PROGRAMMING AND OPERATING MANUAL

Part No. 29033-98000

REAL-TIME EXECUTIVE
FILE MANAGER SYSTEM

– IMPORTANT NOTICE –

This manual contains information on Hewlett-Packard Real-Time Executive Software. The reader is assumed to be a programmer familiar with one of the Hewlett-Packard programming languages, ALGOL, Assembler or FORTRAN.

Microfiche No. 29033-98001 **Printed MAR 1973**

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

HEWLETT PACKARD

AUTOMATIC MEASUREMENT DIVISION

MANUAL UPDATING SUPPLEMENT
for

REAL-TIME EXECUTIVE FILE MANAGER PROGRAMMING AND OPERATING MANUAL

DATE: AUG. 1973

Manual Part No.	29033-98000
Microfiche No.	29033-98001
Manual Printed:	MARCH 1973

This supplement is used to report improvements made to the product after the manual printing. The supplement is also used to report manual errors and to provide improved or additional operating and service instructions.

CHANGE NO.	DESCRIPTION
1	This updating supplement contains Appendix I which provides the File Manager subroutine calls detailed in HP FORTRAN. Add the Appendix to the rear of the File Manager Manual.
2	Page 2-14, IR parameter. Correct as follows: IR = 0 or not present IR ≠ 0
3	Page 2-17, last sentence under the LEN parameter. Correct as follows: If LEN = -1, then an end-of-file has been read.
4	Page 2-23, under "f". Correct as follows: USERF*x* USERF*y*
5	Page 3-4, under heading Record Size. Add a colon in the ABS example following the -10 digit, as follows: ABS: -10: -3: 2: 40: 6
6	Page 3-20 in the PURGE box under where. Strike out "or logical unit number" as follows: *namr* is the name of the file to be purged.
7	Page 3-25, under "C". Every place SEG10 appears, change to SUB10.

Printed in U.S.A.

974 E. Arques Ave., Sunnyvale, California 94086. Telephone 408-735-1550

CHANGE NO.	DESCRIPTION
8	Page A-3, Table A-3, right hand column. Replace the portion of the table in the manual with the new one provided in this supplement.

For type 0 files, entries 2 through 6 are:

2	0 (type)
3	Logical unit number
4	End-of-file code
Specified at Creation	01XX for MT 10XX for paper tape 11XX for line printer

NOTE

XX = logical unit number

5	Spacing legal code Bit 15 = 1 backspace legal Bit 0 = 1 forward space legal
6	READ/WRITE Code Bit 15 = 1 input legal Bit 0 = 1 output legal

The DCB is free for other use after a PURGE, CLOSE and NAMF call. Once a file is open, the DCB is used to reference the file, the name no longer being needed or used.

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

GENERAL DESCRIPTION

SECTION I
GENERAL DESCRIPTION

INTRODUCTION

The HP Real-Time Executive File Manager Package (FMP) is a programmer/operator interface to the Real-Time Executive (RTE) that is used to organize and systematize files on the RTE System to make access and maintenance easier. The FMP maintains files on an HP Real-Time System on one or more discs, and provides a uniform access method to all standard HP Input/Output devices. The FMP provides easy-to-use access methods in the form of operator commands entered through the system input device, or user program commands accessing one of the many FMP subroutines. All details of file access, including multiprogramming protection, are built into the FMP.

CAUTION

To avoid disruption of data storage, the disc controller and removable disc cartridge must not be exposed to an area where a strong magnetic field may be present.

The File Manager performs the following functions:

- Provides security on system and file level to prevent unauthorized access.
- Creates new files; modifies existing files; and purges obsolete files.
- Provides exclusive control of I/O devices, other than discs, which use standard peripheral calling sequences.

SOFTWARE

Table 1-1 is a list of the software and other components that make up the File Manager Package.

Table 1-1. FMP Components

Description	Part Number
FMP Program Tapes	29033-60001 1 of 3 29033-60001 2 of 3 29033-60001 3 of 3
FMP Library Tape	29033-60002 1 of 1
FMP Manual	29033-98000
FMP Listings	29033-98002

The File Manager operates within the HP RTE software environment as follows:

HP RTE Software compatible with the File Manager

29014A or later RTE Moving Head Generator

29015A or later RTE Fixed Head Generator

29016D or later RTE System.

29022A or later RTE Relocating Loader

29029A or later RTE Multiple-Device System Control Driver.

HARDWARE

The File Manager operates within the HP RTE hardware environment which consists of an HP 9600 Series System.

HP 9600E – Disc-based Real-Time Executive using moving head disc.

HP 9600F – Disc based Real-Time Executive using fixed-head-per-track disc.

FMP PHILOSOPHY

A file management system has the task of organizing information into an easy-to-use scheme. However, what is a file management system? Because of the many different things that can be filed, and the many different ways they can be filed, there are many different views on exactly what a filing system is all about. The HP File Manager system (FMP) is versatile enough to allow a user to configure his own filing system to suit his individual needs. But before an attempt is made to describe the technical aspects of the FMP, including all the versatility, a common ground must be established with regards to file management. A simple analogy can be used that will aid in understanding what it does. Refer to Figure 1-1 for the analogy example. For the more technically oriented individual that understands a file management system, refer to the technical discussion later in this section. For the uninformed – read on.

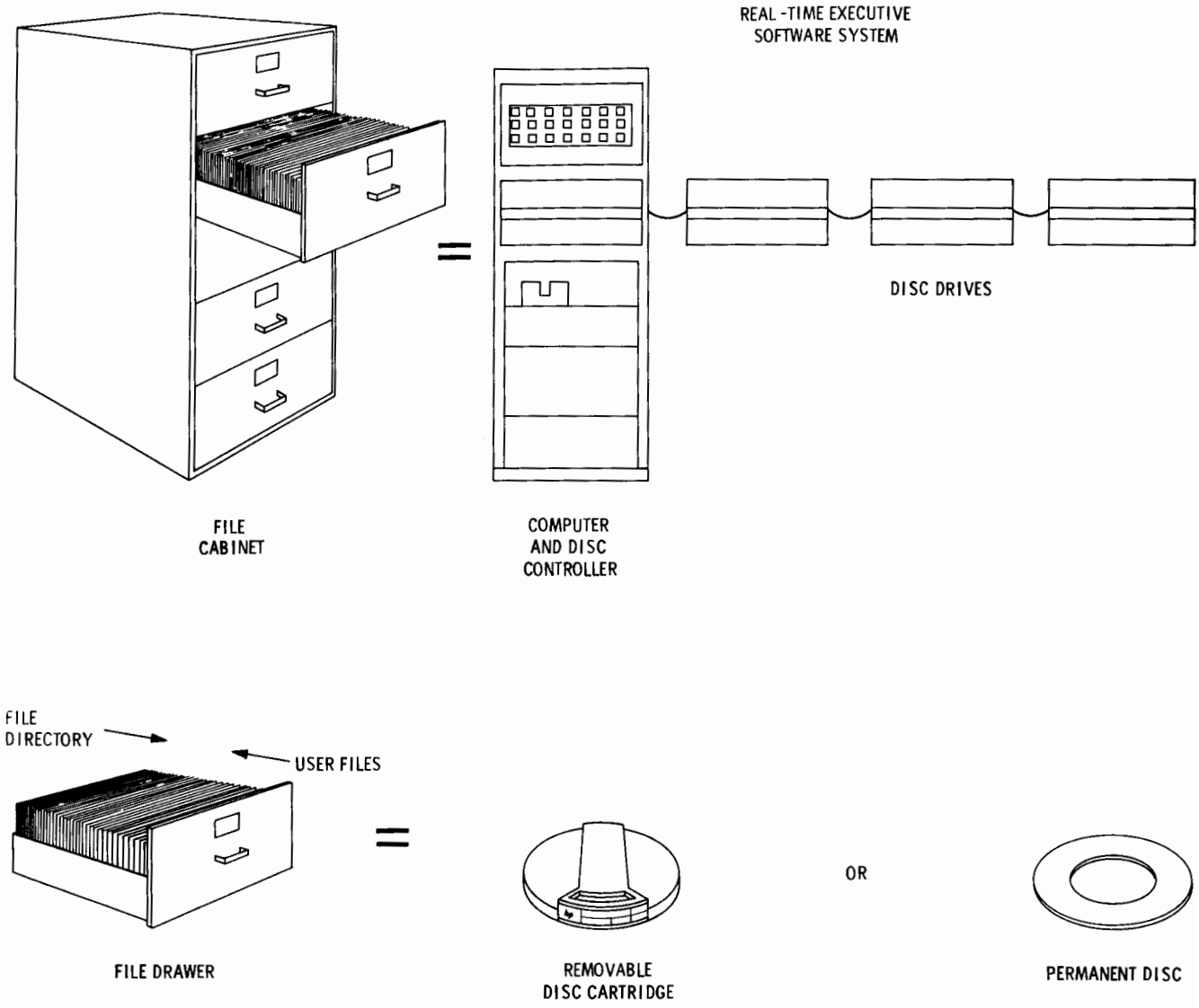


Figure 1-1. Analogy of the RTE FMP

FMP GENERAL DESCRIPTION

As soon as anyone accumulates a few thousand of anything (except money), he may have a problem storing his accumulation, and then finding what he needs when it is required. As the collection grows and as time passes, the memory dims and the point is reached when this file grows so large that needed references cannot be found, and that forgotten items are without access. There are then, three basic problems to overcome;

Problem	Solution
• Where do you store vast collections?	Within the HP RTE System
• How do you manage the collections?	With the HP RTE File Manager.
• How do you remember what's in the collection?	Through a unique index scheme of your own built upon the index features of the File Manager.

SOLUTION DEFINITIONS

HP RTE System is a multiprogramming system that allows several programs to operate concurrently, each program executing during the unused central processor time of the others. (A complete description is contained in the RTE Programming and Operating Manual 02005-90001.) RTE can provide from 2.5 million to 10 million 16-bit words of storage on fixed head or moving head discs, or a combination of both.

HP RTE File Manager is a programmer/operator interface to the RTE that is used to organize and systematize files on the RTE System to make access and maintenance easier.

Index Schemes come in many different forms for many different applications. The user must be concerned with a minimum of two indexes; one that the File Manager itself forms and maintains, and one that the user should form for cross-referencing purposes.

ANALOGY

As shown in Figure 1-1 the HP RTE System is likened to a large filing cabinet that is capable of storing a very large quantity of information. The HP RTE File Manager (FMP) can be compared to an office clerk that is responsible for taking care of the filing system. The responsibilities of the FMP, using the office clerk analogy, would be as follows:

- To set up the entire filing system with a system security code according to the dictates of the user. The user provides the individual file names and the type of each file. The user can also flag each file with a security code so the FMP can check another user's right to examine a particular file.

- The FMP will allow the user to:

  Add new files

  Open an existing file to:
  - Add records
  - Delete records
  - Change records
  - Copy records or information

  Close an opened file (return it to the system)

  Purge an obsolete file.

- The FMP will resolve conflicts between users who want to open the same file. A user may request that a file be opened exclusively to him. This will be granted by the FMP only if the file is not in use. A user may request that a file be opened non-exclusively, that is, he will share it. Each time a request is received to open a file, exclusively or non-exclusively, all users that currently have that file open are checked. If they are not currently using the file, it is closed to them and returned to the system. The FMP will grant the non-exclusive opening of a file only if it is not already exclusively open, or already being shared by more than seven users.

- The FMP can understand users that talk in FORTRAN IV, FORTRAN II, ALGOL, and HP ASSEMBLER.

- The FMP will communicate messages to the user that will inform him of mistakes he has made in filing. FMP will also communicate messages in regard to overall system performance.

The previously described responsibilities or qualifications are only a general indication of how the FMP manages and organizes user files. The FMP is versatile enough so that the user can configure the filing system to suit his individual needs. The FMP will recognize user requests via the written memo method (programming) where the user details his instructions completely, or by the simpler verbal directive method (entries through the system input device). Both methods are described in Sections II and III, respectively.

A key to effective use of the FMP is organization. How to organize large collections in order to find any single item easily. The FMP will do the actual filing and housekeeping but you must organize and index your items for efficient retrieval. Refer to Appendix H at the rear of this manual for additional information on indexing.

FMP TECHNICAL DISCUSSION

Many different terms related to the HP File Manager appear throughout this manual. For convenience these terms are all defined below.

CARTRIDGE — This term is used interchangeably with disc and means a rotating random access storage device. Cartridge usually implies removability.

FILE — A collection of records usually terminated by an End-Of-File (EOF). A file may have zero or more records in it. In the FMP, a disc file is usually designated by a unique six character name.

RECORD — A collection of 16-bit words usually terminated by an End-Of-Record (EOR) mark. A record may have zero or more words in it. FMP Record formats are described in detail under File Types.

EOF — An end-of-data mark of higher level than an EOR mark. The particular type of EOF mark is determined by the type of device referenced.

FMP EOF Conventions:

a. When the FMP is performing a non-disc READ:

   1. Magnetic tape, cassette, and card reader use bit 7 in the STATUS word, – or –

   2. A zero length record and the device is up (card reader encountering a blank card), – or –

   3. A zero length record and the device type is less than 10 (i.e., DVR00 to DVR09).

b. When the FMP is performing a non-disc WRITE:

   1. An EOF is requested for a magnetic tape or cassette, – or –

   2. A double space (two carriage return/line feeds) is entered on the teletype, – or –

   3. A page eject control function is used for the line printer, – or –

   4. A leader control request is used for paper tape.

c. When the FMP is performing a disc READ or WRITE, the EOF is defined by the file type (see File Types).

SECTOR — A sector is part of a disc track 64 words in length.

BLOCK — Two disc track sectors of 64 words each (total of 128 words).

FMP ORGANIZATION

File Organization. Files may exist on the disc tracks known by the system (i.e., system and auxiliary discs), and on peripheral discs. Tracks on the system and auxiliary discs are FMP assigned so that only FMP routines may write on them. Files are directed to preferred discs by the user. If a particular disc is not selected in the call, the FMP will direct the file to a convenient location – the user does not have to remember track and sector addresses, just the file name and security code.

Disc Organization. Disc files are located on contiguous tracks within the FMP area, with user files beginning on the first (lowest numbered) FMP track and working up, and directory entries for the user files beginning on the last (highest numbered) FMP track and working down. No file will be allowed to cross a disc boundary; however, the assigned file area may include several discs. All track numbers refer to relative track numbers, that is relative to the track on the given unit established as track 0 at system generation time. For fixed-head discs, physical track 0 and relative track 0 are the same. For moving-head discs, relative track 0 is established during RTGEN time.

NOTE

For moving-head discs, all starting track numbers must be the same if the cartridges are to be exchanged LU to LU or system to system. This is due to the fact that FMP uses relative track addresses in the directory.

All files are subject to moving as part of any packing operation done on a disc. This makes files relocatable; therefore, no absolute file addresses should be retained in any file or program.

Blocks are defined as two disc sectors of 64 words each. All files are constrained to start on even sector boundaries and all accesses, except to type 1 files, are 128-word accesses. Therefore, all accesses are directed to even sectors. No cyclic checking is done, and all disc errors are passed back to the caller for action. The error code is printed on the system log device when using the FMGR operator interface routine, or passed to the user program in parameter IERR when using one of the library subroutines.

BAD TRACKS

Bad track information is entered using the FMGR Initialization Command. This information is kept in the affected disc's first directory entry. If, at creation or during a packing move, the area for the file includes a bad track, the first track of that file will be increased until the file no longer contains any bad tracks. If the CREAT routine is to use the rest of the disc, the whole area will be above the highest bad track. If, during packing, a file is found to include a declared bad track, the file will be purged. (Note: This can only happen through an explicit declaration of a bad track by an operator who knows the system security code.) In this manner, only the PACK and CREAT routines need be aware of bad tracks. Bad tracks discovered by the FMP result in an error return to the caller.

DIRECTORIES

The FMP creates and maintains two indices. The master index, or disc directory, contains information on all discs assigned to the FMP, and the local index or file directory contains information on each file on that particular disc.

Disc Directory. The first two sectors of the last track of the system disc (LU2) that is assigned to the FMP contains information on all discs which are currently mounted and which contain tracks assigned to the FMP. This directory also contains a setup code word and the master security code. The layout of this directory is shown in Appendix A.

File Directory. Each disc which has tracks assigned to the FMP will contain a file directory. Each file directory entry uses 16 words. The file directory starts in sector 0 of the last track which is available to the FMP. The first entry in the file directory will pertain to the disc itself and will contain label and track information. Each subsequent entry will contain information on a file. The last entry will be followed by a zero word. The first entry and each entry for purged files has the sign bit set on its first word to indicate that it is normally to be ignored. Refer to Appendix A for the file directory format.

FILE TYPES

Files are divided into three categories shown below:

Category	Type	Description
Control	0	Non-disc file
Fixed lengths, random access, non-extendable	1	128-word record length
	2	User selected record length
Random length, sequential access, automatic extents	3	Random record length
	4	Source program
	5	Relocatable program
	6	RTE load module
	7	Absolute program
	>7	User defined

Type 0. A type 0 file is created with the CR Operator Command, and is used to control devices other than discs. Type 0 files also afford a measure of device independence, in that the standard file commands can be used to control the device. A type 0 file can only be created on the system disc (LU2); therefore, the file directory entry will be located in the system disc file directory, and have some special entries relating to logical unit number and end-of-file code. Cooperating programs may use type 0 files as a means of controlling access to a device (see exclusive open).

Type 1. Type 1 files have fixed record lengths of 128 words. Since the smallest addressable block is 128 words, short blocks will be filled from the user's buffer. Transfers to or from type 1 files are done directly to or from the user's buffer, and may be any length; thus, this type file has the fastest transfer rate. Positioning on type 1 files assumes a record length of 128 words. Any file except type 0 may be opened as a type 1 file.

Type 2. Type 2 files have record lengths that are user defined. Each transfer is one and only one record long, and must go through a packing buffer (128 word buffer in the Data Control Block); thus, the transfer rate will be slower than type 1 files.

Type 3. Type 3 files are packed on the disc and contain data, source, relocatable, etc. Each transfer is one and only one record long, and must go through a packing buffer (128 word buffer in the Data Control Block).

Type 4. Same as type 3, except the FMGR operator interface routine recognizes (defaults) type 4 files as containing source programs.

Type 5. Same as type 3, except the FMGR operator interface routine recognizes (defaults) type 5 files as containing relocatable binary code.

Type 6. Same as type 3. This file is created by the Save Program (SP) operator command as a type 6 file, but is always accessed as a type 1 file. Refer to RECORD FORMAT heading under type 6, and the SP Command for more information.

Type 7. Same as type 3, except the FMGR operator interface routine recognizes (defaults) type 7 files as containing absolute binary code.

Type $\geq$7. Same as type 3 – user defined.

All files, except type 0 files, may be accessed as type 1 (i.e., direct transfer through user's buffer); however, no other type modification is allowed. The major difference between the files is the speed of positioning. A type 1 or 2 file may be positioned without any disc accessing, whereas other type files require at least one, and in some cases, several accesses. The user in no case need worry about a file crossing sector or track boundaries.

Update Files. Update implies that a file is to be modified in some manner. It is usually desirable to modify only a portion of the file without disturbing the remainder of the contents.

The end-of-file mark of type 1 and 2 files is the last word of the last block in the file; however, in other type files it is a special word (– 1) for the length of a record. In order to preserve data in a type 2 file, it should be opened for update whenever any non-sequential accesses are to be performed, and the file is to be modified. That is to say, a type 2 file should be modified in non-update 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/non-update has no effect on reading or positioning.

In files of type 3 and above, an end-of-file mark is written in the non-update mode. In either mode, records written beyond an end-of-file are written in non-update fashion. That is, replace the end-of-file with the new record, followed by an end-of-file.

Extendable Files. An extendable file (type 3 and up) is a file that is automatically extended in response to a write request to points beyond the range of the currently defined file. The extent is created with the same name and size, and the access is continued. Extensions will be of the same size as the base file, and all extensions of any given file will be on the same disc. No flags or pointers relating to the next extent are kept in the file area. Open flags are kept only in the base file directory entry.

RECORD FORMATS OF FILES

Type 0. The type 0 file's record is defined by the device type. The following description describes type 0 files from paper tape and card devices. (These conventions are incorporated in the device driver and not the FMP.)

a. ASCII

   1. ASCII records are punched according to the table in Appendix E.

   2. On paper tape, records are ended with a line feed character (usually preceded by a carriage return). A rubout character anywhere in the record indicates that the record is to be ignored. Backspace characters cause the previous character and backspace character to be deleted from the record.

   3. On cards, records are terminated explicitly by the highest non-blank character position, less than or equal to the read request length.

b. Relocatable Binary Records.

   1. The complete format is given in Appendix F. The FMGR program uses the length and checksum words to do checksum on relocatable records.

c. Binary Records.

   1. Binary records have the length in word one (first character) and the rest is unspecified. These records are generated by binary write requests to the formatter directed to a paper tape device. A checksum is not performed.

d. Absolute Binary.

1. The Absolute format is given in Appendix G. This record type is used on tapes usually loaded by the Basic Binary Loader (BBL) or the Basic Binary Disc Loader (BBDL). The FMGR program uses the first and last words to perform a checksum on these records. It is also necessary to set subfunction bits (23_8) to read absolute records.

Type 1 and 2. Type 1 and 2 files are written as presented. They are packed, and may cross sector and track boundaries. The end-of-file is the last word of the last block in the file.

Type 3, 4, 5, 6, 7 and Above. Type 3 and above records are preceded and followed by a length word which contains the length of the record (exclusive of the two length words). A zero length record consists of two zero words. An end-of-file is indicated by a – 1 for the first length word. Words following the end-of-file are undefined.

Type 6. Program Save Files are created as type 6 files by the SP Command, and are always accessed as type 1 files by FMGR.

The first two sectors of the file are used to record ID information on the program as follows:

Word No.	Contents
0	– 1 (EOF if not forced to type 1)
1 thru 5	Not used
6	Priority
7	Primary entry point
8 thru 13	Not used
14	Program type
15 thru 16	Not used
17 thru 21	Time parameters
22	Low main address
23	High main address
24	Low base-page address
25	High base-page address
26 thru 27	Not used
28	Checksum of words 0 thru 27
29	System setup code word (same as disc directory word 125)
30 thru 127	Not used

The remainder of the file will be an exact copy of the original program tracks.

MULTIPROGRAMMING CONSIDERATIONS

File Conflict. In order to prevent two or more programs from destructively interfering with each other, a file may be opened either exclusively or non-exclusively.

Exclusive Open. An exclusive open is granted only to one program per file at a time.

Non-Exclusive Open. 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.

Directory Conflict. The FMP keeps open flags in the directory. This means that the directory will be modified by the following functions:

- Creating a file
- Opening a file
- Closing a file
- Purging a file
- Changing a file name.

Since all these actions may be programmed by the user, it is clear that there could be conflict. The FMP resolves this conflict by allowing only one program to write on the directory tracks. This program is the D.RTR routine. D.RTR is scheduled with wait whenever a directory change is required; that is, for all of the above functions. D.RTR must be either foreground disc resident or core resident for this reason. The D.RTR program will own outright the directory on LU2, and LU3, if any.

Security. There are two levels of security, file security and system security.

File Security. Each file has a security code. This code may be positive, negative or 0. If positive and the code at open does not match, the file may be read but not written on. If negative and the code at open does not match, the file will not be opened. If zero, the file may be opened to any caller with no restrictions.

System Security. During system setup, a system security code is entered. If zero, no security is given. If non-zero, the code must be known in order to get directory listings to include file security codes, and in order to redefine the FMP disc areas.

FMP PROGRAM CALLS

The programmer communicates with the FMP through the type 6 or 7 utility subroutines provided on the FMP library tape. The subroutines will perform the following functions:

- Position a file to a known record address
- Close a file
- Create a file
- Send RTE control request to a type 0 file
- Return 125 words of Disc Directory
- Return file status
- Rename a file
- Open a file
- READ/WRITE a specific record
- Purge a file
- READ a file
- Reset a file to first record; if type 0 file, a rewind request is generated.
- WRITE a file.

FMGR OPERATOR COMMANDS

The operator controls the FMP with operator requests entered through the system input device. The commands can be entered manually through the system teleprinter, or on paper tape, cards, or other form through the appropriate input device. The commands can also be stored in a file, with control being transferred to that file for execution.

The operator first gains the attention of the FMP by scheduling the software module FMGR with the ON FMGR operator command. When FMGR runs, a colon (:) prompter is returned on the initial keyboard input device. If the input device is not a keyboard, the first command is read from the device that is the input.

If desired, each command will be echoed on the system log device (refer to Severity Command). If an error is detected (input or reference), an error number is printed on the log device and input control is switched to that device. This halts the process and requires operator intervention to correct the error.

SECTION II

FMP PROGRAM CALLS

SECTION II
FMP CALLS

This section describes the basic format of Assembly Language calls to the various subroutines that interface the user program to the File Manager. Table 2-1 is a summary of the calls in this section and shows an interrelationship between the File Manager, program calls, Data Control Block (DCB), and user's program. The table is intended as an aid in understanding which calls to use for a given task, and how the DCB will be affected by the call. The error codes associated with the FMP calls are listed in Section IV. (Note that all error codes are also returned in the computer's A-Register.) Refer to Appendix C at the rear of this manual for a summary of the FMP calls and required parameters.

SUBROUTINE STRUCTURE

All the interface routines are type 6 or 7 utility subroutines and are located on a single library tape. Type 7 utility subroutines are subroutines which cannot be shared by several programs because of internal design or I/O operations. A copy of the utility routine is appended to every program that calls for it. The RTE library subroutine FRMTR, which carries out FORTRAN I/O operations, and the PAUSE subroutine are examples of type 7 utility routines. Type 6 re-entrant or privileged library routines are sharable and may be loaded in the core resident library area at system generation. The FTN IV Formatter is an example of a re-entrant routine, and .ENTR (the subroutine entry routine) is an example of a privileged routine.

When RTGEN creates the disc-resident RTE System, all library subroutines not included in the resident library are stored on the disc in relocatable format as utility routines. They are appended to programs by the RTE Relocating Loader in its background loading process (see RTE Relocating Loader in the RTE Manual).

DEFINITIONS

DATA CONTROL BLOCK (DCB)

The DCB is a 144 word array provided by the user program with one DCB required for each file opened. Once a file is open, the DCB is used to reference the file, the name no longer being needed, or used. The DCB includes a 128 word buffer area (block) that is used for data transfers.

BLOCK

A block is two sectors of 128 words.

SECTOR

The RTE moving head disc driver regards a track as having a user defined number of sectors of 64 words each. For programming purposes, all disc transfers are a maximum of 128 words per transfer (one block) except for type 1 files. (Type 1 files do not use the DCB buffer area and are therefore not restricted to 128 word transfers.)

DISC DIRECTORY

The first two sectors of the last track of the system disc (LU2) are devoted to the master index, or disc directory. The disc directory is always located on the system disc, and contains information on all discs which are currently mounted, and which contain tracks assigned to the FMP. This directory also contains a setup code word and the master security code. The layout or format of this directory is shown in Appendix A.

FILE DIRECTORIES

Each disc which has tracks assigned to the FMP will contain a file directory. Each file directory entry uses 16 words of a sector. The file directory starts in sector 0 of the last track which is available to the FMP. The first entry in the file directory will pertain to the disc itself and will contain label and track information. Each subsequent entry will contain information on a file. The last entry will be followed by a zero word. The first entry and each entry for purged files, has the sign bit set on its first word to indicate that it is a label and not the same name as a file (normally to be ignored). Refer to Appendix A for the file directory format. Note that while it is possible to have two files with the same name located on different cartridges, it is not possible to have two files with the same name on the same cartridge.

D.RTR

D.RTR is a foreground program which is an integral part of the FMP. D.RTR is the only program which is allowed to write on the directories, and may only be scheduled by some other program. D.RTR is never directly called by the user, it is called by the user interface subroutines only. D.RTR is called by a schedule with wait call to the RTE EXEC, and five parameters define the function desired. D.RTR will return up to five parameters which the calling routine may access with RMPAR.

Table 2-1. FMP Calls and Data Control Block Relationship

Subroutine Entry Point	Function	DCB Status At Entry	DCB Status After Call	Directory Access
APOSN	Positions a file to a known record address	MBO	OPN	EXTENTS
CLOSE	Closes a file	MBO	CLO	YES
CREAT	Creates a file	CBO	OPNEX	YES
FCONT	Sends RTE control request to type 0 files	MBO	OPN	NO
FSTAT	Returns 125 words of Disc Directory	–	–	–
LOCF	Returns File status	MBO	OPN	NO
NAMF	Renames specified file	CBO	CLO	YES
OPEN	Opens desired file	CBO	OPN	YES
POSNT	Directs next READ/WRITE to a specific record	MBO	OPN	EXTENTS
PURGE	Removes file and directory entry	CBO	CLO	YES
READF	Transfers one record from file to user buffer	MBO	OPN	EXTENTS
RWNDF	Resets file to first record; if type 0 file, a rewind request is generated	MBO	OPN	EXTENTS
WRITF	Transfers one record from user buffer to file	MBO	OPN	EXTENTS

Legend:

MBO	=	Must be open	OPNEX	=	Open exclusively
OPN	=	Open	EXTENTS	=	Directory is accessed if there is a change of extents as a result of the call.
CBO	=	Can be open – if DCB is open, the current file opened to the DCB will be closed.			

Summary:

- As indicated from the table a file must be open to a DCB before making any of the following calls.

APOSN	FCONT	POSNT	RWNDF
CLOSE	LOCF	READF	WRITF

- The only calls that result in a closed DCB being left open are OPEN and CREAT.
- The above conditions imply that the general structure of a program using files be as follows:

A. To open a DCB, call { OPEN, CREAT }

B. To do something with a file, call { APOSN, FCONT, LOCF, POSNT, READF, RWNDF, WRITF }

C. To close a file, call CLOSE

–OR–

D. To close a file and open another, call { OPEN, CREAT }

- A file should always be closed before program termination. Failure to do so may result in the last block not being posted on the disc.

LOGICAL READ VS. PHYSICAL READ

A logical read occurs each time the user requests a record from a type 2 and above file. At that time FMP checks the specified Data Control Block (DCB) buffer area to determine if the requested record is already in core. If in core, the record is transferred to the user's record buffer without actually physically reading the disc. If the record is not present in core, the necessary disc transfers are performed (physical reads – and writes, if necessary) to bring the record into core. If the record size is less than 128 words, then enough records (including possible partial records) are brought into core at once to total 128 words.

LOGICAL WRITE VS. PHYSICAL WRITE

A logical write occurs each time a user requests that a record be written to a type 2 and above file. At that time, FMP determines if that record is present in the DCB buffer area; if it is, FMP simply transfers the data in the user's record buffer to the DCB buffer area and flags it as "must be written." Each succeeding read or write is treated in the same manner until one of the following occurs:

a. A logical record transfer occurs for which the record is not wholly in core.

b. Until the file is closed.

c. Until the file is repositioned to another block.

In any one of these cases, the FMP must physically write on the disc (post) the records in the DCB buffer area. Following this posting, any remaining words are moved to the DCB buffer area.

If the record is not present in core on a write request, FMP locates the record on the disc and if update mode, transfers it physically into the DCB buffer area. The data to be written is then transferred from the user buffer to the DCB buffer area and flagged as "must be written." The read before write is necessary because records do not necessarily fall on sector boundaries in the disc. If a CLOSE, POSNT, APOSN, or RWNDF request occurs, all buffers flagged are written to the disc.

NOTE

Type 0 and 1 files are transferred directly to or from the I/O device or disc (posted immediately).

PARAMETERS

Most of the 13 subroutine calls shown in Table 2-1 use common parameters. Rather than list the meanings of these parameters 13 times, they are described here in detail as a preface to all the calls. Any parameter that is unique to a call is described within the call itself.

GENERAL PARAMETERS

IDCB is a 144-word array to be used as a Data Control block (DCB) for each file opened. Note that the DCB is free for other use after a PURGE, CLOSE and NAMF call. (The DCB is fully defined in Appendix A.) Once a file is open, the DCB is used to reference the file, the name no longer being needed, or used.

NOTE

Since the DCB's exist in the user area and are not protected, caution must be taken not to modify them in any way.

IERR See Table 4-1 for error codes. For OPENs that are successful, the type of file is returned in IERR. For successful CREATs, the number of *sectors* is returned in IERR. Note that this is 1/2 the number of blocks in the file. This number is only useful if the CREAT specified the rest of the disc (see CREAT) but is always returned. IERR is also returned in the A-Register.

NAME is a six-character name array for or of the file. A legal file name is six ASCII characters in length. The first character must not be a blank or a number. All characters must come from the set ↑b thru ← (see Appendix E) exclusive of special characters "+" (plus), "–" (minus), "," (comma), and ":" (colon). Imbedded blanks are not allowed. Short (less than 6 characters) names must be padded with trailing blanks. In the FORTRAN calling sequence *name* must be converted from ASCII to octal and stored in the NAME array. Refer to the table in the RTE Manual for the ASCII to octal conversion. If FORTRAN IV is used, *name* can be written using Hollerith constants. Refer to the FORTRAN IV Manual.

IBUF is the read/write array. This parameter defines the buffer address to/from which reads/writes are to take place.

OPTIONAL PARAMETERS

Most of the subroutines have one or more optional parameters. These parameters always appear at the end of the calling sequence. If it is desired to use an optional parameter which is preceded by other optional parameters in the calling sequence, then all parameters up to the desired parameter must be supplied. In general, unsupplied optional parameters are assumed to be zero.

The two most commonly used optional parameters are the cartridge label (ICR) and security code (ISECU). These are defined as follows:

ISECU The security code can be ASCII or numeric characters, or a combination of both. The largest allowable number is 32767. When a code is not specified it defaults to zero. The security code parameter is given in the call must match the file's code when truncating, remaining, or purging the file. The security code parameter is defined as follows:

ISECU < 0. File is protected and cannot be opened without the correct security code.

ISECU = 0. File is not protected and can be read or changed by using any security code.

ISECU > 0. File is write protected but can still be read without security code. The 2's complement of a positive code, as well as the positive code, will open the file for both read and write.

ICR The ICR parameter has three meanings as shown below:

ICR > 0. Directs file to the cartridge with the positive cartridge reference number (CR). CR is a numeric identifier assigned to all cartridges in the system. A listing of the cartridge directory (CL Operator Command) would reveal a number under the heading CR. This is the cartridge label and is directly associated with a logical unit number.

ICR = 0. Directs file to all cartridges or, for CREATs, the first cartridge found with enough room. For OPEN, PURGE and NAMF calls, 0 causes the first file found with the required name to be used. Note that cartridges are searched in the order they appear in the disc directory. For more information on this order, refer to the DC Command in Section III.

ICR < 0. Directs file to a disc logical unit number (e.g., ICR = –14 means logical unit number 14).

APOSN

Purpose

This routine sets the address of the next record for any file except type 0.

```
        EXT     APOSN
         .
         .
         .
        JSB     APOSN   Subroutine call
        DEF     RTN     Return address
        DEF     IDCB
        DEF     IERR
        DEF     IREC
        DEF     IRB     Optional
        DEF     IOFF    Optional
RTN     return point    Continue execution.
         .
         .
         .
IDCB    BSS     144     144 word DCB buffer.
IERR    BSS     1       Error code returned here.
IREC    DEC     a       Record number of next record.
IRB     DEC     b       Relative block address of next record.
IOFF    DEC     c       Block offset of next record.
```

COMMENTS

- If a type 1 or 2 file is being positioned, the optional IRB and IOFF parameters are not required. Actually, for type 1 or 2 files, APOSN is not needed (see READF, WRITF) for positioning. However, APOSN will function correctly to allow sequential access of any cartridge file from the specified record.
- APOSN allows random access of sequential files. The parameters needed for APOSN are relative and are not affected by packing; they are obtained using a LOCF call.

SEQUENCE OF OPERATIONS

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; however, no file READS.
   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. Set record number.
10. Return.

CLOSE

Purpose

This routine closes the DCB and makes the file available to other callers. CLOSE also optionally truncates the file size.

```
        EXT     CLOSE
         .
         .
         .
        JSB     CLOSE       Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR
        DEF     ITRUN
RTN     return point        Continue execution.
         .
         .
         .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code returned here.
ITRUN   DEC     n           +n = number of blocks to be deleted
                                from the end of the file when
                                it is closed.

                            -n = retain main file, delete extents.

                             n = 0 = standard close.
```

COMMENTS

- The following conditions must be met for the parameter ITRUN to be recognized:
  a. The file must be opened exclusively.
  b. The current position must be in the main file (not an extent).
  c. Security codes must have matched at open.
  d. The file type must not be equal to zero.
  e. The number of blocks to be deleted must be less than or equal to the number of blocks in the file.

NOTE

If the number of blocks to be deleted equals the number of blocks in the file, then the file is purged.

SEQUENCE OF OPERATIONS

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.

CREAT

Purpose

This routine closes the DCB (if open), and then creates the named file on the specified disc with the specified number of blocks. The file is left open exclusively to the caller on successful completion of the call. This call will not create a type 0 file.

```
        EXT     CREAT
         :
        JSB     CREAT   Subroutine call
        DEF     RTN     Return address
        DEF     IDCB
        DEF     IERR
        DEF     NAME
        DEF     ISIZE
        DEF     ITYPE
        DEF     ISECU   Optional
        DEF     ICR     Optional
RTN     return point    Continue execution.
         :
IDCB    BSS     144     144 word DCB buffer.
IERR    BSS     1       Error code—or number of sectors (twice the number of blocks) if CREAT successful.
NAME    ASC     3,name  File name.
ISIZE   DEC     a       Number of blocks in the file. If negative, use all of disc (see Comments).
        DEC     b       Record length (used for type 2 file only).
ITYPE   DEC     c       File type (see Comments).
ISECU   DEC     d       Security code.
ICR     DEC     e       Cartridge label.
```

COMMENTS

- The ISIZE parameter is a two-word integer array. The first word should contain the number of blocks desired in the file. If the number is negative, the rest of the disc will be used. Unused area may be returned with the CLOSE call, ITRUN parameter. The second word will be stored in the directory as the record size. It is used as the record size only if you are creating a type 2 file.

- Since all type 2 files are addressed by record number, the maximum number of records allowed in a type 2 file is limited to 32767. That is:

$$\frac{\text{number of blocks} * 128}{\text{record size}} \quad \text{must be} < 32768$$

- The ITYPE parameter refers to the file type and is defined as follows:

  1 – 128 word record length, random access.

  2 – user selected record length, random access.

  3 – (and greater) random record length, sequential access.

  4 – source program.

  5 – relocatable program.

  6 – RTE load module.

  7 – absolute program.

  > 7 – user defined.

Note that the FMP will recognize only the above; however, the user may define any number of additional types. The FMP will treat these as type 3.

- If type 3 or greater, an EOF mark is written at the beginning of the file.

SEQUENCE OF OPERATIONS

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 CREAT 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.
11. If file type ≥3, set "written-on-flag" and EOF in DCB.
12. Return.

FCONT

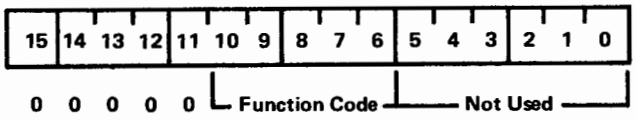
Purpose

This routine sends the standard RTE I/O Control request to type 0 (non-disc) files. It has no effect on other files.

```
                EXT     FCONT
                 :
                JSB     FCONT   Subroutine call
                DEF     RTN     Return address
                DEF     IDCB
                DEF     IERR
                DEF     ICON1
                DEF     ICON2   Optional
RTN             return point    Continue execution.
                 :
IDCB            BSS     144     144 word DCB buffer.
IERR            BSS     1       Error code or file type returned here.
ICON1           OCT     conwd   See Control Word.
ICON2           DEC     n       Required for some functions (see Control Word).
```

CONTROL WORD

- The control word value (*conwd*) has one field that is ORed with the device logical unit number.



Function Code (Octal)	Action
00	Unused
01	Write end-of-file (mag tape)
02	Backspace one record (mag tape)
03	Forward space one record (mag tape)
04	Rewind (mag tape)
05	Rewind standby (mag tape)
06	Dynamic status (mag tape)
07	Set end-of-paper tape
10	Generate paper tape leader
11	List output line spacing
12	Write gap (mag tape)
13	Forward space file (mag tape)
14	Backward space file (mag tape)

- The following functions are defined for DVR00, DVR01 and DVR02:

  20 Enable terminal–allows terminal to schedule its program when any key is struck.

  21 Disable terminal–inhibits scheduling of terminal's program.

  22 Set time out–the optional third parameter is set as the new time out interval.

  23 Ignore all further action requests until:

  a) The Queue is empty

  b) An input request is received

  c) A restore control request is received.

  24 Restore output processing (this request is usually not needed).

- Function Code 11_8 (list output line spacing), requires the optional parameter ICON2. ICON2 must designate the number of lines to be spaced on the specified logical unit. A negative parameter specifies a page eject on a line printer. For details of line printer formatting consult Appendix F in the RTE Manual.

SEQUENCE OF OPERATION

1. If DCB is not open, reject call.
2. Check if file type = 0.
3. Issue control EXEC call.
4. Return.

FSTAT

Purpose

This routine returns information on all cartridge labels in the system.

```
        EXT     FSTAT
         .
         .
         .
        JSB     FSTAT   Subroutine call
        DEF     RTN     Return address
        DEF     ISTAT
RTN     return point    Continue execution.
         .
         .
         .
ISTAT   BSS     125     Buffer of 125 words.
```

COMMENTS

- The cartridge status is contained in four word increments.

Word	Meaning
0	Logical unit number (first disc).
1	Last track for FMP.
2	Cartridge label.
3	Locking program's ID segment address, or 0 if not locked.
4	Logical unit number (second disc).
⋮	
124	0

NOTE

This list is terminated with a zero.

LOCF

Purpose

This routine formats and returns location and status information from the DCB.

```
        EXT     LOCF
                :
        JSB     LOCF    Subroutine call
        DEF     RTN     Return address
        DEF     IDCB
        DEF     IERR
        DEF     IREC
        DEF     IRB     Optional
        DEF     IOFF    Optional
        DEF     JSEC    Optional
        DEF     JLU     Optional
        DEF     JTY     Optional
        DEF     JREC    Optional
RTN     return point    Continue execution.
                :
IDCB    BSS     144     144 word DCB buffer.
IERR    BSS     1       Error code returned here.
IREC    BSS     1       Next record number.
IRB     BSS     1       Relative block of next read.
IOFF    BSS     1       Block offset of next record.
JSEC    BSS     1       Number of sectors in the file.   } See Comments
JLU     BSS     1       File logical unit.
JTY     BSS     1       File type.
JREC    BSS     1       Record size.
```

COMMENTS

- A further elaboration on the parameters follows:

IREC is the number of the next record.

IRB is the relative block of the next record (same as IREC for type 1 files). IRB is not set for a type 0 file.

NOTE

IRB includes extend information (i.e., IRB = JSEC/2*Extent number plus relative block in current extent).

IOFF is the block offset of the next record (0 for type 1 files). IRB and IOFF need not be coded for type 1 files, or for the LOCF call if the information is not required. For files other than type 1 and 0 (IOFF not set for type 0 file), IOFF will be constrained as follows:

$$0 \leqslant \text{IOFF} < 128$$

JSEC is the number of **sectors** in the main file. JSEC is not set for type 0 file.

JLU is the logical unit the file is on (disc or non-disc).

JTY is the file type as set in DCB (will be 1 if file was forced to type 1 at open).

JREC is the record size word. This will be as it is in the directory entry except for type 1 (actual or forced) files, in which case it is set to 128 on the first read/write access. If file is type 0, then JREC will be the read/write code.
Bit 15 = 1 implies read legal; bit 0 = 1 implies write legal.

SEQUENCE OF OPERATIONS

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.

NAMF

Purpose

This routine closes the DCB (if open) and then renames the specified file.

```
        EXT     NAMF
         .
         .
         .
        JSB     NAMF        Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR
        DEF     NAME
        DEF     NNAME
        DEF     ISECU       Optional
        DEF     ICR         Optional
RTN     return point        Continue execution.
         .
         .
         .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code returned here.
NAME    ASC     3,name      File's present name.
NNAME   ASC     3,nuname    File's new name.
ISECU   DEC     a           Security code.
ICR     DEC     b           Cartridge label.
```

COMMENTS

- The file specified by *name* must not be open when this call is made. If *name* is a non-zero security, the proper security code must be included. If ICR is furnished, only that cartridge is searched for *name*; otherwise, all mounted cartridges are searched. In the case where all cartridges are searched, only the first file found with the given *name* will be changed.
- *nuname* is the new name of the same file.

SEQUENCE OF OPERATIONS

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.

OPEN

Purpose

This routine closes the DCB (if open), and then opens the named file.

```
        EXT     OPEN
         .
         .
         .
        JSB     OPEN        Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR
        DEF     NAME
        DEF     IOPTN       Optional
        DEF     ISECU       Optional
        DEF     ICR         Optional
RTN     return point        Continue execution.
         .
         .
         .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code, or file type if OPEN successful.
NAME    ASC     3,name      File name.
IOPTN   OCT     a           Open options (see Comments).
ISECU   DEC     b           Security code.
ICR     DEC     c           Cartridge label.
```

COMMENTS

- The IOPTN (open option) parameter is defined as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	X	A	K	V	M	0	0	F	T	U	E

Where

E = 0 for exclusive open.

E = 1 for non-exclusive open.

U = 0 for non-update open.

U = 1 for update open.

T = 0 means file type defined in directory.

T = 1 means force to file type 1 (extents are not addressable).

F = 0 uses function code defined in directory.

F = 1 uses function code defined in bits 6-10 of this word for type 0 file only.

The following codes are used only if F = 1 and the file type is 0.

M = 0 for ASCII.

M = 1 for binary.

V = 1, and M = 1, causes the length of punched tape input to be determined by the word count in the first non-zero character read from the tape.

V = 0, and M = 1, the length of the punched tape input is determined by the buffer length specified in the call.

M determines the mode of the data transfer (if applicable).

K = 1 causes keyboard input to be printed as received. If K = 0 input from the keyboard is not printed.

A = 1 designates punching ASCII characters on the teleprinter (M = 0). ASCII is usually printed; but since it is sometimes desirable to punch ASCII tapes, this option is provided. (If A = 0, M determines mode of transfer.)

X = When paper tape devices are used, "X," in combination with "M" and "V" will indicate an honesty mode.

On input, if "X," "M," and "V" are set, absolute binary tape format is expected and handled. If "X" and "M" are set, and "V" is not, leader is not skipped and the specified number of words are read. On output, the record terminator (usually four feed frames) is not punched.

On input, if "X" is set and "M" is not, ASCII tape format is expected. Leader is not skipped, bit 8 is stripped, but otherwise, all characters are passed to the user's buffer. The only exception is line-feed, which terminates the record. On output, carriage return and line-feed are suppressed; any trailing left arrow is not (i.e., left arrow is transmitted but carriage return/line feed is not).

- UPDATE OPEN. Update implies that a block is read before it is modified so that existing records within the block will not be destroyed. Type 1 files are inherently update files. In type 1 and 2 files the end-of-file is the last word of the last block in the file. However, in other type files it is a special word (-1) for the length of a record. In order to preserve data in a type 2 file, it should be opened for update whenever any non-sequential accesses are to be performed, and the file is to be modified. That is to say, a type 2 file should be modified in non-update 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/non-update has no effect on reading or positioning.

  In type 3 and above files, an end-of-file mark is written after each record that is written in the non-update mode. In either mode (update/non-update), records written beyond an end-of-file are written in non-update fashion. That is, replace the end-of-file with the new record, followed by an end-of-file.

- EXCLUSIVE OPEN. In order to prevent two or more programs from destructively interfering with each other, a file may be opened either exclusively or non-exclusively. An exclusive open is granted only to one program at a time. If the call is rejected because another program has the file open, you must make the call again; it is not stacked by the FMP.

- NON-EXCLUSIVE OPEN. 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 a file, all programs currently having that file open will be checked. If a program is dormant (i.e., its point of suspension is zero), the file will be closed to that program. This type of close will not post the packing buffer nor will it close the DCB. This type of close is required in case of a program being aborted or otherwise terminating without closing a 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.)

SEQUENCE OF OPERATIONS

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.

POSNT

Purpose

This routine causes the next read or write to access the given record in any file type.

```
        EXT     POSTN
        .
        .
        .
        JSB     POSNT       Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR
        DEF     NUR
        DEF     IR          Optional
RTN     return point        Continue execution.
        .
        .
        .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code returned here.
NUR     DEC     a           New record number.
IR      DEC     b           Absolute vs. relative control parameter for NUR.
```

COMMENTS

- The NUR (new record) parameter is defined as follows:

  NUR > 0 Forward positioning, skip NUR records.

  NUR < 0 Backward positioning, skip NUR records.

  NUR = 0 No operation.

  NOTE

  If a type 0 file is involved the motion requested must be legal for the device.

- The IR (absolute vs relative) parameter is defined as follows:

  IR = 0 or nor present — The new record indicated by NUR is relative to the present position in the file.

  IR = 0 — The new record indicated by NUR is the absolute number starting from the first record in the file (record number 1). Note that NUR must be > 0.

- Forward positioning on a type 0 file is accomplished by reading records until one less than the desired record is read, or an EOF is read. In all cases, EOF terminates positioning.

- Backspacing on type 0 files can be accomplished three ways. Briefly, if the first record backspaced over is an EOF, and the only EOF, no error occurs. If an EOF is encountered anywhere else (i.e., as a subsequent record), a -12 error will occur, and the call will be terminated after forward spacing to position the file immediately following the EOF.

  Examples (refer to Figure 2-1):

  1. File position is immediately after EOF2 and NUR indicates backspace four records. Final file ends up immediately after EOF1 with no error.

  2. File position is at 2A and NUR indicates backspace five records. Error -12 results, the file position ends up immediately after EOF2, and the call is terminated.

3. File position is immediately after EOF2 and NUR indicates backspace five records. Error -12 results, the file position ends up immediately after EOF1, and the call is terminated.

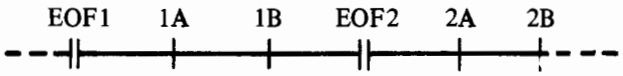


Figure 2-1. Record Positioning Example Using NUR in POSNT

- Type 3 and above files are treated as mag tape files. To be correct, a backspace should be issued after an EOF is read, and before continuing to write. This action simulates a magnetic tape, that is, the EOF is spaced over. It is also legal to continue without backspacing.

SEQUENCE OF OPERATIONS

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.

PURGE

Purpose

This routine closes the DCB (if open), and then deletes the named file and all of its extents.

```
        EXT     PURGE
                .
                .
                .
        JSB     PURGE       Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR
        DEF     NAME
        DEF     ISECU       Optional
        DEF     ICR         Optional
RTN     return point        Continue execution.
                .
                .
                .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code returned here.
NAME    ASC     3,name      File name.
ISECU   DEC     a           Security code.
ICR     DEC     b           Cartridge label.
```

COMMENTS

- The file must not be open to any other program when this call is made.
- This routine will not purge type 0 (non-disc) files.
- If the file being purged is the last file on the disc (excluding a type 6 file), all area from the last unpurged or type 6 file up to and including the purged file, is returned to the system. If a purged file is not the last file, then its area may be recovered only by packing, or by purging all files after it in the directory. Type 6 file area may only be recovered by packing.

SEQUENCE OF OPERATIONS

1. Check for enough parameters.
2. Call OPEN to open the file exclusively.
3. If OPEN errors, exit
4. Check security flag in DCB; if mismatch, close and exit.
5. Close and truncate file to zero length.
6. Return.

READF

Purpose

This routine reads a record from the file currently open to the DCB, to the user buffer.

```
        EXT     READF
         :
        JSB     READF   Subroutine call
        DEF     RTN     Return address
        DEF     IDCB
        DEF     IERR
        DEF     IBUF
        DEF     IL      Optional
        DEF     LEN     Optional
        DEF     NUM     Optional
RTN     return point    Continue execution.
         :
IDCB    BSS     144     144 word DCB buffer.
IERR    BSS     1       Error code returned here.
IBUF    BSS     IL      Data buffer.
IL      DEC     a       Read request length (see Comments).
LEN     BSS     b       Actual read length returned here (see Comments).
NUM     DEC     c       Record number to be read (see Comments).
```

COMMENTS

- The IL (request length) parameter takes on different meanings depending on the file-type. Refer to Table 2-2.
- The LEN (actual length) parameter provides the actual number of words transferred to IBUF. LEN can never exceed IL; therefore, if LEN = IL the record may have been too long (unless a type 1 file is being read). If LEN = 1, then an end-of-file has been read.
- NUM is the random access record number and is used only when the file type is 1 or 2. Record number 1 is the first record in a file. If NUM = 0 or is absent, then the transfer starts at the current record number. If NUM is positive, then the transfer starts at the record number indicated by NUM. If NUM is negative, then the transfer starts at the current record plus NUM. Therefore:

File Type	Meaning
0	Not used.
1	NUM defines the first record transferred, and IL defines the number of records through the length in words.
2	NUM defines the record to be transferred.
3 and above	Not used.

Examples (refer to Figure 2-2):

1. If NUM = 0, transfer starts at record number 5.
2. If NUM = 6, transfer starts at record number 6 (same as NUM).
3. If NUM = -3, transfer starts at record number 2 (5 + (-3) = 2).

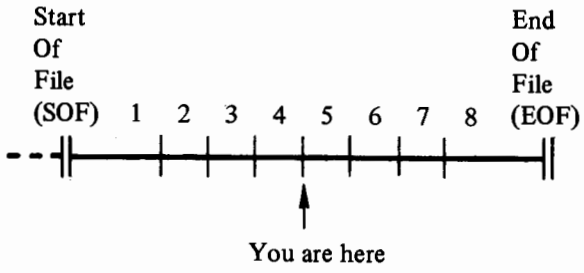


Figure 2-2. Record Positioning Example Using NUM in READF

Table 2-2. Read Request File Length (IL) vs. File Types

Request Length / File Type	IL > 0	IL = 0 (Indicates zero length record) (Not recommended)	IL not supplied
0	The routine will transfer up to IL words. If the record length is smaller than IL, only the record is transferred. If the record is the same size, smaller, or larger than IL, no error is indicated.	A zero length record is passed to the device; usually causes a record to be skipped and the record is counted.	A zero length record is passed to the device; usually causes a record to be skipped and the record is counted. (Not recommended)
1	The routine reads exactly IL words. Therefore, since the record length is fixed at 128 words, less than a full record or more than one record may be transferred. All of IL, however, must be within the file.	A zero length read from the disc is done, which results in no position change; the record is not counted.	128-word record is assumed.
> 1	The routine will transfer up to IL words. If the record length is smaller than IL, only the record is transferred. If the record is the same size, smaller or larger than IL, no error is indicated.	Causes a record to be skipped and the record is counted.	Actual record length is used.

- There are three levels of EOF associated with file types that are defined below.

File Type	Meaning
0	When an EOF is read, parameter LEN is set to -1 with no error. Accesses may continue beyond the EOF.
1 and 2	When an EOF is read, parameter IERR is set to -12 indicating an error condition. No accesses beyond the EOF.
3 and above	The first EOF read causes parameter LEN to be set to -1 with no error. A read access beyond this EOF will cause an error condition (IERR = -12). A write access may continue beyond this EOF with no error.

SEQUENCE OF OPERATIONS

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 block if it was written on.
7. If type 2 or above, and reading flag set, and DCB buffer is empty, then read a block 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.
18. If type 2, go to 11.
19. If type 0, go to 30.
20. If type 1, round up length to even 128 words and save as increment in record count.
21. Check if request is within the file.
22. If track switch, compute maximum words this accesss.
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.

RWNDF

Purpose

This routine rewinds type 0 files, and sets disc files so that the next record is the first record in the file.

```
        EXT     RWNDF
        .
        .
        .
        JSB     RWNDF       Subroutine call
        DEF     RTN         Return address
        DEF     IDCB
        DEF     IERR        Optional
RTN     return point        Continue execution.
        .
        .
        .
IDCB    BSS     144         144 word DCB buffer.
IERR    BSS     1           Error code returned here.
```

SEQUENCE OF OPERATIONS

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.

WRITF

Purpose

This routine writes a record on the file currently open to the DCB.

```
        EXT     WRITF
         .
         .
         .
        JSB     WRITF   Subroutine call
        DEF     RTN     Return address
        DEF     IDCB
        DEF     IERR
        DEF     IBUF
        DEF     IL
        DEF     NUM
RTN     return point    Continue execution.
         .
         .
         .
IDCB    BSS     144     144 word DCB buffer.
IERR    BSS     1       Error code returned here.
IBUF    BSS     IL      Data buffer.
IL      DEC     a       Write request length (see Comments).
NUM     DEC     b       Record number to be written (see Comments).
```

COMMENTS

- The IL (write request length) parameter takes on different meanings depending on the file type. Refer to Table 2-3.

- NUM is the random access record number and is used only when the file type is 1 or 2. Record number 1 is the first record in a file.) If NUM = 0 or is absent, then the transfer starts at the current record number. If NUM is positive, then the transfer starts at the record number indicated by NUM. If NUM is negative, then the transfer starts at the current record plus NUM. Therefore:

File Type	Action
0	Not used.
1	NUM defines the first record to be written, and IL defines the number of records through the length in words.
2	NUM defines the record to be transferred.
3 and above	Not used.

Examples (refer to Figure 2-3):

1. If NUM = 0, transfer starts at record number 5.
2. If NUM = 6, transfer starts at record number 6 (same as NUM).
3. If NUM = -3, transfer starts at record number 2 (5 + (-3) = 2).

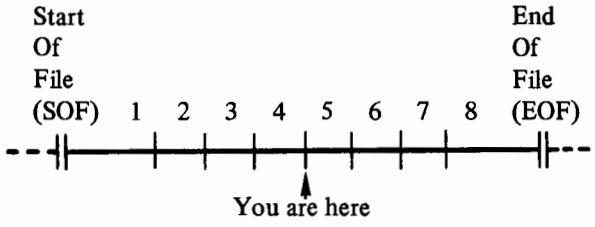


Figure 2-3. Record Positioning Example Using NUM in WRITF

Table 2-3. Write Request File Length (IL) vs. File Type

Request Length / File Type	IL > 0	IL = 0	IL = -1	IL < -1 (Not (recommended)	IL not supplied
0	The routine will write exactly IL words.	A zero length record is written.	An end-of-file is written.	The length (IL) is interpreted as a character count.	A zero length record is written.
1	Data is written in 128 word blocks. For example, IL is rounded up to 128 (1 block) if IL is between 1 and 127. IL is rounded up to 256 (2 blocks) if IL = 129 to 255.	No action.	No action.	No action.	128 is used.
2	IL is ignored and the file defined record length is used.	IL is ignored and the file defined record length is used.	No action.	No action.	The file defined record length is used.
> 2	The routine will write exactly IL words.	A zero length record is written.	An end-of-file is written.	Undefined.	A zero length record is written.

SEQUENCE OF OPERATIONS

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 DCB length to 128.
7. If type 1 or 2 and EOF write, then exit, else 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 a block 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 go 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.
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.

EXAMPLE PROGRAM

The following listings show a sample program (DEMO1) and subroutine (IERR). The program is designed to sort a file and place the results in another file using most of the calls described in this section. Subroutine IERR is called by the program to check if any errors were detected.

Operation is simple and straightforward. Most of the instructions are "commented" on the first page of the program. Proceed as follows:

a. Generate the code on tape or cards, or enter the code directly into a file with the Store Command.

```
:ST, 1, DEMO1S:::4:24
  :
```

Use Control D for end-of-file.

b. Submit a file to be sorted and call it USERF*x* where *x* is a number from 0 - 9.

:ST, *nam1*, USERF*x*

c. Create a file that is the target file for the sorted data, and is called USERF*y* where *y* is a number from 0 - 9. One suggestion is to create a type 0 file referencing the line printer (list device = LUG).

:CR, USERF*y*, 6, WR

d. Move the program into the RTE system and compile it with the FORTRAN Compiler. Once its compiled save the binary code in a file for future use.

e. Put the program into the system with the RTE loader.

f. Turn the program on with the required parameters.

*ON, DEMO1, *x*, *y*, *z*

where

USER*x* is the file to be sorted (only *x* is required).

USER*y* is the target file (only *y* is required).

z is the key word to be sorted on. For example, if the file to be sorted consisted of a group of names, *z* could be 1 indicating sort on the first two letters of the name.

```
DEMO1S T=00004 IS ON CR00100 USING 00024 BLKS R=0000

0001  FTN,L
0002        PROGRAM DEMO1(3,90)
0003  C
0004  C           THE PURPOSE OF THIS PROGRAM IS TO SORT A FILE CALLED
0005  C           USERFX AND PLACE THE RESULTING DATA IN FILE USERFY.
0006  C
0007  C           THE SORT WILL BE KEYED ON WORD POSITION Z.
0008  C
0009  C           X,Y AND Z ARE THE THREE TURN ON PARAMETERS.
0010  C
0011  C           A THIRD FILE (USERFA) IS CREATED BY THIS PROGRAM AND
0012  C           THEN PURGED AT TERMINATION.  THIS FILE WILL BE
0013  C           TYPE 2 WITH 4 WORD RECORDS.
0014  C
0015  C           METHOD OF APPROACH:
0016  C
0017  C           1. USERFX WILL BE READ AND THE ADDRESS OF EACH
0018  C              OF ITS RECORDS ALONG WITH THE KEY WORD (WORD Z)
0019  C              WILL BE SAVED IN USERFA.
0020  C
0021  C           2. USERFA WILL BE SORTED ON THE KEY WORD USING A
0022  C              "BUBBLE SORT".
0023  C
0024  C           3. THE ADDRESSES SAVED IN USERFA WILL THEN BE READ
0025  C              SEQUENTIALLY AND USED TO ADDRESS THE MATCHING
0026  C              RECORD IN USERFX.  THIS RECORD WILL THEN
0027  C              BE READ AND WRITTEN ON USERFY.
0028  C
0029  C
0030        DIMENSION IUSRX(144),IUSRY(144),IUSRA(144),IBUF(128),NUSR(4)
0031        DIMENSION IADRS(4)
0032  C
0033  C           SET UP THE FILE NAME ARRAYS
0034  C
0035        DATA NUSR/2HUS,2HER,0/,IR/43060B/,IUA/2HFA/
0036  C
0037  C           GET THE PRAMETERS AND SET UP DEFAULTS AS FOLLOWS:
0038  C
0039  C           USERFX DEFAULT IS USERF0
0040  C           USERFY DEFAULT IS USERF1
0041  C           Z      DEFAULT IS 2
0042  C
0043        CALL RMPAR(IUSRX)
0044        IUX=IR+IUSRX(1)
0045        IUY=IR+IUSRX(2)
0046        IWORD=IUSRX(3)
0047        IF(IUY-IR)20,10,20
0048   10   IUY=IUY+1
0049   20   IF(IWORD)40,30,40
0050   30   IWORD=2
0051  C
0052  C           OPEN THE USER FILES AND CREAT THE TEMP FILE.
0053  C
0054   40   NUSR(3)=IUX
0055   50   CALL OPEN(IUSRX,IER,NUSR,1)
0056  C
0057  C           NON-EXCLUSIVE OPEN FOR X SO OTHERS CAN READ IT TOO.
0058  C
```

```
0059          IF(IERR(IER,IUX))50,60
0060    C
0061    C           NOW AN EXCLUSIVE OPEN OF Y
0062    C
0063     60   NUSR(3)=IUY
0064     70   CALL OPEN(IUSRY,IER,NUSR)
0065    C
0066    C           CHECK FOR ERRORS
0067    C
0068          IF(IERR(IER,IUY))70,75
0069    C
0070    C           SET UP SIZE ARRAY FOR CREAT 100 BLOCKS,4 WORD RECORDS
0071    C
0072     75   IBUF(1)=100
0073          IBUF(2)=4
0074    C
0075    C           CREAT USERFA
0076    C
0077          NUSR(3)=IUA
0078     80   CALL CREAT(IUSRA,IER,NUSR,IBUF,2)
0079          IF(IERR(IER,IUA))80,90
0080    C
0081    C           ALL FILES ARE NOW OPEN
0082    C
0083    C           PHASE ONE BEGIN.
0084    C
0085    C           SET NUMBER OF RECORDS IN FILE TO 0
0086    C
0087     90   NOREC=0
0088    C
0089    C           FOURCE SHORT RECORDS TO TOP (I.E. DEFAULT IS A ZERO)
0090    C
0091    100   IBUF(IWORD)=0
0092    C
0093    C           GET CURRENT POSITION
0094    C
0095          CALL LOCF(IUSRX,IER,IADRS(2),IADRS(3),IADRS(4))
0096          CALL IERR(IER,IUX)
0097    C
0098    C           READ THE RECORD AND SET THE KEY
0099    C
0100          CALL READF(IUSRX,IER,IBUF,IWORD,L)
0101          CALL IERR(IER,IUX)
0102          IADRS(1)=IBUF(IWORD)
0103    C
0104    C           IF EOF THEN GO TO PHASE TWO.
0105    C
0106          IF(L)200,120
0107    C
0108    C           WRITE USERFA ENTRY
0109    C
0110    120   CALL WRITF(IUSRA,IER,IADRS)
0111          CALL IERR(IER,IUA)
0112    C
0113    C           STEP THE RECORD COUNT AND GO READ THE NEXT RECORD
0114    C
0115          NOREC=NOREC+1
0116          GO TO 100
0117    C
0118    C
```

```
0119  C           BEGIN PHASE TWO - BUBBLE SORT OF USERFA ON
0120  C           THE FIRST WORD (I.E. THE KEY WORD)
0121  C
0122  C           FIRST RE OPEN USERFA AND SET FOR UPDATE.
0123  C
0124   200  CALL OPEN(IUSRA,IER,NUSR,2)
0125        CALL IERR(IER,IUA)
0126  C
0127  C           SORT INITIALIZE.  IBUF WILL CONTAIN
0128  C           THE TWO RECORDS TO BE TESTED.
0129  C           ICUR WILL POINT AT THE CURRENT WINNER AND
0130  C           ITEST WILL POINT AT THE TEST RECORD.
0131  C
0132        ICUR=1
0133        ITEST=5
0134  C
0135  C           ISWCO WILL BE SET NON ZERO WHENEVER A RECORD IS SWAPPED.
0136  C
0137  C           START THE SORT
0138  C
0139        DO 295 K=2,NOREC
0140  C
0141  C           LAST WILL CONTAIN THE NUMBER OF THE LAST RECORD FOR
0142  C           THE CURRENT PASS.
0143  C
0144        LAST=NOREC-K+2
0145        ISWCO=0
0146  C
0147  C          READ THE CURRENT RECORD FOR THIS PASS
0148  C
0149        CALL READF(IUSRA,IER,IBUF(ICUR),4,L,K-1)
0150        CALL IERR(IER,IUA)
0151  C
0152  C           START SINK SECTION
0153  C
0154   205  DO 250 I=K,LAST
0155  C
0156  C           READ THE CURRENT TEST RECORD
0157  C
0158        CALL READF(IUSRA,IER,IBUF(ITEST),4,L,I)
0159        CALL IERR(IER,IUA)
0160  C
0161  C           TEST THE FIRST WORD AGAINST CURRENT WINNER.
0162  C
0163        IF(IBUF(ITEST)-IBUF(ICUR))225,210,210
0164  C
0165  C           ORDER O-K SET NEW WINNER.
0166  C
0167   210  J=ITEST
0168        ITEST=ICUR
0169        ICUR=J
0170        GO TO 250
0171  C
0172  C           OUT OF ORDER - SWAP THE RECORDS
0173  C           TEST TO CURRENT RECORD LESS ONE , CURRENT WINNER TO CURRENT
0174  C           RECORD POSITION
0175  C
0176   225  CALL WRITF(IUSRA,IER,IBUF(ITEST),4,I-1)
0177        CALL IERR(IER,IUA)
0178  C
```

```
0179  C          NOW THE CURRENT.  NOTE WE CAN USE A SEQUENTIAL WRITE HERE
0180  C
0181        CALL WRITF(IUSRA,IER,IBUF(ICUR))
0182        CALL IERR(IER,IUA)
0183  C
0184  C          SET THE SWAPPED FLAG
0185  C
0186        ISWCO=-1
0187   250  CONTINUE
0188  C
0189  C          END OF SINK SECTION
0190  C
0191  C          IS FILE ORDERED??
0192  C
0193        IF(ISWCO)255,300
0194  C
0195  C          NO.  NOW DO THE FLOAT SECTION
0196  C
0197   255  ISWCO=0
0198        DO 290 I=K,LAST
0199  C
0200  C          FLOAT SECTION GOES UP THE LIST SO IREC MUST BE COMPUTED
0201  C
0202        IREC=LAST-I+K-1
0203  C
0204  C          IREC IS THE CURRENT RECORD ADDRESS
0205  C          READ A RECORD
0206  C
0207        CALL READF(IUSRA,IER,IBUF(ITEST),4,L,IREC)
0208        CALL IERR(IER,IUA)
0209  C
0210  C          ITEST THE RECORD AGAINST THE CURRENT WINNER
0211  C
0212        IF(IBUF(ITEST)-IBUF(ICUR))260,260,280
0213  C
0214  C          ORDERED SO SET NEW WINNER.
0215  C
0216   260  J=ITEST
0217        ITEST=ICUR
0218        ICUR=J
0219        GO TO 290
0220  C
0221  C          NOT ORDERED SO SWAP THE RECORDS
0222  C
0223   280  CALL WRITF(IUSRA,IER,IBUF(ICUR),4,IREC)
0224        CALL IERR(IER,IUA)
0225  C
0226  C          HERE AGAIN BY CHOSING THE ORDER WE CAN USE A SEQUENTIAL
0227  C          WRITE
0228  C
0229        CALL WRITF(IUSRA,IER,IBUF(ITEST))
0230        CALL IERR(IER,IUA)
0231  C
0232  C          SET THE SWAPPED FLAG.
0233  C
0234        ISWCO=-1
0235   290  CONTINUE
0236  C
0237  C          END OF FLOAT SECTION.  WERE ANY SWAPPS MADE??
0238  C
```

```
0239          IF(ISWCO)295,300
0240    C
0241    C           YES SO CONTINUE THE SORT
0242    C
0243     295  CONTINUE
0244    C
0245    C           END OF SORT
0246    C
0247    C           TRANSFER THE FILE IN SORTED ORDER
0248    C
0249     300  DO 390 I=1,NOREC
0250    C
0251    C       READ THE ADDRESS
0252    C
0253          CALL READF(IUSRA,IER,IADRS,4,L,I)
0254          CALL IERR(IER,IUA)
0255    C
0256    C           SET FILE USERFX TO THE ADDRESS.
0257    C
0258          CALL APOSN(IUSRX,IER,IADRS(2),IADRS(3),IADRS(4))
0259          CALL IERR(IER,IUX)
0260    C
0261    C           READ THE RECORD
0262    C
0263          CALL READF(IUSRX,IER,IBUF,128,L)
0264          CALL IERR(IER,IUX)
0265    C
0266    C          WRITE THE RECORD ON USERFY
0267    C
0268          CALL WRITF(IUSRY,IER,IBUF,L)
0269          CALL IERR(IER,IUY)
0270     390  CONTINUE
0271    C
0272    C           ALL OF THE FILE IS TRANSFERED SO WRITE AN EOF AND
0273    C           CLOSE ALL THE FILES.
0274    C
0275          CALL WRITF(IUSRY,IER,IBUF,-1)
0276          CALL IERR(IER,IUY)
0277    C
0278    C           PURGE USERA USING THE TRUNCATE OPTION
0279    C
0280          CALL CLOSE(IUSRA,IER,100)
0281          CALL IERR(IER,IUA)
0282    C
0283    C           CLOSE IUSERX AND IUSERA
0284    C
0285          CALL CLOSE (IUSRX,IER)
0286          CALL IERR(IER,IUX)
0287          CALL CLOSE(IUSRY,IER)
0288          CALL IERR(IER,IUY)
0289    C
0290    C           END OF PROGRAM   TERMINATE.
0291    C
0292          END
0293          FUNCTION IERR(ICOD,ID)
0294           DIMENSION MESS(13)
0295    C
0296    C           THIS FUNCTION CHECKS FOR ERRORS AND IF ICOD IS <0
0297    C           THEN PRINTS IT ON THE SYSTEM TTY.  IT THEN PAUSES.
0298    C
```

```
0299  C           IF THERE IS NO ERROR IT JUST RETURNS.
0300  C
0301  C           IN ANY CASE THE ERROR CODE ICOD IS RETURNED AS THE VALUE
0302  C
0303  C
0304  C           SET UP THE MESSAGE
0305  C
0306        DATA MESS/2H  ,2HER,2HRO,2HR ,2H  ,2H O,2HN ,2HFI,2HLE,
0307       C           2H  ,2HUS,2HER,2H  /
0308  C
0309  C           SET THE RETURN CODE
0310  C
0311        IERR=ICOD
0312  C
0313  C           WAS THERE AN ERROR??
0314        IF(ICOD)10,99
0315  C
0316  C           YES SET UP THE MESSAGE.
0317  C
0318   10   I=-ICOD/10
0319        J=-ICOD-I*10
0320        MESS(5)=I*4008+2H00+J
0321        MESS(13)=ID
0322        CALL EXEC(2,1,MESS,13)
0323        PAUSE
0324   99   END
0325        END$
```

SECTION III

OPERATOR REQUESTS

SECTION III
OPERATOR REQUESTS

The operator controls the HP RTE File Manager (FMP) by operator requests entered through the system input device. These operator requests will cause the FMP to perform the functions shown in Table 3-1.

Table 3-1. FMGR Operator Commands

Command Format	Description	Page
CL	List the cartridge directory.	3-7
CO	Copy all files on one cartridge to another cartridge.	3-8
CR	Create a file–does not store data.	3-8
DC	Request FMP to logically disconnect the cartridge from the system.	3-10
DL	List the file directory.	3-11
DU	Dump a file–does not create a file.	3-13
IN	Initialize a cartridge.	3-15
LI	List file contents.	3-16
LL	Change assignment of output list device.	3-16
LO	Change assignment of log device.	3-17
MC	Notify FMP a cartridge has been mounted.	3-17
MR	Move a relocatable binary file to the load-and-go tracks.	3-18
MS	Move a source file to the logical source tracks.	3-18
PK	Pack one or all mounted cartridges.	3-19
PU	Purge a file.	3-20
RN	Rename a file.	3-20
RP	Restore a saved program.	3-20
SP	Save an RTE System program – creates a type 6 file.	3-21
SA	Create a file and save the logical source or load-and-go tracks.	3-22
ST	Store a file–creates a new file.	3-22
SV	Change error print-out severity code.	3-24
TR	Transfer input control to another file, or logical unit number.	3-24
??	Expand error message.	3-25
EX	Terminate FMGR.	3-25

COMMAND STRUCTURE

The operator gains the attention of the FMP by scheduling the software module FMGR with the ON FMGR operator request. When FMGR runs, a colon (:) prompter is returned on the initial keyboard input device. If the input device is not a keyboard, the first command is read from the device that is the input. Each command is parsed, or resolved, by a central routine that accepts certain conventions. Command syntex is described in Table 3-2 and, with the conventions described next, must be followed exactly.

CONVENTIONS

- A leading plus (+) sign in a numeric parameter is ignored. The number is assumed to be positive unless preceded by a minus (-) sign.
- A numeric parameter immediately followed by the letter 'B' indicates the parameter is octal.
- Each command from a device other than a TTY must be preceded by a colon (:). Entries from a TTY will be prompted for by a system generated colon – the operator must not enter a second colon.
- Each parameter is tested on the assumption that it is numeric. If it fails to convert it is then assumed to be ASCII. If the parameter is ASCII, the first six characters are saved. If less than six characters are entered, the parameter is padded to six characters with trailing blanks. For example, if 6 followed by a blank is entered as a parameter, the Parse routine would assume it to be numeric. If 6 blank x (a non-numeric character) is entered, the Parse routine would assume it to be ASCII.
- Two commas or colons in a row mean a parameter assumes its default value.
- Leading blanks and blanks on either side of a comma or semicolon are ignored.
- An EOF encountered on the command input file causes a TR with no parameters.
- All parameters are initially filled with zeros and flagged as not supplied, so FMGR can distinguish null (space or nothing) from zeros.

Table 3-2. Conventions in Operator Command Syntax

Item	Meaning
UPPER CASE ITALICS	These words are literals and must be specified as shown.
lower case italics	These are symbolic representations indicating what type of information is to be supplied. When used in text, the italics distinguishes them from other textual words.
REad	A combination of the above shows that the RE is a literal and is specified as shown. The remainder of the characters are for information only and may or may not be entered.
[, *item*]	Items with brackets are optional. However, if *item* is not supplied, its position must be accounted for with a comma. This causes *item* to default.
[, *item 1*] [, *item 2*] [, *item 3*]	This indicates that exactly one specified *item* may be specified.
... .	This notation means "and so on."
item 1 *item 2* *item 3*	This indicates that there is a choice of entries for the parameter, but one parameter must be specified.
DL [, *cl*[, *msc*]]	The placement of the brackets indicates that the *cl* and *msc* parameters are both optional. However, if *cl* is not supplied, its position must be accounted for with a comma. This causes *cl* to default. Example: :DL, , JB
namr	This is considered to be one parameter with up to five subparameters separated by colons (:).

namr = file name [:*security code*[:*label*[:*file type*[:*file size*[:*record size*]]]]]

– or –

namr = logical unit number

NOTE

File size is specified in blocks, and record size is required only for type 2 files.

NAMR

namr is the symbolic representation of the general file reference parameter for most of the FMGR commands. *namr* is considered to be one parameter, a file name, with up to five subparameters separated by colons (:).

namr = **file name [:security code [:label [:file type [:file size [:record size]]]]]**

– or –

namr = **logical unit number**

namr can be a logical unit number of a TTY, punch, photo reader, card reader, magnetic tape, or any such device except a disc.

FILE NAME

A legal file name is six ASCII characters in length. The first character must not be a blank or a number. All characters must come from the set b thru ← (see Appendix E) exclusive of special characters "+" (plus, "–" (minus), "," (comma), and ":" (colon). Imbedded blanks are not allowed. Only the first and second parameter of a FMGR Command can be a *namr* or have subparameters, and only the first and second subparameters may be non-numeric.

NOTE

All of the following described subparameters are initially filled with zeros prior to each command; thus, omitted parameters are taken as zeros. (Two colons in a row mean a subparameter is omitted or is defaulted to zero.)

SECURITY CODE

The security code can be ASCII or numeric characters, or a combination of both. The largest allowable number is 32767. When a code is not specified it defaults to zero. The security code parameter is defined as follows:

security code = 0 – file is not protected and can be read or changed by any user.

security code > 0 – file is write protected but can still be read without security code. If 2's complement of positive code is given, then the file will be opened for both read and write.

security code < 0 – file is protected and can not be opened without the correct security code.

LABEL

The label parameter has three meanings as shown below:

label > 0 – indicates a cartridge reference (CR) number (positive integer) that is a numeric identifier assigned to all cartridges in the system. A listing of the cartridge directory (CL Operator Command) would reveal a number under the heading CR. This is the cartridge label and is directly associated with a logical unit number.

label = 0 – indicates the first available disc which satisfies the request to be used. Note that the discs are searched in the order they appear in the disc directory.

label < 0 – indicates the logical unit number of the disc to be used (e.g., -14 indicates logical unit number 14).

NOTE

The following subparameters, FILE TYPE, FILE SIZE, and RECORD SIZE, are used only if the file being described is to be created by the command.

FILE TYPE

File types are defined as follows:

0 – non-disc file

1 – 128-word record length, random access

2 – user selected record length, random access

3 – (and greater) random record length, sequential access

4 – source program

5 – relocatable program

6 – RTE load module

7 – absolute program

> 7 – user defined.

FILE SIZE

File size is always specified in blocks, where two sectors (128 words) equal one block. If a negative file size is specified, all of the available file space is allocated to that file. When the file is finished being loaded, the unused area is returned to the File Manager (i.e., the file is truncated to the area used. This is useful when it is desired for a file to not have any extents.

RECORD SIZE

Record size is required in the command only if the file is type 2. When a file is created (except type 0 files), the record size parameter is put in the directory and may subsequently be recovered by the user interface (see the LOCF call) or by the LI Command.

Examples of how *namr* may be specified are shown below.

10	File is on $LU10_{10}$, a non-disc file.
20B	File is on $LU20_8$ ($LU16_{10}$), a non-disc file. The B suffix indicates octal.
$XYZ:SC	File name is "$XYZ" with ASCII security code "SC."
ABS: -10-3:2:40:6	File to be created will be named "ABS", have security code -10, be on LU3, type 2, 40 blocks long, and have 6-word records.
A23456 : : : 20	File to be created will be named "A23456" and be type 20; security and cartridge reference are not supplied.

LOCKING

When a cartridge is locked, only the locking program may access files or make directory changes on that cartridge. To other callers, it appears as if the cartridge is not mounted. A lock can only be obtained if all files on that cartridge are closed. A failure to obtain a lock is indicated by error -8.

A cartridge is locked whenever FMGR:

- is packing it,
- is initializing it,
- is removing it,
- is creating a type 0 file on it,
- is purging a type 0 file on it,
- mounts it and finds it has not been initialized.

If a users program attempts an access of a locked cartridge, error code -13 will be transmitted. For example, if FMGR is packing a cartridge in the background, and another program in the foreground requests that a file on that cartridge be opened; the -13 cartridge locked error will result. This is because the cartridge is locked by FMGR when performing the pack.

FMGR TURN-ON SEQUENCE

The File Manager operator interface routine, FMGR, can be turned on using the RTE System command ON, FMGR; or by internal scheduling with a user's program. However, before a brand new virgin moving head cartridge can be used on any RTE System it must be properly formatted. Formatting consists of doing format data commands on the whole cartridge and flagging any defective tracks as bad tracks. Formatting may be done by the controller diagnostic routine (HP 13041) or by the moving head RTE Generator (HP 29014). If the cartridge has been previously formatted, the following procedure is not necessary – go on to the ON, FMGR Command.

VIRGIN CARTRIDGE INITIALIZATION

During initial moving head system generation, the operator is given the option of formatting all sub-channels. The following procedure may be used to format new packs, or packs that may not have been mounted during the initial generation. Refer to the RTE manual, Section VI, parts 1 and 3 for more details on cartridge formatting.

1. Load the moving head RTGEN (HP 29014) using the BBL.
2. Load and configure a TTY SIO driver.
3. Start RTGEN at octal address 100; clear the switch register and push RUN.
4. Answer the cartridge channel question as in a normal generation.
5. a. Define the areas on the respective subchannels (number of tracks and starting track number) which are to be initialized.

   b. Define a subchannel higher than any defined in "a" that is not mounted (i.e., either the unit is not present or it is unloaded). For example; if the system has one drive (i.e., subchannels 0 and

1), then define some tracks on subchannel 2. Then enter the /E.

CAUTION

Do not define any cartridge areas in "a" that have existing systems or data that is to be preserved on them. Steps 6 through 9 are important to prevent RTGEN from automatically formatting a cartridge area containing data.

6. Assign the dummy subchannel as the system subchannel.

7. Do not assign an "AUX DISC".

8. Assign the dummy subchannel as the scratch data.

9. Set scratch origin to 0.

10. The number of 128 word sectors per track is 48 for the HP 7900/7901 and 24 for the HP 2870.

11. Answer the rest of the questions as per a normal generation.

12. When RTGEN asks:

    INITIALIZE SUBCHNL:
    X?

    answer YES for those channels to be initialized.

13. After the last subchannel is initialized, RTGEN will try to initialize the dummy subchannel (because it is the system subchannel) and will find it down. This causes message:

    READY DISC AND PRESS RUN

At this time the cartridges are formatted and the procedure is complete.

ON, FMGR

Purpose

To schedule the background program FMGR that allows communication between the FMP and operator.

Format

*ON, FMGR[, *input*[, *log*[, *list*[, *severity code*]]]]

Where

input is the logical unit number of the input device. Default is LU1 (system teleprinter).

log is the logical unit number of a two-way device for logging an error detected on a command from the input device. If an entry is not given then *log* assumes the logical unit number given for *input*, only if *input* is a TTY; otherwise, *log* assumes logical unit number 1.

list is the logical unit number of the list device. Default is LU6 (standard list device).

severity code is the error message severity code. Default is 0. The options are:

0 – All commands are echoed on the log device. Any error causes an appropriate error number to be printed.

1 – Inhibit command echo on log device.

2 – Inhibit error messages to log device, unless the error is severe enough to cause control to transfer to the log device for a command input.

FMGR SCHEDULE

Purpose

To internally schedule the File Manager operator interface routine FMGR.

```
        EXT     EXEC
                .
                .
                .
SCHED   JSB     EXEC        Transfer control to RTE
        DEF     RTN         Return point
        DEF     ICODE       Request code
        DEF     FMGR        Name of program to schedule
        DEF     P1          }
                .           }
                .           }  Up to five optional parameters
                .           }
        DEF     P5          }
RTN     SZA                 If not scheduled Loop.
        JMP     SCHED
                .
                .
                .
ICODE   DEC     9 or 10     9 = schedule with wait, 10 = no wait
FMGR    ASC     3,FMGR      Name of program to schedule
P1      }
        }                   Up to five optional parameters; see Comments
P5      }
```

COMMENTS

- The optional parameters are defined in the ON,FMGR Command on the previous page. If desired to turn on FMGR to read its commands from a file name, P1 through P5 may be defined as:

P1	ASC 1, FN	File name
P2	ASC 1, AM	
P3	ASC 1, E	
P4	DEC x	Severity code
P5	DEC y	List logical unit number

FNAME can be any ASCII file accessible with a cartridge ID of 0, and a security code which allows reading (i.e., + or 0). The log unit in the above call is defaulted to LU1.

- If FMGR is dormant, it is scheduled and a zero is returned to the calling program in the A-Register.

- If FMGR is not dormant, it is not scheduled by this call, and its status (which is some non-zero value) is returned to the calling program in the A-Register.

WAITING AND NO WAITING

- A schedule with waiting (ICODE = 9) causes RTE to put the calling program in waiting status. The called program runs at its own priority, which may be greater than, less than, or equal to that of the calling program. Only when the called program terminates does RTE resume execution of the original program at the point immediately following the schedule call.

- A background disc-resident program must not schedule another background disc-resident program with waiting. RTE aborts the calling program and prints the error message "SC03". A real-time disc-resident program, however, may schedule another real-time disc-resident program with waiting, because real-time disc-resident programs are swapped according to their priority when they conflict over use of their core area.

- All other schedule combinations are legal: a disc-resident can call a core-resident, a core-resident can call a disc-resident, and a core-resident can call a core-resident.

- A Schedule EXEC Call with no waiting (ICODE = 10) causes the specified program to be scheduled for execution according to its priority.

PARAMETERS

- If FMGR is scheduled with wait and an MS command is executed, the scheduling program may call RAMPR to get the LS track and logical unit. These will be returned in parameter two in the same format as the base page LS word; i.e., the sign bit set indicates logical unit 3, else 2 and the track number is in bits 14 through 7.

- Schedule with wait example:

```
      DIMENSION I(5),NAME(3)
      EQUIVALENCE (IA,REG)
      DATA NAME/2HFM,2HGR,2H    /
        :
 5    REG = EXEC(9,NAME,2HMY,2HFI,2HLE,1)
 C    IF NOT SCHEDULED, LOOP IF (IA)5,10,5
 C    PUT LS TRACK IN I(2)
10    CALL RMPAR(I)
        :
```

Cartridge List

Purpose

To list the active cartridge labels and their status.

Format

: CL

COMMENTS

- The computer lists the status information on the list file. Note that the list file can be a device (e.g., line printer), or a file (refer to the LL command). The information is preceded by the following heading:

```
LU      LAST TRACK      CR      LOCK
```

Where

LU	=	logical unit number of the cartridge.
LAST TRACK	=	last track on that LU assigned to FMP.
CR	=	cartridge reference number.
LOCK	=	the locking program's name (or blank).

COpy files

Purpose

To copy or transfer all files from one cartridge to another cartridge.

Format

:CO, *label1, label2*

Where

label1 is the cartridge label where the files are presently residing.

label2 is the new cartridge label to which the files are being transferred.

NOTE

Both *label1* and *label2* can be either an LU(-) or CR(+).

COMMENTS

- All files being transferred must have unique names (i.e., no two files having the same name can reside together on the same cartridge). If files with the same name are located on both *label1* and *label2* the file on *label1* is not copied onto *label2*, but is reported with a FMGR -002 (duplicate name) error. For example, if files A, B, C, and D on cartridge *label2* are to be copied to cartridge *label3*, and file C already exists on *label3*, the following information would be printed on the log device:

```
:CO,2,3
A
B
C
FMGR -002
D
:
```

- Restrictions associated with the COPY routine are as follows:

  a. *label1* and *label2* are both mounted.

  b. *label1* is not *label2.*

  c. Type 3 or above files must contain an EOF. This implies that all records between the beginning of the file and the end-of-file must be valid.

NOTE

The COPY routine transfers files record per record. Records longer than 128 words are truncated.

CReate file

Purpose

To create a file name on a cartridge.

Format

:CR, *namr*

Where

namr is the file name and parameters.

COMMENTS

- Only a file name is created, no data is transferred.
- Parameters not included in *namr* default to 0. The file type must be specified and not be type 0, and the size must be greater than 0.
- If the file type is 3 or greater, an EOF mark is written at the beginning of the file.

CReate type 0 file

Purpose

To create type 0 file with special directory entries for device control.

Format

```
                  ,REad  [,BSpace [,EOf    [,BInary ]]]
:CR, namr, lu     ,WRite  ,FSpace  ,LEader   ,AScii
                  ,BOth   ,BOth    ,PAge     ,numeric
                                   ,numeric
```

Where

namr is the file name and security code parameter. The file type parameter is 0 which is default.

lu is the logical unit number of the device (not a disc) to be controlled.

The *REad, WRite*, and *BOth* parameters indicate the legal input/output.

REad indicates an input request is legal (implies *FSpace*).

WRite indicates an output request is legal (implies no *FSpace*).

BOth indicates both input and output requests are legal.

The *BSpace, FSpace* and *BOth* parameters indicate legal spacing. Default is no spacing.

BSpace indicates backspacing is legal.

FSpace indicates forward space is legal.

BOth indicates both backspace and forward spacing is legal.

The *EOf, LEader*, and *PAge* parameters specify the control subfunction for the end-of-file WRITE request. Default is *LEader* if the driver number is 02_8. If the driver number is greater than 16_{89}, default is *EOf*; otherwise the default is *PAge.*

EOf specifies an end-of-file mark (magnetic tape).

LEader specifies paper tape leader.

PAge specifies page eject for line printer, or two line feeds for a TTY.

numeric user specified control subfunction. The number entered is interpreted as decimal (unless followed by a B) with only the last five bits used.

The following parameters specify the subfunction portion of the input/output request. Default is *AScii.*

BInary specifies binary data transfer. (*BI* is used here. *BN* is used in Dump and Store.)

AScii specifies ASCII data transfer.

numeric user specified control subfunction. The number entered is interpreted as decimal (unless followed by a B) with only the last five bits used.

COMMENTS

- A type 0 file may exist only in the file directory for the system disc (LU2). When created, the type 0 file will be entered prior to all files as the first entry in the directory. This means that in order to assume this position the system disc must be locked when creating the type 0 file.
- A type 0 file has a special directory entry (Appendix A), and is used to control devices other than discs which use standard peripheral calling sequences. Type 0 files may exist only in the system discs' file directory and may only be created and purged by the operator interface FMGR. Cooperating programs may use type 0 files as a means of controlling access to a device (see exclusive open in Section I). Type 0 files also afford a measure of device independence in that the standard file commands can be used to control the device.

Dismount Cartridge

Purpose

To make a cartridge unavailable to the FMP, and remove its entry from the disc directory.

CAUTION

The DC command must be issued before a cartridge is removed.

Format

: DC, *label*

Where

label is the parameter relating to *namr*.

label > 0 indicates CR.

label < 0 indicates a logical unit number.

label = 0 is not allowed.

COMMENTS

- The cartridge designated by *label* is first locked (refer to the heading LOCKING). The cartridge's directory entry is then removed from the cartridge directory, and the last FMP track reported on the log device. It is suggested that this last track number be written on the platter to facilitate the future restoration of the cartridge with the MC Command.

- The File Manager will not allow cartridges assigned to LU2 or LU3 to be physically removed with the DC Command. This is because LU2 and LU3 are both located in the RTE System track assignment table, and must stay mounted in order to properly configure the table when the system is initially started (booted). If the DC Command is issued to LU2 or LU3, the cartridge is locked and removed from the disc directory list. The cartridge is then immediately remounted at the bottom of the list. With this feature, the DC Command can be used to change the order of the cartridge directory list. For example, the directory list shown below is the order of cartridges immediately after the MC Command was used to mount the disc at LU14.

LU	LAST TRACK	CR	LOCK
02	0201	00002	
03	0201	00003	
14	0201	00055	

To place the peripheral disc (LU14) at the top of the directory list, issue the following commands (for this example only):

: DC,2

: DC,3

The File Manager will change the directory list as follows:

LU	LAST TRACK	CR	LOCK
14	0201	00055	
02	0201	00002	
03	0201	00003	

- Application. The restriction on physically removing the cartridge assigned to LU3 can be circumvented if the following requirement and procedure are adhered to.

- Requirement. All cartridges to be mounted at LU3 must be initialized to use the same first track. It is recommended that track 0 be selected as the first track to avoid the possibility of the loader or system placing a program in the area.

- Procedure. Enter the following commands.

  :DC, -3 (Insures all files are closed)

  Remove the cartridge from the drive and insert the replacement.

  :DC, -3 (Places new cartridge label in the disc directory)

  The above procedure will work only if both cartridges have been initialized to use the same first track (recommended track 0). If the new cartridge has not been initialized FMGR will lock it. An attempt to initialize the new cartridge at this point will result in FMGR error 59 because the directory tracks are already assigned to D.RTR. This is solved as follows:

  * RT, D.RTR (RTE command to release D.RTR tracks)

  * ON, FMGR (FMGR will reassign D.RTR tracks on LU2, then terminate)

  * ON, FMGR

  : IN, *master security code*, -3 etc. (Initialize LU3)

  This completes the procedure.

- Application. Operator commands that use *namr* can take advantage of the default characteristic of the *namr* subparameter *label*. For example, when creating a file with the CR Command, and *label* is allowed to default to 0, the file is placed on the disc at the top of the directory list. Refer to the heading NAMR for more information on *label*.

Directory List

Purpose

To list the contents of file directory of one or all of the mounted cartridges.

Format

: DL [, *label*[, *master security code*]]

Where

label is the cartridge identification, positive for CR or negative for logical unit number.

master security code is the two-character FMP master security code designated at initialization time.

NOTE

If the master security code is 0, default in command will not obtain long list – a code (any code) must be supplied.

COMMENTS

- The *label* parameter indicates which cartridge file directory is to be examined. If *label* is not present, or it is a 0, all mounted cartridge directories will be listed. There are two list formats. Figure 3-1 shows the format when the master security code is not provided.
- Figure 3-2 shows the format when the master security code is provided. In addition to the information described in Figure 3-1, the file security code plus track and sector addresses are provided.

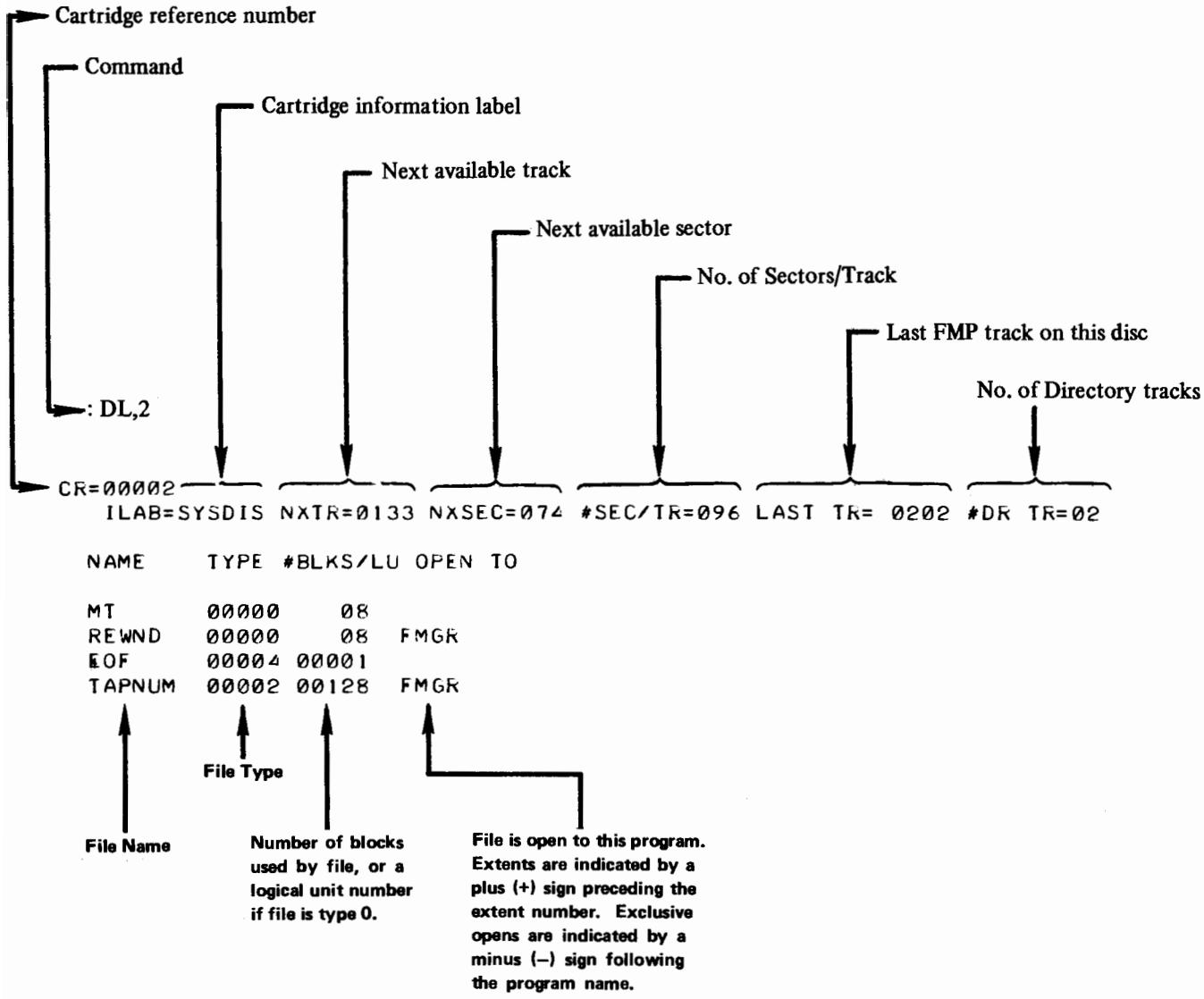


Figure 3-1. Short List

: DL,2,55

```
CR=00002
 ILAB=SYSDIS NXTR=0133 NXSEC=074  #SEC/TR=096 LAST TR= 0202 #DR TR=02

NAME    TYPE   #BLKS/LU SCODE   TRACK SEC   OPEN TO

MT      00000      08    00000
REWND   00000      08    00000                FMGR
EOF     00004   00001    00000   0100 000
TAPNUM  00002   00128    12345   0100 002    FMGR
```

File security code

Track and sector address

Figure 3-2. Long List

DUmp

Purpose

To transfer the contents of a file to another file.

Format

:DU, *namr1,namr2* [, *record format,EOF control* [, *file #* [, *#files*]]]

— or—

:DU, *namr1,namr2* [, *record format* [, *file #* [, *#files*]]]
[, *EOF control*

Where

namr1 is the name of the file, or logical unit number from which data is to be transferred.

namr2 is the name of an existing file, or logical unit number (other than a disc) to which *namr1* is to be transferred.

record format is the record format as applied to *namr2*. Refer to Comments for more information.

EOF control saves or inhibits end-of-file marks on *namr2*. Refer to Comments for more information.

NOTE

Only one comma is required when *record format* and *EOF control* parameters are omitted. No comma required if only one of the parameters omitted.

file # indicates the relative file to be written in on *namr2*. This parameter must be greater than 0 if supplied. Default is 1. For example, if *file #* is 3, then 2 files will be skipped on *namr2*, and the data will be placed in the third file. If *namr2* is a disc file, the indicated number of zero length records are skipped.

#files indicates the number of files or zero length records to be transferred from *namr1*. This parameter must be greater than 0 if supplied. Default for *#files* is as follows (refer to Figure 3-3 in Comments):

namr1 is a Disc File.

a. *file #* not supplied, and *#files* not supplied.

#files defaults to 9999 files and all of *namr1* is transferred.

b. *file #* is supplied, and *#files* not supplied.

#files defaults to 1 sub-file.

namr1 is a Non-Disc File.

a. *file #* supplied or not supplied, and *#files* not supplied.

#files defaults to 1 sub-file.

In the event it is not known how many files or sub-files exist, *#files* can be an exceptionally large number and only what is required will be transferred.

COMMENTS

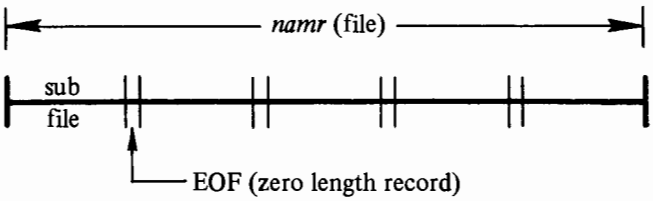


Figure 3-3. Sub-File Example

- *namr1* and *namr2* must be defined. If either one refers to an illegal logical unit number, FMGR will be aborted.
- The third parameter (*record format*) selects the record format. If this parameter is missing, default is derived from the file type as specified in the file directory entry for *namr1*. If the file type for *namr1* is not specified, final default is ASCII. The record format choices are:

  AScii indicates that ASCII records are to be transferred.

  BReloc. indicates that binary relocatable records are to be transferred. A checksum is done.

  BNary indicates that binary records are to be transferred without checksum.

  BAbs. indicates that binary absolute records are to be transferred. A checksum is done.

  MTape indicates that magnetic tape ASCII records are to be transferred.

  MS indicates that magnetic tape SIO (System Input/Output) records are created on *namr2*. Standard records are expected on *namr1*.

  MSBR indicates magnetic tape SIO binary relocatable records (same as MS + BR).

  MSBA indicates magnetic tape SIO binary absolute records (same as MS + BA).

- The *EOF control* parameters (*IHibit,SAve*) controls end-of-file (EOF) marks when transferring data. (Refer to Section I, FMP Technical Discussion for more information on EOF marks.) If this parameter is missing, an EOF is written at the end of the data (end of file on input). In addition, if *namr2* is a logical unit number and refers to a punch device, or if *namr2* refers to a type 0 file which was created with the *LEader* option, then an EOF (leader) is written at the beginning of the file. In either case, any zero length records (disc file), or imbedded EOF's (non-disc file) on *namr1* are ignored. For example, DU,8,9,AS,1,3 would merge three files on LU8 into one file on LU9.

  IHibit inhibits writing an end-of-file mark after the data. Useful only when *namr2* is not a disc file.

  SAve saves any EOF's in *namr1* on *namr2*. In disc files, EOF's are saved by writing a zero length record, and interpreted by reading a zero length record. This is effective when several tapes or magnetic tape files are to be stored in one disc file and then separated again when the file is dumped. On magnetic tape the separation is by end-of-file marks, and on paper tape by leader. On relocatable files saved from the load-and-go area the EOF's will be between programs.

- The Dump and Store Commands differ as follows:

  a. Store creates a file (unless logical unit version of *namr2* is used), while Dump does not.

  b. Store applies *file #* and *MS* against the input file; Dump against the output file.

  NOTE

  The DUMP routine transfers file record per record. Records longer than 128 words are truncated.

- Normally, when a TTY is used for output, it is treated as a list device. This means that its EOF default is a double space. If the TTY is to be used to punch tape, this default must be avoided using one of the following means.

  a. Use the IH option – an EOF is not written. However, leader is not punched.

  b. Create a type 0 file referring to the TTY, and specifying *LEader* as the EOF parameter. In dumping to the type 0 file (TTY) leader will be punched before and after the data. For example:

  :CR,TTY,1,BO,,LE

  :DU, *namr1*,TTY

- The DU Command can be used to concatonate files. For example, to add file B to the end of file A the command would take the form of:

  : DU,B,A,,2

  File B may be any legal *namr* and the omitted parameters may be as required to correctly read file B.

INitialize

Purpose

To initialize cartridge parameters and clear a cartridge directory, or change the master security code.

NOTE

This command does not format a brand new virgin cartridge. Refer to Virgin Cartridge Initialization under the FMGR TURN-ON heading.

Format (initialize parameters, clear directory)

:IN, [*master securitycode*], *label1*, *label2*, *id* [,*1st trk* [,*#dir trks* [,*#sec/trk* [, *bad tracks*]]]]

Format (change master security code)

:IN, *master security code*--*new security code*

NOTE

-- are two minus signs.

Where

master security code	is the two-character FMP master security code designated at initialization time. When changing codes, *new* implies a new code and is separated from the old code by two minus (-) signs.
label1	is the cartridge reference label, positive for CR or negative for logical unit number. If the cartridge has not been initialized, a CR label will not have been established; therefore, *label1* must be a negative logical unit number.
label2	is the new cartridge reference label and must be > 0.
id	is the cartridge information label (ILAB – refer to Figure 3-1/3-2), and must have the characteristics of a legal file name. Refer to text under the heading NAMR for a description of a legal file name. *id* is printed in the head of the directory listing.
1st trk	is the first track to be used on the cartridge (relative to the first track assigned to RTE; see Comments). Null defaults to track 0.
#dir trks	is the number of directory tracks. Null defaults to 1 directory track. This parameter must be less than 48.
#sec/trk	is the number of 64 word sectors per track. Null is allowed only for cartridges on same channel as LU2 or LU3 in which case this parameter is ignored.
bad tracks	is the bad track list. Up to six track numbers may be entered separated by commas

INITIALIZE PARAMETER COMMENTS

- Before a new cartridge can be initialized with the IN Command, it must first be mounted with MC Command. For a brand new virgin cartridge, refer to the ON, FMGR Command.
- On re-initialization, FMGR first checks that all files are closed (refer to LOCKING). If not, FMGR error -8 results. FMGR then checks the new track parameters entered. If the new parameters raise the first track, or lower the directory into an existing file, the message FMGR 060 is printed. This is a caution message that allows you to abort the initialization if incorrect track parameters have inadvertently been entered. Entering NO causes the initialization to be aborted (the ?? Command also aborts the initialization). **Entering YES causes all files on the disc to be purged.**
- Initialization and re-initialization of the system and auxiliary discs (LU2 and LU3) also present possible track assignment conflicts. If additional tracks are requested (i.e., first track parameter lowers FMP area into RTE system area), FMGR checks on the availability of the required tracks, and assigns them if they are available. If all the required tracks are not available, the highest required but not available track number is reported with a FMGR 059 error, and the IN Command is aborted. The IN Command can then be re-entered with the first track parameter equal to the track reported plus one. If fewer tracks are assigned during re-initialization, the excess tracks are returned to the RTE System. Note that if the first track is lowered, the disc must be packed to use the new area (see the PK Command).
- Bad track information is returned to the user during RTGEN or, if discovered by the FMP, as an error return to the caller. Bad track information is entered using the FMGR Initialization Command. This information is kept in the affected disc's first directory entry. If, at creation or during a packing move, the area for the file includes a bad track, the first track of that file will be increased until the file no longer contains any bad tracks. If the CREAT routine is to use the rest of the disc, the whole area will be above the highest bad track. If, during packing, a file is found to include a declared bad track, the file will be purged. (Note: This can only happen through an explicit declaration of a bad track by an operator who knows the system security code.) In this manner, only the PACK and CREAT routines need be aware of bad tracks. Bad tracks discovered by the FMP result in an error return to the caller.

CHANGE SECURITY CODE COMMENTS

- The current master security code is changed when two new characters are supplied for the *new security code parameter*. Exceptions are as follows:
  a. Colon, comma, or a leading blank are not allowed.
  b. Non-printing characters are acceptable.
  c. The master security code may be changed to zero (no security) by specifying blanks.
- If the current master security code is zero (no security) then any two characters may be used for the security parameter; however, two characters must be entered to maintain positional relationship.
- Be sure to remember the new code. Once it is entered, it cannot be obtained with any FMGR Command.

LIst

Purpose

To print the contents of a file on the current list file.

Format

:LI, *namr* [, *Source* | , *Binary* | , *Directory*]

Where

namr is the name of the file, or a logical unit number.

Source indicates ASCII source format.

Binary indicates binary format.

Directory indicates that only the header containing file information (directory entry) is to be printed.

NOTE

Only the first letter is required for the above format parameter.

COMMENTS

- If *namr* refers to a file, the file is listed with the following heading:

  file name T = *file type* IS ON CR *cartridge label* USING *nnnnn* BLKS R = *record length*

- If *namr* is a logical unit number, then stars (*) are printed in place of the file name as shown below.

  ***** T = 0 IS ON LU*xx*

- The default format specifications are related to file types. If the format is not specified, source is assumed for type 0, 3 and 4 files. Binary is assumed for all other type files.

- If the list device is a TTY, the listing starts in column one; otherwise, each line is preceded by two blanks.

- Source format consists of a line number followed by the ASCII text.

- Binary format for each record consists of:

  a. A blank line.

  b. The record number: REC#*nnnnn*

  c. A blank line.

  d. As many lines as needed for the record with eight words per line. Each word is printed in octal followed by ASCII if a legal ASCII character is found (illegal characters are filled with blanks). The octal and ASCII representations are separated by a vertical column of asterisks. Refer to Figure 3-4 for an example printout.

  Zero length records will be printed out as just a record number (parts a, b, and c only).

- List does not support absolute binary files from paper tape.

- Binary format for a full line is 72 characters on a TTY. On any other device, 74 characters are sent. Lines are truncated after the last non-blank character. The star separator is considered blank for this test.

- List does not support records of more than 128 words, and if source mode, truncates the total line to 72 characters.

```
REC# 00001

010400 020000 ***** 000000*      CDVR12
000000 000000 ***** 000000*
000000
```

Figure 3-4. Example List of a Binary Record

LLu (list file change)

Purpose

To change the current assignment of the list file.

Format

:LL, *namr*

Where

namr is the name of a file, or a logical unit number.

COMMENTS

- *namr* may refer to any existing file or logical unit.

- The list file is where the commands LL, CL, and DL direct their output.

LOglu (log device change)

Purpose

To change the logical unit number of the system log device.

Format

:LO, *lu*

Where

lu is the logical unit number of the new TTY type log device (i.e., DVR00 or 05). Note that *lu* can not be a file name.

COMMENTS

- None

Mount Cartridge

Purpose

To notify the FMP that a cartridge has been mounted and is available for use.

NOTE

A brand new virgin cartridge must be formatted prior to using the RTE System. Refer to Virgin Cartridge Initialization under the FMGR TURN-ON Heading.

Format

:MC, *lu* [, *last track*]

Where

lu is the logical unit number of the cartridge being mounted.

last track is the last track on the cartridge available to the FMP.

NOTE

For a new cartridge, the *last track* parameter establishes the last track. For a previously initialized cartridge, use the last track reported in the DC Command when the cartridge was removed.

COMMENTS

- The following conditions apply to the *last track* parameter:

  a. If *lu* is 3, then *last track* is ignored and the RTE System defined last track is used.

  b. If *last track* is absent, and the disc driver is DVR31 (moving head), the *last track* parameter is defaulted to the RTE System defined last track.

  c. *last track* must be supplied if the disc driver is DVR30 (fixed head), and *lu* is not 3.

- This command places the cartridge at the bottom of the disc directory.

  NOTE

  If you desire to have this cartridge moved to the top of the directory, refer to the DC Command.

- The *lu* parameter may be negative (to conform to the *label* definition) or positive, but in either case it is a logical unit and must refer to a disc.

- If a cartridge is already mounted at *lu*, or the cartridge being mounted has the same label as a currently mounted cartridge, FMGR error 012 results and the command is aborted. Refer to the DC Command.

- FMGR checks the cartridge's directory (at the last track) to determine if the cartridge has been previously initialized. If the cartridge has been previously initialized, the cartridge label must be unique. If not unique, FMGR 012 results. If the cartridge has not been previously initialized, it is locked (i.e., made unavailable). An Initialize Command must be issued to initialize and unlock the cartridge.

Move Relocatable

Purpose

To transfer the file at *namr* to the load-and-go area.

Format

:MR, *namr*

Where

namr is the name of the file, or logical unit number to be transferred.

COMMENTS

- A checksum is performed on the data transferred.
- *namr* may contain one or more relocatable modules.
- If an EOF is detected on *namr* and the last record transferred was not an END record (see Appendix F), FMGR will print:

  FMGR 006

  and suspend. The operator should load the remainder of the relocatable module and enter:

  *GO, FMGR

  on the system TTY.

Move Source

Purpose

To transfer a file to a logical source (LS) track before operating on it with EDIT or the Assembler or one of the compilers.

Format

:MS, *namr* [, *prog name* [, *IH*]]

Where

namr is the name of the file, or logical unit number to be transferred.

prog name is the name of a program. If given, the LS tracks are assigned to that program, if not given, the LS tracks are assigned to the program EDIT.

IH is a literal. If supplied, the LS word on the base page is not set, and the RTE System LS command is required.

COMMENTS

- When this command is typed, the first LS track is reported on the log device as follows:

  FMGR 015

  LS IS ON LU *n* TRACK *nnn*

  If FMGR was internally scheduled with waiting (refer to ON, FMGR), the scheduling program may obtain the LU and track number by calling RMPAR after it regains control. The LU and track numbers are returned in RMPAR parameter number two in the same format as the base page LS word. That is:

  Bit 15 = 0 is logical unit 2

  Bit 15 = 1 is logical unit 3

  Bits 14–7 is the track number

  Bits 6–0 is the sector address (always = 0).

- The MS command will not support files with record lengths greater than 128 words.
- It is not advisable to use FMGR as *prog name*, because all tracks assigned to FMGR are released upon entry to the MS command, and on termination of FMGR.
- The Assembler and compilers (ASMB, FTN, FTN4, and ALGOL) also release any tracks assigned to them upon termination. Consequently, if another program in the RTE System is requesting tracks, your source program could be lost. (It is still accessible in the file however.)

PacK

Purpose

To close up gaps in between files and utilize the tracks left from purged files.

Format

:PK[, *label*]

Where

label is the cartridge reference label, positive for CR or negative for logical unit number. Note that if *label* = 0, or is not entered (default = 0), then all cartridges are packed. Exceptions are described in Comments.

COMMENTS

- The PK Command purges all files which contain bad tracks. The bad tracks must have been previously declared in the Initialize Command. Refer to Section I, FMP Technical Discussion for more information on bad tracks.

  NOTE

  If you do not want the file with bad tracks purged, save it on another cartridge.

- PK moves each file, if necessary, and updates the file's directory entry to reflect the new address. When the end of the file directory is sensed (i.e., all files have been packed), the PK Command then packs the directory removing any entries for purged files.

- If during execution of the PK Command, the system should fail, it is possible to lose, at most, one file. The following data is provided to aid in determining which file, if any, may have been lost.

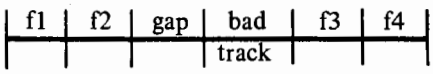
  a. A system failure during any part of a pack will leave the disc locked. This lock may be cleared by issuing a Dismount Cartridge Command (of course, you must then issue a Mount Cartridge Command unless the cartridge is LU2 or LU3).

  b. Files are moved into gaps (created by purges) one at a time as follows:

     1. If the file is smaller than the gap, then the file exists in two places. The original place still contains good data, and is accessible through the directory – no file is lost in this case.

     2. If the file is larger than the gap, then when the gap is filled, a portion of the file being moved is overlayed. If a system failure occurred at this time, the directory could point to bad data. The affected file may be found by examination of the length, first track, and sector addresses the directory list provides. The file following any gap is the affected file. Its size may be compared with the gap to see if it was lost. Remember, the file's directory entry is updated last and any bad tracks affect the gaps left between files (see below).

     3. The directory entry is updated in place after the file is successfully moved.



        If f3 fits in the gap, f3 will be transferred; if not, the gap will be retained.

  c. After all files are moved, the directory is packed as follows:

     1. The directory entries are first written on a disc track obtained by FMGR from the system. Only good entries are transferred; purged directory entries are omitted. A system failure at this time would lose nothing.

     2. This track is passed to D.RTR which in turn writes it into the directory area overlaying the old directory. A system failure at this time could result in the directory containing duplicate entries. This condition may be corrected by:

        a. Storing the affected files in new files with either different names or on different cartridges.

        b. Purging the affected files.

        c. Changing the names of the moved files back to the original names.

- A non-lock error (–8) or a disc not found error (–6) will be followed by an additional message indicating which disc could not be locked or found. When the *label* parameter is not specified, the cartridge will be reported as a –LU. If *label* is specified, the report will be the cartridge reference number (CR). Examples:

  :PK, 30 (30 not previously defined)
  FMGR –006
  FMGR 030

  :PK
  FMGR –008 (not lockable)
  FMGR –002 (LU2)

  In the last example LU2 will not be packed because there is at least one open file on it; however, all other mounted cartridges will be packed.

PUrge

Purpose

To remove a file and all its extents from the system. Note that this command is the only method available to remove a type 0 file.

Format

:PU, *namr*

Where

namr is the name of the file, or logical unit number to be purged.

COMMENTS

- If the file being purged is the last file on the disc (excluding a type 6 file), all area from the last unpurged or type 6 file up to and including the purged file, is returned to the system. If a purged file is not the last file, then its area may be recovered only by packing, or by purging all files after it in the directory. Type 6 file area may only be recovered by packing.

Re Name

Purpose

To change a file name to a new name.

Format

:RN, *namr, nuname*

Where

namr is the existing file name and parameters.

nuname is the new name unique to the cartridge.

COMMENTS

- The file specified by *namr* must not be open when this command is issued. If *namr* has a non-zero security, the proper security code must be included. If *namr* includes a cartridge reference number, only that cartridge is searched for *namr*; otherwise, all mounted cartridges are searched. In the case where all cartridges are searched, only the first file found with the given *namr* will be changed.

- *nuname* is the new name of the same file. Sub-parameters cannot be changed with this command.

Restore Program

Purpose

To restore a program (file) saved by the FMP, so that it may be run in the RTE System.

Format (to restore a program)

:RP, *namr*

Format (to assign *program's* ID segment to *namr*)

:RP, *namr, program*

Format (to release *program's* ID segment)

:RP, , *program*

Where

namr is the name of a type 6 file on LU2 or LU3 which was created on the current system using the SP Command. Note that a logical unit number is not allowed.

program is a program name.

COMMENTS

- The program assigned as an RTE System program may be accessed by all the usual RTE System Commands. The program is given the same time parameters and priority assignment as when it was saved.

- *namr* must refer to a type 6 file on LU2 or LU3 which was created on the current system using the Save Program Command. (Note that *namr* may not be a logical unit number.) The first five characters of *namr* define the program name which need not be the name the program had when it was saved with the SP Command. For example, a program with an actual name of SMPLE (NAM statement in program) is saved as SMPLE. The file name is later changed from SMPLE to TEST01 with the

  RN, SMPLE, TEST01

  command. When it is restored with the RP Command, TEST01 (the type 6 file name) would be used. Then when the program is scheduled in the RTE System,

  *ON, TEST0

  would be used.

- If the program saved is still in the system, and the program is restored using the same name it was saved with, FMGR error 023 results (Duplicate Program Name).

- In the first Format example shown, the first five characters of the file name will be the program name. A blank ID segment is established to point to the tracks belonging to the named file. If FMP cannot find a blank ID segment, an FMGR error 014 results.
- In the second Format example shown, the program called *namr* would use *program's* ID segment. *Program* must be inactive and must have been set up by a previous RP Command. *Program* is removed from the RTE System. If *program* does not have an ID segment, an FMGR error 009 results and RP reverts to the first Format. If *program* is not inactive, FMGR error 018 results. This problem is solved by issuing the RTE System Command:

  OF, *program*, 1
- If the *program* has completed and it is desired to return the ID segment to the RTE System, the third Format example is used. If *program* did not have an ID segment, FMGR 009 results but does not transfer to the log device.
- The file *namr* may be purged while an ID segment points to it, and while it is running in the system. The program is not affected in any way (in the RTE System) by this action. To recover the file's space on the cartridge (with the Pack Command), the file's ID segment must first be released with the command:

  :RP, , *program* – or – *OF, *program*, 8

  Failure to return the ID segment before issuing the Pack Command results in the FMGR 011 error.
- The RP Command does not permanently modify the RTE System area of the cartridge. Therefore, any program assigned will not be present on rebooting of the system.

Save Program

Purpose

To place a disc resident program into a type 6 file.

Format

:SP, *namr*

Where

namr is the name of a disc resident program. Note that a logical unit number is not allowed.

COMMENTS

- *namr* is created as a type 6 file, and the current system program with the same name as *namr* (first five characters), is stored in *namr*.
- A program's name is a maximum of five characters long (i.e., SMPLE). A file name can be six characters long. Therefore, it is possible to create a type 6 file called SMPLE1 for one version of the program, and another called SMPLE2 for a second version of the same program. If the program's name is less than five characters long, the program must be saved using only the characters in the name. For example, if a program's name is LOCF, it must be saved using the

  :SP, LOCF

  command. To change the program name use the RN Command thus,

  :RN, LOCF, TEST01

  When restoring the program use

  :RP, TEST01

  When scheduling the program in the RTE System use

  *ON, TEST0
- The SP Command is usually used in conjunction with the RP Command. For example, you have a source program that you have debugged, edited, compiled, and loaded, and have successfully run in the RTE System. You want to save this program in its present loaded form for a future use, without having to recompile, reload, etc.. Issue the SP Command using the program's name, and a type 6 file with the same name is created. Later, to rerun the program, use the RP Command, and then issue the RTE System ON Command. The program can be saved while it is running. After it has been saved it can be aborted with the RTE System Command:

  *OF, *program*, 1

  – or –

  *OF, *program*, 8

  NOTE

  The SP Command will store the program file on any mounted cartridge (through the *label* subparameter of *namr*). However, the file must be located on LU2 or LU3 before the RP Command will accept the file. The file may be moved with the ST Command.
- The following *namr* parameters are treated as shown:

  security code is defaulted to 0, otherwise user selected.

  label is defaulted to -2.

  file type is forced to type 6.

  file size is forced to the required size.

  record size is forced to 128.

SAve

Purpose

To save the logical source (LS) or load-and-go area (LG) in a file.

Format

:SA ,*LS* / ,*LG* , *namr*

Where

LS	indicates the logical source area
LG	indicates the load-and-go area
namr	is the name of a file (created by this command), or a logical unit number.

COMMENTS

- The SA, *LG* implies a checksum. After each end record, including the final end record, either a zero length record (disc file), or an EOF (non-disc file) is written, depending on the file.
- The file size, if not specified, is defaulted as follows:
  a. The SA, *LG* Command computes the maximum possible file size from the amount of LG area used, and uses this size to create the file. Therefore, extents will never be created, and the file will only be as long as needed.
  b. The SA, *LS* Command sets the size to one-half the number of blocks on a system disc track.
- After the EOF is written, its position is checked. If extents were created, the file size is not shortened. If the EOF is within the main file, the remaining unused disc space is returned to the system.
- The file type, if not specified, is defaulted as follows:

  :SA, *LS* – defaults to type 4 (source)

  :SA, *LG* – defaults to type 5 (relocatable)
- If there is not enough cartridge space available to accommodate the file, an FMGR –6 error results, and any portion of the file already saved is purged.

NOTE

The Save Command does not support records longer than 128 words.

STore

Purpose

To transfer or store records from a file or logical unit number, to another file or logical unit number. Note that *namr2*, if not a logical unit, is created by this command.

Format

:ST, *namr1*, *namr2* [, *record format*, *EOF control* [, *file #* [, *#files*]]] (hp)

– or –

:ST, *namr1*, *namr2* [, *record format* [, *file #*[,*#files*]]] / , *EOF control*

Where

namr1	is the name of the file, or logical unit number from which data is to be transferred.
namr2	is the name of the file, or logical unit number to which *namr1* is to be transferred. *namr2* is created by this command.
record format	is the record format as applied to *namr1*. Refer to comments for more information.
EOF control	saves or inhibits the end-of-file marks contained in *namr1*.

(hp)

NOTE

Only one comma is required when *record format* and *EOF control* parameters are omitted. No comma required if only one of the parameters omitted.

file #	indicates the relative position the file is to be read from on *namr1*. This parameter must be greater than 0 if supplied. Default is 1. For example, if *file #* is 3, then 2 files (or zero length records) will be skipped on *namr2*, and the third file will be read.
#files	indicates the number of files or zero length records to be transferred from *namr1*. This parameter must be greater than 0 if supplied. Default for *#file* is as follows (refer to Figure 3-3 in Comments):

namr1 is a Disc File.

a. *file # not supplied, and #file* not supplied.

#file defaults to 9999 files and all of *namr1* is transferred.

b. *File #* is supplied, and *#file* not supplied.

#file defaults to 1 sub-file.

namr1 is a Non-Disc File.

a. *File #* supplied or not supplied, and *#file* not supplied.

#file defaults to 1 sub-file.

In the event it is not known how many files or sub-files exist, *#files* can be an exceptionally large number and only what is required will be transferred.

COMMENTS

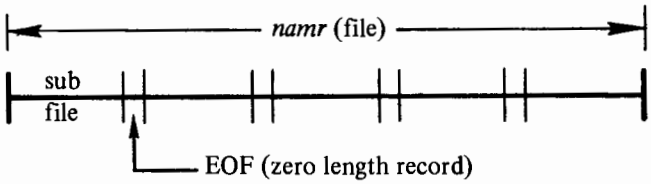


Figure 3-5. Sub-File Example

- *namr1* must be defined. If either *namr* refers to an illegal logical unit number, FMGR will be aborted.
- The file type parameter for *namr2* is defaulted as follows.

  To *namr1*'s type if *namr1* is a disc file.

  To 3 when using MT, MS, AS.

  To 5 when using MSBR, BR.

  To 7 when using MSBA, BA.

  To 3 when none of the above is specified.

  File size parameter for *namr2* is defaulted to one-half a system disc track.
- The *namr2* parameter can be a logical unit number (non-disc file), but not a type 0 file.
- After the EOF is written, its position is checked. If extents were created, the file size is not shortened. If the EOF is within the main file, the remaining unused disc space is returned to the system.
- The third parameter selects the record format. If this parameter is missing, default is derived from the file type as specified in the file directory entry for *namr1*. If the file type for *namr1* is not specified, final default is ASCII. The record format choices are:

  AScii indicates that ASCII records are to be transferred.

  BReloc. indicates that binary relocatable records are to be transferred. A checksum is done.

  BNary indicates that binary records are to be transferred without checksum.

  BAbs. indicates that binary absolute records are to be transferred. A checksum is done.

  MTape indicates that magnetic tape ASCII records are to be transferred.

  MS indicates that magnetic tape SIO (System Input/Output) records are expected on *namr1*. Standard records are written on *namr2*.

  MSBR indicates magnetic tape SIO binary relocatable records (same as *MS* + *BR*).

  MSBA indicates magnetic tape SIO binary absolute records (same as *MS* + *BA*).

- The *IHibit, SAve* portion of the third parameter controls end-of-file (EOF) marks when transferring data. (Refer to Section I, FMP Technical Discussion for more information on EOF marks.) If this parameter is missing, an EOF is written at the end of the data (end-of-file on input). In addition, if *namr2* is a logical unit number and refers to a punch device, or if *namr2* refers to a type 0 file which was created with the *LEader* option, then an EOF (leader) is written at the beginning of the file. In either case, any zero length records (disc file), or imbedded EOF's (non-disc file) on *namr1* are ignored. For example, DU, 8, 9, AS, 1, 3 would merge three files on LU8 into one file on LU9.

  IHibit inhibits writing an end-of-file mark after the data. Useful only when *namr2* is not a disc file.

  SAve saves any EOF's in *namr1* on *namr2*. In disc files, EOF's are saved/interpreted by writing/reading a zero length record. This is effective when several tapes or magnetic tape files are to be stored in one disc file and then separated again when the file is dumped. On magnetic tape the separation is by end-of-file marks, and on paper tape by leader. On relocatable files saved from the load-and-go area the EOF's will be between programs.

- The Dump and Store Commands differ as follows:

  a. Store creates a file (unless logical unit version of *namr2* is used), Dump does not.

  b. Store applies *file #* and *MS* against the input file; Dump against the output file.

  NOTE

  The STORE routine transfers file record per record. Records longer than 128 words are truncated.

- The input file can be declared the system TTY (*namr1* = LU1). For example, the ST Command is entered on the system TTY as follows:

  :ST, 1, FILE1 : : : 4:2

  After the last subparameter, carriage-return, line-feed is entered. When this is completed the TTY just sits there waiting for your input (the colon prompt is absent). Type in as many lines as desired (extents are automatically made if required). When the file is completed enter Control D for the EOF mark. Control will then return to FMGR (as indicated by the colon prompt).

SeVerity code change

Purpose

To change the system log severity code to a new number.

Format

:SV, *number*

Where

number is the new severity code number.

0 – All commands are echoed on the log device. Any error causes an appropriate error number to be printed.

1 – Inhibit command echo on log device.

2 – Inhibit error messages to log device, unless the error is severe enough to cause control to transfer to the log device for a command input.

COMMENTS

- If an error is detected, and the echo is inhibited, the erroneous command will be printed prior to the error message.

TRansfer control

Purpose

To transfer control of FMGR to a file name or logical unit number.

Format

:TR[,*namr* / ,-*integer*]

Where

namr is the name of a file, or a logical unit number.

-*integer* is a negative integer that denotes a transfer back that many steps.

COMMENTS

- There are three variations to the Transfer Command.
  a. TR, *namr*. This command transfers control to *namr* (either a file or a logical unit number). Before the transfer is executed, the *namr* in control is saved in a stack. If *namr* is a disc file, the current record number is also saved.
  b. TR (no parameter). This command causes control to be returned to the previous input *namr*. If it was a disc file, the records already read are skipped. If there was no previous input *namr*, the FMGR is terminated.
  c. TR, -*integer*. This command causes the specified number of transfers (e.g., TR, -2) with no parameters (jump back through stack) to be executed.
- Up to 10 input *namrs* may be stacked and then returned to.
- If an error is detected at any time, and it is severe enough to transfer to the LOG unit, a :TR,LOG (unless current input is from the LOG) is executed. As many commands as desired may be entered from the LOG device followed by a TR Command with no parameters to get back to the statement following the erroneous statement.

TRANSFER COMMAND EXAMPLES

The following examples show a dialog between the system and operator. For clarity, the operator responses are shaded.

a. Suppose several subroutines have been saved in the file system and are often needed by programs being developed. Suppose these subroutines are named SUB1 thru SUB10 and that SUB1 calls SUB2 thru SUB5 and that SUB6 calls SUB1 and SUB7 thru SUB10.

Given this, we create the following files:

```
file # 1    SUBS1
            :MR, SUB1
            :MR, SUB2
            :MR, SUB3
            :MR, SUB4
            :MR, SUB5
            :TR

file #2     SUBS6
            :MR, SUB6
            :TR, SUBS1
            :MR, SUB7
            :MR, SUB8
            :MR, SUB9
            :MR, SUB10
            :TR
```

Then, after compiling a program, requiring SUB1, into the load-and-go area, we can run the FMGR as follows:

```
*ON, FMGR
:TR, SUBS1          Transfer to SUBS1
:MR, SUB1   }
:MR, SUB2   }
:MR, SUB3   }       Echo of SUBS1 Commands
:MR, SUB4   }
:MR, SUB5   }
:TR         }       Transfer back to TTY
:EX                 Terminate FMGR
$END FMGR
```

We could now run the LOADR to load the program.

If we needed SUB6, the dialogue would have been:

```
*ON, FMGR
:TR, SUBS6        Transfer to SUBS6
:MR, SUB6    }    Echo from SUBS6
:TR, SUBS1   }    SUB6 transfers to SUBS1
:MR, SUB1    }
:MR, SUB2    }
:MR, SUB3    }    Echo from SUBS1
:MR, SUB4    }
:MR, SUB5    }
:TR          }
:MR, SUB7    }
:MR, SUB8    }
:MR, SUB9    }    Continuation of SUBS6
:MR, SUB10   }
:TR          }    Return to TTY
:EX               Terminate FMGR
$END FMGR
```

b. We could also save a segmented program and then to restore it, we generate the file:

```
RSSEG

        :RP, MAIN
        :RP, SEG1
        :RP, SEG2
        :TR
```

which might be invoked to restore the program.

c. If we inadvertently transfer to, for example, SEG10 (from Example a), from the log device:

:TR, SEG10	Transfer
A%&#ZZX@$SEG10WT@	Echo of ASCII of NAM record
FMGR 010	Error Message
A?	Error Message (colon missing)
:	On log device

A TR at this point would just go back to SEG10; what we need is to go back to the file prior to SEG10, so we enter:

:TR, -2 (on log device)	Transfer back to file prior to error file
: etc. . .	

??

Purpose

To expand the last error message.

Format

:?? [,*number*]

Where

number is the error code number.

If *number* = blank – Last error code issued is expanded.

If *number* = *xx* – The *xx* error code is expanded.

If *number* = 99 – All error code messages are printed on the list file.

COMMENTS

- The ?? Command may be issued after any error number and will expand that message. If issued in response to the Initialize Command error 60, the message is printed and the Initialize Command is aborted.

EXit

Purpose

To terminate the File Manager (FMGR).

Format

:EX

OPERATOR COMMAND EXAMPLES

The operator commands presented in this section can be applied by two different methods. In first method, the operator manually enters each command on the TTY to accomplish a given task. Many commands could be involved, requiring a great amount of manual labor to enter. The second method uses the Transfer Command. The operator lists all commands required to do his task in program form, that is, serially as shown in the following examples. These commands can then be punched on paper tape or cards (or typed directly into a file). A file can be created and these commands stored in it; control is then transferred to this file and the task is executed. Also, if desired, the commands can be entered through the paper tape reader or card reader by a Transfer Command to the appropriate logical unit number.

The following examples describe in first person tutorial form one of the many different procedures that can be used to accomplish a task. Once the user becomes familiar with the HP File Manager and its vast powers, he can see how to alter these procedures in many different ways.

EXAMPLE NO. 1

This example shows a method of recording a set of programs on magnetic tape (hereafter called "mag tape") as relocatable binary in SIO format for use in generating future RTE Systems. As new or updated programs become available, the File Manager will be used to update and rewrite the mag tape.

Assume you will start with each of the relocatable binary programs stored on the disc as a separate file under control of the File Manager. For illustrative purposes, the files are named:

EXEC
SCHED
RTIOC
DVR00
DVR31
ASMB
LOADR
UPROG1
UPROG2

For later use in building the mag tape, create and store a file containing nothing more than an end-of-file (EOF). At the console, enter:

```
*ON, FMGR

:CR, EOF:::4:1   (Create a file called "EOF"
                  containing only an EOF)
```

Now, plan to dump the disc files onto mag tape. Enter each file in turn, with due consideration of order of loading for later system generation (refer to the RTE Manual). Assume the mag tape logical unit number is 8, and it is enabled.

```
:DU, EOF, 8              (Transfer the EOF file to
                          the mag tape)

:DU, EXEC, 8, MSBR, IH   (Transfer first program to
                          mag tape)
```

The third parameter, MSBR, transfers binary relocatable records onto mag tape (LU8) in SIO format. The fourth parameter, IH, inhibits writing an end-of-file after EXEC. This is necessary for subsequent use of the mag tape for RTE System generation.

```
:DU, SCHED, 8, MSBR, IH
  .
  .
  .
```

Enter the commands to transfer each program onto the mag tape, including the two user programs. Make one change to enter the last program. To provide an EOF at the tail end of the program sequence, omit the inhibit (IH) parameter.

```
:DU, UPROG2, 8, MSBR
```

The mag tape is now complete and can be used in an RTE System generation. An extension of the process shown here could create a mag tape, holding a library of many programs, from which could be drawn a sub-set of programs selected for a particular RTE System, and easily duplicated for use on other copies of that system.

Now, assume that the time has come to update the mag tape. The third program of the sequence on the mag tape, which happens to be RTIOC, is to be replaced with an updated version. The process to be used is to separate the program set after the third program; saving the first piece in a temporary file, dropping out the obsolete program, picking up the remaining programs, and storing this piece into another temporary file. A separate file is assumed to contain the new version of RTIOC. The various files will be recombined and transferred from the disc onto the mag tape as a complete, updated set.

There is a problem to be solved first. Remember that the programs were written onto the mag tape without any EOF separators between them. An easy way to replace these separators is to let FMGR inserts EOF's (actually zero-length records). Remember that FMGR inserts these zero length records after each end record on transfers made from the load-and-go tracks. Assume the mag tape and system have been made ready, that is, load-and-go tracks were established with an RTE System LG Command. Turn on the FMGR (ON, FMGR) and enter the command to read the mag tape into the disc file TEMP1; and from here, to the LG tracks. Note that the MR Command cannot be used directly because it does not accept SIO records.

```
:ST, 8, TEMP1, MSBR      (Create the file TEMP1)

:MR, TEMP1               (Transfer file to load-and-
                          go tracks)
```

Now, move the load-and-go tracks to a new file, TEMP2.

```
:SA, LG, TEMP2           (Create the file TEMP2)
```

In the Save process, FMGR inserted an "EOF" between each program. The programs are now separable. TEMP1 can be purged, if desired, as it is no longer needed.

Transfer the first two programs of file TEMP2 into another temporary disc file, TEMP3.

```
:ST, TEMP2, TEMP3, BR, 1, 2   (Create the file TEMP3)
```

Starting with the first file, two files are stored in TEMP3.

The third program (RTIOC) is skipped, and the fourth thru the last program is stored in TEMP4.

:ST, TEMP2, TEMP4, BR, 4, 10 (Create the file TEMP4)

The first three programs were skipped by setting the file number parameter to 4. TEMP4 now contains the remainder of the set, starting with the program DVR00. Refer to Figure 3-6. Note that the *SAve* parameter was not specified; therefore, TEMP3 and TEMP4 no longer contain the zero length records.

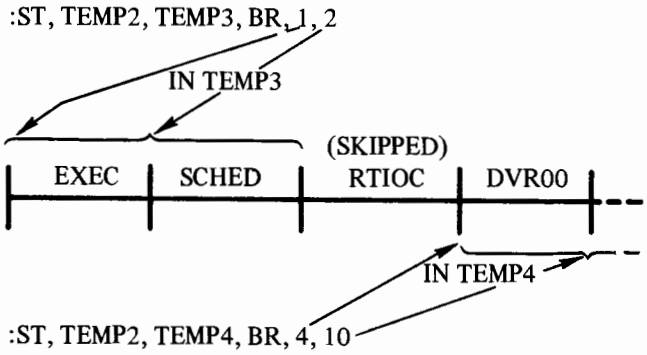


Figure 3-6. Example File Structure

The final parameter (10) is the number of files to be transferred and can be any number greater than the remaining set.

Now, combine the first section of the old mag tape; the new version of RTIOC; and the remainder of the old mag tape into one updated version. Assume the mag tape unit is enabled. Put an EOF at the head end, as before.

:DU, EOF, 8

:DU, TEMP3, 8, MSBR, IH (Dump TEMP3 to tape)

Add on the new RTIOC. Assume it is in a file named RTIOC.

:DU, RTIOC, 8, MSBR, IH (Dump new RTIOC to tape)

The inhibit (IH) is needed to prevent an EOF from being written after the files, since we intend to add more files. Now, add the remaining programs stored in TEMP4. Omit "IH" to have the final program end with an EOF.

:DU, TEMP4, 8, MSBR (Dump TEMP4 to tape)

The updated mag tape is ready. Purge all old files that are left over from the operations just performed.

EXAMPLE NO. 2

This second example relates the commands required to have the File Manager store, list, and edit a source program. Then compile, load, save, and later run the program without any intervening paper tapes.

Assume for purposes of illustration that you have written a program named "JEVA" in FORTRAN IV, and have prepared it as a punched paper tape. It is your ultimate aim to have the program JEVA stored in a file on the system disc. You plan to store the program as a load module so that it can be brought into core from time to time to execute in the background disc resident area. First, you wish to file temporary copies of the source program through various stages of editing. Second, you wish to keep copy, in the form of a file, of the relocatable binary code that results from compilation.

The first action is to load the paper tape into a source program file (type 3) on the system disc. (Note, for simplicity, it is assumed that file security is not used on this system. Also, the system disc is assumed to be first in the disc directory so that parameters will automatically default to the system disc.)

Place the paper tape in the photoreader, and enter the following commands at the system console. (It is assumed that all is in readiness for the system to operate.)

*ON, FMGR

:ST, 5, JEVA1 (Create a file called JEVA1)

The paper tape is read and transferred to a disc file JEVA1 created as a result of this command. Default values for parameters 3, 4, and 5 fit this case and simplify the entry of the command.

In preparation for editing the source program, use the FMGR List Command to obtain a listing complete with line numbers.

:LI, JEVA1 (List the file)

The listing is printed on the list device (LU6) since default parameters were used.

Move the file to the LS tracks where the RTE System Editor can access it.

:MS, JEVA1 (Transfer JEVA1 to LS tracks)

FMGR replies:

FMGR 015

LS IS ON LU2 TRACK *yy* (Indicates LS pointer)

JEVA1 has now been transferred to the RTE System LS tracks under control of the program EDIT. The LS word on the base page is set due to a default parameter in the MS Command. The RTE System LS Command is unnecessary. The FMGR should now be terminated in order to bring up the system Editor.

```
:EX
```

The FMGR replies:

```
$END FMGR
```

Enter commands to the RTE System.

```
*ON, EDIT, 1, 2, 2
```

The Editor replies:

```
/EDIT: ENTER EDIT FILE
```

Enter the contents of the edit file on the system console.

```
/R, xx
  .
  .
  .
/E
```

The Editor accepts the edit file, works it against the program JEVA1, and puts the edited version on its tracks. The Editor prints the message:

```
/EDIT:  TRACKS IN NEW FILE

/EDIT:  nn, xx

/EDIT:  END OF EDIT RUN.
```

The LS pointer must now be set to the new source file:

```
*LS, nn, xx
```

With the assumption that the edited version is acceptable, move the logical source tracks to a new file, JEVA2.

```
*ON, FMGR

:SA, LS, JEVA2          (Create the file JEVA2)
```

At this point, the source program named "JEVA" has been edited and stored in the file JEVA2. For cleanup, the pre-edit version in the file JEVA1 should be purged.

```
:PU, JEVA1              (Purge JEVA1)
```

Next, compile the program. Since the LS pointer still points to a copy left on the LS tracks, it is convenient to use that copy. However, should the copy on the LS tracks have vanished due to intervening use of the LS word on base page by another user (or by a system reboot), the file copy in JEVA2 can be recalled to the LS tracks. Terminate the FMGR, define load-and-go tracks, and call for the FORTRAN compiler.

```
:EX
```

The system replies:

```
$END, FMGR
```

Enter:

```
*LG, 2                  (Allocate 2 load-and-go tracks)
*ON, FTN4, 2, 99        (Compile JEVA2)
```

The system replies:

```
$END, FTN
```

Parameter "2" in the ON, FTN4 Command directs the compiler to compile the contents of the LS tracks. Parameter "99" causes the results of the compilation, the relocatable binary program, to be transferred to the load-and-go tracks previously defined. The binary program should now be saved in a new file.

```
*ON, FMGR
:SA, LG, JEVAR          (Create the file JEVAR)
```

Next, it is planned to load the program with the RTE Loader. After the loading operation, the resulting object code could be executed as a background disc resident program (refer to Part 5, RTE Manual). However, in this case the object code will instead be stored as a load module with the Save Program Command, and later returned to core by the Restore Program Command. Refer to the SP and RP Commands in Section III of the RTE File Manager Manual.

Since the LG pointer is still directed to the program copy on the load-and-go tracks, that copy will be loaded. If the copy were no longer available, then the file JEVAR could be recalled to the LG tracks. To load the program, terminate FMGR and call for the RTE Loader:

```
:EX
```

The system replies:

```
$END, FMGR
```

Enter:

```
*ON, LOADR, 99
```

The loader prints:

```
JEVA READY-LOADING COMPLETE
```

Note that the program name rather than the file name is reported by the loader in this case. Should the RP Command have been used to return a load module stored in a disc file to the load-and-go tracks, the first 5 characters of the file name would define the program name. To save the program turn on the FMGR and use the SP Command.

```
*ON, FMGR

:SP, JEVA          (Create type 6 file called JEVA)
```

The program JEVA (name defined by the first 5 characters in the file name) is now saved in the new file. Note that the file could be created on any of the discs under FMP control. However, when it is desired to return the load module to core, JEVA must be residing on LU2 or LU3.

Later, to restart JEVA, the RP Command is used.

```
*ON, FMGR

:RP, JEVA          (Restore JEVA as a system program)
```

The system assigns a blank ID segment (without tracks assigned if possible), set up to point to the file area. JEVA is set up as a system program and can be accessed by any of the usual system commands.

If the program is dormant, its ID segment can be returned to the system with the RP Command:

```
:RP, , JEVA        (Release ID segment)
```

If the program is to be unconditionally aborted use the RTE System Command:

```
*OF, name, 8
```

SECTION IV

ERROR CODES

SECTION IV
ERROR CODES

FMGR AND FMP ERROR CODES

The error code structure is divided into positive and negative error codes. A negative error code is usually the result of an improper interface routine call. A positive error code is usually the result of an improper FMGR command. Since many of the FMGR operator commands use an interface routine (e.g., the Save Command uses the Create Routine to create the *namr2* file), negative errors can be generated when using the FMGR operator commands. Due to the internal interaction of calls within the FMP, an improper command may cause FMGR to abort (returns control to RTE System), and leave files that are empty or otherwise incomplete on the system. An example of this sort of error is:

```
:ST,4,XYZ
```

Here 4 is assumed to be the punch. The command will create file XYZ, and then FMGR will be aborted when it tries to read from 4; XYZ will therefore be empty and corrupt (i.e., no end-of-file is contained in it). XYZ should be purged as soon as practical. You must issue the ON,FMGR command to regain control and then purge the corrupt files.

Unless otherwise stated, an error will cause a transfer to the log device without aborting FMGR. This is an actual TR command generated by the FMP to notify you that an error has been made, and remedial action is required. From this point input control is recognized only from the log device. As many commands as desired may be entered from the log device followed by a TR command (no parameters) to return to the statement following the erroneous statement.

When packing with the PK Command, a non-lock error (–8) or a disc not found error (–6) will be followed by an additional message indicating which disc could not be locked or found. When the *label* parameter is not specified, the cartridge will be reported as a –LU. If *label* is specified, the report will be the cartridge reference number (CR).

Examples:

```
:PK, 30     (30 not previously defined)
FMGR –006
FMGR 030

:PK
FMGR –008     (not lockable)
FMGR –002     (LU2)
```

In the last example LU2 will not be packed because there is at least one open file on it; however, all other mounted cartridges will be packed.

Negative error codes are shown in Table 4-1. All routines (except STATUS) are listed across the top of the table, while errors are listed down the left side. This forms a matrix that shows which interface routines can be involved in an error. For example, error –8 can occur in the PURGE, OPEN, or RENAME routines.

All positive errors are listed in numerical order following Table 4-1 and are self explanatory. Note that all interface routine error codes are also returned in the computer's A-Register.

Table 4-1. Negative Error Codes

Error Code	Error Description	Creat	Purge	Open	Close	Read	Write	Lo-cate	*Aposi-tion	Re-wind	*Posi-tion	Re-name	*Con-trol
⩾0*	None	X	X	X	X	X	X	X	X	X	X	X	X
-1	Disc down	X	X	X	X	X	X		X	X	X	X	
-2	Duplicate name	X										X	
-3	Backspace Not Legal (Type 0)										X		
-4	File too long or REC Size Error (Type 2)	X											
-5	Attempt to read or position to a record not written, or on update write an illegal record length.					X	X		X		X		
-6	Cartridge not found or file not found or no room	X	O	X	X		X						
-7	Invalid security code		X	X			X					X	
-8	File currently open: eight PGM or exclusive or LOCK rejected		X	X								X	
-9	Attempt to open type 0 as type 1 or to use APOSN on type 0			X					X				
-10*	Not enough parameters	X	X	X	X	X	X	X	X		X	X	
-11*	DCB not open				X	X	X	X	X	X	X		X
-12	SOF or EOF read or sensed					X	X		X		X		X
-13	Cartridge locked	X	X	X								X	
-14	Directory full	X					X						
-15	Illegal name	X										X	
-16	Illegal Type or Size = 0 (Creat)	X	X										
-17	Attempt to Read or Write on type 0 which does not support the operation.						X				X		
-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 first). Possible operator error (e.g., physical removal of cartridge without issuing DC Command).												
	X – implies a TRANSFER to LOG device. O – implies no TRANSFER. * – implies error or function is never reported on the log device.												

Table 4-2. FMGR Error Messages

Error Number	Associated Routine	Meaning and Action	Additional Information
001	General	Disc error.	Logical unit.
002	System Initialize	Initialize LU2.	–
003	System Initialize	Initialize LU3.	–
004	System Initialize	Illegal response to 002 or 003.	–
005	System Initialize MS	Required track not available.	Relative TAT position.
006	MR,ST,DU	FMGR is suspending itself; ready device and GO.	–
007	MR,SA,DU,ST	Checksum error.	–
008	System Initialize	D.RTR not found in ID segments. Load D.RTR.	–
009	RP	ID segment not found (no implied transfer).	–
010	Parse Routine	Input error, re-enter statement. Caused by: Missing initial colon (non-TTY input). Supplied initial colon (TTY input). Command undefined. ASCII subparameter in subparameters 3 thru 5. Subparameters in other than first two parameters. More than 5 subparameters. Command too long.	Portion of line up to and including error is printed, followed by a "?."
011	PK	Some system ID segments point to the disc to be packed. Do RP,,xxxxx or OF,xxxxx,8 on all named programs; then re-enter PK Command.	List of program names.
012	MC	Duplicate logical unit or label.	–
013	TR	Transfer stack overflow.	–
014	MS,SP,RP	Program not found in system ID segments. Also, RP, no blank ID segments.	–
015	MS	Logical Source track report follows (No implied transfer).	Track and LU of created LS file.
016	RP	The named file is not on logical unit 2 or 3. Move file to LU2 or 3 and re-issue.	–
017	RP	The named ID segment was not set up by FMGR – cannot be used or cleared.	–
018	RP	The named ID segment shows the program is not dormant; issue OF,XXXXX,1 and then re-issue RP.	–

Table 4-2. FMGR Error Messages (Continued)

Error Number	Associated Routine	Meaning and Action	Additional Information
019	RP	Checksum or setup code failed. File was not set up by an SProgram command on the current system.	–
020	CR	The given logical unit is illegal (creating a type 0 file).	–
021	CO	One of the specified discs is not mounted or both specifications refer to the same disc.	–
022	CO	The copy has been terminated. (NOTE: this is true of most copy errors regardless of this message being printed.)	–
023	RP	The given program name is already defined in a system ID segment.	–
024–049		Not defined.	
050	IN,MR,SA,SP	Illegal number of parameters; usually too few.	–
051	IN,DL	Illegal master security code.	–
052	IN,MC	Wrong logical unit. In response to 002 or 003, or no disc on given LU.	–
053	IN	Illegal disc label. Reference label must be positive non-zero integer. Information label must be a legal file name.	–
054	IN,DC,PK,DL	Disc not mounted. Issue MCartridge command.	–
055	IN,MC,DC,CR, DU,ST	Missing parameter; re-enter command; could be a required parameter which is usually optional.	–
056	IN,DU,ST,CR, LO,SA,LI	Bad parameter. No file tracks (i.e., all directory or last track below first track). ASCII code undefined.	–
057	IN	Bad track error. Track not within file area or track is in directory area.	–
058	SA	Load and go area is undefined or empty.	–
059	IN	Track not available for re-INitialization (aborts Initialization).	Highest non-available track.
060	IN	INitialization will cause loss of all files on this disc. Do you really want to? Answer: YES or NO. Caused by: a. First track is larger. b. Directory will extend into a file.	–

SECTION V

FMP SYSTEM INSTALLATION

SECTION V
FMP SYSTEM INSTALLATION

GENERAL INFORMATION

File Manager installation consists of two parts. The first part involves incorporating the package into the RTE System. The second part involves initializing the system and auxiliary discs with FMGR.

FMP INSTALLATION

The FMP must be configured into the RTE System during the program input phase of RTGEN – no provisions exist for on-line loading. (Refer to Section VI, Program Input Phase, of the HP Real-Time Manual.) The various modules of the FMP are numbered according to the order of their loading; other distinctions are as follows.

FMGR

The operator interface FMGR has a priority of 90, is segmented into five parts, and requires at least 5k of background area.

D.RTR

Subroutine D.RTR has a priority of one, requires a few words over 1k of area, and is supplied as a foreground disc resident program. However, if space permits, it is recommended that it be made core resident for greater speed. This change can be made during the parameter input phase of RTGEN. The only constraint is that D.RTR have a higher priority than any program using the FMP.

FMGR INITIALIZATION

Each time the RTE System is loaded from the disc the FMGR program is scheduled. The first time that FMGR is scheduled it detects that the FMP has not been initialized to the system and auxiliary discs. Once this initialization procedure (described in the following steps) takes place, later RTE System loads will still schedule FMGR, but initialization is no longer necessary.

When FMGR is scheduled, it obtains all available tracks on the system and auxiliary discs, and assigns the tracks to itself. FMGR then prints on the system teleprinter (TTY):

```
FMGR 002
:
```

This is a request for the user to initialize the system disc (LU2) using the IN Operator Command.

NOTE

> The security code entered at this time will be the system master security code. The code can be blank (no security) or any two characters (except a colon, comma, or leading blank). The two characters need not be printing characters. Remember, **the code may not be obtained with any FMGR Command once it is set, so be sure and remember it.**

After a successful initialization of the system disc, FMGR checks to see if there is an auxiliary disc. If so, FMGR prints on the system TTY:

```
FMGR 003
:
```

This is a request for the user to initialize the auxiliary disc (LU3) using the IN Operator Command. If no tracks are to be assigned for the auxiliary disc, its label should be specified as zero. For example:

```
:IN, JB, -3, 0
```

When a successful initialization is completed, FMGR assigns the tracks as per the IN Command parameters. FMGR then terminates (no message is printed) and returns control to the RTE System.

If the initialization was not successful, that is, if any tracks were not available, FMGR prints:

```
FMGR 005
FMGR xxx
```

005 identifies the message and *xxx* is the track's relative position in the track assignment table. This implies the track number is printed for the system disc, and the track number plus the number of tracks on the system disc is printed for the auxiliary disc. Recoveries from this error condition are:

a. Make the track available and then enter the following system commands:

```
*RT, D.RTR
*ON, FMGR
```

   This will force the FMGR to re-try the track assignment table setup.

b. Re-initialize the affected disc declaring the unavailable tracks as bad (message will still be printed but may be ignored).

c. Re-initialize the affected disc changing the first track to be above the unavailable track.

If the unavailable track is the last system track, procedure "a" must be used; i.e., this track must be available to the FMP.

APPENDIXES

Appendix

APPENDIX A
TABLES

This Appendix contains the following tables.

Table A-1. Cartridge Directory Format

The Cartridge Directory is located on the first two sectors of the last track on the system cartridge (LU2). It contains key pointers to all mounted cartridges within the FMP realm.

Sectors	Word	Contents
0/1	0	LU first cartridge
	1	Last track for FMP
	2	Label word
	3	Lock word
	4	LU second cartridge etc. • • • 0 – end of cartridge specs (up to 31 cartridges)
	124	0
	125	Set up (initialized) code word
	126	Set up security code
	127	Spare

Word 125 is the sum of the words in locations 1650 through 1657, and 1742 through 1764.

This order of cartridges can be changed. Refer to the DC Command.

Lock word is either 0, not locked, or ID segment address of locking program. Only FMGR locks the cartridge in supported software. Locked cartridges are available only to the locker to other callers; they are considered not mounted.

Table A-2. File Directory Format

The first entry in each File Directory is the specification entry for the cartridge itself. Each entry is 16 words long. The directory will start on the last FMP track of each cartridge in sector 0 for all but the system cartridge (LU2), on which it starts in the next logical directory block.

Word	Contents
0 – 2	Six-character information pack label
3	Label word (positive integer)
4	First available track
5	Next available sector
6	Number of sectors per track
7	Last available track for files + 1 (i.e., lowest directory track)
8	Negative of number of tracks in directory
9	Next available track
10	First bad track (or zero)
⋮	⋮
15	Sixth bad track (or zero)

Each cartridge has a short label (one word) and a long label. The short label is used for addressing the cartridge, the long label (six ASCII characters) is never used by the FMP and is only used for further ID when listing directories.

The sign bit will be set on word 1.

The directory sector address is obtained from the block address by the following formula:

Sector address = (block *14) mod #S/T
where #S/T is the number of sectors per track. Directory blocks are 128 words long.

After the specification entry comes the entries for each file. Each entry is 16 words long. Note that for type 0 file entries, words 3 through 7 are different.

Word	Contents
0	NAME 1,2
1	NAME 3,4
2	NAME 5,6
3	file type
4	Track
5	Extent/sector

Table A-2. File Directory Format (Continued)

Word	Contents
6	Number of sectors in file
7	Record length (type 2 files only)
8	Security code
9	↑
10	↑
11	↑
12	Open Flags
13	↓
14	↓
15	↓

For type 0 files, entries 3 through 7 are:

3	0 (type)
4	Logical unit number
5	End-of-file number
Specified at Creation	01XX for MT 10XX for paper tape 11XX for line printer

NOTE

XX = logical unit number

6	Spacing legal code Bit 15 = 1 backspace legal Bit 0 = 1 forward space legal
7	READ/WRITE Code Bit 15 = 1 input legal Bit 0 = 1 output legal

If word 1 = 0 then the end of directory.

If word 1 = -1 then the entry was purged.

Up to seven PGM'S may have the same file open at any given time. The open flags are either zero (not open), or are the ID address of the program the file is opened to. An exclusive open is indicated by the sign being set on the only open flag.

Each time a file is opened, the ID tables for each program that already has the file open are checked to see if the program is dormant. If dormant, i.e., if the point of suspension is zero, the open flag will be set to zero. This does not close the DCB nor post a possibly unwritten record therein.

Table A-3. Data Control Block Format

The Data Control Block (DCB) is a 144-word array provided by the user program with one DCB required for each file opened. The DCB is used to:

1. Prevent unnecessary directory accesses (usually an access is not needed once a file is opened).
2. Keep track of current position within file.
3. Provide a packing/unpacking buffer.

Word	Contents
0	Track/LU } Address of directory entry for the file
1	Offset Sector } Address of directory entry for the file
2	File type (may be overridden at open)
3	Track
4	Sector
5	Number of sectors in file
6	Record length (type 2 files only)
7	Security code and open mode

Bit	Meaning
15 = 1	Codes agree
15 = 0	Codes do not agree
0 = 1	Update open
0 = 0	Standard open

Word	Contents
8	Number of sectors per track
9	Open – Close flag open = ID segment address close ≠ ID segment address
10	Track } Current location in file
11	Sector } Current location in file
12	Location of next word } Current location in file

Table A-3. Data Control Block Format (Continued)

Word	Contents
13	In buffer and write flag:

Bit	Meaning
15 = 0	Not in core
15 = 1	In core
0 = 1	Written on
0 = 0	Not written on
14 = 1	If EOF read
14 = 0	If EOF not read

Word	Contents
14	Record count
15	Extent number
16	Buffer 128 words
⋮	
143	

For type 0 files, entries 3 through 7 are:

3	0 (type)
4	Logical unit number
5	End-of-file code
Specified at Creation	01XX for MT 10XX for paper tape 11XX for line printer

NOTE

XX = logical unit number

6	Spacing legal code Bit 15 = 1 backspace legal Bit 0 = 1 forward space legal
7	READ/WRITE Code Bit 15 = 1 input legal Bit 0 = 1 output legal

The DCB is free for other use after a PURGE, CLOSE and NAMF call. Once a file is open, the DCB is used to reference the file, the name no longer being needed or used.

Table A-4. Record Format For Disc Files

FIXED RECORD LENGTH FILES

Type 1 and 2 files will be written as presented. They will be packed and may cross sector and track boundaries.

RANDOM LENGTH FILES

Type 3 and above records will be preceded and followed by a length word which contains the length of the record exclusive of the two length words. A zero length record will consist of two zero words. An end of file will be indicated by a -1 for the length.

SAVE PROGRAM FILES

Save program files are created by the SP command as type 6 files; however, they will always be accessed as type 1.

The first two sectors of the file will be used to record ID information on the program as follows:

Word	Contents
0	-1 (EOF if not 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 – 21	Time parameters
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 through 27.
29	System setup code word (same as disc directory word 125)
30 – 127	Not used

The rest of the file will be an exact copy of the original program tracks.

APPENDIX B
SUMMARY OF FMP ERROR PRINTOUTS

FMGR ERROR CODES

ERROR	MEANING
-17	ILLEGAL READ/WRITE ON TYPE Ø FILE
-16	ILLEGAL TYPE OR SIZE=Ø
-15	ILLEGAL NAME
-14	DIRECTORY FULL
-13	DISC LOCKED
-12	EOF OR SOF ERROR
-11	DCB NOT OPEN
-10	NOT ENOUGH PARAMETERS
-09	ATTEMPT TO USE APOSN OR FORCE TO 1 A TYPE Ø FILE
-08	FILE OPEN OR LOCK REJECTED
-07	BAD FILE SECURITY CODE
-06	CR OR FILE NOT FOUND OR NO ROOM
-05	RECORD LENGTH ILLEGAL
-04	MORE THAN 32786 RECORDS IN A TYPE 2 FILE
-03	BACKSPACE ILLEGAL
-02	DUPLICATE FILE NAME
-01	DISC ERROR
ØØØ	NO ERROR
ØØ1	DISC ERROR-LU REPORTED
ØØ2	INITILIZE LU 2!
ØØ3	INITILIZE LU 3!
ØØ4	ILLEGAL RESPONSE TO ØØ2 OR ØØ3
ØØ5	REQUIRED TRACK NOT AVAILABLE - RELATIVE TAT POSITION REPORTED
ØØ6	FMGR SUSPENDED
ØØ7	CHECKSUM ERROR
ØØ8	D.RTR NOT LOADED
ØØ9	ID-SEGMENT NOT FOUND
Ø1Ø	INPUT ERROR
Ø11	DO OF,XXXXX,8 ON NAMED PROGRAMS
Ø12	DUPLICATE DISC LABEL OR LU
Ø13	TR STACK OVERFLOW
Ø14	REQUIRED ID-SEGMENT NOT FOUND
Ø15	LS TRACK REPORT
Ø16	FILE MUST BE AND IS NOT ON LU 2 OR 3
Ø17	ID SEGMENT NOT SET UP BY RP
Ø18	PROGRAM NOT DORMANT
Ø19	FILE NOT SET UP BY SP ON CURRENT SYSTEM
Ø2Ø	ILLEGAL TYPE Ø LU
Ø21	ILLEGAL DISC SPECIFIED
Ø22	COPY TERMINATED
Ø23	DUPLICATE PROGRAM NAME.
Ø5Ø	NOT ENOUGH PARAMETERS
Ø51	ILLEGAL MASTER SECURITY CODE
Ø52	ILLEGAL LU IN RESPONSE TO ØØ2 OR 3
Ø53	ILLEGAL LABEL OR ILABEL
Ø54	DISC NOT MOUNTED
Ø55	MISSING PARAMETER
Ø56	BAD PARAMETER
Ø57	BAD TRACK NOT IN FILE AREA
Ø58	LG AREA EMPTY
Ø59	REPORTED TRACK UNAVAILABLE
Ø6Ø	DO YOU REALLY WANT TO PURGE THIS DISC? (YES OR NO).

APPENDIX C
SUMMARY OF FMP CALLS

Consult Section II for the complete details on each FMP call. The general format of an FMP call in Assembly language is:

```
        EXT   name    Declaration of subroutine.
         :
        JSB   name    Transfer control to name.
        DEF   RTN     Return address.
        DEF   IDCB    144 word DCB buffer address.
        DEF   IERR    Error code buffer address.
        DEF   p1      Define addresses
         :
        DEF   pn      of parameters.

RTN     return point
         :
IDCB    BSS 144       144 word DCB buffer.
IERR    BSS 1         Error code returned here.
p1      ---           Actual parameter
pn      ---           values.
```

For each FMP call (except FSTAT and RWNDF), this appendix includes only the parameters *p1* through *pn*.

APOSN Sets address of next record.

IREC	DEC	Record number of next record.
IRB	DEC	Relative block address of next record.
IOFF	DEC	Block offset of next record.

CLOSE Closes DCB.

ITRUN	DEC	+*n* = number of blocks to be deleted from the end of the file when it is closed. −*n* = retain main file, delete extents. *n* = 0 = standard close.

CREAT Creates a file.

NAME	ASC	File name.
ISIZE	DEC	Number of blocks in the file. If negative, use all of disc (see Comments).
	DEC	Record length (used for type 2 file only).
ITYPE	DEC	File type (see Comments).
ISECU	DEC	Security code.
ICR	DEC	Cartridge Label.

FCONT RTE I/O control request.

ICON1		See Section II for control information.
ICON2		Function code - see Section II.

FSTAT Return cartridge status.

ISTAT	BSS	125	Buffer of 125 words.

LOCF Return information on DCB.

IREC	BSS	Next record number.
IRB	BSS	Relative block of next read.
IOFF	BSS	Block offset of next record.
JSEC	BSS	Number of sectors in the file.
JLU	BSS	File logical unit.
JTY	BSS	File type.
JREC	BSS	Record size.

NAMF Renames specified file.

NAME	ASC	File's present name.
NNAME	ASC	File's new name.
ISECU	DEC	Security code.
ICR	DEC	Cartridge label.

OPEN		Opens the named file.
NAME	ASC	File name.
IOPTN	OCT	Open options (see Section II).
ISECU	DEC	Security code.
ICR	DEC	Cartridge label.
POSNT		Positions next access to specific record.
NUR	DEC	New record number.
IR	DEC	Absolute vs. relative control parameter for NUR.
PURGE		Purge named file.
NAME	ASC	File name.
ISECU	DEC	Security code.
ICR	DEC	Cartridge label.
READF		Reads a record from open file.
IBUF	BSS	Data buffer.
IL	DEC	Read request length.
LEN	BSS	Actual read length returned here.
NUM	DEC	Record number to be read.
RWNDF		Rewind type 0 file/reset disc file.
No optional parameters.		
WRITF		Write a record to an open file.
IBUF	BSS	Data buffer.
IL	DEC	Write request length.
NUM	DEC	Record number to be written.

APPENDIX D
SUMMARY OF OPERATOR REQUESTS

namr = file name [:*security code* [:*label* [:*file type* [:*file size* [:*record size*]]]]]

ON, FMGR [,*input* [,*log* [,*list* [,*severity code*]]]]	Schedule operator interface FMGR.
CL	List active cartridge labels.
CO, *label1*, *label2*	Copy files from *label1* to *label2*.
CR, *namr*	Create file name.
CR, *namr*, *lu* [,*REad* | ,*WRite* | ,*BOth*] [,*BSpace* | ,*FSpace* | ,*BOth*] [,*EOf* | ,*LEader* | ,*PAge* | ,*numeric*] [,*BInary* | ,*AScii* | ,*numeric*]	Create type 0 file.
DC, *label*	Dismount cartridge.
DL [,*label* [,*master security code*]]	List contents of file directory.
DU, *namr1*, *namr2* [,*record format*, *EOF control* [,*file #* [,*#files*]]]	Transfer contents of one file to another file (does not create *namr2*).
— or —	
DU, *namr1*, *namr2* [,*record format* [,*file #* [,*#files*]]] [,*EOF control*]	
IN, [*master security code*] ,*label1*, *label2*, *id* [,*lst trk* [,*#dir trks* [,*#sec/trk* [,*bad tracks*]]]]	Initialize cartridge parameters .
IN, *master security code* - - *new security code*	Change master security code.
LI, *namr* [,*Source* | ,*Binary* | ,*Directory*]	Print file contents.
LL, *namr*	Change assignment of list file.
LO, *lu*	Change assignment of log device.
MC, *lu* [,*last track*]	Mount cartridge.
MR, *namr*	Transfer relocatable file to load-and-go area.
MS, *namr* [,*prog name* [,*IH*]]	Transfer source file to EDIT or prog name.
PK [,*label*]	Pack a cartridge to recover purged areas.
PU, *namr*	Purge a file and its extents from system.

RN,*namr, nuname*	Change the name of a file.
RP,*namr*	Restore program saved with SP Command.
RP,*namr, program*	Assign program's ID segment to *namr*.
RP, ,*program*	Release program's ID segment.
SP,*namr*	Save a disc resident program.
SA, LS,*namr*	Save logical source area.
SA, LG,*namr*	Save load-and-go area.
ST,*namr1, namr2* [,*record format, EOF control* [,*file #* [,*#files*]]] – or – ST,*namr1, namr2* [,*record format* [,*file #* [,*#files*]]] [,*EOF control*	Store contents of one file to another file (creates *namr2*).
SV,*number*	Change system log severity code.
TR [,*namr* ,*-integer*]	Transfer control.
?? [,*number*]	Expand error message.

APPENDIX E
HP CHARACTER SET

RTE SPECIAL CHARACTERS

MNEMONIC	VALUE	USE
SOM (CONTROL A)	1	BACKSPACE
S1 (2600 BACKSPACE) (CONTROL Y)	31	BACKSPACE
EOT (CONTROL D)	4	SIMULATE END TAPE

b_7				0	0	0	0	1	1	1	1
b_6				0	0	1	1	0	0	1	1
b_5				0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1								
0	0	0	0	NULL	DC_0	b	0	@	P		
0	0	0	1	SOM	DC_1	!	1	A	Q		
0	0	1	0	EOA	DC_2	"	2	B	R		U
0	0	1	1	EOM	DC_3	#	3	C	S		N
0	1	0	0	EOT	DC_4 (STOP)	$	4	D	T		A
0	1	0	1	WRU	ERR	%	5	E	U	UNASSIGNED	S
0	1	1	0	RU	SYNC	&	6	F	V		I
0	1	1	1	BELL	LEM	(APOS)	7	G	W		G
1	0	0	0	FE_0	S_0	(	8	H	X		N
1	0	0	1	HT / SK	S_1	)	9	I	Y		E
1	0	1	0	LF	S_2	*	:	J	Z		D
1	0	1	1	V_{TAB}	S_3	+	;	K	[		
1	1	0	0	FF	S_4	(COMMA)	<	L	\		ACK
1	1	0	1	CR	S_5	-	=	M	]		①
1	1	1	0	SO	S_6	.	>	N	↑		ESC
1	1	1	1	SI	S_7	/	?	O	←		DEL

Standard 7-bit set code positional order and notation are shown below with b_7 the high-order and b_1 the low-order, bit position.

	b_7	b_6	b_5	b_4	b_3	b_2	b_1
EXAMPLE: The code for "R" is:	1	0	1	0	0	1	0

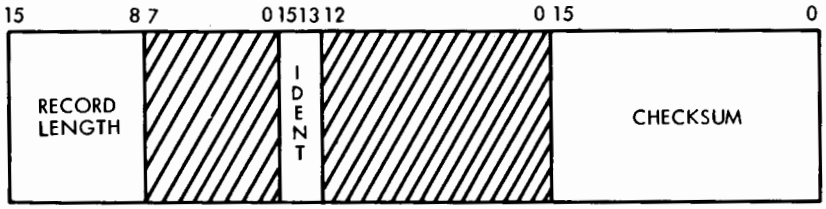
LEGEND

NULL	Null/Idle
SOM	Start of message
EOA	End of address
EOM	End of message
EOT	End of transmission
WRU	"Who are you?"
RU	"Are you...?"
BELL	Audible signol
FE_0	Format effector
HT_0	Horizontal tabulation
SK	Skip (punched card)
LF	Line feed
V_{TAB}	Vertical tabulation
FF	Form feed
CR	Carriage return
SO	Shift out
SI	Shift in
DC_0	Device control reserved for data link escape
DC_1-DC_3	Device Control
DC_4(Stop)	Device control (stop)
ERR	Error
SYNC	Synchronous idle
LEM	Logical end of media
S_0-S_7	Separator (information)
b	Word separator (space, normolly non-printing)
<	Less than
>	Greater than
↑	Up arrow (Exponentiation)
←	Left arrow (Implies/Replaced by)
\	Reverse slant
ACK	Acknowledge
①	Unassigned control
ESC	Escape
DEL	Delete/Idle

APPENDIX F
RELOCATABLE TAPE FORMAT

NAM RECORD

CONTENT

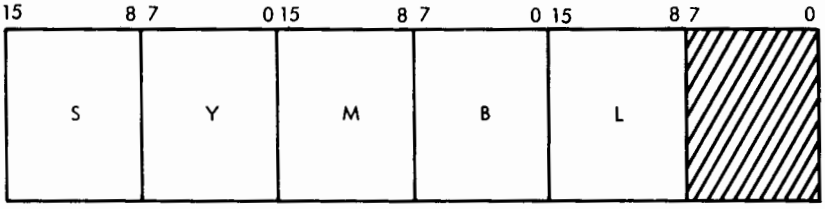


EXPLANATION

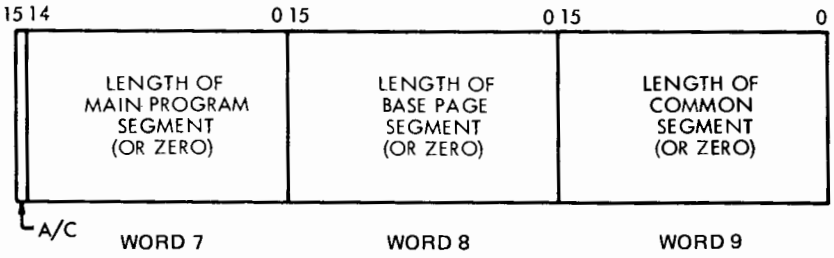
RECORD LENGTH = 17 WORDS

IDENT = 001

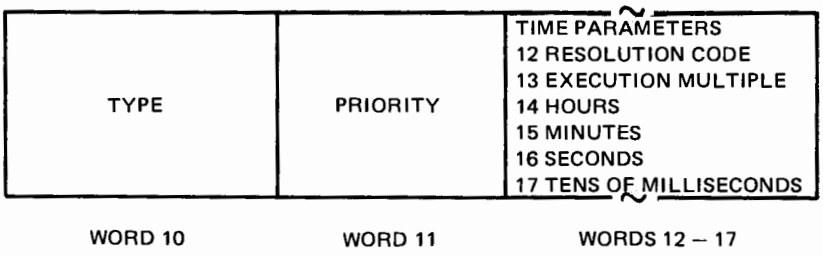
CHECKSUM: ARITHMETIC TOTAL OF ALL WORDS IN RECORD EXCLUDING WORDS 1 AND 3.



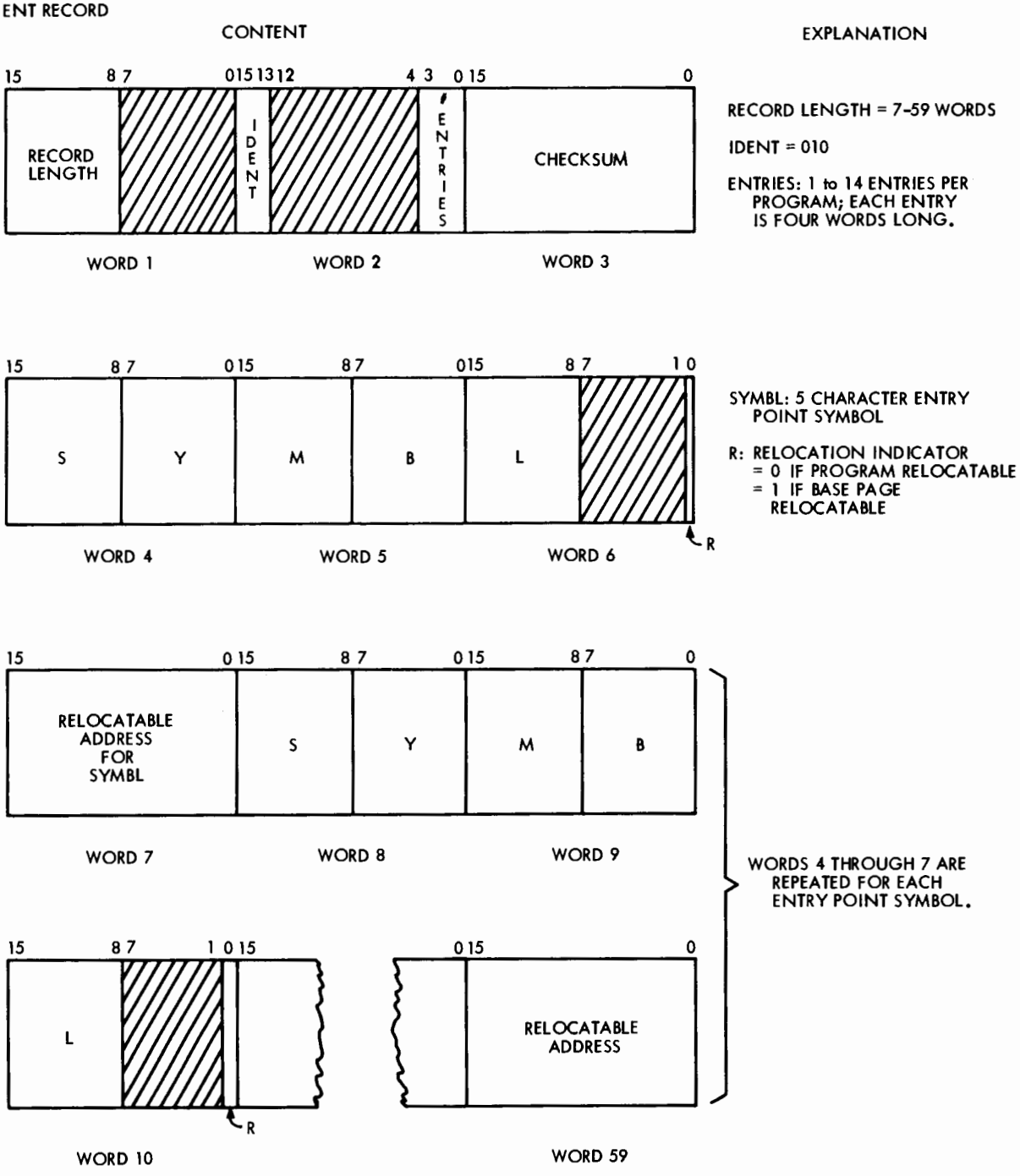
SYMBL: FIVE CHARACTER NAME OF PROGRAM



A/C: BINARY TAPE PROCESSOR
= 0 IF ASSEMBLER PRODUCED
= 1 IF COMPILER PRODUCED



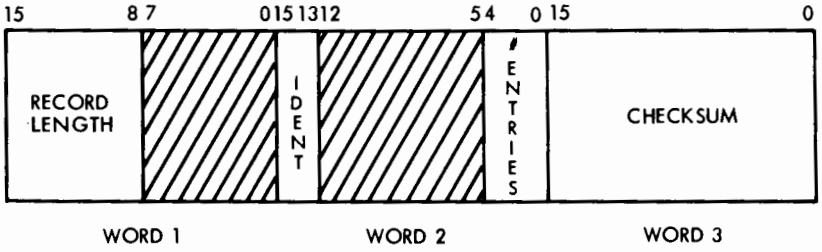
ENT RECORD


CONTENT
EXPLANATION
15
8 7
0 15 13 12
4 3
0 15
0
RECORD LENGTH
IDENT
ENTRIES
CHECKSUM
WORD 1
WORD 2
WORD 3
RECORD LENGTH = 7-59 WORDS
IDENT = 010
ENTRIES: 1 to 14 ENTRIES PER PROGRAM; EACH ENTRY IS FOUR WORDS LONG.
15
8 7
0 15
8 7
0 15
8 7
1 0
S
Y
M
B
L
R
WORD 4
WORD 5
WORD 6
SYMBL: 5 CHARACTER ENTRY POINT SYMBOL
R: RELOCATION INDICATOR
= 0 IF PROGRAM RELOCATABLE
= 1 IF BASE PAGE RELOCATABLE
15
0 15
8 7
0 15
8 7
0
RELOCATABLE ADDRESS FOR SYMBL
S
Y
M
B
WORD 7
WORD 8
WORD 9
WORDS 4 THROUGH 7 ARE REPEATED FOR EACH ENTRY POINT SYMBOL.
15
8 7
1 0 15
L
R
WORD 10
0 15
0
RELOCATABLE ADDRESS
WORD 59

EXT RECORD

CONTENT

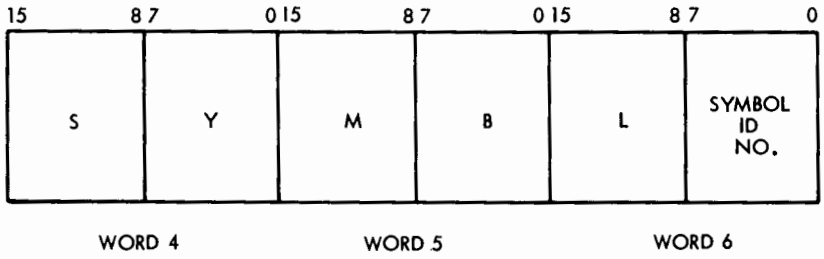
EXPLANATION



RECORD LENGTH = 6-60 WORDS

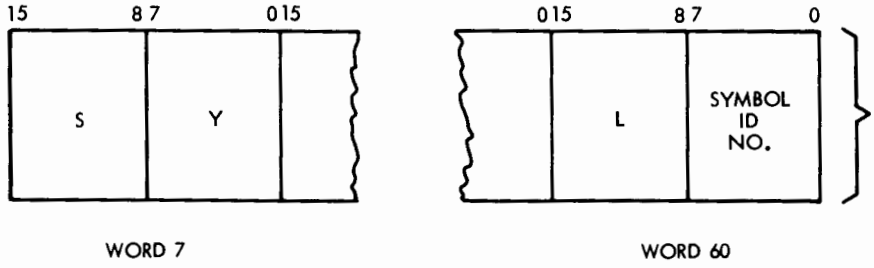
IDENT = 100

ENTRIES: 1 TO 19 PER RECORD; EACH ENTRY IS THREE WORDS LONG



SYMBL: 5 CHARACTER EXTERNAL SYMBOL

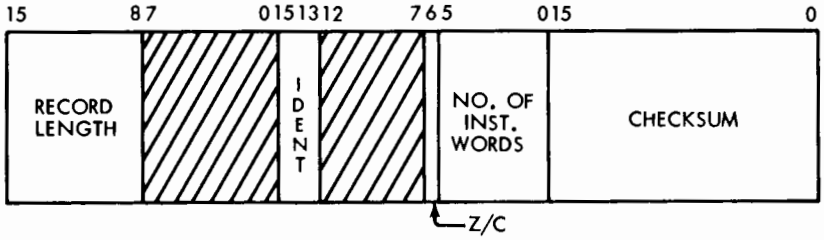
SYMBOL ID. NO.: NUMBER ASSIGNED TO SYMBL FOR USE IN LOCATING REFERENCE IN BODY OF PROGRAM.



WORDS 4 THROUGH 6 REPEATED FOR EACH EXTERNAL SYMBOL (MAXIMUM OF 19 PER RECORD).

DBL RECORD

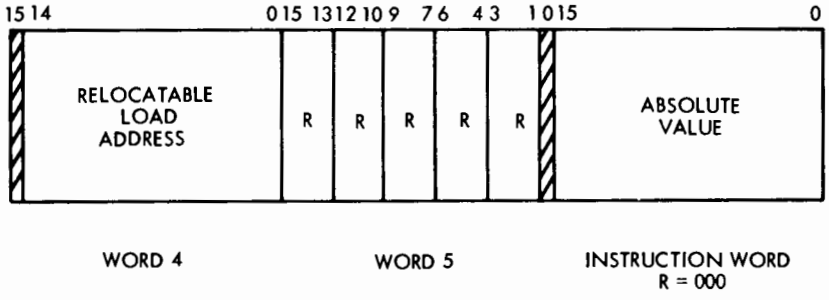
CONTENT



EXPLANATION

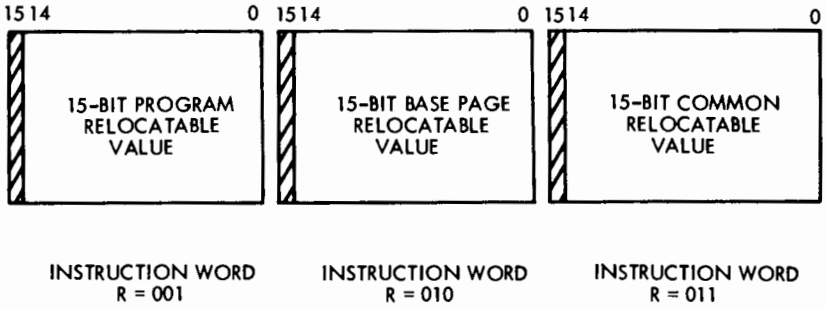
RECORD LENGTH = 5-60 WORDS
IDENT = 011
Z/C: BASE/CURRENT PAGE LOADING
= 0 FOR BASE PAGE
= 1 FOR CURRENT PAGE

NO. OF INST. WORDS: 1 TO 45 LOADABLE INSTRUCTION WORDS PER RECORD



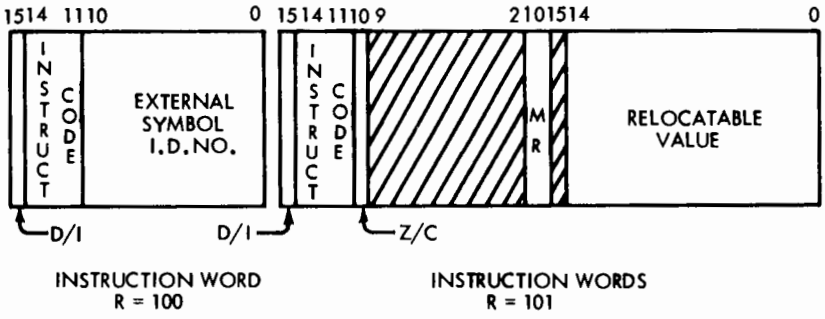
RELOCATABLE LOAD ADDRESS: STARTING ADDRESS FOR LOADING THE INSTRUCTIONS WHICH FOLLOW.

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



R_1 IS RELOCATION INDICATOR FOR INSTRUCTION $WORD_1$; R_2, FOR INSTRUCTION $WORD_2$; ETC-MEMORY REFERENCE INSTRUCTIONS USE TWO WORDS, WITHIN THE TWO-WORD GROUP, "MR" INDICATES RELOCATABILITY OF OPERAND SPECIFIED IN SECOND WORD:

00 = PROGRAM RELOCATABLE
01 = BASE PAGE RELOCATABLE
10 = COMMON RELOCATABLE



D/I: INDIRECT ADDRESSING

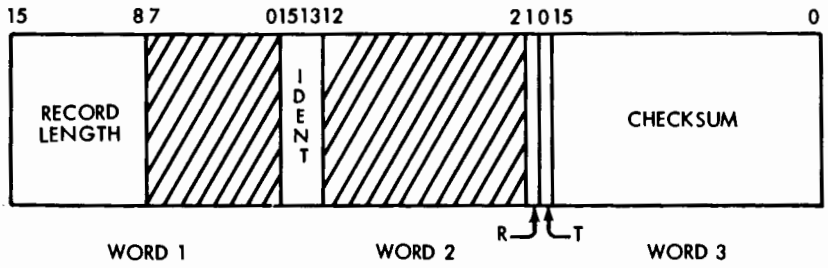
0 = DIRECT
1 = INDIRECT

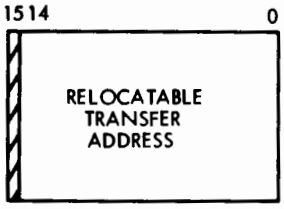
Z/C: BASE/CURRENT PAGE LOCATION OF OPERAND ADDRESS AS DETERMINED BY LOADER.

0 = BASE PAGE
1 = CURRENT PAGE

END RECORD

CONTENT





EXPLANATION

RECORD LENGTH = 4 WORDS

IDENT = 101

R: RELOCATION INDICATOR FOR TRANSFER ADDRESS

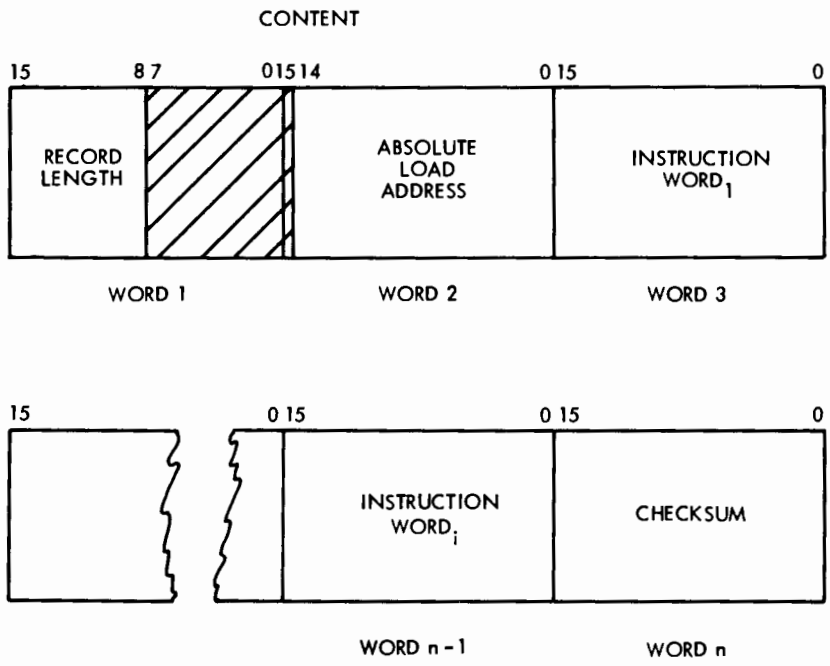
= 0 IF PROGRAM RELOCATABLE
= 1 IF BASE PAGE RELOCATABLE

T: TRANSFER ADDRESS INDICATOR

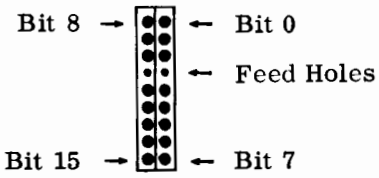
= 0 IF NO TRANSFER ADDRESS IN RECORD

= 1 IF TRANSFER ADDRESS PRESENT

APPENDIX G
ABSOLUTE TAPE FORMAT



†Each word represents two frames arranged as follows:



APPENDIX H
THE BASIC PRINCIPLES OF INDEXING

BASIC PRINCIPLES

There are six basic principles which apply whether one is dealing with a collection of complex programs, or thousands of technical articles reduced to abstract form.

1. Balance input and output effort.
2. Evaluate single entry and multiple entry files.
3. Describe items fully.
4. Control the vocabulary.
5. Know the subject.
6. Select appropriate index scheme.

Consider each point and see how they tie together to produce an effective index and retrieval system, but first, here are some necessary definitions.

DEFINITIONS

Data are numeric or quantitative notations. Data concepts are mutually exclusive. Data are easily manipulated by machines. The boiling point of lead is 1620°C. That is data.

Information is knowledge concerned principally with qualitative concepts or ideas. Information concepts are not mutually exclusive; the concepts interact and overlap. Information is not easily manipulated by machines as data. Information includes data. A qualitative discussion of ablation of plastics in heat shields of space vehicles would be information. Data are relatively easy to handle; information is more difficult.

Records are reports, abstracts, books, articles and catalogs.

Terms are used to represent concepts. In various systems they may be called descriptors, key words, uniterms, or subject headings. Terms identify the elements of information present in an item. Terms are stored as references to information.

There are three distinct levels of retrieval.

Reference Retrieval results when a search provides a list of item numbers or titles as the result. The inquirer gets the items and finds the answers.

Record Retrieval results when a search provides the documents or articles themselves as the results. The inquirer looks through the records for the answers.

Information Retrieval results when a search provides information as the direct answer to the question. The inquirer reviews the material to see if it satisfies his request. The same would be true for data retrieval.

Although there is much talk about information retrieval, in most cases it is reference or item retrieval.

BALANCE THE INPUT AND OUTPUT EFFORT

The usual approach is to spend a minimum of effort on organizing an index. When items are needed, the work begins. Long, frustrating, sequential searches are made through the index looking for specific items to fill an immediate need. Study the economics of the situation. Look at both input and output. Pause for a minute and analyze how large your index is and how much it is expected to grow each year. Assume you have 1000 items which can be filed in chronological order. Each time material is needed it might be necessary to search these 1000 documents by going through the index serially from the first to the last. If many searches are made, a comprehensive index to the collection may be justified. However, if a search is made only once a year, or less, the cost of installing a detailed system could probably not be justified. Therefore, consider how many items are in the file, how many more will be coming into the collection, and how often these will be searched.

EVALUATE SINGLE ENTRY AND MULTIPLE ENTRY FILES

Before you store, consider how you will search. An insurance company keeps its records by account. All records on Charles Walker are kept in one file. The request usually is, "Bring me the file on Charles Walker." Since only a single entry to the file is required, it can be kept in an ordered arrangement by a unique name. The same is true in a store that keeps its credit accounts by customers. Here files are kept in a single entry arrangement since this is the way the file will be used.

A single entry system can be used in chemistry for a file on physical properties of chemical compounds, if the only questions asked of the file are to supply the physical properties of specific compounds. In this case, the most logical

filing order is by compound name. In science and technology, however, this last situation does not usually occur. After a period of time of filing physical properties under the names of chemical compounds, along comes the problem in which someone needs to know the names of all chemical compounds with a certain physical property value. Now the file must be approached from a new viewpoint; there is a need to know both the physical properties of a given compound, and all compounds with a given physical property. The need for multiple entry to collections is common in science and technology and most other areas. A body of knowledge is usually approached from more than one direction. When an engineer requires the test results from a single instrument for a certain event, retrieval may be difficult if the results were filed only under the event name.

Early in your work decide whether your index can be managed with a single entry approach or whether it requires multiple entry search flexibility.

DESCRIBE ITEMS FULLY

If you are going to have a single entry system, pick the proper entry characteristic and set up your index. More likely you will need multiple entry and more effort is required.

Traditional systems describe an item by perhaps one to four terms. This provides a limited number of ways of finding an item. More often it is desirable and necessary to index an item by perhaps ten, fifteen, or more terms in order to provide the proper degree of flexibility in finding what you want quickly and economically.

Full description of the item is needed because most collections require multiple entry. If you are going to be able to answer a variety of questions, then you need to provide for this search flexibility. This is done at input by describing each item fully, by indexing the item in anticipation of all the probable questions. The key questions to ask yourself are these: "In what ways am I apt to want this item in the future? What questions am I apt to ask for which I would like this returned?" Thinking about future use will help you describe the item properly so that you can find it easily later.

CONTROL THE VOCABULARY

The language for an information system is one of the most important features, yet it is often overlooked. Terms must be selected which bridge the communications gap from the language used by the originator of the item, to the person who stores it away, and then later, to the terms used in searching the index. The terms must be organized to take care of cross references and to provide consistency of input and search. However, the language must be flexible enough so that modifications and extensions can be handled easily. A modest amount of vocabulary control can assure that information is not lost because ideas are referenced in one way and searched for in another. Before setting up a system think about the concepts which cover the area of interest and reflect on the kinds of questions that are apt to be asked of the collection. Some thought at this point and some setting down of these terms will be a major assist in organizing the vocabulary for efficient operation.

KNOW THE SUBJECT

The best qualified person to describe a collection is one who is intimately familiar with the area covered. The organization of a system is done best by the person who will be using it. Finally, the best system will be the one installed and operated by someone knowledgeable in the subject area covered. Most efficient information systems in chemistry, for example, are operated by chemists. Often a system is turned over for indexing to a secretary or a clerk who may not have sufficient subject competence to describe an item well enough so that questions can be handled effectively.

SELECT APPROPRIATE INDEX SCHEME

Some type of index scheme is needed to provide the multiple entry. The FMP has two directories of its own; the main directory (disc directory) which contains information about all the area available to the FMP and the local directory (file directory) which contains information on each file located on the disc or "file drawer." A supplementary index should be formed as a master list for all files and records, their security codes, and for cross referencing purposes.

Remember,

> "Those who cannot remember the past are condemned to repeat it." – *Santayana.*

APPENDIX I

FMP CALLS IN FORTRAN

In FORTRAN and FORTRAN IV the FMP call consists of a CALL statement and a series of assignment statements defining the variable parameters of the call.

DIMENSION (IDCB (144)
CALL *name* (IDCB, *p2* . . . , *pn*)

Where:

IDCB is the Data Control Block, and *p2* through *pn* are either integer values or integer variables defined elsewhere in the program. The underlined parameters are optional.

APOSN

Purpose:

This routine sets the address of the next record for any file except type 0.

DIMENSION IDCB (144)
IREC = *a* Record number of next record.
IRB = *b* Relative block address of next record (optional).
IOFF = *c* Block offset of next record (optional).
CALL APOSN (IDCB, IERR, IREC, IRB, IOFF)
Error code is returned in IERR.

CLOSE

Purpose:

This routine closes the DCB and makes the file available to other callers. CLOSE also optionally truncates the file size.

DIMENSION IDCB (144)
ITRUN = *n* +*n* = number of blocks to be deleted from the end of the file when it is closed.

-*n* = retain main file, delete extents.

n = Ø = standard close.

CALL CLOSE (IDCB, IERR, ITRUN)
Error code is returned in IERR.

CREAT

Purpose:

This routine closes the DCB (if open), and then creates the named file on the specified disc with the specified number of blocks. The file is left open exclusively to the caller on successful completion of the call. This call will not create a type 0 file.

DIMENSION IDCB (144), NAME(3), ISIZE(2)

NAME(1)	= *xxxxx*B	First two characters
NAME(2)	= *xxxxx*B	Second two
NAME(3)	= *xxxxx*B	Last two characters
ISIZE(1)	= *a*	Number of blocks in the file. If negative, use all of disc.
ISIZE(2)	= *b*	Record length (used for type 2 file only).
ITYPE	= *c*	File type.
ISECU	= *d*	Security code (optional).
ICR	= *e*	Cartridge Label (optional).

CALL CREAT (IDCB, IERR, NAME, ISIZE, ITYPE, ISECU, ICR)
Error code, or number of sectors, is returned in IERR.

FCONT

Purpose:

This routine sends the standard RTE I/O Control request to type 0 (non-disc) files. It has no effect on other files.

DIMENSION IDCB (144)

ICON1 = *a*B	Control word.
ICON2 = *b*	Control word (optional).

CALL FCONT (IDCB, IERR, ICON1, ICON2)
Error code is returned in IERR.

FSTAT

Purpose:

This routine returns information on all cartridge labels in the system.

DIMENSION ISTAT (125) Status buffer
CALL FSTAT (ISTAT)

LOCF

Purpose:

This routine formats and returns location and status information from the DCB.

DIMENSION IDCB (144)
CALL LOCF (IDCB, IERR, IREC, IRB, IOFF, JSEC, JLU, JTY, JREC)
Error code returned in IERR
Next record number returned in IREC
Relative block of next read returned in IRB
Block offset of next record returned in IOFF
Number of sectors in the file returned in JSEC
File logical unit returned in JLU
File type returned in JTY
Record size returned in JREC

NAMF

Purpose:

This routine closes the DCB (if open) and then renames the specified file.

DIMENSION IDCB (144), NAME(3), NNAME(3)

NAME(1)	= *xxxxx*B	Present name, first two characters
NAME(2)	= *xxxxx*B	Second two
NAME(3)	= *xxxxx*B	Last two characters
NNAME(1)	= *xxxxx*B	New name, first two characters
NNAME(2)	= *xxxxx*B	Second two
NNAME(3)	= *xxxxx*B	Last two characters
ISECU	= *a*	Security code (optional)
ICR	= *b*	Cartridge label (optional)

CALL NAMF (IDCB, IERR, NAME, NNAME, ISECU, ICR)
Error code is returned in IERR

OPEN

Purpose:

This routine closes the DCB (if open), and then opens the named file.

DIMENSION IDCB (144), NAME(3)

NAME(1)	= *xxxxx*B	First two characters
NAME(2)	= *xxxxx*B	Second two
NAME(3)	= *xxxxx*B	Last two characters
IOPTN	= *a*	Open control word
ISECU	= *b*	Security code (optional)
ICR	= *c*	Cartridge label (optional)

CALL OPEN (IDCB, IERR, NAME, IOPTN, ISECU, ICR)
Error code is returned in IERR

INDEX

INDEX (Continued)

MANUAL PART NO.
29033-98000

PRINTED IN U.S.A.