

Set -1P to the value in -2P

↑
value?

2-46. IF - CONDITIONAL SKIP

The IF command compares two values (usually globals) and skips a specified number of commands depending on the result of the comparison.

Format

```
:IF,p1,operator,p2 [,skip]
```

Parameters

p1,p2 Values to be compared; one or both may be global parameters

operator Relative operator used to compare values of *p1* and *p2*; entered as one of the following two-character keywords:

Keyword	Operation
EQ	$p1 = p2$
NE	$p1 \neq p2$
LT	$p1 < p2$
GT	$p1 > p2$
GE	$p1 \geq p2$
LE	$p1 \leq p2$

skip Skip count; positive or negative integer specifying the number of commands to skip when relation between *p1* and *p2* is true; forward skip if positive, backward if negative; if omitted, one command is skipped; if relation not true, next sequential command is executed.

NOTE

IF will not be executed from an interactive device; it must be within a procedure file or a batch job.

The specified relation between *p1* and *p2* is examined and if it is true then commands are skipped. One command is skipped if *skip* is not specified. If *skip* is specified, then that number of commands are skipped. A skip of -1 causes the IF command to be repeated. To skip back to the preceding command, specify -2.

IF will not skip past the beginning or the end of the procedure file. Such an attempt will cause a skip to the beginning or end of file mark; no error message is issued.

When a negative skip is used, the file must be on a device that recognizes a backspace. For example, it is useless to attempt a negative skip on a paper tape reader. Jobs that are spooled (refer to Section IV) recognize backspace commands. The following relations obtain for mixed types:

```
null < numeric < ASCII
```

This corresponds to the type codes: null=0, numeric=1, ASCII=3.

FMGR Operator Commands

If *p1* and *p2* are both ASCII, the comparison is based on the ASCII collating sequence (refer to Appendix A).

Examples

1. Procedure file COMP tests if 1G is ASCII. If it is, a source program name is assumed, the program is moved to the LS area and 1G is set to 2 to compile from that area. If it is not, it is assumed to be the logical unit of an input device other than 2.

Procedure file COMP:

```
:IF,-36,NE,3,2 ← IF 1G not ASCII, skip two commands
:MS,1G
:CA,1,2 } ← move to LS area and set 1G to LU 2
:RU,FTN4,1G,99 ← compile program
::
```

To perform procedure COMP:

```
::COMP,FILENAME ← transfer to COMP with 1G a file name
OR
::COMP,5 ← transfer to COMP with 1G a logical unit number
```

2. Test if program has been loaded. If it has, 10G is set to program name and 1P is greater than 0, if not 1P is 0. If not loaded, send message:

Procedure file FORTLG:

```
:LG,1
:RU,ASMB,99 ← source input from default logical unit 5
:RU,LOADR,99
:IF,1P,GT,0 ← skip one command if true (default skip is 1)
:PA,7,***LOAD*** ← if 10G not a program name, execute PA
:RU,10G
:TR
```

Transfer to FORTLG from logical unit 7:

```
::FORTLG
: ← message sent here if 10G not loaded
```

3. Procedure IDLE sets up a loop that executes a set of instructions 1000 times:

Procedure IDLE:

```
:SV,2 ← set severity so commands are not echoed
:CA,1,0
:CA,1,1G,+,1 ← loop increments 1G until it equals 1000
:IF,1G,LE,1000,-2
```

To perform IDLE:

```
::IDLE
```

General Example Using Procedure File with Globals

This example illustrates all the commands discussed with procedure files. A procedure file MOVE is used to copy a number of files from one FMP cartridge to another without copying them all as is the case with the CO command (paragraph 2-61). Four message files are used to request input specifying the cartridge to be copied, the cartridge to which files are copied, a security code for the copied files, and the names of the files themselves. If desired, the copied file can be given a new name.

When you transfer to MOVE with the TR or RU command, you should enter the logical unit of the terminal at which you want to enter input as the first parameter to be passed to MOVE. If omitted, a message is displayed at the log device and you must re-enter the command with the input parameter.

The messages asking for input are displayed at the terminal you specify. The procedure then pauses so that you can return the information as parameters in the : version of the TR command.

The message files are:



File #MESS1

PRECEDE ANSWERS TO THE FOLLOWING QUESTIONS WITH A COLON AND A COMMA:
 :ANSWER1,ANSWER2,...

File #MESS2

ENTER THE 'CR' OF THE CARTRIDGE WITH THE FILES TO BE MOVED:

File #MESS3

ENTER DESTINATION CARTRIDGE'S CR, FILE'S NEW SC:

File #MESS4

NOW ENTER SOURCE AND DESTINATION FILE NAMES. IF NEW FILE IS TO HAVE THE SAME NAME AS OLD, ONLY ONE NAME IS REQUIRED. END LIST WITH /E.

The file MOVE:

```

:SV,1
:CA,4,1G ← save terminal LU in 4G
:IF,-36P,EQ,1,2 ← if not specified, display error message
:DP,YOUR LU WAS NOT GIVEN
: ← and return
:DU,#MESS1,4G ← send first message to terminal
:DP
:DU,#MESS2,4G
:PA,4G ← at pause, operator enters TR followed by information
:CA,6,1G ← CR of cartridge containing files is saved in 6G
:DU,#MESS3,4G
    
```


FMGR Operator Commands

```
:PA,4G ← enter cartridge and security code at pause
:CA,8,2G ← security (SC) in 8G
:CA,7,1G ← and cartridge (CR) in 7G
:DU,#MESS4,4G
:DP ← note use of DP to generate blank line
:SE,0,0,0 ← clear 1G, 2G, and 3G prior to entry of files
:PA,4G,FILES: ← enter files
:IF,1G,EQ,/E,5
:IF,1G,EQ,,-3
:IF,-32P,GT,1 ← test for new file name
:CA,2,1G
:ST,1G::6G,2G:8G:7G::-1 ← transfer file
:IF,A,EQ,A,-8 ← return for next file
:SV,0
::
```

To use this procedure:

```
:RU,MOVE
:SV,1
YOUR LU WAS NOT GIVEN
:RU,MOVE,1
PRECED E ANSWERS TO THE FOLLOWING QUESTIONS WITH A COLON AND A COMMA:
: ,ANSWER1,ANSWER2,...
```

ENTER THE 'CR' OF THE CARTRIDGE WITH THE FILES TO BE MOVED:

```
:PA,4G
::,2
ENTER DESTINATION CARTRIDGE'S CR, FILE'S NEW SC:
```

```
:PA,4G
::,13,JG
NOW ENTER SOURCE AND DESTINATION FILE NAMES. IF NEW FILE IS TO HAVE
THE SAME NAME AS OLD, ONLY ONE NAME IS REQUIRED.
END LIST WITH /E.
```

```
:PA,4G,FILES:
::,BEV,JOAN ← enter first file to be moved and rename it
:PA,4G,FILES:
::,MIKE ← enter next file and use same name for copy
:PA,4G,FILES:
::,/E ← no more files
::
:
```

2-47. BATCH JOB CONTROL

Whenever a set of activities is to be combined in one job, it can be delimited with the JO and EO commands. JO defines the beginning of a job and EO the end. Other commands are used to

set time limits to the job, to switch logical units for the duration of the job, to print messages on the job list device, or to terminate the job before the EO is reached.

With these commands, it is possible to define one or more jobs that are entered and controlled through devices other than the system console. For instance, a job can be punched on paper tape or on cards and completely controlled with little or no operator intervention. This type of operation is generally referred to as "batch processing". Using the FMGR job capabilities, the entire job stream consisting of commands, source language programs, and data can be read in from a peripheral device and output sent to whatever device is specified in the job stream.

When FMGR (not a copy of FMGR) encounters a :JO command, a batch environment is set up. FMGR, and not a copy of FMGR, is the only program that can do this. If the :JO command was inspoiled by program JOB, a batch environment with spooling is set up, as explained in Section IV "Using The Spool Monitor". If the :JO command was not inspoiled by program JOB, a batch environment without spooling is set up, as explained in the following paragraphs.

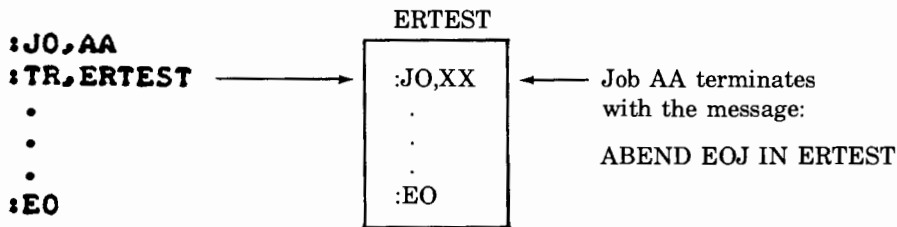
Commands are part of the job stream and can be saved as a procedure file (refer to paragraph 2-39) to which a TR command within the job transfers control. If not saved on a file, commands are expected from the input device specified in the *RU,FMGR command.

FMGR Batch Control

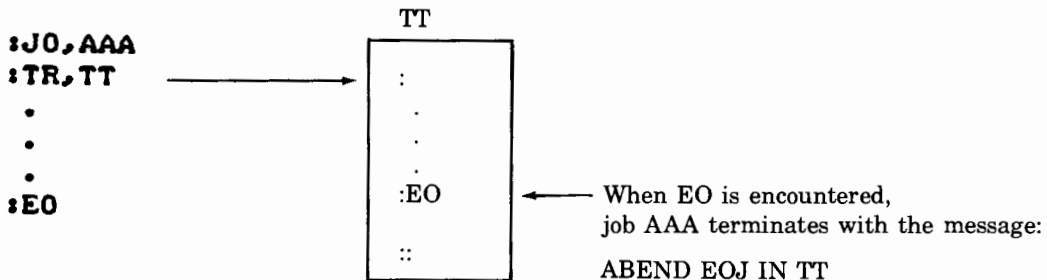
Within the boundaries of the JO and EO commands, the FMGR program operates in batch mode. JO and EO delimit an environment that is distinct from interactive FMGR operations. Any FMGR command can be used in a batch job. The LU command to switch logical units only can be used in the batch job environment. Previously set global parameter values or a severity code are set to FMGR default values by JO and then reset to these values when EO terminates the batch environment. LG tracks are also released by EO. This means that any logical unit switching, global manipulation, severity code settings within a job are operative for that job only.

Using Procedure Files in Job

Errors that might not occur in other environments may occur in a batch job that transfers to procedure files. For example, if a job transfers to a procedure file that contains another job, the first job is terminated as soon as it encounters the second job since jobs must not be nested.

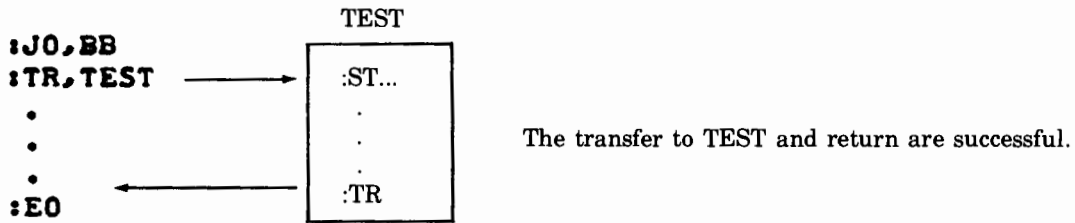


An error will also occur if the EO command is not at the same level as the JO command. For example:



FMGR Operator Commands

Jobs can, and often do, transfer control to procedure files. Such jobs work as long as the procedure file does not contain a JO or EO command.



Setting Severity Code

Within a job, the severity code should be set to a value greater than zero in order to inhibit command echo which is only useful in interactive mode. If the severity code is set to 4, the job will continue automatically, despite any error condition. If the severity code is set to 3, the job will continue, despite errors that require operator intervention, when the error is corrected. If set to 2, the job terminates whenever an error occurs requiring operator intervention. If set to 1, all errors are displayed but certain errors do not cause the job to terminate.

To illustrate, when an error such as FMGR -006 occurs that does not cause transfer to the log device, SV = 1 causes the code to be listed at the console and the job continues executing, for SV = 2, the job also continues but no error code is listed. On the other hand, if an error such as FMGR 020 occurs causing transfer of control to the log device, both SV = 1 and SV = 2 cause the error to be logged and the job to be terminated.

Severity code should be set at the beginning of each job with the SV command (paragraph 2-15). If more than one batch job is being processed, either 1 or 2 would be useful codes so that the job terminates in case of serious error. This allows FMGR to proceed with the next job. If an operator is monitoring the jobs, then severity of 3 is useful since it allows the job to continue after correction. A severity code of 4 allows the job to continue in any case.

Logical Unit Switching

Should you write a program using logical unit numbers that do not correspond to the logical units established at system generation, you can switch all logical unit references in your program to refer to the actual logical unit. This is done by the LU command and is effective only for the duration of a job as defined by JO and EO. Logical unit switching is also used to equate a logical unit in your program to a spool file for input or output spooling (see Section IV).

Batch Logical Unit Switch Table

FMGR uses a Logical Unit Switch Table to keep track of switched logical units. This table is only available in FMGR under batch job control, with or without spooling. The table is set up for the life of a batch or spooled job to equate logical units referenced in programs or commands to actual logical units or spool files. The table consists of pairs of logical units, one the logical unit referenced by calls or commands in the job, and the other an actual logical unit in the Device Reference Table (DRT) established at system generation. If spooling is used, the actual logical unit may be a spool logical unit.

Batch Control With Spooling

The Spool Monitor extends the FMGR batch capabilities by transferring job input and output to special disc files, the spool files. (Refer to Section IV for a full description of the Spool Monitor.) When spooling is used, the spool input program (JOB) reads each job into a disc file. As soon as at least one job has been read, the spool input program automatically schedules the FMGR program to execute the job stream. The Spool Monitor sends all spooled output from jobs to disc so that actual output does not interfere with other tasks.

2-48. JO - START OF JOB

The JO command defines the beginning of a job, gives the job a name, and sets up parameters for the job. JO should be used only with program FMGR, not a copy. When FMGR processes a :JO command that has not been inspoiled by program JOB a batch environment without spooling is set up. Refer to Section IV for a description of the :JO command in a batch environment with spooling.

Format

```
:JOB[name[:hr:min:sec]]
```

Parameters

<i>name</i>	6-character name; follows file name conventions (paragraph 2-8); identifies the job; if omitted, job has no name.
<i>:hr:min:sec</i>	CPU time limit for job in hours, minutes, seconds specified as one or two decimal digits; executing job is terminated when limit exceeded; if omitted, no limit to job time.

NOTE

The JO command has additional parameters used only for spooling; refer to Section IV, paragraph 4-7 for complete JO command format.

The JO command automatically performs a set of housekeeping functions for the job it initiates; it:

- terminates any previous job if EO not specified for that job
- assigns default log and logical list units for the duration of the job:
log default = logical unit 1 (system console)
list default = logical unit 6 (line printer)
- clears globals 1G through 9G and 1P through 5P to null values
- clears LG area
- resets Logical Unit Switch Table to generation values
- prints start of job message on list device (logical unit 6).

FMGR Operator Commands

Additional housekeeping functions are performed when FMGR processes a :JO command in a batch environment with spooling (see Section IV).

The time limit specifies central processor time, *not* elapsed time from beginning to end of the job.

All job input is assumed to be from the logical unit at which the JO command is entered.

When JO is entered, it prints a start of job message on the default list device:

JOB name ON AT hr:min:sec.ms ON day month year

2-49. EO - END OF JOB

The EO command indicates the end of a job; all commands, files, or data between JO and EO constitute the job. See Section IV for further use of EO with spooling.

Format

:EOJ

As JO performs housekeeping functions to initialize a job, EO performs functions that terminate a job; it:

- prints an end of job message on the list device
- resets the log and list logical units to the values 1 and 6 respectively
- resets severity code to zero
- releases any LG tracks
- clears globals 1G through 9G
- clears Logical Unit Switch Table

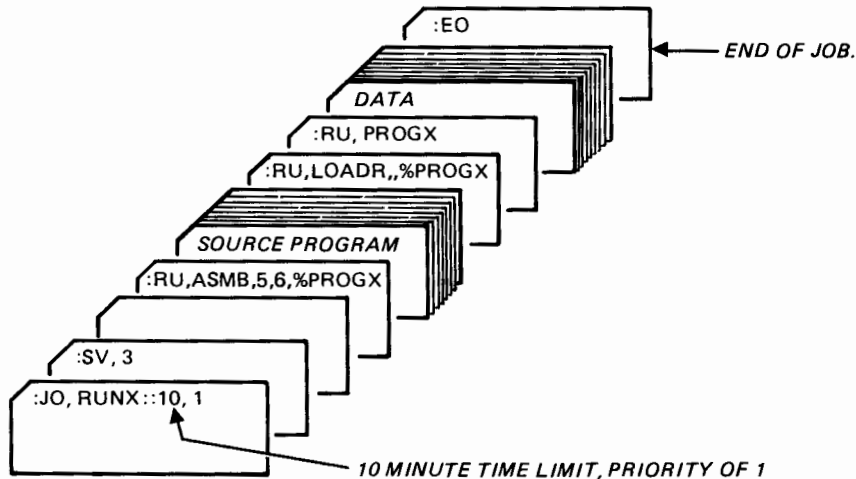
When spooling is used, EO performs additional housekeeping; refer to paragraph 4-8. When EO is entered, the following end of job message is printed on the list device:

JOB name OFF AT hr:min:sec.ms ON day month year

EXECUTION TIME: hr:min:sec.ms

Examples

1. In an RTE-IV Operating System, job RUNX assembles, loads, and executes program PROGX with all input including the job control commands from the card reader (logical unit 5): Load the following card deck:



Then run FMGR with input from LU5, the standard input device, in this case the card reader:

```
*RU, FMGR, 5
```

The above command causes FMGR to process the cards in logical unit 5. When it processes the :JO command it will set up a batch environment without spooling.

The EO command will also terminate FMGR if there is not further input from logical unit 5; if this job is followed by another job, FMGR will not terminate until the card reader is empty.

2. In an RTE-IV Operating System, two jobs J1 and J2 transfer control to procedure file FORTLG that compiles, loads, and runs a FORTRAN program using global 1G for the source and 10G for the loaded program:

Procedure file FORTLG:

```
:RU,FTN4,1G,6,%TEMP
:RU,LOADR,,%TEMP
:RU,10G
:TR
```

Jobs that execute source programs SRCE1 and SRCE2:

```
:JD,J1
:SV,3
:TR,FORTLG,&SRCE1
:JD,J2 ← JO performs all EO functions
:SV,3
:TR,FORTLG,&SRCE2
:EO
: ← job, but not FMGR, terminated
```

FMGR Operator Commands

3. Use AN command (paragraph 2-17) to annotate a job on the list device:

```
:JO,MYJOB::15
:AN, **MYJOB TIMES OUT AFTER 15 MINUTES CPU TIME**
.
.
.
:EO
```

The line printer will then print at the start of the job:

```
JOB MYJOB ON AT 12:10:36.6 ON 10 JUN 1975 ← or whatever time
:AN, **MYJOB TIMES OUT AFTER 15 MINUTES CPU TIME** and date
```

2-50. TL - SET RUN TIME LIMIT

TL sets a time limit for the execution of programs within a job.

Format

:TL:hr:min:sec	
Parameters	
:hr:min:sec	Time limit for execution of any programs run with RU command subsequent to TL command; specified as hours, minutes, seconds each two decimal digits: if omitted, job time limit is used.

The time specified must be less than any time limit specified for the job in the JO command. TL sets this time limit for each program executed between TL and EO. If the remaining time for the job is less than the limit in TL, the remaining job time is used to limit program execution time. If any program exceeds this limit, it is terminated and the job is aborted; RTE issues an error message.

The specified TL limit applies only to programs executed by a RU command. Actual program run time is subtracted from the remaining job time. For example, suppose the job time limit is 15 minutes and TL sets a run time limit of 5 minutes for program XX. If XX starts executing after the job has run for 12 minutes, then XX is given a run time limit of only 3 minutes, the remaining job time. If XX times out in that time, zero minutes remain of the job time. If XX starts executing after 10 minutes of job time and finishes in three minutes, then the remaining job time is two minutes (15-(10+3)).

TL, like the job time limit, limits central processor time and not the elapsed time for the program that could include waiting for an I/O device or for program swapping, or be processor time devoted to unrelated real-time tasks.

Example

Procedure file FORTLG:

```
:RU,FTN4,1G,6,%TEMP
:RU,LOADR,,%TEMP
:TL::5 ←————— time limit for execution of 10G is 5 minutes
:RU,10G
:TR
```

Job that transfers to FORTLG:

```
:JD,RUN:::10 ←————— time limit for entire job is 10 minutes
:SV,1
:TR,FORTLG,&PROGA
:EO
```

2-51. LU - LOGICAL UNIT SWITCH

Within a batch job, without spooling, the LU command can be used to exchange a referenced logical unit for an actual system defined logical unit.

For a description of the LU command in batch with spooling, refer to Section IV.

Format

```
:LU,lu1,lu2
```

Parameters

<i>lu1</i>	Logical unit number referenced from a call or command within a batch job.
<i>lu2</i>	Logical unit number of actual unit for batch logical unit switch; if 0, resets <i>lu1</i> to definition prior to switch.

NOTE

The LU command has additional parameters used only for spooling; refer to Section IV, paragraph 4-9 for the complete LU command format.

The *lu1* parameter specifies a logical unit number referenced in a program or command which is to be switched through the Logical Unit Switch Table (paragraph 2-47) to an actual logical unit associated with a device at system generation. This first logical unit must not be 0, 2, or 3; nor may it be a unit associated with any DVR30, 31, 32, or 33 type device (refer to Appendix C for device types).

FMGR Operator Commands

The *lu2* parameter must be a positive logical unit number. If *lu2* is zero, the first logical unit (*lu1*) is reset to its system defined logical unit and the switch table entry is cleared.

NOTE

If the LU command is included in a batch job, it only works if the job is run by program FMGR, *not* a copy of FMGR such as FMG07.

Examples

1. Suppose your program writes its list output to logical unit 10, but the desired list device is logical unit 6:

```
*R11,FMGR ← must be FMGR, not a copy
:JOB,LIST
:LU,10,6 ← sets LU 10 equivalent to LU 6 in Logical Unit Switch Table
:R11,XX
:
WRITE(10,....) ← XX contains a statement to write to LU 10
:
:EO ← clears Logical Unit Switch Table
```

2. Program YY contains references to logical unit 5 as the standard assignment for magnetic tape when in actual fact, the system assignment for magnetic tape is 8. In the same job, program ZZ uses logical unit 8 for magnetic tape:

```
:JO,JOAN
:LU,8,5
:RU,YY ← YY uses LU 5 for magnetic tape
:
:
:LU,8,0 ← set LU 8 to system definition
:RU,ZZ ← program ZZ uses LU 8 for magnetic tape (actual logical unit)
:
:
:EO
```

2-52. AB - ABORT JOB

AB terminates a job at any point within the job and performs the end-of-job housekeeping normally performed by EO.

Format

```
:AB
```

Whenever there is reason to terminate a job before the normal end of job, the AB command can be specified. Any subsequent commands will be read but not processed until a JO or EO is encountered, or the card hopper is empty.

AB causes the message ABEND OPERATOR to be logged.

When an error in a job causes control to transfer to the log device and the severity code is set to 3, the job will not terminate. The operator at the log device can then correct the error and return control to the job with TR, or he may abort the job with AB.

Examples

1. In an RTE-IV Operating System, procedure file FTNLG

```

:RU,FTN4,1G,6,%TEMP
:LG,1
:RU,LOADR,,%TEMP
:IF,10G,GT,0,2
:AN,**NO LOAD**
:AB ←──────────────────────────────── terminate if program not loaded
:RU,10G                               continue if program loaded
:TR
    
```

Transfer to FTNLG from job AA

```

:JO,AA
:SV,1
:TR,FTNLG,&SRCE
:EO ←──────────────────────────────── normal job termination
    
```

2. Job XX contains an error that causes transfer to the log device but does not terminate the job; if the error cannot be corrected, the job may be aborted with AB at the log device:

```

:JO,XX
:SV,3
:ST,AA,TEST ←────────── TEST is an existing file
                error causes transfer to log device ───────────→ :ST,AA,TEST
                use AB to terminate job ─────────────────────────→ FMGR -006
                :AB
:EO
    
```

2-53. FMP CARTRIDGE MANIPULATION

In the File Management Package, disc manipulation is performed in terms of FMP cartridges. An FMP cartridge is a logical area on a disc consisting of an area for file storage and a directory of the files stored. Each cartridge is a physically contiguous area on the disc identified in FMP calls and commands by a unique cartridge reference number. It also has an ASCII identifier and must be associated with a unique logical unit number.

Since many logical units can refer to the same disc subchannel, many cartridges can exist on a given subchannel *provided they do not overlap*. Generally, one cartridge per subchannel is used. Where one subchannel per disc platter is specified at system generation (required for the 7900 disc), the disc and the FMP cartridge are generally the same. On discs where subchannels can cross disc platter boundaries, the cartridge definition generally corresponds to the subchannel definition.

Every cartridge used by the File Management Package, including the system and auxiliary cartridges, must be mounted and initialized before they can be used. The system disc (logical unit 2) is initialized when RTE is loaded from disc for the first time after system generation (refer to FMGR initialization, Section VII). If you use an auxiliary system disc (logical unit 3), it too is initialized at this time. Note that unless you have a specific need for the auxiliary system disc, it is much simpler to avoid using it altogether and use only the system and peripheral discs.

Peripheral discs on which user files are stored must be mounted and initialized with FMGR commands described in this part of the manual. When a cartridge is physically mounted, FMGR must be so informed with a mount cartridge command. This command establishes an entry for the cartridge in the cartridge directory on the system disc and specifies the last track used by the cartridge for its file directory. To be fully defined for use, a cartridge must also be initialized. Initialization establishes the first track used by the cartridge and also specifies its logical unit number, cartridge reference number, ASCII identifier, and the number of file directory tracks to be allocated to the cartridge.

Once a cartridge has been initialized, it need not be re-initialized unless the cartridge definition needs to be changed. Re-initialization may necessitate purging all files on the cartridge. It must be mounted with a mount cartridge command each time it is physically placed in the disc drive. Whenever a cartridge is physically removed, FMGR must be informed with a dismount cartridge command in order to clear the cartridge entry from the cartridge directory.

When there is more than one cartridge on the disc platter, only the cartridge on which your files are stored need be mounted when the platter is placed in the disc drive. When removed, any currently mounted cartridges on that platter must be dismounted.

CAUTION

To avoid disruption of data stored on disc, the disc controller and removable disc platters must not be exposed to areas with a strong magnetic field.

Within each FMP cartridge, all file names must be unique. File names can be duplicated if they appear on different cartridges since the file can be uniquely identified by the cartridge reference number and the file name.

FMP peripheral cartridges can be protected from alteration by user programs. This is accomplished during system generation (see paragraph 7-3). After the system is booted-up and running, user programs can read information from protected cartridges, but cannot alter files residing on them, except through the FMP commands and subroutines.

2-54. LOCKING A CARTRIDGE

Under circumstances requiring modification of the cartridge's file directory, FMGR will lock a cartridge to insure that it cannot be accessed by other users.

FMGR locks the cartridge whenever:

- the cartridge is mounted but has not been initialized (MC command)
- the cartridge is being initialized (IN command)
- a type 0 file is being created on the cartridge (CR command)
- a type 0 file is being purged from the cartridge (PU command)
- the cartridge is being packed (PK command)
- the cartridge is being dismounted (DC command)

A locked cartridge appears to other users as if it were not mounted and attempts to access the cartridge result in error code -013. A cartridge can be locked only when all files on the cartridge are closed.

On the other hand, if you attempt a function that requires the cartridge to be locked, but it is impossible for FMGR to lock the cartridge, a FMGR -008 error is issued. Suppose you attempt to pack a cartridge from the system console, but a user program has opened a file on that cartridge, error code -008 is sent to the log device.

2-55. MC - MOUNT CARTRIDGE

A cartridge must be mounted with the MC command before it is included as an entry in the FMP cartridge directory making it available to the file management system.

Format

:MC, <i>lu</i> [, <i>last track</i>]	
Parameters	
<i>lu</i>	Logical unit number of the cartridge; may be either positive or negative.
<i>last track</i>	Last track on cartridge available to the FMP; if omitted, the last track defined at system generation for the disc; last track must be specified for a fixed head disc (driver type DVR30) that is not logical unit 2 or 3.

Logical Unit

The *lu* parameter may be positive or negative, but in either case it is a logical unit and must refer to a disc. Only one FMP cartridge can be assigned to each logical unit, so if a cartridge is already mounted on the specified logical unit an FMGR error 012 is issued and the command is not executed.

Last Track

The last track on the FMP cartridge contains the cartridge entry in the file directory established for the cartridge by the IN command (paragraph 2-56). The *last track* parameter provides the link to this information. If the cartridge is on LU 2 or LU 3 (system or auxiliary) or is on a moving head disc, *last track* may be omitted. In these cases, the system defined last track is assumed.

When a cartridge is dismounted, the system reports the last track and this should be noted. When an initialized cartridge is mounted, FMGR checks the file directory to see that the cartridge reference number (CR) is unique. If not, the error message FMGR 012 is issued and MC is not executed. You should dismount the cartridge with the duplicate CR number and then request MC again.

Note that a cartridge cannot be used until it is both mounted and initialized. In general, a cartridge on a removable platter is initialized once, but mounted and dismounted often.

Cartridge Directory

When MC is executed, an entry is established for the cartridge at the bottom of the FMP cartridge directory on the system disc (refer to Appendix C for an illustration of the directory format).

Examples

1. Mount an FMP cartridge on logical unit 14:

```
:MC, 14           last FMP track defaults to 202, the last track
                   on the 7900 moving head disc
```

2. Mount three cartridges on logical units 15, 16, and 17:

```
:MC, 15, 256
:MC, -16, 512
:MC, 17, 768      } all three FMGR cartridges could be on the same disc
```

2-56. IN - INITIALIZE CARTRIDGE

Every FMP cartridge must be initialized once with the IN command before it can be used. IN establishes an entry for the cartridge in the file directory maintained for each cartridge. The command can also be used to change this cartridge entry or to assign a new master security code to the cartridge.

Formats

1. :IN,[*master security*],*cartridge*,*label*,*id* [,*first track* [, #*dir tracks* [,*sec/track* [,*bad tracks*]]]]

2. :IN,*master security*--*new security*

The first format initializes a cartridge or changes the description of an initialized cartridge; the second format changes the master security code of the cartridge.

Parameters

<i>master security</i>	Security code that governs access to the FMP cartridge directory and to all file security codes; must be 2 ASCII characters; if omitted, directory and file security codes can be accessed with any code or none.
<i>cartridge</i>	Cartridge identifier; if positive specifies cartridge reference number, if negative, the logical unit number; must be negative the first time cartridge is initialized.
<i>label</i>	Cartridge reference number (CR) that identifies the cartridge; must be positive integer from 1 through 32767 or two ASCII characters.
<i>id</i>	ASCII identifier assigned to cartridge; up to 6 ASCII characters specified exactly like an FMP file name (see <i>namr</i> description, paragraph 2-8).
<i>first track</i>	First FMP track on cartridge; a positive integer; for the system disc (LU 2), it must be at least 8 greater than the last system track; if omitted, track 0 is assumed.
# <i>dir tracks</i>	Number of directory tracks used by file directory on cartridge; positive integer from 1 through 48 and less than one ninth of the total tracks on the cartridge; if omitted, one track is assumed.
<i>sec/track</i>	Number of 64-word sectors per track; if cartridge is on same channel as (that is, the same hardware select code) LU 2 or LU 3, this parameter is ignored; on any other channel, it must be specified.
<i>bad tracks</i>	Up to six track numbers, separated by commas, specifying bad tracks on the cartridge; if omitted, all tracks are assumed to be usable.

NOTE: For a system with mixed discs, the number of sectors per track must be specified. It is not an optional parameter. For example, in a system with a 7905 system disc and a 7900 disc, you must specify 96 sectors per track to initialize the 7900.

FMGR Operator Commands

Example

```
:IN,SC,-14,9700,NEWSYS,1 ← raise first track from 0 to 1
FMGR 060
:YES ← enter YES to purge all files
:IN,SC,-14,6500,CLEAR ← re-initialize cartridge and retrieve track 0
:PK (first track is 0 by default)
```

Bad Tracks

Bad track information is returned during RTE generation or, if discovered by the File Management Package, is returned as a -001 error code and reported on the system console. When you know a track is bad, you must enter the track number as a parameter in the IN command. This information is kept in the cartridge directory and FMP compensates for bad tracks when it assigns tracks during file creation or packing as follows: The first track of a new file is increased until the file contains no bad tracks. If a created file is to use the rest of the disc, it is allocated an area above the highest numbered bad track. During packing, if a file is found to include a declared bad track, the file is purged.

Note that in order to declare bad tracks, you must know the master security code.

Changing the Master Security Code

Anyone knowing the master security code has access to all the file security codes on the cartridge. For this reason, it is never printed and if you know the code, you must remember it. There is no other way to retrieve it.

If the current master security code is zero (default if omitted), you must still enter some code, any code will do, in order to assign a new master security code. If the master code is other than zero, you must enter the exact code in order to change it.

The new security code can be any two ASCII characters except:

colon (:), comma (,), a leading blank.

Non-printing characters are acceptable; in fact, such characters provide greater security since they are never printed or displayed.

To remove an existing master security code, the new security code can be set to two blanks. The blanks need not be specified; they are supplied by the parameter parsing routine.

Examples

1. To assign a code where none existed:

```
:IN,AA--SC
  ↑ ↑ ↑
  | | |
  | | | ← new master security code protecting all FMP cartridges
  | | | ← two minus signs separate old code from new
  | | | ← any two characters
```

- To remove an existing master security code:

:IN,SC-- *blanks may be typed or omitted after the minus signs*

- A code may be changed to the ASCII equivalent of two integers:

:IN,AG--12

When entered in subsequent commands as a separate parameter, the ASCII 12 will appear to FMGR as an integer. Therefore, to correctly specify the new master security code in later commands, it must be entered as the octal equivalent of the ASCII characters "12", that is, as 30462B.

Define a Three-Cartridge Disc Platter

Assume a subchannel is associated with three logical unit numbers 10, 11, and 12 at system generation. If this subchannel is on a 7900 disc, then it is equivalent to one disc platter. Using the MC and IN commands, Figure 2-4 defines three cartridges for this platter, one for each of the three logical units.

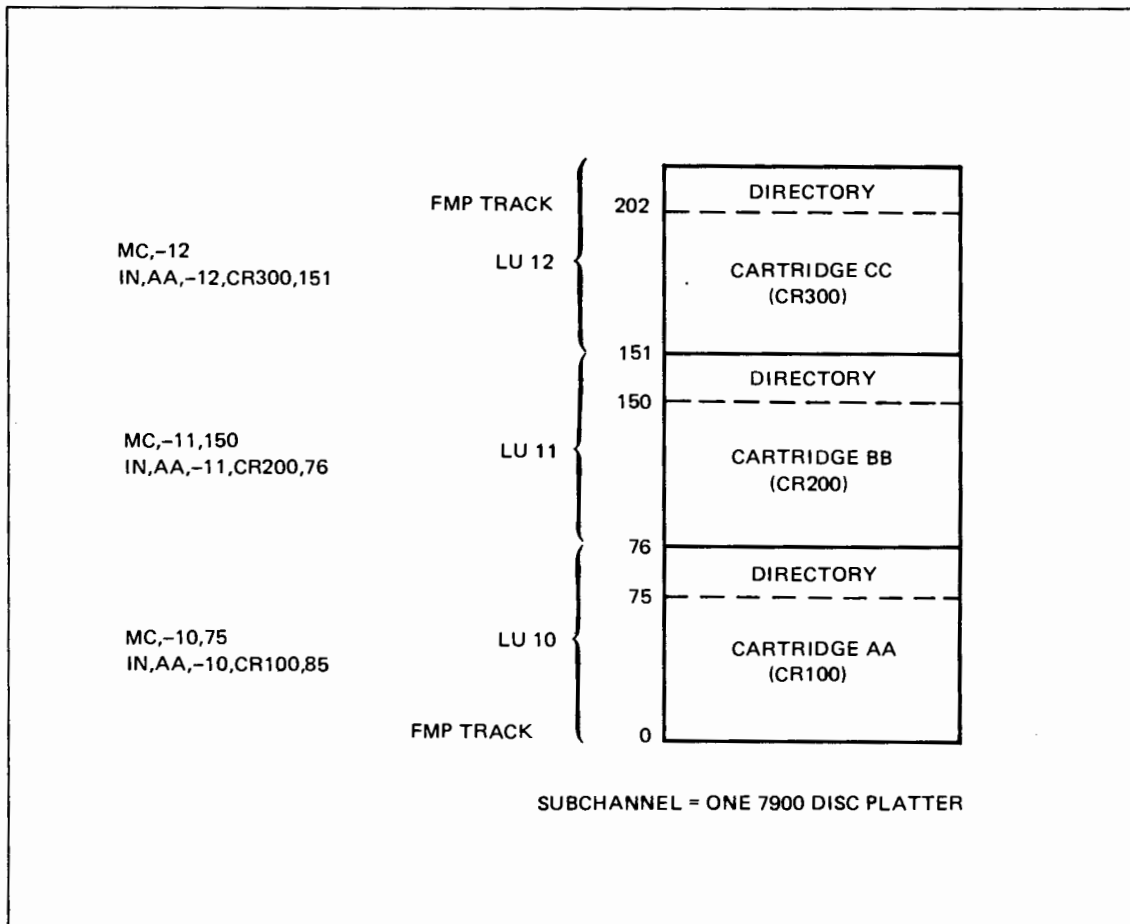


Figure 2-4. Disc Platter With Three Cartridges

2-57. DC - DISMOUNT CARTRIDGE

Before a mounted cartridge is physically removed from the disc drive, the DC command should be entered to logically remove the cartridge. DC makes the cartridge unavailable to FMP by removing its entry from the cartridge directory.

Format

:DC, <i>cartridge</i>	
Parameter	
<i>cartridge</i>	Cartridge identifier; positive cartridge reference number assigned to cartridge or negative logical unit number associated with cartridge

The cartridge designated by *cartridge* is locked and then its entry is removed from the cartridge directory. The last FMP track is reported on the log device. This track number should be written on the removed cartridge to facilitate the future restoration of the cartridge with the MC command.

Cartridge Directory Manipulation

FMGR does not allow the system or auxiliary cartridges to be removed. They must be mounted in order to properly configure the RTE track assignment table when the system is re-started from disc (booted). The DC command may, nevertheless, be issued for these cartridges. The cartridge is locked and removed from the cartridge directory list, but it is then immediately remounted at the bottom of the list. Because of this feature, the DC command can be used to alter the order of cartridge entries in the cartridge directory.

Whenever a search is made for a file and a particular cartridge is not specified, the search begins with the cartridge at the top of the directory list. For this reason, there may be an advantage to moving a particular cartridge to the head of the directory and this may be accomplished only by moving the system and auxiliary cartridges to the bottom.

Examples

- To illustrate the effect of the DC command on LU 2, first list the cartridge directory (use CL command described in paragraph 2-59) and then dismount LU 2. The effect will be to move the cartridge to the bottom of the list:

```

:CL
  LU  LAST TRACK  CR  LOCK
  02   0202      00002
  13   0202      00013

:DC,2
:CL
  LU  LAST TRACK  CR  LOCK
  13   0202      00013
  02   0202      00002

```

- To return cartridge 13 to the bottom of the list, dismount it and then remount it (it need not be physically removed and replaced) with the DC and MC commands:

```

:DC,13
LAST TRACK 0202
:MC,13,202
:CL
  LU  LAST TRACK  CR  LOCK
  02   0202      00002
  13   0202      00013
    
```



When a file is created, it can be placed on the first cartridge in the cartridge directory that has enough room. To do this, simply omit the *cartridge* subparameter in the *namr* describing the file. The system automatically places the file on the first cartridge it finds on which the file fits.

Changing Auxiliary Cartridges

Since the auxiliary cartridge (LU 3) cannot be dismounted with the DC command, another technique must be used if you want to replace one auxiliary cartridge with another. All cartridges to be mounted on logical unit 3 must be initialized to the same first track, preferably track 0 since this prevents the loader or the system from placing a program in this area.

To change auxiliary cartridges, use the DC command as follows:

```

:DC,-3 ←————— this insures that all files are closed
    
```

Remove the cartridge from the drive and insert the replacement.

```

:DC,-3 ←————— places new cartridge in disc directory
    
```

Note that MC is not used to mount the cartridge. This is because DC mounts the cartridge as part of its procedure when the logical unit is 2 or 3.

Be sure that the new auxiliary cartridge has been initialized to the same track as the old cartridge. If it has not been initialized, FMGR will lock it and a subsequent attempt to initialize the cartridge results in FMGR 059 error message. The error is caused because the directory tracks are already assigned to D.RTR. You must, therefore, release the D.RTR tracks and then re-assign them by scheduling FMGR. After assigning the D.RTR tracks, FMGR terminates and you must schedule it again in order to enter FMGR commands. This special case, where FMGR terminates immediately, occurs only when the D.RTR tracks are unassigned.

Example

```

:EX

SEND FMGR

*RT,D.RTR ←————— release the D.RTR tracks

*RU,FMGR ←————— scheduling FMGR assigns D.RTR tracks on LU 2; FMGR terminates

*RU,FMGR ←————— re-schedule FMGR

:IN,SC,-3,3,AUX ←————— initialize new auxiliary disc on LU 3
    
```

2-58. CL - CARTRIDGE LIST

A list can be requested of all cartridges in the cartridge directory with the command CL.

Format

```

:CL
    
```

The cartridge list is issued to the list device, normally LU 6, the line printer. This device may be changed with LL (see paragraph 2-13). The list contains the following categories:

- LU Logical unit number of the cartridge
- LAST TRACK Last track assigned to the FMP on that cartridge
- CR Cartridge reference number
- LOCK Name of program locking the cartridge; blank if not locked

Example

```

:CL ←———— request from input device

    LU        LAST TRACK        CR        LOCK        }
    02        0175            00002               } output to list device
    13        0202            00104               }
    14        0202            06500               }
    
```

2-59. DL - DIRECTORY LIST

Each mounted cartridge has on its last tracks a directory for the cartridge. The directory contains an entry describing the cartridge followed by an entry describing each file on the cartridge. A listing of this directory can be requested with the DL command.

Format

```
:DL,namr [,master security]
    or
:DL [,cartridge [,master security]]
```

Parameters

<i>namr</i>	namr parameter. See paragraph 2-8 for format definition. See below for conditions for using this version. The long list format (Table 2-5) is supplied.
<i>cartridge</i>	Cartridge identifier; positive for cartridge reference number, negative for logical unit number; if omitted or zero, the directories of all mounted cartridges are listed.
<i>master security</i>	Code assigned to the system at initialization; if correctly specified, directory list includes file security code and track and sector address for each file (long list).

The directory list is provided in two formats: a short list and a long list (see figure 2-5). Both lists have the same header information describing the cartridge itself; they differ in the file information provided. The long list includes a file security code for each file and the track and sector address for the file.

:DL,namr format

The purpose of this DL command format is to allow you more options when specifying which directory entries you want listed. When this format is used, the following conditions must be met before a given file entry will be listed:

- a. The file name must match the name portion of < namr > except that the minus sign “-” if used in < namr > “matches” any character.
- b. Zero as a subparameter matches any actual subparameter, however, if a non-zero subparameter is used, it must match the files actual parameter.
- c. The standard security code match is used, i.e.,

- n matches	n and - n
+ n matches	n only

Examples:

:DL,A lists all files whose name is A.
:DL,A---- lists all files whose name starts with A.
:DL,--A--A:-- 5:2 lists all files whose 3rd and 6th name characters are A and whose security code is -5 or +5 and which are on CR2.
:DL,-----:1 lists all files of length 1.
:DL,-----:16 lists all files with record length 16.

2-60. PK - PACK CARTRIDGE

The PK command is used to recover the tracks assigned to purged files and close any gaps between files.

Format

:PK [,*cartridge*]

Parameters

cartridge Cartridge identifier; a positive reference number or, if negative, the logical unit number of the FMP cartridge to be packed; if omitted, all mounted cartridges are packed.

When PK executes, it moves files into empty spaces left from purging, if possible, and updates the file addresses in the file directory. When all files are packed, it then packs the directory removing any entries for purged files.

PK will purge all files that contain bad tracks reported by the IN command (paragraph 2-56). If you do not want the file with bad tracks purged, you must save it on another cartridge using the ST command (paragraph 2-22).

Example

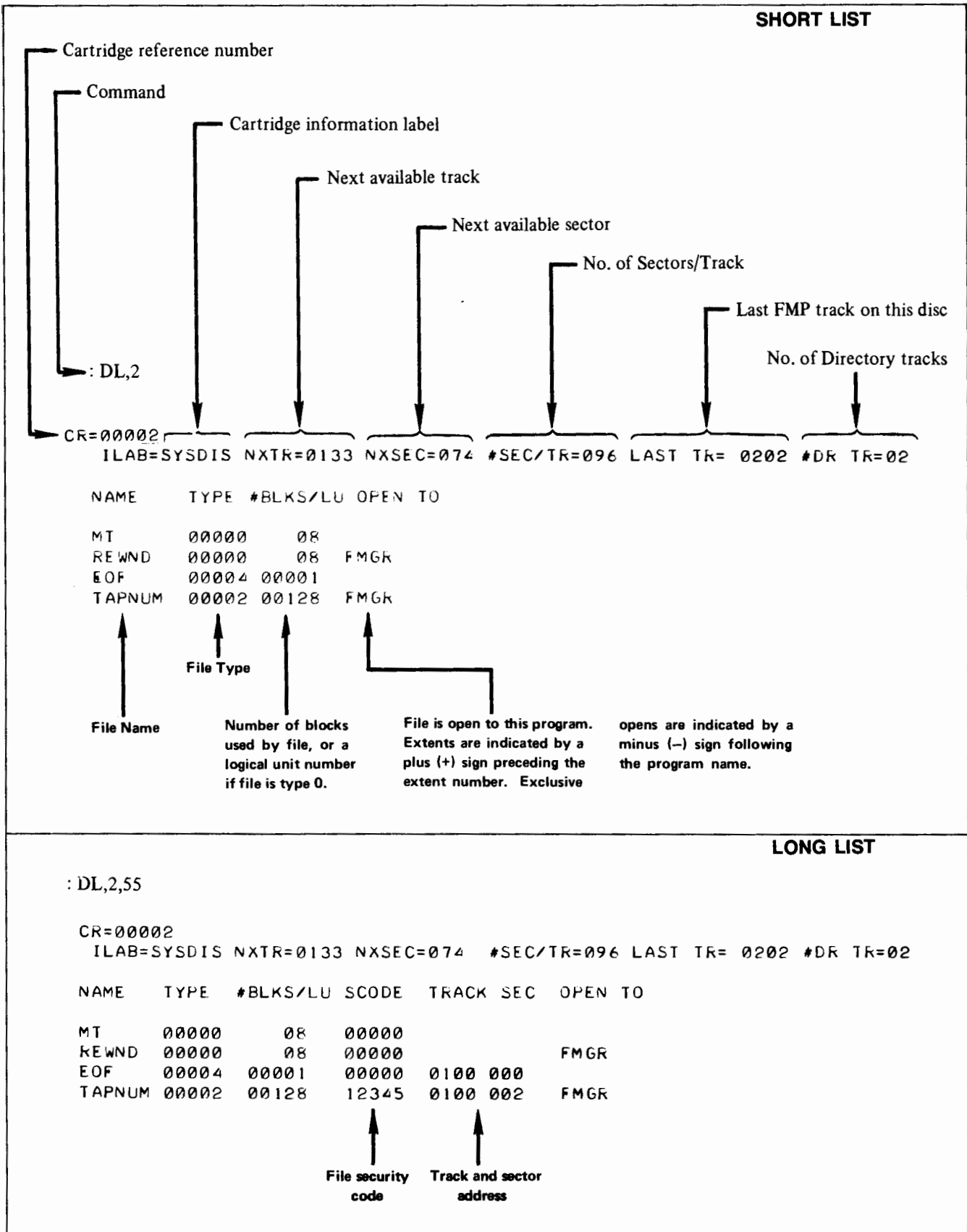


Figure 2-5. Two Types of Directory List

Packing Considerations

Since the PK command locks the cartridge(s) being packed, you must be sure that there are no open files on the cartridge(s) before requesting PK. If files are open, a FMP -008 error message is issued followed by the cartridge reference number if *cartridge* was specified or the logical unit if not. All cartridges except the specified logical unit(s) will be packed. If the cartridge being packed contains any programs restored by RP (paragraph 2-34), a FMGR 011 error is issued followed by a list of all the programs. Programs must be terminated with an OF, *program*, 8, command before packing can take place.

NOTE

To pack a cartridge containing spool files, you must first shut down the spool system with the ↑SD command (see Section V).

If the system fails during execution of PK, it is possible to lose, at most, one file. In order to determine which file, if any, has been lost, you need a copy of the file directory listing before PK was entered. This list should be the long list requested with the master security code so that it shows length, first track and sector addresses for each file. When power is returned, get another such list and compare the two lists. Look for the first file with an old disc address preceded by a file with a new address. Since the directory is updated after each file is successfully moved, this unchanged entry may indicate that the file has been lost. To illustrate, suppose the directory list before packing shows:

NAME	TYPE	#BLKS/LU	SCODE	TRACK	SEC	OPEN TO
A	00003	00001		0100	000	
B	00004	00003		0100	002	
C	00004	00002		0100	010	
D	00003	00003		0100	014	

← 2 sector gap follows B
too small for C

If the directory after packing shows C in sector 8 and D in sector 12, no files have been lost; PK has completed its operation. If, however, either file C or D is listed in the same sector (10 or 14 respectively), there is a good chance that it was being moved when the system failed and is now lost. If the second directory shows duplicate entries this simply means the directory was being re-written when the failure occurred and no files were lost. This can be corrected by storing the affected files in new files, purging the affected files, and changing the names of the new files back to their original names.

Examples

```

:DL,100,SC ← get directory listing before packing
:PK,100 ← then pack the cartridge
:DL,100,SC ← and get new listing
    
```

The new listing will show the new next available track (NXTR in heading) and new track sector addresses for any files that were moved. Figure 2-6 shows how packing might affect a cartridge.

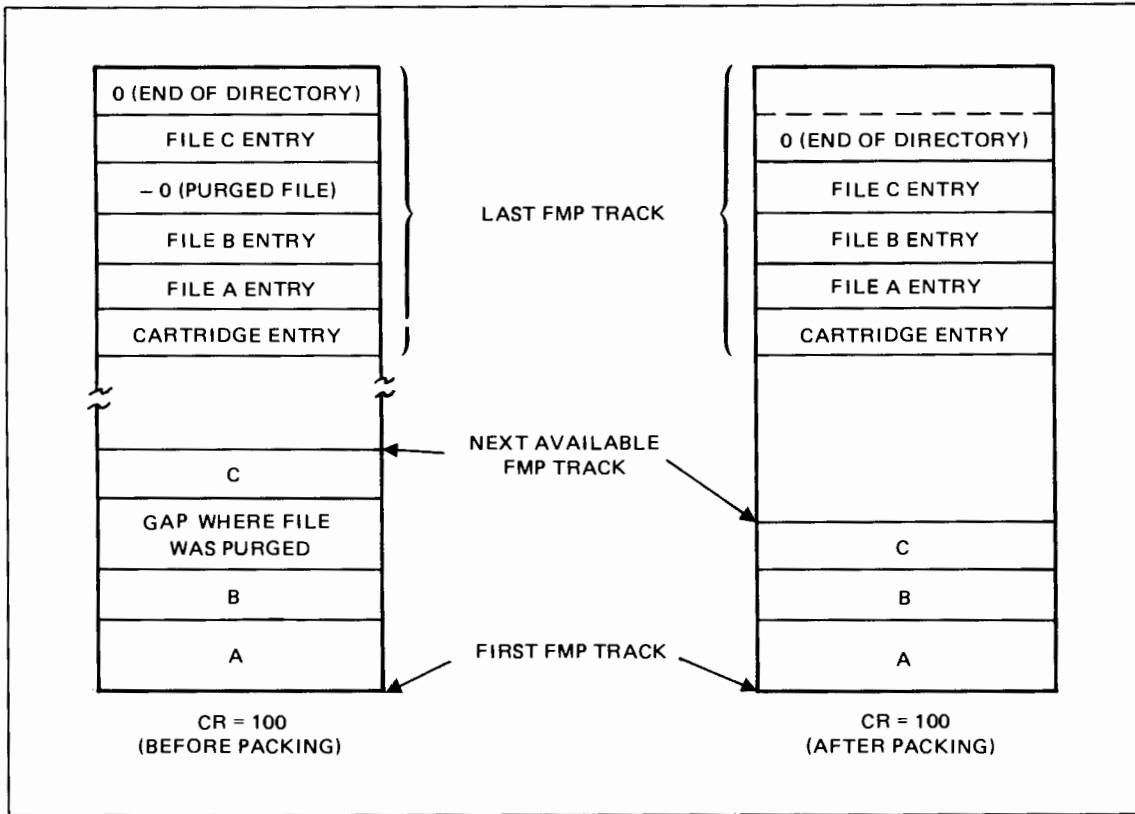


Figure 2-6. Packing a Cartridge

2-61. CO - COPY ONE CARTRIDGE TO ANOTHER

All files currently on a mounted cartridge can be copied to another mounted cartridge with the CO command.

Format

```
:COPY,cartridge1,cartridge2
```

Parameters

cartridge1 Cartridge reference number of mounted cartridge containing files to be copied; if negative, identifies cartridge logical unit number.

cartridge2 Cartridge reference number of mounted cartridge to which files are to be transferred; if negative, identifies cartridge logical unit number.

NOTE

Files are transferred record by record; records longer than 128 words are truncated.

All files transferred should have unique names. If a file on *cartridge2* has the same name as a file being transferred from *cartridge1*, the file is not transferred and an informative message is sent to the log device. As each file is copied, its name is displayed on the log device.

The files being copied are not affected by the copy. If files already exist on the cartridge to which files are being copied (*cartridge2*), these files are not affected; the copied files follow the existing files. Entries for the copied files are added to the end of the file directory on *cartridge2*. If there are any entries for purged files in this directory, entries for the copied files may be interspersed with entries for existing files. To know where the new file entries are placed, request a directory list with the DL command.

Example

Assume files A, B, C, and D on *cartridge1* are to be copied to *cartridge2*, and a file C already exists on *cartridge2*:

```

:CO,LA,LB ← where LA is cartridge1 and LB is cartridge2
A
B ← system prints file names as they are copied
C
FMGR -002 ← message indicating C is a duplicate name; it is not copied
D
    
```

2-62. MISCELLANEOUS COMMANDS

Using the commands described in the following paragraphs, you can execute RTE system commands without exiting FMGR and you can denote one or more lines of your own comments in the command entry list.

2-63. SY — EXECUTE RTE SYSTEM COMMAND

RTE system commands can be executed from FMGR by using the SY command as a prefix to a system command.

Format

```

:SYcommand Privileged Command (see paragraph 2-7)

where: command is the desired system command mnemonic code. No delimiter is
        permitted between SY and command.
    
```

FMGR strips off the SY prefix and passes the remaining characters to the system for execution. Any messages resulting from execution of the system command are passed back through FMGR to the log device, but do not force a transfer to the log device.

If you do not specify a logical unit number in an SY command request for the system command RU or ON, the SY command will supply the log device as the default logical unit.

Examples

:SYUP,6 ←————— Make logical unit 6 available to the system.
 :SYTI ←————— Request the time and date from the system.

2-64. ** — COMMENTS

You can include lines of comments within your command entry list to explain the command flow.

Format

<pre> :**comment line or, :*,comment line or, :* comment line </pre>	<i>Privileged Command (see paragraph 2-7)</i>
--	---

where: *comment line* is any string of alphanumeric characters. These characters must be separated from the first asterisk (*) by another asterisk, a comma, or a blank.

The lines denoted as comments do not affect command execution. Globals may be specified; however, the constructed line (with replaced global values) must not exceed line limitations (see paragraph 2-7).

Example for RTE-II/III Systems

```

:**ALLOCATE 3 TRACKS TO LG AREA
:LG,3
:*,ASSEMBLE SOURCE PROGRAM IN LS; RELOCATABLE TO LG
:RU,ASMB,2,99
:* SAVE RELOCATABLE AS FILE
:SA,LG,BFILE
:* RETURN LG TRACKS TO SYSTEM
:LG

```


SECTION III

FILE MANAGEMENT THROUGH FMP PROGRAM CALLS



INDEX TO FMP CALLS

Optional parameters are underlined

NAME	FORTTRAN CALL	FUNCTION	PAGE
APOSN	CALL APOSN(<u>IDCB</u> , <u>IERR</u> , <u>I</u> REC, <u>IRB</u> , <u>IOFF</u>)	Position disc file to record IREC using block (IRB) and word in block (IOFF) returned by LOCF.	3-34
CLOSE	CALL CLOSE(<u>IDCB</u> , <u>IERR</u> , <u>ITRUN</u>)	Close file NAME to further access by caller.	3-18
CREAT	CALL CREAT(<u>IDCB</u> , <u>IERR</u> ,NAME, <u>ISIZE</u> , <u>ITYPE</u> , <u>ISECU</u> , <u>ICR</u> , <u>IDCBS</u>)	Create file NAME of size ISIZE, type ITYPE.	3-8
FCONT	CALL FCONT(<u>IDCB</u> , <u>IERR</u> , <u>ICON1</u> , <u>ICON2</u>)	Write EOF or position non-disc type 0 file.	3-41
FSTAT	CALL FSTAT(ISTAT)	Return status of mounted cartridges in ISTAT.	3-44
IDCBS	ISIZE= <u>IDCBS</u> (<u>IDCB</u>)	Return actual DCB buffer size in A register.	3-46
LOCF	CALL LOCF(<u>IDCB</u> , <u>IERR</u> , <u>I</u> REC, <u>IRB</u> , <u>IOFF</u> , <u>JSEC</u> , <u>JLU</u> , <u>JTY</u> , <u>JREC</u>)	Returns information on open file; next record in IREC, next block (IRB), next word (IOFF), etc.	3-31
NAMF	CALL NAMF(<u>IDCB</u> , <u>IERR</u> ,NAME, <u>NNAME</u> , <u>ISECU</u> , <u>ICR</u>)	Assigns new name (NNAME) to file NAME.	3-47
OPEN	CALL OPEN(<u>IDCB</u> , <u>IERR</u> ,NAME, <u>IOPTN</u> , <u>ISECU</u> , <u>ICR</u> , <u>IDCBS</u>)	Open file NAME for access by calling program.	3-13
POSNT	CALL POSNT(<u>IDCB</u> , <u>IERR</u> , <u>NUR</u> , <u>IR</u>)	Position disc or non-disc file NUR records from current position or to record NUR.	3-36
POST	CALL POST(<u>IDCB</u> , <u>IERR</u>)	Transfer contents of DCB buffer to disc.	3-48
PURGE	CALL PURGE(<u>IDCB</u> , <u>IERR</u> ,NAME, <u>ISECU</u> , <u>ICR</u>)	Purge file NAME and its extents from disc.	3-12
READF	CALL READF(<u>IDCB</u> , <u>IERR</u> , <u>IBUF</u> , <u>IL</u> , <u>LEN</u> , <u>NUM</u>)	Read record from open file to buffer (IBUF).	3-21
RWNDF	CALL RWNDF(<u>IDCB</u> , <u>IERR</u>)	Rewind or position to first record in file.	3-39
WRITF	CALL WRITF(<u>IDCB</u> , <u>IERR</u> , <u>IBUF</u> , <u>IL</u> , <u>NUM</u>)	Write record from buffer (IBUF) to file.	3-26

ASSEMBLY LANGUAGE CALLS
<p>Assembly language uses the same parameters as FORTRAN plus a return location parameter RTN.</p> <p>The general form is:</p> <pre> EXT routine-name . . JSB routine-name DEF RTN DEF p1 . . parameters DEF pn RTN return location </pre> <p>For example:</p> <pre> EXT CLOSE . . JSB CLOSE DEF RTN DEF IDCB DEF IERR . RTN . . IDCB BSS 144 IERR BSS 1 </pre>

COMMON PARAMETERS
<p>IBUF user-defined integer array used as read/write buffer for READF and WRITF calls.</p> <p>ICR one-word integer variable set to cartridge reference number of cartridge containing file:</p> <p style="padding-left: 40px;">positive integer = cartridge label negative integer = logical unit number</p> <p>IDCB user-defined integer array (Data Control Block) containing file control information on open file (16 words) plus packing buffer for data transfer (minimum 128 words); IDCB assumed to be 144 words unless IDCBS is specified.</p> <p>IDCBS one-word integer variable containing exact number of words in IDCB when IDCB greater than 144.</p> <p>IERR one-word variable where negative error code is returned, or for successful OPEN, file type, for successful CREAT, number of 64-word sectors.</p> <p>NAME 3-word integer array containing legal 6-character file name; must not begin with blank or number; no embedded blanks; use any printable ASCII character.</p> <p>ISECU one-word security code; integer or two ASCII characters:</p> <p style="padding-left: 40px;">positive = write protected negative = read/write protected zero = not protected</p> <p>OPTIONAL PARAMETERS IN FORTRAN CALLS ARE UNDERLINED.</p>

3-1. INTRODUCTION

The FMP program calls provide an interface between programs and the File Management utility routines. With these calls, you can open, close, read from and write to files from your program. In addition, the calls allow you to create or purge disc files, position either disc or non-disc files and directly control non-disc files.

Calls to the FMP routines may be written in Assembly Language, FORTRAN II or IV, ALGOL, or Multi-User Real-Time BASIC. Only FORTRAN IV is documented in this manual; calls from other languages use the same parameters and generate the same code.

3-2. FMP CALLS

Table 3-1 lists all the FMP calls according to general function and indicates the status, before and after the call, of the Data Control Block, a user array assigned to each open file. It also indicates when and if the file directory is accessed by the call.

3-3. THE DATA CONTROL BLOCK

The Data Control Block is a block of words defined within your program that acts as an interface between the program and the File Management Package. You must supply one Data Control Block for each open file. It is an array which contains control information for the file including the file name, type, size, and location on disc if the file is a disc file. In addition, it acts as a buffer for the physical transfer of data between a file and your program. The Data Control Block is used to:

- Avoid directory access for file information
- Keep track of current record position in file
- Provide a buffer for transfer of data between a file and the program.

Once a file is open, the Data Control Block is referenced for file information and the file name is no longer needed or used in FMP calls.

Each Data Control Block is an array with a minimum of 144 words. The first 16 words are a *control block* to provide all the file information required by the FMP calls. The remaining words are a *buffer* used for the transfer of data in blocks of 128 words. The 16-word control area is maintained and used only by FMP and must not be modified directly. Refer to Appendix C for the format of this area.

A Data Control Block buffer of 128 words is the minimum that can be specified. The buffer may be larger, as large as available memory, but any file can be accessed with the minimum 128-word buffer regardless of the buffer size specified at creation.

Table 3-1. FMP Call Summary

CATEGORY	ROUTINE	FUNCTION	DCB STATUS		DIRECTORY ACCESS	PARAGRAPH
			AT ENTRY	AT RETURN		
File Definition	CREAT	Enter file in directory; open exclusively for update.	CBO	OPNX	YES	3-8
	PURGE	Close file and remove from directory.	CBO	CLOS	YES	3-9
	OPEN	Open file.	CBO	OPN	YES	3-10
	CLOSE	Close file.	MBO	CLOS	YES	3-11
File Access	READF	Transfer record from file to user buffer.	MBO	OPN	EXTENTS	3-13
	WRITF	Transfer record from user buffer to file.	MBO	OPN	EXTENTS	3-14
File Position	LOCF	Retrieve current position and status of open file.	MBO	OPN	NO	3-16
	APOSN	Position disc file relative to a particular record.	MBO	OPN	EXTENTS	3-17
	POSNT	Position disc or non-disc file relative to current record.	MBO	OPN	EXTENTS	3-18
	RWNDF	Position file to first record.	MBO	OPN	EXTENTS	3-19
Special Purpose Routines	FCONT	Specify control functions for non-disc file.	MBO	OPN	NO	3-21
	FSTAT	Retrieve contents of cartridge directory.	—	—	—	3-22
	IDCBS	Retrieve actual size of DCB buffer used by FMP.	MBO	OPN	NO	3-23
	NAMF	Rename existing file.	CBO	CLOS	NO	3-24
	POST	Write DCB buffer to disc.	MBO	OPN	NO	3-25
Legend:						
CBO		Can be open: DCB can be assigned to open file; that file will be closed and, in case of CREAT and OPEN, file specified in call will be opened.				
MBO		Must be open: DCB must be assigned to open file.				
OPN		Open: File assigned to DCB is opened or is left open.				
OPNX		Open with extents: New file is assigned to DCB and is opened exclusively for update.				
CLOS		Closed: File assigned to DCB is closed; DCB is available for other use.				
EXTENTS		Directory is accessed only if call changed extents.				

Data Transfer

In addition to the Data Control Block buffer, another buffer must be defined in your program for transferring individual records. This buffer, the user buffer, is where a record to be written is specified and into which a record is read. The relation between the user buffer, the Data Control Block buffer, and a disc file is illustrated in Figure 3-1.

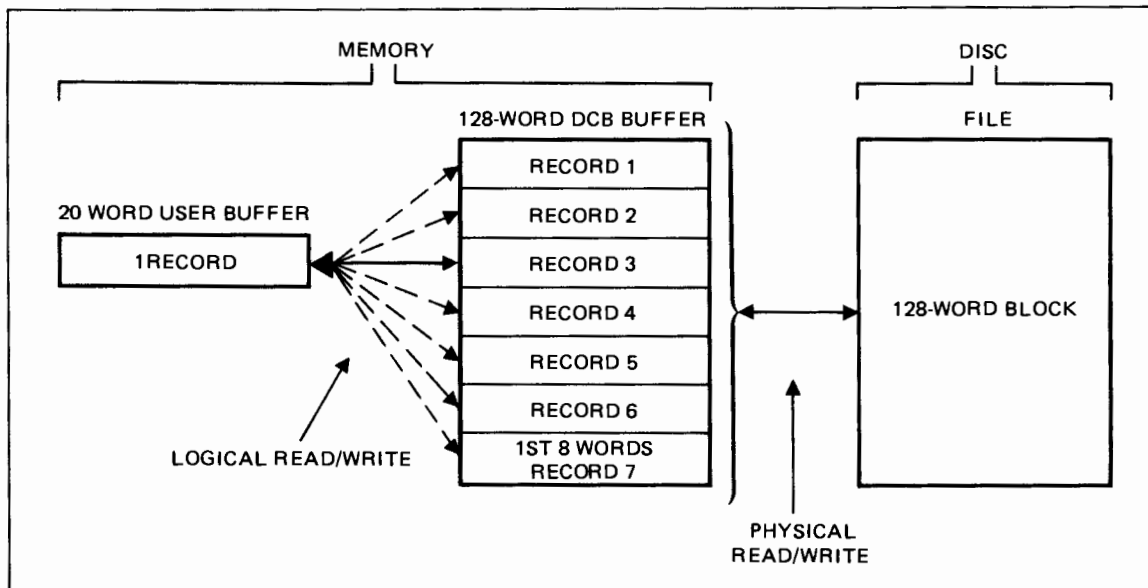


Figure 3-1. Sequential Transfer Between Disc File and Buffers

Each call to read or write a record transfers one record between the user buffer and the Data Control Block buffer. This type of transfer within memory is known as a logical read or write.

A physical read or write transfers a 128-word block between the disc file and the Data Control Block buffer. A physical write is performed automatically when the Data Control Block buffer is full, when the file is closed, or when a specific request is made with the POST call. On a read request, a block of data is physically read into the Data Control Block buffer from the disc only if the requested record is not already in that buffer. Any time a record being read or written is not wholly contained in the Data Control Block buffer (refer to record 7 in Figure 3-1), then the File Management Package reads or writes blocks until the entire record has been transferred.

When type 2 files are accessed randomly, the process is essentially the same as the sequential access illustrated in Figure 3-1 except that physical transfers may be more frequent since successive references are less likely to be to records in the same block in the Data Control Block buffer.

Since each record in a type 1 file is 128 words, the intermediate transfer to the Data Control Block buffer is omitted and each record is transferred directly between the user buffer and the file as illustrated in Figure 3-2. This type of access is the most efficient. A full 144 word Data Control Block must still be specified in the user program.

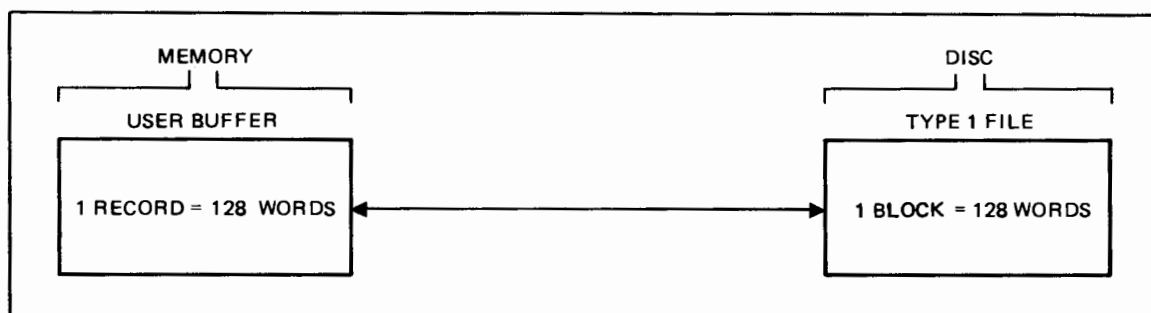


Figure 3-2. Data Transfer With Type 1 Files

Type 0 (non-disc) files also bypass the Data Control Block during transfers. Records in these files are written or read directly to or from the device identified as a type 0 file. A specific

number of words, rather than a record, is the unit transferred by a read or write request to this file type. The transfer can thus be tailored to the particular record length of the device.

3-4. PROGRAM CALLS

When a calling sequence is encountered during execution of a user program, the File Management Package executes the call according to the value of the parameters in the calling sequence, and/or returns information to areas defined in the parameter list. The number, type, and meaning of the parameters depends on the subroutine and are described in detail in this section.

For FORTRAN, the general form of the calling sequence is:

```
CALL name(p1,p2 . . . ,pn)
```

where:

name is the subroutine name.

*p*₁ through *p*_{*n*} are the parameters; parameters name a real or an integer array or variable or, if the parameter is one-word, a value. Position determines parameter meaning.

For Assembly Language, the general form of the calling sequence is:

```
EXT  name
:
:
JSB  name
DEF  RTN (or *+(n+1) where n is the number of parameters, may be zero)
DEF  p1
DEF  p2
:
:
DEF  pn
RTN  (return location)
```

where:

name is the subroutine name; it must always be defined as an external with an EXT statement before it is called.

RTN is the label of the location to which the subroutine returns upon completion; it must always follow the last parameter in the calling sequence.

*p*₁ through *p*_{*n*} are the parameters; each parameter names an integer array or variable.

3-5. COMMON PARAMETERS

Parameters used frequently in FMP calls are described here and in paragraph 3-6.

IDCB

This parameter specifies the array used as a Data Control Block. It must be at least 144 words, 16 words for file control information and 128 for the minimum buffer. For faster processing a larger buffer can be specified. In general, the larger the usable buffer, the faster the transfer rate. For example, a usable buffer of 256 words (IDCB=272 words) will nearly double the transfer rate for sequential accesses.

While a file may be created with a large DCB buffer, it may be accessed with any buffer that is at least 128 words long (IDCB=144 words). All transfers of data use the full actual buffer size. This size is always 128, or a multiple of 128, words. The **actual** size used is determined from the size requested for the Data Control Block at OPEN in conjunction with the file size in the following manner.

If the requested buffer size is greater than or equal to the file size, FMP allocates an actual Data Control Block equal to the file size +16 control words.

If the requested buffer size is less than the file size, FMP determines the actual Data Control Block buffer size according to the following rules:

- It must be a multiple of 128 words.
- It must be less than or equal to the size specified for IDCB by the user.
- It can be evenly and exactly divided into the total file size.

To illustrate, Table 3-2 shows the relation between file size, requested DCB buffer size and the actual size assigned by FMP for two file sizes, one with factors and the other a prime number. File size is given in 128-word blocks for convenience.

Table 3-2. Relation of Actual to Requested DCB Buffer Size

FILE SIZE IN BLOCKS	REQUESTED DCB BUFFER*		ACTUAL DCB BUFFER	
	BLOCKS	WORDS	BLOCKS	WORDS
10	1	128	1	128
	2	256	2	256
	3	384	"	"
	4	512	"	"
	5	640	5	640
	6	768	"	"
	7	896	"	"
	8	1024	"	"
	9	1152	"	"
	10	1280	10	1280
13	1	128	1	128
	2	256	1	128
	.	.	"	"
	.	.	"	"
	.	.	"	"
	12	1536	1	128
13	1664	13	1664	
*16 words must be added to the buffer size when dimensioning the Data Control Block array (IDCB).				

A call to routine IDCBS (paragraph 3-23) may be made to determine the actual size of the DCB array (actual buffer +16 control words).

IERR

When an error occurs during a subroutine call, a negative error code is returned in this parameter and also in the A-register. (Appendix B contains a list of the FMP error codes and their meaning.) For successful OPEN calls, the file type is returned as a positive integer in IERR; for successful CREAT calls the number of disc sectors used is returned in IERR as a positive integer. In calls where IERR is the last parameter, it is optional. It should be omitted only if errors are checked in the A-register. Negative error codes allow for easy error checking and error checking should not be omitted as a general practice.

NAME

For calls requiring a file name, the name is specified in this 3-word array. File names are six ASCII characters and must conform to the following rules:

- only printable characters may be used (blank ^ through _ or ←)
- first character must not be blank or a number
- if less than six characters, must be padded with trailing blanks
- embedded blanks are not allowed
- plus (+) minus (-) colon (:) or comma (,) are not allowed

Duplicate file names are not allowed on the same cartridge.

IBUF

This array is the user buffer and is included in the calls that transfer records: READF and WRITF. It should be as long as the longest record to be transferred. The record to be written must be in this array; when a record is read it is placed in this array.

3-6. OPTIONAL PARAMETERS

Most subroutines have one or more optional parameters. They always appear at the end of the calling sequence and may be omitted only from the end. Optional parameters are underlined in the formats. If an optional parameter is needed that is preceded by other unnecessary optional parameters, all the parameters up to the desired parameter must be included. Unused optional parameters should be set to zero. Three commonly used optional parameters are ISECU, ICR, and IDCBS.

ISECU

The file security code is specified in ISECU. It can be a positive or negative integer or two ASCII characters representing a positive integer. When characters are used, FMP converts them to their integer equivalent. If omitted, ISECU is set to zero.

ISECU is set at creation to zero, a positive value, or a negative value:

Creation	File Protection	File Reference
ISECU=0	unprotected	ISECU=any value for any access
ISECU=+n	write protected	ISECU=+n or -n to write or purge ISECU=any value to open or read
ISECU=-n	read/write protected	ISECU=-n for any access

ICR

This parameter is specified to restrict the file search to a particular cartridge or logical unit number. The search is for space if the call is CREAT, for a file name in other calls. ICR may be a positive or negative integer. If omitted, it is set to zero.

- If ICR = 0 The file search is not restricted to a particular cartridge. A CREAT call locates the file on the first cartridge with enough room; other calls using ICR search the cartridges in the order they appear in the cartridge directory to find the specified file.
- >0 File search is restricted to the cartridge identified by the cartridge reference number (ICR), an identifier assigned to all cartridges in the system. A list of cartridge reference numbers can be obtained with the FSTAT call or the :CL command (Section II).
- <0 File search is restricted to the cartridge associated with the logical unit number (-ICR). To illustrate, if ICR is -14, the file search is restricted to logical unit 14.

IDCBS

When a Data Control Block larger than 144 words is specified in parameter IDCB, then parameter IDCBS must also be specified. It informs FMP of the number of words available in the DCB buffer for data transfer. Any positive number can be specified; the actual usable buffer is always determined by FMP as described in the discussion of the IDCB parameter (paragraph 3-5). This size is never larger than the size specified in IDCBS, but it may be smaller. For example if IDCBS is less than 256 (any value between 1 and 255), then 128 words are used for the DCB buffer (144 for the entire Data Control Block).

Normally, you will specify IDCBS as 16 words less than the array size specified for IDCB.

3-7 FILE DEFINITION

A file may be defined in terms of its name, size, type, and where it is located. The CREAT call defines a disc file in this way and causes an entry to be made for the file in the file directory (refer to File Directory Format, Appendix C). Once defined, the file may be opened for access by any program with the proper security code. To open a file means to transfer the necessary information from the file directory to the control words of the Data Control Block and thus make a logical connection between the file name in the directory and the Data Control Block for the file. The CREAT call opens the file it creates to the calling program only, and only for update. For other types of access by other programs, the OPEN call must be used.

Following access, the file may be closed with the CLOSE call. Closing a file means that the connection between the Data Control Block and the file directory entry for the file is severed. The Data Control Block is freed for other uses, but the file is still defined in the file directory. PURGE performs all the CLOSE functions and, in addition, flags the directory entry so that the file is no longer defined.

3-8. CREAT

A call to CREAT creates a disc file; that is, it makes an entry in the File Directory for the file and allocates disc space to the file. This call only creates disc files. To create a non-disc (type 0) file, use the FMGR :CR command (paragraph 2-20).

Following execution of CREAT, the file is left open in the update mode for exclusive use of the program performing the call. If you want the file open in any other mode or for more than one program, use the OPEN call (paragraph 3-10).

Format

```
CALL CREAT(IDCB,IERR,NAME,ISIZE,ITYPE,ISECU,ICR,IDCBS)
```

Parameters

IDCB	Data Control Block; an array of 144 + n words where n is positive or zero.
IERR	Error return; one-word variable in which a negative error code is returned. If no error, it is set to the number of 64-word sectors (twice the number of 128-word blocks) in the created file.
NAME	File name; 3-word array containing ASCII file name.
ISIZE	File size; 2-word array with number of blocks in first word; if negative, rest of cartridge is allocated to file; second word, used only for type 2 files, contains record length in words.
ITYPE	File type; 1-word integer variable in range 1-32767; types 1-7 are FMP defined (see paragraph 1-10), higher types are user-defined special-purpose files.
ISECU	Security code; optional 1-word variable in range 0 through ± 32767 ; if omitted, code is set to zero and file is not protected; positive value is write protect only, negative value for read and write protect.

ICR	Cartridge reference; optional 1-word variable; if omitted, ICR is set to zero and space for the file may be allocated to any cartridge; if positive, cartridge is identified by the cartridge reference number, if negative, by logical unit number.
IDCBS	DCB buffer size; optional 1-word variable; set to number of words in DCB buffer if larger than 128; if omitted, FMP assumes DCB size (control words + buffer) is 144 words regardless of IDCB dimensions.

(See paragraphs 3-5, 3-6 for further discussion of commonly used parameters.)

File Size

Since records are addressed by number, the number of records must not exceed the maximum address (32767). Number of records can be determined by:

$$\frac{\text{words in file (blocks in ISIZE x 128)}}{\text{record length in words (ISIZE(2))}} = \text{number of records} \leq 32767$$

To obtain access of the 32676 maximum number of records in a type 2 file (see "File Type" below), the record length must be 128 or a multiple of 128.

When the exact size of the file is not known, an indefinite size can be specified by setting ISIZE to a negative number. The rest of the cartridge, but not more than 16383 blocks, is allocated to the file in this case (valid only for files of type 3 or greater). Any area that is unused may be returned with the ITRUN parameter when the file is closed. (Refer to CLOSE, paragraph 3-11.) Note that a file using all the remaining cartridge is not extendable since a file may not cross cartridge boundaries. Extents are created automatically when a write goes beyond the end-of-file of an extendable file (type 3 and above).

File Type

There are seven standard FMP file types as follows:

1. fixed length 128-word records; random access
2. fixed length user-defined record; random access
3. (and greater) variable length records; sequential access; extents provided
 4. source program
 5. relocatable program
 6. memory-image program
 7. absolute binary program

File types greater than 7 are user defined but are treated by FMP as type 3. Any special processing based on file type is not provided as a default, but must be specified.

When any file of type 3 or greater is created, FMP writes an EOF mark at the beginning of the file. As records are written to the file, the EOF is moved automatically to follow the last record.

Examples

1. Create a type 2 file called FIX with 100 blocks, 62 words per record, security code AB, and a DCB buffer of 128 words:

```

DIMENSION NAM1(3), ISIZ(2), IDCB1(144) ← 16 control words +
DATA NAM1/2HFI,2HX,2H / ← 128-word buffer
ISIZ=100 ← number of blocks
ISIZ(2)=62 ← record size

CALL CREAT(IDCB1,IERR,NAM1,ISIZ,2,2HAB) ← file type
IF(IERR.LT.0) GO TO 900 ← security code
.
.
. ← process any errors at 900

```

2. Create a type 3 file PROG1 with 60 blocks, no security code, using a 256 word DCB buffer, and locate it on logical unit 14.

```

DIMENSION NAM2(3), ISIZE(2), IDCB2(272) ← 16-control words + 256-word
DATA NAM2/2HPR,2HOG,2H1 / ← buffer
ISIZE=-1 ← file uses rest of cartridge, record length unspecified
ICR=-14 ← file located on logical unit 14
ITYPE=3 ← file type is 3
IDCBS=256 ← DCB buffer size

CALL CREAT(IDCB2,IERR,NAM2,ISIZE,ITYPE,0,ICR,IDCBS)
IF(IERR.LT.0) GO TO 900 ← error check
.
.
.

```

Another method is to use literals for all one-word variables:

```

DIMENSION NAME2(3), ISIZE(2), IDCB2(272)
DATA NAME2/2HPR,2HOG,2H1 /
.
.
.
CALL CREAT(IDCB2,IERR,NAME2,-1,3,0,-14,256)
IF(IERR.LT.0) GO TO 900
.
.
. ← continue after successful call

```

Care should be taken when literals are used as parameters since this practice can result in problems if the values are changed by the called routines.

Sequence of Operations (CREAT)

1. Check for enough parameters.
2. Close the DCB using CLOSE (ignore not open error).
3. Check legality of NAME.
4. Check legality of ITYPE.
5. Check legality of ISIZE, and if ITYPE = 2, then check that no more than 32767 records will fit in allotted size. (This is to prevent the file from containing more records than possible to address.) If ITYPE = 1, force record size to 128.
6. Get a system track and write skeleton entry on the track.
7. Schedule D.RTR to create the file.
8. Return the track.
9. Check for D.RTR error; if any, exit.
10. Open the file to the DCB.
 - a. Read the directory entry to the DCB buffer area.
 - b. Transfer directory parameters to the DCB.
 - c. Set current position pointers in the DCB.
 - d. Set update mode bit in DCB.
11. If file type ≥ 3 , set "written-on-flag" and EOF in DCB.
12. Return.



CALL OPEN(IDCBS,IERR,NAME,IOPTN,ISECU,ICR,DCBS)

Parameters

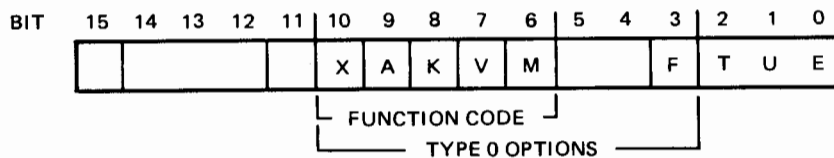
IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned if unsuccessful, file type if successful.
NAME	File name; 3-word array containing ASCII file name.
IOPTN	<p>Open Options; optional 1-word variable set to octal value to specify non-standard opens. If omitted or set to zero, the file is opened by default as follows:</p> <ul style="list-style-type: none"> • Exclusive use — only the calling program can access the file. • Standard sequential output — each record is written following the last, destroying any data beyond the record being written. • File type defined for file at creation is used for access. • Type 0 files use function code defined at creation. <p>To open a file with other options, set IOPTN as described below under OPEN Options.</p>
ISECU	Security code; optional 1-word variable; must be specified to open file created with negative code or to write on file protected with positive code; may be omitted if file not protected at creation.
ICR	Cartridge reference; optional 1-word variable; If set, FMP searches that cartridge for file; if omitted, it searches cartridges in the cartridge directory order and opens first file found with specified name.
IDCBS	DCB buffer size; optional 1-word variable; set to number of words in DCB buffer if larger than 128; if omitted, FMP assumes DCB size (control words + buffer) is 144 words regardless of IDCB dimensions.

See paragraphs 3-5, 3-6 for further discussion of commonly used parameters.

When a file is opened, it is positioned at the first record in the file.

OPEN Options

The IOPTN parameter is defined as follows:



The following bits may be set for any file type:

- E (bit 0) = 0 File opened exclusively for this program
 1 File may be shared by up to seven programs
- U (bit 1) = 0 File opened for standard (non-update) write
 1 File opened for update
- T (bit 2) = 0 Use file type defined at creation
 1 File type is forced to type 1

The following bits are used for type 0 files only (they are ignored when opening other file types):

- F (bit 3) = 0 Use function code defined at creation
 1 Use function code defined in bits 6-10 of IOPTN

Bits 6-10 correspond exactly to the function code used for RTE READ/WRITE EXEC call. The definition is repeated here for convenience.

- M (bit 6) = 0 ASCII data
 1 Binary data
- V (bit 7) = 0 If M=0 (ASCII) use column 1 for carriage control on line printer.
 If M=1 (binary) length of punched tape input determined by buffer length specified in READF call.
 1 If M=0 (ASCII) print column 1 on line printer.
 If M=1 (binary) length of punched tape input determined by word count in first non-zero character read.
- K (bit 8) = 0 Keyboard input is not printed.
 1 Keyboard input printed as received.

A with M determine whether binary and ASCII data are punched or printed on ASR35 Teleprinter-type device.

- A (bit 9) = 0 If M=0 (ASCII) output is printed but not punched.
 If M=1 (binary) output is punched but not printed.
 1 If M=0 (ASCII) output is punched but not printed.
 If M=1 (binary) output is punched but not printed.

X in combination with M and V determines what is read or written on paper tape.

X (bit 10) = 0 No function.

- 1 If M=1 and V=1, absolute binary tape format is expected; any leader, special control characters, etc. are skipped; ("honesty mode").

If M=1 and V=0, on input number of words in word count are read, leader is not skipped; on output, record terminator (4 feed frames) is not punched.

If M=0 and V=0, ASCII tape format is expected. On input, parity bit 8 is stripped and terminating line feed suppressed, but all other characters, including leader, are transmitted to user buffer; on output, carriage return and line feed are suppressed, but any trailing underscore or back arrow is punched.

Exclusive Open

By default, a file is opened for exclusive use of the calling program. An exclusive open is granted to only one program at a time. If the call is rejected because the file is open to another program, you must make the call again; it is not stacked by FMP. Exclusive open is useful in order to prevent one or more programs from destructively interfering with each other.

Non-Exclusive Open

If more than one program needs to access the file, it should be opened non-exclusively by setting the IOPTN E bit. A non-exclusive open may be granted to as many as seven programs per file at one time. A non-exclusive open will not be granted if the file is already opened exclusively. Each time an open is requested for the file, all programs currently having the file open are checked. If any program is dormant, the file is closed to that program. That type of close does not free the DCB and does not post the contents of the DCB buffer to the file. Any open flag will also be cleared if it does not point to a valid ID segment (possible if the file was left open on a different system).

Update Open

In update mode, IOPTN U bit set, the block containing the record to be written is read into the DCB buffer before it is modified. This insures that existing records in the block will not be destroyed. This mode of open has no effect on reading or positioning. File type governs the choice between update and standard (non-update) open.

Update mode should be used to write to type 2 files. A type 2 file should be opened in standard mode only when originally writing the file or adding to the end of the file, and then only if it is to be written sequentially.

Update mode is ignored for type 1 files. Although, like type 2, they are designed for random access with fixed length records and the end-of-file in the last word of the last block, each record is the same length as the block transferred so that there is no danger of writing over existing records.

For type 3 and above files, update mode is not generally used; most writes are sequential with an end-of-file mark written after each record. These files should be opened for update only if a record previously written to the file is being modified. In this case, care must be taken not to change the length of the modified record. If it is changed, a -005 error is issued. Regardless of the mode of open (update or standard) a record written beyond the end-of-file replaces the end-of-file and is followed by a new end-of-file.

Type 1 Access

Any file may be forced to type 1 access by setting the IOPTN T bit. Type 1 access is faster because it bypasses the Data Control Block buffer and transfers a sector of data directly to the user buffer defined as IBUF in a READF or WRITF call. The file type defined at creation is not affected; the file is treated as type 1 only for the duration of this open. You are responsible for any packing or unpacking of records in files forced to type 1. That is, if the records are less than 128 words, you must determine the start and end of each record. (Refer to READF and WRITF for particulars of type 1 access.)

Examples

1. Open a type 2 file named FIX for update in non-exclusive mode. The security code at creation was AB:

```

DIMENSION NAME(3),IDCB1(144)
DATA NAME/2HFI,2HX 2H /
.
.
.
CALL OPEN(IDCB1,IERR,NAME,3,2HAB)
IF (IERR ,NE. 2) GO TO 900
.
.
.

```

*set bit 1 for update and
bit 0 for non-exclusive open*

test for error; file type expected

2. Open type 3 file PROG1 with default options. The file is located on logical unit 14:

```

DIMENSION NAM(3),IDCB2(144)
DATA NAM/2HPR,2HOG,2H1 /
.
.
.
CALL OPEN(IDCB2,IERR,NAM,0,0,-14)
IF (IERR ,NE. 3) GO TO 900

```

default options

no security code

Although PROG1 was created with a DCB buffer of 256 words (refer to CREAT examples, paragraph 3-8), it can be opened and accessed with the minimum DCB buffer of 128 words.

3. Open type 0 file PTAPE (created with :CR command) and set options so that binary data is punched on paper tape without leader:

```

DIMENSION NAME(3),IDCB3(144)
DATA NAME/2HPT,2HAP,2HE /
IOPTN=2110B
.
.
.
CALL OPEN(IDCB3,IERR,NAME,IOPTN)
IF (IERR ,NE. 0) GO TO 900
.
.
.

```

set M, V, and X for "honesty mode"

Sequence of Operations (OPEN)

1. Check for enough parameters.
2. Close the DCB using CLOSE (ignore not open error).
3. Format and issue D.RTR request.
4. Call RMPAR to get D.RTR return parameters.
5. If D.RTR error, exit.
6. Open the file to the DCB.
 - a. Read the directory entry for the file into the DCB buffer area.
 - b. Transfer directory parameters to the DCB.
 - c. Set current position pointers in the DCB.
7. If OPEN protected (i.e., security code < 0), and security code mismatches, close the file and exit.
8. If type = 0 and subfunction option present (Bit 3 = 1), replace subfunction.
9. Return.

3-11. CLOSE

To close a file after use, call the CLOSE routine. The file remains in the system available to other programs following the close; the Data Control Block is freed for association with other files. A disc file opened for exclusive use of the calling program may be truncated to its actual length.

Format

CALL CLOSE(IDCB,IERR,ITRUN)

Parameters

IDCB	Data Control Block, an array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned if truncation unsuccessful; only required when ITRUN is specified.
ITRUN	Truncation; optional 1-word variable containing integer number of blocks to be deleted from the file at closing; if omitted or zero, the file is closed without truncation; if negative, only extents are truncated.

See paragraph 3-5 for further discussion of commonly used parameters.

File Truncation

When a file has been created with more blocks than are actually needed to accommodate the data in it, it can be truncated at closing to save disc space. A file may be truncated only if:

- the file is a disc file
- the current position is in the main file, not in an extent
- the file was opened with the correct security code
- the file was opened for exclusive use of the calling program
- the number of blocks deleted are less than or equal to the total number of blocks in the file.

If all these conditions are met, the value of ITRUN can be a:

- positive integer — to specify the number of blocks to be deleted from the end of the main file: any extents are automatically truncated: if equal to the total number of blocks in the file, the file is purged.
- negative integer — to specify that any extents be deleted from the file; the main file is not affected.

The value of ITRUN when positive can be determined from information returned by a previous call to LOCF (paragraph 3-16) assuming the current position is at the end of the data in the file. In this case, the last block number written or read (IRB-1) is subtracted from the total blocks with which the file was created (JSEC/2) and assigned to ITRUN. When negative, ITRUN can be any value. The number of extents need not be known. If the file is currently positioned in an extent, it can be re-positioned to the main file with RWDF (paragraph 3-19).

A zero value for ITRUN is exactly the same as omitting this parameter; a standard close is performed with no truncation.

Examples

1. Close file FIX (IDCB1) with no truncation; assume FIX has been opened:

```

DIMENSION IDCB1(144) ←————— file name associated with IDCB1
.
.
.
CALL CLOSE(IDCB1, IERR) ←————— error return optional but good practice
IF (IERR .LT. 0) GO TO 999
.
.
.

```

IDCB1 is freed for other files.

2. Close type 3 file PROG1 and truncate it to exactly the number of blocks used. Assume the file was written sequentially and the current location is at the end of the data in the file.

```

DIMENSION IDCB2(144) 4)
.
.
CALL LOCF(IDCB2,IERR,I,IRB,I,JSEC) ← call LOCF for file length and
ITRUN=JSEC/2-(IRB-1) ← next block
CALL CLOSE(IDCB2,IERR,ITRUN) ← IERR required with truncation
IF(IERR .LT. 0) GO TO 900
.
.
.

```

Since JSEC contains the number of sectors (2 blocks per sector) it must be divided by 2 for the number of blocks. IRB is the next block; for the current block subtract 1.

3. Close the same file, but delete only the extents; use the RWNDF call (paragraph 3-19) to insure that current position is in the main file:

```

DIMENSION IDCB2(144)
.
.
.
IF(RWNDF(IDCB2))900,14,14
10 CALL CLOSE(IDCB2,IERR,-1) ← ITRUN negative to delete all extents
IF(IERR .LT. 0) GO TO 900
.
.
.

```

Sequence of Operations (CLOSE)

1. Check for enough parameters.
2. If DCB is not open, reject call.
3. If block currently in core was written on, write it to the file.
4. Check file type, current extent, and security for truncate option.
5. Call D.RTR to close the file (D.RTR makes the rest of the truncate checks).
6. Return.

3-12. FILE ACCESS

Information in files is accessed with the READF and WRITF routines. Calls to these routines are basically the same whether the file is a device (type 0) or a disc file (type 1 and above). Whether reading or writing, you can specify that exactly one record be transferred or you may specify a particular number of words. In general, it is good practice to specify the number of words since this permits generality among file types.

The normal mode of access for files of type 3 and above is sequential. Such files are created with an end-of-file in the first record. The first record written overrides the end-of-file and a new end-of-file is written immediately following the record. As each subsequent record is written, the process is repeated so that the end-of-file always follows the last record written.

Variable length records are assumed for these file types. For this reason it is necessary to specify the number of words when the record is written. On a read, if the number of words is not specified, FMP determines its length and reads exactly one record.

For file types 1 and 2, random access is the normal mode. The end-of-file is written at the end of the file according to the file size at creation. Since each record is a fixed length determined at creation, the file is easily positioned to a particular record. Generally, one record is written or read at a time, although more may be transferred when accessing a type 1 file.

When accessing a type 0 file, the number of words should always be specified unless a zero-length record is to be read or written or a record skipped. End-of-file marks are not written automatically to type 0 files; you must specify the end-of-file.

3-13. READF

This routine reads a record from an open file to the user buffer. Either one full record or a specified number of words is read. The record to be read may be the record at which the file is currently positioned or, for type 1 and 2 files, it may be any specified record.

Format

```
CALL READF(IDCB,IERR,IBUF,IL,LEN,NUM)
```

Parameters

IDCB	Data Control Block; an array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
IBUF	User buffer; array into which the record is read; it should be large enough to contain the record; if IL is specified, it should be length IL.
IL	Length in words; optional 1-word variable specifying number of words to be read; should not be omitted for type 0 files, for other files, 1 record is read if IL is omitted; refer to Table 3-3 for details of IL use.

LEN	Words read; optional 1-word variable in which actual number of words read is returned; set to -1 if end-of-file read; if omitted, information not supplied.
NUM	Record number; optional 1-word variable set to record number to be read if positive, to number of records to backspace if negative; used only for type 1 and 2 files; if omitted, record at current position is read.

See paragraphs 3-5, 3-6 for further discussion of commonly used parameters.

Relation of IL to File Type

It is a good idea to specify IL for file type 0 and it doesn't hurt to specify it for other file types. If you do not know the length of a disc file record, IL can be specified as the user buffer length to prevent reads beyond this area. If the record is shorter than IL, the exact record length is read for file types greater than type 1. Table 3-3 illustrates the effect of IL depending on file type.

Table 3-3. Effect of IL Parameter in READF

IL VALUE	FILE TYPE 0	FILE TYPE 1	FILE TYPE > 1
IL > 0	Up to IL words are read; if less than IL, file defined record length is read.	Exactly IL words are read; IL may be more or less than 128-word record.	Up to IL words are read; if less than IL, actual record length is read.
IL = 0 (not recommended)	Zero length record is read; usually record is skipped and counted as read.	No action. (Zero-length read, no position change.)	Record is skipped and counted as read.
IL omitted	Zero-length record is read; usually record is skipped and counted as read. (Not recommended.)	128-word record is read.	Actual record length is read.

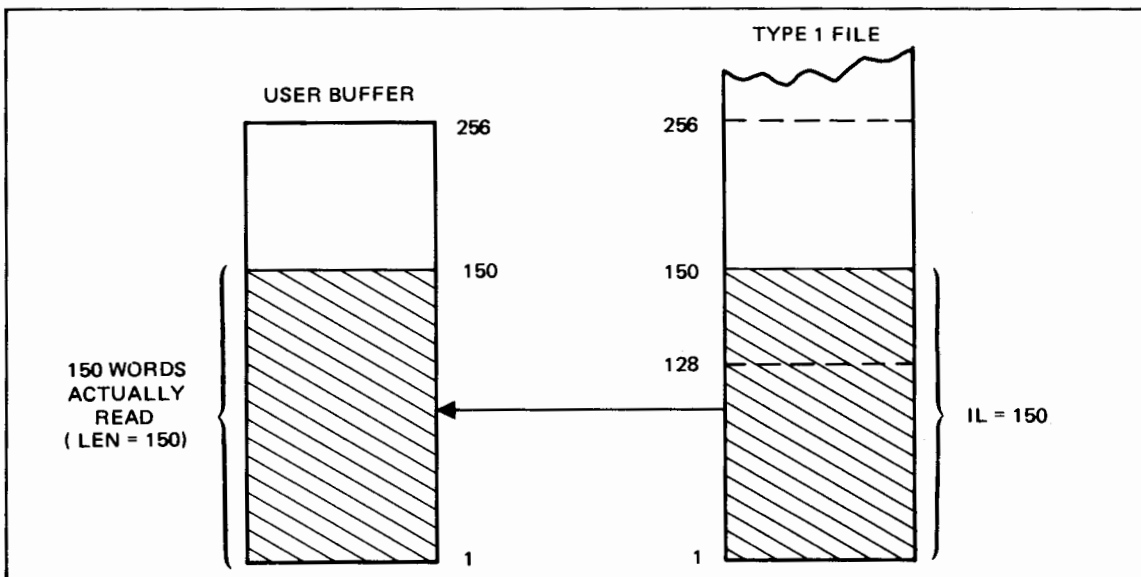


Figure 3-3. Read Type 1 File When IL Greater Than 128

Figure 3-3 illustrates a type 1 file read. The file is read directly to the user buffer when the number of words specified in IL is greater than the 128 words expected for a type 1 file. Other file types may be forced to type 1 at open in order to benefit from this type of transfer.

Using LEN

Upon completion of a read, the actual number of words transferred to the user buffer is returned in LEN. If, however, the number of words in LEN is equal to IL, more words may actually have been in the disc record. This is because LEN is never set to a value greater than IL.

To illustrate, suppose IL is specified as 80 words. If 10 words are transferred, then LEN is set to 10. But whether 80 or more words are in the record, LEN is still set to 80, the value of IL.

LEN can be used to test for possible overflow of the user buffer. Except for type 1 files, the user buffer and IL can be specified one word larger than the largest expected record. If, when tested, LEN equals this size, it is a good indication that the record read was too large for the buffer. Do not use this test for type 1 files since exactly IL words are read for this file type.

Another use of LEN is to test for end-of-file in all file types except 1 and 2. For types 1 and 2, an end-of-file is reported as an error in IERR. Depending on file type, reading an end-of-file results in the following:

- | | |
|-------------|---|
| Type 0 | LEN is set to -1 when EOF is read; no error occurs and access may continue beyond the end of file. |
| Type 1 & 2 | IERR is set to -12 indicating an error. Access is not permitted beyond the end of file. |
| Type 3 & up | LEN is set to -1 for the first EOF read; no error occurs but an attempt to read past this EOF causes an error (IERR = -12); you may not read past the end-of-file, but you may write beyond it. |

Note that length words in variable-length records (file types 3 and above) are not transferred to the user buffer and are not counted in LEN.

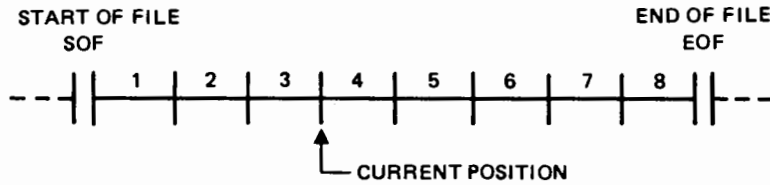
Positioning With NUM



NUM is used only to position file types 1 and 2; it may be specified for other file types but is ignored. If positive, NUM specifies the record number of the record to be read: records are numbered from the first record in the file starting with 1 and proceeding sequentially up to a maximum of 32767. If negative, NUM specifies the number of records to backspace from the current position in the file.

To illustrate, assume the file is positioned at the beginning of record 4:

1. If NUM=0 or is omitted, read record 4.
2. If NUM=6, read record 6.
3. If NUM=-3, read record 1.



Examples

1. Read records in a type 0 file until an end-of-file is reached; the record length is 80 words but 81 words are assigned to the buffer in order to test LEN; assume the file is positioned at the first record:

```

DIMENSION IDCB0(656),IBUF(81)
.
.
.
100 CALL READF(IDCB0,IERR,IBUF,81,LEN)
IF(IERR.LT.0) GO TO 900
IF(LEN.EQ.-1) GO TO 500 ← test for end-of-file
IF(LEN.GT.80) GO TO 550 ← test for record greater than 80 words
.
. ←————— process record
.
GO TO 100 ←————— read next record
.
.
.
500 CALL CLOSE(IDCB0,IERR) ←————— close file at end-of-file
IF(IERR.LT.0) GO TO 910

```

2. Read record 24 from a type 2 file with a record length of 256 words using a 257 word buffer:

```

DIMENSION IDCB1(272),IBUF1(257)
.
.
.
CALL READF(IDCB1,IERR,IBUF1,257,LEN,24)
IF(IERR.LT.0) GO TO 900 ← test for error or EOF
IF(LEN.EQ.257) GO TO 550 ← test for too long record
.
. ←————— process record
.
.

```

3. Read file with variable length records until first end-of-file is reached. Assume the file is positioned at the first record and that no record exceeds 128 words:

```

        DIMENSION IDCB2(144),IBUF2(129)
        .
        .
        .
100     CALL READF(IDCB2,IERR,IBUF2,129,LEN)
        IF(IERR .LT. 0) GO TO 900
        IF(LEN .EQ. -1) GO TO 500 ← test for end of file
        IF(LEN .EQ. 129) GO TO 150 ← test for possible buffer overflow
        . ← process record
        .
        .
        GO TO 100 ← read next record
        .
        .
        .
500     CALL CLOSE(IDCB2,IERR) ← close file at end-of-file
        .
        .
        .

```

Sequence of Operations (READF)

1. Set read flag and fetch parameters.
2. Check for enough parameters.
3. If DCB is not open, reject call.
4. If read or update, set reading flag.
5. If type 1 force DCB length to 128.
6. If type 1 or 2, do random access positioning (implies EOF if initial position not in file). If new position not in currently resident block, then write the DCB buffer if it was written on.
7. If type 2 or above, and reading flag set, and DCB buffer is empty, then read to DCB.
8. If type less than 3, skip to 18.
9. If current position is at EOF set EOF encountered flag in DCB. If already set, set IERR = -12 and exit; else step record count, set LEN = -1, and exit.
10. Read the record length.
11. If IL too short, set skip count.
12. Read the record.
13. If skip count set, skip the rest of the record.
14. If type 2, set record count and exit.
15. Read the length word.
16. If lengths do not match, take error exit.
17. If reading flag not set, set EOF in buffer, set written on flag, step record count and exit; else step record count and exit.

FMP Calls

18. If type 2, go to 11.
19. If type 0, go to 30.
20. If type 1, round up length to even 238 words and save as increment in record count.
21. Check if request is within the file.
22. If track switch, compute maximum words this access.
23. Read the record.
24. If type 0 read, do EOF test and return.
25. If type 1, check for disc errors.
26. Update for rest of record.
27. If more to transfer, go to 22.
28. Update record count.
29. Return.
30. Test read legality bits.
31. Go to 23.

3-14. **WRITF**

A call to this routine transfers a record from the user's buffer to an open file. For files of type 0 or type 3 and above, a specified number of words is written. Type 1 files are written in blocks of 128 words. For type 2 files the exact record length specified at creation is written.

Format

CALL WRITF(IDCB,IERR,IBUF,IL,NUM)

Parameters

IDCB	Data Control Block; an array of 144 +n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
IBUF	User buffer; array containing the record to be written; it should be large enough to contain the largest record to be written.
IL	Length in words; optional 1-word variable specifying number of words to be written; if omitted, one record is written to type 1 and 2 files, zero-length record to other file types; refer to Table 3-4 for details of IL use.
NUM	Record number; optional 1-word variable containing record number to be written if positive, number of records to backspace if negative; used only for type 1 and 2 files; if omitted, record is written to current file position.

See paragraphs 3-5, 3-6 for further discussion of commonly used parameters.

Relation of IL to File Type

IL should be specified for all but type 2 files and may be specified for all files. It is ignored by type 2 files but can be used with type 1 files to write more than one 128-word record at a time. For files of type 3 and above, it is essential to specify record length in IL. To omit IL for these file types is the same as setting IL=0, a zero-length record is written. Refer to Table 3-4.

IL can also be used to write an end-of-file on files of type 0, type 3 or greater. An attempt to write an end-of-file to a type 1 or 2 file is ignored; no error is indicated.

Table 3-4. Effect of IL Parameter in WRITE

IL VALUE	FILE TYPE 0	FILE TYPE 1	FILE TYPE 2	FILE TYPE > 2
IL > 0	Exactly IL words are written.	IL is rounded up to 128 or a multiple of 128.	IL is ignored; file defined record length is written.	Exactly IL words are written.
IL = 0	Zero length record is written.	No action.	IL is ignored; file defined record length is written.	Zero length record is written.
IL omitted	Zero length record is written.	128 words are written.	IL is ignored; file defined record length is written.	Zero length record is written.
IL = -1	End-of-file is written.	No action.	No action.	End-of-file is written.
IL < -1 not recommended	IL is treated as a character count.	No action.	No action.	Undefined.

When IL is not 128 or a multiple of 128 for a type 1 file, it is rounded up so that 128 or a multiple of 128 words are always transferred. The user buffer need be no larger than the size specified in IL. If the exact record size is always read, no problems result from the transfer of words beyond the buffer. Figure 3-4 illustrates a write to a type 1 file with IL = 150 words. In this case, 256 words (the shaded area) are actually transferred.

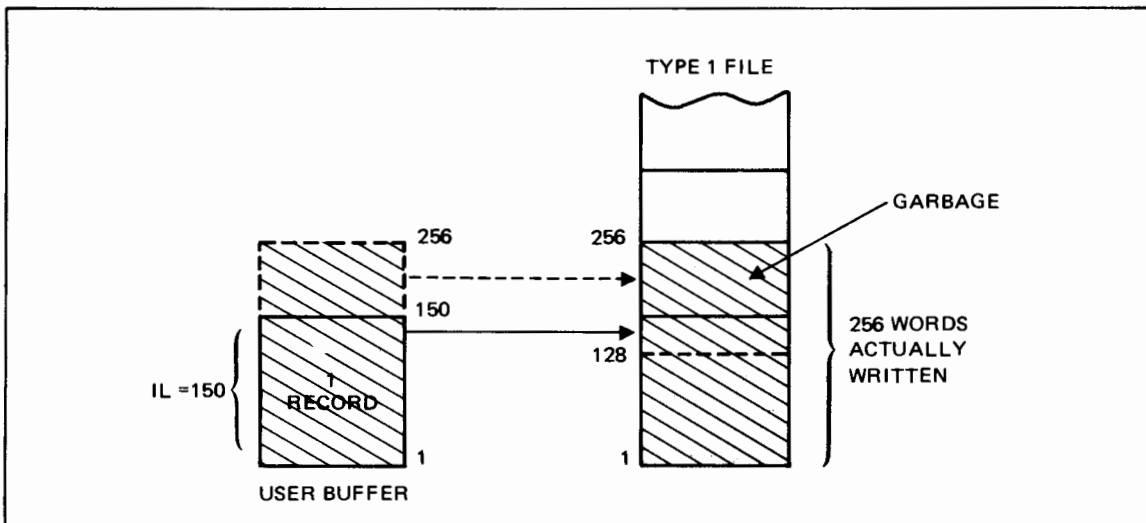


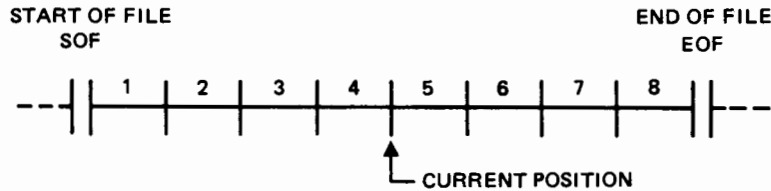
Figure 3-4. Write Type 1 File When IL Greater Than 128

Positioning with NUM

NUM is used only to position file types 1 and 2; if specified for other file types, it is ignored. A positive value causes a write to the specified record number; records are numbered relative to the start of the file beginning with 1. When negative, NUM specifies the number of records to backspace from the current file position.

To illustrate, assume the file is positioned at the beginning of record 5:

1. If NUM=0 or is omitted, write is to record 5.
2. If NUM=6, record number 6 is written.
3. If NUM=-3, record number 2 is written.



NOTE

Although it is possible to rewrite specific records in files of type 3 and above, great care must be taken. If the length of the existing record and that of the replacing record are not identical, the integrity of the file is destroyed.

Examples

1. Write records sequentially to a file starting at record 1; when all records are written, write an end-of-file; set IL to the exact record length of each record using a maximum record length of 100. File could be type 0 or a type 3 or above:

```

        DIMENSION IDCB0(144),IBUF0(100)
        .
        .
        .
100 ←----- move record to IBUF0
        .
        .
        .
        IL = ----- number of words in record
                    (after last record, set IL=-1)
        CALL WRITF(IDCB0,IERR,IBUF0,IL)
        IF(IERR .LT. 0) GO TO 900 ←----- check for error
        IF(IL) 110,100
110   CALL CLOSE(IDCB0,IERR) ←----- close file after EOF
        .
        .
        .
    
```

2. Write record number 24 to a type 2 file with a record length of 256 words:

```

DIMENSION IDCB2(272),IBUF2(256)
.
.
. ←————— move record to IBUF2
CALL WRITF(IDCB2,IERR,IBUF2,0,24)
IF(IERR .LT. 0) GO TO 900
.
.
.

```

Sequence of Operations (WRITF)

1. Set write flag and fetch parameters.
2. Check for enough parameters.
3. If DCB is not open, reject call.
4. Check security.
5. If update, set reading flag.
6. If type 1, force record length to 128.
7. If type 1 or 2 and EOF write, then exit, or do random access positioning (implies EOF if initial position not in file). If new position not in currently resident block, then write the block if it was written on.
8. If type 2 or above, and reading flag set and DCB buffer is empty, then read to DCB.
9. If type less than 3, skip to 19.
10. If current position is at EOF, then clear reading flag.
11. If EOF write: (a) then set EOF in buffer, (b) set written on flag, (c) step record count, and (d) exit.
12. If reading, then this is an update write; check lengths (must match).
13. Write the record length.
14. Write the record.
15. If skip count set, skip the rest of the record.
16. If type 2, then to 11c.
17. Write the length word.
18. If reading flag not set, go to 11a; else 11c.
19. If type 2, go to 14.
20. If type 0, go to 31.
21. If type 2, round up length to even 128 words and save as increment in record count.
22. Check if request within file.
23. If write, use round-up length.
24. If track switch, compute maximum words this access.
25. Write the record.

FMP Calls

26. If type 1, check for disc errors.
27. Update for rest of record.
28. If more to transfer, go to 24.
29. Update record count.
30. Return.
31. Test write legality bits.
32. If EOF write, make control request and return.
33. Go to 25.

3-15. FILE POSITIONING

The FMP positioning routines provide a variety of ways to position a file:

- To a particular record in a sequential (variable-length record) file (LOCF and APOSN).
- To a particular record in a sequential file with variable-length records (LOCF,APOSN,POSNT).
- Relative to the current position in a forward direction in a disc or non-disc file with fixed-length records, or backwards if the file also permits backspacing (POSNT).
- To the first record in the file for any file that permits backspacing (RWNDF).

Disc files with fixed length records (type 1 and 2) can also be positioned to a particular record or backspaced with the NUM parameter of READF or WRITF. However, they may be positioned for sequential access with APOSN or positioned forward with POSNT.

Each time a file is opened, it is positioned by FMP to the first record in the file. For this reason, only POSNT may be useful immediately after an open. All disc files and magnetic tape files may be backspaced. Non-disc files other than magnetic tape usually have fixed length records but cannot be backspaced.

In addition to the position routines, the routine LOCF is essential when a file is to be positioned for sequential access with APOSN. The parameters for such positioning are retrieved from a call to LOCF and can be passed directly to APOSN. LOCF also returns status information on any open file and, thus, has broader uses than simply file positioning.

3-16. LOCF

A call to this routine retrieves status and location information on an open file. The information is obtained from the Data Control Block control words for the file. The minimum information returned is the next record number; all other information is optional.

Format

```
CALL LOCF(IDCB,IERR,IREC,IRB,IOFF,JSEC,JLU,JTY,JREC)
```

Parameters

IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
IREC	Next Record; 1-word variable in which number of next sequential record is returned.
IRB	Next block; optional 1-word variable in which next block number is returned; not returned for type 0 files; identical to IREC for type 1 files; includes extents if file was extended.
IOFF	Next word; optional 1-word variable in which number of next word in record is returned; not returned for type 0 files.

JSEC	File size; optional 1-word variable in which number of sectors in file at creation is returned; not returned for type 0 files; JSEC/2 provides number of blocks.
JLU	Logical unit; optional 1-word variable in which logical unit to which file is allocated is returned.
JTY	File type; optional 1-word variable in which file type at open is returned.
JREC	Record size; optional 1-word variable in which record size of type 1 and 2 files or read/write code for type 0 files is returned; not applicable to files with variable length records (type 3 and above).

See paragraph 3-5 for further discussion of IDCB and IERR.

Location Information

Together, IREC, IRB, and IOFF provide the current position within a disc file; they are not set for non-disc files. The values in these parameters may be passed directly to APOSN (paragraph 3-17) to position the file to this location. The values returned in IRB and IOFF give the exact physical location of the record pointer in the file. The values of IRB and IOFF are based on a Data Control Block buffer size of 128 words. If the actual Data Control Block buffer size is greater than 128, these values are adjusted automatically by APOSN.

IREC numbers records starting with 1 for the first record, 2 for the second, and so forth. IREC alone is sufficient to find the location of type 1 files. IRB numbers blocks starting with 0 for the first block in the file, 1 for the second, and so forth. If the file is extendable (type 3 and above), IRB includes extent information and is specified as:

$$\text{blocks in main file} \times \text{extent number} + \text{block number in current extent}$$

IOFF numbers the words in a record relative to the beginning of the record but beginning with zero. Since all records are assumed to be no greater than 128 words (the size of the minimum Data Control Block buffer), the range of IOFF is 0 through 127.

Status Information

JSEC is always an even number with two 64-word sectors for each 128-word block in a disc file. It is not applicable to non-disc files.

JLU is the logical unit to which a file, disc or non-disc is allocated.

JTY is the file type of the file; if forced to type 1 at open, then 1 is returned.

JREC as a record size is meaningful for type 2 files only; it is the size specified at creation. For type 1 files, whether actual or forced to type 1 at open, JREC is set to 128 on the first read or write access.

For type 0 files, JREC specifies the read/write access code:

bit 15 = 1 read legal

bit 0 = 1 write legal

Examples

1. Determine the actual location of the record pointer in the open file PROG1 defined in IDCBS2:

```

DIMENSION IDCBS2(144)
.
.
.
CALL LOCF(IDCBS2, IERR, IPEC, IRB, IUFF)
IF(IERR .LT. 0) GO TO 900 ← process errors at 900
.
.
.

```

2. Open an existing file DATA and create file NEW with the same file type and size; then transfer all data from DATA to NEW:

```

DIMENSION IDATA(272), INEW(272), NDATA(3), NNEW(3)
DIMENSION IBUF(256), ISIZ(2)
DATA NDATA/2HDA,2HTA,2H /, NNEW/2HNE,2HW ,2H /
CALL OPEN(IDATA, IER, NDATA, 0, 0, 1, 272)
IF(IER .LE. 0) GO TO 900 ← test for error or type 0
10 CALL LOCF(IDATA, IER, 1, 1, 1, ISIZ(1), 1, ITYP, ISIZ(2))
IF(IER .LT. 0) GO TO 920
20 ISIZ(1) = ISIZ(1) / 2 ← set ISIZ to number of blocks
CALL CREAT(INEW, IER, NNEW, ISIZ, ITYP, 0, 0, 256)
IF(IER .LT. 0) GO TO 920
30 CALL READF(IDATA, IER, IBUF, 256, L)
IF(IER .LT. 0) GO TO 920
40 CALL WRITEF(INEW, IER, IBUF, L)
IF(IER .LT. 0) GO TO 920
IF(L) 950, 30 ← return for next record if not end-of-file

```

PROCESS ERRORS AND CLOSE FILES

```

900 IF(IER .NE. 0) GO TO 920
WRITE(1, 910) ← type 0 file cannot be created from program
910 FORMAT("IDATA IS TYPE 0 FILE")
GO TO 950
920 WRITE(1, 930)
930 FORMAT("ERROR ATTEMPTING TO COPY IDATA TO NEW")
950 CALL CLOSE(IDATA, IER)
1000 CALL CLOSE(INEW, IER) ← close files following end-of-file or error
END

```

Sequence of Operations (LOCF)

1. Check for enough parameters.
2. Fetch DCB.
3. Check that file is open.
4. Set IREC.
5. If type 0, go to 8.

6. If type 1 or 2, compute current address from the record number.
7. Compute and set IOFF, IRB, JSEC.
8. Set JTY, JLU, JREC.
9. Return.

3-17. APOSN

This routine is called to position any disc file to a specific record. The record location may be determined by a prior call to LOCF (paragraph 3-16).

APOSN is intended to position sequential files with variable length records prior to a read or write request. It may be used to position random files with fixed length records (types 1 and 2) but it must *not* be used to position non-disc files (type 0). POSNT may be used to position type 0 files.

Format

CALL APOSN(IDCB, IERR, IREC, IRB, IOFF)

Parameters

IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
IREC	Next record; 1-word variable set to number of next sequential record; may be determined by a prior call to LOCF.
IRB	Next block; optional 1-word variable set to next block number; may be determined by a prior call to LOCF; omitted only for files with fixed-length records.
IOFF	Next word; optional 1-word variable set to number of next sequential word in record where number is in range 0-127; may be determined by a prior call to LOCF; omitted only for files with fixed-length records.

APOSN assumes the values entered for the record address are based on a 128-word Data Control Block buffer. The routine adjusts the values to the buffer size currently in use if it is greater than 128.

IREC must be set; if not set to a specific record number by user, it may be set to a value returned by a call to LOCF. The three values IREC, IRB, and IOFF may all be retrieved through LOCF. This permits the resetting of the file location to its position when LOCF was called.

NOTE

Whenever a file with variable-length records is positioned, the two optional parameters IRB and IOFF must be included.

Examples

1. Call LOCF to retrieve the current position parameters. After reading more of the file, re-position the file associated with IDCB2 to the position whose location was saved:

```

DIMENSION IDCB2(144)
*
*
* CALL LOCF(IDCB2,IERR,IREC,IRB,IOFF) ← retrieve location at
IF(IERR .LT. 0) GO TO 900 ← start of access
*
* ← read and process records in the file
*
* CALL APOSN(IDCB2,IERR,IREC,IRB,IOFF) ← position to previously
IF(IERR .LT. 0) GO TO 900 ← saved location
*
*
*

```

The values of IREC, IRB, and IOFF retrieved by the call to LOCF are simply passed to the call APOSN by using the same variable names in both calls.

2. Position type 2 file defined in Data Control Block IDCB1 to the 6th record in the file and then read next 8 records in sequence:

```

DIMENSION IDCB1(144),IBUF(128)
*
*
* CALL APOSN(IDCB1,IERR,6) ← position file
IF(IERR .LT. 0) GO TO 900
10 COUNT=0
CALL READF(IDCB1,IERR,IBUF1,128,LEN)
IF(IERR .LT. 0) GO TO 900
20 IF(LEN .EQ. -1) GO TO 50 ← test for end of file
*
* ← process record
*
COUNT=COUNT+1
IF(COUNT .LT. 8) GO TO 10
*
*
*

```

Sequence of Operations (APOSN)

1. If DCB is not open, reject call.
2. Check if file type = 0.
3. Check for enough parameters.
4. If type 1 or 2, go to 7.
5. Call subroutine LOCF to get the current IRB.

6. Position to the new block without reading file.
 - a. May imply an extent change.
 - b. Implies writing current block if written on, before positioning to a new block.
7. Set current buffer pointer.
8. If record number is less than 1, error exit.
9. Record number is returned.
10. Return.

3-18. POSNT

This routine positions a file relative to the current file position or to a specific record number. It can be used to position all file types.

Format

<p>CALL POSNT(IDCB,IERR,NUR,<u>IR</u>)</p>	
<p>Parameters</p>	
IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
NUR	Number of records; 1-word variable specifying the number of records to position forward if positive, backward if negative; if IR is included as non-zero value, NUR specifies record number to which file is positioned.
IR	Optional 1-word variable set to indicate that NUR be interpreted as record number; if omitted or zero, NUR is treated as number of records to space forward or backward. Refer to Table 3-5.
<p>See paragraph 3-5 for further discussion of IDCB and IERR.</p>	

Table 3-5 Relation Between Parameters NUR and IR

NUR	IR = 0 OR OMITTED RELATIVE POSITION	IR ≠ 0 ABSOLUTE POSITION
NUR > 0	Position forward number of records specified.	Position to record number specified.
NUR = 0	No operation.	No operation
NUR < 0	Position backward number of records specified.	Error

Positioning Type 0 Files

When the file is on a non-disc device, the forward or backward positioning specified by NUR must be legal for the device.

To forward position a type 0 file, the records are read until one less than the specified number of records is read or an EOF is read. In every case, an EOF terminates positioning.

When backspacing a type 0 file, the first record backspaced over may be an EOF. If an EOF is encountered not as the first record backspaced over, an error (-12) is returned and the call terminates after forward spacing to position the file immediately after the EOF.

Positioning Type 1 and 2 Files

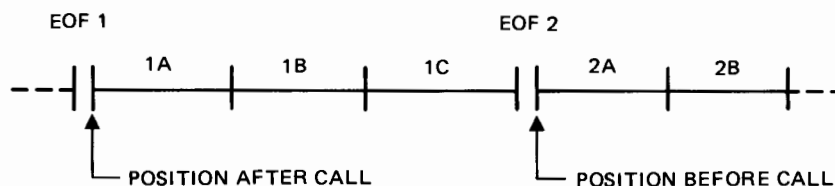
POSNT may be used to position these file types, however, file positioning for files with fixed length records can be specified in the read or write requests and POSNT may not be necessary.

Positioning Type 3 and Above Files

These files are treated as magnetic tape files. To be correct, a backspace should be issued after an EOF is read and before continuing to write on the file. This action writes the next record over the EOF allowing a correct extension of the file for either disc or magnetic tape files. If the file is on disc, it can be extended without backspacing.

Examples

1. Current file position immediately follows EOF2. Backspace four records to final position immediately following EOF1 (no error). Note that EOF2 is counted as a record:



```

DIMENSION IDCB3(144)
.
.
.
NUR=-4
CALL POSNT(IDCB3,IERR,NUR)
IF(IERR .LT. 0) GO TO 900
.
.
.

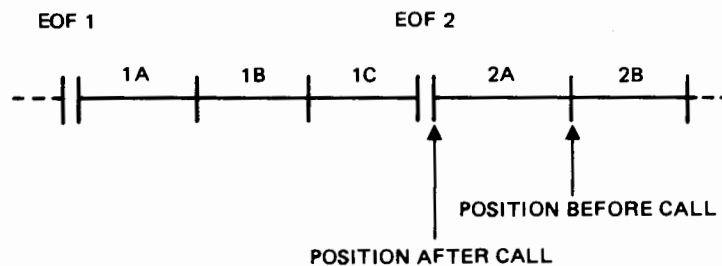
```

2. Current file position follows record 2A. Backspace five records resulting in a -12 error because the end-of-file was not the first record backspaced over. The call is terminated and the file is left positioned immediately after EOF2.

```

      DIMENSION IDCBS(144)
      .
      .
      .
      IERR=0
      NUR=-5
      CALL POSNT(IDCBS,IERR,NUR)
      IF(IERR .LT. 0) GO TO 900
      .
      .
      .
900   WRITE(1,910)IERR
910   FORMAT("POSITION ERROR = ",I5)

```



3. Position file with Data Control Block JDCB to record number 10:

```

      DIMENSION JDCB (144)
      .
      .
      .
      NUR=10
      IR=1
      CALL POSNT(JDCB,IERR,NUR,IR)
      IF(IERR .LT. 0) GO TO 900
      .
      .
      .

```

Sequence of Operations (POSNT)

1. Check for enough parameters.
2. Set reading flag.
3. If DCB is not open, reject call.
4. Compute the relative record number.
5. If type 0, go to 17.
6. If type 1 or 2, go to 14.
7. If forward spacing, call READF to read the required number of records (unless EOF or error).

8. If current position is EOF and EOF read bit is set, clear it and go to 9; else 10.
9. Decrement record number and backspace count; if done, exit.
10. Backspace one word and read it.
11. Backspace over the record.
12. Read length; if no match, error exit.
13. Go to 9.
14. Compute absolute record count; if less than 1, error exit.
15. Set record number.
16. Return.
17. If forward space, go to 7.
18. Check backspace legal flag.
19. Backspace — EXEC call.
20. If EOF encountered and not first backspace, go to 7 with 1 record (READF will return EOF).
21. If not done, go to 19.
22. Return.

3-19. RWNDF

This routine rewinds a magnetic tape or positions a disc file to the first record in the file.

Format

CALL RWNDF(IDCB,IERR)

Parameters

IDCB Data Control Block; array of 144+n words where n is positive or zero.

IERR Error return; optional 1-word variable in which error code is returned; may be omitted if A-register is to be checked for errors.

See paragraph 3-5 for further discussion of these parameters.

If the rewind cannot take place, the call is completed with the file position unchanged; no error is indicated. The rewind will not take place if, for instance, the file being rewound is a paper tape punch, the line printer, or some other device that cannot be positioned in reverse.

FMP Calls

Example

Position a disc file to the first record:

```
DIMENSION IDCB2(144)
*
*
*
CALL RWNDF(IDCB2,IERR)
IF(IERR .LT. 0) GO TO 900
*
*
*
```

Sequence of Operations (RWNDF)

1. If DCB is not open, reject call.
2. If type 0 file, do EXEC rewind request and return.
3. Write out current block if written on.
4. If not in main file, call D.RTR to get the main file address.
5. Reset current position pointers in DCB to beginning of file.

3-20. SPECIAL PURPOSE ROUTINES

FMP provides a set of routines each of which performs a special function not directly related to the standard input/output functions of defining, accessing, and positioning files. The routines with their associated functions are listed here and described in subsequent paragraphs in alphabetic order since they are not functionally related to one another.

Routine	Function
FCONT	Controls input and output of non-disc type 0 files; the functions are identical to those provided in the I/O Control EXEC call for RTE but here the device is identified by the file DCB, not a logical unit.
FSTAT	Returns the status of all cartridges in the FMP cartridge directory; includes the ID segment location of any program that has locked a cartridge.
IDCBS	Retrieves actual size of the DCB buffer for a currently open file; this is the usable buffer size allocated by FMP and determined by the total file size and the requested buffer size.
NAMF	Renames a created file; the file itself is not changed but it can no longer be opened or purged by the old name.
POST	Writes the contents of the DCB buffer to the disc; since FMP normally performs this function when the file is closed or the buffer is full, POST is useful mainly to enable modification of records in a file opened for non-exclusive use.

3-21. FCONT

This routine controls input/output functions on a peripheral device. The device must have been created and opened as a type 0 file. The call has no effect on other file types. It performs the same functions as the RTE I/O CONTROL call, such as to backspace, rewind, write end-of-file to magnetic tape, set the end of paper tape or generate paper tape leader, and control line spacing and top of form on the line printer.

Format

```
CALL FCONT(IDCB,IERR,ICON1,ICON2)
```

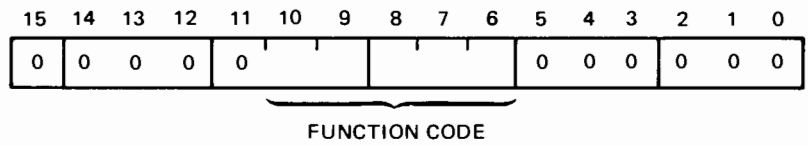
Parameters

IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.

ICON1	Function code; 1-word variable set to an octal code defining the function; see Table 3-6 for the codes.
ICON2	Line spacing if ICON1 = 11 (octal); 1-word variable set to output line spacing control code. File number if ICON1 = 27 (octal); 1-word variable set to file number on 2644 Cartridge Tape Unit minicartridge.
See paragraph 3-5 for further discussion of IDCB and IERR.	

Function Code

Bits 6 through 10 of parameter ICON1 are used for the function code.



The function codes are defined in Table 3-6.

Line Spacing

If the function code is 11 octal, then FCONT expects a value in ICON2. This value controls output line spacing on the line printer or a keyboard display device:

- 0 to suppress line spacing on the next line
- >0 to indicate the number of lines to space before the next line
- <0 to page eject on line printer; space specified lines on keyboard device.

File Number

If the function code is 27 octal, then FCONT expects a value in ICON2. This value declares the absolute file number to be located, in the range 1 through 255.

Table 3-6. FCONT Function Codes

FUNCTION CODE (OCTAL)	FUNCTION	DEVICE
00	Unused	
01	Write end-of-file	Magnetic tape/2644 Cartridge Tape Unit
02	Backspace one record	Magnetic tape/2644 Cartridge Tape Unit
03	Forward space one record	Magnetic tape/2644 Cartridge Tape Unit

Table 3-6. FCONT Function Codes (Continued)

FUNCTION CODE (OCTAL)	FUNCTION	DEVICE
04	Rewind	Magnetic tape/2644 Cartridge Tape Unit
05	Rewind standby Rewind	Magnetic tape 2644 Cartridge Tape Unit
06	Actual device status	Magnetic tape/2644 Cartridge Tape Unit
07	Set end-of-tape	Paper tape/TTY
10	Generate leader Write end-of-file if not just previously written or not at load point	Paper tape/TTY 2644 Cartridge Tape Unit
*11	List output line spacing	Line printer
12	Write 3 inch interrecord gap	Magnetic tape
13	Forward space on file	Magnetic tape/2644 Cartridge Tape Unit
14	Backspace one file	Magnetic tape/2644 Cartridge Tape Unit
15	Conditional top-of-form	Line printer/display device
20	Enable terminal — allows terminal to schedule its program when any key is struck	Codes 20-27 are defined for a keyboard terminal (DVR00). Refer to the DVR00 manual 29029-60001 for other uses.
21	Disable terminal — inhibits scheduling of terminal program	
22	Set time-out — sets new time-out interval	
23	Ignore all further requests until: <ul style="list-style-type: none"> • the request queue is empty • an input request is received • a restore control request is received 	
24	Restore output processing (this request is usually not needed)	
26	Write end-of-data	
** 27	Locate file number	

*When function code 11 is specified in ICON1, then ICON2 must be included in the parameter list to specify the particular line spacing.

**When function code 27 is specified in ICON1, then ICON2 must be included in the parameter list to specify the particular file number.

Examples

1. Backspace one record on file MT previously created and opened with Data Control Block MTDCB:

```

DIMENSION MTBUF(144)
.
.
.
ICONT=200B ← function code 2
CALL FCONT(MTBUF,IER,ICONT)
IF(IER .LT. 0) GO TO 900
.
.
.

```

2. Page eject to line printer created and opened with Data Control Block LPDCB:

```

DIMENSION LPDCB(144)
.
.
.
ICONT=1100B
LPSP=-1 ← page eject on line printer (would space 1 line on
keyboard device)
CALL FCONT(LPDCB,IERR,ICONT,LPSP)
IF(IERR .LT. 0) GO TO 900
.
.
.

```

Sequence of Operations (FCONT)

1. If DCB is not open, reject call.
2. Check if file type = 0.
3. Issue control EXEC call.
4. Return.

3-22. FSTAT

This routine returns the status of all mounted cartridges in the cartridge directory. The status information is essentially a copy of the cartridge directory and, for each cartridge, provides the logical unit number, the last FMP track, the cartridge reference number, and if a program has locked the cartridge, its ID segment.

Format

```
CALL FSTAT(ISTAT)
```

Parameters

ISTAT	Status; 125-word array in which the status of up to 31 cartridges is returned, using 4 words per cartridge; see Table 3-7.
-------	--

Table 3-7. ISTAT Format

ISTAT		
WORD	CONTENTS	CARTRIDGE
1	Logical unit number	First cartridge in directory
2	Last FMP track	
3	Cartridge reference number	
4	ID segment address of locking program or 0 (not locked)	
5	Logical unit number (or 0 if no more discs)	Second cartridge
6	Last FMP track	
7	Cartridge reference number	
8	ID segment address or 0	
9	Logical unit number (or 0 if no more discs)	Remaining cartridges (31 maximum)
.	.	
.	.	
125	0 (terminates list)	

NOTE

The same information can be obtained by the operator command :CL (cartridge list). Refer to paragraph 2-58 for description of this command.

Example

- Find the cartridge reference number of the cartridge associated with logical unit 13 in order to create file XX on that cartridge:

```

DIMENSION ISTAT(125),IDXX(144),NAMX(3),ISIZE(2)
DATA NAMX/2HXX,2H  ,2H  /,ISIZE/128/
CALL FSTAT(ISTAT)
ICR=-1                               set ICR to cartridge number of lu 13
DO 10 I=1,121,4
IF(ISTAT(I) .EQ. AHS(13)) ICR=ISTAT(I+2) ←
10 IF(ICR .LT. 0) GO TO 100 ← lu 13 not found
CALL CREAT(IDCB,IERR,NAMX,ISIZE,3,0,ICR)
IF(IERR .LT. 0) GO TO 150
.
.
.

```

Sequence of Operations (ISTAT)

- Read cartridge directory to ISTAT.
- Return.

3-23. IDCBS

This function returns the number of words in a Data Control Block actually used by the File Management Package for data transfer and file control.

Format

```
ISIZE=IDCBS(IDCBC)
```

Parameters

IDCB Data Control Block; array of 144+n words where n is positive or zero.

See paragraph 3-5 for a further discussion of IDCBC.

When a Data Control Block larger than 144 words is specified for the file at open or creation, the File Management Package may not use the entire DCB buffer area. The actual size used depends on the file size as well as the requested buffer size (refer to paragraph 3-5). This call returns the actual Data Control Block size; the buffer used plus 16 control words.

Example

1. A file has been opened using the Data Control Block MBUF with a size of 5000 words. Use IDCBS to determine how much of MBUF is being used by FMP. If 144 or more words remain, create KFIL using the remainder of MBUF as a Data Control Block:

```

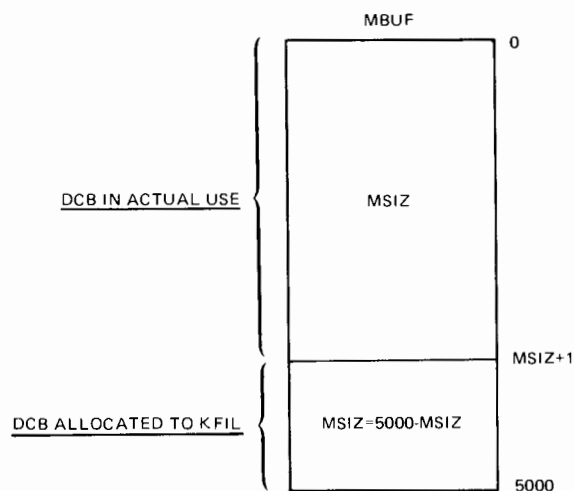
DIMENSION NAM(3),NAM1(3)
DIMENSION ISZ1(2),MBUF(5000)
DATA      NAM/2HMY,2HF1,2HLE/,NAM1/2HKF,2HIL,2HΔΔ/
ICR=- 14

CALL OPEN (MBUF,IERR,NAM,0,ICR,5000-144)
IF(IERR.LT.0) GOTO 150

ISZ1(1)= 5
ITYPE= 3
ISIZE= IDCBS(MBUF)
CALL CREAT(MBUF(ISIZE),IERR,NAM1,ISZ1,ITYPE,0,ICR,5000-ISIZE)

IF(IERR.LT.0) GOTO 150

```



Sequence of Operations (IDCB)

1. Determine if DCB is open.
2. Extract the DCB size.
3. Return.

3-24. NAMF

This routine renames an existing file. If the code was created with a security code, this code must be specified. If the file is open, it is closed and then renamed.

Format

CALL NAMF(IDCB,IERR,NAME,NNAME,ISECU,ICR)

Parameters

IDCB	Data Block; array of size 144 +n where n is positive or zero.
IERR	Error return; 1-word variable in which negative error code is returned.
NAME	File name; 3-word array containing ASCII file name.
NNAME	New file name; 3-word array containing ASCII file name to replace NAME as the file name.
ISECU	Security code; optional 1-word variable in range 0 through ± 32767 ; omitted only if file NAME was created without a security code or a zero code; if specified, the code must match.
ICR	Cartridge reference; optional 1-word variable in range 0 through 32767; if zero or omitted, the first file found with given NAME and matching security code is renamed; if specified, only a file on the specified cartridge is renamed.

Refer to paragraphs 3-5, 3-6 for further discussion of IDCB, IERR, NAME, ISECU, ICR.

Example

Rename file PROG1 as MYFILE. Since a cartridge reference is specified, the system will look for PROG1 on logical unit 14 only. No security code is needed:

FMP Calls

```
DIMENSION IDCB2(144),NAME(3),NNAME(3)
DATA NAME/2HPR,2HOG,2H1 /,NNAME/2HMY,2HFI,2HLE/
ICR=-14
CALL NAMF(IDCB2,IERR,NAME,NNAME,0,ICR)
IF(IERR .LT. 0) GO TO 900
.
.
.
```

Sequence of Operations (NAMF)

1. Check for enough parameters.
2. Check legality of *nuname*.
3. Call OPEN to open the file exclusively.
4. If OPEN errors, exit.
5. Check file security flag in DCB; if mismatch, close and exit.
6. Get a system track.
7. Write new name on track.
8. Pass track to D.RTR; rename request.
9. Return system track.
10. Close the DCB.
11. Check for D.RTR errors.
12. Return.

3-25. POST

This routine is called to post (write) the contents of the Data Control Block buffer to a disc file (type 2 or above). Normally, this is done by the system when the buffer is full or the file is closed. POST provides direct control over the physical write to disc, assures that the next read is from disc, and can be used in a special case to save records in a file opened for non-exclusive use.

Format

```
CALL POST(IDCB,IERR)
```

Parameters

IDCB	Data Control Block; array of 144+n words where n is positive or zero.
IERR	Error return; optional 1-word variable in which negative error code is returned; should be omitted only if A-register is tested for errors.

Refer to paragraph 3-5 for further discussion of IDCB and IERR.

This call is ignored for all files of type 0 or 1 since transfers to these files are always direct, bypassing the Data Control Block buffer. (Refer to paragraph 3-3 for a discussion of data transfer.)

Using POST For Modification

In conjunction with the RTE Resource Numbering call RNRQ, POST allows several programs to modify a file without requiring an exclusive open. (Refer to RTE Operating System Manual for description of RNRQ.) RNRQ is used to lock the file for exclusive use of the calling program, and POST then clears the DCB buffer before modifying the record and again after modification.

The sequence to be followed is:

1. Open the file.
2. Read the file to retrieve the resource number (RN).
3. Call POST to clear the DCB buffer (no data is posted since none was written).
4. Call RNRQ to lock the file for exclusive use of the calling program.
5. Call READF to read the record to be modified.
6. Modify the record and call WRITF to write the record.
7. Call POST to transfer the updated record to the file from the DCB buffer.
8. Call RNRQ to unlock the file for use by other programs.

It is possible that WRITF in step 6 above causes the buffer to be posted to the disc, but POST should be called to insure the transfer.

Example

Assume the resource number is in location IRN; modify record number 5 in a type 2 file opened non-exclusively:

```

30  DIMENSION IDCB(144),IBUF(128)
    CALL POST(IDCB,IERR) ← clear DCB buffer
    IF(IERR .LT. 0) GO TO 150 ← set code to local lock
    ICODE=18 ← lock file for exclusive use
40  CALL RNRQ(ICODE,IRN,ISTAT) ← of program
    IF(ISTAT .NE. 2) GO TO 150 ← error test
50  IL=80
    CALL READF(IDCB,IERR,IBUF,IL,LEN,5) ← read record 5
    IF(IERR .LT. 0) GO TO 150
    IF(LEN .GT. 80) GO TO 150
    *
    * ← modify record in IBUF
    *
60  CALL WRITF(IDCB,IERR,IBUF,IL,5) ← write modified record
    IF(IERR .LT. 0) GO TO 150
70  CALL POST(IDCB,IERR) ← clear buffer again
    IF(IERR .LT. 0) GO TO 150
80  ICODE=48 ← set code to unlock file

```

```
CALL RNRQ(ICODE,IRN,ISTAT)
  •
  •
150  • process errors here and unlock Resource Number
      •
      •
      •
```

Sequence of Operations (POST)

1. Determine that the DCB is open.
2. If the DCB was written into, post it (write it on the disc).
3. Clear the in-core flags and the written-on flags.
4. Return.

3-26. EXAMPLE USING FMP CALLS

The following short sample program creates up to ten data files from one control file. It then reads and checks the data in each data file against control information in the control file.

Program FMPEX

```

C
C   PROGRAM FMPEX,3,99
C
C   DIMENSION LU(5),IBUF(40),IPBUF(33),IHDR(40),IREC(10)
C   DIMENSION IRB(10),IOFF(10),IOPEN(10),ILOCF(10),ISCHK(16),IREG(2)
C   DIMENSION IOCB(144),JOCB(1440),DATA(5)
C
C   EQUIVALENCE (IA,IREG),(IB,IREG(2)),(X,IREG)
C   EQUIVALENCE (FNAM,IPBUF(2)),(ISECU,IPBUF(6)),(ICR,IPBUF(10))
C   EQUIVALENCE (ITYPE,IPBUF(14)),(ISIZE,IPBUF(18)),(IRLNT,IPBUF(22))
C
C   DATA ISCHK/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15/
C
C   THIS PROGRAM IS INTENDED TO MANAGE THE DATA-FILE SCHEME FOR A
C   DATA ACQUISITION SYSTEM. NOT INCLUDED HERE IS THE
C   CODE THAT ACTUALLY OBTAINS THE DATA, SINCE THIS IS
C   SIMPLY AN EXAMPLE OF FILE USAGE.
C
C   THE OPERATION OF THIS ROUTINE FOLLOWS:
C
C   A TYPE 4 'CONTROL' FILE IS CREATED BY THE TEST OPERATOR THAT
C   CONTAINS INFORMATION ABOUT THE TESTS TO BE PERFORMED AND THE
C   FILES INTO WHICH THE DATA MUST GO. THIS ROUTINE REQUESTS
C   THE NAME OF THE FILE, OPENS IT, AND USES INFORMATION
C   IN IT TO CREATE AND USE OTHER DATA FILES.
C
C   THE FORMAT OF THIS CONTROL FILE IS:
C
C   RECORD #      CONTENTS
C
C   1             TEST LABEL : A HEADER FOR DATA FILES
C   2             'NAMR' OF DATA FILE #1
C   3             'NAMR' OF DATA FILE #2
C   4             ETC.
C               END WITH "/E" AS FILE NAMR
C   N             OPERATOR SECURITY CODE (E.G. FOR TEST ACCESS)
C               AS CHECKED AGAINST THE 'ISCHK' VALUES, ABOVE
C
C   NOTE: ANY LINE BEGINNING WITH "**" IS CONSIDERED A COMMENT
C
C   THE DATA, AS IT COMES IN, HAS ASSOCIATED WITH IT A
C   'DATA CODE' INDICATING THE DISPOSITION OF THE DATA. THE
C   CODES ARE LISTED BELOW:
C
C   CODE      FUNCTION
C   0         DATA O.K.   WRITE IT TO THE APPROPRIATE FILE

```



```

C          1      THIS DATA IS TO BE IGNORED
C          2      IGNORE THE PREVIOUS DATA ITEM (ERASE IT FROM THE FI
C          3      REMEMBER THE FILE POINTERS, WE MAY WANT TO GO BACK
C          4      REWIND THE DATA FILE - START OVER AGAIN
C          5      GO BACK TO 'REMEMBERED' POINT IN DATA FILE
C          6          *
C          7          *      SAME AS 0, ABOVE
C          8          *
C          9          *
C         10      END OF TEST - CLOSE OF THE DATA FILE

```

NOTE: DATA IS WRITTEN INTO THE FILES IN ASCII

SCHEDULING PARAMETERS: RU,FMPEX,LU
WHERE LU IS A TERMINAL'S LOGICAL UNIT

```

C
C 1) GET SCHEDULING PARAMETER AND SET UP TERMINAL LU
C
C      CALL RMPAR(LU)
C      IF(LU,EQ,0)LU=1
C      ILU=LU+4000
C
C 2) ASK OPERATOR FOR CONTROL FILE NAME
C      USE ROUTINE 'COLON' TO REPLACE COLONS IN FILE 'NAMR'
C      WITH COMMAS SO WE CAN USE SYSTEM PARSE ROUTINE 'PARSE'
C
C 10  WRITE(LU,100)
C 100  FORMAT("/FMPEX: ENTER CONTROL FILE 'NAMR': ←")
C      X=REIO(1,ILU,IBUF,20)
C      CALL COLON(IBUF,10)
C      CALL PARSE(IBUF,10*2,IPBUF)
C      CALL OPEN(IDCb,IERR,FNAM,0,ISECU,ICR)
C      IF(IERR,GE,0)GO TO 12
C      WRITE(LU,101)IERR
C 101  FORMAT("/FMPEX: FILE ERROR"16".  TRY AGAIN!"/)
C      GO TO 10
C
C 3) READ CONTROL FILE AND:
C      CREATE EACH DATA FILE
C      WRITE DATA FILE HEADER FROM CONTROL FILE
C      CHECK OPERATOR SECURITY CODE
C
C 12  N=1
C      CALL READF(IDCb,IERR,IBUF,20)
C      CALL READF(IDCb,IERR,IBUF,20)
C 14  CALL READF(IDCb,IERR,IBUF,16,IB)
C      IF(IERR,LT,0)GO TO 80
C      IF(IB,NE,-1)GO TO 13
C      WRITE(LU,112)
C 112  FORMAT("/FMPEX: FOUND EOF IN CONTROL FILE.  ERROR!")
C      GO TO 80
C
C 13  IF(IBUF,EQ,2H**)GO TO 14

```

```

IF(IBUF,EQ,2H/E)GO TO 16
IF(N,GT,10)GO TO 14
C
CALL COLON(IBUF,18)
CALL PARSE(IBUF,18*2,IPBUF)
IPBUF(19)=IRLNT
CALL CREAT(JDCB(144*(N-1)+1),IERR,FNAM,ISIZE,4,ISECU,ICR)
IF(IERR,LT,0)GO TO 80
N=N+1
GO TO 14
C
16 N=N-1
CALL READF(IDC8,IERR,IBUF,20,IL)
IBUF(IL+1)=2H,,
CALL CODE
READ(IBUF,*)ISECU
DO 17 I=1,16
  IF(ISECU,EQ,ISCHK(I))GO TO 18
17 CONTINUE
WRITE(LU,117)ISECU
117 FORMAT("/FMPEX: "I6" IS NOT A VALID OPERATOR SECURITY CODE!")
GO TO 82
C
18 CALL RWNDF(IDC8,IERR)
IF(IERR,LT,0)GO TO 80
CALL READF(IDC8,IERR,IHEADR,40,IB)
IF(IERR,LT,0)GO TO 80
DO 19 I=1,N
  CALL WRITF(JDCB(144*(I-1)+1),IERR,IHEADR,IB)
  IF(IERR,LT,0)GO TO 80
19 CONTINUE
C
C
C 4) THIS SECTION WOULD CONTAIN SOME DATA ACQUISYTION ROUTINES,
C FOR THE PURPOSE OF THIS EXAMPLE WE WILL SIMPLY INPUT SOME
C DATA FROM A TELETYPE.
C
C THE DATA WILL CONSIST OF 7 ITEMS:
C ITEM VALUE
C 1 DATA FILE TO BE USED FOR STORAGE
C 2 DISPOSITION FLAG, AS DISCUSSED ABOVE
C 3 THRU 7 DATA VALUES
C
C IN A MORE GENERALIZED CASE, THIS SECTION WOULD ALSO
C INCLUDE DATA PROCESSING OR CORRECTION, DATA CHECKING, AND
C DETERMINATION OF DISPOSITION FLAG VALUE.
C
C
C 20 WRITE(LU,120)
120 FORMAT("/FMPEX: ENTER FILE #,FLAG, AND 5 DATA ITEMS: +")
READ(LU,*)I,IFLG,(DATA(J),J=1,5)
C
C CHECK DISPOSITION FOR 'IGNORE', ERROR IN FLAG, OR END OF TEST
C
IF(IFLG,EQ,1)GO TO 20
IF(IFLG,EQ,10)GO TO 30

```

FMP Calls

```

      IF((IFLG.LT.1).OR.(IFLG.GT.10))GO TO 38
C
C RE=FORMAT THE DATA TO THE FORMAT OF THE DATA FILE
C
      CALL CODE
      WRITE(IBUF,122)I,IFLG,(DATA(J),J=1,5)
122  FORMAT(I4,"I4,5(",",F8.2))
C
C DATA IS MERELY WRITTEN TO THE FILE FOR IFLG=0,6-9
C
      IF((I.NE.0).AND.(I.LE.5))GO TO 22
21  CALL WRITF(JDCB(144*(I-1)+1),IERR,IBUF,30)
      IF(IERR.LT.0)GO TO 80
      GO TO 20
C
C WE BACKSPACE THE FILE TO OVER=WRITE PREVIOUS DATA IF 'IFLG' = 2
C
22  IF(I.NE.2)GO TO 24
      CALL POSNT(JDCB(144*(I-1)+1),IERR,-1)
      IF(IERR.LT.0)GO TO 80
      GO TO 21
C
C 'REMEMBER' THE FILE POINTERS BEFORE WRITING IF 'IFLG' = 3
C A 10=WORD ARRAY IS USED IN ORDER TO MAINTAIN ONE SET OF POINTERS
C FOR EACH DATA FILE
C
24  IF(I.NE.3)GO TO 26
      CALL LUCF(JDCB(144*(I-1)+1),IERR,IREC(I),IRB(I),IOFF(I))
      IF(IERR.LT.0)GO TO 80
      GO TO 21
C
C IF 'IFLG' = 4, WE START THE DATA FILE ALL OVER AGAIN,.. REWIND
C
26  IF(I.NE.4)GO TO 28
      CALL RWNDF(JDCB(144*(I-1)+1),IERR)
      IF(IERR.LT.0)GO TO 80
      GO TO 21
C
C THAT DATA TAKEN SINCE WE NOTED THE FILE POINTERS IS NOW DETERMINED
C TO BE OF NO VALUE. RESTORE THE DATA FILE TO THAT POSITION AND
C OVERWRITE OLD DATA WITH NEW.
C
28  IF(IERR.NE.5)GO TO 35
      CALL APOSN(JDCB(144*(I-1)+1),IERR,IREC(I),IRB(I),IOFF(I))
      IF(IERR.LT.0)GO TO 80
      GO TO 21
C
30  CALL CLOSE(JDCB(144*(I-1)+1),IERR)
      IF(IERR.LT.0)GO TO 80
      N=N-1
      IF(N.EQ.0)90,20
C
35  WRITE(LU,135)IFLG
135  FORMAT("/FMPEX: "I6" IS AN ILLEGAL DISPOSITION FLAG!")
      GO TO 20

```

```

C
38  WRITE(LU,138)I
138  FORMAT("/FMPEX: "I6" IS AN ILLEGAL DATA FILE NUMBER!")
    GO TO 20
C
C
C  FILE ERROR SECTION
C
80  WRITE(LU,180)IERR
180  FORMAT("/FMPEX: FILE ERROR" I6", ABORT!")
    DO 82 I=1,6
      CALL CLOSE(JDCB(144*(I-1)+1),IERR)
82  CONTINUE
C
    CALL CLOSE(IDC B,IERR)
C
C
C  END
C
90  WRITE(LU,190)
190  FORMAT("/FMPEX: DONE!"/)
C
    END
    ENDS

```

Subroutine COLON

COLON is used to convert the colons (:) in *namr* to commas for the PARSE routine. PARSE is in the RTE system library.

```

C
    SUBROUTINE COLON(IBUF,N)
C
    DIMENSION IBUF(2)
C
    THIS SUBROUTINE ACCEPTS A BUFFER CONTAINING AN ASCII STRING
C  (USUALLY AN FMP 'NAMR') AND CONVERTS ALL COLONS (:) TO COMMAS (,)
C  THIS ALLOWS THE USE OF THE SYSTEM PARSE ROUTINE (WHICH REQUIRES
C  COMMAS AS DELIMITERS) TO PARSE OUT FMP 'NAMR'S WHICH USE COLONS
C  AS DELIMITERS
C
C
C
    DO 10 I=1,N
      IF( IAND( IBUF(I),77600B),NE,35000B)GO TO 5
      IBUF(I)=IOR( IAND( IBUF(I),177B),26000B)
5     IF( IAND( IBUF(I),177B),NE,72B)GO TO 10
      IBUF(I)=IOR( IAND( IBUF(I),77600B),54B)
10    CONTINUE
C
    RETURN
C
    END
    ENDS

```


SECTION IV

USING THE SPOOL MONITOR



INDEX TO SPOOL-RELATED FMGR COMMANDS

Command Syntax	Function	Page
<code>:CS,lu</code> <div style="display: inline-block; vertical-align: middle; border-left: 1px solid black; border-right: 1px solid black; border-bottom: 1px solid black; padding: 0 5px; margin-left: 10px;"> ,EN ,RW ,PU ,SA ,PA ,NB ,BU ,NP[,outlu[,priority]] </div>	Change spool set-up attributes.	4-14
<code>:EO</code>	Indicate end-of-job.	4-10
<code>:JO[,name[:hr:min:sec]][,priority [spool priority[,NS]]]</code>	Indicate start-of-job.	4-9
<code>:LU,lu[,namr]</code>	Associate logical unit with spool file (input only).	4-10
<code>:LU,lu[,namr</code> <div style="display: inline-block; vertical-align: middle; border-left: 1px solid black; border-right: 1px solid black; border-bottom: 1px solid black; padding: 0 5px; margin-left: 10px;"> ,HO ,WR ,BO ,WH [,outlu[,priority]] ,BU ,PU ,ST </div>	Associate logical unit with spool file, specify outspool attributes, logical unit, priority (out- put or input/output).	4-10

PROGRAM JOB COMMAND

Schedule JOB from RTE:	<code>*RU,JOB,lu</code> <code>*RU,JOB,fi,le,nm[,priority]</code>	Job or jobs input from device. Job entered from file.
Command Syntax	Function	Page
<code>:XE,namr[,priority]</code>	Specify file or logical unit containing job input to JOB.	4-21

See Section V for Index to GASP Commands.

USING THE SPOOL MONITOR

SECTION

IV

4-1. INTRODUCTION

The Spool Monitor is an optional segment of the complete Batch-Spool Monitor. It operates in conjunction with the File Management Package to provide batch job processing with spooling or, through user program calls, to provide spooling without batch processing. Spooling means that jobs or data are placed on disc files for input and data is sent to disc for output. This allows input and output to be performed independently of each other and of job processing. Spooling allows jobs to be processed without having to wait for completion of input from or output to slow devices. The entire spool process can proceed automatically with virtually no user intervention, or it may be directly controlled during its various phases.

Spooling can be used to increase the throughput of a job stream that is limited by the idle time of slow peripheral devices. It does this by allowing programs to perform I/O to disc files rather than to the slower peripheral devices. The system then manages the I/O between the disc files and the peripheral devices to assure that all I/O reaches its proper destination.

For example, assume there are 10 compute-bound programs that output one line to the line printer approximately once each second. One way to avoid an interleaved listing would be to have the programs each request an LU lock on the printer during its entire execution. The program that requested the lock first would completely shut out the other programs during its entire computation and output. When the first program finished, the next program to receive the lock would again shut out the other programs. This process would continue until all 10 programs have run to completion. This method wastes system resources since only the line printer or the CPU is operating at any one time.

Under spooling, both the line printer and the CPU are used more efficiently. The programs output their listings to disc files which can be done much faster. Additionally, when one of the programs is suspended for a disc write, another program may use the CPU for its computations and then perform its output to its own disc file. The system handles the task of outputting the disc file to the line-printer. Since the disc files are already built up, they can be output as quickly as the line printer can accept them, thereby eliminating the line printer's idle time.

The Spool Monitor Package provides the following capabilities:

- Opens and closes the disc files known as spool files; after close, optionally writes the file contents to a user-selected non-disc device for output.
- Keeps a record of the current status of all jobs and spool files in the system.
- Translates non-disc device references in program I/O calls to references to spool files.

These capabilities can be divided into two distinct functions: spooled batch processing and spooled input/output or non-batch spooling. Spooled batch processing is described in this section and Section V. Non-batch spooling is performed with the SMP program calls described in Section VI.

Spooling is normally an automatic process, however, direct intervention in the batch spooling process is possible with the spool operator commands under control of program GASP. The GASP commands are described in Section V.

4-2. BATCH SPOOLING

Batch processing without spooling is performed by FMGR as described in Section II. Batch processing with spooling is set up when FMGR encounters a :JO command that has been inspoiled by program JOB. Batch processing with spooling involves three phases:

- Inspooling the job
- Processing the job
- Outspooling data produced by job processing

Figure 4-1 is a diagram of these three phases.

① Inspooling

Spooling of batch jobs is initiated by running program JOB from RTE. This program controls the phase known as inspooling. During this phase, JOB makes an entry for each job in the inspool directory file JOBFIL (refer to Appendix C for JOBFIL format). Job priority and status information is maintained in JOBFIL. Jobs entered from non-disc devices are written to disc spool files. If any jobs are already stored on disc, they are not transferred, but an entry is made in JOBFIL and the disc file is treated as a spool file.

② Job Processing

Program JOB schedules FMGR to process jobs as soon as the first job is inspoiled. If FMGR is already scheduled interactively, it automatically checks JOBFIL for jobs to process each time it is terminated by the EX command. Each job is processed in the order of priority recorded in JOBFIL.

NOTE

FMGR, not a copy such as FMG07, is the only program that can process batch jobs. FMGR termination is expedited when JOBFIL is on the first cartridge checked in the FMP cartridge directory.

③ Outspooling

After processing, output from each processed job is written to disc and an entry made for the output in the outspool directory file, SPLCON (refer to Appendix C for SPLCON format). Spooled output is controlled by program SMP. SMP assigns and monitors outspool files, maintains the outspool directory, SPLCON, and monitors the program SPOUT. Program SPOUT, scheduled by SMP, takes job output from the outspool files and directs the output to the actual devices.

Except for JOB, which you must request with the RUN command, each of these programs is scheduled internally as needed.

Program GASP

During batch spooling, you may directly intervene in the inspool and outspool processes with the GASP operator commands (refer to Section V). In particular, these commands are useful to change priorities or the current status of spooled jobs or files.

GASP is also used to initialize the Spool Monitor. You must run GASP once in order to initialize the spool directory file JOBFIL, the outspool directory SPLCON, and the spool files that constitute the pool of disc files used by the Spool Monitor. The process of configuring the Spool Monitor and initializing it with GASP is described in Section VII.

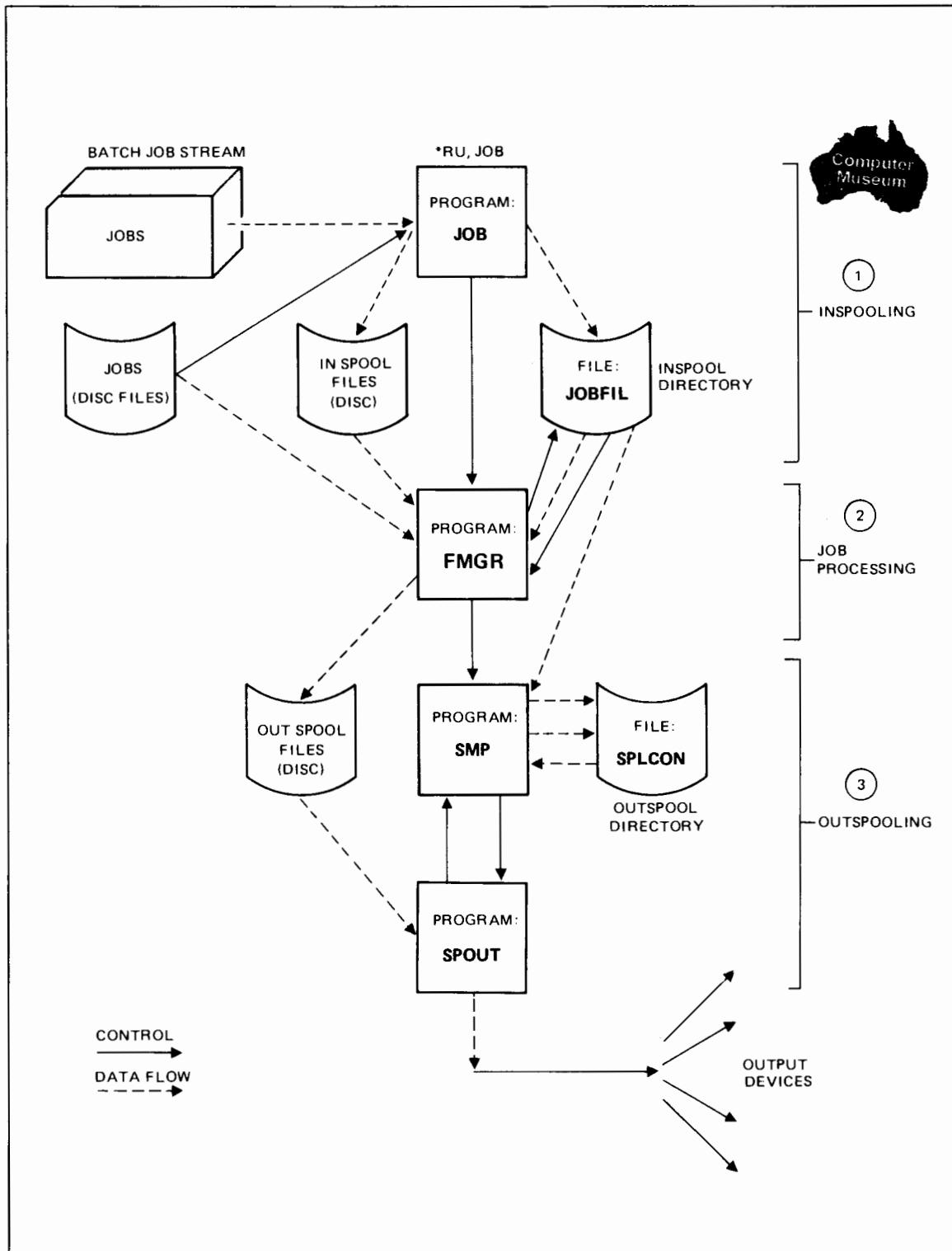


Figure 4-1. Batch Spooling Diagram

Non-Batch Spooling

Data to be input or output can be spooled to or from disc files using the SMP program calls described in Section VI. These calls set up spool files for I/O and control the spooled I/O. The routines permit calls that specify non-disc I/O devices to be actually referred to disc.

4-3. TIMING CONSIDERATIONS

The spool system transfers to or from the disc at a maximum rate of approximately 5K words per second. This rate is much faster than that of most other devices and thus speeds job throughput since time limits for final output are usually imposed by the slower devices. What spooling does is decrease the amount of time a slower device is idle by keeping all output on disc ready to be sent to that device. Maximum gain in the speed of job processing with spooling can be directly equated to the amount of time an output device remains idle waiting for job output when spooling is not used.

The Spool Monitor will, in general, contend with other users for the disc, thus forcing those users to run more slowly. This can be minimized by:

- using a separate disc for spooling
- shutting down the spool system during critical periods
- not using spooling

4-4. SPOOL FILES

The disc files used by the Spool Monitor are called spool files. In general, these files are taken from a pool of up to 80 spool files that are created during initialization of the spool system, by GASP. These files are controlled by the Spool Monitor and are given names of the form SPOL nn , where nn is a number between 01 and 80. The files always start with SPOL01 and are numbered consecutively up to the number requested at initialization. They are known as *spool pool files*.

Besides the spool pool files, any disc file (type 3 or above) can be used as a spool file. To specify an existing file as a spool file you must define it as a spool file with the FMGR LU command (paragraph 4-9) or with the spool setup EXEC call (paragraph 6-3). Such files can be saved after use by the Spool Monitor. Spool pool files, on the other hand, are not saved, but are returned to the pool of spool files after being closed.

Data in spool files has one of two formats: spool file or standard. In either case, a spool file must be a variable-length record file (type 3 or above).

Spool File Format

Spool files used for outspooling are written by the Spool Monitor in spool file format. This format has a special two-word header attached to and preceding each record in the file. This header preserves the I/O control information specified in the original I/O call. It consists of:

- word 1 Control word containing function and subfunction from the original I/O call.
- word 2 Length word or extra control word; length word contains the record length in words (positive) or characters (negative) as given in the original I/O call. If the call is a control request, the extra control word is stored here.

The rest of the record contains the data to be written to a device.

Standard Format

Spool Files that are not used for outspooling or that do not require I/O control information have no header. Such files are written exactly as presented.

NOTE

Since spooling is automatic, jobs may be easily spooled without thorough understanding of the spool system. You may wish to turn directly to paragraph 4-6, Spool Setup, and skip paragraph 4-5 that describes how the system works, not how to use it.

4-5. LOGICAL UNIT SWITCHING

The spool system is structured so that all programs should be able to run with no change in code whether they are spooled or not. Since spooling is a form of disc buffering, normal I/O calls and statements must be transformed to disc I/O calls that require track/sector addresses to result in information compatible with FMP files. User calls, normally directed to I/O devices such as the line printer (LU 6) or the punch (LU 4) are re-directed to a logical unit referencing the spool driver DVS43. This re-direction is done through the Logical Unit Switch Table of the RTE system. The transition from non-disc to disc I/O is made by the spool driver DVS43.

Figure 4-2 illustrates a call to LU 6 that references driver DVR12 when it is not spooled, but if spooled, references DVS43 and ultimately the disc driver DVR31.

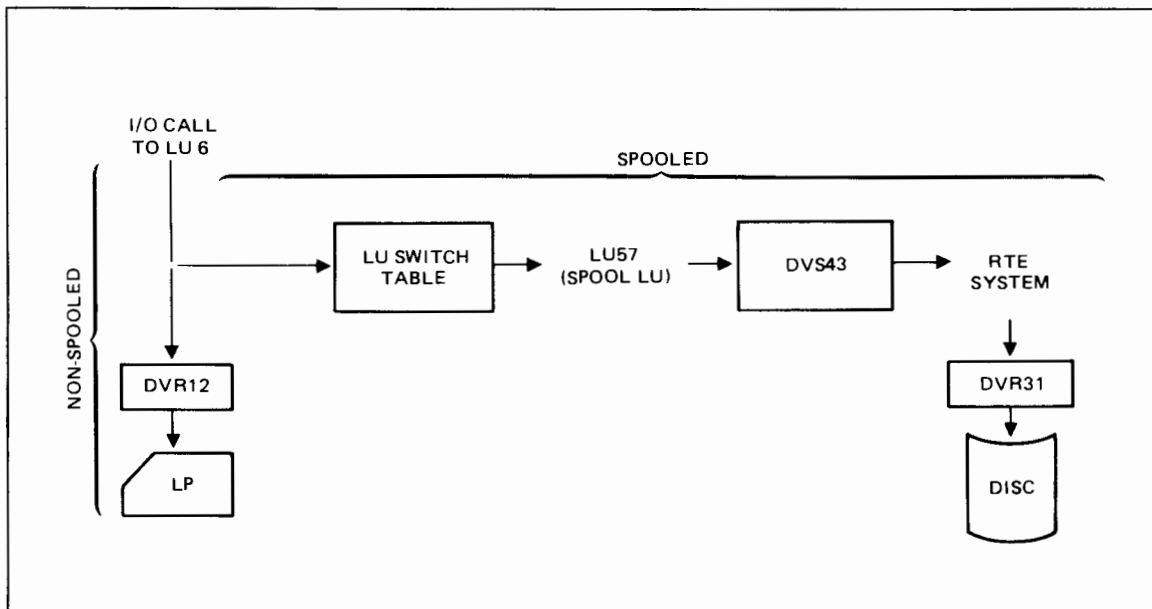


Figure 4-2. Logical Unit Switching

The process of setting up the Logical Unit Switch Table and associating disc I/O with a specific file is called "spool setup". Spool setup is requested by the user through the LU command in batch spooling or a spool setup call for non-batch spooling. The process itself, involving the Spool Monitor Program (SMP), the RTE system itself, the FMGR program as batch processor, and the D.RTR file directory management program, requires no user intervention. The aim of spool setup is to open a spool file to the spool system.

Spool Monitor

Notice that spool setup effectively associates a logical unit with a disc file. Recall that device logical units are linked to Equipment Table (EQT) entries through the Device Reference Table (DRT) in the RTE system (refer to the RTE Programming and Operating manual). The Equipment Table entry in turn specifies the I/O select code and I/O driver associated with the device. This association is illustrated in Figure 4-3.

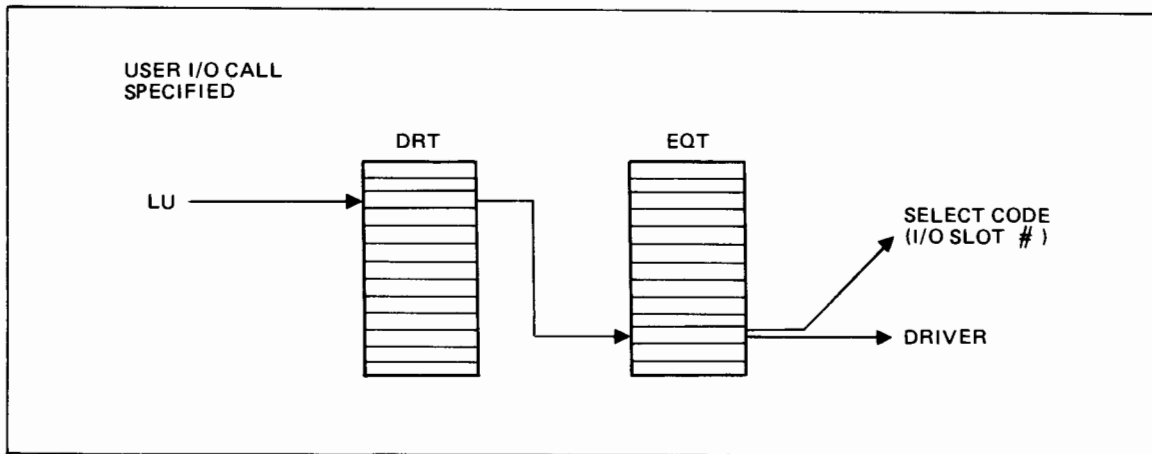


Figure 4-3. Association of Logical Unit to Driver

In order to implement spool setup, an Equipment Table entry must reference the spool driver DVS43 and point to a “dummy” select code (10B through 77B) not used by a real device. An entry in the Device Reference Table must link a logical unit number, the “spool logical unit” to a “spool EQT entry”. These spool entries are all established at system generation time in the same way that EQT and DRT entries are established for standard I/O devices (refer to Section VII).

Spool Logical Units

The spool system requires, at the very least, four spool EQT entries and their corresponding spool logical units in order to function. One logical unit is needed for inspooling by JOB, two for the standard input and output devices used during processing, and one (for SPOUT) for each spooled output device expected to be active at one time. The standard input logical unit, LU 5, and the standard output logical unit, LU 6, are always equated to spool logical units. In addition, you will need one spool logical unit for each LU command expected in a job. In order to use the spool system effectively, a minimum of six spool EQT entries and spool logical units should be allocated at system generation. For each spool logical unit, at least one spool file should be requested during spool initialization with GASP (see Section VII).

Example of Batch Logical Unit Switching

Figure 4-4 illustrates a possible set of entries in the Logical Unit Switch Table, the Device Reference Table (DRT) and the EQT table. The first four entries in the Logical Unit Switch table connect the logical units referenced in a program call or operator command to the spool logical units in the Device Reference Table. The last six entries in the Device Reference Table equate logical units 13 through 18 to the EQT entries for DVS43. The last six entries in the EQT table are associated with DVS43, the special spool driver that references spool files rather than devices.

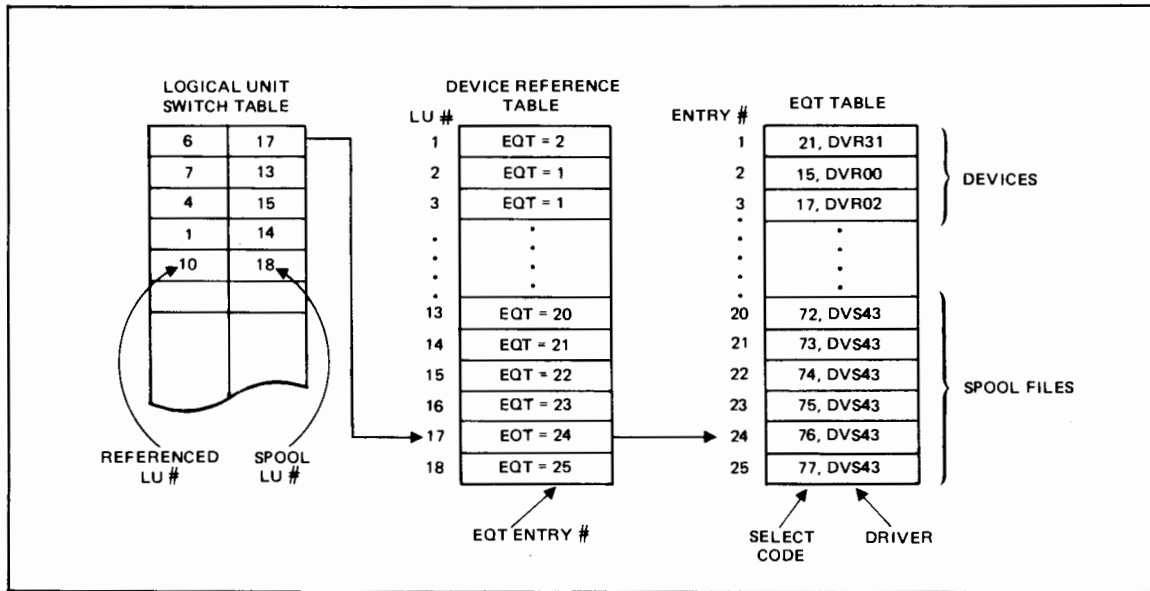
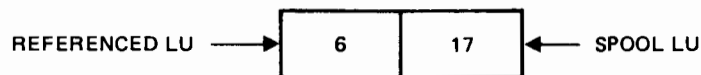
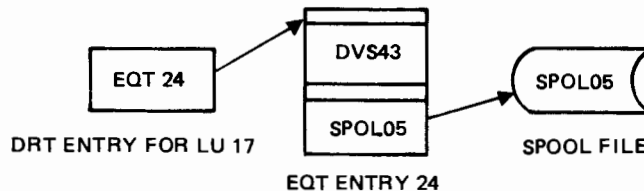


Figure 4-4. Relation between LU Switch Table, DRT, and EQT

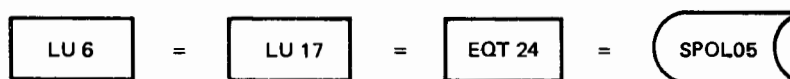
Suppose a reference is made to logical unit 6. In the LU Switch table shown in Figure 4-4, logical unit 6 is associated with spool logical unit 17:



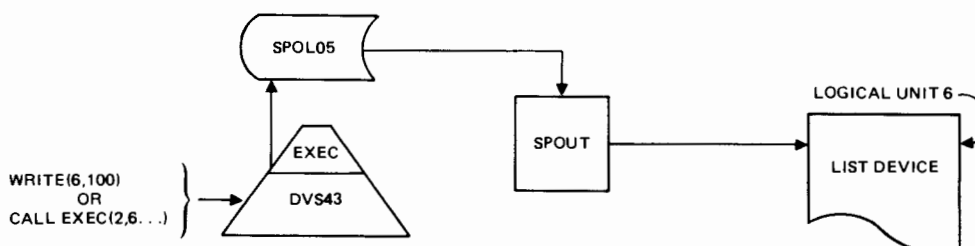
In the Device Reference Table, spool logical unit 17 is associated with EQT entry 24. This entry for the spool driver DVS43 is assigned the next available spool pool file (SPOL05).



In this way, logical unit 6 is associated with spool file SPOL05:



When a standard I/O EXEC call or a program I/O statement writes output to logical unit 6, the data is written to SPOL05. Program SPOUT subsequently gets this data from SPOL05 and writes to logical unit 6, the actual list device:



4-6. SPOOL SETUP

The jobs to be spooled, like the batch jobs described in Section II (2-47), must be preceded by a :JO command and terminated by an :EO command. Additional housekeeping functions are automatically performed by the spool system at the start and termination of each job. The :JO command can also be used to set priorities for job inspooling and outspooling, and to bypass automatic outspooling so that job output may be controlled directly.

If any spool logical units are required by the job in addition to the standard input logical unit 5 and output logical unit 6, these should be specified in an :LU command within the job. :LU is used to specify which spool file to reference and how outspooling is to be performed. For instance, if you want to hold outspooling until another task is completed, this can be specified; or if you want the outspool file buffered or purged upon completion, this too may be specified with :LU. An actual output device other than logical unit 6 can be requested with :LU. If an outspool priority is needed that is different from the job priority, this too can be specified.

NOTE

The standard logical units for input and output, LU 5 and LU 6, are automatically set up for spooling. LU 5 must not be specified in an :LU command.

Any of the outspool attributes specified in an :LU command can be changed with a subsequent :CS command. This command would be useful, for example, to remove a hold on an outspool file that had been set by :LU.

Use of the commands :LU and :CS to set up additional spools is restricted to use within spooled batch jobs.

The spool calls described in Section VI may be used within programs to perform programmatically many of the outspool functions of the :LU and :CS commands. These calls allow a program that is not part of a batch job delimited by :JO and :EO to use the spooling capabilities of the Spool Monitor. In this way, a program can send its standard output to a disc file even if it is limited by the programming language to non-disc logical units.

Priority Assignment

Jobs entered in the spool system for spooling may have a priority. If they don't, the lowest priority, 9999, is assigned to the job. A separate priority can be assigned for outspooling, but if not assigned, the job priority is used as the outspool priority.

Job priority may be assigned by:

- the FMGR spool :JO command
- the program JOB during inspooling if the job is on a disc file
- the JOB command :XE during inspooling if the job is on a disc file.

Outspool priority is the same as job priority unless specifically assigned by:

- the FMGR spool :JO command
- the FMGR spool :LU command

Or changed by:

- the FMGR spool :CS command.

4-7. JO — INITIATE JOB FOR SPOOLING

The :JO command defines the beginning of a job, optionally names the job, and defines parameters for the job. It is the same command described in paragraph 2-48 with additional parameters used only if the job has been inspoiled by program JOB. When FMGR processes a :JO command that has been inspoiled by program JOB, a batch environment with spooling is set up.

Format

```
:JOB[name[:hr:min:sec][,priority[,spool priority [HO
'NS ]]]]
```

Parameters

<i>name</i>	Job name; 6-character name specified according to file name conventions (paragraph 2-8).
<i>:hr:min:sec</i>	CPU time limit for job in hours, minutes, seconds; executing job is terminated when limit is exceeded; if omitted, jobs has no time limit.
<i>priority</i>	Job priority in range from 1 (highest) to 9999 (lowest); if omitted, priority is 9999.
<i>spool priority</i>	Outspool priority; same range as <i>priority</i> ; if omitted defaults to <i>priority</i> .
HO	Hold outspooling, if specified, the JOB's list spool is held until LU6 is reassigned, or until explicitly released, or until EOJ. May be specified anywhere after <i>name</i> , but must be last parameter; if omitted, list output spool is created automatically (but see NS).
NS	No outspooling; if present, may be specified anywhere after <i>name</i> , but must be last parameter; if omitted, list output spool is created automatically.

When :JO is used for spooling, it performs all the housekeeping functions required by a batch job (refer to paragraph 2-48) and also:

- assigns the list output spool to logical unit 6, unless NS is specified
- assigns job input spool file to logical unit 5, unless the job is entered from a logical unit specified by the :XE command (refer to paragraph 4-13) in which case that logical unit is used for job input.

HO Parameter

If you specify HO, job output is spooled to a disc file but is not sent to the line printer (LU6) until EOJ occurs. Or when you explicitly release the file (e.g., :CS,6,EN or :CS,6,PA). Or when you reassign LU6.

NS Parameter

If you specify NS, job output is not spooled to a disc file but is sent directly to the line printer (LU 6). If the line printer is not buffered, this may slow job execution. Usually, NS is specified only if job output might overflow the disc.

4-8. EO — END OF SPOOLED JOB

The EO command indicates the end of a job. It is the same command described in paragraph 2-49 with additional functions performed only if the job is spooled.

Format

```
:EOJ
```

When a job terminated by :EO is spooled, :EO performs all the housekeeping functions required by a batch job (refer to paragraph 2-49) and also:

- closes all spool files
- sets job status to completed and waiting for outspooling
- if command input was from a spool pool file, returns file to pool of available files
- searches JOBFIL for next job and, if any, returns to :JO command processor.

4-9. LU — SPOOL SET UP AND OUTSPOOL CONTROL

The :LU command specifies logical units with which spool logical units or spool files are to be associated. It also may be used to specify outspool attributes, actual output devices other than logical unit 6, and outspool priority.

It has two formats, one for input and the other for input and/or output.

Format

Format (input only):

```
:LU,lu[,namr]
```

Format (output or input/output):

```
:LU,lu[,namr[,attribute[,outlu[,priority]]]]
```

lu The logical unit with which a spool file is to be associated; in range 1-63 except logical units 2, 3 (when used to reference the auxiliary disc), 5 or a spool logical unit, or any logical unit associated with a disc driver (DVR30, 31, 32, or 33).

namr Name of existing file to be used as a spool file with which *lu* is to be associated; if omitted, system assigns a spool pool file; if set to 0, *lu* is returned to its standard system definition, i.e., association with spool file is cleared. A logical unit may be specified for *namr* in a spooled job, if the LU command is used to perform an LU switch (see paragraph 2-51). A logical unit may not be specified for *namr* where a spool reference is intended.

<i>attribute</i>	<p>Defines characteristics of <i>lu</i>, in regard to the job in which it is specified. Defaults are: read only, no buffering, outspool format, no hold, and save (file is saved only if <i>namr</i> is specified); any three of the following attribute codes can be combined in any order, without separators, in order to override defaults:</p> <p>HO hold file; do not queue for outspooling (i.e., do not make an entry for the file in SPLCON) until hold is removed or spool file is closed at end of job.</p> <p>WR write only; file is for output only, no hold. EOF automatically written as the first record.</p> <p>BO read and write with hold.</p> <p>WH write only with hold. EOF automatically written as the first record.</p> <p>BU file is buffered.</p> <p>PU file is to be purged following outspooling (if file is not to be outspooled, it is purged at the end of the job); <i>namr</i> must be specified.</p> <p>ST standard file format (refer to spool file formats paragraph 4-4).</p>
<i>priority</i>	<p>Outspool priority; if omitted, spool priority specified in :JO command is used.</p>
<i>outlu</i>	<p>Logical unit number or function code defining the logical unit to which the spool file will be outspooled. The <i>outlu</i> must have been specified as an outspool destination LU during GASP initialization. It must specify the logical unit of an actual output device; if omitted, the outspool file will not be output to a device. Any file can be outspooled, whether set up for read, or write, or both (the attribute of the spool file is in regard to the job in which the attribute is declared; it has no effect on the eventual outspooling of the file).</p>

NOTE

This form of the :LU command may be specified only in a batch job initiated by the command :JO, controlled by FMGR, and inspoiled through JOB.

Logical Unit Assignment

The system associates a spool logical unit with the specified logical unit. The spool logical unit associated with *lu* can be retrieved by displaying the global parameter 0S with FMGR DP command (paragraph 2-43).

The logical unit (*lu*) specified should correspond to the device type of the actual device for which it is intended (*outlu*). This ensures that control functions and control requests are issued to the device as expected. For example, if output is to be punched, logical unit 4 associated with the punch should be specified. If a logical unit specified for *lu* has not been associated with a particular device, magnetic tape is assumed.

lu must not be any of the following:

- logical unit 2 (system disc), or 3 (if assigned as auxiliary disc)
- any logical unit associated with a disc driver (DVR30, 31, 32, or 33)
- logical unit 5 (standard spool input device). If you want to use the true logical unit 5, use an LU switch command to obtain it (e.g., :LU,50,5 — see paragraph 2-51)
- a spool logical unit

Spool File Assignment

If *namr* is not specified, an available file from the spool pool files (SPOL01-SPOL80) is associated with the logical unit. If a particular file is specified, then that file is associated with *lu* and will be used as a spool file during spooling. The specified file must be an existing user file; :LU does not create the file. The name of the most recently assigned spool file can be retrieved by displaying the global parameter 1S with the FMGR DP command (paragraph 2-43).

Attributes

If *lu* is to be used for output only or for output as well as input, then its attributes must be specified.

No buffering saves System Available Memory space. Generally, there is no need to request buffering with BU. Outspool format is used rather than standard in most cases. The format makes no difference for input. Specify standard format (ST) if you want to read back the file either within the JOB or after the job. If the file is to be outspooled only, it is better to use the outspool format. If ST is specified, no control is passed through to the outspool routine which supplies a default EOF action based on the actual device to which the file is being outspooled.

If you do not want to save the user-defined spool file *namr*, you must specify PU. Spool pool files are never saved. The normal case is to ready the spool file for outspooling as soon as it is opened rather than to hold it. If you do specify HO, the held file will be placed in the outspool queue at job termination (:EO), or you may use the :CS command (paragraph 4-10) to queue the file for outspooling within the job.

The :LU hold attribute is not the same as the GASP hold established with the GASP CS command (see Section V). The :LU hold prevents the file from being placed in the outspool queue (an entry is not made for it in SPLCON) until a :CS command removes the hold or an :EO command is processed. The GASP hold places a hold on a file already entered into the outspool queue (already listed in SPLCON).

Outspool Logical Unit Assignment

The device to which output from spooling is to be directed must be specified with the *outlu* parameter, or the file will not be outspooled. The specified logical unit is not associated with a spool file or spool logical unit. Logical unit 1 may not be specified in *outlu*. Any other spool output logical unit specified at GASP initialization may be used. An octal function code that specifies an I/O function code in bits 6-10 and a logical unit in bits 0-5 can be used when a file with standard format is output; files with spool format use the function code in the second header word. For example, the code 104B indicates punch binary on logical unit 4. The function code is equivalent to the CONWD parameter in the standard I/O EXEC calls (refer to the RTE Operating System Reference Manual).

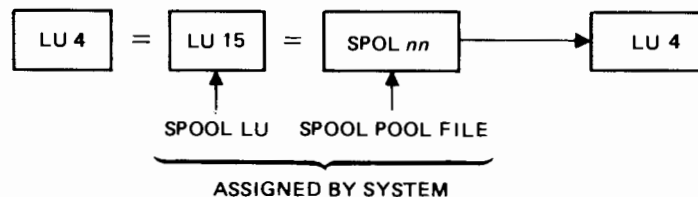
For spool file format, if the file is to be outspooled, be careful that the proper attribute is used to indicate the file type SPOUT will actually see.

If outspool format is specified for a standard file, the validity test performed by SPOUT will usually find the error and change it to standard format, however, one or more records may be transmitted before the test fails.

Examples

1. Request system to assign a spool file to the punch (logical unit 4):

```
:JO,LUEX1
:LU,4,,WRST,4 ← actual output lu is the punch
```



2. Request system to assign a user file for outspooling to the punch (logical unit 4). MYFIL must already exist:

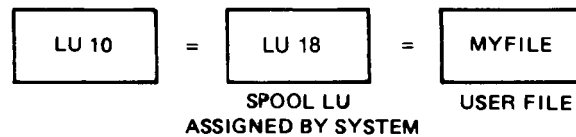
```
:JO,LUEX2
:LU,4,MYFIL,WRST,4
.
.
.
```

Spool Monitor

- Associate logical unit 10 with existing file MYFILE. Data is written to MYFILE whenever logical unit 10 is used in an I/O call or statement. No outspooling occurs:

```
:JO,LUEX3
:LU,10,MYFILE,WR
.
.
.
```

(WRITE(10,100)) writes to spool file MYFILE



- Setup to compile source program in &FILE and send relocatable program to existing file named %FILE:

```
:JO,LUEX4
:LU,50,&FILE
:LU,51,%FILE
:LU,6,,6
:RU,FTN4,50,6,51
```

- Setup logical unit 50 for source input file IFILE and logical unit 4 for relocatable output on OFILE; then compile:

```
:JO,COMPL
:LU,50,IFILE
:LU,4,OFILE,WRST ← WR must be specified for output file
:RU,FTN4,50,,4
.
.
.
:EO
```

4-10. CS — CHANGE SPOOL SET UP

During a job, the spool options for outspooling established by the :LU command can be changed with the :CS command. The changes can include not only the outspool attributes, but the logical unit for output and the outspool priority.

Format

```
:CS,lu [ ,EN
          ,RW
          ,PU
          ,SA
          ,PA
          ,NB
          ,BU
          ,NP[,outlu [,priority]] ]
```

Parameters

lu Logical unit number defined by the spool set-up command :LU.

EN Writes a final end-of-file on spool file and closes the spool; spool file is placed in outspool queue if not already there.

NOTE

Normally, spool files are not closed until job termination. However, when EN is specified in a CS command, the spool system closes the file immediately, and releases its LU Switch Table entry (see Figure 4-4).

RW If spool file is in standard format, resets (rewinds) spool file to first record. Causes next access to be at the beginning of the first record in the file. For example, if the job is required to read a file more than once (via the spool *lu*), the file must be rewound before each successive read operation. This also applies if the job is required to write a file more than once (that is, overwrite a file from the beginning of the file).

If spool file is in outspool format, a rewind is issued to the device associated with the spool file when this point of the outspool operation is reached. No positioning of the spool file occurs.

PU Changes save attribute of outspool file to purge. } named files only
SA Changes purge attribute of outspool file to save. }

PA Changes hold on spool file to pass file to outspool queue.

BU Changes no buffer attribute to buffered for spool file.

NB Changes buffered attribute to unbuffered for spool file.

NP Changes outspool logical unit and priority:

outlu New outspool logical unit number.

priority New outspool priority.

Spool Monitor

The :CS command can be used to change certain of the spool file attributes, either default or assigned by the :LU command. It can also be used to reset the file pointer to the beginning of the file (RW), or to write an end-of-file on the spool file and send it immediately to the outspool queue (EN).

Examples

The examples below all assume the jobs are processed in a batch-spool environment.

1. To remove a hold placed on a file associated with logical unit 6:

```
:JO,LUEX1
:LU,6,LIST,WHS ←————— lu 6 assigned to user file LIST for write only; file is to be held from
                  outspool queue (no entry made in SPLCON) and saved.
.
.
:CS,6,PA ←————— pass file LIST to outspool queue
.
.
:EO
```

2. Spool the input, list, and punch files for an assembly using existing files SOURCE, LIST, and PUNCH; list the assembly listing three times:

```
:JO,LUEX2
:LU,50,SOURCE ←————— set up assembly input file SOURCE
:LU,6,LIST,WR ←————— set up list file LIST
:LU,4,PUNCH,WR ←————— set up file PUNCH for assembly output and save
:RU,ASMB,50,6,4
:CS,6,EN ←————— write end-of-file on LIST
:CA,1,1 ←————— use 1G as counter with initial value 1
:LU,7,LIST,,6 ←————— set up actual logical unit for outspooling LIST
:IF,1G,EQ,3,2 ←————— test if 1G equals 3 and skip to EOJ if so
:CA,1,1G,+,1 ←————— increment 1G by 1
:IF,,EQ,,=4 ←————— loop back including current line in count to list "LIST" 3 times
:EOJ
```

3. Assemble file SOURCE three times, using CS command to rewind source to beginning prior to re-assembling:

```
:JOB,LUEX3
:LU,50,SOURCE ←————— equate LU 50 with file SOURCE: SOURCE must exist
:CA,1,1 ←————— set counter 1G to 1
:RU,ASMB,50 ←————— assemble SOURCE file
:IF,1G,EQ,3,3 ←————— after three assemblies, go to end of job
:CS,50,RW ←————— reset to start of SOURCE
:CA,1,1G,+,1 ←————— increment counter
:IF,,EQ,,=5 ←————— return to assemble SOURCE again
:EOJ
```

4-11. INSPoolING

The process of inspooling means submitting the job to be processed from an input device or file to the Spool Monitor under control of program JOB. Each job to be entered is recorded by JOB on the inspool status file JOBFIL. JOB schedules FMGR which then searches JOBFIL for jobs to process. (Refer to appendix C for JOBFIL format.)

The inspool program JOB recognizes one command that can be used to specify a file or device containing the job to be inspoiled. This command, :XE, can be interspersed with batch jobs when input is from a batch device or it can be entered interactively if input to JOB is from an interactive device.

4-12. RUNNING PROGRAM JOB

Program JOB is scheduled from RTE with the RU command. JOB can be scheduled from FMGR but the job is not processed until you exit from FMGR.

Input to JOB may be a device (default is the standard input, LU 5, usually the paper tape reader or card reader), or input to JOB may be a file. If input is from a device, more than one job can be stacked on the device. When input is from a file, only one job may be on the file.

If the input device is interactive, JOB expects input to be entered from that device in the form of FMGR commands or the :XE command. It prompts for interactive input with a semi-colon.

Format

Format 1 (input from non-disc device)

```
*RUN,JOB[,lu]
```

Format 2 (input from file)

```
*RUN,JOB,filename [,priority] [,cartridge #]
```

Parameters

<i>lu</i>	Logical unit of input device containing jobs to be spooled; if omitted, logical unit 5 is assumed.
<i>filename</i>	File name of file containing single job to be spooled.
<i>priority</i>	Job priority of file for spooling; specified only when input is from a file; if omitted, priority for file input is 9999.
<i>cartridge#</i>	The cartridge reference number of the cartridge where file named resides.

Job Entered From Batch Device

When the first format is used and *lu* is omitted, the jobs are expected to be on the standard batch input device, logical unit 5, which is usually a card or paper tape reader. JOB will read input from the device until it encounters a :JO, or :XE command signalling the start of the job. (Refer to Paragraph 4-13 for a description of the :XE command.) Ignoring all input prior to :JO, it then makes an entry for the job in JOBFIL, allocates a spool pool file for the job, and reads in the rest of the job, writing it to the spool pool file, until an :EO, :JO or :XE command terminates the job. At that point it closes the spool pool file and schedules FMGR to process the job. If FMGR is already scheduled, it will check JOBFIL automatically when it is terminated.

If *lu* is specified and it is a batch input device, JOB proceeds exactly as described above for the standard input device.

NOTE

If JOB is scheduled from a terminal other than the system console in a multi-terminal environment and input is not from that terminal, then the logical unit must be specified; for instance, 07>RU,JOB,5

After processing the first job on an input device, JOB looks for subsequent jobs. If there are none, it terminates. If there are more jobs, it processes them as it did the first job, continuing until there are no more jobs or an end-of-file is reached.

END-OF-FILE PROCESSING. If an end-of-file is encountered within a job (between :JO and :EO), it is entered in the spool file as a zero-length record. An end-of-file before or after a job terminates program JOB.

If an end-of-tape occurs on the paper tape reader within a job, program JOB suspends and the message: JOB WAIT ON PT is issued. To continue processing the job, load the rest of the job in the reader, ready the reader, and type *GO,JOB on the console. When the hopper of a card reader is empty within a job, simply add the rest of the cards and restart the reader; JOB continues reading the cards.

Job Entered Interactively

If the device specified by *lu* in the first format is interactive, program JOB prompts for input with a semicolon (;). You then enter your input at the terminal including any required colons. Interactive input to JOB is terminated by CTRL/D.

You may enter one or more jobs or :XE commands in the same manner as on a batch device. Processing of jobs proceeds exactly as if the job was entered through a batch device. It is often useful to store commonly used command sequences in procedure files. Then, any job may transfer to these files to do some of their work. Another method is to store often used jobs on files and use the :XE command when you want one of the jobs to be run. (Refer to paragraph 4-13 for :XE description.)

CTRL/D. Entering CTRL/D in an interactive environment is like an end-of-file in a batch environment. That is, if CTRL/D is within a job (between :JO and :EO), then a zero-length record is written on the spool file. But if it precedes or follows a job, it terminates program JOB.

Job Entered From a File

When the second format of *RU,JOB is used, JOB expects the job to be stored on the specified file. Only one job should be on the file since FMGR will only process one job per file. :JO should be the first command, :EO the last.

JOB does not read the file; it simply makes an entry for the file in JOBFIL and schedules FMGR to process the file as if it were a spool file.

Examples

1. Send output of program WHZAT to user file named MYFIL. MYFIL must already exist.

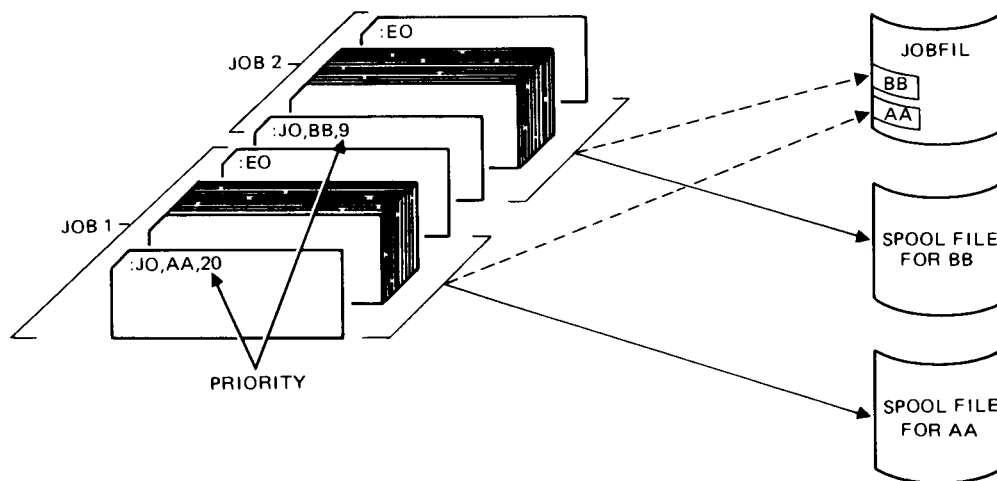
```
*RU, JOB, 1           set up spool environment
; :JO, XYZ           set up batch
; :LU, 6, MYFIL      associate LU 6 with MYFIL; no outspool
; :RU, WHZAT, 6      run WHZAT
; :EO               end batch
; (control D)       end spool
```

A listing of file MYFIL will show the WHZAT listing.

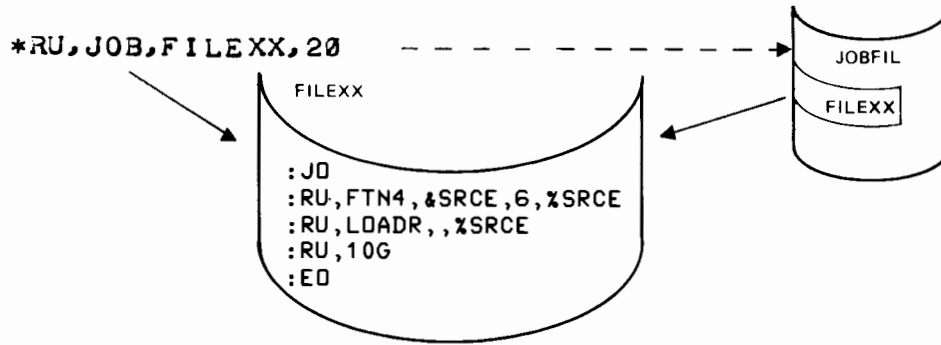


2. Two jobs are on the card reader:

```
*RU, JOB           (from system console)
OR
Ø7>RU, JOB, 5     (from terminal 7 in multi-terminal environment)
```

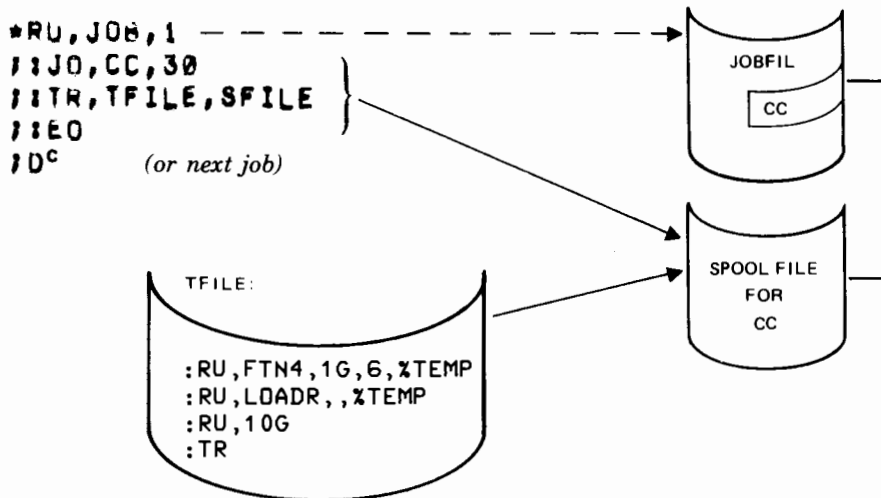


3. In an RTE-IV Operating System, one job is on file FILEXX:



Note that a spool file is not assigned in example 3; file FILEXX is used as the spool file. In this case, the file name is the job name and priority is specified directly to JOB when it is scheduled.

4. Job CC is entered from console (RTE-IV Systems):



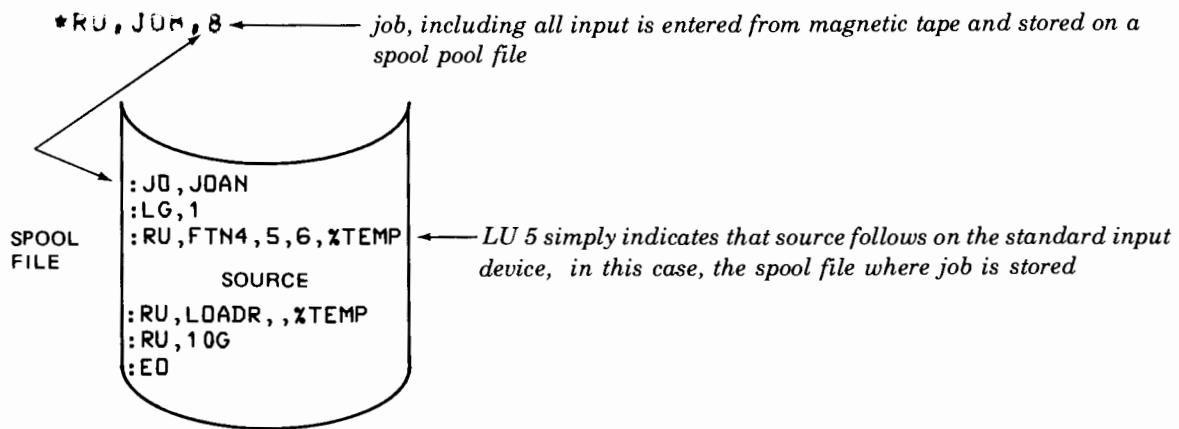
5. If you want to enter a job interactively with input to the job from paper tape, an LU switch is needed since LU 5 (paper tape reader) is redefined as a spool file by JOB (RTE-II/III Systems):

```

*RU, JOB, 1
:JD, COMPL ← enter job
:LG, 1
:LU, 50, 5 ← set up for source input from LU 5.
:PA, 1, LDAU PAPER TAPE LABELED ``COMPL`` IN READER
:RU, ASMB, 50, 99
:ED ← terminate job
D^ ← terminate interactive input
    
```

When the job is processed, the message is printed at the terminal so that the paper tape containing the source file can be loaded.

6. A job entered from any batch device (including LU 5) can request input from LU 5 and it will be entered correctly if the input immediately follows the request for it. To illustrate, consider the following job entered from magnetic tape (LU 8) that contains a request for source input from the standard input device (LU 5):



4-13. XE — JOB INPUT CONTROL

The :XE command indicates to program JOB that the next job to place in the job queue is on the device or file specified by :XE. :XE is used only when JOB is scheduled with input from a logical unit.

Format

```
:XE ,namr[ ,priority[ ,cartridge# ] ]
```

Parameters

<i>namr</i>	Identifies input device containing a job to be placed in job queue, may be a logical unit or the name of an existing file; refer to <i>namr</i> description, paragraph 2-8.
<i>priority</i>	Priority assigned to job; if omitted, priority is 9999.
<i>cartridge#</i>	The cartridge reference number of the cartridge where file named resides.

NOTE

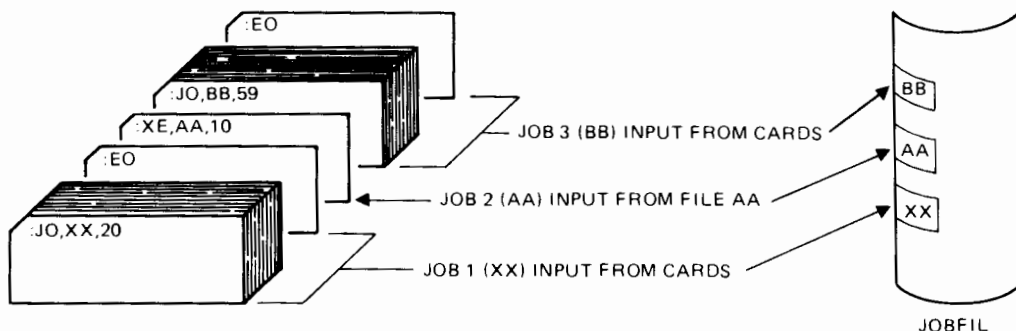
When XE is used to specify job input, JOB does not read the :JO command parameters (job name, priority, etc.); if input is from a file, the file name is used as the job name; if from a logical unit the job name is left blank and a job number assigned for the Entry in JOBFIL. When FMGR processes each job, it recovers the :JO command parameters and places them in JOBFIL.

XE Entered from Batch Device

If JOB expects input from a batch device such as logical unit 5, all jobs on the device must be delimited by :JO and :EO. The command :XE may precede a :JO or follow an :EO command to indicate to JOB that the next job is on the device or file specified by :XE.

To illustrate, three jobs are to be spooled by JOB; two are on LU 5, the card reader, and one is on file AA.

*RU, JOB ← by default, input is on LU 5



:XE Entered Interactively

If JOB expects input from an interactive device, :XE can be entered just like any FMGR command in response to the semicolon prompt issued by JOB. XE must be preceded by a colon. JOB interprets the XE command in this context exactly as if it were specified on a batch device.

For example, to enter a job from a file, XE can be used as follows:

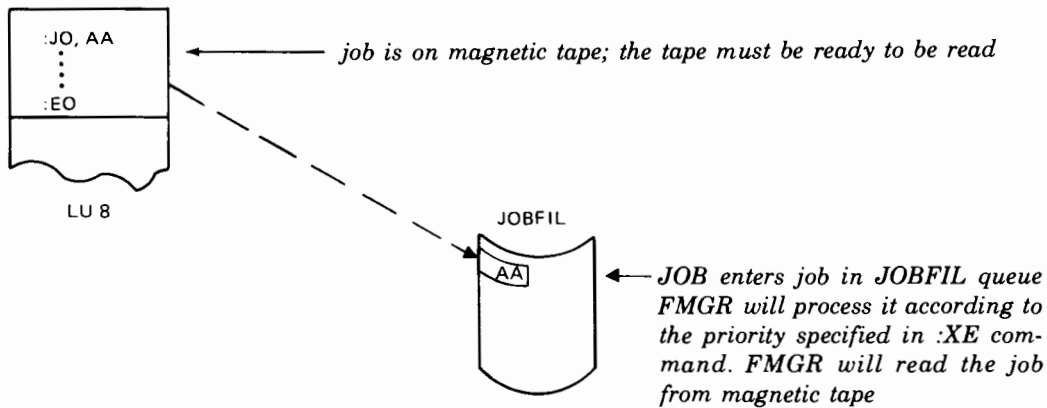
```
*RU, JOB, 1 ← JOB expects input from system console
:XE, FILEXX, 20 ← enter :XE at console
JOBc (or next job)
```

This example performs exactly the same function as example 2 in paragraph 4-12.

XE to Specify Input From a Device

So far, we have shown how :XE can be used to inspool a file. It can also be used to specify direct job input from a logical unit. If *namr* specifies a logical unit, JOB expects the job input to be entered from that device. For example, if logical unit 8 is assigned to magnetic tape, then the following command tells JOB that a job is on magnetic tape:

```
*RU, JOB, 1
;:XE, 8, 10 ← enter XE at console; priority is 10
;Dc
```



4-14. ERROR CONDITIONS

JOB will suspend under the following conditions:

- there is no room for the job entry in JOBFIL
- only one spool file is available for the job; JOB will not use the last spool file since it might be needed for job processing.
- no spool EQT entry is available for the job; unless the job is a file, an EQT entry is needed to assign to a spool file.

No error messages are issued in these cases. The program suspends and will restart automatically when resources become available. If you are not sure what has happened, use the RTE STATUS command (*ST, JOB) to find out if JOB has suspended. If it has, you should wait and try again. When other spooled jobs are complete their spool files and entries in JOBFIL and the EQT table are released automatically and JOB will continue as soon as the necessary resources become available.

Table 4-1 lists the error messages that may be received during JOB execution.

Table 4-1. JOB Error Messages

MESSAGE	CAUSE	CORRECTIVE ACTION
JOB WAIT ON PT	End-of-tape occurred between :JO and :EO commands.	Load remainder of job in reader, ready the reader, and enter *GO,JOB.
JOB WAIT ON SPOOL RESOURCE	Required spool file or logical device cannot be obtained at this time.	None required. JOB will suspend and be automatically rescheduled when the resource becomes available.
JOB WAIT ON EXTENT	Spool file overflows available disc space.	Condition will automatically clear when SMP releases spool files as a result of outspool completion; or you can force a retry using the GASP command, SU; or you can abort JOB.
END JOB ABNORM	JOBFIL could not be opened; or other uncorrectable error occurred; or JOB was run before Spool initialization.	Try re-initialization with GASP after all spool activity is completed.

4-15. OUTSPOOLING

Outspooling is the process of directing output from spooled jobs or outspool files to specified devices according to an outspool priority. The device for outspooling standard job output (list output) is the line printer. Other job output may be directed to other devices.

After JOB records each inspoiled job in JOBFIL, it schedules FMGR if FMGR is not already scheduled, in order to process the jobs recorded in JOBFIL. Jobs are processed according to their priority. As soon as FMGR starts processing a job it calls program SMP to set up the spool to process the standard output from that job.

SMP makes an entry in file SPLCON for each job's output, opens an outspool file for the output, and puts a notation of the spool file in the outspool queue. When that file is at the top of the queue and SPOUT is ready for it, SMP tells SPOUT to start dumping the file. A similar process occurs when the user specifies other outspools with the :LU command.

SPOUT can outspool to several devices at a time as long as there are enough spool logical units available and enough spool files allocated to the Spool Monitor to assure that several spools to various devices can be built concurrently. As SPOUT performs its outspool functions, it communicates with SMP so that SMP can modify the status and queue information in file SPLCON. Once scheduled, SPOUT runs continuously, receiving information from SMP on the outspool files to be processed. When SPOUT is not busy, it is suspended in a Class GET call waiting for more information.

File SPLCON is a directory of all files to be outspooled (refer to Appendix C for SPLCON format). Entries are queued in this file according to their outspool priority. The file also keeps track of the particular status of the spool file: whether it is still in the queue waiting to be outspooled, whether it has been passed to SPOUT for actual outspooling, and whether it is being held from completion of outspooling, or is completed. When completed, the spool file is removed from the outspool queue.

The :LU and :CS commands can be used to specify the type of outspool device to be used as well as the device logical unit and outspool priority. They also specify when the file is to be placed in the outspool queue maintained in SPLCON. For example, HO is specified as an outspool attribute if the file is to be held from the outspool queue until the file is closed. This attribute is not the same as the outspool status Hold maintained in SPLCON.

4-16. SPOOL STATUS

Spool status for jobs being inspoiled is maintained in file JOBFIL; the status of jobs being outspooled in file SPLCON.

4-17. INSPOOL STATUS

There are four states through which a job normally progresses as it is inspoiled and processed. The GASP DJ command may be used to examine the state of an inspoiled job. The four normal states are:

- I In spooling; job being read by JOB.
- R Ready; job is at top of inspool queue and is ready to be run by FMGR.
- A Active; job is being processed by FMGR.
- CS Completed Spooling; job has been processed and is waiting to be outspooled.

An additional state may be specified with the GASP command CJ (see paragraph 5-8). If a job is ready but is not yet active, it can be suspended from further processing. This job status is:

- RH Ready-Hold; a job in the ready state has been held; it will not be processed until specifically released.

4-18. OUTSPOOL STATUS

The state of each outspool file produced by a job is recorded in SPLCON when the file is placed in the outspool queue. The states of all outspool files may be examined using the GASP DS command. The possible states of outspool files are:

- W Wait; file is in queue but is not at the top of the queue.
- A Active; file has reached the top of the queue and is being outspooled.

In addition to these two states, there are two possible hold states. A hold may be placed on a file in the outspool queue or being actively outspooled only through the GASP operator command CS (see paragraph 5-12). The two hold states are:

- H Hold in wait status; file will not be released to SPOUT until you specifically release it with the GASP command RS (paragraph 5-13).
- AH Active-Hold; file was being outspooled when a hold was requested with the GASP command CS; it remains suspended until released by the GASP command RS. The device to which file was being outspooled is idled until this hold is released or the file is killed or re-started.

4-19. OUTSPOOL ERRORS

During outspooling, if the device associated with the outspool file becomes unavailable (down), the file is placed in the active hold state and an error message is displayed:

```
SMP: LU xx DOWN filename HELD
```

where *xx* is the logical unit number of the down device, and *filename* is the name of the outspool file.

For example, the paper tape punch has run out of tape, or the line printer has run out of paper.

To recover, determine which device is down (if necessary) using the GASP command DS to display Spool Status. Correct the problem at the device. Then use the system command UP (SYUP may be used in FMGR) to declare the device available. Next use either the GASP command RS to restart the outspool from the beginning, or CS to release the hold state of the outspool file.

If a file was outspooled and not completed because of spool overflow, the following message is displayed:

```
SMP: LU xx EOF ER filename
```

where *xx* is the logical unit number of the device on which the file was outspooled and *filename* is the name of the outspool file.

To recover, re-run the JOB.

If an ASCII file was outspooled and not completed because of spool overflow, the following message is printed after the last line of the file:

```
BAD EOF
```

To recover, re-run the JOB.

If SMP encounters an FMP error in one of its operations, the following message is displayed:

```
SMP: FMP-nn
```

where *nn* is the FMP error number (see Table B-1).

If an end-of-tape condition occurs on input from the paper tape reader during the JOB run, this message is displayed:

```
JOB WAIT ON PT
```

Load the remainder of the job tape into the reader, ready the reader, and enter GO,JOB (SYGO,JOB in FMGR).

If a required spool file or logical device is temporarily unavailable, the following message is displayed:

```
JOB WAIT ON SPOOL RESOURCE
```

In this case, **JOB** will suspend and be automatically rescheduled when the resource becomes available.

If the spool file overflows available disc space, **JOB** displays the message:

```
JOB WAIT ON EXTENT
```

This condition will automatically clear when **SMP** releases spool files as a result of outspool completion. You can force a retry using the **GASP** command **SU**, or you can abort the **JOB**.

If **JOBFIL** cannot be opened, or some other uncorrectable error occurs, or **JOB** was run before spool initialization, the following message is displayed:

```
END JOB ABNORM
```

You can attempt re-initialization using **GASP** after all spool file activity is completed.

The following abort messages may appear on logical unit 1:

IO20	Read attempted on write only spool.
IO21	Read attempted past EOF.
IO22	Second attempt to read a JCL card from the batch input file by other than FMGR.
IO23	Write attempted on read only spool.
IO24	Write attempted beyond an EOF (usually spool file overflow).
IO25	Attempt to access a spool LU that is not currently connected to a file (i.e., that is not open).

A summary of error messages is included in Appendix B.

SECTION V

SPOOL CONTROL WITH GASP OPERATOR COMMANDS



INDEX TO GASP OPERATOR COMMANDS

Schedule GASP: *RU,GASP[,lu]		
Command Syntax	Function	Page
AB, <i>job #</i>	Abort pending job.	5-6
CJ, <i>job #</i> , <i>priority</i> ,H ,R	Change job status or priority.	5-6
CS, <i>spoolfile</i> , <i>priority</i> ,H ,R	Change outspool status or priority.	5-8
DA KILL SPOOLING?YES/NO	Deallocate spool system; respond YES to execute DA, anything else to terminate DA.	5-16
DJ [, <i>job #</i>] , <i>jobname</i>]	Display all current jobs or particular job, and job status.	5-4
DS[, <i>lu</i>]	Display all current outspools or all spools to a particular device, and spool status.	5-7
EX	Terminate GASP.	5-3
KS , <i>spoolfile</i> , <i>lu</i>	Kill outspool file or file currently outspooling to a device.	5-11
RS, <i>spoolfile</i> [, <i>lu</i>]	Restart outspool from beginning; optionally, to a new device.	5-10
SD [,B] ,S]	Shut down batch processing (pending jobs) or outspooling (pending spools) or both.	5-14
SU [,B] ,S]	Start up batch processing (pending jobs) or outspooling (pending spools) or both.	5-15
??[, <i>error #</i> [, <i>lu</i>]]	Request explanation of error code.	5-3

SPOOL CONTROL WITH GASP OPERATOR COMMANDS

SECTION

V

5-1. INTRODUCTION

Once a job has been entered in the spool system, its processing is automatic. As described in Section IV, priority for execution and output can be specified in the :JO command, spooling priorities and device information in the :LU command. Given these values, spooling proceeds automatically unless you choose to change the sequence of operations with the GASP commands described in this section.

Each time GASP is run, it checks whether the job queue file JOBFIL exists. If it exists, GASP knows that it has previously initialized the spooling system and the user may enter the commands explained in this section. If JOBFIL does not exist, GASP enters the initialization process to set up the spooling system (see Section VII on GASP initialization).

The interactive program GASP provides a set of commands that allow you to:

- change the job priority or abort a job that is not yet processed
- hold a job from processing or release it from hold
- hold (delay) job output from being outspooled to a device or release it from hold
- display the status, priority, job number, and name of all jobs
- display the status, priority, job number, and output logical unit of all outspool files
- restart an outspool or direct an outspool to another device
- "kill" the job output from a completed job
- shut down and start up the spool system without altering jobs or outspools
- remove the spool capability from the system by purging all spool files.



5-2. GASP COMMANDS

A summary of the commands that perform these functions is listed in Table 5-1.

Table 5-1. GASP Operator Command Summary

CATEGORY	COMMAND	FUNCTION	PARAGRAPH
GASP Operation page 5-2	*RU,GASP or nn>RU,GASP	schedule GASP	5-3
	??	request error explanation	5-4
	EX	terminate GASP	5-5
Job Manipulation page 5-4	DJ	display job status	5-7
	CJ	change job status or priority	5-8
	AB	abort a job before it runs	5-9

Table 5-1. GASP Operator Command Summary (Continued)

CATEGORY	COMMAND	FUNCTION	PARAGRAPH
Outspool Manipulation page 5-7	DS	display outspool status	5-11
	CS	change outspool status	5-12
	RS	restart an outspool	5-13
	KS	kill an outspool	5-15
Spool System Manipulation page 5-14	SD	shut down spool system	5-17
	SU	start up spool system after shut down	5-18
	DA	purge spool system files after shut down	5-19

5-3. RUNNING PROGRAM GASP

GASP may be run from RTE or from a copy of FMGR. It can not be scheduled from FMGR unless no jobs are active since FMGR controls job processing during spooling.

Format

*RU,GASP[,*lu*] from RTE

nn>RU,GASP[,*lu*] from a copy of FMGR

Parameters

lu Logical unit of interactive device on which GASP commands are entered; if omitted, logical unit 1 (system console) is assumed. In a multi-terminal environment, *lu* must be specified if it is different from the terminal logical unit.

When GASP is scheduled, it responds with a prompt. On some devices, this is an up-arrow (↑); on others such as the HP 2640 terminal, it is a caret (^). Any of the commands shown in Table 5-1 may then be entered.

Examples

1. Run GASP from system console:

```
*R1,GASP ← input expected from logical unit 1
↑ ← GASP prompt
```

2. Run GASP from terminal 07 using FMGR copy FMG07:

```
*R1,FMG07 ← input expected from logical unit 7
;R1,GASP
↑
```

- Run GASP from system console using FMGR copy FMG07 and direct input to terminal 7:

```
*RU,FMG07
:RU,GASP,7 ←————— input expected from logical unit 7
↑
```

5-4. GASP ERROR EXPLANATION

When an error occurs during GASP operation, an error code is printed in the form of a positive or negative integer. Negative integers indicate FMP errors caused by calls to File Management Package utility programs; positive error codes indicate GASP program errors. The error codes are printed at the terminal. Appendix A contains a complete list of Batch-Spool Monitor error codes with their meaning. You can also request GASP to display the meaning of a particular error or all errors with the ?? command. This explanation is normally displayed at your terminal, but you can ask that it be displayed on another output device.

Format

^??.[error#[,lu]]

Parameters

<i>error #</i>	Positive or negative integer specifying a particular GASP error code; if omitted, an explanation of the most recent error is displayed; if <i>error #</i> is 99, an explanation of all error codes is displayed.
<i>lu</i>	Logical unit on which error code description is displayed; use only if <i>error #</i> is 99 for a listing of all errors; if omitted, display is at terminal.

Examples

- Request meaning of latest error code:

```
↑AB,AA ←————— AB requires a job number, not a name
GASP 3
↑??
GASP 3 BAD JOB NUMBER!
```

- ↑??,99,6 ←————— request list of all error code meanings on line printer

5-5. EX - TERMINATE GASP

When you are through using GASP, you may return to the system from which GASP was scheduled with the EX command.

Format

^EX

GASP prints:

END GASP

and then terminates.

5-6. JOB MANIPULATION

Jobs that have been entered in the directory file JOBFIL by program JOB can be manipulated with the GASP commands:

- DJ display job status
- CJ change job status
- AB abort pending job

JOBFIL is the inspool directory that maintains the queue of inspoiled jobs with information on each one. These jobs may be in one of the following states:

- I in process of being inspoiled by JOB
- R ready to be processed
- RH ready to be processed but being held
- A active, that is, being run
- CS completed and ready for outspooling

5-7. DJ - DISPLAY JOB STATUS

The DJ command displays the job number, job name, job status and priority, and the spool pool files assigned to the job except the job input spool. You may request that all jobs currently in the spool system or any particular job be displayed.

Format

^DJ [,*job #*]
[,*jobname*]

Parameters

job # Job number of particular job to be displayed.

jobname Name of the job or jobs to be displayed.

If both *job #* and *jobname* are omitted, all jobs currently in the system are displayed.

The display is formatted as:

```
##      NAME      STATUS      SPOOLS
```

The first field (##) is the job number. Jobs are numbered consecutively from 1 according to the record they occupy in JOBFIL.

Under NAME is the job name assigned in the :JO command or in the :XE command to program JOB. If a job is entered directly from a device specified in the :XE command, then this name is blank until FMGR processes the job. Refer to XE command description, paragraph 4-13.

The field STATUS lists the job source and priority as well as its status in the form:

x pppps

where *x* is either S or D. S means the job is on a file, D that it is on a device; *pppp* is the job priority; *ss* is the current job status, I, R, RH, A, or CS, (see paragraph 5-6).

Under the heading SPOOLS are the integers identifying any spool pool files associated with the job except the job input spool. For example, if the job uses spool files SPOL01 and SPOL10, then 1 and 10 are listed under SPOOLS. When a job uses more than one spool, the spools are listed in ascending numeric order.

Examples

1. Display all jobs in the system:

```
↑DJ
##      NAME      STATUS      SPOOLS
  1      DLIST      S      10CS
  2      IDLE      S      20 A
  3      COMPL     S      40 R
  4      XX        S      50RH
```

SPOOLS
1 ← outspools
2 ← assigned to
 jobs

Job 1 is processed and ready to be outspooled; job 2 is currently being processed; job 3 is ready to be processed; job 4 is ready but has been held with a CJ command.

2. Display job 2 (IDLE):

```
↑DJ,2
```

or

```
↑DJ, IDLE
```

```
##      NAME      STATUS      SPOOLS
  2      IDLE      S      20CS      2
```

Job 2 is now completed.

5-8. CJ - CHANGE JOB STATUS

Job status or priority can be changed with the CJ command. This command may only be used for a job in R, or RH status; it may not be used if the job is active or has completed processing.

Format

<code>^CJ,job#</code>	<i>,priority</i> ,H ,R
Parameters	
<i>job#</i>	Number assigned to job by spool system; use DJ to display job numbers.
<i>priority</i>	New job priority; only legal before job is active.
H	Hold job from processing; changes R status to RH.
R	Release job for processing; changes RH status to R.

Example

1. `↑CJ,4,H` ← *hold job 4; sets status to RH*
2. `↑CJ,4,R` ← *release job previously held*
3. `↑CJ,3,30` ← *assign new priority to job 3*

5-9. AB - ABORT JOB

Before a job is processed, it may be removed with the AB command; AB deletes the job entry from the file JOBFIL. In order to abort a currently active job, get the attention of RTE and enter the RTE AB command.

Format

<code>^AB,job#</code>	
Parameter	
<i>job#</i>	Number assigned to job by spool system; use DJ to display job numbers.

This command may be used only for jobs in R (ready) or RH (ready and hold) status. The command sets the job status to A (active) so that FMGR removes it from the job queue, as if it had been processed, the next time it scans the queue. FMGR scans the queue when it finishes processing the current job. If FMGR is dormant, GASP schedules it and FMGR scans the queue as soon as it starts executing.

Example

`↑AB,4` *removes job 4 before it is processed*

5-10. OUTSPOOL MANIPULATION

Spool files in the outspool queue maintained in the directory file SPLCON can be displayed or their status changed with the GASP commands:

DS display spool status
 CS change spool status
 RS restart outspool from beginning
 KS purge spool file from outspool queue



An outspool is normally in one of two states: either waiting for a device or actively being written to the device. In either of these two states, an outspool file can be held and then subsequently released. The possible outspool states are:

W waiting for a device, or put into wait state by :CS or :LU command
 A being written to the device
 H held by operator; outspool was in wait status
 AH held by operator; outspool was in active status

When an active file is held, the device to which it is outspooling is locked from further use until the outspool is released, killed, or respoiled. This maintains the integrity of the output to that device.

5-11. DS - DISPLAY SPOOLS

The DS command displays the spool file name, job number, outspool priority, and if an outspool, the outspool status, and the logical unit to which the file is being or will be dumped. You may request either all current spools or all outspools currently assigned to a particular logical unit.

Format

`^DS[,lu]`

Parameter

lu Outspool logical unit; only files directed to this *lu* are displayed; if omitted, all files in the outspool queue are displayed.

The spool display heading is:

LU NAME PRIORITY JOB# STATUS

GASP Operator Commands

Listed below this heading is one entry for each file being spooled, that is, for each file listed in SPLCON.

The first field under LU contains the outspool logical unit number. Under NAME are the names of the files currently assigned for spooling. The next field contains the outspool priority; this is the same as the job priority unless a different outspool priority was specified in the :JO, :LU, or :CS commands (see Section IV). Under JOB# is the number of the job that generated the outspool file. Under STATUS, one of the possible outspool states is listed as: A, W, AH, or H (see paragraph 5-10); or, if the file is not to be dumped the status is "--".

If there are no spools currently being outspooled or read or written by a job, the message NO SPOOLS is printed below the heading. If spooling has been shut down, the message SHUT DOWN is printed.

If the spool file has not been assigned to a device for outspooling, then two dashes (-) are printed under the headings LU and STATUS. This means the spool file has been assigned to a job but is not used for outspooling; that is, it was used for input only or is not to be outspooled.

Examples

1. Display all current spool files:

```
↑DS
```

LU	NAME	PRIOTRITY	JOB#	STATUS
6	SPOL01	010	1	A
6	SPOL02	020	2	W
6	SPOL03	040	3	W
6	SPOL04	050	4	W
--	SPOL05	000	1	--

← SPOL05 not used for outspooling

All spool files except SPOL01 are in wait state; SPOL01 is active.

2. Display all spool files currently assigned to LU 6:

```
↑DS,6
```

The resulting display is identical to the first four entries in the preceding display; that is, all but the entry for SPOL05 is displayed.

5-12. CS - CHANGE SPOOL STATUS

The status of an outspool file can be changed with the command CS. CS will also change spool priority, but only if the outspool file is not active.

Format

```
^CS,spoolfile ,priority
      ,H
      ,R
```

Parameters	
<i>spoolfile</i>	Name of spool file as displayed by DJ.
<i>priority</i>	New outspool priority.
H	Hold spool file; if active, changes status to AH; if waiting, changes status to H.
R	Release spool file that has been held in AH or H status.

When a hold is placed on an active file, the device to which it is outpooling is locked and the device is unavailable until the hold is released with the R parameter of CS, or the file is killed with the KS command or respooled with RS. When a hold is released on a file in AH status, the outpooling is continued from where it was when held.

The priority of an active spool file may be changed; the entry in SPLCON reflects the new priority but the outpooling continues unaffected by the new priority unless the file is respooled. If priority is changed on a file in Wait or Hold status, it may affect when it is outspooled.

For the effects of CS on outspools in various states, refer to Figure 5-1.

Examples

1. Place hold on active spool, SPOL01:

```
↑CS,SPOL01,H ← place hold on SPOL01
↑DS
```

LU	NAME	PRIORITY	JOB#	STATUS	
6	SPOL01	010	1	AH	← SPOL01 was actively outpooling
6	SPOL02	020	2	W	
6	SPOL03	040	3	W	
6	SPOL04	050	4	W	

```
↑CS,SPOL01,R ← SPOL01 will continue outpooling from point where it was held
```

2. Change priority of inactive spool file, SPOL04:

```
↑CS,SPOL04,25
```

```
↑DS
```

LU	NAME	PRIORITY	JOB#	STATUS	
6	SPOL01	010	1	AH	
6	SPOL02	020	2	W	
6	SPOL03	040	3	W	
6	SPOL04	025	4	W	← new priority displayed

5-13. RS - RESTART SPOOL

An active outspool file can be restarted from the beginning with the command RS.

Format

```
^RS,spoolfile[,lu]
```

Parameters

<i>spoolfile</i>	Name of active or active-held spool file in outspool queue.
<i>lu</i>	New logical unit to which file is to be outspooled; if omitted, logical unit previously assigned is used for spool output.

When an active or active-held spool file is restarted, it is restarted from the beginning of the file. The restart automatically removes the hold from an active-held file. If the file is to be outspooled to a different device, it *may* go back into wait state on the new outspool queue.

A file in wait state is not affected by the RS command unless *lu* is specified to change the outspool logical unit, and thereby move the spool to another queue. Files that have completed outspooling are no longer in the SPLCON queue and, thus, may not be restarted.

For the effects of RS on outspools in various states, see Figure 5-1, Changing Outspool File Status.

Examples

1. Restart active spool file, SPOL03:

```
↑RS, SPOL03
```

2. Change the outspool logical unit from the unit 6 to 4:

```
↑RS, SPOL04, 4
```

In this case, the file can be either active or waiting.

5-14. SPOOL FILE STATE DIAGRAM

Under normal spool control, without using GASP, an outspool file in the Wait state (status W) is selected by SMP for outspooling when it reaches the top of the outspool queue in file SPLCON. This procedure can be altered with either CS or RS as illustrated in Figure 5-1.

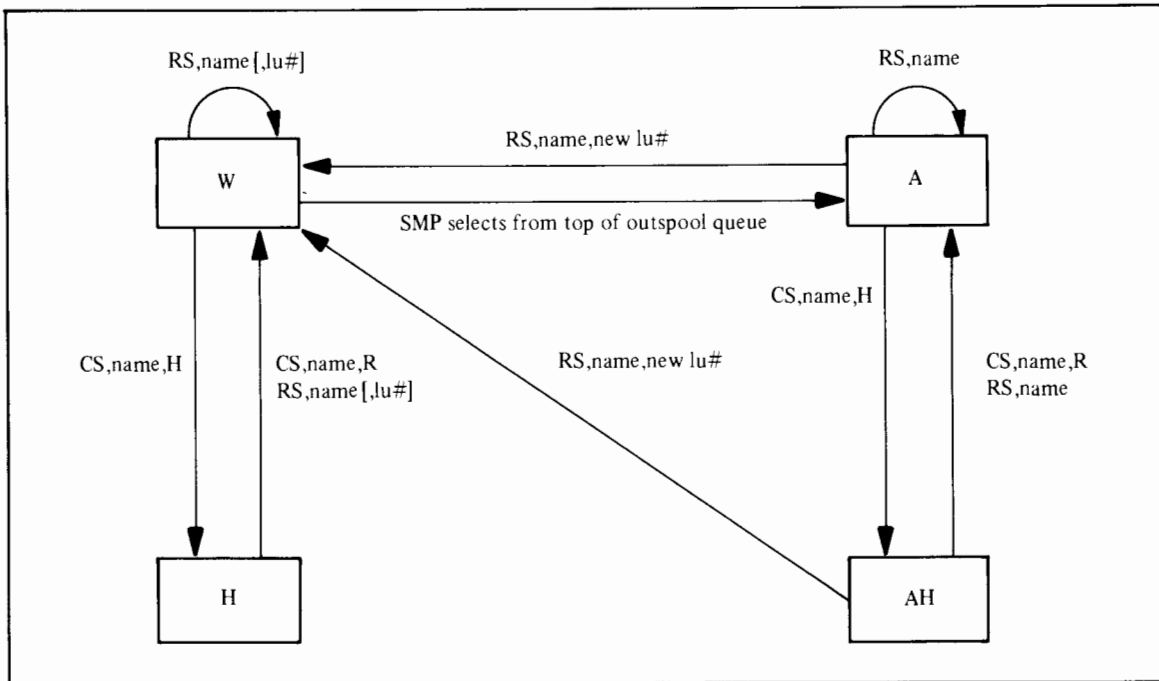


Figure 5-1. Outspool File State Diagram

To summarize, the possible changes:

- From wait state (W), a file can be:
moved to active state (A) by SMP as next outspool
moved to hold state (H) with CS command.
- From hold state (H), a file can only be:
moved to wait state (W) with CS or RS commands.
- From active state (A) a file may be:
returned to wait state (W) with RS command
moved to active-hold state (AH) with CS command.
- From active-hold state (AH) a file may be:
moved to wait state (W) with RS command
moved to active state (A) with CS or RS commands.

5-15. KS — KILL OUTSPOOL

An outspool file can be removed from the outspool queue (killed) with the KS command.

Format

<code>^KS</code>	<i>,spoolfile</i> <i>,lu</i>
Parameters	
<i>spoolfile</i>	Name of outspool file to be removed.
<i>lu</i>	Logical unit of device to which file is being outspooled.

When a file actively being outspooled is killed, the output is terminated. If any other file is in the queue to be outspooled to that device, the file at the top of the outspool queue in SPLCON is started.

You may kill an outspool whose job is still active. However this may cause job or program abortion. In this case GASP will verify the kill command before proceeding.

Example

```
*RU,GASP
ADS
LU NAME PRIORITY JOB# STATUS

 6 CO2801  099  18  A
AKS,CO2801
MAY ABORT PROGRAM OR JOB, OK TO KILL? YES ← Yes, i.e., Kill is verified
^EX
END GASP

I025 ASMB  51725 ← ASMB was listing to CO2801 so it loses on next attempt.
ASMB ABORTED
```

Caution should be used when *lu* is specified. This will purge the file currently being outspooled to that logical unit. If there are other files queued for outspooling to that device, it is possible that the file you intended to purge has completed outspooling, in which case, you will inadvertently purge the next file in the queue since it has become the currently active file to the specified device.

Examples

1. Examine the job list to determine if the job XX has completed processing; if it has, then kill the job's outspool file:

```
↑DJ
##      NAME      STATUS      SPOOLS
 1     OLIST      S      10CS      1
 2     TDLR      S      20CS      2
 3     COMPL      S      40CS      3
 4     XX        S      25CS      4

↑DS,10
LU NAME PRIORITY JOB# STATUS
10 SPOL04  050  4  CS
```

Since job XX using spool file SPOL04 is completed, the job need not be aborted:

↑KS, SPOL04 ←————— *kill spool of inactive job XX*

If job XX had been active, it would have to be aborted before using KS:

*AB, 1 ←————— *abort active job*

*RU, GASP

↑KS, SPOL04 ←————— *then kill spool*

2. If no other jobs are being outspooled to logical unit 10, the spool file can be killed with:

↑DS, 10

LU	NAME	PRIORITY	JOB#	STATUS
10	SPOL04	050	4	CS

10	SPOL04	050	4	CS
----	--------	-----	---	----

↑KS, 10 ←————— *kill file outpooling to logical unit 10*

5-16. SPOOL SYSTEM MANIPULATION

At any time during spooling, either in the job processing or the spool output phase, the spool system can be shut down. Nothing is lost by shut down and start up. The spool shut down effectively suspends batch processing and/or outspool operations until a start up command restarts the spool process. While the spool system is shut down, it may be restarted or else removed from the system using the GASP command, DA (see paragraph 5-19). If removed, all spool files are purged, GASP is terminated, and the spool system is no longer an active part of the Batch-Spool Monitor and must be re-initialized before further spooling can be performed. While it is shut down, all spool files are closed and the cartridges on which they reside can be packed.

The commands to perform these functions are:

SD	shut down spooling
SU	restart spooling after shut down
DA	deallocate the spool system

The SD and SU commands provide a means to shut down and restart the system in an orderly manner. When both job processing and outspooling are shut down and the current job is processed and any outspooling completed, the following events may occur.

- SMP closes SPLCON when the last active file finishes outspooling and the currently active job is completely processed.
- Spool discs may be packed.
- The entire system may be halted and then restarted from disc (booted) and GASP may then be run to start up the spool system from where it left off.

5-17. SD - SHUT DOWN SPOOLING

You may hold all spooled jobs, all spooled output, or both with the SD command.

Format

$$\wedge SD \left[\begin{array}{l} \text{,BATCH} \\ \text{,SPOOL} \end{array} \right]$$

Parameters

B	Hold all pending jobs; outspooling is not affected.
S	Hold all pending outspools; job processing is not affected.
none	If both B and S are omitted, then both job processing and outspooling are held. Inspooling by JOB may continue.

Any jobs or spools that are currently active when SD is specified are allowed to run to completion. New jobs may be submitted after the system is shut down; if jobs are being held,

then the new jobs will also be held until the system is restarted with SU. If only outspooling is shut down (SD,S), then jobs may be run but no output will be spooled until outspooling is restarted.

Examples

1. † SD, B ← *no further batch job processing until system is restarted*
2. † SD, S ← *no further outspooling until system is restarted*
3. † SD ← *no further job processing or outspooling until system is restarted*

5-18. SU - START UP SPOOLING

To start up the spool system after it has been shut down with SD, use the SU command. Also, use SU to start batch and/or spool if SMP or FMGR has been aborted by the operator, or after restarting to process any jobs or files remaining in the queues from previous runs.

Format

\wedge SU [, <u>BATCH</u>] [, <u>SPOOL</u>]	
Parameters	
B	Jobs held with SD are released; does not restart outspooling
S	Outspools held with SD are released; does not restart job processing
none	Both jobs and outspools held by SD are restarted

A job or spool file that was held with a command other than SD will not be restarted. For instance, SU will not restart spooling on a particular outspool file held with CS or a job held with CJ.

Examples

1. † SU, B ← *restart jobs held with SD or SD,B*
2. † SU, S ← *restart outspooling held by SD or SD,S*
3. † SU ← *restart job processing and/or outspooling held by SD*

4. If you want to run a job now, but spool output at a later time, you can use the SD and SU commands:

```
*RU, GASP
† SD, S           shut down outspooling
† EX
```

Jobs can now be inspoiled and processed; there will be no outspooling until:

```
*RU, GASP
† SU, S           start up outspooling
† EX
```

5-19. DA - DEALLOCATE SPOOLING

The spool system can be deallocated from the Batch-Spool Monitor with the DA command. Before using DA, the spool system must be shut down, all files must be closed, and all current job processing and/or outpooling should be completed.

CAUTION

Be certain that all discs containing spool system files are mounted (see paragraph 2-55) before using the DA command.

Format

^DA

Response:

KILL SPOOLING? The system prints this message in response to DA in order to give you a chance to change your mind; answer YES if you want to remove the spool system; any other response is treated as NO and the next GASP prompt is issued.

NOTE

Do not use ^DA if GASP was scheduled from FMGR rather than from RTE since it is possible that JOBFIL is still open.

If any file is currently active because a job is being processed or a file is being outspooled, the deallocation cannot be performed. GASP reports the file name of the open file followed by the message: FILE OPEN OR LOCK REJECTED. It then terminates the DA command and issues the next GASP prompt. You may either purge the file with KS or you may wait for completion and then re-enter DA. As a general rule, you should wait for completion or abort any active job or outspool before entering DA. This allows active jobs and spools to complete and close all files.

If DA is successful, GASP issues the following message and then terminates:

**SPOOL IS DEAD!
END GASP**

Following this message, you must re-initialize the spool system before it can be used again. If you run GASP, it will automatically issue the first prompt of the initialization process rather than the standard GASP interactive prompt (^ or ↑). Refer to Section VII for a description of how to initialize the spool system with GASP.

Examples

1. †SD
 †DA
 KILL SPOOLING?NO ← *any response except YES terminates DA*
 †SU

2. †SD
 †DA
 KILL SPOOLING?YES
 SPOLØ4 ← *SPOLØ4 is open*
 GASP -8 FILE OPEN OR LOCK REJECTED

 †KS, SPOLØ4 ← *purge SPOLØ4*
 †SD ← *shut down system and then deallocate spooling*
 †DA
 KILL SPOOLING?YES

 SPOOL IS DEAD
 END GASP

5-20. COMPREHENSIVE SPOOLING EXAMPLE

The following comprehensive example presents an entire spooling session. First, program GASP is used to initialize the spooling system in Figure 5-2. Figure 5-3 shows jobs X, Y, and Z that are to be processed. Program JOB is used to inspool the jobs as shown in Figure 5-4. Finally, the left side of Figure 5-5 shows the FMGR commands as they would be processed and listed on the system console. The right side of Figure 5-5 shows program GASP running on logical unit 16 to display information about the job queue and spool status.

PROG6 and PROG8 are two programs that write to logical units 6 and 8 respectively. The severity code is set to 4 within jobs X and Y prior to purging files FILEX and FILEY. This ensures that the jobs will not be aborted if the files do not already exist.

```

16>RU,GASP
MAX NUMBER OF JOBS, JOB FILE DISC? 24,31
NUMBER OF SPOOL FILES (5 TO 80)? 20
SIZE OF SPOOL FILES (IN BLOCKS)? 24
NUMBER, LOCATION OF SPOOL FILES? 20,31
NUMBER, LOCATION OF SPOOL FILES? E
MAXIMUM NUMBER ACTIVE AND PENDING SPOOL FILES? 20
ENTER OUTSPOOL DESTINATION LU 6
ENTER OUTSPOOL DESTINATION LU 52
ENTER OUTSPOOL DESTINATION LU E
END GASP

```

Figure 5-2. GASP Initializes Spooling System

```

:JO,X
:PA,14,JOB X:BEGINNING NOW
:SV,4
:PU,FILEX::31
:SV,0
:CR,FILEX::31:3:1
:LU,6,FILEX::31,WRST,6
:RU,PROG6
:LU,6,0
:PA,14,JOB X:ENDING
:EO

:JO,Y
:PA,14, JOB Y: BEGINNING NOW
:SV,4
:PU,FILEY::31
:SV,0
:CR,FILEY::31:3:1
:LU,8,FILEY::31,WH,52
:RU,PROG8
:PA,14,JOB Y:HOLD ON FILEY TO LU 52
:CS,8,PA
:PA,14,JOB Y: ENDING
:EO

:JO,Z
:PA,14,JOB Z: DOING NOTHING
:EO

```

Figure 5-3. Jobs X, Y, and Z

```

16>RU,JOB,16
;:XE,X
;:XE,Y
;:XE,Z
;(control D)

```

Figure 5-4. Program JOB Used to Inspool Jobs X, Y, and Z

This is the command stream as it appears on the system console.

```
:JO,X
:PA,14,JOB X:BEGINNING NOW.
```

(JOB X is in a pause. GASP is run to examine the job queue and spool files)

This is program GASP being used to examine and modify the job queue and the outspool files. This is run on a separate terminal.

```
16>RU,GASP,16
```

```
^DJ
```

##	NAME	STATUS	SPOOLS
1	X	S 9999 A	4
2	Y	S 9999 R	
3	Z	S 9999 R	

```
^DS
```

LU	NAME	PRIORITY	JOB#	STATUS
--	X	0 0	1	--
6	SPOL04	9999	1	A

(JOBS X,Y and Z are in the job queue; X is active and Y and Z are ready. SPOL04 is set up as an outspool file for job X.)

```
TR
```

```
:CR,FILEX::31:3:1
:LU,6,FILEX,WRST,6
:RU,PROG6
:PA,14,JOB X: ENDING
```

(The :PU command is not listed because the severity code is 4. LU 6 has been associated with FILEX. PROG6 performs a write to LU 6 which goes into FILEX to be outspooled to LU 6.)

Figure 5-5. Command Stream and Use of GASP


```

^DJ
##      NAME      STATUS      SPOOLS

  1      X          S 9999  A
  2      Y          S 9999  R
  3      Z          S 9999  R
    
```

```

^DS
LU NAME  PRIORITY  JOB#  STATUS

-- X          0 0    1    --
  6 FILEX    9999    1    A
    
```

(FILEX has been set up for outspooling and is currently active.)

```

TR
:LU,6,0
:EO
    
```

(Job X ends. Job Y begins.)

```

:JO,Y
:PA,14,JOB Y:BEGINNING
    
```

```

^DJ
##      NAME      STATUS      SPOOLS

  2      Y          S 9999  A      4
  3      Z          S 9999  R
    
```

```

^DS
LU NAME  PRIORITY  JOB#  STATUS

-- Y          0 0    2    --
  6 SPOL04  9999    2    A
    
```

(Job X has been removed from the job queue. SPOL04 is set up for job Y.)

Figure 5-5. Command Stream and Use of GASP (Sheet 1 of 3)

```

TR
:CR,FILEY::31:3:1
:LU,8,FILEY,WH,52
:RU,PROG8
:PA,14,JOB Y:HOLD ON FILEY TO LU 52

```

(FILEY has been associated with LU 8, and placed on hold. It will not be placed in outspool queue until :EO or :CS command is processed.)



```

^DJ
##      NAME      STATUS      SPOOLS
      2      Y      S 9999 A      4
      3      Z      S 9999 R

```

```

^DS
LU NAME PRIORITY JOB# STATUS
-- Y      0 0      2      --
 6 SPOL04 9999      2      A
52 FILEY 9999      2      W

```

(FILEY is in wait state. User decides to abort job 3.)

```

^AB,3
^DJ
##      NAME      STATUS      SPOOLS
      2      Y      S 9999 A      4
      3      Z      S 9999 A

```

(The system places Z into active state, causing it to be removed from the job queue the next time FMGR scans the queue.)

Figure 5-5. Command Stream and Use of GASP (Sheet 2 of 3)

```

TR
:CS,8,PA
:PA,14,JOB Y: ENDING

      ^DJ
      ##          NAME          STATUS          SPOOLS
      2          Y             S 9999 A           4
      3          Z             S 9999 A
      ^DS
      LU NAME PRIORITY JOB# STATUS
      -- Y      0 0      2      --
      6 SPOL04 9999      2      A
      52 FILEY 9999      2      A

TR
:EO

      ^DJ
      ##          NAME          STATUS          SPOOLS

      ^DS
      LU NAME PRIORITY JOB# STATUS

      NO SPOOLS

      ^EX
      END GASP

```

Figure 5-5. Command Stream and Use of GASP (Sheet 3 of 3)

SECTION VI

SPOOL CONTROL THROUGH SMP CALLS



A single call to the routine SPOPEN takes a setup buffer established by the user to define the spool file and passes this buffer to SMP. Following the call to SPOPEN, the spool file is open. The spool logical unit associated with the spool file is returned in a SPOPEN parameter. This logical unit is one of those associated with a spool EQT at generation; SPOPEN causes SMP to set up the spool EQT entry.

Format

CALL SPOPEN(IBUFR,ISLU)	
IBUFR	Setup buffer; 16-word array defining spool file, spool attributes, spool priority, and outspool device.
ISLU	Spool logical unit number assigned by SMP returned here.

A spool setup must be established for each logical unit in your program to which you want to assign a spool logical unit equivalence. When SMP is scheduled by this subroutine, it establishes a record in file SPLCON (see format in Appendix C). Recall that SPLCON contains a record for each spool file currently active or queued for spooling. An EQT number is initialized for the spool and its corresponding spool logical unit is assigned. SMP passes back a non-zero logical unit number in ISLU if the call is successful. ISLU is set to a negative error code if the call is unsuccessful (see Appendix B).

Setup Buffer

You must set up IBUFR as described in Table 6-1. SMP adds the spool logical unit and status to the 16-word array IBUFR and writes it to a record in SPLCON.

This buffer establishes the file to be used for spooling and is used by SMP as the spool control record in file SPLCON. The user file is specified in words 2, 3, and 4, and this file must exist before it can be used by SMP.

Batch Checking

Batch checking is not normally requested by user programs. If batch checking is requested, only the program making this call may read records starting with a colon (:). If any other program attempts to read such a record, it will get an EOF on the first attempt and the error code IO22 on any subsequent attempts. Batch checking is normally set by program FMGR to prevent other programs from reading FMGR commands.

Disposition Flags

The disposition flags are similar to the attributes specified in :LU or changed by :CS (refer to paragraphs 4-9 and 4-10). If the entire word is set to zero, then the following default disposition is assumed: the file is unbuffered, input is not from batch job; both read and write are allowed, spool file is formatted with two-word record header, file is a user file; it is passed to outspool queue, and is purged when spooling is completed. The main difference between these attributes and those assumed by :LU is that SMP assumes a user file as the spool file in this case, assumes that both read and write are allowed, and purges the file upon completion of outspooling.

Table 6-1. IBUFR Format

WORD	CONTENTS																																
0	Batch input checking flag; $\neq 0$ if batch input checking wanted; = 0 if no batch input checking. When the flag is set, SMP places the calling program's ID segment address in word 0 when IBUFR is moved to SPLCON.																																
1	Spool logical unit number set by SMP when IBUFR is moved to SPLCON; do not set or set to zero.																																
2,3,4	6-character file name of user file to be used for spooling; may not be a spool pool file (SPOL01 through SPOL80).																																
5	Security code of file; set to value associated with file at creation; if no security, set to zero.																																
6	Cartridge reference number of cartridge containing file; set to value assigned to file at creation; if no cartridge specified, set to zero.																																
7	Octal number of device type for outspool device; i.e., if device is line printer (driver DVR12), set to 12B. This number is placed in the EQT table so that programs testing the device type will find it. If the spool is to be accessed by the Formatter with binary reads or writes, the type should be greater than 17B. If positioning such as back-spacing is to be done, it may be set to 23B (magnetic tape).																																
8	Disposition flags defining spool file characteristics as follows: <table style="margin-left: 40px; border-collapse: collapse;"> <tr> <td style="text-align: center;">15</td><td style="text-align: center;">14</td><td style="text-align: center;">13</td><td style="text-align: center;">12</td><td style="text-align: center;">11</td><td style="text-align: center;">10</td><td style="text-align: center;">9</td><td style="text-align: center;">8</td><td style="text-align: center;">7</td><td style="text-align: center;">6</td><td style="text-align: center;">5</td><td style="text-align: center;">4</td><td style="text-align: center;">3</td><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;">0</td> </tr> <tr> <td style="border: 1px solid black; text-align: center;">BU</td><td style="border: 1px solid black; text-align: center;">BI</td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;">W/R</td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;">ST</td><td style="border: 1px solid black; text-align: center;">SP</td><td style="border: 1px solid black; text-align: center;"></td><td style="border: 1px solid black; text-align: center;">HO</td><td style="border: 1px solid black; text-align: center;">SA</td> </tr> </table> <p>BU = 1 if buffering; = 0 if no buffering (no buffering is recommended since tests have shown it to be faster)</p> <p>BI = 1 if batch input (FMGR procedure file); = 0 otherwise (you should set to zero)</p> <p>W/R = 10 for write only; 01 for read only; 00 for write/read</p> <p>ST = 1 if standard file format; = 0 if spool file format (spool format is recommended if the file is to be outspooled as it retains all control requests)</p> <p>SP = 1 for spool pool file; = 0 for user file (do not set to 1, must be user file)</p> <p>HO = 1 to hold outspooling until file is closed; = 0 to pass to outspool queue immediately (only useful if word 15 set to an outspool logical unit). You should set HO = 1 if you expect to re-position file to be outspooled.</p> <p>SA = 1 to save file; = 0 to purge file (this action is taken when file is closed or, if queued for outspooling, when outspooling is complete)</p>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	BU	BI						W/R				ST	SP		HO	SA
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
BU	BI						W/R				ST	SP		HO	SA																		
9	Spool priority (1-9999) used to determine priority for outspooling; if set to zero, 9999 is assumed; used only if word 15 is set to an outspool logical unit.																																
10	<p>a. If the file is not to be outspooled this word is ignored.</p> <p>b. Null H (110₈) the file is put in HOLD status, the same as if a CS,H command had been issued by GASP. This "operator" hold may only be removed by the GASP CS command.</p> <p>c. If word 10 is not null H (110₈) then the file is put in R status and will be outspooled as soon as the HO bit is cleared in word 8 either by the initial set up or by the "Write EOF and queue for outspooling" or the "Queue for outspooling" request.</p>																																

Table 6-1. IBUFR Format (Continued)

11	Job number; only applicable if BI in word 8 set to indicate batch input; usually set to zero.
12-14	Set to zeros
15	Set to outspool logical unit; set to zero if no outspooling is desired at this time; a logical unit may be assigned later in the program.

Example

Set up to spool on the user file MYFILE (security code SC, cartridge 13) with outspooling to the line printer (logical unit 6, DVR12). The file is for write only, and is to be saved. The spool logical unit is retrieved in ISLU:

```

DIMENSION IUCB(144), IBUFR(16), FNAME(3)
EQUIVALENCE(ISECU, IBUFR(6)), (ICR, IBUFR(7))
DATA FNAME/2HMY, 2HFI, 2HLE/, ISIZE/20/, ITYPE/3/
DATA IBUFR/0, 0, 2HMY, 2HFI, 2HLE, 2HSC, 13, 12B, 401B, 10,
1      0, 0, 0, 0, 0, 6/
.
.
.
CALL CREAT(IUCB, IERR, FNAME, ISIZE, ITYPE, ISECU, ICR)
IF(IERR .LT. 0) GO TO 900
CALL CLOSE(IUCB)
CALL SPOPN(IBUFR, ISLU)
IF(ISLU .EQ. 0) GO TO 900
.
.
.
WRITE(ISLU, 200)
200  FORMAT("THIS OUTPUT IS SPOOLED")
.
.
.
CALL CLOSE(IUCB)
900  (error processing starts here)

```

*set values
for CREAT*

*values for
setup buffer*

create user file MYFILE

close file before calling SPOPN

set up MYFILE as spool file

*check for successful completion
of SPOPN*

6-4. SPOOL CONTROL

A set of EXEC calls may be used to assign new values for spool control. They assume that spool control values have been specified in IBUFR and that a setup call has been used to move IBUFR to the file SPLCON and set up the spool EQT.

The calls have a general form:

CALL EXEC(23,ISMP,*n*,ISLU)

where *n* is the request code for the procedure. The request code for spool setup is zero; the request codes for calls in this group range from 1 through 7. The call using request code 5 has two additional parameters, otherwise they follow the general form exactly.

ISMP is a three-word array containing the program name SMP

ISLU is the non-zero spool logical unit number returned by the spool setup call, SPOPEN; ISLU < 0 if the setup failed.

Table 6-2 summarizes the spool control functions.



Table 6-2. Spool Control Calls

FUNCTION	REQUEST CODE
Change purge to save	1
Change save to purge	2
Queue for outspooling	3
Write EOF and queue for outspooling	4
Change spool options	5
Set buffer flag	6
Clear buffer flag	7

Once you have set up the IBUFR array with default values these calls permit you to change them dynamically during execution of your program. In this way, IBUFR need not be completely re-specified for each new spool setup as long as the same spool file is used. All output to the same logical unit goes to the same file.

6-5. CHANGE PURGE TO SAVE

This call saves the spool file following outspooling.

Format

CALL EXEC(23,ISMP,1,ISLU)

ISMP Program name SMP.

ISLU Spool logical unit number returned by setup request.

Example

To set a flag in IBUFR to save FILEX:

```

DIMENSION IBUFR(16), ISMP(3)
DATA IBUFR/0,0,2HFI,2HLE,2HX ,0,0,128,0,0,0,0,0,0,0,0,0,0,0/
DATA ISMP/2HSM,2HP ,2H /
.
.
.
CALL SPOPN(IBUFR, ISLU) ← setup pool on user file FILEX
IF (ISLU .EQ. 0) GO TO 900
.
.
.
CALL EXEC(23, ISMP, 1, ISLU) ← set flag to save FILEX

```

↑ define IBUFR with
default values in
disposition flags

6-6. CHANGE SAVE TO PURGE

This call causes the pool file to be purged following outspooling.

Format

```
CALL EXEC(23, ISMP, 2, ISLU)
```

ISMP Program name SMP.

ISLU Pool logical unit number returned by setup request.

Since the default is to purge the pool file, this call is only meaningful if the save flag has been set in IBUFR.

Example

Assume the save flag has been set in IBUFR; to purge pool file FILEX:

```

DIMENSION IBUFR(16), ISMP(3)
DATA IBUFR/0,0,2HFI,2HLE,2HX ,0,0,128,1,0,0,0,0,0,0,0,0,0/
DATA ISMP/2HSM,2HP ,2H /
.
.
.
CALL SPOPN(IBUFR, ISLU) ← set up pool file and return pool
IF (ISLU .EQ. 0) GO TO 900                                logical unit
.
.
.
CALL EXEC(23, ISMP, 2, ISLU) ← set flag to purge FILEX

```

↑ save flag is set

Example

Assume that IBUGR has been set up exactly as shown in the preceding example, after writing data to the spool file, to then write an EOF and outspool it:

```

      CALL SPOPN(IBUGR,ISLU) ←————— setup spool file
      IF (ISLU .EQ. 4) GO TO 900
      .
      .
      .
200  WRITE(ISLU,2000), (DATA(I), I=1,5) ← write data to spool file
      FORMAT("DATA:",5(G10,3))
      CALL EXEC(23,ISMP,4,ISLU) ←————— close file and outspool it

```

6-9. CHANGE SPOOL OPTIONS

This call assigns an outspool logical unit and an outspool priority.

Format

```
CALL EXEC(23,ISMP,5,ISLU,NOLU,NPR)
```

ISMP Program name SMP.

ISLU Spool logical unit returned by setup request.

NOLU Logical unit of output device.

NPR Priority for spool file in outspool queue.

A driver type should always be specified in IBUGR corresponding to the actual device to which output is sent. (Refer to Appendix C for a list of driver types). The logical unit specified in NOLU should be the driver type specified in IBUGR; that is, a line printer logical unit if a line printer driver was specified, a magnetic tape logical unit if a magnetic tape driver, and so forth. In the examples in this section, a driver type is always specified in IBUGR and is usually DVR12, the line printer driver.

6-10. SET BUFFER FLAG

Data transmitted to or from spool files may be buffered or unbuffered. This call sets a flag in the spool EQT table to specify buffering.

Format

```
CALL EXEC(23,ISMP,6,ISLU)
```

ISMP Program name SMP.

ISLU Spool logical unit returned by setup request.

6-12. SPOOL POSITIONING

Data is read from or written to spool files one record at a time. The position of the record currently being written or read can be retrieved with one SMP call; another call may be used subsequently to reposition the file to the record position saved in the first call.

6-13. RETRIEVE RECORD POSITION

This call retrieves the current record position in the spool file. The information must be retrieved by a subsequent call to RMPAR.

Format

```
CALL EXEC(23,ISMP,8,ISLU)
```

```
CALL RMPAR(IPRAM)
```

ISMP Program name SMP.

ISLU Spool logical unit returned by spool setup request.

IPRAM 5-word array containing pointers to record position:

word 1 =	}	contain an internal coding of the current position of the referenced file
word 2 =		
word 3 =		
word 4 = not used but should be included in array		
word 5 = not used but should be included in array		

The information retrieved in IPRAM can be set into the three parameters ITRK, ISEC, and IEXT used in the next spool call to (paragraph 6-14) position the disc at the indicated position.

6-14. CHANGE RECORD POSITION

This call positions the spool file at a particular record indicated by its track/sector address, the location within the sector, and the extent number (if any) of the file. This information can be retrieved by the previous SMP call (paragraph 6-13).

Format

```
CALL EXEC(23, ISMP, 9, ISLU, ITRK, ISEC, IEXT)
```

ISMP Program name SMP.

ISLU Spool logical unit returned by setup request.

ITRK

ISEC

IEXT

ITRK, ISEC, IEXT are words 1, 2, and 3 (respectively) from the RETRIEVE RECORD POSITION call (see paragraph 6-13).

In order to use this call, retrieve the location of the current record by a previous EXEC call and call to RMPAR, and save this information so it can be used in the call to change the record position. The information specified in this call must have been obtained in the current setup of the file.

Example

This example illustrates both the record retrieval and positioning calls. To save the location of current record (when data value is -1); then reposition to the saved location (when data value is -2):

```

DIMENSION IBUFR(16), ISMP(3), IP(5), DATA(5)
DATA IBUFR/0,0,2HOU,2HTF,2HIL,RT,13,12B,403B,10,6*0/
DATA ISMP/2HSM,2HP,2H / define IBUFR _____
.
.
.
CALL SPOPN(IBUFR, ISLU) ← setup spool file OUTFIL
IF (ISLU .EQ. 0) GO TO 900
.
.
.
22 READ(1,*) (DATA(I), I=1,5) ← read data from lu 1
   IF (DATA .EQ. 0) GO TO 50 ← terminate on zero input
   IF (DATA .NE. -1) GO TO 24
   CALL EXEC(23, ISMP, 8, ISLU) ← get current record position
   CALL RMPAR(IP)

24 IF (DATA .NE. -2) GO TO 26
   CALL EXEC(23, ISMP, 9, ISLU, IP(1), IP(2), IP(3)) ← reposition file _____
26 WRITE(ISLU, 200) DATA ← write data to spool file
200 FORMAT(" DATA :", 5(G10.3))
   GO TO 22

50 (end-of-data termination)

```

6-15. SPOOL CALL EXAMPLE

The program SPOLX illustrates an application of SMP calls. The program reads data from logical unit 1 and writes it to a spool file. The data is monitored so that when a value indicates a change of state, the next record location is saved and at a second value, the record is re-written from that location. When there is no more data, then the file is closed, an end-of-file is written to it and it is placed in the outspool queue. The file is saved and its contents can be printed at any time.

The particular use for such an application is in situations when data comes in at a relatively slow rate and where the data must be monitored so that only "good" data is written to a device. If the process is monitored over long periods of time, this would make the output device unavailable to other users during that time. This problem can be alleviated by writing to FMP files or, as in this example, by outspooling to a spool-maintained file. The advantage of spooling is that I/O can be handled by standard EXEC calls or formatted I/O calls and the file physically outspooled to the device at some later time.

The program is scheduled with:

```
RU, SPOLX, P1, P2, P3, P4
```

where

p1 is the input logical unit

p2-p4 is the name of the data file to be spooled

When the program has completed, the results can be physically outspooled to the line printer by running the following job:

```
:JOB, DUMP
:LU, 51, FILENM, , 6
:EOJ ↑----- 51 is a dummy value following constraints of LU command

C
C
C   PROGRAM SPOLX, 3, 99
C
C   DIMENSION IRTNBF(5), IPRAM(5), ISMP(5), IOCB(144)
C   DIMENSION DATA(5), ISET(16)
C   INTEGER FNAME(3)
C
C   EQUIVALENCE (LU, IPRAM)
C   EQUIVALENCE (FNAME, ISET(3))
C   EQUIVALENCE (ISECU, ISET(6)), (ICR, ISET(7))
C
C   DATA ITYPE/3/, IBCHK/0/
C   DATA ISMP/2HSM, 2HP, 2H /
C   DATA ISET/0, 0, 2HOU, 2HTF, 2HIL, 2HRT, 13, 12, 403B, 10,
C   1          0, 0, 0, 0, 0, 0/
C
C
C   CALL RMPAR(IPRAM)
C   IF(LU.EQ.0)LU=1
```

```

C
C
C CHECK SCHEDULE PARAMETERS FOR A FILE NAME
C
      IF(IPRAM(2).EQ.0)GO TO 12
      FNAME=IPRAM(2)
      FNAME(2)=IPRAM(3)
      FNAME(3)=IPRAM(4)
C
C CREATE OUTPUT FILE (10 BLKS - IT'LL BE EXTENDED IF NECESSARY)
C ALSO, PURGE OLD VERSION IF NECESSARY
C
12   CALL PURGE(IDCDB,IERR,FNAME,ISECU,ICR)
      ISIZE=10
      CALL CREAT(IDCDB,IERR,FNAME,ISIZE,ITYPE,ISECU,ICR)
      IF(IERR.GE.0)GO TO 15
      WRITE(LU,101)IERR
101  FORMAT("/SPOLX: FILE CREATE ERROR:"I5)
      GO TO 999
15   CALL CLOSE(IDCDB,IERR,0)
C
C      NO BATCH INPUT CHECK
C      TO FILE SPECIFIED ON CR 13, SC = "RT"
C      DRIVER TYPE 12 SINCE WE INTEND TO LIST ON L.P. LATER
C      ATTRIBUTES OF: SAVE, HOLD, WRITE ONLY
C      PRIORITY 10
C      OUTSPOOL LU: 6
C
C (SEE DATA STATEMENT FOR PRE-SET VALUES)
C
      CALL SPOPN(ISET,LUSPOL)
      IF(LUSPOL.GT.0)GO TO 20
      WRITE(LU,110)LUSPOL
110  FORMAT("/XPOLX: SPOOL SET-UP ERROR, ABORT!"I4)
      GO TO 999
C
C ISSUE TOP-OF-FORM TO THE OUTPUT
C
20   ICHWD=LUSPOL+1100B
C
      CALL EXEC(3,ICHWD,-1)
C
C
C NOW, START DATA INPUT, PROCESSING, AND OUTPUT TO THE SPOOL
C THIS SECTION WOULD BE REPLACED BY A MORE FUNCTIONAL
C SET OF CODE UNDER USUAL CIRCUMSTANCES
C
C IN THIS EXAMPLE WE WILL MAKE USE OF THE FIRST DATA ITEM IN THE
C LINE OF FIVE TO SIGNAL THE PROGRAM IN THE FOLLOWING MANNER:
C
C = 0   TERMINATE
C
C = -1  FLAG AS START OF 'CHANGE POINT'. E.G. THE TEST OR
C       PROCESS IS BEING ALTERED AND IS NO LONGER IN
C       EQUILIBRIUM. DATA IS GOOD FOR LOGGING ONLY, AND NOT
C       GOOD FOR PERFORMANCE RESULTS FOR THE PROCESS.

```


SMP Calls

```

C
C = -2   OVER-WRITE DATA FROM 'CHANGE POINT'.  E.G. THE LOGGED DATA
C        IS NO LONGER NECESSARY AND THE FILE SHOULD CONTAIN ONLY
C        GOOD PERFORMANCE DATA.
C
C = -3   THE OUTPUT DEVICE IS NOW AVAILABLE.  START OUTPUT
C        IMMEDIATELY
C
C
22      WRITE(LU,100)
100     FORMAT("/SPOLX: ENTER 5 DATA VALUES: _")
        READ(LU,*)(DATA(I),I=1,5)
        IF(DATA.EQ.0.)GO TO 50
        IF(DATA.NE.-1.)GO TO 24
        CALL EXEC(23,ISMP,8,LUSPOL)
        CALL RMPAR(IRTNBF)
        ITRK=IRTNBF
        ISEC=IRTNBF(2)
        IXTNT=IRTNBF(3)
        GO TO 26
24      IF(DATA.NE.-2.)GO TO 25
        CALL EXEC(23,ISMP,9,LUSPOL,ITRK,ISEC,IXTNT)
        GO TO 26
25      IF(DATA.NE.-3.)GO TO 26
        CALL EXEC(23,ISMP,5,LUSPOL,6,10)
        CALL EXEC(23,ISMP,3,LUSPOL)
        GO TO 26
26      WRITE(LUSPOL,200)ICNT,DATA
200     FORMAT("  ITEMS" I3"  :",5(G10,3))
        GO TO 22
C
C CLEAN-UP
C
C ISSUE TOP-OF-FORM ON OUTPUT
C
50      CALL EXEC(3,ICNWD,-1)
C
C CLOSE THE SPOOL AND PASS IT TO SMP
C
        CALL EXEC(23,ISMP,4,LUSPOL)
C
C TERMINATE
C
999     WRITE(LU,199)
199     FORMAT("/SPOLX: DONE!"/)
        END
0123     END$

```

SECTION VII

FMP AND SM CONFIGURATION AND INITIALIZATION



Summary Section VII

	PAGE
FMP Configuration	7-1
FMGR Initialization	7-3
SM Configuration	7-4
GASP Initialization	7-9

FMP AND SM CONFIGURATION AND INITIALIZATION

SECTION

VII

7-1. INTRODUCTION

The Batch-Spool Monitor consists of two parts: the File Management Package and the Spool Monitor. Each of these parts must be configured into the RTE system; the File Management Package is required, the Spool Monitor only if spooling is to be used.

Configuration is done when the system is originally generated. There is no provision for on-line loading. Refer to the RTE operating system manual for a complete description of the off-line generation process. Refer to the appropriate RTE On-Line Generation Manual for a description of the on-line generation process. The particulars of generating the File Management Package and the Spool Monitor are described in this section.

In addition to configuring the two parts of the Batch-Spool Monitor, the program FMGR (part of FMP) and the program GASP (part of SMP) must be initialized. Initialization of FMGR assigns system tracks to D.RTR, sets up file space boundaries on the system disc, assigns directory tracks, and, optionally, assigns a master security code.

Initialization of GASP sets up and initializes the files JOBFIL, SPLCON, and the spool pool files.

Configuration of the Batch-Spool Monitor for RTE-II, RTE-III, and RTE-IV differs only in two respects:

- Page requirements of RTE-III and RTE-IV force FMGR to a background memory size of 7K words; 6K is needed for the program and 1K for the base page. RTE-II requires 6K words.
- Three programs in the Spool Monitor (SMP, EXTND, and SPOUT) are supplied with different program types in RTE-II, RTE-III and RTE-IV. An additional module (SP.CL) is supplied with RTE-III and RTE-IV.

Apart from these differences, configuration under the two systems is the same. Initialization of FMGR and GASP are identical whether the system is RTE-II, RTE-III, or RTE-IV.

7-2. FMP CONFIGURATION

During the program input phase of system generation, the FMP programs are loaded into the system.

7-3. PARAMETER INPUT PHASE

The parameter input phase of generation allows you to change the program type and/or priority of the two programs supplied with FMP. These programs, FMGR and D.RTR, are

Installation and Initialization

assigned a type and priority in the NAM statement of each. If you want a different type or priority, you may enter them when the system generator issues the prompt PARAMETERS. The response is the form:

```
program name,type [,priority [,execution interval]]
```

execution interval is not applicable to either FMGR or D.RTR since these programs are never scheduled by time intervals.

FMGR Parameters

FMGR is supplied as a disc-resident background program (type 3) with a priority of 90. It is segmented into nine parts and requires 6K words of memory for the program and 1K for the base page in RTE-III and RTE-IV.

D.RTR Parameters

Subroutine D.RTR is supplied as a foreground disc resident program (type 2) with a priority of 1. It requires a few words over 1K words of memory. If space permits, it is recommended that it be made memory resident by changing the program type to 1 during the parameter input phase of generation. This increases its speed. When D.RTR is memory resident, its boundary may precede the page boundary by 200 (octal) words if no links are used, since there is a 128_{10} (200_8) word buffer here.

If its priority is changed, it is essential that the new priority be higher than that of any other program using the File Management Package.

Example

To change the FMGR and D.RTR parameters during system generation:

```
PARAMETERS ← System generator prompt  
FMGR, 3, 30 ← FMGR priority is increased  
from 90 to 30; program type  
is not changed from 3.  
D.RTR, 1 ← D.RTR is made memory re-  
sident; priority is unchanged.
```

More program types are available in RTE-III and RTE-IV than in RTE-II. Refer to the respective operating system manuals for complete lists of available program types.

Protecting Peripheral Cartridges

If you want to protect the FMP peripheral cartridges from alteration by user programs, you may enter a change to an entry point when the generator issues the prompt CHANGE ENTS?. To protect these cartridges, specify:

```
$PDSK, AB, 1
```

7-4. FMGR INITIALIZATION

The first time the system is started up after system generation, FMGR runs automatically. It displays the message FMGR 002 and then issues the standard prompt (:). This is a request to initialize the program by assigning specific system tracks to FMGR. Before it is initialized, FMGR obtains all the available tracks on the system and auxiliary discs and assigns them to itself. After it is initialized, it owns only those tracks specifically assigned to it. Thereafter, each time the system is loaded from disc (booted up), it recovers these tracks automatically and no further initialization is required. After initialization and each time the system is restarted FMGR transfers to the WELCOM file — a file containing user specified commands (see Appendix E).

FMGR initialization is performed with the FMGR command IN used to initialize FMGR cartridges (refer to paragraph 3-18 for a full description of IN). This command is specified in response to the prompt FMGR 002 for system disc initialization, or in response to the prompt FMGR 003 for auxiliary disc initialization. In order to use FMGR, the starting track of the FMGR tracks on the system disc must be assigned. The starting track must be at least 8 tracks greater than the last track used by the system. This provides working tracks between the system and FMGR. System size in tracks and sectors is reported at the end of the generation dialogue. The system always starts at track 0.

If auxiliary disc tracks are not to be assigned to FMGR, the IN command should still be specified in response to FMGR 003, but the cartridge reference number should be specified as 0.

The last track on the system disc is always assigned to D.RTR and is reserved for the FMGR cartridge and file directories. If the auxiliary disc has been initialized to contain FMGR files, then its last track is also assigned to D.RTR and is reserved for the FMGR file directories. If more directory tracks are needed, these must be specified at this time. This is also the time to specify a master security code for FMGR files, an ASCII identifier for the disc, and any bad tracks.

NOTE

If you assign a master security code, remember it since it cannot be recovered.

When a successful initialization is completed, FMGR assigns the tracks as specified in the IN command parameters. FMGR then transfers to the WELCOM file which will not exist at this time — so error FMGR - 006 is generated and control is passed back to the system TTY.

Example

To initialize the system disc for FMGR starting at track 50 and to initialize the auxiliary disc with no tracks assigned to FMGR:

```
*RU, FMGR ← schedule FMGR
FMGR 002 ← request system disc initialization
: IN, XX, - 2, 2, SD02FM, 50 ← start at track 50; set master security to XX
FMGR 003 ← request auxiliary disc initialization
: IN, XX, - 3, 0 ← do not assign auxiliary tracks to FMGR (cr=0)
FMGR-006 ← FMGR fails to find WELCOM and transfers
: ← to the TTY system
```

If there is no auxiliary disc, FMGR terminates after the system disc initialization without requesting auxiliary disc initialization.

Installation and Initialization

If you respond with a command other than IN to the prompts FMGR 002 or FMGR 003, the error message FMGR 004 is issued. If you correctly enter IN but request a starting track that is not available, FMGR 005 is issued. The first available track and sector is reported if you enter ?? to expand the message.

7-5. SM CONFIGURATION

As with FMP configuration the Spool Monitor programs are loaded into the system during the program input phase of system generation. The program type or priority of each program can be changed during the parameter input phase.

Spooling Configuration

If spooling is used, additional memory is required. The Spool Monitor Program needs 3K words of foreground disc memory. DVS43 and EXTND programs need 1.5K words of system and foreground memory. If SPOUT is memory-resident, it requires .5K words of foreground. Each spool EQ requires 34 words (15 EQT + 18 EQT + 1 Interrupt Table). Each spool logical unit requires 2 words.

Since spooling uses System Available Memory (SAM), enough SAM should be allocated during generation to service spooling. The exact requirements depend on the application, but 1024 (decimal) words of SAM is a practical minimum for most systems. More SAM is required if several programs use it simultaneously.

In addition, the Spool Monitor requires entries during the table generation phase. At this time, class numbers are reserved for spooling and other functions, the logical unit switch table is allocated, and resource numbers are specified. Then entries for the Spool Monitor driver (DVS43) must be made in each of the three tables: EQT, Device Reference, and Interrupt.

During the partition definition phase of system generation, you must make sure that there is sufficient System Available Memory to handle the SPOUT program outspool requirements.

7-6. PARAMETER INPUT PHASE

The Spool Monitor programs are loaded from tape with the priority and program type shown in Table 7-1.

Table 7-1. Standard Program Parameters

PROGRAM	PRIORITY	PROGRAM TYPE		
		RTE-II	RTE-III	RTE-IV
JOB	30	2 (foreground disc res.)	2 (real-time disc res.)	2 with TB2
GASP	30	19 (background disc res.)	19 (background disc res.)	19 with TB2
SMP	30	2 (foreground disc res.)	18 (real-time disc res. with SSGA)	18,SSGA,TB2
EXTND	10	1 (core resident)	17 (memory resident with SSGA)	17,SSGA,TB2
SPOUT	10	1 (core resident)	17 (memory resident with SSGA)	17 with TB2
DVS43	—	0 (system module)	0 (system module)	System Driver Area*
SP.CL	—	(not used by RTE-II)	30 (SSGA module)	30 SSGA

TB2 = Table Area II SSGA = Subsystem Global Area *With M option specified.

Optimal performance is provided by these system supplied program types. For some programs, the type code may be changed during configuration if the following rules are observed:

- JOB - For RTE-II must be foreground disc resident (type 2) to avoid competing for memory with FMGR.
 - For RTE-III and RTE-IV may be any disc-resident program type as long as JOB does not compete for the same partition with FMGR. If both JOB and FMGR are background disc resident, there should be enough partitions to avoid competition.
- GASP - For all versions of RTE should be background disc resident because GASP runs only under operator control.
- SMP - For RTE-II must be foreground disc resident (type 2).
 - For RTE-III and RTE-IV should be real-time disc resident (type 18); it may be background disc resident (type 19) but it must be able to access SSGA. Must not be memory resident.
 - For RTE-IV must have Table Area II access.
- EXTND - For RTE-II must be foreground memory resident (type 1). *DO NOT CHANGE THE PROGRAM TYPE.*
 - For RTE-III and RTE-IV must be memory resident (type 17) with access to SSGA.
 - For RTE-IV must have Table Area II access.
- SPOUT - For RTE-II should be foreground core resident (type 1); it may be foreground disc resident (type 2) but this slows the system severely.
 - For RTE-III and RTE-IV should be memory resident (type 17); it may be disc resident (types 18 or 19) but this will slow the system.
 - For RTE-IV must have Table Area II access.
 - For all systems, SPOUT and D.RTR must *not* be in the same disc area.
- DVS43 - For both RTE-II and RTE-III must be system resident (type 0). *DO NOT CHANGE THE PROGRAM TYPE.* During the EQT phase of the RTE-IV generation, M must be specified. This indicates that DVS43 does its own mapping.
- SP.CL - For RTE-III and RTE-IV only provides a spool communication area that must be in the SSGA module (type 30). *DO NOT CHANGE THE PROGRAM TYPE.*

Example

Change JOB and SMP to memory resident programs in RTE-III:



```

PARAMETERS ← System generator prompt
JOB , 1
SMP , 17 ← memory resident with access to SSGA

```


7-7. TABLE GENERATION PHASE

During the system generator interaction, the first questions ask for system resource allocation:

# OF I/O CLASSES?	2	class numbers; one for outspooling and one for communication with SMP, in addition to any numbers required by the rest of the system.
# OF LU MAPPINGS	2+n	logical unit switch table entries; one for the inspool logical unit and one for the outspool, plus one each for the maximum number of LU commands expected in any job.
# OF RESOURCE NUMBERS	4	resource numbers; one for logical unit locking during outspooling, two for locking files JOBFIL and SPLCON, and one for resource waiting, plus any numbers required by the rest of the system.

Next, entries in the three tables must be set up for spooling. At least six entries are needed in each of the tables EQT, Device Reference, and Interrupt. The EQT table entries for spooling are entered:

EQT *nn*?
xx,DVS43,X=18,M

M specified for RTE-IV only

nn is the number of the EQT entry

xx is a unique unused select code in the range 10 through 77 (octal)

DVS43 is the spool driver which must have an EQT extension of 18 words (X = 18).

M specifies that DVS43 does its own mapping (RTE-IV only).

When the EQT entries are complete, the EQT entry number for each entry must be specified in the Device Reference Table:

lu =EQT?
nn

lu is the logical unit number to be associated with *nn*

nn is the EQT entry number of an entry for which DVS43 was specified.

Finally, the Interrupt table entries are made in the form:

xx,EQT,*nn*

xx is the select code specified for the EQT entry *nn*.

Example

To enter the system resources and table entries for six spool files:

```

*# OF I/O CLASSES?
16 ←————— 2 for spooling + 14 for rest of system

*# OF LU MAPPINGS?
8 ←————— total of 8 logical unit switches

*# OF RESOURCE NUMBERS?
32 ←————— 4 for spooling + 28 for rest of system
    
```

```

BUFFER LIMITS (LOW, HIGH)?
100,400
    
```

```

:
:
    
```

```

* EQUIPMENT TABLE ENTRY
    
```

```

:
:
    
```

```

EQT 20?
72,DVS43,X=18,M
    
```

```

EQT 21?
73,DVS43,X=18,M
    
```

```

EQT 22?
74,DVS43,X=18,M
    
```

```

EQT 23?
75,DVS43,X=18,M
    
```

```

EQT 24?
76,DVS43,X=18,M
    
```

```

EQT 25?
77,DVS43,X=18,M
    
```

```

EQT 26?
    
```

```

/E
    
```

} 6 entries in EQT table for DVS43.
The M is specified in RTE-IV only. This indicates that DVS43 does its own mapping.

```

* DEVICE REFERENCE TABLE
:
:
    
```

```

51 = EQT #?
20      * SPOOL LU
52 = EQT #?
21      * SPOOL LU
53 = EQT #?
22      * SPOOL LU
54 = EQT #?
23      * SPOOL LU
55 = EQT #?
24      * SPOOL LU
56 = EQT #?
25      * SPOOL LU
:
:
* INTERRUPT TABLE
:
72,EQT,20
73,EQT,21
74,EQT,22
75,EQT,23
76,EQT,24
77,EQT,25
/E

```

} 6 spool logical units

} 6 spool entries in interrupt table

Note that although select codes are assigned (72 through 77 in example), no I/O devices should be present in these slots.

7-8. PARTITION DEFINITION PHASE

System Available Memory is defined during the partition definition phase of generation (RTE-III) or after foreground memory resident program definition (RTE-II). In both systems, opportunity is provided at this time to increase the allocation for System Available Memory.

One consideration is that the SPOUT program attempts to keep four requests in System Available Memory for each device to which it is outspooling. For this reason, enough System Available Memory must be provided to handle at least four of the longest expected outspool records. To calculate this space, assume a maximum record size of 68 words plus a 10 word header for each record. Thus, 78 words times 4 records = the minimum System Available Memory required by SPOUT.

The dialogue for increasing System Available Memory is:

RTE-II	RTE-III	RTE-IV
SYS AVMEM <i>mem loc</i>	SYS AV MEM: <i>size</i> WORDS	SYS AV MEM: xxxxx WORDS
CHANGE SYS AVMEM?		1ST PART PG xxxxx
<i>new loc</i>	1ST DSK PG <i>mem loc</i>	CHANGE 1ST PART PG?
BIG BOUNDRY <i>mem loc</i>	CHANGE 1ST DSK PG?	
CHANGE BG BOUNDRY?	<i>new loc</i>	
<i>new loc</i>	SYS AV MEM: <i>new size</i> WORDS	

To increase System Available Memory, you may raise the low address of the background boundary (RTE-II) or of the first disc-resident program (RTE-III and RTE-IV).

7-9. GASP INITIALIZATION

The first time GASP is run it initializes the spool system; thereafter it displays the initialization dialogue only if the spool system was removed by the GASP DA command or has been lost for any other reason. When GASP is scheduled, the scheduling program checks whether the file JOBFIL exists and if it does, the standard GASP prompt is issued. If not then the GASP initialization dialogue is started.

The purpose of GASP initialization is to set up the spool directory files JOBFIL and SPLCON, and the spool pool files. (See Appendix C for format and meaning of records in JOBFIL and SPLCON).

GASP is initialized by responding to the following prompts.

MAX NUMBER OF JOBS, JOB FILE DISC?

The response to this question establishes the definition of JOBFIL. Each job entry requires a 16-word record in JOBFIL. These job records are preceded by 16 records of job queue information (1 word/job) and two general information records. The maximum number of jobs allowed by the queue is 254 (16 records times 16 words = 256, less 2 words for a count and resource number). If the maximum number of jobs you expect is over 126, you might as well enter 254 since little disc space is saved unless you have fewer than 126 jobs.

The second part of the question "JOB FILE DISC" asks you to enter the cartridge reference number of the cartridge used for JOBFIL and SPLCON files. You should choose a cartridge that:

- is always in the system (a fixed platter)
- you won't want to pack unless you can shut down spooling while you pack.

Also, if you want to avoid contention for the disc, you can choose a cartridge that is used only for spooling. In order to save FMGR termination time, any cartridge containing JOBFIL should be the first cartridge in the directory on the system cartridge. Further, when a cartridge is used both for spool and user files, the user files may be interspersed with spool file extents. When the spool system is finished, the extents are purged from the disc. In this case, the user files created during execution of the job might have to be repacked (see PK command, paragraph 2-60) to recover full use of the cartridge.

CAUTION

Be certain that any cartridge specified for spooling files is currently mounted (see paragraph 2-55).

NUMBER OF SPOOL FILES (5 TO 80)? SIZE OF SPOOL FILES (IN BLOCKS)?

A number of considerations affect the responses to these questions:

- a) Bear in mind that any job inspooled from a device will be stored in one of these files. It makes no sense to store (typically) 30 or 40-line jobs in a 500-block file.

Installation and Initialization

- b) Nor is it useful to have a small number of files, thereby limiting the number of such inspoiled jobs. Allow enough spool pool files to accommodate the anticipated inspool load and outspool load, and match this (+ the anticipated load of inspoiled user files) to a reasonable number of allowable jobs. Note that the maximum number of jobs is limited both by the number of available pool files and by the number of available job slots in JOBFIL.
- c) If the files are small, extents are automatically created as needed for outspooling. The price is additional overhead and a directory peppered with extent entries.

The proper balance between size and number can be obtained by planning disc and directory resources as follows:

given: N = available disc tracks for spool files

D = # of FMP directory entries available on the disc (about 380 per directory track)

S = size of spool files (blocks)

P = number of spool pool files on the disc

Plan the disc and the directory to have the same capacity:

The number of extents that will fit on the disc:

$$\frac{N * 48}{S} - P \quad (\text{for 7900 disc unit})$$

should equal the number of extents that can be accommodated in the directory, D . Or, having picked P on the basis of the possible # of jobs and outspools, find their size from:

$$S = \frac{N * 48}{D + P}$$

Suppose there are 180 available tracks and directory for 300 entries, if you require 20 spool files, their size should be:

$$S = \frac{180 * 48}{300 + 20} = 27 \text{ blocks}$$

You might choose 24 blocks (half a track) for aesthetic reasons.

NUMBER, LOCATION OF SPOOL FILES?

Enter the number of spool files to be placed on a particular cartridge in the form nn,cr where nn is the number of files and cr is the cartridge reference or logical unit number. This prompt is repeated until you enter E. The total number of spool files allocated to all cartridges should equal the number of spool files specified in response to the second prompt. If spool files are spread over several discs, the size calculation for the prior response should be made for each disc used.

If a GASP 2 error is encountered after ending the responses to this question, the question will be repeated with all previous responses to the question being ignored by GASP.

MAXIMUM NUMBER ACTIVE AND PENDING SPOOL FILES

ENTER OUTSPOOL DESTINATION LU

The responses to these two questions define the outspool directory file SPLCON. Each outspool logical unit establishes an 8-record, 16-word per record, queue in SPLCON with a maximum of ten logical units. Each outspool logical unit queue can accommodate up to 63 files.

Do not use logical unit 1 as an outspool device.

The spool system contains a Queue depth with each outspool LU. This is the number of requests SPOUT attempts to keep in the queue for each LU it is outspooling to. The number may be between 1 and 15 inclusive, and is entered as the second parameter to the

“ENTER OUTSPOOL DESTINATION LU”

question. If there is no entry or zero is entered 4 is used. The number entered here should reflect the relative time it takes to find and extent vs. the time it takes to process a line of output to the given LU. As guide lines we might recommend:

punch - 2
 2607 printer - 3
 2767 printer - 6
 mag tape - 15

Too big a number will cause needless use of buffer memory. Too small a number will cause short pauses in output while the system searches for extents.

Each entry for an active or pending outspool requires one 16-word record, with a maximum of 256 allowed. An entry is opened under the following conditions:

- A file is sent to be outspooled
- A spool pool file is opened for outspooling
- A user file is being processed
- A spool pool file is used for inspooling or processing
- LU equivalence is made to a file

In other words, an entry is used anytime a disc file is used by the spool system.

The important points are:

- a) An entry is required for each pending job in a spool pool file.
- b) An entry is required for each outspool, be it in a spool pool file or a user file. Heavy usage of outspooling could detract from availability of entries for jobs and therefore from full use of JOBFIL. (Why allow a large number of jobs and not enough SPLCON entries?)

A rule-of-thumb would be to respond with a number about 1.5 or 2 times the number of spool pool files. It would be better, however, to anticipate the loading of user files onto the spooling process, in order to allow sufficient SPLCON entries to accommodate them.

Installation and Initialization

The prompt asking for an outspool logical unit is repeated until you type E to terminate. You should enter as many devices as you expect to use for output and then type E. The logical units should correspond to actual devices since the Spool Monitor only writes output to a device specified here. Logical unit 6, the list device, must always be entered and it is most efficient to enter it first.

After a successful initialization, GASP terminates with the message:

END GASP

The next time GASP is scheduled, it issues the standard GASP prompt (↑) or (∧).

It sometimes happens during GASP initialization that a duplicate file name is found for either SPLCON or JOBFIL. In this case, GASP issues the message:

DUP FILE NAME *filename*. DEINITIALIZE?

You may then answer YES so that all spool files will be closed. You may re-initialize by running GASP again. Deinitializing may take considerable time if multiple cartridges are currently mounted.

CAUTION

Be certain that all cartridges required by the spool system are mounted (See MC command, paragraph 3-55) before running GASP.

Example

The following example is a typical initialization where a maximum of 20 jobs are expected:

```
*ON,GASP
MAX NUMBER OF JOBS,JOB FILE DISC? 20,13
NUMBER OF SPOOL FILES (5 TO 80)? 40
SIZE OF SPOOL FILES (IN BLOCKS)? 24
NUMBER,LOCATION OF SPOOL FILES? 40,13
NUMBER,LOCATION OF SPOOL FILES? E
MAXIMUM NUMBER ACTIVE AND PENDING SPOOL FILES? 50
ENTER OUTSPOOL DESTINATION LU 6
ENTER OUTSPOOL DESTINATION LU 4
ENTER OUTSPOOL DESTINATION LU E
END GASP
```

APPENDIX SECTION

APPENDIX SUMMARY

Appendix	Title	Page
A	HP Character Set	A-1
B	FMP Error Codes	B-2
	FMGR Error Codes	B-5
	GASP Error Codes	B-10
	Spool Errors	B-14
C	Standard Logical Units	C-1
	Standard Driver Types	C-1
	Data Control Block Format	C-2
	Cartridge Directory Format	C-4
	File Directory Format	C-5
	JOBFIL Format	C-7
	SPLCON Format	C-9
	Disc File Record Formats	C-12
	Non-Disc Record Formats	C-14
D	Cartridge Formatting	D-1
E	FMGR Copies	E-1
F	Global Equivalence Table	F-1

HP CHARACTER SET

APPENDIX

A

BITS		COLUMN	0 ₀₀	0 ₀₁	0 ₁₀	0 ₁₁	1 ₀₀	1 ₀₁	1 ₁₀	1 ₁₁
b ₇	b ₆ b ₅	ROW	0	1	2	3	4	5	6	7
0	0 0 0	0	NUL	DLE	SP	0	@	P	'	p
0	0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0	0 1 0	2	STX	DC2	"	2	B	R	b	r
0	0 1 1	3	ETX	DC3	#	3	C	S	c	s
0	1 0 0	4	EOT	DC4	\$	4	D	T	d	t
0	1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0	1 1 0	6	ACK	SYN	&	6	F	V	f	v
0	1 1 1	7	BEL	ETB	'	7	G	W	g	w
1	0 0 0	8	BS	CAN	(8	H	X	h	x
1	0 0 1	9	HT	EM)	9	I	Y	i	y
1	0 1 0	10	LF	SUB	*	:	J	Z	j	z
1	0 1 1	11	VT	ESC	+	;	K	[k	{
1	1 0 0	12	FF	FS	,	<	L	\	l	;
1	1 0 1	13	CR	GS	-	=	M]	m	}
1	1 1 0	14	SO	RS	.	>	N	^	n	~
1	1 1 1	15	SI	US	/	?	O	_	o	DEL

32 CONTROL CODES

64 CHARACTER SET

96 CHARACTER SET

128 CHARACTER SET

Upshifted Lower Case

EXAMPLE: The representation for the character "K" (column 4, row 11) is.

	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁
BINARY	1	0	0	1	0	1	1
OCTAL	1	1	3				

* Depressing the Control key while typing an upper case letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.

HEWLETT-PACKARD CHARACTER SET FOR COMPUTER SYSTEMS

This table shows HP's implementation of ANS X3.4-1968 (USASCII) and ANS X3.32-1973. Some devices may substitute alternate characters from those shown in this chart (for example, Line Drawing Set or Scandinavian font). Consult the manual for your device.

The left and right byte columns show the octal patterns in a 16 bit word when the character occupies bits 8 to 14 (left byte) or 0 to 6 (right byte) and the rest of the bits are zero. To find the pattern of two characters in the same word, add the two values. For example, "AB" produces the octal pattern 040502. (The parity bits are zero in this chart.)

The octal values 0 through 37 and 177 are control codes. The octal values 40 through 176 are character codes.

Decimal Value	Octal Values		Mnemonic	Graphic ¹	Meaning	Character	Meaning
	Left Byte	Right Byte					
0	000000	000000	NUL	N _U	Null		Space, Blank
1	000400	000001	SOH	S _H	Start of Heading		Exclamation Point
2	001000	000002	STX	S _T X	Start of Text		Quotation Mark
3	001400	000003	ETX	E _T X	End of Text		Number Sign, Pound Sign
4	002000	000004	EOT	E _T	End of Transmission		Dollar Sign
5	002400	000005	ENQ	E _N Q	Enquiry		Percent
6	003000	000006	ACK	A _C K	Acknowledge		Ampersand, And Sign
7	003400	000007	BEL	B _E L	Bell, Attention Signal		Apostrophe, Acute Accent
8	004000	000010	BS	B _S	Backspace		Left (opening) Parenthesis
9	004400	000011	HT	H _T	Horizontal Tabulation		Right (closing) Parenthesis
10	005000	000012	LF	L _F	Line Feed		Asterisk, Star
11	005400	000013	VT	V _T	Vertical Tabulation		Plus
12	006000	000014	FF	F _F	Form Feed		Comma, Cedilla
13	006400	000015	CR	C _R	Carriage Return		Hyphen, Minus, Dash
14	007000	000016	SO	S _O	Shift Out } Alternate		Period, Decimal Point
15	007400	000017	SI	S _I	Shift In } Character Set		Slash, Slant
16	010000	000020	DLE	D _L E	Data Link Escape		} Digits, Numbers
17	010400	000021	DC1	D ₁	Device Control 1 (X-ON)		
18	011000	000022	DC2	D ₂	Device Control 2 (TAPE)		
19	011400	000023	DC3	D ₃	Device Control 3 (X-OFF)		
20	012000	000024	DC4	D ₄	Device Control 4 (TAPE)		
21	012400	000025	NAK	N _A K	Negative Acknowledge		
22	013000	000026	SYN	S _Y N	Synchronous Idle		
23	013400	000027	ETB	E _T B	End of Transmission Block		
24	014000	000030	CAN	C _A N	Cancel		Colon
25	014400	000031	EM	E _M	End of Medium		Semicolon
26	015000	000032	SUB	S _U B	Substitute		Less Than
27	015400	000033	ESC	E _C	Escape ²		Equals
28	016000	000034	FS	F _S	File Separator		Greater Than
29	016400	000035	GS	G _S	Group Separator		Question Mark
30	017000	000036	RS	R _S	Record Separator		
31	017400	000037	US	U _S	Unit Separator		
127	077400	000177	DEL	Δ	Delete, Rubout ³		

Decimal Value	Octal Values		Character	Meaning
	Left Byte	Right Byte		
64	040000	000100	@	Commercial At
65	040400	000101	A	Upper Case Alphabet, Capital Letters
66	041000	000102	B	
67	041400	000103	C	
68	042000	000104	D	
69	042400	000105	E	
70	043000	000106	F	
71	043400	000107	G	
72	044000	000110	H	
73	044400	000111	I	
74	045000	000112	J	
75	045400	000113	K	
76	046000	000114	L	
77	046400	000115	M	
78	047000	000116	N	
79	047400	000117	O	
80	050000	000120	P	
81	050400	000121	Q	
82	051000	000122	R	
83	051400	000123	S	
84	052000	000124	T	
85	052400	000125	U	
86	053000	000126	V	
87	053400	000127	W	
88	054000	000130	X	
89	054400	000131	Y	
90	055000	000132	Z	
91	055400	000133	[Left (opening) Bracket
92	056000	000134	\	Backslash, Reverse Slant
93	056400	000135]	Right (closing) Bracket
94	057000	000136	^ ↑	Caret, Circumflex, Up Arrow*
95	057400	000137	←	Underline, Back Arrow*

9206-1C

Notes:

¹This is the standard display representation. The software and hardware in your system determine if the control code is displayed, executed, or ignored. Some devices display all control codes as "||", "@", or space.

²Escape is the first character of a special control sequence. For example, ESC followed by "J" clears the display on a 2640 terminal.

³Delete may be displayed as "___", "@", or space.

⁴Normally, the caret and underline are displayed. Some devices substitute the up arrow and back arrow.

⁵Some devices upshift lower case letters and symbols (through ~) to the corresponding upper case character (@ through ^). For example, the left brace would be converted to a left bracket.

Decimal Value	Octal Values		Character	Meaning	
	Left Byte	Right Byte			
96	060000	000140	`	Grave Accent ⁵	
97	060400	000141	a		
98	061000	000142	b		
99	061400	000143	c		
100	062000	000144	d		
101	062400	000145	e		
102	063000	000146	f		
103	063400	000147	g		
104	064000	000150	h		
105	064400	000151	i		
106	065000	000152	j		
107	065400	000153	k		
108	066000	000154	l		
109	066400	000155	m		
110	067000	000156	n		
111	067400	000157	o		
112	070000	000160	p		Lower Case Letters ⁵
113	070400	000161	q		
114	071000	000162	r		
115	071400	000163	s		
116	072000	000164	t		
117	072400	000165	u		
118	073000	000166	v		
119	073400	000167	w		
120	074000	000170	x		
121	074400	000171	y		
122	075000	000172	z		
123	075400	000173	{	Left (opening) Brace ⁵	
124	076000	000174		Vertical Line ⁵	
125	076400	000175	}	Right (closing) Brace ⁵	
126	077000	000176	~	Tilde, Overline ⁵	

RTE SPECIAL CHARACTERS

Mnemonic	Octal Value	Use
SOH (Control A)	1	Backspace (TTY)
EM (Control Y)	31	Backspace (2600)
BS (Control H)	10	Backspace (TTY, 2615, 2640, 2644, 2645)
EOT (Control D)	4	End-of-file (TTY 2615, 2640, 2644, 2645)

9206-1D

BATCH-SPOOL MONITOR ERROR CODES

APPENDIX

B

When the Batch-Spool Monitor is active, numbered error codes and unnumbered error messages may result from improper program calls or command entries. The numbered error codes issued by the Batch-Spool Monitor are either positive or negative. In general negative error codes are a result of improper interface routine (FMP) calls. Positive error codes result from improper commands to the interactive programs FMGR or GASP. However, since these programs often use interface routines to perform the command functions, a negative code can result from a FMGR or GASP command.

The unnumbered error messages result from operator abort requests, job related conditions, and system commands (such as LG) entered through FMGR.

B-1. FMP AND FMGR ERROR CODES

A FMGR error causes a transfer to the log device whenever operator intervention is needed to correct the error. FMGR remains in control and is not aborted unless a batch job is currently active and the severity code for the job is less than 3. When control transfers to the log device and the terminal at which FMGR was scheduled is not the same as the log device, the terminal is no longer in control and all input must be made from the log device. As many commands as desired may be entered from the log device followed by a TR command without a *namr*. The TR command (refer to paragraph 2-41) returns control to the statement following the erroneous statement.

Since many FMGR commands use FMP interface routines, negative errors can be generated when using the FMGR operator commands. For example, the :ST command uses CREAT to create the *namr2* file and READF and WRITF to transfer data to that file.

Interface routine error codes (negative codes) are always returned in the A-register following a call as well as in the error return location.

A complete list of all FMGR and FMP error codes can be requested from FMGR by entering ??,99. The list is printed on the list device. FMP errors are described in detail in Table B-1; a matrix of these errors and the routines that may cause them is provided in Table B-2. Table B-3 describes the FMGR errors and lists the commands that may cause them.

Table B-1. FMP Error Codes

ERROR CODE	MESSAGE	MEANING & CORRECTIVE ACTION
000	(no error)	none
-001	DISC ERROR	The disc is down; try again and then report it to the system manager of facility.
-002	DUPLICATE FILE NAME	A file already exists with specified name; repeat with new name or purge existing file.
-003	BACKSPACE ILLEGAL	Attempt was made to backspace a device (or type 0 file) that cannot be backspaced; check device type.
-004	MORE THAN 32767 RECORDS IN A TYPE 2 FILE	Attempt was made to create a type 2 file with too many records or record size too large; check size parameter.
-005	RECORD LENGTH ILLEGAL	Attempt to read or position to a record not written, or on update to write an illegal record length; check position or size parameters.
-006	CR OR FILE NOT FOUND OR NO ROOM	Attempt to access a cartridge or file that cannot be found or which has no more room; check the file name or cartridge number, if no more room on cartridge try another, or decrease file size.
-007	BAD FILE SECURITY CODE	Attempt to access a file with no security code or the wrong code; find out the correct code and use it or do not access file.
-008	FILE OPEN OR LOCK REJECTED	Attempt to open file already open exclusively or open to eight programs or cartridge containing file is locked; use CL or DL to locate lock; if file being packed, check if spool shut down.
-009	ATTEMPT TO USE APOSN OR FORCE TO 1 A TYPE 0 FILE	Type 0 files cannot be positioned with APOSN or be forced to type 1; check file type.
-010	NOT ENOUGH PARAMETERS	Required parameters omitted from call; enter the parameters.
-011	DCB NOT OPEN	Attempt to access an unopened DCB; use CREATE, or OPEN to open DCB; check for errors.
-012	EOF OR SOF ERROR	Attempt to read or write or position beyond the file boundaries; check record position parameters, result depends on file type & call.
-013	DISC LOCKED	Cartridge is locked; initialize cartridge if not initialized, otherwise keep trying.

Table B-1. FMP Error Codes (Continued)

ERROR CODE	MESSAGE	MEANING & CORRECTIVE ACTION
-014	DIRECTORY FULL	No more room in file directory; purge files and pack directory if possible, or try another cartridge.
-015	ILLEGAL NAME	File name does not conform to syntax rules; correct name.
-016	ILLEGAL TYPE OR SIZE=0	Wrong type code supplied; attempt to create or purge type 0 file or create 0-length file; check size and type parameters.
-017	ILLEGAL READ/WRITE ON TYPE 0 FILE	Attempt to read/write or position type 0 file that does not support the operation; check file parameters, from FMGR check <i>namr</i> .
-020	ILLEGAL ACCESS LU	LU number specified in LU or CS command (see paragraphs 2-51 and 4-10) must be a positive logical unit number; correct command entry.
-021	ILLEGAL DESTINATION LU	LU number specified must be a logical unit number that was allocated by GASP.
-022	NO AVAILABLE SPOOL LU'S	No spool logical units currently available. Re-run after spool LU becomes available.
-023	NO AVAILABLE SPOOL FILES	No spool files currently available. Re-run after spool file becomes available.
-024	NO MORE BATCH SWITCHES	LU switch table full; size of switch table created at system generation not adequate.
-025	NO SPLCON ROOM	SPLCON full. May occur when spool system is competing with programs running outside batch and using their own spooling files and SMP.
-026	QUEUE FULL OR MAX PENDING SPOOLS EXCEEDED	Self-explanatory. Re-run when space becomes available.
-099	D.RTR I/O REQUEST REJECTED BY EXEC	D.RTR's tracks have been released or cartridge mounted that has not been initialized. Initialize disc.
-101	ILLEGAL PARAMETER IN D.RTR CALL	Possible operator error; recheck previous entries for illegal or misplaced parameters.
-102	ILLEGAL D.RTR CALL SEQUENCE	Lock not requested first or file not opened exclusively; possible operator error such as removal of cartridge without DC command.

Except for -10 and -11, any of these error codes can be returned from FMGR.

The matrix in Table B-2 shows which interface routines can be the cause of each error. For example, error -08 can occur as a result of the PURGE, OPEN, or NAMF routines. Since these routines can be called by FMGR, the matrix also indicates the errors that result in a transfer to the log device (X) and those that do not (0).

Appendix B

Table B-2. Relation Between FMP Error Codes and FMP Calls

Code	Message	CREAT	PURGE	OPEN	CLOSE	READF	WRITF	LOCF	APOSN*	RWDF	POSNT	FCONT*	NAMF	POST	IDCBS*
-001	DISC ERROR	X	X	X	X	X	X		0	X	X		X	X	
-002	DUPLICATE FILE NAME	X											X		
-003	BACKSPACE ILLEGAL										X				
-004	MORE THAN 32767 RECORDS IN A TYPE 2 FILE	X													
-005	RECORD LENGTH ILLEGAL					X	X		0		X				
-006	CR OR FILE NOT FOUND OR NO ROOM	X	0	X	X		X						X		
-007	BAD FILE SECURITY CODE		X	X			X						X		
-008	FILE OPEN OR LOCK REJECTED		X	X									X		
-009	ATTEMPT TO USE APOSN OR FORCE TO 1 A TYPE 0 FILE			X					0						
*-010	NOT ENOUGH PARAMETERS	0	0	0	0	0	0	0	0		0		0		
*-011	DCB NOT OPEN				0	0	0	0	0	0	0	0		0	0
-012	EOF OR SOF ERROR					X	X		0		X	0			
-013	DISC LOCKED	X	X	X									X		
-014	DIRECTORY FULL	X					X								
-015	ILLEGAL NAME	X											X		
-016	ILLEGAL TYPE OR SIZE=0	X	X												
-017	ILLEGAL READ/WRITE ON TYPE 0 FILE					X	X				X				

X - transfer to log device (FMGR)

0 - no transfer to log device

* - error never returned to FMGR or routine never called by FMGR

Table B-3. FMGR Error Codes

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
000	FMGR-BREAK	Informative message; no error.	general
001	DISC ERROR <i>nn</i>	Disc associated with <i>nn</i> is down; report problem to system manager.	general
002	INITIALIZE LU 2!	Request for IN command to initialize system disc (lu 2); enter command.	Initialize FMGR
003	INITIALIZE LU 3!	Request for IN command to initialize auxiliary disc (lu 3); enter command.	Initialize FMGR
004	ILLEGAL RESPONSE TO 002 OR 003	Command other than IN entered in response to 002 or 003; enter IN.	Initialize FMGR
005	REQUIRED TRACK NOT AVAILABLE RELATIVE TAT POSITION REPORTED - <i>TAT position</i> ¹	First track specified in IN not available; re-enter IN with first available track report in message. If from MS, <i>TAT position</i> not reported.	Initialize FMGR or MS
006	FMGR SUSPENDED	FMGR suspended itself; ready device and enter GO, FMGR	MR, ST, DU
007	CHECKSUM ERROR	Checksum error on paper tape, or file is not binary (type 5 or 7); check type.	MR, ST, SA, DU
008	D.RTR NOT LOADED	Program D.RTR not found in system; load D.RTR as permanent program.	Initialize FMGR
*009	ID-SEGMENT NOT FOUND	RP was used to deallocate or reassign ID segment to program being restored; system looks for blank ID segment.	RP

*Does not cause transfer to log device.

¹TAT position is track number on LU 2, on LU 3 it is track on LU 3 plus all tracks on LU 2.



Table B-3. FMGR Error Codes (Continued)

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
010	INPUT ERROR (if syntax check failed, command up thru error is displayed followed by ?)	Syntax error in statement; look for missing colon (batch input) or extra colon (interactive), undefined command, error in <i>namr</i> sub-parameters, command too long, etc re-enter command.	Syntax Check
		If AB command, no job was active.	AB
011	DO OF, <i>program</i> , 8 ON NAMED PROGRAM	Attempt to pack disc to which named programs still allocated; RP, <i>program</i> or OF, <i>program</i> , 8 to remove programs.	PK
012	DUPLICATE DISC LABEL OR LU	Attempt to mount cartridge with label or logical unit of mounted cartridge; re-enter with another label or lu, or dismount duplicate cartridge.	MC
013	TR STACK OVERFLOW	More than 10 nested TR commands; correct coding.	TR
014	REQUIRED ID-SEGMENT OR ID EXTENSION NOT FOUND	Program specified for which no ID segment found; check program name or load program.	SP, RP
		MS assigns tracks to EDITR which does not exist. RP the EDITR.	MS
		Blank segment not found for program being restored; OF program to release segment.	RP
*015	LS TRACK REPORT LS LU N TRACK <i>nn</i>	Informative message to report logical unit and track of current LS area.	MS
016	FILE MUST BE AND IS NOT ON LU 2 OR 3	Attempt to restore program file not on system or auxiliary disc; move file to lu 2 or 3 and re-enter command.	RP
017	ID SEGMENT NOT SET UP BY RP	To be released by RP, ID segment must have been set up by RP; try OF, <i>program</i>	RP

*Does not cause transfer to log device.

Table B-3. FMGR Error Codes (Continued)

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
018	PROGRAM NOT DORMANT	RP, <i>namr, program</i> attempted when program is active; enter OF, <i>program</i> then RP	RP
019	FILE NOT SET UP BY SP ON CURRENT SYSTEM	Program file being restored had parity error or was not set up correctly or was not set up by SP in current system; program and try again.	RP, RU
020	ILLEGAL TYPE 0 LU	Attempt to create type 0 file on logical unit not assigned in system; re-enter using another logical unit.	AN, CL, CR (type 0), DC, DL, DU, IN, LI, MC, MR, MS, ST
021	ILLEGAL DISC SPECIFIED	Attempt to copy files to or from same disc or disc not mounted; mount disc or use another.	CO
022	COPY TERMINATED	Copy has been terminated as a result of copy error; check parameters and specified discs.	CO
023	DUPLICATE PROGRAM NAME	Program being restored is already defined in system; change name, OF program, or release ID segment.	RP
024 through 046 are undefined			
047	SPOOL SETUP FAILED (In later systems, this error code has been replaced with FMP error codes -020 through -026 see Table B-1)	No available spool files or logical units or logical unit table full; can try job again, but if error is from lack of spool logical units or logical unit table full, must reconfigure.	JO, LU
048	GLOBAL SET OUT OF RANGE	A global was specified out of the range of globals; check parameters and re-enter command correctly.	CA, SE, TR
049	CAN'T RUN RP'D PROGRAM OR PARTITION TOO SMALL	Program restored from file does not execute; usually attempt to run SP's segment; check program.	RU
050	ILLEGAL NUMBER OF PARAMETERS	Less than the required number of parameters were specified; re-enter correctly.	IN, MR, SA, SP

Table B-3. FMGR Error Codes (Continued)

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
051	ILLEGAL MASTER SECURITY CODE	Attempt to re-initialize or list files with incorrect master security code; re-enter with correct code.	IN, DL
052	ILLEGAL LU IN RESPONSE TO 002 OR 3	Attempt to initialize FMGR using lu other than 2 or 3; Also an attempt to mount an LU which is not a disc cartridge; re-enter correctly specified lu not assigned to a disc; check lu and re-enter correctly.	Initialize FMGR IN, MC
053	ILLEGAL LABEL OR ILABEL	Illegal cartridge reference number or cartridge ID; CR must be positive non-zero integer, ID must be legal file name.	IN
054	DISC NOT MOUNTED	Attempt to reference unmounted disc; mount disc and activate with MC.	IN, DC, PK, DL
055	MISSING PARAMETER	Required parameter is omitted; check and re-enter with missing parameter.	IN, MC, DC, CR, DU, ST
056	BAD PARAMETER	Parameter specified incorrectly or a track parameter specifies track outside range of FMGR tracks; check and re-enter correctly.	IN, DU, ST, CR, LO, SA LI
057	BAD TRACK NOT IN FILE AREA	Specified track in system area or is a directory track; re-enter correctly.	IN
058	LG AREA EMPTY	Attempt to save contents of LG area which is empty; re-compile or use MR.	SA
059	REPORTED TRACK UNAVAILABLE HIGHEST NON-AVAILABLE TRACK <i>nnn</i>	Attempt to re-initialize lowers first track into system area, last system track reported as <i>nnn</i> ; re-enter IN with first track = last track + 8 (min).	IN
060	DO YOU REALLY WANT TO PURGE THIS DISC? (YES OR NO).	A re-initialization raises first track or lowers directory into file and will destroy file; enter ?? or NO to stop re-initialization, YES to continue.	IN

Table B-3. FMGR Error Codes (Continued)

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
061	DO A "DC" AND A "MC" ON THIS CR	<p>Attempt to replace a mounted cartridge without entering a DC and MC command, and then attempt to initialize the new cartridge which has not been previously initialized.</p> <p>Enter a DC and MC command for this cartridge.</p>	IN
062	MORE THAN 31 DISCS	<p>Attempt to mount 32nd cartridge (limit is 31 cartridges).</p> <p>Dismount cartridges to make room, if possible.</p>	MC

B-2. GASP ERROR CODES

GASP errors are reported as either negative or positive codes; negative when an FMP interface routine used by the command finds an error, positive as a direct result of the command. Most positive GASP error codes result from illegal syntax, most negative error codes from file access problems. Note that all the negative codes correspond to FMP codes.

Error messages are not issued for SMP call errors. Improper spool setup is diagnosed by a 0 returned in parameter ISLU; other spool calls are ignored should they be improperly specified.

A complete list of GASP calls can be requested by entering ??,99 in response to the GASP prompt. These messages are described in detail in Table B-4.

Table B-4. GASP Error Codes

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
-001	DISC ERROR	Disc is down; try again and then report problem to system manager.	Any GASP command that tries to open or access JOBFIL or SPLCON (all except ?? or EX); in particular, GASP initialization or DA.
-002	DUPLICATE FILE NAME	JOBFIL or SPLCON or spool pool file already created.	
-004	MORE THAN 32767 RECORDS IN TYPE 2 FILE	Consult system manager.	
-006	CR OR FILE NOT FOUND OR NO ROOM	Generally, no room to create spool files.	
-007	BAD FILE SECURITY CODE	Consult system manager.	
-008	FILE OPEN OR LOCK REJECTED	Possibly, attempt to DeAllocate (DA) with spool system active; shut down spooling (SD) and try again.	
-012	EOF OR SOF ERROR	Consult system manager.	
-013	DISC LOCKED	Cartridge containing JOBFIL or SPLCON locked; initialize or if already initialized, wait and try again.	
-014	DIRECTORY FULL	No more room in file directory for spool files. Pack cartridge and re-run GASP.	
0	NO ERROR	Informative message; no error	General
1	DISC ERROR <i>nn</i>	Disc associated with <i>nn</i> is down; report problem to system manager.	Initialize GASP
2	NUMBER OUT OF RANGE	Number entered in GASP initialization inconsistent with previous entries or exceeds maximum specified at generation; check last entry and change.	Initialize GASP

Table B-4. GASP Error Codes (Continued)

ERROR CODE	ERROR MESSAGE	MEANING AND CORRECTIVE ACTION	ROUTINE OR COMMAND
3	BAD JOB NUMBER!	Specified job number not currently assigned; re-enter command with valid job number.	AB, CJ, DJ
4	ILLEGAL STATUS	Command is not valid for current state of job or spool file; check status with DJ or DS.	CJ, CS, KS
5	ILLEGAL COMMAND	Command not recognized by GASP; check and re-enter command correctly.	Any unrecognized command
6	NOT FOUND	Specified job or spool not currently assigned; check with DJ or DS.	AB, CJ, CS, DJ, KS, RS
55	MISSING PARAMETER	Required parameter omitted; check and re-enter with parameter.	AB, CJ, CS, KS, RS
56	BAD PARAMETER	Specified parameter cannot be recognized; check parameter and re-enter.	Any except DA or EX
/GASP: IRRECOVERABLE INITIALIZE ERROR		GASP is not a background program or SPOOL EQT extensions are less than 18 words long or SPOOL driver is not in system driver area.	During initialization process.

B-3. UNNUMBERED ERROR MESSAGES

Unnumbered error messages may result during FMGR execution because of operator abort requests, job related conditions, or system commands (such as LG) entered through FMGR. Table B-5 describes these messages, lists possible causes, and suggests corrective action to be taken.

Table B-5. Unnumbered Error Messages

COMMAND	ERROR MESSAGE	MEANING	CORRECTIVE ACTION
:AB or *AB	ABEND OPERATOR	<ol style="list-style-type: none"> 1. The job has been aborted by operator request. 2. The job has been aborted because of spool I/O error. 	<ol style="list-style-type: none"> 1. Examine job, correct (if necessary), and re-run job. 2. Check for such errors as write to read-only spool file, or read to write only spool file.
	JOB xxxxx ABORTED	Error encountered during job execution.	Examine job, correct (if necessary), and re-run job.
:EO or :JO	ABEND EOJ IN ssssss	An :EO or :JO command was encountered, but in a different level than the original :JO command. For example, control has transferred from PROG1 to PROG2 and PROG2 contained an :EO or :JO command. ssssss is the file name or logical unit number where :EO or :JO occurred.	Examine job, correct (if necessary), and re-run job.
:LG,#tracks	NO LGO SPACE	More LG tracks were requested than are available.	Check that all possible tracks have been released back to the system. Re-check your command entry. The number of tracks specified must be less than or equal to the available track count.
	LGO IN USE	The Assembler or a compiler has current use of the LG track area.	Wait until the LG tracks are released and re-enter the command.
:OF,program (same as system command OF,program,8)	NO SUCH PROGRAM	The program name specified does not exist as a system main program.	Recheck your command entry. The program name specified must be the name of a system main program.

Table B-5. Unnumbered Error Messages (Continued)

COMMAND	ERROR MESSAGE	MEANING	CORRECTIVE ACTION
:OF,program (continued)	xxxxx ABORTED	A temporary program has been removed from the system, or a permanent program has been aborted (xxxxx is the name of the program).	None Required.
:RT,program	NO SUCH PROGRAM	The program name specified does not exist as a system main program.	Recheck your command entry. The program name specified must be the name of a system main program.
	ILLEGAL STATUS	The program named is not dormant.	It is illegal to attempt releasing tracks for a non-dormant program. Wait until program is dormant and re-enter command.
:RU,program	ABEND xxxxx ABORTED	The program was aborted by the operator or the system following entry of the RU command (xxxxx is the program name).	Check program for errors and re-run.
	*ABEND JOB LIMIT	The job time limit (set via the :JO command) has been exceeded.	Check program for errors and/or extend job time limit.
	ABEND RUN LIMIT	The run time limit (set via the :TL command) has been exceeded.	Check program for errors and/or extend run time limit.
	FMGR WAITING ON LU xx	LU xx is down or locked.	Up the EQT associated with LU xx or unlock the LU.
	LIST OVERFLOW	List file extending to overflow FMGR area on disc; spool file overflow directory overflow.	Obtain more spool room on disc (see PK command) or do not use spooling at this time.
*This message can occur at any time during execution of a job.			

B-4. SPOOL ERRORS

Error conditions occurring during spool processing result in error code or error message reports. These error reports are discussed in the following paragraphs.

B-5. SPOOL I/O ABORT ERRORS

Spool I/O abort error conditions result in the display of error codes in the form:

IO nn

where nn is an error number. Table B-6 defines these error codes.

Table B-6. Spool I/O Abort Error Codes

ERROR CODE	MEANING	ACTION
IO20	Read attempted on write only spool file.	Revise program and re-run.
IO21	Read attempted past end-of-file (EOF).	Revise program and re-run.
IO22	Second attempt to read JCL card from batch input file by other than FMGR.	Revise program and re-run.
IO23	Write attempted on read only spool file.	Revise program and re-run.
IO24	Write attempted beyond end-of file (EOF); usually, spool file overflow.	Obtain more spool room on disc (See PK command, paragraph 2-60) or do not use spooling at this time.
IO25	Attempt to access spool LU that is not currently set up.	May be caused by GASP KS command — if other reason, correct offending programs.

B-6. SMP ERROR MESSAGES

Error messages produced by SMP are reported in the form:

SMP: *error message*

where the error message reported is defined in Table B-7.

Table B-7. SMP Error Messages

ERROR MESSAGE	MEANING	ACTION
SMP: LU _{xx} EOFER <i>filename</i>	File <i>filename</i> just outspooled to logical unit <i>xx</i> overflowed or was otherwise incomplete.	Re-run the JOB.
SMP: LU _{xx} DOWN <i>filename</i> HELD	Logical unit <i>xx</i> down; file <i>filename</i> placed in active hold.	Correct LU down condition and use GASP to restart operation or release hold status on file.
SMP: FMP- <i>nn</i>	FMP error - <i>nn</i> occurred during SMP operation (see Table B-1). Usually indicates loss of JOBFIL or SPLCON.	Use GASP to deallocate and reallocate the spool files.

B-7. OUTSPOOL ERROR MESSAGES

Errors encountered during outspool operations result in the messages defined in Table B-8.

Table B-8. Outspool Error Messages

MESSAGE	CAUSE	CORRECTIVE ACTION
JOB WAIT ON PT	End-of-tape occurred between :JO and :EO commands.	Load remainder of job in reader, ready the reader, and enter *GO,JOB.
JOB WAIT ON SPOOL RESOURCE	Required spool file or logical device cannot be obtained at this time.	None required. JOB will suspend and be automatically rescheduled when the resource becomes available.
JOB WAIT ON EXTENT	Spool file overflows available disc space.	Condition will automatically clear when SMP releases spool files as a result of outspool completion; or you can force a retry using the GASP command, SU; or you can abort JOB.
END JOB ABNORM	JOBFIL could not be opened; or other uncorrectable error occurred; or JOB was run before Spool initialization.	Try re-initialization with GASP after all spool activity is completed.
BAD EOF	Message appears after last line of file. ASCII file outspooling overflowed; or was otherwise incomplete.	Re-run JOB.

TABLES, DIRECTORIES, AND RECORD FORMATS

APPENDIX

C

Reference material frequently used in Batch-Spool Monitor operation is organized in this appendix as follows:

- C-1. Standard Logical Units
- C-2. Standard Drive Types
- C-3. Data Control Block Format
- C-4. Cartridge Directory Format
- C-5. File Directory Format
- C-6. JOBFIL Format
- C-7. SPLCON Format
- C-8. Record Format Disc Files
- C-9. Record Format Non-Disc Files
 - Relocatable Tape Format
 - Absolute Tape Format



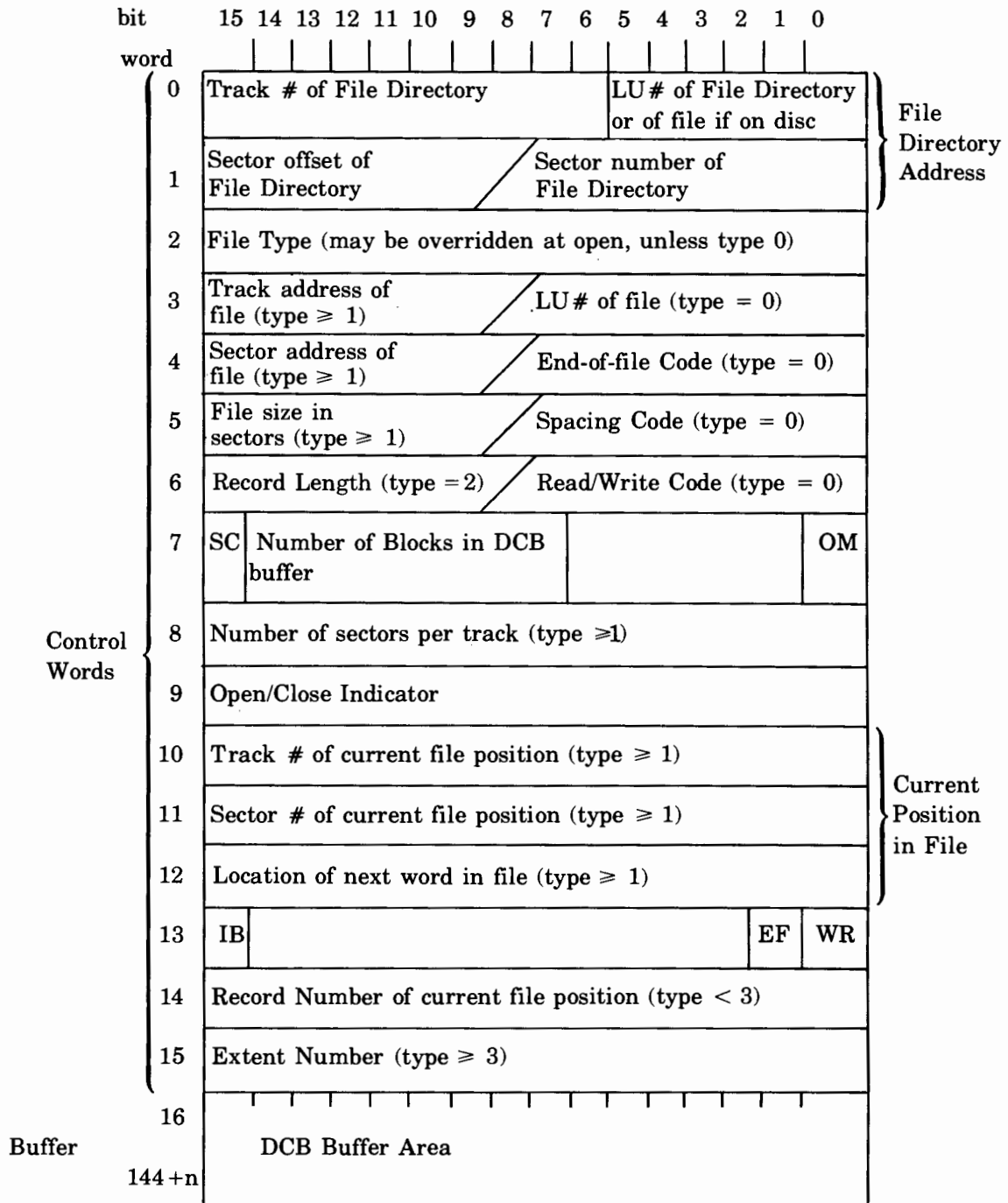
C-1. STANDARD LOGICAL UNITS

LOGICAL UNIT	DEVICE
1	system console
2	system disc
3	auxiliary disc
4	standard output
5	standard input
6	standard list
8	usually magnetic tape

C-2. STANDARD DRIVER TYPES

DRIVER	TYPE	DEVICE
DVR00	00	Teleprinter/CRT Display Terminal
	01	Paper Tape Reader
	02	Paper Tape Punch
DVR05	05	Page Mode CRT
DVR10	10	Plotter
DVR11	11	Card Reader
DVR12	12	Line Printer
DVR15	15	Mark Sense Reader
DVR23	23	9-Track Magnetic Tape
DVR31	31	Moving Head Disc
	32	Moving Head Disc
	33	Moving Head Disc (reserved)

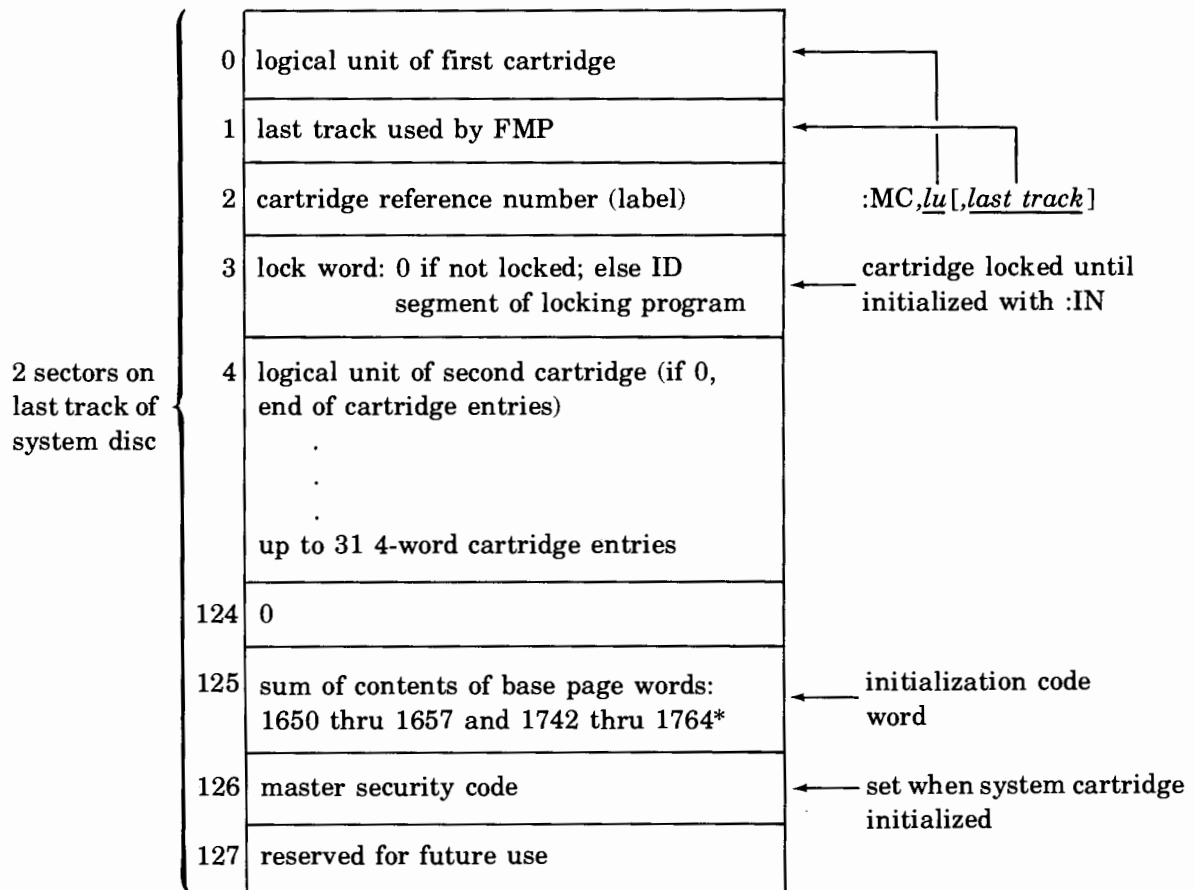
C-3. DATA CONTROL BLOCK FORMAT



Legend for Data Control Block

Word	Content
4	End-of-File Code, type 0 file: 01 <i>lu</i> = EOF on Magnetic Tape 10 <i>lu</i> = EOF on Paper Tape 11 <i>lu</i> = EOF on Line Printer
5	Spacing Code, type 0 file: bit 15 = 1 - backspace legal bit 0 = 1 - forward space legal
6	Read/Write Code, type 0 file: bit 15 = 1 - input legal bit 0 = 1 - output legal
7	Security Code Check/Open Mode/Buffer Size, all files types
(SC) Security Code Check:	bit 15 = 1 - security codes agree = 0 - security codes do not agree
DCB Buffer	bits 14-7 = Number of blocks in DCB buffer
(OM) Open Mode	bit 0 = 1 - update open = 0 - standard open
9	Open/Close Indicator: if open, contains ID segment location of program performing open. if closed, set to zero.
13	In Buffer/To Be Written/EOF Read Flags, all file types
(IB) In Buffer Flag:	bit 15 = 1 - data in DCB buffer = 0 - data not in DCB buffer
EOF Read Flag:	bit 1 = 1 - EOF has been read = 0 - EOF has not been read
(WR) To Be Written Flag	bit 0 = 1 - data in DCB buffer to be written = 0 - data in DCB buffer not to be written

C-4. CARTRIDGE DIRECTORY FORMAT



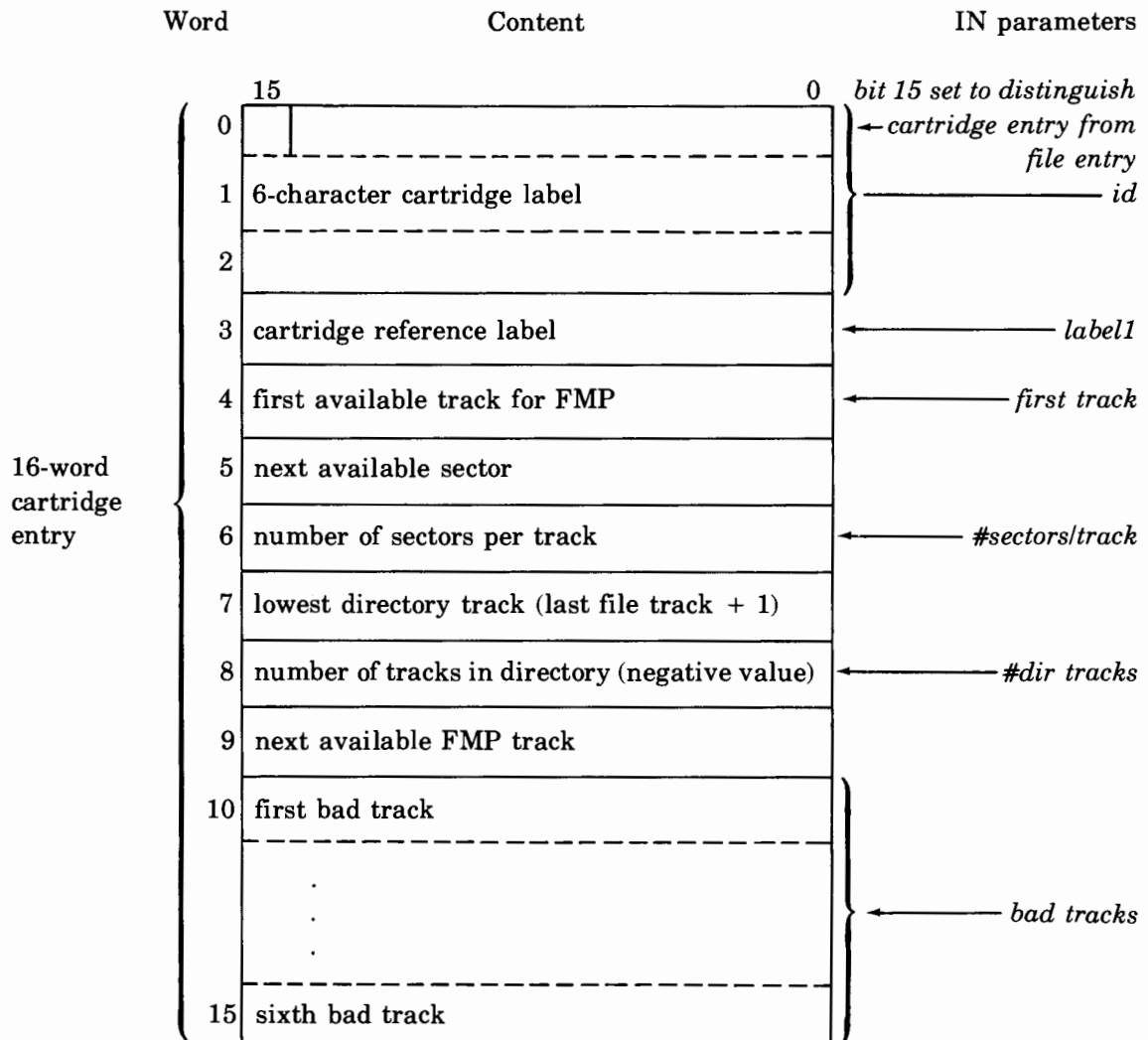
*For RTE-IV, the contents of base page words 1750-1754 are not included in the sum.

C-5. FILE DIRECTORY FORMAT

The first entry in each File Directory is the specification entry for the cartridge itself. The directory starts on the last FMP track of each cartridge in sector zero for all but the system cartridge (LU 2) on which it starts in the next logical directory block. The directory sector address can be obtained from the block address by the following formula:

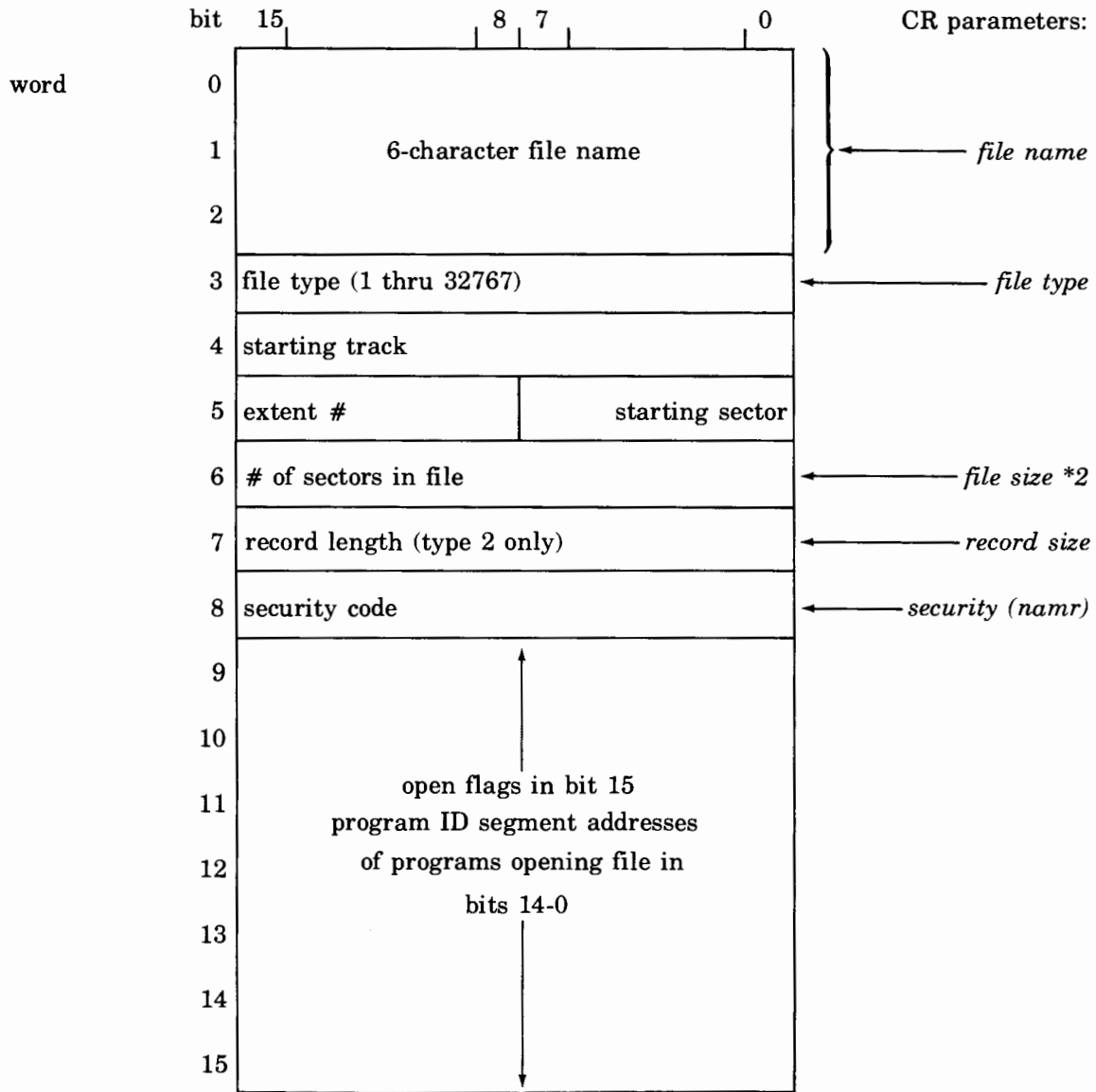
$$\text{sector address} = (\text{block} * 14) \text{ modulo } S/T$$

where S/T is the number of sectors per track. Directory blocks are 128 words long. Each Directory entry is 16 words long.



The 16-word cartridge entry is followed by an entry for each created file; non-disc (type 0) files follow disc files in the formats shown:

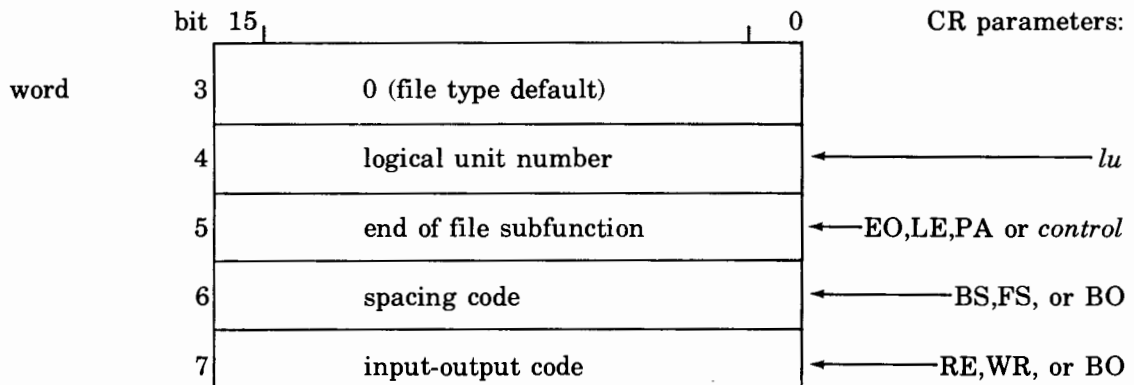
Disc File Directory Entry



word 0 = 0 if the last entry in directory; = -1 if file is purged

Type 0 File Directory Entry

The entries for non-disc (type 0) files differ from those for disc files in words 3 through 7:



Appendix C

JOBFIL Format (Continued)

	Word	Content		
	9	not currently used		
	10			
	11			
	12			
	13	JOBFIL record number of current job		
	14	wait - resource number		
	15	shut down flag for batch processing	← set by SD	
record 18	0	# of files / first file	← set by GASP Initialization	
		cartridge label		
	8 2-word entries	1	repeated for each cartridge containing spool files	
	15			
Job Records records 19 and up 1 record per job	0	job priority	← -1 if no job	
	1	job number (= record number - 18)	← assigned by SM	
	2	job status		
	3	job location - logical unit		
	4	or		
	5	3-word file name	← set if job input from disc	
	6	cartridge identification		
	7	job name		
	8	(3-word name)		
	9			
	10	spool priority		

Appendix C

SPLCON Format (Continued)

	Word	Content		
	0	15	← each bit corresponds to a SPLCON record; set to 1 if record in use.	
record 2	.	.		
	15			
record 3	0	shut down flag for outspooling	← set by SD	
	1	wait - resource number		
	2	words 2 through 15 not used		
	:			
	15			
records 4-8 (not used)	.	.		
	.	.		
	.	.		
Outspool Queue records	9-16 (1 sector) queue for one outspool logical unit	0	queue depth outspool logical unit #	} 1 entry/file
		1	# of entries in queue	
		2	SPLCON record number of file	
		3	spool priority	
	.	.		
	.	.		
	127			

continue with 1 sector (8 records) for each outspool logical unit assigned at GASP initialization.

	Word	Content
Spool Control Records (follow outspool queue)	0	batch checking flag
	1	spool logical unit
	2	file name of user file or spool pool file
	3	
	4	security code
	5	
	6	cartridge reference

1 record per spool file in same format as IBUFR (setup buffer for SPOP call)

SPLCON Format (Continued)

Word	Content
Spool Control Records (follow out-pool queue) 1 record per spool file in same format as IBUFR (setup buffer for SPOPN call)	7 driver type of outspool logical unit
	8 BU BI 0 R-W 0 ST SP 0 HO SA ← disposition flags
	9 spool priority
	10 spool status
	11 job # of spool owner ← used by FMGR only to setup a spool file
	12 0
	14 0
15 outspool logical unit number	
continue with one record for each spool file in SPLCON	.

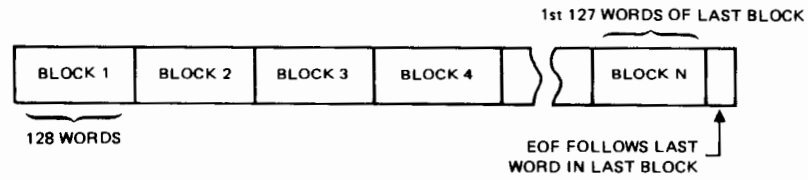


Disposition Flags (word 8):

- SA = 1 save
0 purge at completion of spooling
- HO = 1 hold until close
0 outspool immediately
- SP = 1 spool pool file
0 user file
- ST = 1 standard file format (no record headers on output)
0 spool format
- R/W = 00 write and read
01 read only
10 write only
- BI = 1 batch checking (used only by FMGR)
- BU = 1 buffering
0 no buffering

C-8. DISC FILE RECORD FORMATS

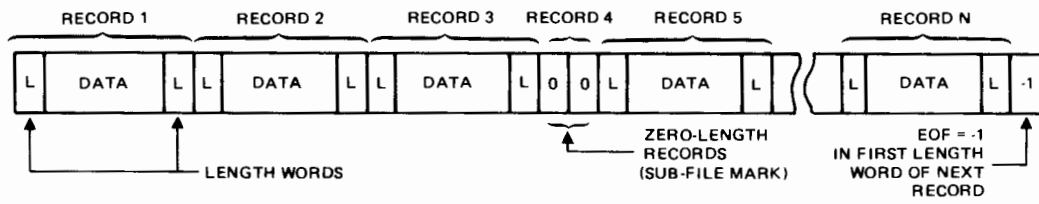
Fixed Length Formats (Types 1 and 2)



Type 1 Record length = Block length = 128 words

Type 2 Record length is user defined; may cross block boundaries but not past EOF

Variable Length Formats (Types 3 and Above)



Memory-Image Program File Formats (Type 6) for RTE-II/III

Files created by the SP command as memory-image program files are always accessed as type 1 files (fixed length, 128-words per record).

Word	Content	
0	-1	← EOF unless forced to type 1
1-5	not used	
6	priority	
7	primary entry point	
8-13	not used	
14	program type	
15-16	not used	
17-19	time parameters	
20	substatus 1 - word 21 of ID segment	
21	substatus 2 - word 22 of ID segment	
22	low main address	
23	high main address	
24	low base-page address	
25	high base-page address	
26-27	not used	
28	checksum of words 0 - 27	
29	setup code word	← sum of contents of words 1650 thru 1657 and words 1742 thru 1764 in base page
30-127	not used	

1st two sectors contain program's ID-segment information

Remainder of file is an exact copy of the program being saved.

Memory-Image Program File Formats (Type 6) for RTE-IV

Files created by the SP command as memory-image program files are always accessed as type 1 files (fixed length, 128-words per record).

Word	Content
0	-1
1-5	not used
6	priority
7	primary entry point
8-13	not used
14	program type
15-16	not used
17-19	time parameters
20	substatus 1 - word 21 of ID segment
21	substatus 2 - word 22 of ID segment
22	low main address
23	high main address
24	low base-page address
25	high base-page address
26-27	not used
28	ID EXT # EMA SIZE
29	High Addr +1 of largest segment
30	not used
31	not used
32	not used
33	checksum of words 0 - 32
34	setup code word
35	ID EXT word 0
36	ID EXT word 1
37-127	not used

1st two sectors contain program's ID-segment information

← EOF unless forced to type 1

← sum of contents of words 1650 thru 1657, words 1742 thru 1747, and words 1755-1764 in base page

Remainder of file is an exact copy of the program being saved.

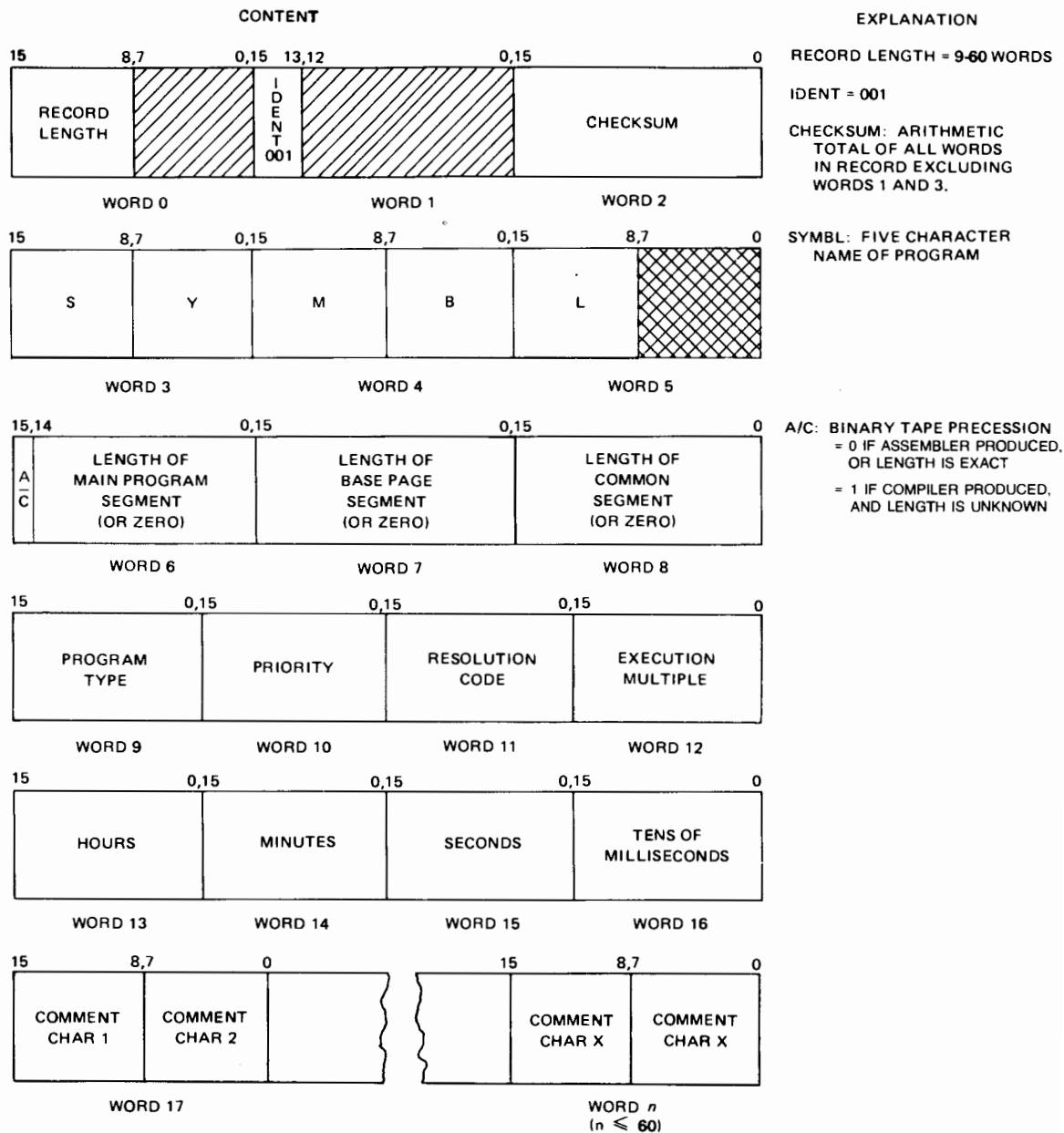
C-9. NON-DISC RECORD FORMATS

The record formats of programs saved on tape as relocatable or absolute binary records as shown here.

RELOCATABLE TAPE FORMATS

Formats are illustrated for NAM, ENT, EXT, DBL, and END records.

NAM Record

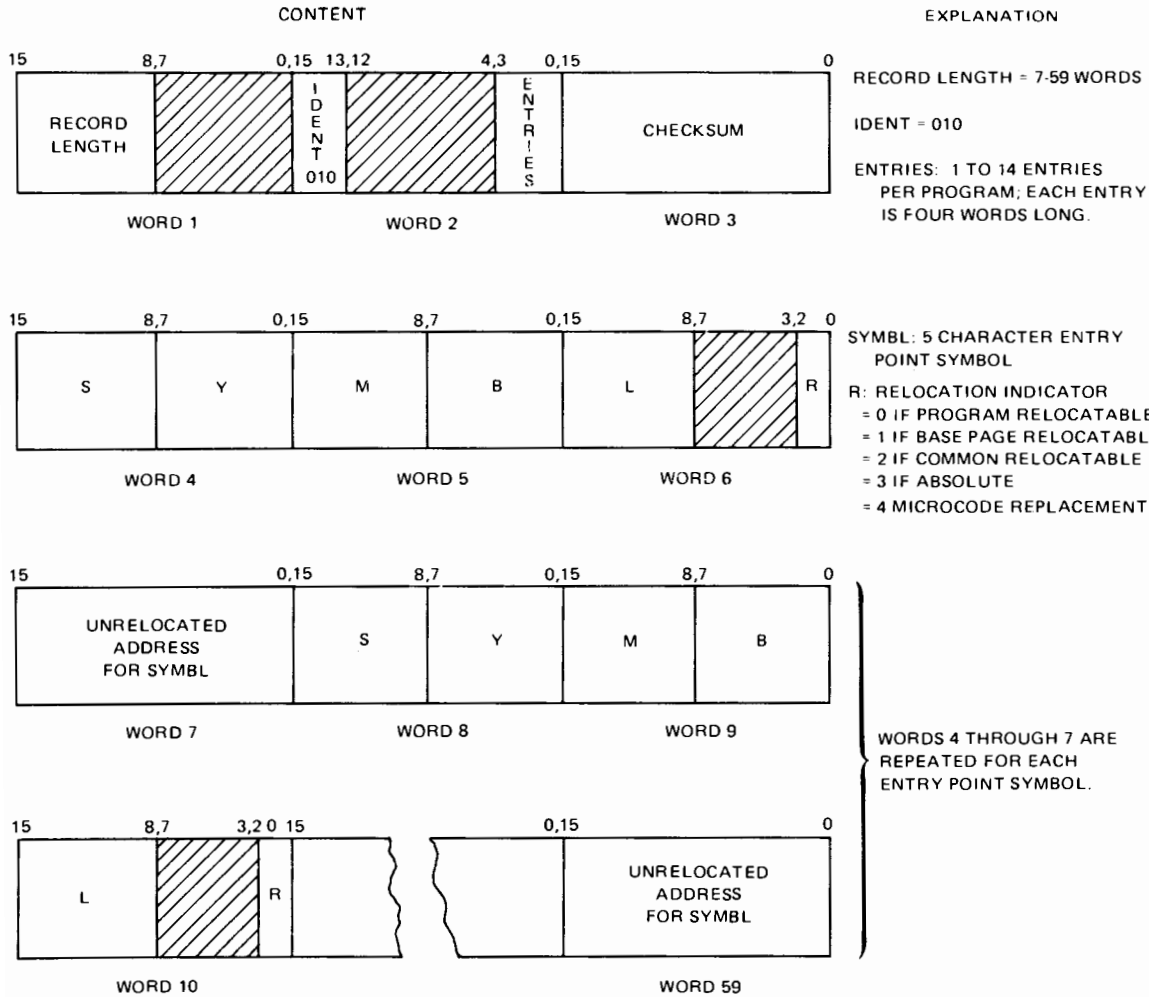


HATCH-MARKED AREAS SHOULD BE ZERO-FILLED WHEN THE RECORDS ARE GENERATED

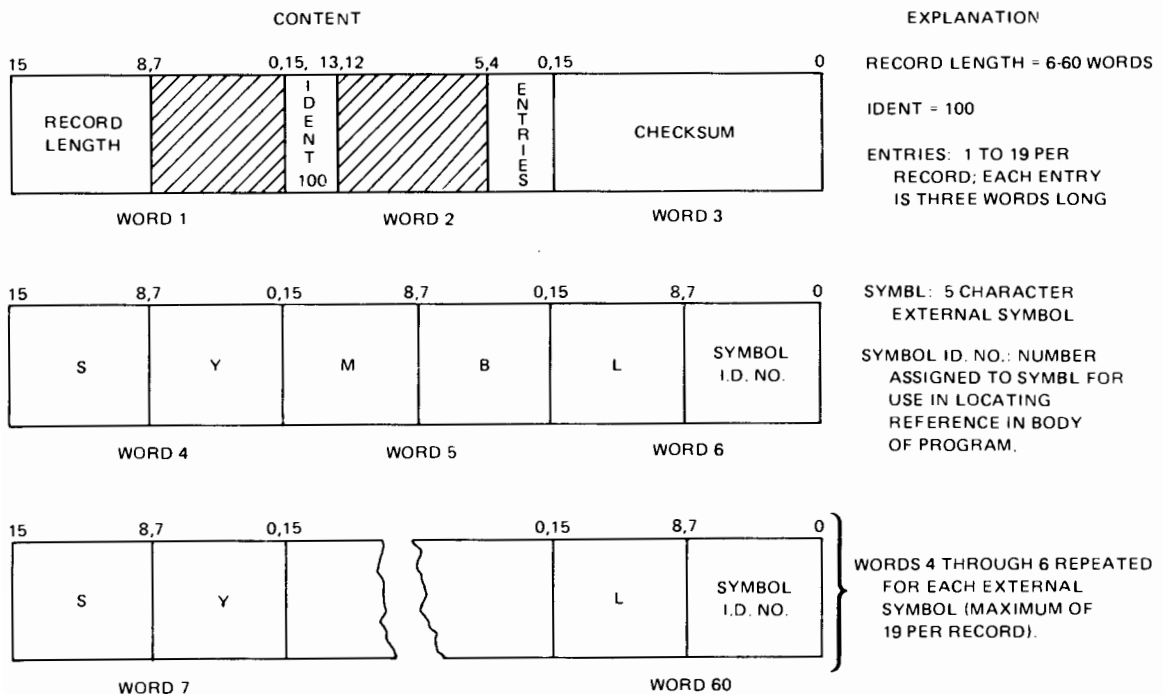
CROSS-HATCHED AREAS SHOULD BE BLANK-FILLED WHEN THE RECORDS ARE GENERATED.

Appendix C

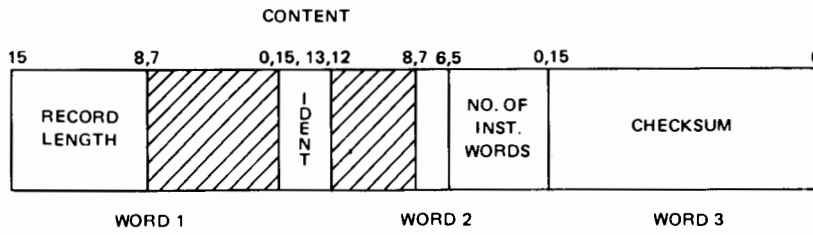
ENT Record



EXT Record

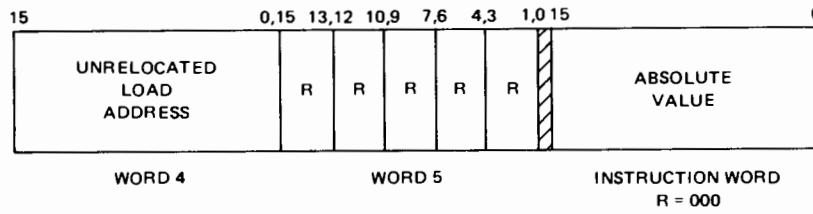


DBL Record



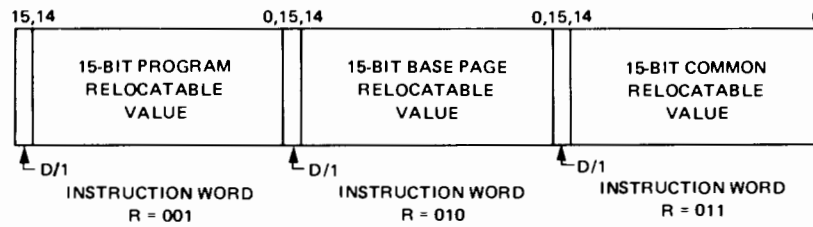
EXPLANATION

RECORD LENGTH = 6-60 WORDS
 IDENT = 011
 Z/C: RELOCATION OF LOAD ADDRESS
 = 0 FOR BASE PAGE
 = 1 FOR PROGRAM
 = 2 FOR ABSOLUTE
 = 3 FOR COMMON
 NO. OF INST. WORDS: 1 TO 45
 LOADABLE INSTRUCTION WORDS PER RECORD

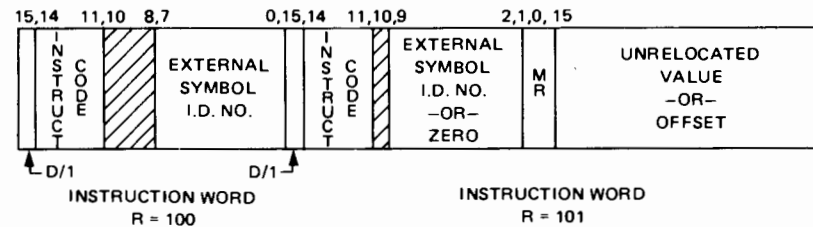


RELOCATABLE LOAD ADDRESS: STARTING ADDRESS FOR LOADING THE INSTRUCTIONS WHICH FOLLOW:

R's: RELOCATION INDICATORS:
 000 = ABSOLUTE
 001 = 15-BIT PROGRAM RELOCATABLE
 010 = 15-BIT BASE PAGE RELOCATABLE
 011 = 15-BIT COMMON RELOCATABLE
 100 = EXTERNAL REFERENCE
 101 = MEMORY REFERENCE



R₁ IS RELOCATION INDICATOR FOR INSTRUCTION WORD₁; R₂ FOR INSTRUCTION WORD₂; ETC.

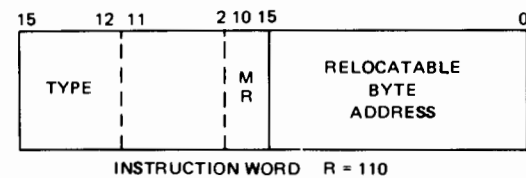


D/I: INDIRECT ADDRESSING

0 = DIRECT
 1 = INDIRECT

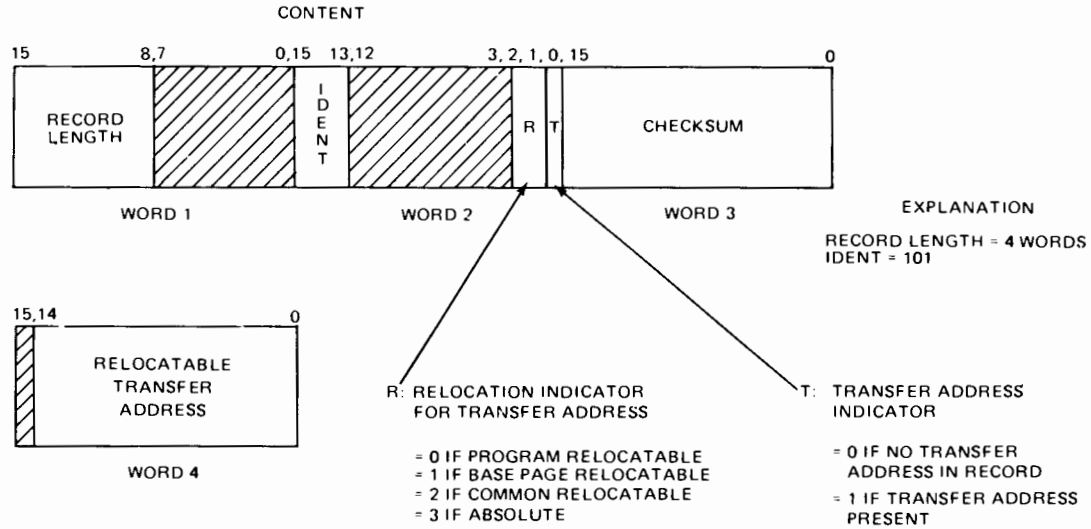
MEMORY REFERENCE INSTRUCTIONS USE TWO WORDS, WITHIN THE TWO-WORD GROUP; "MR" INDICATES RELOCATABILITY OF OPERAND SPECIFIED IN SECOND WORDS:

00 = PROGRAM RELOCATABLE
 01 = BASE PAGE RELOCATABLE
 10 = COMMON RELOCATABLE
 11 = ABSOLUTE



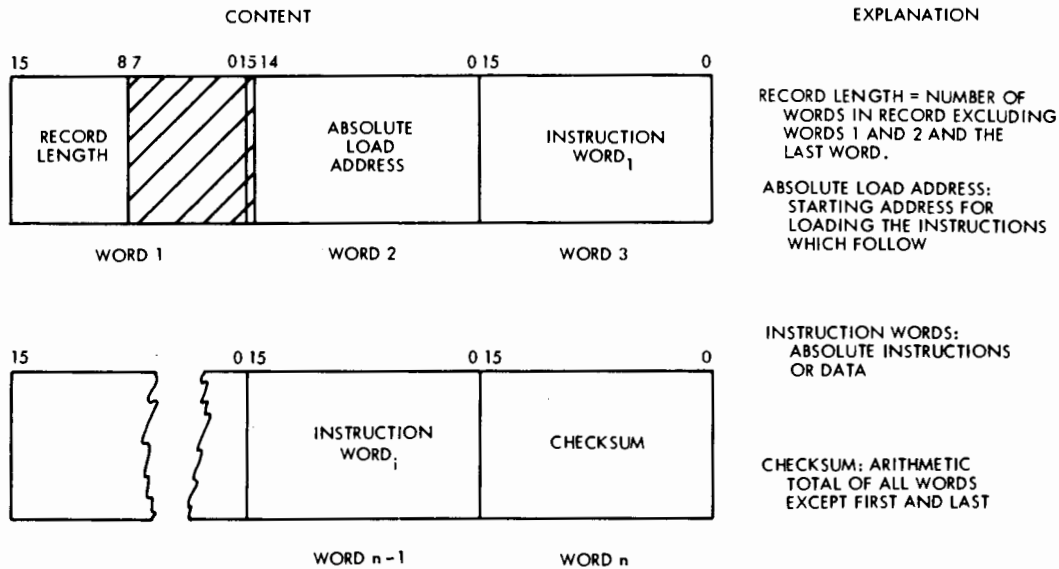
Appendix C

END Record

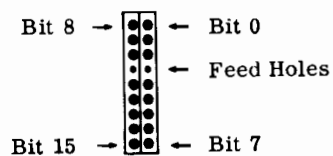


ABSOLUTE TAPE FORMAT

Absolute binary code is written to paper tape in the following format:

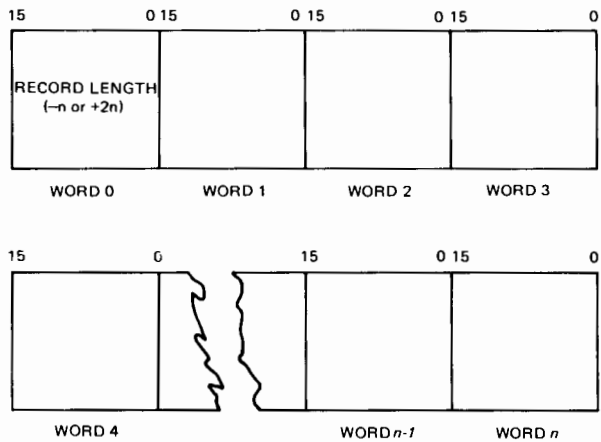


Each word represents two frames arranged as follows:



SIO RECORD FORMAT

Magnetic tape SIO binary records have the following format:

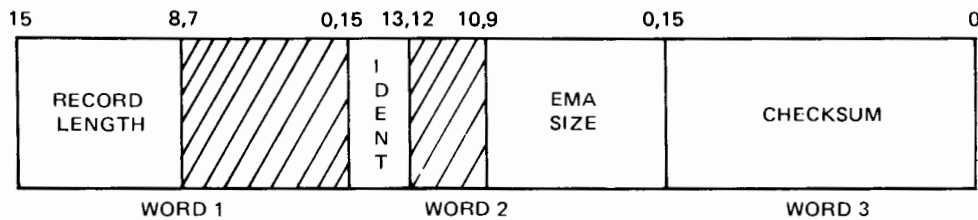


Record length = number of words or characters in record, excluding word 0; negative value denotes words, positive value denotes characters.

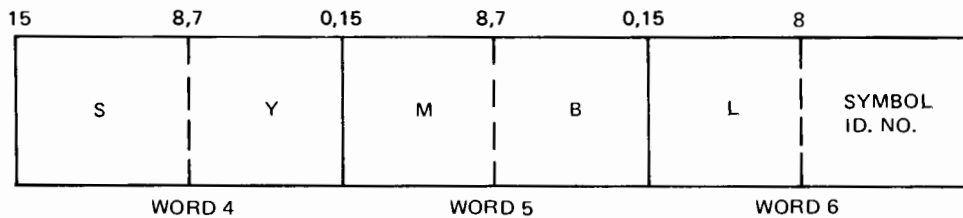
NOTE

The length (word 0) is not considered part of the data record. When written with the MS option of the DU command, the length is supplied by FMGR. When read with the MS option of the ST command, the length is removed (in this case, the length word is used instead of the length supplied by the driver).

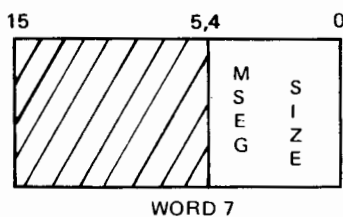
EMA RECORD



EXPLANATION
RECORD LENGTH = 7 WORD
IDENT = 110



SYMBOL ID. NO.: NUMBER ASSIGNED TO SYMBL FOR USE IN LOCATING REFERENCE IN BODY OF PROGRAM.



CARTRIDGE FORMATTING

APPENDIX

D

Cartridge formatting, or “subchannel initialization”, defines the disc track areas of RTE subchannels by writing their physical track and sector addresses in the preamble of each sector. The preamble of tracks containing system code are set to indicate they are write-protected.

A spare track is assigned to replace all defective tracks encountered during the initialization of a 7905/7906/7920 disc subchannel. The preamble of the defective track indicates that it is defective and gives the address of the spare track that replaces it. The preamble of the spare track indicates that it is acting as a spare, and gives the address of the defective track that it replaces. The disc controller will automatically switch to the spare track for all references to the defective track.

If the disc is new, if it has any write-protect flags written on its tracks, or if the subchannel definition of the pack is being changed, the disc must be reformatted.

CAUTION

Cartridge areas containing systems or data that must be preserved should not be initialized. Only the portion of the disc being written on during initialization gets reformatted.

At RTE generation time, program SWTCH formats the system subchannel during the transfer process. In addition, the RTE-IV SWTCH can format any additional subchannels specified during the generation process. Refer to Section V of the RTE-IV On-Line Generator Reference Manual for SWTCH documentation.

The UNIT COPY feature of the off-line disc backup utility can also be used to format discs. Refer to the section on Initializing Discs in the RTE Utility Programs Reference Manual for directions.

To ensure successful initialization when using the disc backup utility or SWTCH, the appropriate disc diagnostic must have been previously run to flag the bad tracks.

MAKING COPIES OF FMGR

APPENDIX

E

In a Multi-Terminal Monitor environment, efficient use of FMGR requires that a copy be made for each active terminal. This provides simultaneous processing with FMGR commands. Copies of program FMGR may be made as type 6 files. Such files are then set up as temporary programs with an ID segment in memory, but not on disc. They can be run with the RU command from RTE exactly as if they were program. Since the ID segments are temporary and will be lost whenever the system is brought down, you might want to restore the copies each time RTE is restarted from disc (re-booted).

This appendix describes two methods for making copies that serve two separate functions:

- Make single copy of FMGR from a terminal for immediate use
- Set up a procedure file "COPY" that can be run each time RTE is restarted from disc; the procedure file makes one copy of FMGR for each active terminal.

NOTE

The names of FMGR copies must start with the letters FM; it is good practice as a memory aid to associate each copy with a terminal logical unit number, for example, FMG07 for terminal 7.

E-1. MAKE SINGLE PROGRAM COPY

The following method makes a single copy of FMGR. You may use these FMGR commands from any active terminal to make a temporary copy for immediate use. The copy will be retained as a type 6 file.

```
:SP,FMGR ← create type 6 program file FMGR
:RN,FMGR,FMG07 ← assign new name to file
:RP,FMG07 ← assign ID segment to FMG07
```

Program FMGR is not affected by this process; the program file FMG07 can now be used from any terminal to perform all the FMGR functions except batch job processing and spooled job processing.

E-2. MAKE MULTIPLE COPIES AT SYSTEM START-UP

A procedure file "COPY" is established that will make copies for each terminal in the multi-terminal environment if COPY is executed whenever the RTE system is restarted from disc (booted-up). The procedure assumes that FMGR has been saved as a file with the command:

```
:SP,FMGR
```

Procedure File COPY:

```
:RN,FMGR,FMG02
:RP,FMG02
:RN,FMG02,FMG05
:RP,FMG05
.
.
.
:RP,FMG0n
:RN,FMG0n,FMGR
::
```

← assign new name to FMGR file
← assign ID segment to FMG02
← rename the file for next copy and assign ID segment to this copy
← continue for as many copies as are needed
← restore last copy and rename it FMGR (so "COPY" will work the next time)
← exit from file

When the system is restarted from disc, run FMGR and enter the following command:

```
::COPY ← transfer to COPY
```

Copies can be made of any other programs using this same method. For example, if you want more than one copy of program EDITR, use the same code substituting EDITR for FMGR. Note that only FMGR requires the copies to start with the same two letters as the original program. EDITR copies could be given your own name, for instance.

The procedure file to make copies can be incorporated in a general file that performs other system initialization functions. If this file is called WELCOM, it will be run automatically at each system startup.

The first time the system is started, FMGR will initialize (refer to paragraph 7-4) and then attempts to TR to WELCOM and, since WELCOM does not exist, will print FMGR - 006 and transfer to logical unit 1 (the system console). You should at this time save FMGR as a file and then create the file WELCOM. For example:

```
:SP,FMGR
:ST,1,WELCOM
} enter commands for WELCOM
Dc
```

Subsequent system starts will cause the file WELCOM to be run.

Refer to Section II, paragraph 2-27 for a description of type 6 program files; to paragraph 2-41 for procedure files and the command : that transfers to them; to paragraphs 2-33 and 2-34 for SP and RP commands; to 2-25 for RN.

GLOBAL EQUIVALENCE TABLE

APPENDIX

F

S	G	P
∅	-2	-48 Type
		-47 1
		-46 2
		-45 3
1	1	-44 Type
		-43 1
		-42 2
		-41 3
2	∅	-40 Type
		-39 1
		-38 2
		-37 3
3	1	-36 Type
		-35 1
		-34 2
		-33 3
4	2	-32 Type
		-31 1
		-30 2
		-29 3
5	3	-28 Type
		-27 1
		-26 2
		-25 3
6	4	-24 Type
		-23 1
		-22 2
		-21 3
7	5	-20 Type
		-19 1
		-18 2
		-17 3
8	6	-16 Type
		-15 1
		-14 2
		-13 3
9	7	-12 Type
		-11 1
		-10 2
		-9 3
10	8	-8 Type
		-7 1
		-6 2
		-5 3
11	9	-4 Type
		-3 1
		-2 2
		-1 3
12	10	∅ Type
		1 1
		2 2
		3 3
13	11	4 4
		5 5
		6 6
		7 7



The standard values are shown within dark lines.

A

AB command, FMGR, 2-78
 GASP, 5-6
 RTE, 2-3
 Absolute binary program (see Type 7)
 Absolute tape format, C-17
 Actual DCB buffer size, 3-5
 AN command, FMGR, 2-18
 APOSN call, FMP, 3-34
 ASCII character set, A-1
 Auxiliary cartridge (disc), 2-80
 initialize, 7-3
 change, 2-89

B

Bad tracks, 2-86
 Batch job control, 2-71
 spooled, 2-73, 4-1
 Batch logical unit switch table, 2-72; 4-5
 Batch mode, 2-1
 Batch processing, 1-13
 spooled, 1-14; 4-1
 Batch-Spool Monitor, 1-1
 components, 1-3
 configuration, 7-1
 environment, 1-1
 installation, 7-1
 Block, 1-6
 BR command, RTE, 2-3
 Buffer, DCB, 1-6; 3-1, 5
 spool files, 4-11, 14; 6-8, 9

C

CA command, FMGR, 2-65
 Cartridge, 1-5; 2-80
 copy, 2-95
 dismount, 2-88
 file allocation, 1-6
 formatting, D-1
 initialize, 2-83
 list, 2-90
 pack, 2-92
 status, 3-44
 Cartridge directory, 1-6, 7; 2-80
 list, 2-90
 format, C-4
 Cartridge reference, ICR parameter, 3-7
 namr parameter, 2-8
 Change outspool options, CS command, 4-15
 SMP call, 6-8
 Change purge to save, SMP call, 6-5
 Change record position, SMP call, 6-10
 Change save to purge, SMP call, 6-6
 CJ command, GASP, 5-6
 CL command, FMGR, 2-90

Clear buffer flag, SMP call, 6-9
 CLOSE call, FMP, 3-18
 CN command, FMGR, 2-33
 CO command, FMGR, 2-94
 Command string passing, 2-13
 Components of system, 1-3
 Conditional skip, 2-67
 Configure system, 7-1
 Copy, cartridge, 2-95
 files, 2-23, 2-28
 CR command, FMGR, 2-18, 20
 disc file, 2-19
 non-disc device, 2-20
 CREAT call, FMP, 3-8
 Create files, 2-18
 disc files, 2-19, 3-8
 non-disc files, 2-20
 program files, 2-43, 44
 CS command, FMGR, 4-15
 GASP, 5-8

D

DA command, GASP, 5-16
 Data Control Block, 3-1
 format, C-2
 Data transfer, FMGR, 2-23, 28
 FMP calls, 3-2
 DC command, FMGR, 2-88
 DCB buffer, 1-6; 3-1
 size, 3-5
 De-allocate spool system, 5-15
 Device, definition, 1-5, 6
 driver types, C-1
 logical unit assignment, C-1
 Directory, cartridge, 1-7; 2-88; C-4
 file, 1-7; 2-31; C-5
 Disc, definition, 1-5
 organization, 1-7
 (also see Cartridge)
 Disc file, 1-6
 create, 2-19; 3-8
 purge, 2-22; 3-12
 record formats, C-12
 Dismount cartridge, 2-88
 Display parameters, 2-63
 DJ command, GASP, 5-4
 DL command, FMGR, 2-91
 DP command, FMGR, 2-63
 D.RTR program, 1-3
 configuration, 7-2
 Driver types, C-1
 DS command, GASP, 5-7
 DU command, FMGR, 2-28
 DVS43 spool driver, 1-3, 4
 configuration, 7-4, 5
 use, 4-5, 6, 7

E

EO command, FMGR, 2-74
 spooled jobs, 4-10
 EOF control, DU command, 2-29
 ST command, 2-26
 Error codes, B-1
 FMGR codes, B-5
 FMP codes, B-2
 GASP codes, B-10
 outspool, B-16
 SMP, B-15
 spool, B-14
 unnumbered, B-12
 EX command, FMGR, 2-14
 GASP, 5-3
 Extents, definition, 1-6
 us, 1-10
 truncation, 3-19
 EXTND program, 1-4
 configuration, 7-4, 5

F

FCONT call, FMP, 3-41
 File access, 1-10; 3-3
 FMP calls, 3-21
 open options, 3-17
 File, 1-5
 create, 2-18, 19, 20, 23, 43, 44; 3-8
 purge, 2-21; 3-12
 File directory, 1-6, 7, 8; 2-83, 91
 formats, C-5, 6
 list, 2-31
 File Management Package (see FMP)
 File Manager (see FMGR)
 File name, 2-8, 3-6
 FMGR request, 2-12
 FMP calls, 3-6
 File number, DU command, 2-29
 ST command, 2-26
 File organization, 1-5
 File size, CREAT call, 3-9
 namr parameter, 2-8
 File status, 3-32
 spool files, 4-24; 5-8
 File type, 1-8, 9, 10
 CREAT call, 3-9
 namr parameter, 2-8
 READF call, 3-22
 WRITF call, 3-27
 Formats & tables, C-1
 FMGR operator commands, 1-12; 2-1
 error codes, B-5
 structure, 2-6
 summary, 2-5
 syntax rules, 2-7
 FMGR program, 1-3
 configuration, 7-2
 copies, 2-2; E-1
 execution, 2-11, 12
 initialization, 7-3

operation, 2-10
 termination, 2-14
 FMP, 1-3
 cartridge, 2-80
 configuration, -1
 library, 1-3
 FMP program calls, 1-11; 3-1
 summary, 3-2
 syntax, 3-4
 Function code, 3-42
 FSTAT call, FMP, 3-44
 GASP operator commands, 1-15/16; 5-1
 summary, 5-1, 2
 GASP program, 1-3, 4; 4-2
 error codes, B-10
 execution, 5-2
 initialization, 7-9
 Global parameters, 2-54
 assign, 2-64
 calculate, 2-65
 display, 2-63
 equivalence, 2-57
 format, 2-58

H

Hold spool file, 4-11; 6-3

I

IBUF parameter, 3-6
 IBUFR parameter, 6-2, 3
 ICR parameter, 3-7
 ID segment, 1-6
 use, 2-37, 46, 47
 IDCB parameter, 3-5
 IDCBS call, FMP, 3-46
 parameter, 3-7
 IERR parameter, 3-6
 IF command, FMGR, 2-67
 IL parameter, READF, 3-21
 WRITE, 3-27
 IN command, FMGR, 2-83
 Initialization, cartridge, 2-83
 FMGR, 7-3
 GASP, 7-8
 Input device, 2-11, 12
 Inspooling, 4-2, 16
 Interactive mode, 2-1
 Interrupting FMGR, 2-3
 IOFF parameter, 3-31, 34
 IOPTN parameter, 3-14
 IR parameter, 3-36
 IRB parameter, 3-31, 34
 IREC parameter, 3-31, 34
 ISECU parameter, 3-6
 ISIZE parameter, 3-8
 ISLU PARAMETER, 6-2, 5
 ISMP parameter, 6-5
 ISTAT parameter, 3-45
 ITRUN parameter, 3-18
 ITYPE parameter, 3-8

J

JLU parameter, 3-32
 JO command, FMGR, 2-73
 spooled jobs, 4-9
 JOB program, 1-4; 4-2
 execution, 4-17
 JOBFIL file, 4-2, 24
 format, C-7
 Jobs, batch, 2-71, 73
 spooled, 4-1
 JREC parameter, 3-32
 JSEC parameter, 3-32
 JTY parameter, 3-32

K

Kill spooling, 5-11
 KS command, GASP, 5-11

L

LEN parameter, 3-22, 23
 LG area, 2-36
 assign tracks, 2-41
 move to, 2-42
 LG command, FMGR, 2-41
 LI command, FMGR, 2-31
 List device, 2-11
 change, 2-14
 List file contents or directory, 2-31
 LL command, FMGR, 2-14
 LO command, FMGR, 2-15
 LOCF call, FMP, 3-31
 Lock cartridge, 2-81
 change, 2-15
 Logical source areas, 2-36
 Logical unit, 1-6
 cartridge reference, 2-9, 79; 3-7
 namr parameter, 2-8
 spool, 4-6, 10, 11
 standard assignment, C-1
 switching, 2-72, 77; 4-5
 LS areas, 2-36
 move to, 2-38
 set pointer, 2-40
 LS command, FMGR, 2-40
 LU command, FMGR, 2-77
 spooling, 4-10

M

Master security code, 1-11; 2-83
 assign, 7-3
 change, 2-86
 MC command, FMGR, 2-81
 Memory-image program, 2-37
 restore, 2-46
 save, 2-44
 (also, see Type 6 files)
 Memory requirements, 1-2; 7-1

Messages, to console, 2-17
 to list device, 2-18
 from job, 2-62
 Mount cartridge, 2-81
 MR command, FMGR, 2-42
 MS command, FMGR, 2-38
 Multi-Terminal Monitor, 1-13; 2-2
 copies of FMGR for, E-1

N

NAMF call, FMP, 3-47
 Namr parameter, 2-8
 Non-disc device, 1-6
 control, 2-33; 3-41
 create as file, 2-20
 position, 3-36
 Non-disc file, 1-8
 create, 2-20
 purge, 2-22
 formats, C-14
 NUM parameter, READF, 3-23
 WRITF, 3-28
 Number of files, DU command, 2-30
 ST command, 2-26
 NUR parameter, 3-36

O

OF command, FMGR, 2-50
 OPEN call, FMP, 3-13
 Open files, 1-10; 3-13
 exclusive use, 3-16
 update 1-10; 3-16
 Outspool, attributes, 4-12
 error messages, B-16
 logical unit, 4-11, 12
 manipulation, 5-7
 SPOPN call, 6-2
 Outspool options, 4-11; 6-3
 change, 4-15; 6-5
 Outspooling, 4-2, 24

P

PA command, FMGR, 2-62
 Parameter syntax, FMGR commands, 2-7
 Parameters, global, 2-54
 FMP calls, 3-4, 5
 Parsing routine, FMGR commands, 2-7
 Pass outspool, CS command, 4-15
 SMP call, 6-2
 Passing command strings, 2-13
 PK command, FMGR, 2-92
 Position file, 3-31
 READF call, 3-23
 spool files, 6-10
 WRITF call, 3-28
 POSNT call, FMP, 3-36
 POST call, FMP, 3-48
 Priority, spooled jobs, 4-8
 JO command, 4-9

P (Continued)

JOB program, 4-17
 LU command, 4-10
 new priority, 4-15; 6-8
 SPOPN call, 6-2
 Privileged commands, 2-5, 8
 Procedure files, 2-54
 in batch jobs, 2-69
 Program calls, FMP, 3-4
 Program development, 2-36
 example, 2-53
 Program files, 2-36
 create, 2-43, 44
 save, 2-43, 44
 release, 2-51
 restore, 2-46
 Programs used by BSM, 1-3, 4
 PU command, FMGR, 2-22
 PURGE call, FMP, 3-12
 Purge files, 2-22, 3-12
 spool files, 4-11, 15, 6-6

Q

Queue for outspooling, CS command, 4-15
 SMP call, 6-7

R

Read record from file, 3-21
 READF call, FMP, 3-21
 Record, definition, 1-6
 Record format, disc file, C-12
 DU command, 2-29
 non-disc file, C-14
 ST command, 2-24
 Record format, magnetic tape, SIO, C-18
 Record size, CREAT call, 3-8
 namr parameter, 2-9
 Release tracks, 2-52
 Relocatable object programs (see Type 5 files)
 Relocatable tape formats, C-14
 Rename file, FMGR command, 2-35
 FMP call, 3-47
 Retrieve record position, SMP call, 6-10
 Rewind magnetic tape, FMGR command, 2-33
 FMP call, 3-39
 spool file, 4-15
 RN command, FMGR, 2-35
 RP command, FMGR, 2-46
 RS command, GASP, 5-10
 RT command, FMGR, 2-52
 RU command, FMGR, 2-48
 RTE, 2-11, 5-2
 RUIH command, FMGR, 2-48
 RWNDF call, 3-39

S

SA command, FMGR, 2-43
 Save spool file, CS command, 4-14

SMP call, 6-5
 SD command, GASP, 5-14
 SE command, FMGR, 2-64
 Sector, 1-5
 Security, 1-10, 11
 Security code, FMP calls, 3-6
 namr parameter, 2-9
 Set buffer flag, SMP call, 6-8
 Setup buffer, IBUFR, 6-2, 3
 Severity code, assign, 2-11
 change, 2-16
 in batch jobs, 2-72
 SIO record format, C-18
 SMP program, 1-4, 15/16; 4-2, 24
 calls to, 6-1
 error messages, B-15
 SP command, FMGR, 2-44
 SP.CL program, 7-5
 SPLCON file, 4-2, 23
 format, C-9
 Spool control, programmatic, 6-5
 Spool errors, 4-23, 25; B-14
 GASP, B-10
 I/O abort, B-14
 Spool files, 4-4
 assign, 4-12
 manipulate, 5-7
 position, 6-10
 Spool logical units, 4-5, 6
 assign, 4-11
 switch, 4-6, 7, 10
 Spool Monitor, 1-14
 configuration, 7-4
 use, 4-1
 Spool pool files, 4-4
 Spool setup, 4-5, 8, 10
 change, 4-15
 programmatic, 6-1
 SPOPN call, 6-2
 Spool status, 4-25
 change, 5-6, 8, 10
 Spool system, configuration, 7-4
 de-allocate, 5-16
 initialization, 7-9
 restart, 5-15
 shut down, 5-14
 Spooled job manipulation, 5-4
 Spooling, 1-14
 batch, 4-1
 non-batch, 4-3; 6-1
 SPOPN call, SMP, 6-2
 SPOUT program, 1-4; 4-2, 23
 ST command, FMGR, 2-23
 SU command, GASP, 5-15
 Subfiles, 2-25
 SV command, FMGR, 2-16
 System cartridge (disc) 1-6; 2-86
 initialize, 7-3
 SY command, FMGR, 2-96

T

TE command, FMGR, 2-17
Terminate FMGR, 2-14
Terms, file management, 1-5
Time limit, job, 2-73; 4-9
 run time, 2-76
TL command, FMGR, 2-76
Top of form, line printer, 2-33, 34, 3-42, 43
TR command, FMGR, 2-59
Track, 1-5
 directory, 2-83
 FMP, 1-7
 system, 1-7; 7-3
Transfer file (see Procedure file)
Type 0 file, 1-8
 create, 2-20
 data transfer, 3-3, 21
 open options, 3-14
 position, 3-37
 purge, 2-22
Type 1 file, 1-8, 9
 access, 3-17
 data transfer, 3-3
 format, C-12
 position, 3-37
Type 2 file, 1-8, 9
 data transfer, 3-3
 format, C-12
 position, 3-37
Type 3 file, 1-8, 9
 access, 3-21

 format, C-12
 position, 3-37
Type 4 file, 1-8, 9
 source program, 2-36, 43
Type 5 file, 1-8, 9
 object program, 2-36, 42, 43
 paper tape format, C-14
Type 6 file, 1-8, 9
 format, C-13
 memory-image program, 2-36, 37, 44, 46
Type 7 file, 1-8, 10
 paper tape format, C-17

U

Unnumbered error messages, B-12
Update open, 1-10; 3-15, 16
User buffer, 1-6
 use, 3-6, 31, 26

W

Write EOF and queue for outpooling,
 CS command, 4-15
 SMP call, 6-7
Write record to file, 3-26
WRITF call, FMP, 3-26
XE command, JOB, 4-21
?? command, FMGR, 2-13
 GASP, 5-3
**command, FMGR, 2-96

READER COMMENT SHEET

**BATCH-SPOOL MONITOR
Reference Manual**

92060-90013

June 1978

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?

Is this manual complete?

Is this manual easy to read and use?

Other comments?

FROM:

Name _____

Company _____

Address _____

FOLD

FOLD

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States Postage will be paid by

Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014
ATTN: Technical Marketing Dept.

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA



FOLD

FOLD

PART NO. 92060-90013
Rev. Code 1805
Printed in U.S.A. 5/78

HEWLETT  PACKARD

Sales and service from 172 offices in 65 countries.
11000 Wolfe Road, Cupertino, California 95014