HEWLETT PACKARD

RTE-6/VM Utility Programs

Reference Manual

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 92084-90007
E0693

Printed in U.S.A. June 1993
Fifth Edition

NOTICE

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

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

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

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

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARs 252.227.7013.

Copyright © 1983-1987, 1989, 1990, 1993 by Hewlett-Packard Company

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed that contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine which manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

Second Edition	Dec 1983	
Update 1	Jun 1984	Correct text, add flag, ext utilities
Reprint	Jun 1984	Update 1 incorporated
Update 2	Jan 1985	Manual enhancements
Reprint	Jan 1985	Update 2 incorporated
Update 3	Jan 1986	Manual enhancements
Update 4	Oct 1986	Add FST
Reprint	Oct 1986	Updates 3 and 4 incorporated
Third Edition	Aug 1987	Additions and Corrections for Rev. 5000 (Software Update 5.0)
Fourth Edition	Jan 1989	Rev. 5010 (Software Update 5.1)
Update 1	Jul 1990	Rev. 5020 (Software Update 5.2)
Fifth Edition	Jun 1993	Rev. 6000(Software Update 6.0)

Preface

This manual describes the use of the interactive utility programs available with the Hewlett-Packard RTE-6/VM Operating System. Before attempting to use the utilities, you should be familiar with the RTE-6/VM Operating System. All manuals associated with the operating system are defined in the *RTE-6/VM Index to Operating System Manual*, part number 92084-90001.

File management information in this manual is based on the File Management Package (FMP). Refer to the *RTE-6/VM CI User's Manual*, part number 92084-90036 for details of the Command Interpreter hierarchical file system.

How This Manual Is Organized

Chapter 1	Provides a general introduction to the operating system utilities.
Chapter 2	Describes the general system utilities: WHZAT, LGTAT, LUPRN, SPORT, COMPL, CLOAD, MERGE, OLDRE, SCOM, KEYS, and KYDMP.
Chapter 3	Describes the online, offline physical backup utilities PSAVE, PRSTR, and PCOPY.
Chapter 4	Describes the FST, FC, and TF utilities, used to copy and restore files between tape and disk media, and the LIF utility, which translates files to and from the Logical Interchange Format (LIF) that is interchangeable with other HP systems.
Chapter 5	Describes the WRITT and READT utilities used to back up cartridges to tape.
Chapter 6	Describes the file system utilities provided for use in the CI file system: FSCON, FPACK, MPACK, FREES, FOWN, and FVERI.
Chapter 7	Describes FPORT, the file transport utility used to move files between the HP 1000 and HP 9000 systems. The Application Migration Package (AMP/1000) is required by FPORT when transporting files.
Chapter 8	Describes the use of the PRINT utility.
Chapter 9	Contains information about FORMC and FORMT, the disk formatting utilities for the HP MAC/ICD and CS/80 disks and the HP 9895 Flexible Disk.
Chapter 10	Defines DRREL and DRRPL, the online driver replacement utilities that let you replace device drivers without having to regenerate the system.
Chapter 11	Describes the file analysis utilities FLAG and EXT, used to identify external references and specific character patterns in source files.
Chapter 12	Describes the help lookup utilities: HELP, CMD, GENIX; CALLS and CALLM.
Appendix A	Describes initializing, formatting and sparing ICD/MAC disks.
Appendix B	Describes the CS/80 Exerciser Utility (EXER) used to diagnose and troubleshoot CS/80 disk drives on HP 1000 systems.

Conventions Used in This Manual

The command syntax and other conventions used in this manual are described in the following paragraphs. Sample terminal displays include both user inputs (underlined) and program prompts and messages. Comments are given in parentheses. For example,

```
CI> dl/derick/casey/@.@          (List all files in subdirectory CASEY
                                  under directory DERICK)
```

When you enter parameters in the runstring, do not include blank spaces between parameters or between the utility call and the first parameter. Although blanks are stripped out of the runstring under Session Monitor control, they are significant in a non-session environment. You should, as a standard at all times, avoid entering blanks in the runstring.

Convention	Meaning

Uppercase characters	Capital letters indicate the exact characters required. However, CI accepts lowercase input. For example, the command syntax for the AS command is:
	
	`AS prog <partition number> [C/D]`
	
	and the actual sample entry can be:
	
	`CI> as testprogram 2 c`

[ ]	Optional parameters are shown in square brackets. If you omit a parameter, use a comma as a placeholder before specifying additional parameters. If the parameter is included in the runstring without brackets, it is required and must be supplied.
,	Use a comma to separate commands and parameters. Unless otherwise noted, command parameters must be given in the order shown in the command runstring. If an optional parameter is omitted, use a comma as a placeholder.
oplop2	Multiple options in a runstring must be entered with no preceding, intervening, or trailing characters (including blanks).
lowercase and < >	Lowercase letters represent user-supplied variables. In long descriptive phrases, the variable may be enclosed in angle brackets for clarity, for example, <partition number>.

Conventions for forming the file descriptor and other standard RTE-6/VM conventions are found in the *RTE-6/VM Terminal User's Reference Manual*, part number 92084-90004, and the *RTE-6/VM CI User's Manual*, part number 92084-90036.

Table of Contents

Chapter 1
Introduction

Chapter 2
General System Utilities

Chapter 3
Online/Offline Physical Backup Utilities

Chapter 4
File Copy and File Interchange Utilities

Chapter 5
Cartridge Backup Utilities

Chapter 6
File System Utilities

Chapter 7
File Transport Utility

Chapter 8
PRINT Utility

Chapter 9
Disk Formatting Utilities

Chapter 10
Online Driver Replacement Utilities

Chapter 11
File Analysis Utilities

Chapter 12
Help Lookup Utilities

Appendix A
Initializing, Formatting, and Sparing ICD/MAC Disks

Appendix B
CS/80 Exerciser Utility (EXER)

List of Illustrations

Tables

1

Introduction

This manual details the format and use of the interactive programs available for use with the Hewlett-Packard RTE-6/VM Operating System. Utilities are provided to display system status, manipulate files, save and restore files and LUs between disks and magnetic tape devices, format disks, replace drivers online, and physically back up disks offline. If special loading considerations exist, they are included in the utility descriptions. Where applicable, examples are provided to illustrate the use of the utilities, and error messages are given as part of each utility description.

Operating Environment

The utilities defined in this manual operate in the following environment:

- HP 1000 M/E/F Systems, Models 60 and 65
- RTE-6/VM Operating System

One or more of the utilities support each of the following disks. Note that the referenced device drivers must be generated into the system.

- ICD (Integrated Controller Disk) − HP Models 9895 (Flexible Disk), 7906H, 7920H and 7925H with driver DVA32.
- MAC (Multiple Access Controller Disk) − HP Models 7905, 7906, 7920 and 7925 with driver DVR32.
- CS/80 Disks − HP Models 7907, 7908, 7911, 7912, 7914, 7933, 7935, 7936, 7937, 7941, 7942, 7945, 7946, 7957, 7958, 7959, 7962, 7963, 9122, 9133, 9134, 9153, 9154, C2200, C2202, and C2203 with drivers DVM33/DVN33.
- HP Model 7900 Disk with driver DVR31.
- HP Model 9895 Flexible Disk with driver DVR33.

The utilities can be called under Session Monitor or in a non-session environment. In a non-session environment, you'll be operating under control of the Multi-Terminal Monitor (MTM). Refer to the *RTE-6/VM Terminal User's Reference Manual* for a discussion of the Session Monitor and the MTM.

The utilities covered in this manual differ in their ability to access both FMGR and CI files, as summarized in Table 1-1 and Table 1-2.

Table 1-1. Access to FMGR and CI Files

Utility Name	Accesses FMGR Files	Accesses CI Files
CI	X	X
FMGR	X	
WHZAT	Independent of file system	
LGTAT		
SPORT		
LUPRN		
HELP	X	X
CMD	X	X
GENIX	X	X
COMPL	X	X
CLOAD	X	X
MERGE	X	X
SCOM	X	X
KEYS	X	X
KYDMP	X	X
OLDRE	X	X
FC	X	
TF	X	X
FST	X	X
WRITT	X	
READT	X	
PSAVE	Independent of file system	
PRSTR		
PCOPY		
LIF	X	X
FPORT	X	
FORMT	Independent of file system	
FORMC		
DRREL	X	
DRRPL	X	
FLAG	X	
EXT	X	

Table 1-2. Access to FMGR and CI Files (continued)

Utility Name	Accesses FMGR Files	Accesses CI Files
FPACK		X
MPACK		X
FREES		X
FOWN		X
FVERI		X
CALLS	X	X
CALLM	X	X
PRINT	X	X

2

General System Utilities

The general system utilities described in this chapter let you determine the current status of your system, compile and load programs, manipulate files, and program the function keys (if present on your keyboard).

The WHZAT and LGTAT utilities display the current status of all memory partitions, scheduled and suspended files, and system and auxiliary subchannels. LUPRN supplies current system device configuration and identifies devices by their driver names. SPORT displays the status of serial ports and verifies that a port is set up properly.

The COMPL utility lets you compile a program for subsequent loading, or you can compile and load a program in one operation using CLOAD.

Two file manipulation utilities are described in this chapter. Files can be merged using the MERGE utility and compared for similarities and differences using SCOM.

The KEYS and KYDMP utilities are used in conjunction with display stations that include the function key cluster on the keyboard. You can program the function key cluster, at the upper-right section of the keyboard, to enter a line of ASCII characters or perform a specific command or text formatting function.

Included in this chapter is a discussion of the record reconfiguration utility, OLDRE. This utility lets you translate RTE-6/VM relocatable record formats to the formats used by earlier language processors.

Status Utility (WHZAT)

The WHZAT utility provides current system environment information. By using a specific option, you can output the status of all memory partitions, all scheduled and suspended files, or just those programs associated with your session.

Calling WHZAT

To run WHZAT, either in session mode or system mode, enter the following runstring:

```
:[RU, ]WH[ZAT[], lu [, option]]
```

The lu is the logical unit number of the device on which the status is to be displayed. The default is the log device.

Option defines the type of status to be output. The default is to output those programs associated with your session. The options are specified as:

AL	All scheduled and suspended programs.
SM	Those scheduled and suspended state 3 programs having a father-son relationship.
PA	All partitions in use.
PL|PR[,prog]	Display information on the program specified, up to 5 characters in length. You may use dashes (−) as a wildcard character to match any program name character. For example, "PL,FMG−−" displays information for all programs that begin with FMG. Default is all programs.

Program States

A program can exist in one of seven states relative to the RTE-6/VM Operating System environment:

0	Dormant − the program is not scheduled to run.
1	Scheduled − the program is in the schedule list.
2	I/O suspended − the system is currently performing an I/O operation requested by the program.
3	General wait − the program requested services or resources that are temporarily unavailable or scheduled a second program, or a device is down.
4	Memory suspended − the program requested an operation for which sufficient System Available Memory (SAM) is not currently available.

5 Disk suspended − the program requested the use of more disk space than is currently available.

6 Operator/program suspended − the program is awaiting further operator action or a program EXEC call before it can continue.

For state 2 programs (I/O suspend), the status output contains the message:

```
2,dev:nnn,AV:n,ST:nnn
```

where:

dev:nnn is the EQT or LU number, in decimal.

AV:n is the driver-independent availability code:

0 = available
1 = EQT disabled
2 = busy
3 = waiting for DMA

ST:nnn is the status (in octal) of the EQT. Refer to the associated driver manual for a description of the status codes.

If the EQT or LU is unavailable (down), the code DN is inserted in the message and the status is noted in the summary message at the end of the WHZAT output.

For state 3 programs, a message defining the reason for the general wait suspension is given. Table 2-1 lists all messages, together with their meaning.

For state 4 programs (memory suspend), the status output summary at the end of the report contains the message:

```
MAX CONT.  SAM AVAIL:         nnn
TOTAL SAM AVAIL:              nnn
MAX CONT.  SAM EVER AVAIL:    nnn
```

where:

nnn is an octal number.

The available memory message is output only when a program is in state 4. When sufficient memory is available for program needs, the message is suppressed, and the program is rescheduled.

Table 2-1. General Wait State Messages

Message	Reason for Wait
LULK lu,LKPRG=progx	The listed program attempted to put a lock on logical unit lu. Program progx already has a lock on lu. The listed program will be rescheduled when progx removes the lock.
RN xx,LKPRG=progx	The listed program attempted to set resource number xx. Program progx already has a lock on the resource number. The listed program will be rescheduled when progx removes the lock.
RESOURCE	The listed program attempted to allocate a resource number. The system has no more resource numbers available. The operating system will reschedule the listed program when a resource number is available.
CLASS #	The listed program requested a class number but the system has no more available. The operating system will reschedule the listed program when a class number becomes available.
CL xx	The listed program is waiting on completion of a class GET to class number xx.
progx	The listed program scheduled progx with wait. The listed program will be rescheduled when progx completes.
progx's QUEUE	The listed program scheduled progx on the queue with wait. Progx is not dormant so the listed program must wait. The listed program will be rescheduled after the scheduling of progx completes.
BL,EQT xx	Upper buffer limit exceeded on EQT xx entry. Will be rescheduled when I/O on EQT xx drops below lower buffer limits.
EQLK xxx,LKPRG=PRGA	Program suspended for a locked EQT. Will be rescheduled when EQT is unlocked.
EQLK TABLE FULL	Program attempts to lock EQT and EQT lock table if full. Will be rescheduled when an entry in EQT lock table is released.

WHZAT Output, AL/SM Options

The output for the AL and SM option states is shown in Figure 2-1 and Figure 2-2. Following the system time line (hour, minute, second, millisecond), the output data column codes are given. The codes are defined in Table 2-2 below; if special codes are used in the column entry, they are also defined.

Table 2-2. Output Data Codes (AL, SM Options)

Code	Description
PRGRM	Program name, ** = father program (precedes name)
T	Program type code as documeted in Programmer's Reference Manual; E = EMA program
PRIOR	Program priority (0−32767), B = batch
PT	Partition number (1−64), 0 = memory-resident A = assigned to partition
SZ	Page size of program, ** = memory-resident
	An I after the program size indicates that it is locked in memory for I/O. An L after the program size indicates that it is locked in memory due to an EXEC memory lock request.
DO	Dormant (state 0)
SC	Scheduled (state 1)
IO	I/O suspended (state 2)
WT	General wait state (state 3)
ME	Memory suspended (state 4)
DS	Disk suspended (state 5)
OP	Operator suspended (state 6)
PRG CNTR	Program counter, in octal. Value is listed for all programs; regardless of state, 000000 = program not yet initiated
SWP	Program is swapped out
NEXT TIME	Time at which program will next execute

The AL/SM output is presented in three distinct physical blocks: user session programs, state 3 programs having a father-son relationship, and, for AL output only, all other scheduled and suspended programs.

The sample output in Figure 2-1 contains examples of deadlock situations. In the first example, SON2 has called on SON3 and suspended itself until SON3 terminates. SON3 calls on the suspended SON2, creating a deadlock. In the second example, LOCKB attempted to set a resource previously locked by LOCKA, creating another deadlock situation. Refer to the

RTE-6/VM Programmer's Reference Manual, part number 92084-90005, for the procedures to recover from a deadlock.

```
:WH,AL

 WHZAT REV.5000.861017 @ 22: 0:53:750

  PRGRM  T PRIOR  PT  SZ DO.SC.IO.WT.ME.DS.OP.      .PRG CNTR.    .NEXT TIME
-------------------------------------------------------------------------------
 **FATHR 3 00099  19   5 * * * *  3,SON1   * * * * * P:40107SWP
   SON1  3 00099  21   5 . . . .  3,SON2   . . . . . P:40111
   SON2  3 00099  20   5 . . . .  3,SON3   . . . . . P:40100
   SON3  3 00099  17   5 . . . .  3,FATHR'S QUEUE  . P:40106
 ** BLOCK **
 *********** DEAD LOCK **
 *** SEE ABOVE FOR REPORT ON FATHR
 **FMG31 3 00090  22  11 * * * *  3,LOCKA  * * * * * P:46722
   LOCKA 3 00099  16   5 . . . .  3,LOCKB  . . . . . P:40145SWP
   LOCKB 3 00099   7   5 . . . .  3,RN 059,LKPRG=LOCKA
                                                     P:40067SWP
 ** BLOCK **
 *********** DEAD LOCK **
 *** SEE ABOVE FOR REPORT ON FMG31
 .
   WHZAT 2 00041   7   5 . 1, . . . . . .  P:43610
 .
 **FMG01 3 00090  15  11 * * * *  3,EDI01  * * * * * P:46722
   EDI01 4 00051  19  16 . . . 2,EQ: 7,AV:2,ST:000   P:26102
 .
   R$PN$ 1 00010   0   . . . . .  3,CL 061  . . . .  P45100
   GRPM  1 00004   0   . . . . .  3,CL 060  . . . .  P:54017
   LOGON 3 00045  13  12 . . . .  3,CL 062  . . . .  P:42107
     .   .   .     .       .       .    .       .      .
     .   .   .     .       .       .    .       .      .
     .   .   .     .       .       .    .       .      .
   UPLIN 1 00003   0 . 0,  . . . . . . . . . . . .   P:00000     8: 0:45:70
   RTRY  1 00020   0  . . . . .   3,CL 059  . . . .  P:64632
   DISPL 3 32767  16   4   1, .   .  .  . . . . .  P:40036
-------------------------------------------------------------------------------
 ALL LUS OK
 ALL EQTS OK
-------------------------------------------------------------------------------
 10:36:18:390
```

Figure 2-1. Program Status Mode (AL) Output

The output summary gives the status of the LUs and EQTs (OK, up, down, or locked). The final entry is the time at which WHZAT exited.

```
:wh,sm

WHZAT REV.5000.861017 @ 22: 1: 8:860


 PRGRM  T PRIOR  PT  SZ DO.SC.IO.WT.ME.DS.OP.       .PRG CNTR.   .NEXT TIME.
**FMG72 3 00051  13  12 * * * *  3,WHZ72 * * * * *   P:42131

  WHZ72 2 00041   8   5 .  1.  . . . . . . . . . .   P:40004
.
**FMG65 3 00051  18  12 * * * *  3,EDI65 * * * * *   P:42131
  EDI65 4 00051  22  16 . . . 2,EQ: 13,AV:2,ST:002   P:25553
**FMG68 3 00051  12  12 * * * *  3,LOA68 * * * * *   P:42131
  LOA68 4 00090  25  27I. . . 2,EQ: 16,AV:2,ST:002   P:30023
.
  FMG73 3 00051  14  12 . . . 2,EQ: 21,AV:2,ST:002   P:54701

ALL LUS OK
ALL EQTS OK

8:16:47:700
```

Figure 2-2. Program Status Mode (SM) Output

False Readings

A WHZAT snapshot takes place over a period of time with a significant delay between each line, creating a blurring effect. One side effect is that more than one program may be the father of the same program, or a program scheduled to be run may be listed as being IO suspended, such as program LIN74 in Figure 2-3. WHZAT brackets (>2<) the program it understands to be next in the schedule at the time that particular line is printed.

Also note that WHZAT may list EXEC instead of EQ when it cannot find the program in the IO tables.

Occasionally, the ID segment address of a program to be printed will disappear before WHZAT can capture the program name. In this case, WHZAT prints <???> in place of the 5-letter program name.

Since WHZAT executes dynamically, while the state of the system is changing, a program whose status has been reported could be called by another program. When the status of the second program is reported, the first program status is reported as the son of the calling program, as in the following example:

```
PRGA              *          3,PRGX
PRGX              . 1
  :
PRGB              **         3,PRGX
*** SEE ABOVE FOR REPORT ON FATHR
```

```
:wh,sm
 WHZAT REV.5000.861017 @ 22: 1:15: 50


   PRGRM T PRIOR PT  SZ DO.SC.IO.WT.ME.DS.OP.     .PRG CNTR.   .NEXT TIME.

 **CI.75 6 00051 28  32  * * * * 3,WHZ75 * * * * * P:34314
   WHZ75 3 00010 13   6  . 1,  . . . . . . . . . . P:42670
 .
 **CI.74 6 00051 17  32  * * * * 3,LIN74 * * * * * P:34314
   LIN74 6E00090 24  50  . . >2<EXEC   . . . . . . P:15675
 **CI.73 6 00051 23  32  * * * * 3,EDI73 * * * * * P:34314
   EDI73 6 00051 29  32  . .  2,EQ: 13,AV:2,ST:002 P:24136
 .
   RTRY  1 00020  0  . . . . . . 3,CL  045 . . . . P:54021
   GRPM  1 00004  0  . . . . . . 3,CL  046 . . . . P:63657
   R$PN$ 1 00005  0  . . . . . . 3,CL  047 . . . . P:64256
   UPLIN 1 00003  0  .  0, . . . . . . . . . . . . P:00000    14:26:17: 10
   SPOUT 2 00011  1   2  . . . . 3,CL  029 . . . . P:40220
   LGOFF 3 00061 32  10  . . . . 3,CL  049 . . . . P:41227
   QCLM  3 00028 22   3  . . . . 3,CL  044 . . . . P:40026
   EXECM 3 00030 13   4  . . . . 3,CL  040 . . . . P:43635SWP
   EXECW 3 00030 21   9  . . . . 3,CL  039 . . . . P:41574
   MATIC 3 00030 35   2 0, . . . . . . . . . . . . P:00000    14:26:13: 80
   OPERM 3 00030  7   4  . . . . 3,CL  038 . . . . P:42714SWP
   PROGL 3 00030 15   9  . . . . 3,CL  033 . . . . P:54512
   PTOPM 3 00030 12   9  . . . . 3,CL  041 . . . . P:45656SWP
   RFAM  3 00030  9  10  . . . . 3,CL  036 . . . . P:54303SWP
   DLIST 3 00030  8   5  . . . . 3,CL  037 . . . . P:44441SWP
   RSM   3 00020 33   4  . . . . 3,CL  043 . . . . P:43656
   LOGON 3 00050 34  12  . . . . 3,CL  048 . . . . P:42755
   RDBAM 4 00030 11   6  . . . . 3,CL  034 . . . . P:32046SWP
   TRFAS 3 00030 10   7  . . . . 3,CL  035 . . . . P:46313SWP
   CLEAN 2 00100  3   3 0, . . . . . . . . . . . . P:40176    14:27: 7:860
   ALARM 3 00050 19  11 0, . . . . . . . . . . . . P:41101SWP  0: 0: 0: 00
   CI.77 6 00051 18  32  . .  2,EQ: 17,AV:2,ST:002 P:63011
   CI.85 6 00051 19  32  . .  2,EQ: 25,AV:2,ST:002 P:63011

 ALL LU'S OK
 ALL EQT'S OK

 14:26:14:590
```

Figure 2-3. Program Scheduling Example

WHZAT Output, PA Option

The partition status output provides a dynamic map of the activity in each partition. The output format is shown in Figure 2-4. Following the date line (hour, minute, second, millisecond), the output data column codes are given. These are defined in Table 2-3 below. If special codes are used in the column entry, they are also defined.

Table 2-3. Output Data Codes (PA Option)

Code	Description
PTN#	Partition number (1-64) M = Mother C = Subpartition, Chain mode S = Subpartition available R = Reserved
SIZE	Program page size
PAGES	Physical pages where program resides
BG/RT	Program run type, BG = Background RT = Real time
SHR/LBL	Shareable EMA partition and Label, SH = Shareable ** = Mother with shareable subpartition * = Subpartition of mother
ACT	Number of current users of shareable EMA
L	Partition lock status L = Locked, I = Locked for I/O
PRGRM	Program name
PTN-PRIOR	Partition Priority

In the output, if a mother partition is currently being used for shareable EMA, the PRGRM entry will show EMA as the name of the program occupying the mother partition and subpartitions.

In RTE-6/VM, partition resources are allocated on a priority basis. This resource is allocated by swapping a lower priority program out of memory, making that partition available for a higher priority program. The partition priority aging (AG) feature allows high-priority suspended (state 3) programs to be swapped out in favor of lower priority programs that are scheduled but waiting for a partition. For additional information on partition priority, refer to the section AG (Modify Partition Priority Aging) in Chapter 4 of the *RTE-6/VM Terminal User's Manual*.

```
:WH,PA

 WHZAT REV.5000.861017 @ 11:13:14:440
```

Ptn#	Size	Pages	BG/RT	SHR/LBL	#ACT L	PROGRM	PTN-PRIOR
1	2	77- 79	RT			SPOUT	32767
2	2	80 - 81	RT			LUQUE	25
3	6	82 - 87	RT			IOMAP	50
4	6	88 - 93	RT			SMP	30
5	10	94 - 103	RT			LUMAP	32767
6 R	45	104 - 148	BG			D.RTR	1
7 M	248	149 - 396	BG			<NONE>	
8 S	32	149 - 180	BG			LOGON	70
9 S	32	181 - 212	BG			PTOPM	32767
10 S	32	213 - 244	BG			QCLM	32767
11 S	32	245 - 276	BG			EXECM	32767
12 S	32	277 - 308	BG			EXECW	32767
13 S	32	309 - 372	BG			DSMOD	26
14 S	32	341 - 372	BG			OPERM	32767
15 S	24	373 - 396	BG			DSRTR	30
16 M	150	397 - 546	BG	SH FST2	0	<NONE>	
17 S	100	397 - 496	BG			<NONE>	
18 S	50	497 - 546	BG			<NONE>	
19 M	126	547 - 672	BG	SH FST1	0	<NONE>	
20 S	32	547 - 578	BG	*		DLIST	32767
21 S	32	579 - 611	BG	*		RFAM	32767
22 S	32	611 - 642	BG	*		TRFAS	32767
23 S	30	643 - 672	BG	*		RSM	32767
24	50	673 - 722	BG			DBCOP	50
25	51	723 - 773	BG	SH SHEMA	0	CI.84	51
26	50	844 - 933	BG			D.ERR	1
27	50	824 - 873	BG			CM	32767
28	32	874 - 905	BG			RDBAM	48
29	32	906 - 937	BG			PROGL	32767
30	32	938 - 969	BG			sylog	74
31	32	970 -1001	BG			MATIC	30
32	22	1002 -1023				LGOFF	32767

```
MAXIMUM PARTITION SIZE AVAILABLE
RT       10 PAGES,   BG     100 PAGES, MOTHER   248 PAGES
MAXIMUM PARTITION SIZE AVAILABLE – DUE TO SHAREABLE EMA
RT       10 PAGES,  BG       50 PAGES, MOTHER   248 PAGES

11:13:17:390
```

Figure 2-4. Partition Status Mode (PA) Output

WHZAT Output, PL Option

The active program status output, shown in Figure 2-5 through Figure 2-7, provides a list of all active programs in the system. Following the date line (hour, minute, second, millisecond), the output data column codes are given. These are defined in Table 2-4 below. If special codes appear in the column entry, they are also defined.

Table 2-4. Output Data Codes (PL Options)

Code	Description
NAME	Program name
TY	Program type (see following discussion)
PRIOR	Program priority, 1−32767
LADDR	Low memory address
HADDR	High memory address
LOBP	Low base page
HIBP	High base page
SZ	Program size, in pages
EMA	Extended Memory Area size
MSEG	Memory Segment size, in pages
LBL	Shareable EMA partition label, if used
PTN	Partition number, if assigned to program
TM	Load type TE = Temporary PE = Permanent ME = Memory-resident
COM	System common type SC = System common RC = Reverse common NC = No common
LU	Disk LU on which program is stored
S−ID	Session identifier if program loaded under session monitor

The status output summary identifies the number of free 33-word program ID segments (long), the number of free 9-word program ID segments (short), and the number of 3-word ID extensions available in the system.

You can invoke the PL option to specify a listing of specific program types, as:

```
:[RU,]WH[ZAT],PL,tt
```

where:

tt specifies one of the following program types.

Where both a mnemonic and a number are given, the type may be specified using either entry. (For a detailed description of program types, refer to the *RTE-6/VM Programmer's Reference Manual*).

RT (or 2)	Real-time programs
BG (or 3)	Background programs
LB (or 4)	Large background programs
EB (or 6)	Extended background programs
1	Memory-resident programs (this program type can only be referenced by number)
5	Program segment (this program type can only be referenced by number)

The PL option also can be specified to list only those programs with like characteristics by including a wildcard character (−) in the "don't care" positions of the name, as:

`WH,PL,$-----`	list only those programs whose name begins with $
`RU,WH,PL,F-----`	list only those programs whose name begins with F
`wh,pl,--FTN-`	list only those files with FTN in the third through fifth positions in the name (list all FTN file modules)

```
:wh,pl,5
 WHZAT REV.5000.861017 @ 22: 2:16:960
```

Name	TY	PRIOR	LADDR	HADDR	LOBP	HIBP	SZ	EMA	MSEG	LBL	PTN	TM	COM	LU	S-ID
GASP1	5		44166	53453	107	212						PE		2	
GASP2	5		44166	56355	107	253						PE		2	
FMGR0	5		41561	61165	36	222						PE		2	
FMGR1	5		41561	54721	36	202						PE		2	
FMGR2	5		41561	56506	36	355						PE		2	
FMGR3	5		41561	53407	36	135						PE		2	
FMGR4	5		41561	52606	36	302						PE		2	
FMGR5	5		41561	45536	36	102						PE		2	
FMGR6	5		41561	54524	36	224						PE		2	
FMGR7	5		41561	53553	36	133						PE		2	
FMGR8	5		41561	51120	36	113						PE		2	
FMGR9	5		41561	53262	36	115						PE		2	
FMGRA	5		41561	54053	36	130						PE		2	
FMGRB	5		41561	56066	36	201						PE		2	
XXDR1	5		34754	55726	560	650						PE		2	
XXDR2	5		34754	35765	560	572						PE		2	
XXDR3	5		34754	46255	560	1076						PE		2	
XXDR4	5		34754	40673	560	635						PE		2	
ASMB0	5		34754	34463	363	733						TE		2	
ASMB1	5		34754	26605	363	512						TE		2	
ASMB2	5		34754	26477	363	454						TE		2	
ASMB3	5		34754	25274	363	400						TE		2	
ASMB4	5		34754	25677	363	377						TE		2	
LODR1	5		35050	56063	510	565						PE		3	
LODR2	5		35050	36066	510	515						PE		3	
LODR3	5		35050	46365	510	1017						PE		3	
LODR4	5		35050	40765	510	562						PE		3	
EDIT0	5		26356	43033	516	1100						TE		2	
EDIT4	5		26356	43237	516	770						TE		2	

<29 SHORT BLANK ID>

55 FREE LONG IDS, 29 FREE SHORT IDS, 10 FREE ID EXTS

8:16:30:970

Figure 2-5. Active Program Mode (PL Output, Type 5 Programs Only)

```
:wh,pl,bg

 WHZAT REV.5000.861017 @ 21:42:31:630
```

Name	TY	PRIOR	LADDR	HADDR	LOBP	HIBP	SZ	EMA	MSEG	LBL	PTN	TM	COM	LU	S-ID
, , , , ,	3	99	34000	41406	2	220	4					PE	NC	2	0
GASP	3	80	34000	53655	2	107	11					PE	NC	2	0
LOGON	3	49	34000	56722	2	477	11					PE	NC	2	0
LGOFF	3	52	34000	53335	2	305	9					PE	NC	2	0
FMGR	3	51	34000	61165	2	36	12					PE	NC	2	0
D.RTR	3	1	34000	63504	2	403	13					PE	NC	2	0
CMM6	3	90	34000	60025	2	416	12					PE	NC	2	0
T5IDM	3	40	34000	44735	2	201	6				10	PE	NC	2	0
PTOPM	3	30	34000	50063	2	155	8					PE	SS	2	0
MATIC	3	30	34000	34716	2	17	2					PE	SS	2	0
DINIT	3	26	34000	53272	2	442	9					PE	SS	2	0
DSMOD	3	26	34000	50404	2	265	8					PE	SS	2	0
METER	3	80	34000	52076	2	360	9					PE	NC	2	0
CLEAR	3	99	34000	34611	2	13	2					PE	NC	2	0
HELP	3	90	34000	45475	2	265	6					TE	NC	2	0
FMG65	3	51	34000	61165	2	36	12					TE	NC	2	65
FMG73	3	51	34000	61165	2	36	12					TE	NC	2	73
DRRPL	3	90	34000	66721	2	654	18					TE	NC	21	0
FMG72	3	51	34000	61165	2	36	12					TE	NC	2	72
FMG68	3	51	34000	61165	2	36	12					TE	NC	2	68
WHO	3	45	34000	52476	2	337	9					PE	NC	2	0

```
    55 FREE LONG IDS,        29 FREE SHORT IDS,        10 FREE ID EXTS

8:14:14:770
```

Figure 2-6. Active Program Mode (PL) Output, Background Programs Only

```
:RU,WHZAT,PL,#-----

 WHZAT REV.5000.861017 @ 20:40:17:580
```

Name	T	PRIOR	LADDR	HADDR	LBP	HBP	SZ	EMA	MSEG	LBL	PTN	TM	COM	LU	S-ID
#SEND	1	3	65606	67066	432	465							SS	2	
#DIAL	2	20	42000	42046	2	2	2					PE	NC	2	0

```
    19 FREE LONG IDS,        21 FREE SHORT IDS,        9 FREE ID EXTS

16:35:39:360
```

Figure 2-7. Active Program Mode (PL) Output, #−−−−− Programs Only

Log Track-Assignment Table Utility (LGTAT)

The Log Track-Assignment Table utility (LGTAT) is designed to display information about the system and auxiliary disk subchannel tracks. Using LGTAT, you can specify the LU to which the output will be directed and request either an abbreviated form or a full display of the output.

The abbreviated LGTAT output shows the total number of available tracks and the number of tracks in the largest contiguous track block. In addition to the abbreviated output information, the complete LGTAT output shows the Track Assignment Table for the system and auxiliary disk subchannels and the location of the start of the logical source tracks.

Calling LGTAT

To run LGTAT, enter the following runstring:

```
:[RU,]LGTAT[,lu[,form]]
```

The lu is the logical unit to which LGTAT will direct the output. The default is the log LU.

Form is the output format. The default is to the abbreviated form. The format is specified as:

0 = abbreviated form
1 = complete form

Abbreviated LGTAT Output

If the abbreviated form is specified, LGTAT will output only the following information:

```
TOTAL AVAILABLE TRACKS= xxx
LARGEST CONTIGUOUS TRACK BLOCK= yyy
```

where xxx and yyy are the available tracks and the largest block of tracks, in decimal.

Complete LGTAT Output

If the complete form is specified, the LGTAT output is a complete listing of the track assignment summary for the system disk subchannel and (if present) the auxiliary disk subchannel, the total available tracks, and the largest contiguous block of tracks.

The Track Assignment Table, shown in Figure 2-8, lists all tracks on the associated disk subchannel. Each entry in the table corresponds to one track. Table 2-5 below defines the entries that can appear in the Track Assignment Table.

LGTAT dynamically reports swapped tracks. Large EMA programs that are swapped may not always appear to be contiguous, due to the slow I/O processing relative to the speed of track allocation and deallocation. That is, while LGTAT is reporting the contents of one track, the contents of other tracks may have changed.

A track is identified by the first program contained in that track. LGTAT labels the track as belonging to that program even if other programs also reside on it. LGTAT reports a system track following the system entry point tracks. (This track is currently reserved for system use.) Occasionally, the complete form display will show a system track in the pool area. The system allocates a track as a system track if a track error occurs in the pool area. If necessary, the track can be spared using the FORMT or FORMC utility described in Chapter 9 of this manual.

Table 2-5. Track Assignment Table Entries

Entry	Description
progx	Track belongs to program progx
progx&	Track holds memory-image form of program progx
progx^	Track holds progx, swapped from memory to disk
—	Free track
ENTS	Track holds point list for system and system library
LS	Track holds logical source tracks
SYSTEM	Track is a system track
GLOBAL	Track is a globally allocated track
FMP	Track is part of the file management package (FMP)
LIBRY	Track is a system library track

```
LGTAT,1,1
TRACK ASSIGNMENT TABLE       & =PROG ^ =SWAP

TRACK     0      1      2      3      4      5      6      7      8      9
   0    SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM SYSTEM
  10    SYSTEM SYSTEM SYSTEM SMP  & IOMAP& LUMAP& LUMAP& JOB  & GASP & GASP2&
  20    LOGON& LOGON& LGOFF& LGOFF& CMM6 & CMM6 & FMGR0& FMGR1& FMGR2& FMGR2&
  30    FMGR4& FMGR5& FMGR6& FMGR7& FMGR9& FMGRA& FMGRB& T5IDM& DINIT& DINIT&
  40    DSMOD& EXECM& EXECW& EXECW& PROGL& PTOPM& PTOPM& REMAT& REMAT& RFAM &
  50    DLIST& RSM  & ACCTS& ACCTS& ACCT1& ACCT2& ACCT3& ACCT4& ACCT5& LOADR&
  60    LODR1& LODR1& LODR3& RDBAM& TRFAS& DSRTR& DSRTR& DSRTR& CI   & CI   &
  70    CI   & CI   & CI   & CIX  & CIX  & CIX  & CIX  & CIX  & D.RTR& D.RTR&
  80    D.RTR& D.RTR& LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY
  90    LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY
 100 - 110   20 Tracks same as above.
 120    LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY  LIBRY   ENTS
 130     ENTS  D.RTR  D.RTR    --     --     --     --     --     --     --
 140      --     --     --     --     --     --     --     --     --     --
 150 - 190   50 Tracks same as above.
 200     FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
 210 - 380  180 Tracks same as above.
 390     FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP   D.RTR



AUXILIARY DISK

   0      --     --     --     --     --     --     --     --     --     --
  10 - 190  190 Tracks same as above.
 200     FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP
 210 - 480  280 Tracks same as above.
 490     FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP    FMP   D.RTR

THE LS TRACK(S) ARE UNDEFINED
TOTAL AVAILABLE TRACKS =   267
LARGEST CONTIGUOUS TRACK BLOCK =   200
```

Figure 2-8. Complete LGTAT Output

System Configuration Utility (LUPRN)

LUPRN supplies the current system device configuration and identifies each device by its true driver name. The configuration tables can be directed to your terminal or to a list device and can be ordered by system LU number session LU number (session mode only), or by device select code. LUPRN executes in both session and non-session environments.

LUPRN uses a file of driver names and descriptions called “LUPRN. This file must reside on /SYSTEM or on any globally mounted FMGR disk LU. You must, therefore, add additional memory to LUPRN to hold this file. Two to three pages hold 200 to 350 driver names (nine words per driver name). LUPRN requires %DECAR as a library to search while loading.

Calling LUPRN

To run LUPRN, enter the following runstring:

```
:[RU,]LUPRN[,list_lu[,AL[,LU[,SC[,TY[,DV[,??]]]]]]]
```

where:

list_lu — is the LU of the list output device. The default is to your terminal. While the other options are not order-dependent, this parameter must be the first named option if you want to specify a list device other than your terminal.

AL — lists all device drivers in the system, sorted by system LU number. If the device is included in your Session Switch Table (or available to your MTM station), the device SLU number is listed. If the device is not available for your use, the SLU column is blank for that device.

LU[:n:n] — lists all device drivers for the system LUs, sorted by system LU number. When the optional range is specified, the list includes only those device drivers with LU numbers within the range given by :n:n. Without the range parameters, this form of the command is equivalent to the AL option. LU references are system LUs; session LUs are listed for those LUs contained in your SST.

If only one :n parameter is given, or if the second :n is smaller that the first, only the first specified LU is listed. If no LUs exist within the given range, LUPRN issues the following message:

```
..no LUs found in specified range..
```

as the body of the table.

SC[:n:n] — lists all LUs assigned to specific select codes, sorted by select code number. When the optional range is specified, the list only includes those device drivers within the range given by :n:n. Select codes are by convention expressed as octal numbers, so you should use a B suffix.

As an example, SC:10b:25b will display information for select codes 10 through 25. If no drivers exist within the specified range, LUPRN returns the following message:

```
..no LUs found in specified range..
```

as the body of the table.

TY[:n:n] lists all device drivers for the System LUs, sorted by driver type. When the optional range is specified, the list only includes those drivers that fall within the range given (use the DV option, if necessary, to get the full list of drivers in the system). Drivers are by convention named in octal, so you should use a B suffix when entering driver types. For example, use TY:12B:15B to list all LUs driven by drivers 12 through 15 octal. If the suffix is omitted, the table will list the driver that is equivalent to the decimal number; that is, if you enter the decimal values :12:15, the list will show the equivalent octal drivers, 14B through 17B. As with the LU option, the selected driver list will be taken from the entire system.

DV lists the system driver table, followed by the LU table. This option is useful when you want to see the total number of drivers and their descriptions.

?? lists the descriptive summary of LUPRN and the available optional runstring parameters.

If LUPRN is called with no options, the list will contain your session LUs only, sorted by session LU number. In a non-session environment, calling LUPRN with no options is equivalent to the call LUPRN,AL.

If you specify both TY and LU with search ranges, an output may not result unless the specified combination matches an existing condition of the system. For example, the following runstring:

```
:RU,LUPRN,LU:6,TY:12B
```

will result in an output only if LU 6 is assigned driver type code 12B. However, the SC option will ignore both LU and TY commands, if used.

If you specify an illegal list LU, the following message is issued:

```
..Illegal LU (nn) for output (down or not defined)
```

and LUPRN exits. The erroneous LU number is given in the message.

If you invoke LUPRN with an illegal option code, you will see the message

```
..Unknown command: xx ...use ?? for help.
```

In this case, xx is the erroneous command code you entered.

Output Table Format

The output table headings list the operating system under which LUPRN is running, together with the system's date code. The "sorted by" message indicates Session LU, System LU, or Select Code, depending on the option selected. The Time Base Generator select code is included, or the entry <none> shows. The select code is given if the Privileged Fence card is included in the system, or the entry <none> shows. The number of memory partitions is given, or the entry <none> shows. The following column headings, applicable for the format, are in the configuration table:

SLU	User session LU numbers (this column is included at both the left and right sides of the table). A D suffix indicates that the LU is down. This column is omitted in non-session environments.
LU	System LU numbers (this column is included at both the left and right sides of the table). A D suffix indicates that the LU is down.
EQT	Equipment table entry number.
sc	Subchannel number.
SCD	Select code.
Flags	D = DCPC, Dual Channel Port Controller B = Buffered P = Driver handles power fail S = Driver handles timeout T = Device has timed out.
Av	Device availability: 1 = Device down 2 = Device busy 3 = Device awaiting DCPC If there is no entry in this column, the device is available.
T.OUT	Timeout in seconds. No entry means timeout = 0.
Stats	Device status octal code. Note that the status is for the last driver activity and may not reflect current status. (See the related device driver reference manual for a description of the status codes.)
Driver	Driver name.
DP	First physical page number of the driver partition. If a dollar sign ($) follows the DP entry, it identifies the partition page as the System Driver Area (SDA).
Device Name	True name of device. If possible, subchannel information is also used to identify the device specified in the driver file "LUPRN.

LUPRN Examples

In the following example, LUPRN is invoked with the DV option. The listing is a copy of the "LUPRN file and is followed by the full configuration listing, sorted by system LU number.

```
CI.73> luprn dv
 LUPRN's Driver List


   NUM Name  =    Description

     1 DVR00     CRT/TTY terminal
     2 DVC00     CRT/TTY terminal
     3 DVD00     Logical Driver
     4 DVT00     306C Centronics
     5 DVM00     1279x 8-ch Mux
     6 DVX00     X.25 virt-driver
     7 DVS00     12790 Mux-DVR00
     8 DVP00     CCS VTEP card
     9 DVV00     DS-Remotemapping
    10 DVR01     Papertape Reader
    11 DVC01     Papertape Reader
    12 DVR02     Papertape Punch
    13 DVS02     Beckman ctrlr.
    14 DVC02     Papertape Punch
    15 DVC05     BACI card
    16 DVR05     BACI card
    17 DVS05     12920 Mu-DVR05
    18 DVX05     CIS BMUX port
    19 DVA05     Terminal w/modem
    20 DVM05     12792x 8-ch Mux
     .   .          .
     .   .          .
     .   .          .
   110 DVM72     Genpurpose Cards
   111 DVS72     GP 16-bit I/O
   112 DVR73     9570 DTU
   113 DVR74     2321 subsystem
   114 DVR76     2320/22 subsystem
   115 DVS76     HS dummy driver
   116 DVT76     HiSpeed 2ch DVR
   117 DVI76     Dual Chan DVR
   118 DVR77     2323 Scanner
   119 DVL77     44611A I/O port
   120 DVI77     Gen Purpose DVR
   121 DVS77     HS transient DVR
   122 DVT77     CDA driver
 :
```

In the following example, LUPRN is invoked to list all system LUs within the range of LU 12 through LU 23. If no LUs are found within the range, the message

```
..no LUs found in specified range..
```

is issued as the body of the table. Note that the table contains entries for unidentifiable drivers, flagged with an asterisk. The configuration list also includes a note, Note 1, related to the unidentified driver entry. LUPRN provides seven notes related to specific conditions of the system. These notes are defined following the examples.

```
:RU,LUPRN,LU:12:23
              RTE System Device Configuration
          RTE System rev = 6000  LUPRN's rev = 6000
     1:52 PM TUE., 6 APR., 1993...Sorted by System LU
Time Base (14B)  Priv.  Fence SC (none)  Partitions (24) Memory size (576K)

SLU LU EQT, sc SCD Flags AV T.out Stats Driver DP Device Name       LU SLU
    12  38,3  31B  B              130B  DVB12  45 2608A read-back   12
    13  39    71B  B                    DVZ12   4 2608A Graphics    13
    15  40    04B                       DVP43   9$Power Fail        15
16  16   1,1  15B D              120B  DV?32*  4 .Unknown Driver.  16  16
    17   1,2  15B D              120B  DVR32   4 7905/6/20/25 DSK  17
18  18   1,3  15B D              120B  DVR32   4 7905/6/20/25 DSK  18  18
    19   3,4  16B D              100B  DV?32* 41 .Unknown Driver.  19
    20   3,1  16B D              100B  DVR32  41 7905/6/20/25 DSK  20
    21   3,10 16B D              100B  DVR32  41 7905/6/20/25 DSK  21
22  22   3    16B D              100B  DVR32  41 7905/6/20/25 DSK  22  22
    23   3,11 16B D              100B  DVR32  41 7905/6/20/25 DSK  23

Note 1:  DV?XX* indicates that the true driver name is not determinable since
         there are other drivers with the same INIT/CONT addresses.

    DP=Driver Partition page ($=SDA), SLU=Session LU
    (T.out is in seconds)                EQT Flags:
    LU # with a      EQT availability:    D=DCPC, B=Buffered, T=Timed-out
    D means the       1=down, 2=busy,     P=Driver handles Powerfail
    LU is down.       3=waiting DCPC      S=Driver handles Timeout

    :
```

In the following example, LUPRN is invoked to list all driver types within the range of 5B through 12B, sorted by system LU number. If no drivers are found within the range specified, the message

```
..no LUs found..
```

is issued as the body of the table. In the listing, those devices designated as spool printers are identified as such in the Device Name column. Note the letter "D" following LU entry 86. This signifies that the LU is down.

```
:RU,LUPRN,TY:5B:12B
                    RTE System Device Configuration
                RTE System rev = 6000  LUPRN's rev = 6000
             1:53 PM TUE., 6 APR., 1993...Sorted by System LU
Time Base (14B)  Priv.  Fence SC (none)  Partitions (24)  Memory size (576K)
 SLU LU EQT,sc SCD Flags AV T.out Stats Driver DP Device Name      LU  SLU

  6   6  6    26B  B       200.00      DVR12  45 2767 80col Prntr   6   6
 11  11 38    31B  B              130B DVB12  45 2608A Printer     11  11
     12 38,3  31B  B              130B DVB12  45 2608A read-back   12
     13 39    71B  B                   DVZ12   4 2608A Graphics    13
 56  56 13    32B  B S             2B DVA05  47 Terminal w/modem   56  56
     57 14    33B  B S   2         2B DVA05  47 Terminal w/modem   57
     58 15    34B  B S   2         2B DVA05  47 Terminal w/modem   58
     59 16    35B    S     200.00   2B DVA05  47 Terminal w/modem   59
     .   .     .    .       .       .   .     .        .
     .   .     .    .       .       .   .     .        .
     .   .     .    .       .       .   .     .        .
     84 20,1  41B  B S     200.00   2B DVA05  47 Left CTU@ LU 63    84
     85 20,2  41B  B S     200.00   2B DVA05  47 Right CTU@ LU 63   85
     86D21,1  42B    S     200.00   2B DVA05  47 Left CTU@ LU 64    86
     87 21,2  42B  B S   2         2B DVA05  47 Right CTU@ LU 64   87
     88 22,1  43B  B S     200.00   2B DVA05  47 Left CTU@ LU 65    88
     89 22,2  43B  B S     200.00   2B DVA05  47 Right CTU@ LU 65   89
     94 37    44B  B S     100.00   2B DVA05  47 Terminal w/modem   94
    101 34    75B                      DVS12   9$Spool....type=12 101
    102 35    76B                      DVS12   9$Spool....type=12 102
    103 36    77B                      DVS12   9$Spool....type=12 103
    117 37    44B  B S     100.00   2B DVA05  47 Terminal w/modem  117
    118 37,1  44B  B S     100.00   2B DVA05  47 Left CTU@ LU117   118
    119 37,2  44B  B S     100.00   2B DVA05  47 Right CTU@ LU117  119

    DP=Driver Partition page ($=SDA), SLU=Session LU
    LU # with a      EQT availability:     D=DCPC, B=Buffered, T=Timed-out
    D means the       1=down, 2=busy,      P=Driver handles Powerfail
    LU is down.       3=waiting DCPC       S=Driver handles Timeout

:
```

The following example invokes LUPRN with the SC option to list all devices within the range of 10B through 23B, sorted by select code. The listing contains the entry "<Empty Select Code>" to indicate that no Equipment Table entry was found for this device select code. The Configuration Table also contains the entries "Unknown Driver" for those drivers whose identity could not be determined, and includes an explanatory note. (Refer to the section "LUPRN Notes" for the text of all notes that can be listed.)

```
:RU,LUPRN,SC:10B:23B
                    RTE System Device Configuration
               RTE System rev = 6000   LUPRN's rev = 6000
            1:54 PM TUE., 6 APR., 1993...Sorted by Select Code
Time Base (14B)  Priv.  Fence SC (none)  Partitions (24)  Memory size (576K)

SLU LU EQT,sc SCD Flags AV T.out  Stats Driver DP Device Name      LU  SLU
              10B  <Empty Select Code>
    104 41    11B   PS              26B DVA66  53 HDLC/BiSync card 104
    106 43    12B   PS               4B DVA66  53 HDLC/BiSync card 106
  7   7 10    13B   PS        .03    4B DVA65  51 DS1000 to 1000     7  7
              14B                       <RTE>     Timebase Generator
  2   2  1    15B D                 120B DV?32*  4 .Unknown Driver.  2  2
  3   3  3    16B D                 100B DV?32* 41 .Unknown Driver.  3  3
      1  2    17B    S   2 327.67    2B DVR00  45 CRT/TTY terminal   1
 55   5  5    20B                   40B DVR01  45 Papertape Reader  55  5
 44   4  4    21B  B       200.00       DVR02  45 Papertape Punch   44  4
 54  54 12    22B  B S     200.00       DVA37  49 59310 HPIB Card   54 54
  9   9  9    23B   PS        .03       DVA65  51 DS1000 to 1000     9  9

Note 1:  DV?XX* indicates that the true driver name is not determinable since
         there are other drivers with the same INIT/CONT addresses.

  DP=Driver Partition page ($=SDA), SLU=Session LU
  (T.out is in seconds)                EQT Flags:
  LU # with a      EQT availability:     D=DCPC, B=Buffered, T=Timed-out
  D means the       1=down, 2=busy,      P=Driver handles Powerfail
  LU is down.       3=waiting DCPC       S=Driver handles Timeout

:
```

The following example invokes LUPRN with no options, to list the configuration of all devices available to the session, sorted by session LU number. Note the unknown driver entries and the explanatory note following the table.

```
:RU,LUPRN
                    RTE System Device Configuration
                RTE System rev = 6000  LUPRN's rev = 6000
             1:55 PM TUE., 6 APR., 1993...Sorted by Session LU
Time Base (14B)  Priv.  Fence SC (none)  Partitions (24)  Memory size (576K)
```

SLU	LU	EQT,sc	SCD	Flags	AV	T.out	Stats	Driver	DP	Device Name	LU	SLU
1	67	48	70B	ST			2B	DVV05	51	Logical DVR = 5	67	1
2	2	1	15B	D			120B	DV?32*	4	.Unknown Driver.	2	2
3	3	3,16	16B	D			100B	DV?32*	41	.Unknown Driver.	3	3
4	92	25	45B	B		200.00	40B	DVM00	55	12792A 8-CH MUX	92	4
5	93	26	45B	B		5.00	40B	DVM00	55	12792A 8-CH MUX	93	5
6	6	6	26B	B		200.00		DVR12	45	2767 80col Prntr	6	6
7	7	10,1	13B	PS		.03		DVA65	51	DS1000 to 1000	7	7
8	8	8	24B	B	S	5.00	4B	DVR23	41	9TK Mag Tape #0	8	8
9	9	9,1	23B	PS		.03		DVA65	51	DS1000 to 1000	9	9
10	10	7	27B	S		5.00	1B	DVR23	41	9TK Mag Tape #0	10	10
11	11	38	31B	B			130B	DVB12	45	2608A Printer	11	11
16	16	1,1	15B	D			120B	DVR32	4	7905/6/20/25 DSK	16	16
18	18	1,3	15B	D			120B	DVR32	4	7905/6/20/25 DSK	18	18
22	22	3	16B	D			100B	DVR32	41	7905/6/20/25 DSK	22	22
37	67	48	70B	ST			2B	DVV05*	51	.Unknown Driver.	67	37
38	38	3,5	16B	D			100B	DVR32	41	7905/6/20/25 DSK	38	38
39	69	50	70B				40B	DVV00	51	DS-Remotemapping	69	39
41	41	3,8	16B	D			100B	DVR32	41	7905/6/20/25 DSK	41	41
44	4	4	21B	B		200.00		DVR02	45	Papertape Punch	4	44
54	54	12,2	22B	B	S	200.00		DVA37	49	HPIB address # 2	54	54
55	5	5	20B				40B	DVR01	45	Papertape Reader	5	55
56	56	13	32B	B	S		2B	DVA05	47	Terminal w/modem	56	56

```
Note 1:  DV?XX* indicates that the true driver name is not determinable since
         there are other drivers with the same INIT/CONT addresses.

    DP=Driver Partition page ($=SDA), SLU=Session LU
    (T.out is in seconds)                EQT Flags:
    LU # with a      EQT availability:     D=DCPC, B=Buffered, T=Timed-out
    D means the       1=down, 2=busy,      P=Driver handles Powerfail
    LU is down.       3=waiting DCPC       S=Driver handles Timeout

:
```

The last example invokes LUPRN with the AL option to list the configuration of all LUs in the system, sorted by system LU. Where a spool LU is unassigned, the entry <idle> is shown in the Device Name column. This listing also identifies all LUs that are unassigned in the system.

```
:RU,LUPRN,AL
                    RTE System Device Configuration
                 RTE System rev = 6000  LUPRN's rev = 6000
             1:58 PM TUE., 6 APR., 1993...Sorted by System LU
Time Base (14B)  Priv.  Fence SC (none)  Partitions (24)  Memory size (576K)

SLU LU EQT,sc SCD Flags AV T.out  Stats Driver DP Device Name      LU  SLU

      1  2    17B    S     327.67   2B  DVR00  45 CRT/TTY terminal   1
  3   3  3,16 16B D      2        101B  DV?32* 41 .Unknown Driver.   3  3
 44   4  4    21B  B       200.00       DVR02  45 Papertape Punch   44  4
 55   5  5    20B                 40B  DVR01  45 Papertape Reader  55  5
  6   6  6    26B  B       200.00       DVR12  45 2767 80col Prntr   6  6
  7   7 10,1  13B   PS        .03       DVA65  51 DS1000 to 1000     7  7
  8   8  8    24B  B S      5.00   1B  DVR23  41 9TK Mag Tape #0     8  8
  9   9  9,1  23B   PS        .03       DVA65  51 DS1000 to 1000     9  9
 10  10  7    27B    S      5.00   1B  DVR23  41 9TK Mag Tape #0    10 10
 11  11 38    31B  B              130B  DVB12  45 2608A Printer     11 11
     12 38,3  31B  B              130B  DVB12  45 2608A read-back   12
     13 39    71B  B                    DVZ12   4 2608A Graphics    13
     14 11    61B  B       200.00  40B  DVR00  45 CRT/TTY terminal  14
     15 40    04B                       DVP43   9$Power Fail        15
 16  16  1,1  15B D              120B  DVR32   4 7905/6/20/25 DSK  16 16
     17  1,2  15B D              120B  DVR32   4 7905/6/20/25 DSK  17
 18  18  1,3  15B D              120B  DVR32   4 7905/6/20/25 DSK  18 18
     19  3,4  16B D      2       100B  DVR32  41 7905/6/20/25 DSK  19
      :   :     :       :         :     :     :        :          :
  5  93 26    45B  B        5.00  40B  DVM00  55 12792X 8-CH MUX    5 93
     94 37    44B  B S    100.00   2B  DVA05  47 Terminal w/modem  94
     95 28    45B  B      200.00  40B  DVM00  55 12792A 8-CH MUX   95
     96 29    45B  B      200.00  40B  DVM00  55 12792A 8-CH MUX   96
     97 30    45B  B      200.00  40B  DVM00  55 12792A 8-CH MUX   97
     98 31    72B                       DVS43   9$Spooling (idle)   98
     99 32    73B                       DVS43   9$Spooling (idle)   99
    100 33    74B                       DVS43   9$Spooling (idle)  100
    101 34    75B                       DVS12   9$Spool....type=12 101
    102 35    76B                       DVS12   9$Spool....type=12 102
    103 36    77B                       DVS12   9$Spool....type=12 103
    104 41    11B   PS                26B  DVA66  53 HDLC/BiSync card 104
    105 42    11B   PS                     DVA66  53 HDLC/BiSync card 105
    106 43    12B   PS                 4B  DVA66  53 HDLC/BiSync card 106
    107 44    12B   PS                     DVA66  53 HDLC/BiSync card 107
    108 45    62B                          DVA66  53 HDLC/BiSync card 108
    109 46    62B                          DVA66  53 HDLC/BiSync card 109
    110 12    22B  B S    200.00           DVA37  49 59310 HPIB Card  110
    111 12,1  22B  B S    200.00           DVA37  49 HPIB address # 1 111
    112 12,2  22B  B S    200.00           DVA37  49 HPIB address # 2 112
    113 12,3  22B  B S    200.00           DVA37  49 HPIB address # 3 113
    114 12,4  22B  B S    200.00           DVA37  49 HPIB address # 4 114
    115 51    67B                          DV?77+ 57 .Unknown Driver. 115
    116 52    66B                          DV?77+ 59 .Unknown Driver. 116
    117 37    44B  B S    100.00   2B  DVA05  47 Terminal w/modem 117
    118 37,1  44B  B S    100.00   2B  DVA05  47 Left CTU@ LU117  118
    119 37,2  44B  B S    100.00   2B  DVA05  47 Right CTU@ LU117 119

        ************  System LU's 120 thru 122 Unassigned  *************
```

```
Note 1:  DV?XX* indicates that the true driver name cannot be determined since
         there are other drivers with the same INIT/CONT addresses.

Note 2:  DV?XX+ indicates that no entry point in the system matches either
         INIT/CONT address in this EQT.  Possibly a sysgen error or
         incomplete patch has been made, or this is a dummy driver.

    DP=Driver Partition page ($=SDA), SLU=Session LU
    (T.out is in seconds)                EQT Flags:
    LU # with a      EQT availability:     D=DCPC, B=Buffered, T=Timed-out
    D means the        1=down, 2=busy,     P=Driver handles Powerfail
    LU is down.        3=waiting DCPC      S=Driver handles Timeout
```

LUPRN Notes

The following explanatory notes can be displayed by LUPRN as specific related conditions are found in the system configuration:

```
Note 1: DV?XX* indicates that the true driver name cannot be determined since
        there are other drivers with the same INIT/CONT addresses.

Note 2: DV?XX+ indicates that no entry point in the system matches either
        INIT/CONT address in this EQT.  Possibly a system error or incomplete
        patch has been made, or this is a dummy driver.
```

The above two notes are related to the symbols * and + as they appear next to the driver number in the Driver column.

```
Note 3: 'No EQT' indicates that no EQT claims use of this select code.
        Possibly a sysgen error was made in the EQT for an interrupt address.
```

The above note is related to the entry or entries identified by the message

```
<Unknown Interrupt Table Entry - no EQT>
```

following the SCD (select code) column.

```
Note 4: The program name listed will be scheduled upon an interrupt from this
        Select Code.  There is no EQT or driver associated with this S.C.
```

The above note is related to the entry or entries identified by the message

```
<Interrupt schedules program: (prog name) ...see Note 4>
```

following the SCD column. The program name is given in the message.

```
Note 5: The value listed is not an ID segment address and no EQT points to
        this Select Code.
```

The above note is related to the entry or entries identified by the message

```
<Interrupt table entry unknown (value)...See Note 5>
```

following the SCD column. The value is given in the message. This condition may be due to an incorrect patch or a corrupt system.

```
Note 6: A subchannel for CS/80 disk is not part of track map table.  This LU
        cannot be accessed.
```

The above note is related to the entry or entries identified by the message

```
<CS/80 subchan bad, (nn)>
```

following the SCD column. The subchannel is given in the message. This may be due to a system error or as a result of assigning an LU to a non-existent disk subchannel.

```
Note 7: Unable to find track map table for CS/80 disk driver(s).  The TMT is
        not in the system entry point list.
```

The above note is related to the fact that LUPRN only searches the system entry point list for the track map tables. This allows LUPRN to continue without asking the driver for the TMT entry. The CS/80 driver requires an I/O card to return the TMT entry.

Customizing Driver Names

LUPRN obtains driver names and descriptions from an external file, either ”LUPRN::SYSTEM or ”LUPRN::0. For ”LUPRN::0, the file should be stored on a globally mounted FMGR disk LU. You can modify ”LUPRN to add new drivers or change the description. If duplicate driver names are encountered, only the first name is used.

The format is as follows:

Columns	Description
1−5	Actual driver name (that is, DVR05). If first character = *, the line is a comment.
6	A blank or any non-blank character for special subchannel handling.
7−22	16-character description of driver or card.

Note Special subchannel handling refers to HP-supported drivers where additional information is overlaid on top of the description. For example, most driver 5 terminals allow subchannels to specify internal or external slaved devices (cassettes, printers). For user-written drivers, leave this field blank. Special handling varies with different drivers and applies only to standard HP drivers and subchannels.

LUPRN Errors

On an error condition, LUPRN issues the message:

```
...LUPRN: entry error code = nn
```

where nn identifies the I/O call error code (such as IO07). Refer to the *RTE-6/VM Quick Reference Guide*, part number 92084-90003, for a description of the I/O error codes.

Serial Port Analyzer (SPORT)

SPORT displays the status of serial ports, which are driven by DDC00 in RTE-A and by DVC00 or DV800 in RTE-6/VM. The program verifies that a port was set up in the desired manner, and it is useful for troubleshooting problems involving serial ports.

Calling SPORT

To run SPORT, enter the following runstring:

```
CI> [ru,]SPORT [port_lu [,display_lu [,waitflag]]]
```

The port_lu is the serial port LU for which the current status is requested, and the display_lu is the LU to which the output is directed. Output can be directed to another LU, but not to a file. Both LUs default to the session LU.

The waitflag parameter allows you to specify a non-zero value to force a "wait" for a busy port; in other words, you can have SPORT "hang" on the port until the current request completes.

All parameters must be entered in the runstring, as the program is non-interactive. The online help function is available when you enter ??, as shown in the following example:

```
CI> ru,sport,??

Sport Rev 5.20 890802
invocation is:  SPORT Port_LU Display_LU WaitFlag
     Port_LU    = system LU number of port to be examined.
     Display_LU = system LU number for output list
     Set WaitFlag non-zero to force a wait for a busy port


     Both LUs default to session LU
     Indicate LU = 100001B for system console
     The Port_Lu parameter can be a range of LUs in form X:Y
```

As shown above, the program displays information about one or more serial ports.

SPORT Operation

The default operation of SPORT is to first check the port LU to see if it is busy, down, or locked. If it is not, the program issues a Special Status Read (described in the appropriate serial driver reference manuals) to obtain the current status of the port.

If the port is not available, SPORT invokes a subroutine called FakeSpStatus to read as much of the status as is available in the system tables. Since SPORT is used as a troubleshooting tool, this is usually the most desirable mode of operation. For the occasions when you want SPORT to "hang" on the port until the current request completes, you can set the WaitFlag to any non-zero value.

SPORT Examples

When the status is obtained from a special status read, the program formats the result into a display similar to the following example:

```
CI.84> sport 84
Status for LU 84:
Device Driver:    DV801 Rev. 6000    Driver type = 05
 Firmware:              Rev. 5.20
CN20: Schedule Program: PRMPT (Enabled)
CN17: 000000B No user defined terminators
CN22:  65535  Timeout = 655.35 seconds
CN30: 010133B Frame=8/1 None BRG1 9600 baud Port 3
CN31: 000000B
CN33: 000000B
CN34: 000002B HP Protocol
DV20: 000000B Character mode
EQT Address: 03132B
```

When the status is obtained from a call to FakeSpStatus, the display is similar to the following example:

```
CI.84>  sport 84
EQT of LU 84 is busy:
Status for LU 84:
Device Driver:    DV?0? Rev. -1      Driver type = 05
 Firmware:              Rev. 255.55
CN20: Schedule Program: PRMPT (Enabled)
CN17: 000000B No user defined terminators
CN22:  65534  Timeout = 655.34 seconds
CN30: 010133B Frame=8/1 None BRG1 9600 baud Port 3
CN31: 000000B
CN33: 000000B
CN34: 000002B HP Protocol
DV20: 000000B Character mode
EQT Address: 03132B
```

On the first line of the display, the words “Static status report” are shown in half-intensity flashing video (on terminals so equipped) as a warning that this is synthesized information. Note that many of the fields now contain “odd values” because the information is not available from the system tables.

Including SPORT in a User Program

The SPORT program functionality can be included in a user program by calling HpZPrintPort. For example:

```
CALL HpZPrintPort(Port_LU, Display_LU, WaitFlag)

integer*2 Port_LU, Display_LU, WaitFlag
```

where:

Port_LU	is the LU number of the port to display (in the range 1..255) in the format of the first word of the XLUEX control word.
Display_LU	is the LU number of the display LU (in the range 1..255) in the format of the first word of the XLUEX control word. If it is not in the valid range, it will default to the call LU.
WaitFlag	specifies whether to get real or "fake" status in the event that port_LU is busy. If WaitFlag is 0, the true status is displayed. If WaitFlag is nonzero, a call to FakeSpStatus is made.

This subroutine prints out the port status (using a special status read) of Port_LU to Display_LU. If Port_LU = 1, it is converted to the true LU number.

The subroutine verifies that the LU is neither locked, down, nor busy. Otherwise, a message is output to the calling LU.

This is designed to be a simple addition to a program such as the following:

```
Ftn7x,Q,S
      program HpCrt
      integer prams(5),obuf(0:79)
      call rm par(prams)
      call HpZDefObuf(Obuf)
      call HpZPrintPort(prams(1),prams(2),prams(3))
      end
```

Compile Utility (COMPL)

The COMPL utility allows you to automatically invoke the appropriate HP supported compiler or assembler for a specified source file.

Calling COMPL

To call COMPL, enter the following runstring:

```
:[RU,]COMPL,source[,list[,reloc[,compiler[,option]]]]
```

Source is the source file to be compiled or assembled Source is the only required parameter. Default values for the other parameters depend on the format or the file contents of the source file.

List is the disk file or LU to which the compiler or assembler listing is to be sent. If the specified LU was set up at GASP initialization and is not an interactive device, the output is automatically spooled to that LU. In this case, the following message is issued:

```
SPOOL FILE = COXXYY
```

where:

CO	identifies the operation as a compile.
XX	is the session identification number (or the terminal LU number if operating under the Multi-Terminal Monitor).
YY	is the number of the COMPL or CLOAD spool file. A session is limited to a maximum of 80 COMPL or CLOAD spool files. (Refer to the discussion of the Compile and Load Utility later in this chapter for a description of CLOAD.) COMPL will try to create this spool file on the spool disk. If the spool disk is almost full, the following warning will be issued:

```
/COMPL: Warning, spool disk is getting full
/COMPL: Inform system manager.
```

COMPL assumes that SPLCON and JOBFIL reside on the spool disk. If you do not want automatic spooling, specify the LU as “lu:NS” for non-spooling (for example, 8:NS).

If a minus sign is specified for the list parameter, then

IF

1. The source file is found on a FMGR cartridge, AND
2. the first character of the source file is an ampersand (&);

OR

1. The source file is found on a hierarchical directory, AND
2. the source file has a valid type extension (refer to the appropriate compiler or assembler manual for details);

an appropriate list file name will be created by the scheduled compiler or assembler. If the above conditions are not met or the list parameter is omitted, the listing goes to your terminal.

Reloc is the file descriptor in which the relocatable code is to be placed. If this parameter is specified with a minus sign or omitted, then

IF

1. The source parameter resides on a FMGR disk, AND
2. the first character of the source file is an ampersand (&);

OR

1. The source file resides on a hierarchical directory, AND
2. the source file has a valid type extension (refer to the appropriate compiler or assembler manual for details);

an appropriate relocatable file name will be created by the scheduled compiler or assembler. Otherwise, the relocatable code will go to the bit bucket and not be printed.

The compiler is the name of the compiler to use.

Option is the file descriptor containing a control string option list for the PASCAL compiler or an option string that will be passed to all other compilers or assemblers. Refer to the appropriate compiler or assembler manual for the valid options.

COMPL Examples

Example 1:

```
:COMPL,&PROG
```

Since the source file name starts with an ampersand, the listing goes to your terminal, and the relocatable code is placed in a file named %PROG.

Example 2:

```
:COMPL,&PROG,-,-
```

Since the source file name starts with an ampersand, and the minus sign is specified for the list and reloc parameters, the list file goes to a disk file named 'PROG, and the relocatable code is placed in a disk file named %PROG.

Example 3:

```
CI.74> COMPL PROG.FTN - -
```

Since the source file name has a valid type extension, and the minus sign is specified for the list and reloc parameters, the list file goes to a disk file named PROG.LST, and the relocatable code goes to a disk file named PROG.REL.

Example 4:

```
:COMPL,PROG
```

The source file PROG is compiled, the listing goes to the terminal, and the relocatable code goes into the bit bucket.

Example 5:

```
:COMPL,PROG,-,%PROG,FTN7X,LA
```

The source file PROG is compiled using compiler FTN7X, the listing goes to your terminal, and the relocatable code is placed in disk file %PROG.

Example 6:

```
:COMPL,&PROG,6,-
```

The source file &PROG is compiled. The listing is spooled to LU 6 (the line printer), and the relocatable code is placed in the disk file %PROG.

Example 7:

```
CI.74> COMPL PROG.FTN 6:NS -
```

The source file PROG.FTN is compiled. The listing goes to LU 6 (the line printer), but is not spooled. The relocatable code is placed in disk file PROG.REL.

Compile and Load Utility (CLOAD)

The CLOAD utility lets you automatically invoke the appropriate HP supported compiler or assembler for a specified source file. In addition, CLOAD schedules either LINK or LOADR to relocate the compiled or assembled code. The default is LINK. If a LINK/LOADR command file is not given in the runstring, the default options will be scheduled. Refer to the *RTE-6/VM Loader Reference Manual*
(part number 92084−90008) and the *RTE-6/VM Link User's Manual*, part number 92084-90038, for a detailed description of LOADR and LINK.

Calling CLOAD

To call CLOAD, enter the following runstring:

```
:[RU,]CLOAD,source[,list[,reloc[,compiler[,option[,ldcmd[,loader]]]]]]
```

Source is the source file to be compiled or assembled. Source is the only required parameter. Default values for the other parameters depend on the format or the file contents of the source file.

List is the disk file or LU to which the compiler or assembler listing is to be sent. If the specified LU was set up at GASP initialization and is not an interactive device, the output is automatically spooled to that LU. In this case, the following message is issued:

```
SPOOL FILE = COXXYY
```

where:

CO — identifies the operation as a compile.

XX — is the session identification number (or the terminal LU number if operating under the Multi-Terminal Monitor).

YY — is the number of the CLOAD spool file. A session is limited to a maximum of 80 CLOAD or CLOAD spool files. CLOAD will try to create this spool file on the spool disk. If the spool disk is almost full, the following warning will be issued:

```
/CLOAD: Warning, spool disk is getting full
/CLOAD: Inform system manager.
```

CLOAD assumes that SPLCON and JOBFIL reside on the spool disk. If you do not want automatic spooling, specify the LU as "lu:NS" for non-spooling (for example, 6:NS).

If a minus sign is specified for the list parameter, then

IF

1. The source file is found on a FMGR cartridge AND
2. the first character of the source file is an ampersand (&),

OR

1. The source file is found on a hierarchical directory AND
2. the source file has a valid type extension (refer to the appropriate compiler or assembler manual for details),

an appropriate list file name will be created by the scheduled compiler or assembler. If the above conditions are not met or the list parameter is omitted, the listing goes to your terminal.

Reloc is the file descriptor in which the relocatable code is to be placed. If this parameter is specified with a minus sign or omitted then

IF

1. The source parameter resides on a FMGR disk AND
2. the first character of the source file is an ampersand (&),

OR

1. The source file resides on a hierarchical directory AND
2. the source file has a valid type extension (refer to the appropriate compiler or assembler manual for details),

an appropriate relocatable file name will be created by the scheduled compiler or assembler. Otherwise the relocatable code will go to the bit bucket and not be printed.

Compiler is the name of the compiler to use.

Option is the optional file descriptor containing a control string option list for the PASCAL compiler or an option string that will be passed to all other compilers or assemblers. Refer to the appropriate compiler or assembler manual for the valid options.

The ldcmd is the optional LINK or LOADR command file descriptor that contains the load commands. If the LINK program is to be scheduled, the file name for the command file should start with a number sign (#) or have a '.LOD' type extension.

Loader is the name of the loader to use.

CLOAD Examples

Example 1:

```
:CLOAD
```

By invoking CLOAD with no parameters, you enter interactive mode and CLOAD prompts with:

```
SOURCE,LIST,RELOC,[C.S.],[OPTION],[LDCMD],[L.S.]
```

You can respond with the file descriptor of the source program and any desired optional parameters.

Example 2:

```
:CLOAD,&PROG
```

Since the source file name starts with an ampersand, the listing goes to your terminal, the relocatable code is placed in a file named %PROG, and LINK relocates %PROG, creating an executable file called PROG.

Example 3:

```
:CLOAD,&PROG,-,-
```

Since the source file name starts with an ampersand and the minus sign is specified for the list and reloc parameters, the list file goes to a disk file named ‘PROG, the relocatable code is placed in a disk file named %PROG, and LINK is scheduled to relocate %PROG, creating an executable file called PROG.

Example 4:

```
CI.74> CLOAD PROG.FTN - -
```

Since the source file name has a valid type extension and the minus sign is specified for the list and reloc parameters, the list file goes to a disk file named PROG.LST, the relocatable code goes to a disk file named PROG.REL, and LINK is scheduled to relocate PROG.REL creating an executable file called PROG.RUN.

Example 5:

```
:CLOAD,PROG
```

The source file PROG is compiled and the listing goes to the terminal, the relocatable code goes into the bit bucket. Neither LINK nor LOADR is scheduled.

Example 6:

```
:CLOAD,PROG,-,%PRG,FTN7X,LA
```

The source file PROG is compiled using compiler FTN7X, the listing goes to your terminal, and the relocatable code is placed in disk file %PRG.

Example 7:

```
:CLOAD,&PROG,6,-,,,,LOADR
```

The source file &PROG is compiled. The listing is spooled to LU 6 (the line printer), and the relocatable code is placed in the disk file %PROG; then LOADR relocates %PROG, creating a temporary program.

Example 8:

```
CI.74> CLOAD PROG.FTN 6:NS - FTN4X S PROG.LOD LINK
```

The source file PROG.FTN is compiled. The listing goes to LU 6 (the line printer), but is not spooled. The relocatable code is placed in disk file PROG.REL. The compiler is FTN4X, and the 'S' option is used. Link will relocate PROG.REL using the command file PROG.LOD, and an executable file called PROG.RUN will be created.

File Manipulation Utilities (MERGE, SCOM)

The file manipulation utilities allow you to merge files, concatenate and store files, and compare files to determine line-by-line differences. Two utilities, MERGE and SCOM, comprise the file manipulation group.

Concatenate Many Files into One (MERGE)

MERGE collects input files and sends their data to a single composite output file. This lets you form a library of binary relocatable files to be searched by the linker and to concatenate a program and all user-written subroutines.

Calling MERGE

To call MERGE, enter the following runstring:

```
CI> [RU,] MERGE [-options] [fromfile fromfile...] [destfile]
```

Fromfile is one of the following:

- A command file containing a list of names of files to be merged. To be a command file, fromfile must have an .MRG or .MERG type extension; alternatively, it can be an FMGR file starting with "*" (for example, *MERGM), or a device LU. Masks may be used in a command file to indicate the files to be merged. If only one fromfile is entered and it is of type 3, 4, or 0, it is assumed to be a command file for backward compatibility.
- A file to be merged into destfile. To force the merging of a file that would otherwise be a command file, precede the file name with a backslash (\) (as in "\MergeThis.merg"). If you do this from CI, note that you must precede the file name with two backslashes, rather than one, since CI uses the backslash as a quoting character.
- A mask of files to be merged. For example, the mask "@.ftn" merges all the files with the extension "ftn" on the working directory. A mask may also be preceded by a backslash, which lets you enter masks that start with a dash, as in "\−q@," which specifies all files with a "q" in the second character. Note that two backslashes must be entered from CI, because CI uses the backslash as a quoting character.

You may specify multiple fromfiles in a runstring, each of which is interpreted as above.

If a fromfile is an interactive LU or you do not specify one, MERGE prompts for the input data files as follows (after prompting for the destination file name):

```
Merge file:
```

Enter a file name or mask to be merged. MERGE continues prompting until you enter /E, press RETURN, or press control-D at the prompt.

The destfile is the file or LU that receives the merged files. It may safely be the same as the first input file. If you name a file that does not exist, it is created. An existing file may be overwritten

without notification unless you use the -V option (described below). The destfile is the last file name in the runstring.

MERGE Options

You may enter options anywhere in the runstring, each one preceded by a dash (−). The options are summarized in Table 2-6.

Table 2-6. MERGE Options Summary

Options	Description
−L	Suppresses the listing of file names being merged
−D	Removes Debug records from a relocatable file as it is merged
−O	Overwrites existing files without asking for user confirmation (if MERGE is linked to ASK by default)
−V	Verifies that an existing output file is to be overwritten
−Z	Suppresses zero-length records normally output between files

The MERGE Operation

If a command file is specified in the runstring (for example, “these.mrg”), the names of the files to be concatenated are read from it until the program encounters an EOF.

A zero-length record is written to the output after each input file, unless you selected the -Z option. This creates a subfile structure that is useful to some utilities and the FMGR ST command.

MERGE always removes index records (created by LINDX) from a file that it merges because the indexes are no longer valid for the new, merged file. To create a new index, run LINDX on the file after MERGE completes its operation.

Debug records (created by a compiler such as FTN7X or MACRO) can also be removed with the -D option. If you are not running the DEBUG/1000 program against the merged file, removing Debug records creates a smaller destination file.

It is legal (and useful) to make the output file name identical to the first input file name. For example, suppose that program source TRYIT.FTN has relocatable object code TRYIT.REL and requires the subroutines in TRYLIB.REL.

```
CI> MERGE TRYIT.rel TRYLIB.rel TRYIT.rel
```

TRYIT.REL now contains both the main program and the subroutines.

Break Detection

MERGE recognizes a BReak command prior to reading the file name to be added to the output file when the command device is a file or non-interactive LU. The BR command is entered as follows:

```
CI> BR,MERGE
```

MERGE Examples

Example 1: Ask for input and destination files from the terminal.

```
merge
```

Example 2: Create the library UTIL.LIB, using the list of files in UTIL.MRG. Remove Debug records, and do not list the files merged.

```
merge -dl util.mrg util.lib
```

Example 3: Merge all the macro sources everywhere into file BIG.MAC.

```
merge @.mac.e big.mac
```

Example 4: Merge file A, the files listed in B.MRG, and file C into file MEGAFILE. Do not list the files merged, and verify that an existing MEGAFILE may be overwritten.

```
merge -lv a b.mrg c megafile
```

Loading MERGE

Merge can be linked to use either the −V or −O option by default if neither option is specified in the runstring. The supplied MERGE.LOD uses the −O option by default. To use −V instead, edit MERGE.LOD and follow the instructions contained therein.

File Comparison (SCOM)

The SCOM utility compares two input files and identifies their differences by matching sequences of lines and flagging those lines that are unique to a single file. By specifying options, you can configure a listing of only those lines unique either to one or to both files, or only those lines that are common to both, with or without line numbers.

Calling SCOM

SCOM can be run programmatically or from your terminal. You may call SCOM with the following runstring:

```
CI> SCOM file1 file2 [[+|~]listfl] [options] [rematchlns] [maxchars] [difflimit]
```

File1 and file2 are the files to compare. Type 1 or 6 files force a binary block compare (see the discussion of the "bb" option, below); type 2 or 5 files assume a binary record compare (but see the discussion of the "tc" option, below).

Listfl is the listing file. The default is the terminal. The entry +listfl appends to an existing file, and ~listfl overlays an existing file.

Rematchlns indicates the number of consecutive lines that must match before a mismatch is ended; that is, the files are considered to be "rematched." The default is 3 lines.

Maxchars is the maximum number of characters per record in both file1 and file2. For hierarchical files, this is automatically set to the greater of the two maximum record lengths as given in the directory. For FMGR file system files, it is defaulted to 256; for binary block (bb) compares, it is forced to 256. The maximum value is 1024.

Difflimit is the maximum number of differences to report. If the limit is reached, the following message appears, and the comparison terminates as if complete:

```
* Mismatch limit reached
```

If difflimit is not specified, all the differences found are reported.

SCOM Options

The options available for use with the SCOM utility are summarized in Table 2-7. Additional information about some of the options is presented in the following sections.

Table 2-7. SCOM Options Summary

Options	Description
F1	Report lines unique to file1 (default).
F2	Report lines unique to file2 (default).
BO	Report lines common to both files.
NN	Suppress line numbers on report.
NH	No heading on listing (parameter information).
NT	No trailer on listing (number of mismatches).
TB	Trailing blanks are significant.
IB	Trailing blanks are insignificant.
D<x>	Wildcard; <x> matches any character in the other file.
C<x>	Ignore lines with <x> in column 1 but otherwise blank when rematching.
IT	Ignore EDIT/1000 time stamps.
IC	Ignore compile times in relocatable records for binary record compares.
ET	Create an EDIT/1000 transfer file to convert file1 to file2.
ER	Same as ET but add an EDIT “ER” command to the end of the transfer file.
BR	Binary record-by-record compare.
BB	Binary block-by-block compare.
TC	Force text compare on binary files.

F1, F2, BO

If none of these options is specified, options F1 and F2 are used by default for the standard differences report.

NN

This option suppresses line numbers on report.

NH

When this option is specified, no heading (parameter information) appears on the listing.

NT

When this option is specified, no trailing information (number of mismatches) appears on the listing.

TB

This option is the default for the ET, ER, BB, and BR options.

IB

When this option is specified, trailing blanks are ignored when matching occurs. This is the default for the normal report (ET, ER, BB, and BR not given).

D<x>

The D<x> option lets you specify a wildcard character that matches any character in the comparison file that is in the same line position as the wildcard. This can be any ASCII character, including a blank, except a comma, which is used as a runstring parameter delimiter. This option is useful in creating a mask file for comparison, filling the wildcard fields with the D option character.

As an example, entering d~ causes all tildes (~) in one file always to match the corresponding character in the other file, even if it is past the end of the other line.

C<x>

The C<x> option forces SCOM to ignore specified lines in searching for a rematch of the files. If an otherwise blank line has the C option character in column 1, the line is not considered in the search for a rematch. This option is useful for ignoring blank comment lines in a file.

<x> is typically c or * for FORTRAN or Macro source files. It may also be a blank, forcing SCOM to ignore totally blank lines in a file. For example, c* causes blank lines with a star (*) in column 1 to be ignored when rematching after a mismatch.

IT

If both files contain an EDIT/1000 timestamp of the form <nnnnnn.nnnn>, where n = 0..9, different times do not cause a mismatch; in other words, the n values can differ and still “match”.

IC

XNAM checksums, compile times, language processor revision codes, and comment strings and XEND checksums are forced to be equal when you specify this option.

ET, ER

When these options are specified, the listing file is an EDIT/1000 transfer file that contains the necessary insert, replace, and kill commands to make file1 identical to file2. For example:

```
scom ver2 ver1 ver2_1.edit er
```

The above command creates an EDIT/1000 transfer file named VER2_1.EDIT that contains the EDIT commands needed to modify file VER2 to the same text as VER1. To accomplish this, enter:

```
edit -b ver2 tr ver2_1.edit
```

The F1, F2, BO, NN, D<x>, BR, BB, and IT options are not meaningful for this mode. If the files are binary, you must specify the TC option, otherwise an error occurs.

BR, BB

In a binary record compare, each record of file1 is compared, record-by-record, against the corresponding numbered record in file2, and differences are reported in octal format. This is assumed if either file is type 2 or type 5.

In a binary block compare, each block of file1 is compared, block-by-block, against the corresponding numbered block in file2, and differences are reported in octal format. This is forced if either file is type 1 or type 6.

For both the BR and BB comparisons, mismatching blocks or records are dumped in 12-byte groups, side-by-side in both octal and ASCII (if standard printing character) format. The octal representation for file2 bytes appears as "..." if this byte is the same as the byte for file1 listed directly above. If only blanks appear in that field, the byte position is past the end of the record for that file. The last byte in each file is suffixed with a backslash (\). For example:

```
Record 12
  1    1:  377  377  000  000  000  000  000  000  000  000  000  000
 13     :  000  132  004  320  000  000  000  000  000  \               Z
       13:  ...  143  ...  000  ...  ...  \                            c
```

This says that in record 12 of both files, bytes 1-13, 15, and 17-18 match. Bytes 14 and 16 mismatch. Bytes 19-21 in file1 are past the end of the file2 record. Byte 14 is octal 132, ASCII "Z" in file1; that byte is octal 143, ASCII "c" in file2.

TC

This option forces a text compare on binary files.

The Compare Operation

SCOM compares the two input files and, when a mismatch is found, begins searching for a rematch. The files are considered rematched only when the number of lines (including blank lines) specified in rematchlns are identical.

Invalid options in the runstring are ignored, and SCOM executes using only the valid options. Any entry in the nmatch or nchar parameters other than a positive integer is ignored, and SCOM executes using the default values. SCOM does not issue a status message if any options are ignored; the run state is defined in the header of the comparison display.

Returned Values

Numeric parameters returned on exit are as follows:

$return1 =	0	If comparison is complete, this is the number of differences found. 0 means the files are identical.
		$return2 = 1 if records were truncated, 0 if not
$return1 =	−1	If an FMP error occurred.
		$return2 = FMP error code $return3 = 1 for file1, 2 for file2, 3 for listfile
$return1 =	−2	If SCOM error occurred.
		$return2 = 1 for runstring errors, others for internal errors

Status Interrogation

When SCOM executes a file compare, you may request the current status of the compare by entering break mode (striking any key on the keyboard) and using the BR command. SCOM responds with one of the following messages, followed by the “More ... ” prompt:

```
Files presently match at file1 line XX and file2 line YY
ZZZ subsequent matching lines in each file compared
```

or

```
Files presently mismatch at file1 line XX and file2 line YY
ZZZ subsequent lines compare in each file without rematch
```

and

```
More...
```

SCOM Examples

The following examples show the summary line that gives the result of the comparison. Summary information is included in the output only when the destination is your terminal, and omitted if the output destination is a disk file.

Example 1:

```
CI> SCOM,TEST1,TEST2
Scom: comparison performed Sat Nov 21, 1992  4:32:17 pm
   file1 is TEST1
   file2 is TEST2
   options: f1f2;  rematchlns: 3;  maxchars: 36
------------------------------------------------------------------------
-- beginning of file --
1           FTN,L
      1     FTN^^
2     2           PROGRAM FTEST
------------------------------------------------------------------------
5     5           COMMON /MISC/ LUT,W1,W2,WBOTH
6                 LUT = 1
7     6           WRITE (1,10)
------------------------------------------------------------------------
16    15    C
17                END
18          $
      16          END$
-- end of file --

3 differences were found.
CI>
```

In the above example, all defaults were taken. Column one represents file1 line numbers, and column two represents file2 line numbers. When the line contents differ, the line number appears in the appropriate file column, and the areas of difference are bracketed with dashed lines. By default, the F1 and F2 options are in effect, and only the differing lines are displayed. Around each area of difference, SCOM adds one preceding and one following line that are common to both files.

Example 2:

```
CI> SCOM,TEST1,TEST2,1,NN
Scom: comparison performed Sat Nov 21, 1992  2:38:58 pm
   file1 is TEST1
   file2 is TEST2
   options: f1f2nn;  rematchlns: 3;  maxchars: 36
-- beginning of file --
------------------------------  file1 only  --------------------------
FTN,L
------------------------------  file2 only  --------------------------
FTN^^
-----------------------------------------------------------------------
      PROGRAM FTEST
      COMMON /MISC/ LUT,W1,W2,WBOTH
------------------------------  file1 only  --------------------------
      LUT = 1
-----------------------------------------------------------------------
      WRITE (1,10)
C
------------------------------  file1 only  --------------------------
      END
$
------------------------------  file2 only  --------------------------
      END$
-----------------------------------------------------------------------
-- end of file --

3 differences were found.
CI>
```

In this example, the NN option suppresses line numbering. In this case, the areas that contain lines that differ are identified by the dashed line headers “file1 only” or “file2 only”. As in the previous example, SCOM adds the preceding and following lines that are common to both files around each area of difference.

Example 3:

```
CI> SCOM,TEST1,TEST2,,F1F2BO
Scom: comparison performed Sat Nov 21, 1992  4:33:16 pm
   file1 is TEST1
   file2 is TEST2
   options: f1f2bo;  rematchlns: 3;  maxchars: 36
1           FTN,L
      1     FTN^^
-------------------------------------------------------------------------
2     2           PROGRAM FTEST
3     3           IMPLICIT INTEGER (A-Z)
4     4           LOGICAL W1,W2,WBOTH
5     5           COMMON /MISC/ LUT,W1,W2,WBOTH
6                 LUT = 1
-------------------------------------------------------------------------
7     6           WRITE (1,10)
8     7     10    FORMAT("N = "')
9     8           READ(1,*) N
10    9           CALL ERROR(N)
11    10          END
12    11          BLOCK DATA MISC
13    12    C
14    13          LOGICAL W1,W2,WBOTH
15    14          COMMON /MISC/ LUT,W1,W2,WBOTH
16    15    C
17                END
18          $
      16          END$
-------------------------------------------------------------------------

3 differences were found.
CI>
```

In this example, the BO option is used to list all lines that are common to both files. When BO is specified, F1 and F2 are no longer defaulted and they must be specified as well to include the lines that differ.

Example 4:

```
CI> SCOM,TEST1,TEST2,,NNF1F2BO
Scom: comparison performed Sat Nov 21, 1992  4:34:04 pm
   file1 is TEST1
   file2 is TEST2
   options: f1f2bonn;  rematchlns: 3;  maxchars: 36
------------------------------  file1 only  --------------------------
FTN,L
------------------------------  file2 only  --------------------------
FTN^^
-------------------------------------------------------------------------
      PROGRAM FTEST
      IMPLICIT INTEGER (A-Z)
      LOGICAL W1,W2,WBOTH
      COMMON /MISC/ LUT,W1,W2,WBOTH
------------------------------  file1 only  --------------------------
      LUT = 1
-------------------------------------------------------------------------
      WRITE (1,10)
10    FORMAT("N = "')
      READ(1,*) N
      CALL ERROR(N)
      END
      BLOCK DATA MISC
C
      LOGICAL W1,W2,WBOTH
      COMMON /MISC/ LUT,W1,W2,WBOTH
C
------------------------------  file1 only  --------------------------
      END
$
------------------------------  file2 only  --------------------------
      END$
-------------------------------------------------------------------------

3 differences were found.
CI>
```

This example shows the NN and BO options together.

Example 5:

```
CI> SCOM,TEST1,TEST2,,D^F1F2BO
Scom: comparison performed Sat Nov 21, 1992  4:35:42 pm
   file1 is TEST1
   file2 is TEST2
   options: f1f2bod^;  rematchlns: 3;  maxchars: 36
1     1     FTN,L
2     2           PROGRAM FTEST
3     3           IMPLICIT INTEGER (A-Z)
4     4           LOGICAL W1,W2,WBOTH
5     5           COMMON /MISC/ LUT,W1,W2,WBOTH
6                 LUT = 1
-------------------------------------------------------------------------
7     6           WRITE (1,10)
8     7     10    FORMAT("N = "')
9     8           READ(1,*) N
10    9           CALL ERROR(N)
11    10          END
12    11          BLOCK DATA MISC
13    12    C
14    13          LOGICAL W1,W2,WBOTH
15    14          COMMON /MISC/ LUT,W1,W2,WBOTH
16    15    C
17                END
18          $
      16          END$
-------------------------------------------------------------------------

2 differences were found.
CI>
```

In this example, the ^ is specified as a wildcard character; thus, the first lines of the two programs now match. Note that the text of matching lines is always read from file1.

Example 6:

```
CI> SCOM,TEST3,TEST4,,F1F2BO,1
Scom: comparison performed Sat Nov 21, 1992  4:42:45 pm
   file1 is TEST3
   file2 is TEST4
   options: f1f2bo;  rematchlns: 1;  maxchars: 16
1     1     ASMB,R,L,C
2     2           NAM TEST3
3     3     *
4           B     EQU 1
      4     A     EQU 0
-------------------------------------------------------------------------
5     5     *
      6     B     EQU 1
      7     *
-------------------------------------------------------------------------
6     8     START  HLT
7     9           END START

2 differences were found.
CI>
```

In this example, two similar files are compared with “rematchlns” set to 1 line, for rematch of identical lines. The difference between the files is the addition of two instructions (lines 4 and 5) to file TEST4:

```
 4   *
5   A    EQU 0
```

In this case, when SCOM compares the files it does not detect the difference, since it finds that line 5 in each file matches. Because of this, SCOM cannot detect the more preferable rematch of line 4 in TEST3 and line 6 in TEST4.

This problem of extraneous identical lines causing erroneous rematches occurs most often with blank comment lines. Example 7 demonstrates a method of avoiding this problem.

Example 7:

```
CI> SCOM,TEST3,TEST4,,C*F1F2BO,1
Scom: comparison performed Sat Nov 21, 1992  4:44:10 pm
   file1 is TEST3
   file2 is TEST4
   options: f1f2boc*;  rematchlns: 1;  maxchars: 16
1     1     ASMB,R,L,C
2     2           NAM TEST3
3     3     *
      4     A     EQU 0
      5     *
-------------------------------------------------------------------------
4     6     B     EQU 1
5     7     *
6     8     START  HLT
7     9           END START

1 difference was found.
CI>
```

In this example, the C* option specifies that all otherwise blank lines with * in column 1 are ignored when searching for a rematch in the files. The result is a more desirable rematch after the mismatch at line 4 in each file.

SCOM Error Messages

Error messages are only displayed at the log terminal; they are not included in the output file. Fatal errors that cause SCOM to abort are identified with (F) following the error description.

WARNING: RECORD TOO LONG

The program read a record whose length is greater than that specified by the MAXCHARS parameter. The record is truncated and the comparison may be invalid. This is only a warning and does not abort SCOM.

#1. USER SUPPLIED MAXIMUM RECORD SIZE IS TOO LARGE

The internal buffer space is insufficient to accommodate the specified MAXCHARS. Rerun SCOM and specify a smaller MAXCHARS value. (F)

#2. ILLEGAL INTERNAL SUBROUTINE PARAMETER

The subroutine was called with an invalid parameter; a utility self-check has failed. Reload SCOM and rerun. (F)

#3. CACHE DATA STRUCTURE CORRUPT

The internal check of the buffer data structure shows corruption; a utility self-check has failed. Reload SCOM and rerun. (F)

Function Key Manipulation Utilities (KEYS, KYDMP)

Two utilities, KEYS and KYDMP, are provided for use in conjunction with display stations that include the function key cluster on the keyboard. KEYS lets you define a specific set of functions to be performed by the keys. For example, you can define a function to output an often used text combination (up to 80 characters), system logon/logoff messages, or text formatting commands. The KYDMP utility programs the keys, termed "softkeys," to perform the functions defined by KEYS.

If you are using an HP 264x series terminal, any softkeys programmed using KEYS and KYDMP can be reset to their default values by pressing the RESET TERMINAL key twice. The current status of the softkeys can be displayed by simultaneously pressing the CNTL and NEXT PAGE keys on the display station keyboard. If you are using another terminal, refer to the user's guide for that terminal for information about softkey use.

KEYS

KEYS lets you define a specific set of functions to be performed by the eight softkeys (f1 through f8) on the display station keyboard. Using KEYS, you can create a set of commands, list an existing command set file, modify the command set, and store the new or modified file on a disk or mini-cartridge.

KEYS is an interactive utility that issues prompts for each step in the creation/modification of a command set.

Calling KEYS

To call KEYS, enter the following runstring:

```
:[RU,]KEYS[,console[,list]][-s]
```

Console is the LU of the display terminal to receive the KEYS prompts. The default is to LU 1 (your terminal).

List is the device on which the command set file is to be listed. The default is to your terminal.

The −s parameter inhibits screen display of softkeys definitions.

KEYS displays the prompt:

```
keys>
```

If you enter M for modify,

```
keys> M
```

the following sequence of prompts is issued:

```
SOFTKEY NUMBER TO BE PROGRAMMED (1-8): 1
```

KEYS then prompts for a label assignment. The label you assign to identify each key's function is displayed at the top of the screen when the keys are programmed by KYDMP.

```
Label (1-16 characters) or [RETURN] for no change: Key 1 LABEL
```

The next prompt asks you to define the command string type:

N[ormal] = command string is transmitted to the computer and echoed at your terminal.

L[ocal] = command string is transmitted in local mode (to your terminal only).

T[ransit] = command string is transmitted to the computer but not echoed, and a carriage return/linefeed is generated.

The default is a transit type command string.

```
TYPE (N[ormal], L[ocal], or T[ransit]): N

/Softkey Text (1-80 characters) or [RETURN]: KEY1TEXT

keys>
```

The L(ist) command displays the current softkey definitions on your terminal. The listing for the softkey created above is

```
keys> 1
Softkey #1-- Label:  KEY1 LABEL      Type:  Normal
Text:  KEY1  TEXT

         :
```

The F(ile) command reads softkey definitions from a previously saved file into the workspace. If no file name is specified, KEYS reports the current file name (the name of the file loaded or the last file written to). To load the definitions from file /SOFT/KEYS:

```
keys> f /soft/keys
keys> f
current file is /SOFT/KEYS:::3:24
```

The W(rite) command writes the current definitions in the workspace to a file or device. If no file name is specifed, the file originally loaded with the F(ile) command or the last file written to is used. To write the workspace to file /SOFT/KEYS:

```
keys> w /soft/keys
```

The EX(it) command terminates keys as follows:

```
keys> ex
C1.69>
```

The KEYS commands are summarized in Table 2-8 below.

Table 2-8. KEYS Commands Summary

Command	Description
F [file]	Loads the current command set from file to be modified. If file is not given, KEYS reports the current file name (the name of the file loaded or the last name written to). `keys> f /soft/keys`
M [key#]	Modify the softkey definition for softkey key#. If key# is not given you are prompted for it. `keys> m 3`
L	List the current softkey definitions to the terminal.
W [file]	Write the current softkey definitions to a file. If file is not given, the last file name used is replaced.
V	Redraw the softkey screen display.
?	Help
EX	Exit

KEYS Error Message

KEYS can generate the following message:

NO FILE NAME HAS BEEN SPECIFIED.

The W command has been given without a file name when no existing file has been read in.

KYDMP

KYDMP programs the display station softkeys to perform the functions defined in the command set generated with KEYS.

To load the terminal softkeys from a file created by KEYS, use FMGR command DU:

```
:DU,KEYFIL,1
```

or use CI command LI:

```
CI> LI,KEYFIL
```

KEYFIL is the name of the file created by KEYS, and 1 is the session LU number of your terminal.

Calling KYDMP

To program the softkeys from a KEYS file stored on disk, call KYDMP by entering the following runstring:

```
CI> [RU],KYDMP,[options,][console,]keyfile
```

The options are as follows:

−L Do not dump labels, just program the softkeys.

−M Do not memory-lock labels at the screen top.

Keyfile is the name of the KEYS file containing the command set.

KYDMP reads the softkey command set from the specified source and outputs it to the specified display station to program the softkeys. The function key labels assigned with KEYS are displayed in inverse video at the top of the screen, as shown in Figure 2-9. If a function key has not been assigned a label, that label field is blank. The display is protected from being written over and remains at the top of the screen. Each time the function keys are reprogrammed, the label field is updated and protected.

Figure 2-9. Softkey Label Display Format

Record Reconfiguration Utility (OLDRE)

OLDRE is a stand-alone utility that provides backward compatibility with loaders and generators that do not accept extended record formats. OLDRE can translate the new RTE-6/VM extended relocatable record formats to the non-extended formats for loaders that require them.

The term “extended record” refers to the set of extended relocatable records which allow 16-character EXT and ENT names and other extended features that are supported by the Macro Assembler and by language compilers. Additional information can be found under Relocatability in the *Macro/1000 Reference Manual*, part number 92059-90001. For other languages, check the appropriate language reference manual for the record forms used.

OLDRE Extended Records

Extended records are identified as 7 in the identifier field (bits 13-15) of record word 2, and with a sub-identifier in bits 6-12, as shown below:

SubIdent	Extended Record	Type
0	GEN	Generator information/command record.
1	XNAM	Extended name relocatable program. Superset of NAM.
2	XENT	Extended entry point. Superset of ENT.
3	XDBL	Extended define left byte address. Superset of DBL.
4	XEXT	Extended external reference. Superset of EXT.
5	XEND	Extended terminate program. Superset of END.
6	RPL	Contains code replacement information.
8	MSEG	Declare EMA memory segment size.
9	ALLOC	Combined ENT/EXT for FORTRAN block data.
10	DEBUG	Contains information for DEBUG/1000.
20	LOD	Contains information/command for Loader/Linker.

Calling OLDRE

To call OLDRE, enter the following runstring:

```
CI> [RU,] OLDRE, file descriptor
```

The file descriptor identifies the relocatable file that combines old format and extended format records to be translated. It must be specified in the OLDRE runstring. The relocatable file must be a type 5 (object program) file.

OLDRE Operation

OLDRE creates a scratch file in which the translated relocatable file is built. Each record is scanned and, if a non-translatable feature is encountered, the record is skipped and an appropriate message is issued.

The program displays the following message when the first extended record is encountered:

```
OLDRE: Converting new relocatable records to old format
```

OLDRE does not terminate on an error, but continues to translate the entire file. If the program encounters non-translatable code, the scratch file is purged after the translation attempt, and the original relocatable file is left unchanged.

If the relocatable file is not type 5, or the file does not reside on a mounted cartridge, OLDRE issues one of these messages and exits:

```
OLDRE: Input must be relocatable file - type 5
OLDRE: Relocatable must be a disk file
```

OLDRE also aborts if either a bad record checksum is encountered, the break flag is set, or an FM error occurs.

After successful translation, the original relocatable file is purged. The scratch file is truncated to the exact size required to hold the translated code, and renamed with the original file name. The resulting translated file can then be loaded to execute under the appropriate operating system.

The OLDRE exit message defines the success or failure of the translation process:

```
OLDRE: End OLDRE - No errors
OLDRE: End OLDRE - Relocatable file unchanged.
```

OLDRE processes Macro/1000 code at approximately 800 lines per second and FORTRAN code at approximately 133 lines per second.

Translation Results

The input instruction to OLDRE and the translated output are listed below.

Input	Output	Comment
NAM	NAM	
XNAM	NAM	Non-zero EMA size, SAVE size, or pure code size will cause an error. Program names longer than five characters will cause an error.
ENT	ENT	
XENT	ENT	An ENT to EMA or externals will cause an error. An entry point name longer than five characters will cause a warning.
EXT	EXT	
XEXT	EXT	Weak external will be translated to a strong reference. Names longer than five characters are warnings.
DBL	DBL	
XDBL	DBL	Byte externals and DDEFs to EMA will cause an error. If the memory area to be initialized is not absolute, common, or program relocatable an error will be issued.
ALLOC	ENT	Translation if ALLOC is to be labeled common in FORTRAN BLOCK DATA. Any DBLS to this ALLOC have external ID number deleted.
ALLOC	EXT	Translation for all other cases except ALLOC to SAVE (illegal). Any DBLs to this external are illegal. Only one symbol per record is legal and it must be a COMMON ALLOC. Note that only program type and name will appear in the NAM record for this module.
MSEG		Error. Will not be translated.

Input	Output	Comment
EMA	EMA	
RPL	ENT	
END	END	
XEND	END	
GEN	GEN	
LOD	LOD	
DEBUG		Ignored. No error or record output.

Program Restrictions

The following restrictions must be observed in preparing the source programs to ensure that the relocatable code can be successfully translated.

Macro/1000

1. External, entry point, and variable names cannot exceed five characters. Names exceeding five characters are truncated to characters 1−4 and the last character.
2. The ALLOC, DDEF, and MSEG commands cannot be used.
3. The RELOC and ORG commands cannot be used with EMA or SAVE, or with pure code.
4. Two-word RPLs cannot be used.
5. DBLs and DBRs cannot refer to externals.
6. No more than 255 externals can be referenced in one module.
7. Negative and multiple relocatability cannot be used (for example, constructs such as DEF −W or DEF W+W, where W is not absolute).

Pascal

1. Level 1 procedure names and program names cannot exceed five characters, unless ALIASed to an external name of less than six characters. Names exceeding five characters in length are truncated to characters 1−4 and the last character.
2. No more than 255 level 1 procedures can be contained in one program.

FORTRAN

1. The SAVE command cannot be used.
2. Subroutine, function, program or variable names cannot exceed five characters. Names exceeding five characters are truncated to characters 1 − 4 and the last character.
3. Local EMA cannot be specified.
4. Labeled EMA common cannot be used.
5. Character data cannot be in labeled common.
6. No more than 255 common block names, external subroutines, or function calls can be contained in one module.

OLDRE Error Messages

The following messages describe conditions encountered during the translation process and define errors that cause cancellation of the translation.

OLDRE: Local EMA, SAVE and CODE area illegal

OLDRE: New feature not available in old format

OLDRE: New feature in RPL illegal in old record

OLDRE: More than 255 externals found

The following message is treated as a warning and, if the only problem, does not cancel the translation:

OLDRE: Symbol name truncated to 5 characters

Note that in this case, the load may fail due to duplicate externals caused by the loss of the truncated characters.

3

Online/Offline Physical Backup Utilities

RTE-6/VM provides several physical disk image backup utilities for backing up and verifying disk LUs. This chapter describes these utilities and their use.

Using the Utilities

The online and offline physical backup utilities PSAVE, PRSTR, and PCOPY are used to transfer data between disk and tape transports or tape cartridges. The data transfer is on a track-by-track basis and does not assume that the disk subchannel contains valid FMGR information. You can specify the transfer of all tracks on a named subchannel or all subchannels on a named disk unit. Track sparing is provided and mounted FMP cartridges are protected.

PSAVE (SA in offline mode) saves disks to tape, PRSTR (RE in offline mode) restores the saved tapes to disks, and PCOPY (CO in offline mode) copies data from disk to disk. In offline mode, the program BCKOF schedules the appropriate backup utility and provides the help function and I/O configuration information.

The offline mode basically supersedes online mode by allowing track-map table modification. Unit saves also may be specified in pushbutton-restorable format, either offline or online, if the destination device is a CS/80 tape cartridge and the source is a CS/80 disk.

Compatibility with Other Disk Backup Utilities

The RTE-6/VM physical backup utilities replace the ICD/MAC and HP7900 backup utilities. Tapes created by PSAVE are not compatible with ICD/MAC and 7900 utilities; however, PRSTR can read tapes created in either the ICD/MAC or 7900 format. The physical backup utilities are not compatible with tapes created by READT/WRITT, READR/SAVER, FC, or other file backup utilities.

Compatibility Among Disk Drives

The RTE-6/VM physical backup utilities do not use any file structure information. This means that source and destination disks used in copy and restore operations must have the same track size (except as noted below). If you must transfer between disks with different track sizes, use the FC file backup utility to transfer the files individually or in groups, rather than on a track-to-track basis.

The track sizes in the subchannel definitions for CS/80 disks are logical track sizes and may not be equal to the physical track size of the disk. Using the offline utility, any CS/80 disk can be restored to any other CS/80 disk regardless of the physical track size. The online utility must protect the system integrity by not allowing a disk to be restored if it cannot be referenced properly by the existing system subchannel definitions.

Table 3-1 defines the source and destination compatibilities of the various disk drive models.

Table 3-1. Disk Drive Source/Destination Compatibility

	Destination						
Source	**7900**	**7905**	**7906**	**7910**	**7920**	**7925**	**CS/80**
7900	OK	*	*	N/A	*	N/A	+
7905	*	OK	OK	N/A	OK	N/A	+
7906	*	*	OK	N/A	OK	N/A	+
7910	N/A	N/A	N/A	OK	N/A	N/A	+
7920	*	*	*	N/A	OK	N/A	+
7925	N/A	N/A	N/A	N/A	N/A	OK	+
CS/80	+	+	+	+	+	+	OK

CS/80 Disks — Includes the 7907, 7908, 7911, 7912, 7914, 7933, 7935, 7936, 7937, 7941, 7942, 7945, 7946, 7957, 7958, 7959, 7962, 7963, 9122, 9133, 9134, 9153, 9154, C2200, C2202, and C2203.

N/A — Not available.

* — If restore is offline, you must supply subchannel definition.

\+ — Logical track sizes for the CS/80 sealed disks have been set at 96 sector per track. This is variable, so any CS/80 source can go to any CS/80 destination offline. Online operations use the current subchannel definitions.

Note Because PRSTR only knows the disk types returned by LDTYP, you must be very careful when restoring a tape offline using the track map definitions on the tape; Table 3-1 should be consulted. The offline option UD (refer to the section Restore Tape File <RE> for details), forces you to specify the track map definitions. This is automatic when the disk types (CS/80, MAC, ICD) are different.

Tape Handling

Tape handling considerations differ between tape transports and CS/80 cartridges because the cartridge is a streaming mode, blocked record device, and the transport is a random-length record storage device. (Refer to the section "Data Structures" for details of the tape formats.)

Efficient handling of cartridge tape is provided by the CS/80 device driver disk-caching routines that control online transfers to the cartridge. Data to be transferred is buffered on the integrated disk drive and transferred in a steady stream to the cartridge tape, thus allowing continuous movement of the tape. Since caching is not possible offline, PB (pushbutton) saves and restores are substantially faster in this mode. (Refer to the *RTE-6/VM DVM33/DVN33 Driver Reference Manual*, part number 92084-90025, for details of the disk-caching scheme.)

When the source or destination for the backup operation is a CS/80 cartridge tape (CTD), the utility prompts with:

```
DEFINE CTD LU SUBCHANNEL -

CURRENT CTD DEFINITION IS:

ADDRESS, UNIT#, VOLUME#,

   0,      1,      0,

ADDRESS, UNIT#, VOLUME#,
```

and waits for your response. Enter the correct configuration followed by a carriage return. The HP-IB address for the CTD is set using the address switch at the rear of the drive unit. Refer to the operation and installation manual provided with your CS/80 drive for details of the switch settings. (Note that UNIT# and VOLUME# appear only if the operation is an offline backup.)

Tape Positioning

When the backup device is a tape transport, the save and restore functions start at the current tape position unless a file number is specified. If specified, the number is interpreted as relative to the start of the present tape, and the tape is rewound. (Note that the desired file number need not be on the currently mounted tape volume of a multi-volume set.) For the SE (selective restore) and UN (unit restore) options, all subsequent file numbers are relative to this initial positioning. Since the tape is not rewound after a save or restore, you can save more than one LU on a tape. Online, you can use the FMGR CN command (control non-disk devices) to position the tape before an

operation. Offline, the tape movement commands are a part of the utility tasks that can be specified.

If an end-of-tape mark (EOT) is detected during an operation, a message is issued and the utility waits for you to mount a new tape and enter the command to continue.

When the backup device is a CS/80 cartridge, PSAVE always assumes the save is to start at the first logical file on the cartridge unless you specifically designate a file number greater than one. Since file seeks on cartridges always start at block 0, it is advisable to use the multiple LU save option (MU) if multiple saves are going to the same cartridge. End-of-cartridge is handled the same as for tape transport EOT.

Caution If a magnetic tape position is specified beyond the end of all data on the tape, a tape runaway condition may be encountered.

The last save written by PSAVE is followed by two end-of-file (EOF) marks. This constitutes an end-of-data mark to the utilities. A warning is issued if you specify a file beyond a double EOF mark.

Unit Save Tape Files

In a unit save, which is simply a series of LU saves, each disk subchannel is saved on tape as one file. LU saves are assigned sequential tape file numbers relative to the current position of the tape: last save file# + 1, or file #1 if the tape was rewound or is a CS/80 cartridge. The save file numbers on the tape are thus independent of the LU number of the subchannel saved to tape. As a result, you must specify the tape file number, not the subchannel number, when you are doing a selective restore either online or offline.

For any save not in pushbutton-restore format (PB option), on a CS/80 cartridge, the cartridge contains an EOF mark at the beginning of the tape. The EOF mark prevents inadvertent restoring of the cartridge using the front-panel pushbutton controls, since the manual restore operation stops at file marks. The empty file is interpreted as file 0; thus, the first save file on a CS/80 cartridge is file 1, just as it is for tape transports.

Multiple-Volume Tape Sets

Volumes in a multi-volume tape set are numbered consecutively, and each volume begins with a full set of headers. If a single LU save crosses over tapes, the save definition header contains the track and sector at which the save resumes on the tape. If the destination is a tape cartridge, the current checksum record is written in the last two blocks of the cartridge.

Data Verification

The online utilities PSAVE and PRSTR both provide a data verification option (VE). The offline commands SA and RE also allow the VE option. Data verification is identical for both offline and online saves and restores, except for PB operations. When the operation is a restore, specifying the VE option also causes the destination disk tracks to be spared in a prepass over the disk (except CS/80 disks).

Verification of Saves

Verification of saves is done in two steps: 1) verify that the disk was read correctly and 2) verify that the tape was written correctly.

The first step is accomplished during the save operation. When the disk is read, a checksum is calculated. The disk is then re-read and a new checksum is calculated. If these two checksums agree, the data is assumed to be correct. If they do not agree, the above process is repeated until two successive reads yield the same checksum or until the calculation has been tried ten times. If no two reads are the same, the disk is read sector by sector to recover as much of the save data as possible.

The second step, verifying the tape write, is accomplished in a second pass, which occurs when the last save in a series is completed or when a new tape must be mounted. The utility rewinds the tape whenever possible, but if the first save in the series did not start at the beginning of the tape, the files are backspaced to the beginning of the first save. Each tape record is verified by testing for a correct checksum. Note that the save utility does not recognize the break flag (BR) during the verify pass.

Verification of Restores

Verification of restores is done in two steps: 1) verify that the tape was read correctly and 2) verify that the disk was written correctly.

The first step is accomplished by testing for correct checksums as the tape records are read. In the second step, the disk is re-read when it is written to determine if the correct data was transmitted to the disk. This process is repeated up to ten times; if it is still not successful, a sector-by-sector write is done in an attempt to correctly restore as much data as possible. Note that the restore is verified as the disk is written; the tape does NOT need to be re-read in a second pass.

Disk operations may cross track boundaries. The utility that reports an error does not know this and thus reports only the track number where the transfer started.

When sparing the reported track, check with either FORMT or FORMC to be sure it is the spared track (not a track following the reported track).

Pushbutton Save/Restore Data Verification

Data integrity for pushbutton save/restore operations is verified using the cyclic redundancy check performed by the CS/80 controller. If, however, a bad region exists on the media (tape or disk), the PB operation could fail. The VE option, in conjunction with a PB restore, ensures that a verify pass will take place on the destination disk after the restore operation.

On a PB save, the VE option causes a verify pass to take place on the tape at the completion of the save operation. If a bad block should occur, the number is reported and a verify error is returned. (Sparing is automatic if a data error occurs.) Thus, if a PB save fails to verify the data, repeating the operation will not fail to verify the restore operation for the block.

Pushbutton (PB) Operations

The physical backup utilities are capable of emulating the offline cartridge tape backup capability of many of the CS/80 disk drives. This manual backup operation is referred to in this section as a PB (PushButton) operation.

The HP 7908, 7911, 7912, 7914, 7942, and 7946 disk drives in the standard configuration have integrated cartridge tape devices (CTDs) and can perform PB operations either manually or by using the utilities. However, there are options with these disk drives that will delete the cartridge tape device or provide separate controllers for the disk and CTD. The information in this section does not apply to these configurations nor to the HP 7907, 7933, 7935, 7941, and 7945 disk drives, which do not have an integrated CTD for backup. None of these drives is capable of performing a PB operation either manually or by using the utilities.

A PB operation on a CS/80 disk with CTD is performed manually from the front panel of the disk drive. This operation is also emulated from the physical backup using the PB option when saving or restoring. Physical backup utilities provide the capability of a post save or restore verify pass (and in the case of a restore, a pre-verify and spare pass) if you specify the VE option. Some devices are capable of performing a PB save or restore using two tapes, and this feature is also emulated by the physical backup utilities.

The PB save or restore operation from the utilities is merely a command to the device controller to perform a copy from a source to a destination for a unique length. This means that if you specify a disk LU that is on a different bus or a different bus address than the cartridge tape device LU, the operation cannot be performed. The disk and the CTD must be integrated on a single controller, at a single HP-IB address on a bus, in order for the PB operation to take place.

Please note the capacities of the tapes that you are using as specified in the operation and installation manual supplied with your disk drive. If you are using a 64-megabyte disk drive, you will need a cartridge tape capable of holding 64 megabytes to perform a PB backup of the entire disk.

The following example demonstrates the use of the PB option for a save operation. In the example, the save operation completes successfully after invoking auto-sparing to spare a block that had been found bad in some earlier operation to the tape. On the post verify pass, one block was found to require a retry to get the correct data (as determined by internal CRCs). Note that status messages are preceded with “CS/80 NOTE:”; errors are always preceded with “CS/80 ERROR”.

```
:RU,PSAVE
ENTER OPTIONS
vepb                                ! Verify data, PushButton operation
ENTER SOURCE DISK LU
2                                   ! CS/80 Disk
ENTER TAPE LU
13                                  ! CS/80 CTD on same bus & HPIB address
ENTER HARDCOPY LU
1                                   ! Default to terminal scheduling PSAVE

SCHEDULING PUSHBUTTON OPERATION   ! PSPAR being scheduled

COPY FROM DISK TO TAPE PROCEEDING.! Save has started.
CS/80 NOTE:  AUTO SPARING WAS INVOKED.  NO DATA WAS LOST.
CS/80 - FULL DEVICE STATUS:
000060B: IDENTIFICATION FIELD.
PARAMETERS - 000000B, 000000B, 130201B, 239481B, 293212B.
(SEE DVM33/DVN33 REFERENCE MANUAL FOR MORE INFORMATION.)

VERIFYING TAPE.
CS/80 - FULL DEVICE STATUS:
000020B: IDENTIFICATION FIELD.
PARAMETERS - 000000B, 000000B, 001010B, 121212B, 121212B.
(SEE DVM33/DVN33 REFERENCE MANUAL FOR MORE INFORMATION.)
COPY COMPLETED SUCCESSFULLY.
PSAVE NORMAL END OF JOB.
```

The following is an example of using the PB option in a restore operation. There are no errors on either the pre-verify and sparing pass, the restore, or the post-verify pass on the disk.

```
:RU,PRSTR
ENTER OPTIONS
pbve
ENTER DEST DISK LU
14
ENTER TAPE LU
13
ENTER HARDCOPY LU
<cr>

SCHEDULING PUSHBUTTON OPERATION

COPY FROM TAPE TO DISK PROCEEDING.

VERIFYING DISK.
COPY COMPLETED SUCCESSFULLY.
PRSTR END OF JOB.
```

Using the BReak Command

Both the online and offline utilities check the break flag prior to each tape or disk I/O operation. The current operation is ended and all resources are released when a break is detected. The one exception is during data verification, when the break condition has no effect.

The online backup utility terminates when the break flag is detected, while the offline utility returns to the interactive TASK? prompt state. When the BR command is used, the interrupted operation must be considered invalid.

Breaking a save causes the utility to write double EOFs at the end of the current save to mark the end of data on the tape.

Online Physical Backup Utilities (PSAVE, PRSTR, PCOPY)

PSAVE, PRSTR and PCOPY run under control of RTE-6/VM; thus, the logical units and disk subchannel definitions are determined by the tables in the current system. Each of the utilities can be run interactively, by using a command runstring, or by means of a command file. When the utilities are invoked using a runstring and you specify an erroneous parameter, the utilities enter interactive mode and prompt for the correct parameter.

Restoring the Disks

PRSTR and PCOPY lock the disk being restored and will not restore to an already mounted FMP cartridge. To restore a disk LU online, you must first dismount the cartridge with the FMGR DC command. To restore a disk unit online, you must dismount all LUs on that unit. However, PRSTR issues a message listing the mounted LUs and gives you the option of restoring those LUs that are not mounted. If all cartridges are mounted, PRSTR terminates with an error message after listing the LUs that must be dismounted.

Note, however, that the system and auxiliary disks LU 2 and LU 3 cannot be dismounted, and any attempt to do an online LU, UNit, or SElective restore to these LUs will result in an error. A UNit or SElective restore will attempt to restore those LUs that were dismounted and lockable, if you respond with Yes to the OK TO PROCEED (Y or N)? prompts. The tape files containing the PSAVE of LU 2 and LU 3 will be skipped in the restoration process.

The PushButton option does allow you to restore LU 2 and LU 3 online. When the following warnings are printed, you must answer Y(es) to all the questions before the restore will take place:

```
*WARNING* SYSTEM LU <n> WILL BE OVERLAYED
OK TO PROCEED (Y or N)?

******  W A R N I N G  ******
ALL ACTIVITY MUST BE TERMINATED BEFORE RESTORE PROCEEDS.
OK TO PROCEED (Y or N)?
```

The first question shown above may be suppressed using the DE option, but you must answer the second one interactively. You can enter EX or /E to any of these questions and the utility will terminate. After the restoration on the system disk completes, the system will be HALTed to let you reboot. This is because the system on the disk and the system in memory cannot be guaranteed to be the same. If PRSTR is executing under Session environment, you must have a capability level of at least 60 to do an online PushButton restore to the disk containing LU 2 or LU 3.

A unit restore in CS/80 pushbutton format is not restorable selectively since the pushbutton controls on the drive unit allow only full save or full restore. Therefore, all cartridges on the CS/80 disk must be dismounted before a data restore can be done.

Loading the Online Utilities

You can load the utilities by using the LINK loader and the LINK command files PSAVE.LOD, PRSTR.LOD, PCOPY.LOD, and PSPAR.LOD.

PSAVE

PSAVE saves one disk LU, a group of disk LUs, or an entire disk unit to tape transports or tape cartridges. As an option, you can save an entire disk unit in CS/80 pushbutton-restorable format.

Calling PSAVE

To call PSAVE enter the following runstring:

```
:[RU,]PSAVE[,input[,srceLU[,destLU[,file#[,opts[,hcpy[,tape density[,title]]]]]]]]
```

In the runstring, input is the LU (or file) from which the PSAVE parameter inputs are to be read. If input is from a file or a non-interactive device, no other parameters may be specified. The default is to your log device.

The srceLU is the LU of the disk subchannel to be saved. If the UN (unit save) option is specified, the srceLU parameter is a target to the unit and may be any LU on the disk. If the MU (multiple save) option is specified, this parameter is meaningless and will be ignored. In any other case, this is a required parameter.

The destLU is the LU of the tape transport or tape cartridge to receive the saved data. Under session monitor, the LU specified must be in your Session Switch Table (SST). The default is to LU 8.

File# is the integer number to be assigned to the saved file. This parameter specifies the start location on the tape for the save. Before writing, file# − 1 tape files are skipped from the start of the tape. The default is to the current tape position for tape transports or file #1 for CS/80 cartridges.

Opts are any of the following two-character ASCII option codes. The options can be specified in any order and may not have intervening characters (including blanks) between options. If no options are specified, the default is to LU save without verify.

VE	Turn on verify option. Track sparing is done only when the verify option is specified. PSPAR and FORMT must be loaded into memory for this option.
LU	Save is to be an LU save.
UN	Save is to be a disk unit save. The source LU parameter can be any LU on the disk unit.
PB	Unit save in CS/80 pushbutton-restorable format. The destination LU must be a CS/80 cartridge, and the source LU must be a CS/80 disk. PSPAR must be loaded into memory for this option.
MU	Multiple LU saves in one pass. The source LU parameter should be omitted, PSAVE will prompt for the source LUs. All LUs must be from the same class of disks, such as all MAC disk LUs.

The hcpy is the LU of the device on which information about the save is to be printed as a record of the save operation. The default is to your log device.

The tape density must be 800, 1600, or 6250. The default is to use the last tape density specified.

The title is the title (to a maximum of 40 ASCII characters) that will be placed in the tape header.

If you enter at least one parameter in the runstring, PSAVE defaults all optional parameters and prompts you for all required parameters not included in the runstring.

Calling PSAVE Interactively

To run PSAVE interactively, enter the utility call PSAVE or RU,PSAVE. Enter a valid response to each prompt, or enter a carriage return to use the default value. If a response is not valid, an appropriate error message is issued and the prompt is repeated. PSAVE prompts for input in the following order. (Note: This is the order that should be used in preparing an input command file.)

Prompt	Response
`ENTER OPTIONS`	Enter any of the above option codes. Since the UN, LU, PB and MU options are mutually exclusive, only two options are possible, at most. The options must be specified without any intervening characters, including blanks. (If you respond with "??", PSAVE will display a list of available options.)
`ENTER SOURCE DISK LU`	If the selected option is UN or PB, this parameter can be any LU on the disk unit. If the selected option is LU, this should be the LU to be saved. If the selected option is MU, a string of LUs may be entered, separated by commas. To continue the list on a second line, end the line with a comma.
`ENTER TAPE LU`	Enter the LU of the destination tape transport or CS/80 cartridge. Enter a carriage return to specify the default, tape LU 8.
`ENTER FILE NUMBER`	Enter the file number to specify the point at which the save is to begin on the tape. (File# − 1 files are skipped.) Enter a carriage return to specify the defaults: current tape position on tape transport, file #1 on CS/80 cartridge.
`ENTER HARD COPY LU`	Enter the LU of the device on which the save operation information is to be printed. Enter a carriage return to specify the default LU, the log device.
`ENTER TITLE`	Enter up to 40 ASCII characters of label information to be stored in the tape header. Enter a carriage return if no title is desired. The title may be entered in uppercase characters, lowercase characters, or a combination of both.
`ENTER TAPE DENSITY`	Enter the tape density to be used (800, 1600, or 6250). Enter a carriage return to specify the default, the tape density last specified for the tape. (This prompt is only given if the tape drive is an HP 7974A, 7978A/B, or 7980A magnetic tape drive and the drive is positioned to the beginning of the tape.)

A response of AB, /A, EX or /E to any of the above prompts causes the program to terminate cleanly, releasing all resources. Once the program has started processing the tape, it can be

terminated by entering break mode (hitting any key on the keyboard) and entering BR,PSAVE. Do not use the OF command to terminate PSAVE.

Before beginning the save operation, PSAVE checks that the tape is online and write-enabled. If either condition is not met, the appropriate error message is issued and the utility prompts with:

```
TYPE 'GO' WHEN READY ('PA' TO SUSPEND)
```

When the tape is online and write-enabled, enter GO and PSAVE will continue. If you enter PA, the utility suspends itself after issuing the message:

```
TYPE GO, PSAVE TO CONTINUE
```

PSAVE prints the tape number, file number (0 if the current tape position is not at the beginning of the tape), and the tape header records on the log device and the hard-copy LU (if specified) for each save as it is processed. If any serious errors are encountered, the utility terminates with an error message.

If PSAVE is run from a command file, information is returned to the father program by a PRTN call. The first parameter is either a zero, signifying valid completion, or an error number. (The error number returned corresponds to the error numbers assigned to the error messages.) If the scheduling program is FMGR, the error is returned in the 1P global. If you are under session monitor, an error is posted to the SCB. The error number and the subroutine in which the error occurred are displayed on the log device.

PSAVE Examples

In the following example, PSAVE is invoked to perform multiple LU saves in a single pass (MU option) and to verify the saved data. Although the source LU is specified in the runstring, PSAVE ignores it and prompts for the parameter. Two files are specified in the prompt response; however, the MU option also can be specified to save a single LU. The destination is defaulted to tape transport LU 8, and the starting file is specified as file #1. The title "PBU Test 091581" is added to the tape header for both files, and the tape density is defaulted. In this example, the TAPE UNIT OFFLINE message is issued after PSAVE checks the status of the destination LU.

```
:RU,PSAVE,,,,,VEMU,1,,PBU Test 091581

TAPE LU DEFAULTED TO           8
ENTER DISK LU(S)
45,61

TAPE UNIT OFFLINE
ENTER "GO" WHEN READY ("PA" TO SUSPEND)
go

SAVING DISK LU   45 TO FILE   1  TAPE    1

TAPE NUMBER:   1   SAVE FILE:   1   USER: BILL.JT
PROGRAM:  PSAVE    OPTIONS:  VE  MU   DATE: WED 9 SEPT, 1988
DISK LU:  45       TITLE:  PBU TEST 091581
SECTION:   1       (TRK    0    SEC     0)

LU 45 SAVED ON FILE   1   TAPE   1

SAVING DISK LU   61 TO FILE   2  TAPE    1

TAPE NUMBER:   1   SAVE FILE:   2   USER: BILL.JT
PROGRAM:  PSAVE    OPTIONS:  VE  MU   DATE: WED 9 SEPT, 1988
DISK LU:  61       TITLE:  PBU TEST 091581
SECTION:   1       (TRK    0    SEC     0)

LU 61 SAVED ON FILE   2   TAPE   1

PSAVE NORMAL END OF JOB
   :
```

In the following example, the FMGR command CL is used to first obtain a listing of all available LUs, and PSAVE is then invoked interactively. The "??" response to the ENTER OPTIONS prompt causes PSAVE to display all of the options allowed with this utility. The VE option is then specified to verify the LU save (the default if the LU, UN, or MU option is not given). Since the file number is defaulted, the number assigned to the saved LU will be determined by the current position of the magnetic tape on LU 8.

```
:cl

     LU   LAST TRACK  CR   LOCK  P/G/S

     14      00202      L           P
     32      00202      S           P
     19      00202      O           P
     22      00202      E4          P
     37      00202      PV          P
     16      00202      CS          G
     02      00255   00002          S
     03      00255   00003          S
     50      00424   00050          S
     51      00202   00051          S


:psave
 ENTER OPTIONS
 ??
 ALLOWED OPTIONS (SAVE):
 (1) VE => VERIFY
 (4).LU => LU SAVE/RSTR/COPY
 (4) UN => UNIT (FULL VOL) SAVE/RSTR/COPY
 (4) PB => PUSHBUTTON UNIT IMAGE SAVE/RSTR
 (4) MU => SAVE OF MULTIPLE LUS

THE NUMBERS IN ()'S REPRESENT CATEGORIES AND
ONLY ONE OUT OF EACH CATEGORY CAN BE USED.

ENTER OPTIONS
ve
ENTER SOURCE DISK LU
37
ENTER TAPE LU
8
ENTER FILE NUMBER
<cr>

ENTER HARD COPY LU
6
ENTER TITLE
This is my private cartridge.


 SAVING DISK LU  37 TO FILE   1 TAPE   1
 TAPE NUMBER:   1  SAVE FILE:     1 USER: BILL.JT
 PROGRAM: PSAVE    OPTIONS: VE  LU  DATE: 10:36 AM TUE., 18 MAY,1988
 DISK LU:    37    TITLE: This is my private cartridge.
 SECTION:     1    (TRK      0  SEC      0)

 LU  37 SAVED ON FILE   1 TAPE   1

 VERIFYING TAPE   1

 PSAVE NORMAL END OF JOB
:
```

PRSTR

Note When restoring tapes saved using the ICD/MAC or HP 7900 utilities, be aware that the USAVE and SAVE UNIT utilities store a disk unit as a single tape file. These unit saves are therefore not selectively restorable, and efforts to position the tape beyond the start of the save will fail.

PRSTR restores a single subchannel LU, a group of LUs, an entire disk drive unit, or selected files from a tape transport or tape cartridge previously saved by PSAVE. This utility also can be used to read tapes written in SAVE, LSAVE, and USAVE formats. When the verify option is selected, PRSTR spares the destination disk tracks in a prepass over the disk before data is written. (Sparing is not done for 7900 disks.)

Calling PRSTR

To call PRSTR, enter the following runstring:

```
:[RU,]PRSTR[,input[,dest LU[,srce LU[,file#[,opts[,hcpy]]]]]]
```

In the runstring, input is the LU (or file) from which the parameter inputs are to be read. If the input is from a file or a non-interactive device, no other parameters may be specified. The default is to the log device.

The dest LU is the LU of the disk subchannel on which the data is to be restored. If the UN (unit save) option is specified, the LU is a target and may be any LU on the disk. If the selective restore (SE) option is specified, this parameter must be omitted. In all other cases, the parameter is required.

The srce LU is the LU of the tape transport or CS/80 cartridge from which the save file is to be read. Under session monitor, the LU specified must be in your Session Switch Table (SST). The default is LU 8.

The file# is the integer number of the file to be read. File# − 1 files will be skipped on the tape before the read begins. The default is the current tape position for tape transports, or file #1 for CS/80 cartridges.

The opts are any of the following two-character ASCII option codes. The options can be specified in any order, and may not have intervening characters (including blanks) between options. If no options are specified, the default is either LU restore with no VErify if the save was an LU save, or UN if the save was a unit save.

VE	Turn on data verify option. PRSTR also spares the destination disk tracks in a prepass over the disk before data is written. PSPAR and FORMT must be loaded into memory for this option.
DE	Do not prompt for verification of tape file header. Do not prompt for OK TO CONTINUE? when disk source and destination sizes do not match. (PRSTR will continue with the restore.)

UN	Unit restore. This is the default if the tape is a unit save or a from-to save. (Note that from-to saves performed using !DSKUP must use the UN option to restore the save.)
LU	LU restore. This is the default if the tape is an LU or MU save.
SE	Selective restore from a unit save, or multiple LU restores in one pass. If this option is selected, the destination LU parameter must be omitted; PSAVE will prompt for the file:LU pairs. Not valid for a unit save in pushbutton-restorable format or a unit save in formats other than those created by PSAVE. All LUs must be of the same disk class. If the 'VE' option is also selected, restore only one LU at a time to avoid conflicting LU locks by PRSTR and FORMT.
PB	Unit save in CS/80 pushbutton-restorable format. PSPAR must be loaded into memory for this option.

The hcpy parameter is the LU of the device on which information about the restore is to be printed. The default is to your log device.

If you enter at least one parameter in the runstring (beyond the input parameter), PRSTR defaults all of the optional parameters and prompts for the remaining required parameters not included in the runstring.

If the SE option is selected, PRSTR prompts with the message:

```
ENTER FILE:LU PAIRS
```

and waits for you to enter the file number and destination LU. The file numbers can be entered in random order; they are reordered and restored in ascending order. A string of file/LU pairs may be entered, using a comma to separate the entries. To continue the list on another line, end the current line with a comma. If duplicate file/LU pairs are entered, only the first entry pair is restored; duplicates are ignored.

Calling PRSTR Interactively

To run PRSTR interactively, enter the utility call PRSTR or RU,PRSTR. Enter a valid response to each prompt or enter a carriage return to use the default value. If a response is not valid, an appropriate error message is issued and the prompt is repeated. PRSTR prompts for input in the following order. (Note: This is the order that should be used in preparing an input command file.)

Prompt	Response
`ENTER OPTIONS`	Enter any of the above option codes. Since the SE, PB, UN, and LU options are all mutually exclusive, only three options are possible, at most. The options must be specified without any intervening characters, including blanks. (If you respond with "??", PRSTR will display a list of available options.)

If you select the UN, PB or LU option:

`ENTER DEST DISK LU`	If the PB or UN option is selected, this parameter can be any LU on the disk drive unit. If the LU option is selected, this must be the destination LU for the restore operation.

If you select the SE option:

`ENTER FILE:LU PAIRS`	Enter the restore file number and the destination LU, separated by a colon. File numbers can be entered in random order; they are reordered and restored in sequence. A string of entries must be separated by commas. To continue the list on another line, end the current line with a comma. If a file number:LU pair is duplicated, only the first entry is restored; subsequent entries are ignored.
`ENTER TAPE LU`	Enter the LU of the source tape transport or CS/80 cartridge. Enter a carriage return to specify the default tape transport LU 8.

If you do not select the SE option:

`ENTER FILE NUMBER`	Enter the number of the saved file to be restored. Enter a carriage return to specify the defaults: current tape position on tape transport, file #1 on CS/80 cartridge. If the UN option is selected, the tape must be positioned at the first file in the original unit save or an error will be returned.
`ENTER HARD COPY LU`	Enter the LU of the device on which the restore operation information is to be printed. Enter a carriage return to specify the default LU, your log device.

A response of AB, EX, /A. or /E to any of the above prompts causes the program to terminate cleanly, releasing all of its resources. Once the program has started processing the tape, it may be terminated by entering break mode (hitting any key on the keyboard) and entering BR,PRSTR. The OF command should not be used to exit PRSTR.

Before restoring any files, PRSTR reads the first record header and reports those LUs that cannot be restored because they cannot be locked, or they are mounted FMP cartridges, or, in the case of a unit restore, there is no subchannel on the target disk corresponding to that on the tape. PRSTR issues a prompt giving you the option of aborting the operation or continuing with those saves that can be done.

When restoring to any disk unit, all drive protect switches must be turned off and the format switch must be turned on. (Since PRSTR does not recognize upper- and lower-platter protect, both protect switches must be turned off when restoring a 7905 or 7906 disk LU.) If either condition is not met, an error message is issued and PRSTR prompts with:

```
TYPE 'GO' WHEN READY ('PA' TO SUSPEND)
```

When the switch setting is correct, enter GO and the program continues. If you enter PA, the program suspends itself after issuing the message:

```
TYPE GO, PRSTR TO CONTINUE
```

After the tape header is printed on the log device, PRSTR issues a prompt asking you to verify that this is the correct file (unless the DE option was selected). If your response is YE, the tape file is restored to disk. If your response is NO, PRSTR rewinds the tape and again requests the file number. The file number entered is now relative to the start of the present tape. The tape is advanced the entered number of files minus one, and the header is printed. PRSTR again asks for verification and, after a YE response, restores the file to the original disk LU. Subsequent file advances on SE or UN restores will then be relative to this new position.

The rewind feature allows you to position the tape to a file preceding the one that was rejected. This capability should be used with caution in SE and UN restores. Specifying a file number using the reject/select process will impact all subsequent restores, since files are restored by their position on the tape relative to the first file and are restored to the absolute LU specified when PRSTR is invoked.

Suppose, for example, that a tape contains four files, and you specify an SE restore using the pairs 1:20 and 2:22. You then find when the first file header is displayed, that you really wanted to restore the second file. You can either enter /E to terminate PRSTR, or respond to the file number prompt with “2” to bypass file #1 and restore file #2 as the first file. In this case, file #2 will be restored to LU 20, and the next file on the tape (#3) will be restored to LU 22. Further, if the original SE pairs were 1:20 and 3:22, repositioning the first restore to file #2 would cause file #4 to be restored to LU 22, and file #3 would not be restored.

Note, however, that PRSTR prompts with the file verification message and the prompt OK TO PROCEED? This allows you to reposition the tape to absolute file #3, if desired, and continue.

On all restores, tapes must be mounted sequentially, and no tape volumes in a set may be skipped. Internal checks are made to ensure this proper sequencing of tapes.

As each restore is processed, PRSTR displays the tape number, file number and the tape title information on the log device. If fatal errors are encountered, PRSTR terminates with an error message.

If PRSTR is run from a command file, information is returned to the father program by a PRTN call. The first parameter is either a zero, signifying valid completion, or an error number. (If the scheduling program is FMGR, the error number is returned in the 1P global.) If you are under session control, an error is posted to the SCB.

Caution Be aware that the directory tracks might not be copied when a disk LU with FMGR files and directory tracks is restored to a destination disk LU with fewer tracks than the source disk. If the directory tracks are not written, the restore operation will not be valid.

If the source disk LU saved on the tape file and the destination disk LU do not have the same number of tracks, PRSTR issues the message:

```
xxxxx TRACKS IN SOURCE LU
yyyyy TRACKS IN DEST LU
OK TO PROCEED? (YE/NO)
```

If you respond YE, PRSTR copies tracks until it has filled the last track of the destination LU or has restored the last track of the source file LU, whichever is smaller. (The question is not asked if the DE option is specified, but the condition is reported.)

When restoring to a destination disk LU with more tracks than the source disk LU, you must use the FMGR MC command and specify the number of tracks restored from the source disk LU to allow FMGR to locate the correct directory tracks.

PRSTR Examples

In the following example, the SE (selective restore) option is specified, and PRSTR prompts for the file/LU pairs. In this example, the first file is continued on another volume of the tape set, so PRSTR issues the MOUNT NEXT TAPE message and waits. Note that the file/LU pairs are specified separated with commas. If the list were to continue on a second line, the first line would end with a comma. The XXXXX-XXXXX entry in the file header display signifies the PSAVE and PRSTR software part numbers.

```
:RU,PRSTR,,12,,1,SE

 ENTER FILE:LU PAIRS
 2:28,3:28
CREATED USING:   PSAVE   92084-16656 REV.2121  < 810905.1311 >
           READ USING:      PRSTR   92084-16657 REV.2121  < 810714.0904 >
           TAPE NUMBER:       2     SAVE FILE:   2    USER: BILL.JT
           PROGRAM: PSAVE           OPTIONS:    MU    DATE: WED,9 SEPT. 1988

           DISK LU: 28 TITLE: PBU TESTING PRSTR
           SECTION: 1 (TRK 0 SEC 0)

 OK TO PROCEED? (YE/NO)
 ye
 RESTORING DISK LU 28

 MOUNT NEXT TAPE
 ENTER "GO" WHEN READY ("PA" TO SUSPEND)
 pa

 ENTER GO,PRSTR WHEN READY
 go,prstr

RESTORING DISK LU 28
RESTORED DISK LU 28

CREATED USING:  PSAVE  92084-16656 REV.2121  < 810905.1311 >
READ USING:     PRSTR  92084-16657 REV.2121  < 810714.0904 >
TAPE NUMBER:      2    SAVE FILE:   3    USER: BILL.JT
PROGRAM: PSAVE         OPTIONS:    MU     DATE: WED,9 SEPT.  1988
DISK LU:    28         TITLE:   PBU TESTING PRSTR
SECTION:     2         (TRK       1   SEC     0)


 OK TO PROCEED? (YE/NO)
 ye
 RESTORING DISK LU 28
 RESTORED DISK LU 28
 PRSTR NORMAL END OF JOB
:
```

The following example illustrates two error conditions that can arise to abort PRSTR: source disk mounted and PSPAR not scheduled. In both cases, the condition is corrected and PRSTR rescheduled. PRSTR is invoked in this example to perform a selective restore (SE option) with verify (VE option). In the third call to PRSTR, both the tape LU and the hard-copy LU parameters are defaulted, the tape LU defaults to LU 8 (magnetic tape transport), and the

hard-copy LU defaults to the log terminal. The DE option (suppress file-header verification and OK TO CONTINUE prompts) also is specified in the third call to PRSTR.

```
    :prstr
     ENTER OPTIONS
     ?
     ALLOWED OPTIONS (RSTR):
     (1) VE => VERIFY
     (2) DE => DEPRESS USER QUESTIONS
     (4) LU => LU SAVE/RSTR/COPY
     (4) UN => UNIT (FULL VOL) SAVE/RSTR/COPY
     (4) PB => PUSHBUTTON UNIT IMAGE SAVE/RSTR
     (4) SE => SELECTIVELY RESTORE LUS


THE NUMBERS IN ()'S REPRESENT CATEGORIES AND
             ONLY ONE OUT OF EACH CATEGORY CAN BE USED.

      ENTER OPTIONS
      vese
      ENTER 'FILE:LU' PAIRS
      1:37
      ENTER TAPE LU
      8
      ENTER HARD COPY LU
      1

     CREATED USING:  PSAVE 92084-16656 REV.2121 < 810905.1311 >
     READ USING:     PRSTR 92084-16657 REV.2121 < 810714.0904 >
     TAPE NUMBER:  1 SAVE FILE:    1   USER: BILL.JT
     PROGRAM: PSAVE  OPTIONS: VE  LU   DATE: 10:36 AM TUE.,18 MAY,1989
     DISK LU:    37  TITLE: This is my private cartridge.


 SECTION:          1   (TRK        0   SEC     0)
      OK TO PROCEED? (YE/NO)    ye
      LU    37 IS MOUNTED
      DISK LU IS MOUNTED
      PRS68 ERROR#23                        *disk-mounted error; PRSTR EXITS
      REPORTING MODULE='MANDL '.
     :dc,-37                                * run DC to dismount LU
      DISK CRN    PV   LU   37  INACTIVE
     :prstr                                 * rerun prstr
      ENTER OPTIONS
      vese
      ENTER 'FILE:LU' PAIRS
      1:37
      ENTER TAPE LU
      8
      ENTER HARD COPY LU

     CREATED USING: PSAVE 92084-16656 REV.2121 < 810905.1311 >
     READ USING:    PRSTR 92084-16657 REV.2121 < 810714.0904 >
     TAPE NUMBER:  1  SAVE FILE:    1  USER: BILL.JT
     PROGRAM: PSAVE   OPTIONS: VE  LU  DATE: 10:36 AM TUE.,18 MAY,1988
     DISK LU:    37   TITLE: This is my private cartridge.
```

```
 SECTION:  1  (TRK   0   SEC     0)
 OK TO PROCEED? (YE/NO)  ye
 VERIFYING DISK LU W/SPARING
 PSPAR SCHEDULE FAILURE
 PRS68 ERROR#43                        *PSPAR not-scheduled error,
 REPORTING MODULE='SPARE '.            *PRSTR exits
:rp,pspar                              *RP PSPAR
:prstr                                 *sigh, re-rerun PRSTR
 ENTER OPTIONS
 vesede
 ENTER 'FILE:LU' PAIRS

 1:37
 ENTER TAPE LU

 TAPE LU DEFAULTED TO      8
 ENTER HARD COPY LU

 CREATED USING: PSAVE 92084-16656 REV.2121 < 810905.1311 >
 READ USING:    PRSTR 92084-16657 REV.2121 < 810714.0904 >
 TAPE NUMBER:  1  SAVE FILE:    1  USER: BILL.JT
 PROGRAM: PSAVE   OPTIONS: VE  LU  DATE: 10:36 AM TUE.,18 MAY,1988
 DISK LU:    37   TITLE: This is my private cartridge.
 SECTION:     1   (TRK      0  SEC      0)
 VERIFYING DISK LU W/SPARING
 RESTORING DISK LU       37
 RESTORED  DISK LU       37
 PRSTR NORMAL END OF JOB
:
```

PCOPY

PCOPY executes a fast disk-to-disk copy operation. The source and destination disk LUs need not be on the same disk unit, but they must have the same track size. (Note that CS/80 tracks are logical tracks, not physical tracks; thus, a CS/80-to-CS/80 copy is valid regardless of the physical track size, as long as the logical track sizes are identical.)

Calling PCOPY

To call PCOPY enter the following runstring:

```
:[RU,]PCOPY,[input],srce LU,dest LU[,VE[,hcpy]]
```

Here input is the LU (or file) from which the parameter inputs are to be read. If input is from a file or a non-interactive device, no other parameters may be specified in the runstring. The default is to the log device.

The srce LU is the LU of the disk subchannel to be copied.

The dest LU is the LU of the disk subchannel to which the data is to be copied.

VE is the verify option. When this option is selected, PCOPY spares the destination subchannel tracks in a prepass over the disk before data is written. The default is to suppress verify. PSPAR must be loaded into memory for this option.

The hcpy parameter is the LU of the device on which the read or verify errors are logged. The default is to the log device.

Calling PCOPY Interactively

To run PCOPY interactively, enter the utility call PCOPY or RU,PCOPY. Enter a valid response to each prompt or enter a carriage return to use the default value. If a response is not valid, an appropriate error message is issued and the prompt is repeated. You may enter AB, /A, EX, or /E to any of the prompts to terminate the utility. Following are the PCOPY prompts.

Prompt	Response
ENTER OPTIONS	Enter VE if the verify option is desired. Enter a carriage return to bypass the VE option.
ENTER SOURCE DISK LU	Enter the LU of the disk subchannel from which the data is to be copied. This parameter may not be defaulted.
ENTER DEST DISK LU	Enter the LU of the disk subchannel to which the data is to be copied. This parameter may not be defaulted.
ENTER HARD COPY LU	Enter the LU of the device on which the disk copy or verify errors are to be logged. Enter a carriage return to specify the default LU, your log device.

Once PCOPY starts copying the data, you can terminate the utility by entering break mode (by hitting any key on the keyboard) and entering BR,PCOPY. PCOPY should not be terminated with the OF command.

Caution Be aware that the directory tracks might not be copied when a disk LU with FMGR files and directory tracks is copied to a destination disk LU with fewer tracks than the source disk LU. If the directory tracks are not written, the copy operation is invalid.

If the source disk LU and the destination disk do not have the same number of tracks, PCOPY issues a warning and waits for your response to continue or to abort the copy. If you respond YE to continue, PCOPY copies tracks until it has filled the last track of the destination disk or has copied the last track of the source disk, whichever is smaller.

When copying to a destination disk with more tracks than the source disk, you must use the FMGR MC (Mount Cartridge) command on the destination disk and specify the number of tracks copied from the source disk to allow FMGR to locate the correct directory tracks.

PCOPY Example

In the following example, PCOPY is invoked interactively. The source and destination disks do not have the same number of tracks, so PCOPY issues a warning.

```
:RU,PCOPY
 ENTER OPTIONS
 <cr>
 ENTER SOURCE DISK LU
 16
 ENTER DEST DISK LU
 30
 ENTER HARD COPY LU
 <cr>  203 TRACKS IN SOURCE LU
 100 TRACKS IN DEST LU
 OK TO PROCEED? (YE/NO)
 YE
 PCOPY NORMAL END OF JOB
:
```

Offline Physical Backup

Offline backup is implemented as a set of stand-alone save and restore tasks, with subtasks that allow you to display the I/O configuration, position the tape, or specify the input device:

IO	Display current system I/O configuration.
TR	Read parameter inputs from a specified tape unit or terminal.
SA	Save disk to tape.
CO	Copy one disk subchannel to another.
RE	Restore tape file to disk.
RW	Rewind transport tape or seek cartridge tape block 0.
FF	Forward space transport tape or cartridge tape a specified number of files.
BF	Backspace transport tape a specified number of files.

Since the offline utility does not have access to any of the online system tables, information about system LUs and disk subchannel definitions must be supplied. If the operation is a restore, the utility can retrieve the subchannel definition from the disk definition record in the tape header. (Refer to the section "Data Structures" for details of the tape format.) If the operation is a save or copy, however, you must supply all subchannel definitions. The information can be read from a minicartridge or tape transport. (Refer to the SA and CO command discussions later in this chapter for details of the subchannel definition process.)

Referencing Devices Offline

LUs for devices such as terminals, tape transports, or tape cartridges are referenced in the same manner both online and offline. However, LUs that refer to offline disk units actually are target LUs that point to the associated disk driver select codes. Target LUs for each device are specified in the !BCKOF I/O configuration. Each disk type has one target LU; that is, one target LU relates to the HP 7900 disk driver select code, one LU relates to the CS/80 driver select code, and separate target LUs relate to the MAC and ICD disk driver select codes. Once the target LU has been selected, specific subchannels of that LU can be defined and operations performed selectively on the defined subchannels. A target LU must be specified under the following conditions:

1. The destination disk type for the restore is not the same as the save disk type. For example, if you want to restore an ICD disk to a MAC disk, you must specify the target LU of the MAC disk-driver select code in the disk LU parameter of the RE command runstring.

2. An offline utility is generated with more than one target LU for a given type of disk. For example, if target LUs 10 and 11 both point to ICD disk select codes, you must specify the LU that points to the driver select code of the disk you intend as the destination of a restore operation or as the source (or destination) of a copy operation.

Loading the Offline Utilities

The physical backup utilities can be supplied in either of two ways, as described below.

3. Three cartridge tapes, where:

   a. the first cartridge tape, !BCKO1, contains the memory-based (offline) operating system, reconfigurator, and the startup program BCKOF;

   b. the second cartridge tape, !BCKO2, contains the offline utilities FORMT and PSPAR that are scheduled for execution in restore or copy operations where the VErify option has been selected, or in restore operations where the pushbutton option (PB) has been selected;

   c. the third cartridge tape, !BCKO3, contains the offline PRSTR, PCOPY, and PSAVE utilities.

   If you want to create your own copy of the Offline Physical Backup Utilities, enter the command

   ```
   TR,*BCKCT
   ```

   from your RTE-6/VM Primary System and follow the directions given in the system response to the command.

4. One magnetic tape reel that contains all of the files !BCKO1, !BCKO2, and !BCKO3, together with a cartridge tape containing !MTLDR.

   If you want to create your own copy of the offline physical backup utilities on magnetic tape, use the transfer file *BCKMT supplied with the RTE-6/VM Primary System. Mount and load a magnetic tape and then enter the command

   ```
   TR,*BCKMT,<mt>
   ```

   where <mt> is the LU of your magnetic tape device. To create your own copy of the cartridge tape !MTLDR, insert a cartridge tape in the left CTU drive of an HP 264x type terminal (LU 4) and enter the command

   ```
   ST,!MTLDR,4,BA
   ```

   from your RTE-6/VM Primary System. The file !MTLDR will be copied to your cartridge tape.

Loading the Offline Utility from Cartridge Tape into Memory

The following steps should be performed in the order given to load the offline physical backup utilities from cartridge tape into memory.

1. Place the cartridge tape labeled !BCKO1 in the left CTU of your HP 264x terminal (LU 4) and load the memory-based operating system as follows:
   a. HALT the computer.
   b. Select and CLEAR the S-Register, then
      - set bits 15−14 to the location of the Cassette Loader ROM.
      - set bits 11−6 to the HP 264x terminal select code.
      - set bit 5 (to indicate a slow boot is desired).
   c. Press STORE, PRESET, IBL, then PRESET again.
   d. Press RUN. A HLT 77 should occur (102077 octal displayed).
2. You are now ready to load the reconfigurator and startup program into memory. See the section “Loading and Using the PBU I/O Reconfigurator” for loading instructions.

Loading the Offline Utility from Magnetic Tape into Memory

Two procedures are described below for loading the offline physical backup utilities from magnetic tape into memory. Procedure One should be followed if your system does not contain a Magnetic Tape Loader ROM; it includes instructions for placing a Magnetic Tape Loader, !MTLDR, from cassette tape into memory. Procedure Two applies only to systems that include the Loader ROM.

Procedure One: Loading Utilities WITHOUT a Magnetic Tape Loader ROM.

1. Place a cassette tape containing !MTLDR in the left CTU of your HP 264x terminal and load !MTLDR into memory as follows:
   a. HALT the computer.
   b. Select and CLEAR the S-Register, then
      - set bits 15−14 to the location of the Cassette Loader ROM.
      - set bits 11−6 to the HP 264x terminal select code.
   c. Press STORE, PRESET, IBL, then PRESET again.
   d. Press RUN. A HLT 77 should occur (102077 octal displayed).
2. Mount and place the offline backup utility tape online.

3. Load the memory-based operating system as follows:
    a. Select and CLEAR the P-Register, then set bit 1 (the P-Register now contains 2 octal).
    b. Press STORE.
    c. Select and CLEAR the S-Register, then set bits 11−6 to the select code for the magnetic tape unit.
    d. Press STORE.
    e. Press RUN. A HLT 77 should occur (102077 octal displayed).
4. You are now ready to load the reconfigurator and startup program into memory. Refer to the section “Loading and Using the PBU I/O Reconfigurator” for instructions.

Procedure Two: Loading Utilities Using the Magnetic Tape Loader ROM.

1. Mount and place the offline backup utility tape online.
2. Load the memory-based operating system as follows:
    a. HALT the computer.
    b. Select and CLEAR the S-Register, then
        - set bits 15−14 to the location of the Magnetic Tape Loader ROM.
        - set bits 11−6 to the magnetic tape unit select code.
        - set bit 5 (to indicate a slow boot is desired).
    c. Press STORE, PRESET, IBL, then PRESET again.
    d. Press RUN. A HLT 77 should occur (102077 octal displayed).
3. You are now ready to load the reconfigurator and startup program into memory, as described in the following section.

Loading and Using the PBU I/O Reconfigurator

Loading the PBU I/O Reconfigurator. The reconfigurator and startup program should be loaded into memory as follows:

1. Select and CLEAR the P-Register, then set bit 1 (the P-Register now contains 2 octal).
2. Press STORE.
3. Select and CLEAR the S-Register, then
   a. set bit 15 (to indicate that you are going to reconfigure)
   b. set bits 11−6 to:
      - select code of the H-Series disk (ICD) you will be using, or
      - 0 if you will NOT be using an H-Series disk.
   c. set bits 5−0 to the system console select code.
4. Press STORE.
5. Press RUN.

After the reconfigurator and startup program have been read from tape, the reconfigurator will automatically start executing with display and input at your system console.

Using the PBU I/O Reconfigurator. The initial I/O configuration (known to !BCKOF) now must be changed to the configuration of your system. It is important to have your CPU I/O configuration readily available for this procedure.

The initial !BCKOF configuration is shown below, followed by directions for modifying that configuration. Note that your configuration may not be identical to that shown, due to the automatic reconfiguration that took place for your system (and ICD disk, if used) when you last set the S-Register.

```
START RECONFIGURATION
LIST DEVICE LU #?                              *Direct output to your
1                                              *  terminal
I/O RECONFIGURATION ALREADY PERFORMED:
CURRENT SELECT CODE#,NEW SELECT CODE#?
15,15          *SYSTEM CONSOLE
                                               *ICD disk if S-Register set


CURRENT I/O CONFIGURATION:

SELECT CODE 11= TBG                            *Time Base Generator card
SELECT CODE 15= EQT  1,TYPE  5                 *System Console
SELECT CODE 17= EQT  8,TYPE 23                 *Magnetic Tape Unit card #1
SELECT CODE 20= EQT  8,TYPE 23                 *Magnetic Tape Unit card #2
SELECT CODE 21= EQT 10,TYPE 23                 *7974A Magnetic Tape Unit
SELECT CODE 50= EQT  9,TYPE 31                 *7900 Disk card #1
SELECT CODE 51= EQT  9,TYPE 31                 *7900 Disk card #2
SELECT CODE 52= EQT  3,TYPE 32                 *79xx MAC Disk
SELECT CODE 53= EQT  4,TYPE 32                 *79xx ICD Disk
SELECT CODE 54= EQT  5,TYPE 33                 *79xx CS/80 Disk
SELECT CODE 60= EQT  2,TYPE  0  PRMPT          *Interactive Terminal
SELECT CODE 61= EQT  7,TYPE  0  PRMPT          *Interactive Terminal
SELECT CODE 62= EQT  6,TYPE  5  PRMPT          *Interactive Terminal
I/O RECONFIGURATION? (YE/NO)
YE
```

Reconfigure your system as follows:

Note It is not necessary to reconfigure the select code of your system console as was done when you last set the S-Register.

---

1. You may have to reconfigure the select code of your Time Base Generator (TBG) if it is not the same as the one shown above (11 octal). In this case enter:

   ```
   11,<sc>
   ```

   where <sc> is the octal select code of the TBG in your system.

2. You may have to reconfigure the select codes of your magnetic tape unit if they are not the same as the ones shown above (17 and 20 octal). Note that you must reconfigure *two* select codes for *one* magnetic tape unit. Enter:

   ```
   17,<sc lc>
   20,<sc hc>
   ```

   where <sc lc> is the lower numbered select code of the magnetic tape unit and <sc hc> is the higher numbered select code. Note that an HP 7974A magnetic tape unit only requires one select code, select code 21.

3. You may have to reconfigure the select code of the disk you want to use. If you will use only an ICD disk, the disk will have been configured when you last set the S-Register and does not need to be reconfigured. If you are NOT using an ICD disk or will be using other disks in addition to an ICD disk, you must reconfigure the appropriate disks as shown below:

   ```
   <!BCKOF disk sc><your disk sc>
   ```

   where <!BCKOF disk sc> is the select code of the type of disk (CS/80, MAC, or 7900 disk) you want to reconfigure, and <your disk sc> is the select code of the same type of disk that you want to use in your system. Be aware that if you are reconfiguring a 7900 disk, there are TWO disk select codes for ONE 7900 disk unit. The lower numbered select code in the !BCKOF system should be reconfigured to the lower numbered select code in your system.

4. Enter /E to terminate the I/O reconfiguration. The current configuration is then displayed as shown on the following page. When the OK TO PROCEED? prompt appears, review your changes to the original configuration. If the changed configuration is correct, enter YE to continue. If the changes are incorrect, enter NO and correct any errors in the configuration (steps 1 through 4 above, as required). When the configuration has been corrected, enter /E to terminate the reconfiguration, and respond YE to the OK TO PROCEED? prompt.

5. Since memory reconfiguration is not necessary, enter NO when the MEM RECONFIGURATION? (YE/NO) prompt is displayed.

6. The operating system will now start its initialization pass and then start executing the startup (and user control) program !BCKOF.

   A sample system configuration display is shown below.

```
SET TIME
        LU#  EQT#  SUBC#   S.C.  TYPE  DESCRIPTION
          1     1      0    15B    5B  INTERACTIVE TERMINAL
          4     1      1    15B    5B  LEFT C.T.U.
          5     1      2    15B    5B  RIGHT C.T.U.
          7    10      0    21B   23B  7974A MAG TAPE/MASS STORAGE
          8     8      0    17B   17B  MAG TAPE/MASS STORAGE
          9     5      1    11B   33B  CS/80 CARTRIDGE TAPE
         10     2      1    60B    0B  INTERACTIVE TERMINAL
         11     7      0    61B    0B  INTERACTIVE TERMINAL
         12     9      0    50B   31B  7900 DISK
         13     3      0    52B   32B  79XX (MAC) SERIES DISK
         14     4      0    53B   32B  79XX (ICD) SERIES DISK
         15     5      0    11B   33B  79XX (CS/80) SERIES DISK

 Please enter tape-LU for reading !BCK02: n
```

Your response to the “Please enter tape-LU” prompt above is determined by the tape media on which the physical backup utilities are supplied. If the utilities are supplied on cartridge tapes, refer to the next section for loading instructions; if the utilities are supplied on magnetic tape, refer to the section “Loading Utilities supplied on Magnetic Tape”.

Loading Utilities Supplied on Cartridge Tape

After the memory-based operating system, reconfigurator, and startup program have been read in from cartridge tape, enter LU 4 (if the cartridge is to be installed in the left CTU of the system console) or LU 5 (if the cartridge is to be installed in the right CTU of the system console).

A prompt now appears to insert the next cartridge tape, !BCK02, and to enter a space character and carriage return. (See the description of the contents of !BCK02 at the beginning of this section.) When the cartridge is properly inserted, the offline backup utilities startup program, BCKOF, reads in two utilities that are scheduled by a save, copy, or restore operation.

After these are loaded into memory, you are prompted to insert the next cartridge tape, !BCKO3, and to enter a space character and carriage return. This cartridge tape contains the utility programs that perform the three operations, REstore, COpy, and SAve. Note that these programs are not read into memory until the desired operation is specifically requested.

At this point, the following prompt is displayed to indicate that the physical backup utilities are available for the offline save, restore, or copy operations:

```
OFFLINE DISK-TAPE SAVE/RESTORE/COPY SYSTEM, !BCKOF
TASK?
```

You now may specify any of the offline utilities by entering the associated response to the TASK? prompt. Refer to the individual utility runstring descriptions in the remainder of this chapter.

Loading Utilities Supplied on Magnetic Tape

After the memory-based operating system, reconfigurator, and startup program have been read in from magnetic tape, enter LU 8 to load all of the programs for the different offline operations into memory from the tape. When the load is complete, the following prompt is displayed:

```
OFFLINE DISK-TAPE SAVE/RESTORE/COPY SYSTEM, !BCKOF
TASK?
```

You may now specify any of the offline utilities by entering the associated response to the TASK? prompt. Refer to the individual utility runstring descriptions in the remainder of this chapter.

The TASK? prompt indicates that the offline utility set has been loaded and you can begin specifying the save, restore, or copy tasks as defined in the following sections. If an error occurs when processing any task, the processing of that task is aborted. The offline utility issues the appropriate error message and returns to the TASK? prompt.

Any task in process can be halted by entering break mode (by hitting any key on the keyboard) and entering BR, TASK. The OF command should not be used to abort any of the offline tasks.

Offline System Console Operations

When using the physical backup utilities offline, you can enter the non-session operating environment by exiting the utility (/E or EX in response to the TASK? prompt) and striking any key on the system console. When the asterisk prompt (*) appears, you can execute any of the system break mode commands that are valid in your system configuration. One usage of this would be to run the FORMT utilitiy to initialize or format a MAC disk (see Chapter 9 for information on using the FORMT utility). Another system command that may be useful is the "UP,<eqt>" command to "UP" a device that has gone down.

Offline Commands

The commands are described in detail in the sections that follow.

Help Function (HE)

Purpose: Lists all available commands.

Syntax: `HE or ??`

Description:

When HE or ?? is entered in response to the TASK? prompt, the following is displayed:

```
TASK?

??
IO - Display current I/O configuration.
SA - Save disk regions to tape.
RE - Restore tape file to disk.
CO - Copy one disk subchannel to another.
RW [,lu] - Rewind magnetic tape, seek to
     CTD block zero.  (Default lu=8)
FF [,[lu],[n]] - Forward space magnetic tape or
     CTU n files.  (Default lu=8, n=1)
BF [,[lu],[n]] - Backspace magnetic tape n files.
     NOP for CTU.  (Default lu=8, n=1)
TR,lu - Read commands from 'lu'.
/E, EN, or EX - Terminate program.
```

Disk-to-Disk Copy (CO)

Purpose: Copies one disk subchannel to another.

Syntax: `CO[,input],targ LU,dest LU[,VE[,hcpy]]`

input	The LU of the device from which the parameter inputs are to be read. If input is a non-interactive device, no other parameters should be specified in the command. The default is the system console, LU 1.
targ LU	The LU of the target disk from which the subchannel is to be copied. This parameter is required for CO to select the format of the subchannel definition prompt set.
dest LU	The LU of the disk to which the subchannel is to be copied. This parameter is required for CO to select the format of the subchannel definition prompt set.
VE	Turn on the data verify option. Track sparing is done only when this option is specified.
hcpy	The LU of the device on which copy errors are to be logged. The default is the system console, LU 1.

Alternatively, you can specify CO interactively and have the utility prompt you for the parameters as follows:

Prompt	Response
`ENTER OPTIONS`	If verification and track sparing are desired, enter VE to specify the data verify option.
`ENTER SOURCE LU`	Enter the target disk from which the subchannel is to be copied. The source and destination can be the same disk LU.
`ENTER DEST LU`	Enter the disk to which the subchannel is to be copied.
`ENTER HARD COPY LU`	Enter the device that is to receive data error messages.

After the utility has been specified interactively, CO prompts for the subchannel definitions as above.

Description:

The CO command allows you to copy one disk subchannel to another. The offline utility also recognizes this command if it is entered as “PC” or “PCOPY”. Optionally, the subchannel definitions may come from a tape file.

CO prompts for the subchannel definitions in one of the following formats. The same prompt is used for both the source and destination disk subchannel definitions, with the word SOURCE or DESTINATION in the DEFINE TRACK MAP statements. After the subchannels are defined, CO begins the copy task and displays the message COPYING. At the end of the task, CO issues the message PCOPY NORMAL END OF JOB and exits to the TASK? prompt.

For CS/80 disks, the following prompt sequence is issued:

```
DEFINE (srce,dest) TRACK MAP
DEFINE SUBCHANNELS, LU = n
DEVICE (MODEL,HP-IB ADDRESS,UNIT,VOLUME)
mmmm,aaaaa,uu,vvv

xxxxxx BLOCKS REMAINING
SUBCHANNEL 0 (TRACKS,BLOCKS/TRACK)
ttt,bbb
```

For MAC source and destination disks, the following prompt sequence is issued:

```
DEFINE (srce,dest) TRACK MAP
DEFINE SUBCHANNELS,LU=n

MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,
SUBCHANNEL 00?

mmmm,ttt,ccc,h,s,a,sss
```

where:

model#	is the disk drive model name.
#tracks	is the number of tracks on the subchannel (the actual amount of storage).
1st cyl	is the cylinder number where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.
head#	is the head number where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.
#surf	is the number of surfaces that are used in the subchannel.
address	is the unit number associated with each disk drive. It is always displayed on the front panel. There may be eight units on a single MAC controller, numbered 0 through 7.
#spares	is the number of spare tracks allocated to the subchannel. These are for replacing unusable tracks on the subchannel. Use approximately 6 spare tracks per 200 data tracks.

Table 3-2 provides MAC Disk information.

Table 3-2. MAC Disk Information

Model Names	Words/ Sector	Sectors/ Track	Tracks/Surface (cylinders/drive)	Surfaces/ Drive	Tracks/ Drive
7905	64	96	411	3	1233
7906	64	96	411	4	1644
7920	64	96	823	5	4115
7925	64	128	823	9	7407

For ICD source and destination disks, the following prompt sequence is issued:

```
DEFINE (srce,dest) TRACK MAP
DEFINE SUBCHANNELS,LU=n

 MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,UNIT#,
 SUBCHAHNNEL 00?
 mmmm,ttt,ccc,h,s,a,sss,u
```

where:

model#	is the disk drive model name.
#tracks	is the number of tracks on the subchannel (the actual amount of storage).
1st cyl	is the cylinder number where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.
head#	is the head number where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.
#surf	is the number of surfaces that are used in the subchannel.
address	is the HP-IB address associated with each of up to two drives. Some drives display it on the front panel. There may be eight addresses, numbered 0 through 7.
#spares	is the number of spare tracks allocated to the subchannel. These are for replacing unusable tracks on the subchannel. Use approximately 6 spare tracks per 200 data tracks.
unit#	is the unit number for HP 9895 drives only. The two drives on an HP 9895 controller are addressed by their unit numbers (0 or 1), which refer to the left and right drives, respectively.

See Table 3-3 below for ICD Disk information.

Table 3-3. ICD Disk Inforamation

Model Names	Words/ Sector	Sectors/ Track	Tracks/Surface (cylinders/drive)	Surfaces/ Drive	Tracks/ Drive	Unit #
9895	64	60	77	2	154	0 or 1
7906H	64	96	411	4	1644	0
7920H	64	96	823	5	4115	0
7925H	64	128	823	9	7407	0

For HP 7900 disks, the following prompt sequence is issued:

```
DEFINE (srce,dest) TRACK MAP
DEFINE SUBCHANNELS,LU = n
FIRST 7900 SUBCHANNEL NUMBER (0=<n<=s)
0
#TRACKS, 1ST CYL
SUBCHANNEL 00?
ttt,c
```

When a CS/80 copy operation is to begin at a disk location other than block 0 (as does the first disk subchannel on a system disk), enter a negative value as the TRACKS parameter of the subchannel definition. This, in effect, creates a "hole" on the disk and allows you to access the desired block for the copy. As an example, if the copy is to begin at block 33600 of an HP 7908 CS/80 disk, the sequence of definitions is as follows:

```
64750 BLOCKS REMAINING
SUBCHANNEL 0 (TRACKS,BLOCKS/TRACK)
-300,48                              *sets starting block of
                                     *next subchannel to 14400

50350 BLOCKS REMAINING
SUBCHANNEL 0 (TRACKS,BLOCKS/TRACK)
-200,48                              *sets starting block of
                                     *next subchannel to 24000

40750 BLOCKS REMAINING
SUBCHANNEL 0 (TRACKS,BLOCKS/TRACK)
-200,48                              *sets starting block of
                                     *next subchannel to 33600

31150 BLOCKS REMAINING
SUBCHANNEL 0 (TRACKS,BLOCKS/TRACK)
200,48                               *positive value defines
                                     *subchannel at block 33600
```

The negative track values can be taken from the subchannel definitions contained in your system generation listing.

Refer to the *RTE-6/VM System Manager's Reference Manual* (part number 92084-90009) for further details of the CS/80 disk configuration.

Display I/O Configuration (IO)

Purpose: Displays current I/O configuration.

Syntax: `IO`

Description:

When I/O is entered as the task, the current I/O configuration is displayed in the format shown in this example:

```
TASK?

LU#   EQT#   SUBC#   S.C.  TYPE  DESCRIPTION
 1     1        0    25B    5B   INTERACTIVE TERMINAL
 4     1        1    25B    5B   LEFT CTU
 5     1        2    25B    5B   RIGHT CTU
 7    10        0    21B    23B  7974A MAG TAPE/MASS STORAGE
 8     8        0    23B    23B  MAG TAPE/MASS STORAGE
 9     2        1    15B    33B  CS/80 CARTRIDGE TAPE
11     7        0    25B     0B  INTERACTIVE TERMINAL
12     2        0    15B    33B  79XXA (CS/80) SERIES DISK
13     3        0    16B    32B  7905 (MAC) SERIES DISK
14     4        0    21B    31B  7900 DISK
15     5        0    77B    32B  7905H (ICD) SERIES DISK
TASK?
```

Restore Tape File (RE)

Purpose: Restores an LU save or unit save tape file to disk.

Syntax: `RE[,input[,targ LU[,srce LU[,file#[,opts[,hcpy]]]]]]`

input	The LU of the device from which the parameter inputs are to be read. If input is a non-interactive device, no other parameters may be specified in the runstring. The default is the system console, LU 1.
targ LU	The LU of the target disk. Refer to the section “Referencing Devices Offline” for a description of the use of this parameter. There is no default for this parameter.
srce LU	The LU of the tape transport or CS/80 cartridge from which the save file is to be read. The default is tape transport LU 8.
file#	The integer number of the file to be read. File# − 1 files will be skipped before the read begins. Default is the current tape position for tape transports or file #1 for CS/80 cartridges.

opts	Any of the two-character ASCII option codes described in the next section. Options can be specified in any order, and may not have any intervening characters (including blanks) between options. If no options are specified, the defaults are no verify, read subchannel definitions from the tape file record (TA), prompt for file verify, and LU restore if the save was an LU or MU save or UN restore if the save was a unit save. If you respond with "??" RE will display a list of available options.
hcpy	The LU of the device on which errors are logged and track map definitions are echoed. The default is the system console, LU 1.

Alternatively, you specify RE interactively and have the utility prompt you for the parameters. The prompts and responses are shown below. Enter a carriage return at any prompt to default any option.

Prompt	Response
`ENTER OPTION`	Enter any of the valid option codes. Since the UN, LU, SE, and PB options are mutually exclusive, and TA and UD are mutually exclusive, only four options are possible, at most. If you respond "?" or "??", RE displays a list of all allowable options.
`ENTER DEST DISK LU`	Enter the target LU of the disk to which the files are to be restored.

If you select the SE option:

`ENTER FILE:SUBCHANNEL PAIRS`	Enter the file number and target subchannel to which the file is to be restored. If the track map definitions come from tape (TA option), any positive subchannel numbers entered are ignored. If a negative subchannel number is entered, RE responds with the message "ILLEGAL SUBCHANNEL" and again prompts for the file:subchannel pair. (Although these subchannel numbers are ignored they must still be legal.) The subchannel numbers may be given sequentially starting with 0, and the track maps may then be defined in the same order. Multiple entries should be separated by commas. To continue on the next line, end the current line with a comma.
`ENTER TAPE LU`	Enter the LU of the tape unit containing the saved file. The default is the tape transport, LU 8.

If you select the LU option:

`ENTER FILE NUMBER`	Enter the tape file number of the file to be restored. The default is file #1 on CS/80 cartridges and the current tape position on tape transports.
`ENTER HARD COPY LU`	Enter the LU number of the device on which errors will be logged and track map definitions displayed. The default is the system console, LU 1.

If you do not select the DE option:

– tape header –

`OK TO PROCEED?(YE/NO)`	Enter YE if this is the correct tape. If your response is NO, the ENTER FILE NUMBER prompt is repeated.

Description:

The offline utility also recognizes the RE command if it is entered as “PR” OR “PRSTR”. When restoring to disk, be sure that the saved-file disk type and the restored disk type are compatible. Refer to Table 3-1 to determine disk compatibility before specifying the destination disk.

RE prompts for the subchannel definitions in one of the following formats.

```
DEFINE SUBCHANNELS, LU =      n
DEVICE (MODEL,HP-IB ADDRESS,UNIT,VOLUME)
mmmm,aaaaa,uu,vvv

xxxxxx BLOCKS REMAINING
SUBCHANNEL   nn (TRACKS,BLOCKS/TRACK)
ttt,bbb
```

For MAC disks, the following prompt sequence is issued.

```
DEFINE SUBCHANNELS,LU=n

MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,
SUBCHANNEL 00?
mmmm,ttt,ccc,h,s,a,sss
```

where:

model#	is the disk drive model name.
#tracks	is the number of tracks on the subchannel (the actual amount of storage).
1st cyl	is the cylinder number where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.
head#	is the head number where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.
#surf	is the number of surfaces that are used in the subchannel.

address — is the unit number associated with each disk drive. It is always displayed on the front panel. There may be eight units on a single MAC controller, numbered 0 through 7.

#spares — is the number of spare tracks allocated to the subchannel. These are for replacing usable tracks on the subchannel. Use approximately six spare tracks per 200 data tracks.

Refer to Table 3-2 for MAC Disk information.

For ICD disks, the following prompt sequence is issued:

```
DEFINE SUBCHANNELS,LU=n

MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,UNIT#,
SUBCHANNEL 00?

 mmmm,ttt,ccc,h,s,a,sss,u
```

where:

model# — is the disk drive model name.

#tracks — is the number of tracks on the subchannel (the actual amount of storage).

1st cyl — is the cylinder number where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.

head# — is the head number where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.

#surf — is the number of surfaces that are used in the subchannel.

address — is the HPIB address associated with each of up to two drives. Some drives display it on the front panel. There may be eight addresses, numbered 0 through 7.

#spares — is the number of spare tracks allocated to the subchannel. These are for replacing unusable tracks on the subchannel. Use approximately 6 spare tracks per 200 data tracks.

unit# — is the unit number for HP 9895 drives only. The two drives on an HP 9895 controller are addressed by their unit numbers (0 or 1) which refers to the left and right drives, respectively.

Refer to Table 3-3 for ICD Disk information.

For 7900 disks, the following prompt sequence is issued.

```
DEFINE SUBCHANNELS,LU = n
FIRST 7900 SUBCHANNEL NUMBER (0=<n<=s)
0
#TRACKS, 1ST CYL
SUBCHANNEL 00?
ttt,c
```

Unless the UN restore is selected, only the current track map definition is displayed as each tape file record is encountered. With the UN option selected, all track map definitions are displayed. Since you must specify all subchannels prior to the first restore when the UD option is selected, the SE option should be used with the UD option only when all of the source tape and destination

disk track map information is known. If this information is not known, the restore should be performed as multiple tasks using the LU option.

A response of AB, /A, EX, or /E to any of the prompts will cause PRSTR to restore according to the subchannel disk definitions found on the tape.

When restoring to any disk unit, the protect switch must be turned off and the format switch must be turned on. If either condition is not met, an error message is printed and the utility prompts with:

```
TYPE 'GO' WHEN READY ('PA' TO SUSPEND)
```

When the unit switch setting is correct, enter GO and the restore continues. If you enter PA, the utility suspends itself after printing:

```
TYPE GO,PRSTR TO CONTINUE
```

RE Command Options

The RE command options parameter lets you specify one or more of the options shown below:

VE	Turn on data verify option. Destination track sparing is performed only when this option is selected.
DE	Do not prompt for verification of tape file.
UN	Unit restore. This is the default if the save tape contains a unit save.
LU	LU restore. This is the default if the save tape contains an LU or MU save.
SE	Selective restore from a unit save or multiple LU restores in one operation. This is not a valid option if the save was a unit save in pushbutton-restorable format or a unit save in LSAVE, USAVE, or SAVE formats.
PB	Restore a CS/80 cartridge in pushbutton-restorable format. This is a unit restore.
TA	Restore according to the subchannel definitions on the tape. This is the default option. The subchannel definitions are displayed at the console as they are read.
UD	Read subchannel definitions from LU defined by input parameter.

When the UD option is selected, you are prompted to enter all subchannel definitions. These may be entered interactively or by TRansferring to a tape file containing the definitions. Where the destination disk type is not the same as the source disk listed in the tape header, the subchannels may need to be redefined. Subchannel definitions are also forced when the operating systems is not the same for the save and restore.

RE Example

In the following example, note that RE is invoked with the call “PRSTR”; the offline utility also accepts a call of the form “PR”. In this example, RE is called interactively, with the first response a “?” to request a listing of allowable options (either a single “?” or double “??” can be entered as the request). The VE, TA, and SE options are specified, and the file:subchannel pairs are defined for the selective restore operation. The tape header is then displayed, followed by the subchannel definition and the OK TO PROCEED? prompt. If the response is YE, the restore operation begins by first verifying the destination disk and sparing tracks if required.

While the restoration is in process, the message RESTORING DISK LU 2 (in this example) is displayed. When the restoration is complete, the RESTORED DISK LU 2 message is displayed. Since this example is a selective restore of two disks, the tape header of the second disk is then displayed and the sequence of prompt/verify/restore is repeated, followed by the TASK? prompt.

In the example, the TASK? prompt response is an unrecognizable set of characters to which REstore responds with the UNRECOGNIZED COMMAND message. Following the available command display (?? response), the EX command is issued and BCKOF terminates. By entering a carriage return (any keyboard character can be used), the system enters non-session mode and returns the * prompt. At this point, any of the system break mode commands that are valid for your configuration can be executed. To return to the offline backup utilities, enter a carriage return and enter RU,BCKOF, as shown in the following example.

```
TASK?
PRSTR
ENTER OPTIONS
?
ALLOWED OPTIONS (RSTR):
(1) VE => VERIFY
(2) DE => DEPRESS USER QUESTIONS
(3).TA => USE TAPE SUBCHANNEL DEFS
(4) UD => GET USER SUBCHANNEL DEFS
(4).LU => LU SAVE/RSTR/COPY
(4) UN => UNIT (FULL VOL) SAVE/RSTR/COPY
(4) PB => PUSHBUTTON UNIT IMAGE SAVE/RSTR
(4) SE => SELECTIVELY RESTORE LUS

THE NUMBERS IN ()'S REPRESENT CATEGORIES AND
ONLY ONE OUT OF EACH CATEGORY CAN BE USED.



ENTER OPTIONS
VETASE
ENTER DEST DISK LU
15
ENTER 'FILE:SUBCHANNEL' PAIRS
1:0,2:1
ENTER TAPE LU
<cr>
TAPE LU DEFAULTED TO       8
ENTER HARD COPY LU
<cr>
CREATED USING:  PSAVE 92084-16656 REV.2121 < 870805.1433 >
READ USING:     PRSTR 92084-16657 REV.2121 < 870805.1433 >
TAPE NUMBER:      1   SAVE FILE:      1  USER: FUBAR.TEST
```

```
 PROGRAM: PSAVE  OPTIONS: VE MU  DATE: 5:16 PM  MON., 22 FEB.,1989
 DISK LU:     2    TITLE: PS#604 CS/80 REV.2226
 SECTION:     1    (TRK      0  SEC      0)
 OK TO PROCEED? (YE/NO) YE
 VERIFYING DISK LU W/SPARING
 RESTORING DISK LU        2
 RESTORED  DISK LU        2
CREATED USING:  PSAVE 92084-16656 REV.2121 < 870805.1433 >
READ USING:     PRSTR 92084-16657 REV.2121 < 870805.1433 >
TAPE NUMBER:      1   SAVE FILE:      1  USER: FUBAR.TEST

 PROGRAM: PSAVE  OPTIONS: VE MU  DATE: 5:16 PM  MON., 22 FEB.,1982
 DISK LU:     3    TITLE: PS#604 CS/80 REV.2226
 SECTION:     1    (TRK      0  SEC      0)
 OK TO PROCEED? (YE/NO) YE
 VERIFYING DISK LU W/SPARING
 RESTORING DISK LU        3
 RESTORED  DISK LU        3
 TASK?
 LKSJDF
 UNRECOGNIZED COMMAND (ENTER ?? FOR HELP)
 TASK?
 ??
 IO - Display current I/O configuration.
 SA - Save disk regions to tape.
 RE - Restore tape file to disk.
 CO - Copy one disk subchannel to another.
 RW [,lu] - Rewind magnetic tape, seek to  CTD block zero.  (Default lu=8)
 FF [,[lu],[n]] - Forward space magnetic tape or
    CTU n files.  (Default lu=8, n=1)
 BF [,[lu],[n]] - Backspace magnetic tape n files.
    NOP for CTU.  (Default lu=8, n=1)
 TR,lu - Read commands from 'lu'.
 /E, EN, or EX - Terminate program.

 TASK?
 EX
 BCKOF TERMINATED BY USER
 <cr>
 *RU,BCKOF

 OFFLINE DISK-TAPE SAVE/RESTORE/COPY SYSTEM, !BCKOF
 TASK?
```

Save Disk to Tape (SA)

Purpose: Saves disk information in LU save, unit save format, or pushbutton-restorable format.

Syntax:

```
SA[,input[,srce LU[,dest LU[,file#[,opts[,hcpy[,tape density[,title]]]]]]]]
```

input — The LU of the device from which the parameter inputs are to be read. If input is a non-interactive device, no other parameters may be specified. The default is to the system console, LU 1.

srce LU — The LU of the disk subchannel from which the information is to be saved. This parameter is required for SA to select the format of the subchannel definition prompt set.

dest LU — The LU of the tape transport or CS/80 cartridge to receive the saved data. The default is to the tape transport, LU 8.

file# — The integer number to be assigned to the saved file. This parameter specifies the start location on the tape for the save file. Before saving, file# − 1 files are skipped from the start of the tape. The default is the current tape position on tape transports and file#1 on CS/80 cartridges. This parameter is meaningless if the option is MU and will cause an error if specified.

opts — Any of the following two-character ASCII option codes. Options must be specified with no intervening characters (including blanks). If no options are specified, the default is to LU save and no verify.

- VE — Turn on data verify option. Track sparing is performed only when this option is selected.
- UN — Unit save.
- LU — LU save.
- PB — Unit save in CS/80 pushbutton-restorable format. The destination LU must be CS/80 cartridge, and the source LU must be a CS/80 disk.
- MU — Multiple LU saves in one pass. The source LU parameter should be omitted; PSAVE will prompt for the source LUs. All LUs must be from the same class of disks, such as all MAC disk LUs.

hcpy — The LU of the device on which information about the save is to be printed. The default is the system console, LU 1.

title — The title (to a maximum of 40 ASCII characters) that will be placed in the tape header. The title may be entered in either uppercase or lowercase letters or a combination of both.

tape density — The tape density must be 800, 1600, or 6250. The default is to use the last tape density specified.

Alternatively, you can specify SA interactively and have the utility prompt you for the parameters as follows:

Prompt	Response
`ENTER OPTIONS`	Enter any of the above option codes. Since the UN, LU, PB and MU options are mutually exclusive, only two options are possible, at most. If you respond with "??", SA will display a list of available options.
`ENTER SOURCE DISK LU`	Enter the LU of the disk to be saved.
If you select the MU option:	
`ENTER DISK SUBCHANNEL`	Enter the subchannels to be saved, separated with commas. To continue on the next line, end the line with a comma. Normally, the entries will be sequential starting with 0, but you can specify subchannels (up to a total of 64) in any order.
`ENTER TAPE LU`	Enter the LU of the tape transport or CS/80 cartridge. The default is the tape transport, LU 8.
`ENTER FILE NUMBER`	Enter the file number at which the save is to begin. The default is the current tape position for tape transports and file #1 for CS/80 cartridges.
`ENTER HARD COPY LU`	Enter the LU of the device to record information concerning the tape saves. The default is the system console, LU 1.
`ENTER TITLE`	Enter the title, up to 40 ASCII characters, to be stored in the tape header.
`ENTER TAPE DENSITY`	Enter the tape density (800, 1600, or 6250). The default is the tape density last specified for the tape. (This prompt is only given if the tape drive is a 7974A, 7978A/B, or 7980A magnetic tape drive and the drive is positioned at the beginning of the tape.)

When you call SA interactively, SA prompts for the subchannel definitions as shown above.

Before beginning the save operation, SA checks that the tape is online and write enabled. If either condition is not met, an error message is printed and the utility prompts with:

```
TYPE 'GO' WHEN READY ('PA' TO SUSPEND)
```

When the switch settings are correct, enter GO and SA continues. If you enter PA, SA suspends itself after printing the following message. You can enter a carriage return after any prompt to specify the default. If you enter an invalid response, the prompt is repeated.

```
TYPE GO, PSAVE TO CONTINUE
```

Description:

Note that when you use a CTD, it is more efficient to save the disk online since the CTD cannot be cached offline. (Refer to the section on "Tape Handling" for a description of the CTD caching scheme.)

When you call SA in the runstring, SA prompts for the subchannel definitions in one of the following formats.

For CS/80 disks, the prompt sequence is:

```
DEFINE SUBCHANNELS, LU = n
DEVICE (MODEL,HP-IB ADDRESS,UNIT,VOLUME)
mmmm,aaaaa,uu,vv
NUMBER OF BLOCKS ON DEVICE
bbbbb
```

The "NUMBER OF BLOCKS ON DEVICE" prompt appears only if the model number is not recognized.

For MAC disks, the following prompt sequence is issued. The prompt is repeated for each subchannel on the disk or until you enter /E to indicate the end of the subchannel definitions for the save.

```
DEFINE SUBCHANNELS,LU=n

MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,
SUBCHANNEL 00?

 mmmm,ttt,ccc,h,s,a,sss
```

where:

model#	is the disk drive model name.
#tracks	is the number of tracks on the subchannel (the actual amount of storage).
1st cyl	is the cylinder number where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.
head#	is the head number where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.
#surf	is the number of surfaces that are used in the subchannel.
address	is the unit number associated with each disk drive. It is always displayed on the front panel. There may be eight units on a single MAC controller, numbered 0 through 7.
#spares	is the number of spare tracks allocated to the subchannel. These are for replacing unusable tracks on the subchannel. Use approximately 6 spare tracks per 200 data tracks.

Refer to Table 3-2 for MAC Disk Information.

For ICD disks, the following prompt sequence is issued. The prompt is repeated for each subchannel on the disk or until you enter /E to indicate the end of the subchannel definitions for the save.

```
DEFINE SUBCHANNELS,LU=n

MODEL#,#TRACKS,1ST CYL,HEAD#,#SURF,ADDRESS,#SPARES,UNIT#
SUBCHANNEL 00?

 mmmm,ttt,ccc,h,s,a,sss,u
```

where:

model#	is the disk drive model name.
#tracks	is the number of tracks on the subchannel (the actual amount of storage).
1st cyl	is the cylinder number of where the logical track 0 is located. Cylinders are numbered sequentially starting from 0.
head#	is the head number of where the logical track 0 is positioned. Heads are numbered sequentially starting from 0, one for each surface.
#surf	is the number of surfaces that are used in the subchannel.
address	is the HP-IB address associated with each of up to two drives. Some drives display it on the front panel. There may be eight addresses, numbered 0 through 7.
#spares	is the number of spare tracks allocated to the subchannel. These are for replacing unusable tracks on the subchannel. Use approximately 6 spare tracks per 200 data tracks.
unit#	is the unit number for HP 9895 drives only. The two drives on an HP 9895 controller are addressed by their unit numbers (0 or 1), which refer to the left and right drives, respectively.

Refer to Table 3-3 for ICD Disk Information.

For HP 7900 disks, the following prompt sequence is issued. The prompt is repeated for each subchannel on the disk or until you enter /E to indicate the end of the subchannel definitions for the save.

```
#TRACKS, 1ST CYL,
SUBCHANNEL 00?
ttt,c
```

SA Example

In the following example, SA is invoked interactively and the defaults are specified by entering a carriage return in response to the prompts. The UN unit save option is selected and the tape LU is defaulted to tape transport LU 8. The file number is defaulted and is determined by the current tape position of the tape transport (in this example the tape is at file position 3). The title

OFFLINE 7900 UNIT SAVE is assigned to both files. The subchannel definition prompt is then issued.

Since only two subchannels are to be saved, the SUBCHANNEL 02? prompt response is /E to indicate the end of the definition. The SAVING SUBCHANNEL message, followed by the tape header, is displayed as the save is in process. At the end of each save, the SUBCHANNEL SAVED message is issued. When the save is complete, SA exits and the TASK? prompt is issued.

```
TASK?
SA

ENTER OPTIONS
UN

ENTER SOURCE DISK LU
14

ENTER TAPE LU
<cr>
TAPE LU DEFAULTED TO        8
ENTER FILE NUMBER
<cr>
ENTER HARD COPY LU
<cr>
ENTER TITLE
OFFLINE 7900 UNIT SAVE

DEFINE SUBCHANNELS,LU=     14

#TRACKS,  1ST CYL,
SUBCHANNEL    00?
203,0

SUBCHANNEL    01?
203,0

SUBCHANNEL    02?
/E

SAVING DISK SUBCHANNEL  0 TO FILE   3 TAPE   1

TAPE NUMBER:  1  SAVE FILE:   3  USER:
PROGRAM: PSAVE   OPTIONS:    UN   DATE:  8:00 AM TUE, 14 JUL,1989
DISK LU:    14      TITLE: OFFLINE 7900 UNIT SAVE
SECTION:     1      (TRK      0  SEC      0)

SUBCHANNEL  0 SAVED ON FILE   3 TAPE   1

SAVING DISK SUBCHANNEL  1 TO FILE   4 TAPE   1

TAPE NUMBER:  1  SAVE FILE:   4  USER:
PROGRAM: PSAVE   OPTIONS:    UN   DATE:  8:00 AM TUE, 14 JUL,1989
DISK LU:    14      TITLE: OFFLINE 7900 UNIT SAVE
SECTION:     1      (TRK      0  SEC      0)

SUBCHANNEL  1 SAVED ON FILE   4 TAPE   1

PSAVE NORMAL END OF JOB
TASK?
```

Transfer to Input LU (TR)

Purpose: Causes the utility to transfer to the specified LU to read subsequent parameter inputs.

Syntax: `TR,lu`

lu The LU from which to read parameter inputs.

Description:

When the utility is initially booted, inputs are expected from the console device named in the configuration procedure. If a TR command is entered, the utility transfers to the specified LU for inputs.

A TR command can be entered in response to any prompt at the console or specified in the input parameter of the SA, RE, or CO commands. TR commands can also be nested; that is, if another TR command is encountered in the input, the preceding prompt is repeated, and the utility transfers to the named LU for the response.

If an error is encountered in the input or an end of file is read on the transfer LU, control returns to the console device (a TR,1 command is effectively executed) with an appropriate error message. The current activity is aborted and the TASK? prompt is issued.

If the transfer LU is the same as the destination LU on a save or the source LU on a restore, the utility will request that you mount the tape and enter GO when ready. The effect is that of a TR, 1 command, and subsequent parameter input is expected to come from the console device.

Tape Movement Functions

The offline utility also includes the following tape movement control commands:

`RW[,lu]`	(Rewind tape to the beginning.)
`FF[,lu[,n]]`	(Forward space tape n files. The default is FF LU 8 one file.)
`BF[lu[,n]]`	(Backspace tape n files. The default is BF LU 8 one file.)

Data Structures

This section describes the data structures used when saving disks to magnetic tapes or to CS/80 cartridge tapes (CTD). The data structures differ somewhat between the two device types, principally because the CTD is a fixed-block-length data-recording device, and the 800/1600-bpi magnetic tape transport is a variable-block-length data-recording device.

PSAVE Format on Magnetic Tape (Reel-to-Reel)

An LU save using PSAVE is written to magnetic tape in PSave Magnetic Tape File (PSMTF) format, where the end of the LU is indicated on tape by an end-of-file mark (EOF), and the end of the PSAVE operation (end-of-data) is indicated by a second EOF. This format is

where X (the tape position for the LU save starting point) is determined by the file number specified in the PSAVE runstring or by the current position of the tape if no file number is specified.

A UNit save is merely a series of LU PSave MT Files followed by EOFs. The save starts with the lowest number LU on the unit, proceeds through the highest number LU on the unit, and ends with a second EOF to mark the end of the PSAVE operation. A MUltiple LU save is in the same format except that the LUs are saved in the order in which they are specified in the command. The tape organization is

where:

n	is the number of LUs saved in the one PSAVE operation, and
X	(the tape position for the starting point) is determined by the file number specified in the PSAVE runstring or is the current position of the tape if no file number is specified.

The PSave Magnetic Tape File (PSMTF) format for an LU consists of several different types of records, as shown below.

Records 1 through 6 are ASCII information (header) records:

```
CREATED USING: PSAVE xxxxx-xxxxx REV.xxxx < yymmdd.tttt >
READ USING:    PRSTR xxxxx-xxxxx REV.xxxx < yymmdd.tttt >
TAPE NUMBER:    n   SAVE FILE:      n  USER: BILL.JT
PROGRAM: PSAVE      OPTIONS: aa  aa    DATE: xxxxxxxxxxxxxxxxxx
DISK LU:    nn      TITLE: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
SECTION:     n      (TRK   nnnn  SEC   nnnn)
```

Records 7 and 8 are save and disk definition records in a format similar to those built by physical backup utilities (with the exception of the leading checksums). The content of the save definition record (record 7) is:

Word	Content
1	Checksum
2	Checksum
3	Checksum
4	Checksum
5	Tape number
6	Unused (initialized to binary zero)
7	Number of saves remaining (Unit save only)
8	Word size of tape records
9	Word size of disk definition record
10	Options set for save (MU, UN, or LU)
11	Relative file number
12	Disk LU number saved
13	Disk type (from LDTYP)
14	Section number
15	Starting track number
16	Starting sector number
17	Operating System type (from .OPSY)
18−128	Unused (initialized to binary zero)

The content of the disk definition record (record 8) is:

Word	Content
1	Checksum
2	Checksum
3	Checksum
4	Checksum
5−15	Disk definition for this LU (from DSCPR)
16	Subchannel
17−25	Unused: initialized to binary 0
26−EOR	Disk definitions for remaining saves (contained only in the first LU PSMTF of a UNit save)

Records 9 through the end of the file contain data from the disk. Each of these records is an image of a certain number of sectors from the disk, except that the first four words of each record are used for checksums.

PSAVE Format on CS/80 Cartridge Tape

The format of an LU PSave to CS/80 Cartridge Tape File (PSCTF) is very similar to the PSAVE format for saves to magnetic tape. However, PSAVE places an end-of-file mark (EOF) in the first block of any cartridge tape that will contain PSAVE format data. This is included to provide protection from possible overlay of a disk if the pushbutton at the front of the CTD device is used. That is, when the pushbutton capability of the CTD is used on a PSAVE format tape, the EOF in the first block is read and, since this signifies the end of the file, the restore operation is immediately terminated and no data is overwritten on the disk.

The PSAVE Cartridge Tape File format for LU saves is shown below.

A UNit save is merely a series of LU PSave CT files followed by EOFs. The save starts at the lowest number LU on the unit, proceeds through the highest numbered LU on the unit, and ends with a second EOF to mark the end of the PSAVE operation. A MUltiple LU save is in the same format, except that the LUs are saved in the order in which they are specified in the command. The PSAVE Cartridge Tape File (PSCTF) format for MUltiple LU saves and UNit saves is shown below.

Note that any PSAVE operations on cartridge tape will begin in block 0 of the tape if file number 1 is specified or if the file number is allowed to default. This file-numbering distinction is described in more detail in the section on “Tape Handling.”

Since each record on a cartridge tape is a fixed length block of 512 words, the format of the tape files differs from that for magnetic tape. These differences in format for the PSave Cartridge Tape Files can be seen in the format of the blocks described below.

The first block of the file is a series of information (header) lines:

```
CREATED USING: PSAVE xxxxx-xxxxx REV.xxxx < yymmdd.tttt >
READ USING:    PRSTR
TAPE NUMBER:    n   SAVE FILE:      n  USER: BILL.JT
PROGRAM: PSAVE      OPTIONS: aa  aa    DATE: xxxxxxxxxxxxxxxxxx
DISK LU:    nn      TITLE: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
SECTION:     n      (TRK   nnnn  SEC   nnnn)
```

Note that the first word of this block is the length in words of the header lines. This length includes a carriage return and linefeed character that is added to the end of each line.

The save and disk definition blocks on cartridge tape (block 2 and block 3 of the file) are identical in format to the save and disk definition records on magnetic tape except that they are in fixed length (512-word) blocks.

The save definition block (block 2) format is:

Word	Content
1	Checksum
2	Checksum
3	Checksum
4	Checksum
5	Tape number
6	Unused (initialized to binary zero)
7	Number of saves remaining (UNit save only)
8	Word size of tape records
9	Word size of disk definition record
10	Options set for save (MU, UN, or LU)
11	Relative file number
12	Disk LU number saved
13	Disk type (from LDTYP)
14	Section number
15	Starting track number
16	Starting sector number
17	Operating System type (from .OPSY)
18−128	Unused (initialized to binary zero)

The content of the disk definition block (block 3) is:

Word	Content
1	Checksum
2	Checksum
3	Checksum
4	Checksum,
5−15	Disk definition for this LU (from DSCPR)
16	Subchannel number
17−25	Unused (initialized to binary 0)
26−EOR	Disk definitions for remaining saves (contained only in the first LU PSCTF of a UNit save).

Blocks 4 through the end of the file contain data from the disk. Each of these blocks is an image of 512 words of data from the disk, except that two identical checksum blocks are periodically written on the tape. These checksum blocks are written after every 480 blocks during the save operation, at the end of a save, and at the end of tape (if the end of tape is reached). Each of the checksum blocks has the following format:

Word	Content
1−480	A one-word checksum for each corresponding preceding block. That is, word 1 is the checksum for the first block following the last checksum block or definition records for the save file, word 2 is the checksum for block 2, ... word 480 is the checksum for block 480.
481−511	Unused, initialized to binary 0.
512	Checksum for this checksum block.

Pushbutton Image Format on CS/80 CTD

A pushbutton operation produces identical results whether it is performed using the front panel control on the cartridge tape drive or by using the pushbutton (PB) option programmatically from the backup utilities. In both cases, the controller in the device receives a request to do a "Copy Data" operation for the entire CTD tape or the full disk.

For disk-to-CTD saves, four 128-word disk sectors are read sequentially from the disk starting with block address 0. The data are passed through the buffer of the disk controller and written sequentially to the tape in 512-word blocks, starting with CTD tape block 0. The save terminates when either the end of tape is sensed or the last block on the disk has been written to tape. If the end of disk terminates the save, an end-of-file is written to the tape to mark the end of the backup operation.

If you are restoring a tape to disk, the copy terminates when the end of tape has been reached, the last block on the disk has been written, or an end-of-file mark is found on the tape.

When you use the backup utilities with the programmatic pushbutton option to perform the save or restore, you should specify the VE option to verify the destination data. The verification process is merely a request from the backup utilities to the disk controller to verify that all of the data on the tape/disk is correct, using the checksums internally generated with each block and stored on the tape/disk by the device controller.

With both copy methods, internal checksums moved with each block of data are used to verify (and sometimes correct) the data being read. This operation is automatic and is a function and feature of the hardware.

Error Messages

The following errors can be seen by the online and offline save, restore, and copy utilities. If the scheduling program for the online utilities is FMGR, the error number is returned to the 1P global. The utilities in which the error can be generated are indicated parenthetically following the definition of the message. Although only the online utilities are listed, the applicability of the error condition extends to the offline SA, RE, and CO tasks. Those errors that can occur offline only are identified at the start of the error-message description.

When an error results in operation abort, the error number, the name of the subroutine in which it occurred, and the utility that terminated is displayed at the log device.

1: SIZE UP THE PROGRAM!

Insufficient buffer space is available to perform the operation. (PRSTR,PSAVE,PCOPY)

2: INVALID PARAMETER

The utility terminated due to an invalid parameter input. This same error code is returned if the utility is terminated using /A or AB. (PRSTR,PSAVE,PCOPY)

3: TAPE MOUNT ERROR

The tape was not correctly mounted. (PSAVE)

4: TAPE COULD NOT BE LOCKED

The tape lock LURQ call failed. (PSAVE,PRSTR)

5: TAPE IS OFFLINE

The tape is not set to online. (PSAVE)

6: INVALID FILE NUMBER

A file number was given that was past the end-of-data mark. For a UN restore, this could mean that the file at which the tape is positioned is not the first file in the UN save. (PSAVE,PRSTR)

7: PUSHBUTTON [SAVE] [RESTORE] FAILED

A pushbutton save (PSAVE) or pushbutton restore (PRSTR) did not complete successfully. (PB option)

8: NO DISK PARAMETERS

A call to DSCPR returned no parameters. This is an internal error. (PSAVE, PRSTR, PCOPY)

9: BREAK SENSED

The break flag is set. (PSAVE,PRSTR,PCOPY)

10: UNRECOVERABLE VERIFY

The destination media failed to verify. (PSAVE,PRSTR)

11: SPARE ATTEMPT FAILED.

A disk that supports sparing could not be spared. (PRSTR,PCOPY, VE option only)

12: TAPE WRITE ERROR

An exec write request to the tape LU has failed. (PSAVE)

13: UNRECOGNIZED TAPE FORMAT

The tape is not in any supported format. (PRSTR)

15: NO MATCHING SUBCHANNEL

The destination LU of a unit restore has fewer subchannels than the source unit. Can be seen offline for SE restores. This message is seen if you try to do an online restore of a SAVE from-to save tape. (PRSTR)

16: DEST TRACK MISMATCH

The destination LU has a different number of tracks than the source LU. The decision has been made to abort the operation after being prompted to continue. (PRSTR,PCOPY)

17: DEST SECT/TRACK MISMATCH

The destination LU has a different number of sectors per track than the source LU. (PRSTR,PCOPY)

18: TAPE READ ERROR - HARDWARE

An EXEC read request from a tape failed due to hardware problems. Tape read checksum errors do not abort the operation, but are reported. (PRSTR)

Disk operations may cross track boundaries. The utility that reports the error does not know this and only reports the track number where the transfer started.

When sparing the reported track, check with either FORMT or FORMC to be sure it is the spared track (not a track following the reported track).

19: TAPE CHECKSUM BLOCKS BAD

Both checksum blocks on a cartridge restore contain bad data. This is a fatal error. (PRSTR)

20: DISK IS WRITE PROTECTED

The disk is write-protected. If subsequent operations exist (that is, an SE restore) you may elect to continue. (PRSTR,PCOPY)

21: FORMAT SWITCH IS OFF

The format switch for the destination disk LU is not on. If subsequent operations exist, you may elect to continue the operation. (PRSTR,PCOPY)

22: DISK LU LOCK FAILURE

An LURQ call to lock a disk LU failed, or a lock failed and you elect to terminate the operation. The disks are not locked offline, so this check is not made when the utility is running offline. (PRSTR,PCOPY)

If you are in session mode online, dismount any cartridges to be restored or copied to, but do not release them. If they are released, that disk LU is removed from your SST (Session Switch Table) and an LU that is not in your SST cannot be locked, resulting in this error.

23: DISK IS MOUNTED

A disk LU is mounted, and you elected to terminate any remaining operations. This check is not made when running the utilities offline. (PRSTR,PCOPY)

24: BAD LU WAS ACCESSED

An internal error. If this error is encountered, contact your HP service representative for assistance. (PRSTR,PCOPY)

25: UNEXPECTED EXEC CALL FAILURE

An internal error in the same category as error 24. Offline, however, it could indicate that incorrect track map entries were defined. This would likely be due to defining disk attributes in a way that contradicts the hardware characteristics, such as too many cylinders or too many sectors per track. (PRSTR,PCOPY)

26: UNEXPECTED END OF DATA

An end-of-data was encountered with some tape reads still expected. This should only occur if the tape save using PSAVE is halted by setting the break flag. (PRSTR)

27: UNEXPECTED END OF FILE

An end-of-file was encountered with some tape reads still expected. If this error occurs, contact your HP service representative. (PRSTR)

28: SE RESTORE, NOT PSAVE TAPE

An SE restore was selected, but the saved tape is not in PSAVE format. The SE option is not supported in this case. (PRSTR)

29: UNIT RESTORE, NOT UNIT SAVE

A UN restore was selected, but the saved tape is not a UN save. (PRSTR)

30: INVALID TAPE NUMBER MOUNT

An incorrect tape was mounted for a restore operation. Initially, this must be tape number 1, and subsequent tapes must be sequentially ordered. (PRSTR)

31: INVALID MU SAVE

An MU save was requested from disks of differing classes.

32: PB RESTORE, NOT PB SAVE

A PB restore of a non PB saved tape was requested. (PRSTR)

33: INVALID TRANSMISSION LOG

The transmission log on a tape read was not of the expected value. (PRSTR)

34: NOT ENOUGH SUBCHANNELS

(Offline only) Insufficient track-addressing subchannel definitions were specified. The original operation cannot be completed; however, you can continue and the utility ignores any subchannel numbers larger than the maximum number defined. (PSAVE,PRSTR)

35: INVALID TR COMMAND

(Offline only) An illegal TR command was issued. The command is ignored and the utility continues using the previous input device. (PSAVE,PRSTR,PCOPY)

36: UNRECOGNIZED MODEL NUMBER

(Offline only) When specifying the track map, an unidentifiable model number was specified. The command is ignored. (PSAVE,PRSTR,PCOPY)

37: DISK MODEL TYPE CHANGED

(Offline only) A disk model number was specified that is different from the source or destination LU. The command is ignored. (PSAVE,PRSTR,PCOPY)

39: PASCAL RANGE ERROR

(Offline only) A specified track parameter is larger than its field in the track-map structure. This is a fatal error. (PSAVE,PRSTR,PCOPY)

40: SON PROGRAM NOT SCHEDULED PROPERLY

(Offline only) The offline utilities check for proper scheduling from the father program to ensure that the common routines are correctly initialized and to prevent problems with sharing resources by more than one utility. Note that the offline utilities may not be run using the FMGR RU command. (PSAVE,PRSTR,PCOPY)

41: UNEXPECTED ERROR

This error should not be seen in the debugged programs. However, if it should occur, contact your HP service representative for assistance. (PRSTR,PCOPY)

42: TAPE READ ERROR-CHECKSUM

The tape information was invalid. This error does not cause program termination. (PRSTR)

43: XXXXX SCHEDULE ERROR

The son program (for example, PSPAR) could not be scheduled. Program probably is not loaded. (PSAVE,PRSTR,PCOPY)

44: INVALID PB DEVICE(S)

A pushbutton operation was requested for non-CS/80 devices or for CS/80 devices not on an integrated unit. (PSAVE,PRSTR)

45: TAPE NOT INITIALIZED

A save to a CS/80 cartridge was requested to an uninitialized tape medium. FORMC should be used to initialize the tape, and the utility should be rerun. (PSAVE,PRSTR)

46: SON PROGRAM TERMINATED ABNORMALLY

A son program scheduled by a backup utility (for example, PSPAR or FORMC) was terminated, most likely using the FMGR OF command. (PSAVE,PRSTR,PCOPY)

47: CANNOT DEFAULT DISK LU

(Offline only) The disk LU has been defaulted in an offline system and the utility cannot find an LU of the appropriate type. (RE)

48: FROM-TO SAVE INVALID HERE

An attempt was made to restore a from-to SAVE tape online or to specify an LU/SE/PB option. (PRSTR)

49: CAPABILITY <60

(Online only) A user must have a capability greater than or equal to 60 in order to do a PushButton restore online to a disk that contains LU 2 or LU 3. (PRSTR)

4

File Copy and File Interchange Utilities

RTE-6/VM includes three utilities, FC, TF, and FST, that copy files between disk cartridges and tapes. FC copies files in the FMGR file system and supports disk-to-disk, disk-to-tape, and tape-to-disk copying. TF and FST can copy in both CI and FMGR file systems. They support only tape-related operations (they do not support disk-to-disk copies). TF and FST support incremental backups of just those files that were modified, and they allow you to append new backups to old backups. FST is faster than TF and supports streaming on streaming tape drives.

FC, TF, and FST can also be used to interchange files between RTE systems (RTE-A, RTE-6/VM), other Hewlett-Packard systems, and UNIX-based systems.

The LIF utility is used primarily for file interchange but can also provide file backup. LIF translates a file from the RTE FMP format to a standard Logical Interchange Format (LIF) and vice versa.

Refer to the individual sections in this chapter on FC, TF, FST, and LIF for a complete description of each utility.

CI is the RTE-6/VM command interpreter and can be used to copy files on disk. See the *RTE-6/VM CI User's Manual* for a complete description of CI.

File Interchange on RTE-6/VM

You can use the FST, TF, FC, and LIF utilities to interchange files between RTE systems (RTE-A, RTE-6/VM), other Hewlett-Packard systems, and UNIX*-based systems. These utilities support a Verify option to ensure the validity of data copied. Table 4-1 shows which utilities to use to move data between systems. The table indicates which utilities are recommended for saving and restoring files from/to a system. For example, "FST/TAR" means that FST is recommended for copying files from one system (the source), and TAR is recommended for restoring files to the other system (the destination).

Table 4-1 . File Interchange Utilities

From System \ To System	RTE-A or RTE-6/VM (4) CI files	RTE-A or RTE-6/VM (4) FMGR files	UNIX
RTE-A or RTE-6/VM(4) CI Files	FST/FST	FST/FST	FST/TAR (3)
RTE-A or RTE-6/VM(4) FMGR files	FST/FST	FST/FST	(2)(3) FST/TAR
UNIX	(3) TAR/FST	(1)(3) TAR/FST	_

Notes:
(1) File names truncated to 6 characters, time stamps lost
(2) Restrictions on use with FMGR files (see TF/FST)
(3) UNIX compatibility restrictions (see TF/FST)
(4) Revision 2540 or later

*UNIX is a registered trademark of UNIX System Laboratories, Inc. in the U.S.A. and other countries.

File Storage to Tape (FST)

FST is a high performance, logical (file-by-file) backup utility. It copies files faster than TF and FC because of its streaming capability. It also performs faster than TF and FC on tape units that do not support streaming. FST supports back ups and restores to and from magnetic tape, CS/80 cartridge tape drives, and archive files on disk.

FST reads files from and writes files to both CI volumes and FMGR cartridges. FST does not, however, back up type 0 files. This utility saves files with all the needed extent information so that you can restore them to their original layout, if desired. FST replaces reserved characters in FMGR file names (for example, ".", "/", or "@") with non-reserved characters and sends a message to the terminal or log device/file when it renames a file.

A virtual memory scheme, using a scratch file, lets you save or restore a virtually unlimited number of files. The only restriction is imposed by the amount of available disk space where the scratch file exists. Any overflow of the scratch file is reported before the files are saved. You may decide where to locate the scratch file.

FST uses a two-pass approach to its backup and restore process. First, you select the files to back up or restore. The selected file names and some other file information are kept in a directory file on the disk. During this first pass, you can add files, remove files, and list file names in the directory file. You may also perform other functions, such as setting the log device/file or changing the selected tape LU.

The second pass begins after you select all the desired files and set all the other backup/restore parameters. During this pass, the files you selected and the directory file itself are transferred between tape and disk. File names and file masks for FST commands conform to FMP standards.

You can use multiple reels for large backups that require more than a single tape. Since a file can cross tape boundaries, files that are larger than an entire tape and multiple files that require more than one tape can be handled.

Each archive created by FST contains a header, an optional comment file, and a directory file, followed by the files saved from disk. A file header immediately precedes each saved file.

You may run FST interactively or programmatically. Parameters in the command string determine the mode. Interactive and programmatic mode are discussed in the following sections.

Calling FST

You may run FST programmatically by entering all the desired commands in the runstring. When FST executes all the commands, or an unrecoverable error occurs, FST exits. The maximum length of the runstring is the limit imposed by CI, 256 characters.

Separate commands in the runstring by a vertical bar (|). You must rename files used in the runstring if their names contain a vertical bar.

FST aborts if an error occurs in programmatic mode. If FST ends abnormally, the error is indicated by a "−1" in the first return parameter kept by CI and FMGR.

In the following example, all the files in the working directory are backed up to LU 8 with the Verify option selected:

```
CI> fst ba @|verify|mt 8|go
```

If you do not enter any commands in the FST runstring, FST assumes you are using interactive mode. Commands are entered at the FST> prompt. FST executes each command before prompting for the next one and continues to prompt for commands until you enter the EXIT command, as follows:

```
CI> fst
FST> <command>
FST> <command>
.
.
.
FST> ex
CI>
```

You may enter just the first two characters of a command or up to the whole name; for example, EX, EXI, and EXIT all cause FST to exit.

To specify more than one command on a line, separate the commands by a vertical bar (|) as follows:

```
CI> fst
FST> <command> | <command> | ... | <command>
FST>
```

When FST is scheduled, it will look for the start-up command file, FST.RC, in the user's home directory. If FST cannot find a FST.RC file in the home directory, it will look for /CMDFILES/FST.RC. Every command in the FST.RC file will then be executed, without the normal command echoing, before any runstring or interactive command is processed. Any valid FST command can exist in the FST.RC file. Note that since FST does not allow nested transfer files, it is not possible to transfer to another file from the FST.RC file.

The FST.RC file can be used to set up defaults for FST. For example, the file can be used to override the normal defaults for any of the FST options. The file can also be used to set up a default output device by specifying the MT command in the file.

When FST exits normally, the $RETURN parameters are set as follows:

$RETURN1 = Number of errors encountered during the last save/restore pass.
$RETURN2 = Number of verify errors during the last verify pass.
$RETURN3 = Number of warnings generated during the last pass.
$RETURN4 = Number of files saved.
$RETURN5 = Number of files restored.

FST Commands

Table 4-2 summarizes the FST executable commands.

Table 4-2. FST Commands Summary

Commands		Description
Information Commands		
HElp (or ?)		Provide a summary of commands and syntaxes
SHow		Display the DF, MT, TI, SC, LL, and option settings
Backup/Restore and Related Commands		
BAckup	mask	Select files to back up
DF Directory Files	file_desc	Specify a non-default directory file
GO Begin Backup/Restore		Begin executing backup/restore
REstore	mask	Select files to restore
SC Select Comment File	filename	Select the tape's comment file (backup only)
TA UNIX TAR Format		Select UNIX TAR archive format
TItle	title	Specify a title for the archive (backup only)
UNselect	mask	Unselect files
Listing Commands		
DL Directory List	mask	Display the archive's directory file
LC List Comment File		List the archive's comment file
LH List Header		List the archive's header
LIst Selected Files	mask	List the files selected for backup/restore
LL Select Log Device/Files	device/file	Set log device or file for FST activities
LN List Non-Selected Files	mask	List the non-selected files (restore only)
Tape LU Control Commands		
MT Specify Tape LU or archive file	tape_LU/ file name	Set the magnetic tape LU or archive file
NExt	append_#	Advance the tape to another append
POsition	append_#	Position the tape to a specific append
PRevious	append_#	Rewind the tape to a previous append
SD Set Tape Density	density	Set the tape density (HP 7974/7978 only)
SEcure		Lock the tape LU and check the tape status or open archive and check the file's status
Transfer and Exit Commands		
EXit		Exit FST
RUn	program	Run an external program
TR Transfer to Command File	filename	Begin executing a transfer file
/ Command Stack		Display the command stack

Command Stack (/)

Purpose: Displays the FST command stack.

Syntax: `/[n]`

n	The optional command line count that specifies the number of command lines from the last command entered to be displayed.

Description:

FST uses the RTE standard command stack, which supports finds, page movement, line marking, and various other operations. For a full description, refer to the online help by typing "? STACK" from CI.

Backup (BA)

Purpose: Selects a file or group of files to back up from disk.

Syntax: `BA mask [dest_mask] [sec_code]`

mask	The file or group of files to be backed up from disk. (This can be a disk LU number).
dest_mask	The optional mask that changes the characteristics of the files (for example, the path name and file type extension) as they are saved on the archive.
sec_code	The system master security code.

Description:

To preserve the security codes when saving FMGR files, specify each file with its individual security code, or enter the system master security code with the mask. If this is not performed, FST will not restore the security code with the FMGR file.

You may execute BA as many times as desired before you start the data transfer to tape. Note that you may not use both the BA and RE (Restore) commands when you select files to transfer.

Directory File (DF)

Purpose: Specifies the name and location of the directory file.

Syntax: `DF file_desc`

file_desc — The partial file descriptor that can include a path name, file name, and block size.

Description:

The file descriptor can specify any directory or FMGR cartridge that is not write-protected. FST uses the default path name, file name, or size of the directory file if you do not enter them. The default location of the directory file is the /SCRATCH global directory. If /SCRATCH does not exist, FST creates a directory file on the first available FMGR cartridge. The default file name is a unique file name created by the FmpOpenScratch call. The default size is 500 blocks.

Each selected file requires a minimum of two blocks in the directory file. For large backups that require more than 500 blocks for the directory file, extents are needed for the directory file, and directory file access is slower. To avoid extents, specify the size of the directory file.

Although FST creates a directory file if you do not, DF lets you create directory files for a particular location, name or size. Note that DF can be used only when no files have yet been selected using BA or RE. The following example specifies location, name and size:

```
FST> DF /BigLU/FSTdirFile::::10000
```

The next example just specifies the location:

```
FST> DF /LonelyDisk/
```

List Directory (DL)

Purpose: Lists the directory of files on the archive.

Syntax: `DL [mask]`

mask — The optional mask that specifies a file or group of files in the archive directory. (This can be a disk LU number.)

Description:

The directory of the archive is displayed and logged to the user log device/file. If you do not specify a mask, the entire directory on the archive is displayed; otherwise, only those files in the archive directory that match the mask are displayed. DL does not modify the list of selected files already obtained by the BA or RE commands.

Exit (EX)

Purpose: Exits FST.

Syntax: `EX`

Description:

This command returns you to the environment from which FST was run. (See the section on "The Keep (K) Option" later in this chapter for information on exiting without waiting for the tape to rewind.)

Begin Backup/Restore (GO)

Purpose: Begins the data transfers to or from the archive.

Syntax: `GO`

Description:

After you select all the files for the backup/restore, use GO to start the actual transfer of the files.

Help (HE)

Purpose: Displays the command help information.

Syntax: `HE [command]`

or

`? [command]`

command Any FST command or option.

Description:

If you do not specify the optional command, general help information is displayed; otherwise, information for the specified command or option is shown.

List Comment File (LC)

Purpose: Lists the comment file of the archive.

Syntax: `LC`

Description:

The comment file from the archive is displayed and logged to the log device/file.

List Header (LH)

Purpose: Lists the archive header.

Syntax: `LH`

Description:

The archive header (format, title, and creation date) is displayed and logged to the log device/file.

List Selected Files (LI)

Purpose: Displays all the files that were selected for backup/restore.

Syntax: `LI [mask]`

mask	The optional mask that specifies a file or group of files in the directory file to be displayed. (This can be a disk LU number.)

Description:

If you do not specify a mask, all selected files in the directory file are displayed and logged to the log device/file. Otherwise, just those files that match the specified mask are displayed and logged. Refer to the "Disk Directory File" section later in this chapter for a description of the directory file.

Select Log Device/File (LL)

Purpose: Changes or selects the log device/file.

Syntax: `LL <device/file> [a] [o]`

device/file	The device or file to which the output from the listing commands and FST messages is routed.
a	The option to append to the specified log file, if one exists.
o	The option to overwrite the specified log file, if it exists.

Description:

Using a log file lets you check the file listings and FST messages following a backup/restore. Specifying LL 1 routes the output to your terminal.

List Non-Selected Files (LN) (Restore Only)

Purpose: Displays the non-selected files; that is, the files that are on the archive but are not being restored to disk.

Syntax: `LN [mask]`

mask	The optional mask that specifies a file or group of files on the tape. (This can be a disk LU number.)

Description:

If you do not specify a mask, all non-selected files in the directory file are displayed and logged to the log device/file. Otherwise, only those that match the mask are displayed and logged. This command should be used only during a restore operation.

Specify Tape LU/Archive File (MT)

Purpose: Selects the tape Logical Unit (LU) number or an archive file name.

Syntax: `MT [tape_LU | filename]`

tape_LU The selected tape LU number.

filename The file descriptor of the archive file.

Description:

If you do not specify a tape LU, the current tape LU is set to 0. If you specify an LU other than a tape LU, an error is reported, and the LU value is set to 0. If you specify an LU other than the current tape LU and the current tape LU is locked by FST, FST unlocks the current tape LU and may take it offline before it selects the new LU. Refer to the section on the Keep option later in this chapter for information on when the tape LU is taken offline.

If you specify an illegal tape LU or LU 0, you must use MT to set a legal tape LU before you can execute a backup or restore operation.

If an archive file is specified instead of a tape LU, FST uses this archive file for backups or restores. Archive files are type 1 files and the data can be in either FST format or TAR format. Using an archive file with FST is essentially the same as using a tape. The main difference is that the tape control commands NE, PO, PR and SD are not supported with archive files and the append feature is not supported.

TAR archives are written with a blocking factor of 20 (where one TAR block equals 512 bytes).

Next (NE)

Purpose: Advances the tape to another append of data.

Syntax: `NE [append_#]`

append_# The number of appends to move the tape forward.

Description:

If you do not enter an append number, the tape moves forward to the next append of data. This lets you move the tape and examine various backups that were appended to the tape. If the append number is larger than the number of remaining appends, the tape is positioned at the last append.

Position (PO)

Purpose: Positions the tape at a specific append.

Syntax: `PO [append_#]`

append_# The specific append number to which the tape is moved.

Description:

If you do not specify an append number, the current position is reported. The main backup of the tape is position zero (0). The first append is position one (1), and so on. If you specify an append number larger than the number of appends on the tape, the tape is positioned at the last append.

Previous (PR)

Purpose: Rewinds the tape to a previous append of data.

Syntax: `PR [append_#]`

append_# The number of appends to move the tape backward.

Description:

If you do not specify an append number, the tape is rewound to the previous append of data. This lets you move the tape and examine various backups that were appended to the tape. If you specify an append number that is larger than the number of previous appends on the tape, the tape is positioned at the beginning of the tape.

Restore (RE)

Purpose: Selects a file or group of files from the mounted FST archive to restore to the disk.

Syntax: `RE [mask] [dest_mask] [gr|eg|ag]`

mask	The optional mask that specifies the file or group of files to be restored to disk. If you do not use a mask, all the files on the archive are restored. (This can be a disk LU number.)
dest_mask	The optional mask that renames the characteristics of the files (for example, the path name and file type extension) as they are restored to disk. (This can be a disk LU number that is used by FST if it needs to create a global directory that does not already exist on the system.)
gr	Begin group restore.
eg	End group restore.
ag	Abort group restore.

Description:

You may execute RE as many times as desired before you start the data transfer from the archive. Note, however, that RE cannot be used with BA.

A mask is used to select a specific file or group of files. If you do not use a mask, all the files on the archive are restored with the same path and, if possible, on the original disk LU.

If you are trying to selectively restore many files from a backup tape with a large directory file, the selection process can take quite a long time before the data transfer even begins. This is because when each RE command is executed, a pass is made through the entire directory file. Note that the files are not actually restored until you issue the GO command. Note that with TAR and TF format the GR option is not allowed. RE is handled differently in these cases. See the section TAR or TF Compatibility for more information.

You may use Group commands to speed up large restores. When FST encounters a GR command, it keeps track of all subsequent RE commands, but does not execute them immediately. Instead, they are all executed at once, as a single operation, when the EG (End Group) command is encountered. Thus FST can make a single pass through the directory file, matching all the masks within the group.

Streaming may not occur when you use RE, depending upon the type of tape drive, the size of the SHEMA (shareable EMA) buffer, and the size of the files being restored. However, restore operations imply extraordinary circumstances, such as a corrupt disk or a system that is down, and thus occur much less frequently than backup operations.

Run (RU)

Purpose: Runs a program external to FST.

Syntax: `RU prog_name`

prog_name The name of the program to run.

Description:

RU provides more flexibility when you run FST, as it lets you execute non-FST commands without exiting FST and losing the current FST settings and selections.

For example, you can obtain a directory listing from disk by using the CI DL command with the following command string:

```
FST> ru dl /progs/@.ftn
```

Or, you can create a comment file to save to the tape from within FST by running EDIT/1000 with the following command string:

```
FST> ru edit /myfiles/fst/comments
```

Select Comment File (SC) (Backup Only)

Purpose: Selects the comment file for the archive.

Syntax: `SC [filename]`

filename The file name of the comment file for the archive you are backing up.

Description:

If you do not specify a file name, the current comment file is no longer selected. Comment files are useful for detailed archive identification or restoration instructions.

Set Tape Density (SD) (Backup Only)

Purpose: Sets the tape density for the HP streaming magnetic tape drives.

Syntax: `SD [density]`

density The desired density in bpi for writing to the tape.

Description:

If you do not specify a density number, the current tape drive density setting is displayed. Setting the tape density does not apply to appends; all data on one tape must be at the same density. Refer to your tape drive manual for the proper tape drive density setting.

Secure (SE)

Purpose: Secures (locks) the tape LU and checks the tape status or secures (opens) the archive file and checks the file's status.

Syntax: `SE`

Description:

SE locks the tape LU specified in the MT command and makes sure the tape is online. SE lets you immediately protect an online tape from other users if it is not write-protected. Using MT to specify a different tape LU unlocks the current tape LU.

Show (SH)

Purpose: Shows the user-selected states of the FST program.

Syntax: `SH`

Description:

SH displays the states of the option commands, the tape LU or archive file selected, the file count with the number of 128-word blocks and kilobytes represented by the file count, your title and comment file (backups only), the name of the directory file, the log file, and the amount of tape footage needed for the tape access.

TAR (TA)

Purpose: Specifies the UNIX TAR format for the archive.

Syntax: `TA, [ON/OFF/A/B][,c]`

ON	Turns ON the TA option. For backups, only type 4 files are converted to a UNIX file format. For restores, all selected files are restored as converted ASCII files. ON is the default.
OFF	Turns OFF the TA option.
A	(ASCII) Turns ON the TA option and, for backups, converts type 3 and above files to UNIX file format. Restores are handled the same as with TAR ON.
B	(Binary) Turns ON the TA option for binary file formats. For backups, files are not converted to UNIX file format and are saved as blocks of data. For restores, all selected files are restored as blocks of data from the archive.
C	Allows the selection of case sensitive file names from a TAR archive.

Description:

Although TA resembles an option, it is a command and thus cannot be entered in a string of options.

When you back up files, you must select all files under the same format. In other words, you may not back up some files in TAR format and other files in FST format on the same archive.

When you restore a TAR archive, FST sets the TA option to ON automatically upon recognizing the archive format. On restores, the default setting restores and converts all files as ASCII files. If a binary restore is required, you must specify the B parameter.

When the 'C' option is enabled, the FST 'RE' command is case sensitive. Also, the FST 'DL' command preserves the case of the file names on the TAR archive when the directory of files on the archive is displayed. Since UNIX files are case sensitive, it is possible to have multiple files on the same archive which, when shifted to uppercase, result in the same name.

For example, to restore both of the “readme” files from a TAR archive that contains ‘README’ and ‘readme’:

```
FST> ta,,c
FST> re,README,uppercase
FST> re,readme,lowercase
FST> go

Tape format:        TAR

Copying README : : : 4 : 1 to UPPERCASE : : : 4 : 1
Copying readme : : : 4 : 1  to LOWERCASE : : : 4 : 1
```

Title (TI) (Backup Only)

Purpose: Specifies a title for the archive header.

Syntax: `TI title`

title — The text, up to 72 characters long, that describes the contents of the backup.

Description:

You can use the LH command later to examine the title for the current backup in order to help identify the archive. If you do not specify a title, the last BA command mask entered is used as the default.

Transfer to Command File (TR)

Purpose: Transfers control to a command file.

Syntax: `TR filename`

filename — The name of the command file.

Description:

FST executes commands from the file you specify. This is advantageous when you use the same sequence of commands frequently. A command file is quicker to reference, removes the possibility of typing errors, and does not require operator intervention to execute a series of commands.

To use other commands along with TR in a runstring, specify them before the TR command. FST does not execute any commands in a command string that occur after TR.

Command lines that begin with an asterisk (*) are ignored by FST and can be used to include comments in a command file.

Command files cannot be nested.

Unselect (UN)

Purpose: Removes a file or a group of files from selection in the directory file.

Syntax: `UN [mask]`

mask — The optional mask that specifies the file or group of files to unselect from the directory file. (This can be a disk LU number.)

Description:

If you do not specify a mask, UN unselects all the files and purges the directory file from the disk. This command only modifies the directory file and does not affect the files on tape or disk. You can use UN as many times as needed before you actually begin the data transfer to/from the archive.

FST Options

Options enhance the usability of backing up and restoring files. Table 4-3 provides a summary of the FST options. Note that all options do not apply to all commands. You may specify the options, in any order, by entering the first character of the option or up to the entire option name. For example, B, BR, BRI, BRIE, and BRIEF all specify the Brief option.

The SH command displays the current state of the options. All options are initially OFF. Currently selected options are ON.

Set options by entering one or more options on a line and specifying ON or OFF. ON is the default. You may specify an option on the same line as an FST command; to do so, separate the option from the command by a vertical bar (|). You need not set all the options to be used for one backup or restore operation on the same line. The syntax for setting options in interactive mode is as follows:

```
FST> option [option]...[option] [ON/OFF]  [option]...[option]  [ON/OFF]
```

This allows multiple options to be set ON or OFF in one line.

You may also specify options in the FST runstring as follows:

```
CI> fst option [option]...[option] [ON/OFF] [option]...[option] [ON/OFF]
```

The examples that follow show how to set options ON or OFF (the examples are entered in interactive mode, but options can also be entered in the FST runstring from the CI> prompt, as shown above):

Example 1: Set Verify option ON.

```
FST> verify on
```

Example 2: Set Append and Clear options ON.

```
FST> a c
```

Example 3: Set Brief, Keep, and Yes options ON; set Lock, Original, and Purge options OFF.

```
FST> b on lock off keep y on orig pu off
```

Table 4-3. FST Command Options Summary

Options	Description
Append	Append this backup to the data already on the tape.
Brief	Only show errors and status messages.
Clear	Clear the disk file's backup bits.
Duplicate	Replace duplicate files.
Faulty	Restore files from a partially overwritten tape.
Inhibit	Inhibit the tape rewind between backups.
Keep	Keep tape online when backup/restore is complete.
Lock	Lock any disk LUs used.
MinDir	Minimize the FST directory file size during restores.
Normal	Back up symbolic links as normal files (follow links). (RTE-A, VC+ only)
Original	Restore files to their original main size.
Purge	Purge the disk files after backing up the files.
Quiet	Report messages only to the log device/file.
RwndOff	Rewind and go offline at exit.
SrchApp	Search through appends during RESTORE.
Update	Replace duplicate file if file has been updated.
Verify	Verify the files during the backup/restore.
Whole	Back up all the blocks reserved for the file.
Yes	Write over the tape without asking.
Z	Pause during restore phase for disk full errors.

Append (A) (Backup Only)

Append adds the backed-up files at the end of the tape contents, rather than replacing the previous files. You can only append to tapes that you previously backed up using FST. The Append option is used only with the BA command.

If the mounted tape is not an FST tape, but the Yes option is ON, the tape is overwritten.

Brief (B)

Whenever you do a backup or restore, a "Copying" message is displayed and logged to the log device/file for each file copied. If you specify the Brief option, these messages are not displayed, and only the start and stop of the backup/restore, along with any errors that occur during the process, are shown.

Clear (C)

This option clears the backup bit of copied files that were backed up or restored. In the hierarchical (CI) file system, each file has a backup bit, which indicates whether the file was backed up. When you specify Clear, Verify is automatically ON. If you do not want to verify the backup, Verify may be turned OFF without affecting the status of the Clear option.

Duplicate (D) (Restore Only)

The Duplicate option lets you replace any file on the disk that has the same name as a file being restored from the tape. If you do not specify the Duplicate option, files with duplicate names are not restored.

Note that when Duplicate is ON, Update is turned OFF, and vice versa.

Faulty (F) (Restore Only)

The Faulty option lets you restore FST files from a partially overwritten tape in which the original directory file and probably some actual file data were partly or completely destroyed. Simply position just beyond the overwritten portion of the tape, then build a directory file from the remaining, uncorrupted data. For more detailed information, see the section "Rescuing Files from an Overwritten Tape."

Inhibit (I)

This option inhibits the normal rewind that occurs after a backup. This does not inhibit the rewind that occurs after FST exits. The Inhibit option is useful when making a backup tape consisting of several appends.

Keep (K)

The Keep option causes the tape to remain online/loaded after you exit FST or select another tape LU. Online refers to 1/2-inch magnetic tapes; loaded refers to CTD (cartridge tape drive) tapes. Note that if the tape is left online/loaded, another user can write over the tape if it is not write-protected. Leaving the tape online also lets FST exit without waiting for the current rewind to complete (except for HP 797x streaming magnetic tapes).

FST uses the following rules for leaving the tape online/loaded:

- If the tape is write-protected, it is left online/loaded, independent of the Keep option setting.
- If the tape is not write-protected and the Keep option is not set, the tape is taken offline/unloaded.
- If the tape is not write-protected and the Keep option is set, the tape is left online/loaded.

Note that the Keep option assists when using the HP 35401A Autochanger. Refer to the "Multiple Reels" section in this chapter for more information.

Lock (L) (Backup Only)

Setting the Lock option ON locks the disk LUs that are accessed during the backup selection. Setting this option OFF unlocks all locked disk LUs. The Lock option is useful because of the two-phase process that FST uses for its backup operation.

There is a time lapse between the first pass (when all the file information is collected for each file) and the second pass (when the files are transferred). If any file changes during this interval, incorrect or incomplete data may be saved to the tape. The Lock option prevents this by locking the disk LUs needed for collecting data.

This option must be used carefully. When a disk LU is locked, all other users are prevented from accessing that LU. Commonly used disk LUs should only be locked for a short period of time.

MinDir (M) (Restore Only)

The MinDir option is only available when performing an FST restore. Normally, FST restores the entire directory file to disk when restoring any files from an archive. Use of this option limits the user to a single restore mask or group with the RE command.

Normal (N) (Backup Only; RTE-A VC+ Only)

The Normal option causes FST to back up the data of the file pointed to by a symbolic link. The default behavior of FST is to back up the symbolic link file itself. (This option is only functional on RTE-A VC+ systems.)

Original (O) (Restore Only)

The Original option lets you restore files to the disk with their original main block size. Usually, files of type 3 and above are restored to disk at a block size that contains all the data in the file without creating extents. Sometimes, however, a file must be restored with the same main block size as when it was backed up. You can specify the Original option to accomplish this. This option has no effect on type 1 or type 2 files; they are always restored in their original format.

Note that the Original and Whole (described below) options are not identical. Original determines the main size of a file being restored, while Whole determines how much of a file is backed up.

Purge (P) (Backup Only)

Purge lets you purge the source disk files after they are backed up and verified by FST. When you select Purge, Verify is automatically set (files cannot be purged without verification). The Purge option does not apply to the RE command.

Quiet (Q)

The Quiet option prevents FST output from being displayed on a terminal. Any errors, warnings, or messages are placed in the log device/file. You should specify a log device/file when you use this option; otherwise, it is almost impossible to determine the output of the backup or restore operation. This option is most useful when used programmatically or from a transfer file, but is allowed interactively.

RwndOff (R)

The RwndOff option is mutually exclusive of the Keep option. This option causes the tape to rewind and go offline when FST exits, regardless of the write protect status of the tape.

SrchApp (S) (Restore Only)

The SrchApp option causes FST to search automatically through all of the appends on an FST tape when restoring files. Because this option would typically be used on an FST tape that contains incremental appends, selecting this option automatically enables the UPDATE option. Selecting SrchApp will also initialize a group restore (see Restore command). The file names and/or masks from each RE command are stored and used later to search each append's directory file. Restarting grouping, for example,

```
RE [mask] [dest_mask] GR
```

will have the effect of reinitializing grouping, and the previous RE commands will be invalidated. After the GO command is issued, the directory file for the current append is searched for selected files. After restoring any selected files, the tape is positioned to the next append and the process

repeated until the last append has been searched. The SrchApp option can only be used with tapes in FST format.

When SrchApp is enabled, the files may not be selectively unselected with the unselect (UN) command. An unselect command (specified without a mask) may be issued to unselect all of the previous selections. This has the same effect as restarting grouping with a RE command.

Update (U)

Update causes FST to restore any duplicate files whose update times on the archive are later than the update times on the disk. This option does not apply to FMGR cartridge restoration, since FMGR files do not have update times.

Update and Duplicate cannot be used at the same time. When the Update option is turned ON, the Duplicate option is turned OFF, and vice versa.

Verify (V)

When you specify the Verify option, FST goes through another pass of the archive after the files are backed up or restored and compares the data on the tape with the data on the disk to verify that the data was transferred correctly. Streaming may not occur during the verify pass, depending upon the tape drive, the size of the SHEMA buffer, and the size of the files.

Whole (W) (Backup Only)

Normally, FST uses the end-of-file position, specified in the disk directory for the file, to determine how many blocks of data to save to tape. When Whole is ON, FST ignores the end-of-file position and copies all the blocks reserved for the file.

When you back up files to the tape, there is usually no need to save any data beyond the end-of-file position specified in the disk directory entry of each file. When the Whole option is OFF, the end-of-file position in the disk directory is used to calculate how many blocks of data are actually saved to tape. However, sometimes the end-of-file position is corrupt or does not accurately represent the data to be saved. In that case, the Whole option should be ON.

This option only applies to the BA command. Specify Whole as follows:

1. Set Whole ON.
2. Enter the BA command or commands for the files to be copied with Whole.
3. Set Whole OFF.
4. Enter the BA command for the files to be copied without the Whole option.

Yes (Y)

When you write to an archive that already has data on it, FST asks if you want to write over the archive. You may use the Yes option to suppress this question and have FST copy over the archive automatically.

Z (Z)

The Z option causes FST to pause when the restoration of a file causes a disk full error. This allows the user to free some disk space and restart FST with a "GO" command. FST will then retry the restore of the file that caused the error.

File Masking and Renaming

Although FST masking is designed to be consistent with CI masking, there are differences, depending upon which command is being executed. Any differences, however, should not affect situations where data could be lost ("unsaved").

The BA and RE commands refer to copying files, so a D qualifier (described below) is forced into the mask. The DL (List Directory), LI (List Selected Files), LN (List Non-Selected Files), and UN (Unselect) commands simply display file information, so no qualifiers are added by FST. The D, K, N, and S qualifiers are described in the next section.

D, K, N, and S Qualifiers

The D, K, N, and S qualifiers are used when you select files to copy or display. All the qualifiers can be used together; however, the K qualifier overrides the D.

The D qualifier is forced for the BA and RE commands. It has two functions:

- If any directory matches the mask, everything within the directory also matches.
- It preserves the subdirectory path structure of files being copied. (See Example 1 under Backing Up or Example 1 under Restoring in the next sections.)

The N qualifier is almost the reverse of D. N prevents directories from matching and causes subdirectory structures to be "unpreserved" when used with the D qualifier. Since D is sometimes forced, you can use N to help nullify its effects. (See Example 3 under Backing Up or Example 3 under Restoring in the next sections.)

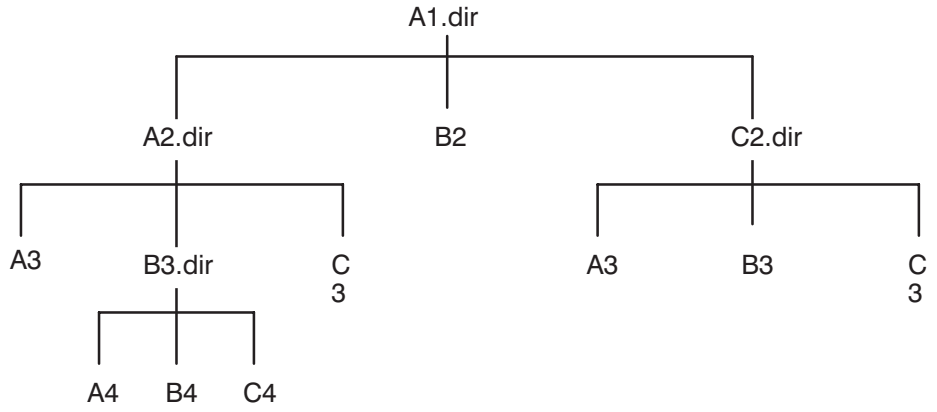
The S qualifier searches down through the entire directory structure. (See Example 3 under Backing Up in the next section.)

The K qualifier searches down through the entire directory structure and preserves the subdirectory path. (See Example 2 under Backing Up or Example 2 under Restoring in the next sections.)

Backing Up

When you select files to save, use the same rules as the CI CO (Copy) command. Files are matched by name, type extension, or any other characteristic given in the mask, and the D qualifier is forced into the mask. File names and paths can be modified by entering a destination mask with the BA command. Qualifiers on the destination mask are ignored.

Following is a directory tree structure and three examples of file selections for backup. The LI command shows each selected file and how it was renamed. Assume the working directory is A1.dir.



Example 1: Back up the files that match the C@ mask.

Note that the files in subdirectory C2.dir are selected even though they do not match C@, because the mask is expanded to C@.@.D and subdirectory structure is preserved under /Z/.

```
FST> ba c@ /z/
4 files selected; 4 total
FST> li

C2.DIR:::2:64:32 to /Z/C2.DIR:::2:64:32
C2/A3:::3:24 to /Z/C2/A3:::3:24
C2/B3:::3:24 to /Z/C2/B3:::3:24
C2/C3:::3:24 to /Z/C2/C3:::3:24

FST>
```

Example 2: Back up the files that match the A@.@.k mask.

Note that the path structure is preserved, every lower path is searched, and the effect of the D qualifier is overridden. That is, files do not match just because their directories do.

```
FST> ba a@.@.k /z/
4 files selected, 4 total
FST> li

A2.DIR:::2:64:32 to /Z/A2.DIR:::2:64:32
A2/A3:::3:24 to /Z/A2/A3:::3:24
A2/B3/A4:::3:24 to /Z/A2/B3/A4:::3:24
C2/A3:::3:24 to /Z/C2/A3:::3:24

FST>
```

Example 3: Back up all the non-directory B files using the masks B@.@.ns.

Note that N causes no directories to match nor subdirectory structures to be preserved, and S causes the search to examine every path.

```
FST> ba b@.@.ns /z/
    3 files selected; 3 total
FST> li

    B2:::3:24 to B2::Z:3:24
    A2/B3/B4:::3:24 to B4::Z:3:24
    C2/B3:::3:24 to B3::Z:3:24

FST>
```

Restoring

When you select files to restore, you follow most of the same rules that apply to the CI CO command. Files are matched by name, type extension, or any other characteristic given in the mask, and the D qualifier is forced into the mask. You can match file names and paths by entering a destination mask with the RE command. Qualifiers on the destination mask are ignored.

The “mask” parameter in the RE command can also be a disk LU number that selects the files that were backed up from that disk LU. In this case, unless the “dest_mask” parameter is specified as a disk LU number in the RE command, FST tries to restore the file to its original LU. The LU specified in the “dest_mask” parameter is used by FST when FST needs to create a global directory. If the required global directory already exists, FST uses the existing directory.

The differences between restore masking and standard CI masking are as follows:

- Only the D, K, N, and S qualifiers are usable; all other qualifiers are ignored.
- A mask of @, or equivalent, matches every file on the archive, regardless of its path. If you do not enter a mask, the default is @.
- If you do not enter a directory structure with the mask, all paths match and only the file names, type extensions, and other file characteristics are checked for a match (see Example 1 below).
- Global directories, like subdirectories, are preserved as subdirectories if you enter a destination mask (see Example 1 below).

Examples 2, 3, and 4 below provide explanations for the most commonly asked questions about masking, including restoring files to the working directory with path preservation, without path preservation, and with partial path preservation.

Below is a listing of a directory tree structure, backed up to the tape, followed by several examples of file selections for restoring. The LI command displays each file selected and shows how it was renamed. Note that this tree and its files are identical to those shown in the previous section.

```
/A.DIR:::2:64:32
/A1/A2.DIR:::2:64:32
B2::A1:3:24
/A1/C2.DIR:::2:64:32
/A1/A2/A3:::3:24
/A1/A2/B3.DIR:::2:64:32
/A1/A2/C3:::3:24
/A1/A2/B3/A4:::3:24
/A1/A2/B3/B4:::3:24
/A1/A2/B3/C4:::3:24
/A1/C2/A3:::3:24
/A1/C2/B3:::3:24
/A1/C2/C3:::3:24
```

Example 1: Restore the files that match the C@ mask.

Note that the files in subdirectory C2.dir are selected even though they do not match C@, because the mask expands to C@.@.D, and the directory structure is preserved under /Z/.

```
FST> re c@ /z/
6 files selected; 6 total
FST> li

/A1/C2.DIR:::2:64:32 to /Z/A1/C2.DIR:::2:64:32
/A1/A2/C3:::3:24 to /Z/A1/A2/C3:::3:24
/A1/A2/B3/C4:::3:24 to /Z/A1/A2/B3/C4:::3:24
/A1/C2/A3:::3:24 to /Z/A1/C2/A3:::3:24
/A1/C2/B3:::3:24 to /Z/A1/C2/B3:::3:24
/A1/C2/C3:::3:24 to /Z/A1/C2/C3:3:24

FST>
```

Example 2: Restore the files matching the A@.@.k mask, and place them in your working directory.

Note that the path structure is preserved, and the effect of the D qualifier is overridden (That is, files do not match just because their directories do.).

```
FST> re a@.@.k  @
5 files selected; 5 total
FST> li

/A1.DIR:::2:64:32 to A1.DIR :::2:64:32
/A1/A2.DIR:::2:64:32 to A1/A2.DIR:::2:64:32
/A1/A2/A3:::3:24 to A1/A2/A3:::3:24
/A1/A2/B3/A4:::3:24 to A1/A2/B3/A4:::3:24
/A1/C2/A3:::3:24 to A1/C2/A3:::3:24

FST>
```

Example 3: Restore all the non-directory B files using the mask B@.@.ns, and place them into the working directory, flattening the directory structure.

```
FST> re b@.@.ns @
3 files selected; 3 total
FST> li

B2::A1:3:24 to B2:::3:24
/A1/A2/B3/B4:::3:24 to B4:::3:24
/A1/C2/B3:::3:24 to B3:::3:24

FST>
```

Example 4: Restore the files and directory structure under the /A1/A2.dir directory to the working directory.

Note that the A1/A2 directory path is not preserved in the destination.

```
FST> re /a1/a2/@ @
6 files selected; 6 total
FST> li

/A1/A2/A3:::3:24 to A3:::3:24
/A1/A2/B3.DIR:::2:64:32 to B3.DIR:::2:64:32
/A1/A2/C3:::3:24 to C3:::3:24
/A1/A2/B3/A4:::3:24 to B3/A4:::3:24
/A1/A2/B3/B4:::3:24 to B3/B4:::3:24
/A1/A2/B3/C4:::3:24 to B3/C4:::3:24

FST>
```

Incremental Backup

Incremental backup is a procedure that involves periodically backing up all the files and doing frequent backups only of files that were changed since the previous backup. The backup bits in a file's directory entry are used for this, as explained later, so FMGR files do not apply. The initial backup is called a "full backup," and subsequent, selective backups are called "delta backups."

For example, on Friday night, do a full backup of a particular directory. On Monday, Tuesday, Wednesday, and Thursday nights, take delta backups (and append to the same tape that contains the full backup).

Delta backups are done through the use of the backup bit in all hierarchical file directory entries. If the backup bit is set, the file was not backed up; if it is clear, the file was backed up. Whenever a file is created or modified, the backup bit for that file is set. The B qualifier can be used in a mask to select just those files in the working directory whose backup bits are set.

Backup is performed as follows:

1. Use FST to back up all the files, specifying the Clear option to clear all the backup bits of the files that get saved. This is the full backup.
2. Select only those files that have their backup bits set (This is specified by the B qualifier in the file mask.), again using the Clear option to clear the backup bits. This is the delta backup.

For example:

Saturday:

```
FST> ba /important_data/@
1000 files selected; 1000 total
FST> ti IMPORTANT DATA - FULL BACKUP
FST> C
Clear ON
Verify ON
FST> go
```

Monday:

```
FST> ba /important_data/@.@.b
90 files selected; 90 total
FST> ti IMPORTANT DATA - MONDAY DELTA
FST> c a (the 'a' is optional)
Append ON
Clear ON
Verify ON
FST> go
```

Tuesday:

```
FST> ba /important_data/@/@.b
120 files selected; 120 total
FST> ti IMPORTANT DATA - TUESDAY DELTA
FST> c a (the 'a' is optional)
Append ON
Clear ON
Verify ON
FST> go
```

Wednesday: etc.

Thursday: etc.

Restoring from Incremental Backups

The SrchApp option can be used to restore selected files from an incremental backup. The SrchApp option automatically moves through each append and searches for files to restore.

Using the previous incremental backup example:

```
FST> s
SrchApp ON
Initialize Group restore
Update ON
FST> v
Verify ON
FST> re @.golf
FST> re four@
FST> re @.day
FST> go
1 files selected; 1 total

     Tape format:FST
     Title      :IMPORTANT DATA - FULL BACKUP
     Created    :Sat Jun 24, 1989   1:42:14 pm

Copying /IMPORTANT_DATA/NO_GOLF_2.DAY:::4:2:36

Verifying archive

            1 files selected
            1 files restored
            1 files successfully verified.
Positioned at append #1
2 files selected; 2 total

     Tape format:FST
     Title      :IMPORTANT DATA - MONDAY DELTA
     Created    :Mon Jun 26, 1989   12:57:14 pm

Copying /IMPORTANT_DATA /TROYS_SCORES.GOLF:::4:4:36
Copying /IMPORTANT_DATA /FOUR_EYES.ONLY:::4:1:8

Verifying archive

            2 files selected
            2 files restored
            2 files successfully verified.
Positioned at append #2
1 files selected; 1 total
```

```
     Tape format:FST
     Title      :IMPORTANT DATA - TUESDAY DELTA
     Created    :TUE Jun 27, 1989   1:01:03 pm
Copying /IMPORTANT_DATA/TROYS_SCORES.GOLF:::4:8:36

Verifying archive

            1 files selected
            1 files restored
            1 files successfully verified.
Positioned at append #0
FST>
```

As an alternative to the SrchApp option, you may also choose to use the positioning commands within FST to select the desired append. The FST commands, NE (Next), PR (Previous), and PO (Position), allow tape movement from one append to another.

The Update option only restores a file on the tape if it is newer than the file on the disk. You can use a command file and the Update option to devise a general method for restoring just the latest copy of a file.

Using the incremental backup example shown in the previous section, the following general command file example restores a particular file from an incremental backup tape:

```
* Position to the last append (Thursday's)
*
po 4
* Turn on the update and verify options.
Update
Verify
* restore the file (if it's there)
re /important_data/vacation
go
*
* Position to Wednesday's append
*
po 3
re /important_data/vacation
go
*
* Position to Tuesday's append
*
po 2
re /important_data/vacation
go
.
.
.
```

You may also locate the file manually or with a command file, and then position to the correct append and restore the file. The following example shows a command file that is used to find the location of the desired file:

```
d1 /important_data/vacation
ne
d1 /important_data/vacation
ne
d1 /important_data/vacation
ne
d1 /important_data/vacation
ne
d1 /important_data/vacation
```

When you run this command file, you can see which append has the latest copy of the file, position to that append, and restore the file.

Appending Data

Tape appends are individual backups on the same tape. Each append is separated by tape marks on the tape and has its own tape header, comment file, and directory listing. Each append is independent of the others. You may use the Append option to specify a backup as an append to a tape. The NE, PR, and PO commands are used to examine the individual appends on a tape. The maximum number of appends allowed on a single tape is 1023. Appends are not supported when backing up to an archive file.

Consecutive Backups

The BA command appends to the list of already selected files, regardless of the GO command. GO does not clear the selection of files for BA. If you do not want previously selected files for a backup, you must use the UN command to unselect the files. A common mistake is to select one set of files, copy them to tape using GO, then select a second set without exiting FST, and copy them to tape using GO. Thus, the first set of files are included with the second set on the second copy, because the first set was never unselected.

Multiple Reels

FST supports multiple reels, but handles them differently than other logical backup utilities. When FST backs up files, it splits them across tape boundaries when it reaches the end of a tape. This lets FST back up single files that are too large for one reel.

To restore a single file from multiple reels requires only the reel or reels on which the file is contained. You need not start with the first reel. For example, if you know a file exists on reel 9 of a backup, you can mount the ninth reel and restore the file without mounting tapes 1 through 8.

The tape format and DL command help determine where a file exists among multiple tapes. The comment file and directory file are at the front of each reel in a multiple reel backup. When the directory file is written to the second reel, the directory is updated, specifying each file that was

written to the first reel. Directory updating is also done on all succeeding tapes. This lets you do a DL command on the last tape of a backup to locate a file on any tape in the multiple reel backup.

The Keep option determines the state of the mounted tape when its end is reached during a multiple reel backup. If Keep is ON, the tape remains online/loaded after it is filled. You'll see a message telling you to "Enter 'GO' when a new tape is online/loaded." (Online refers to 1/2 inch magnetic tapes. Loaded refers to CTD tapes.)

If Keep is OFF, the tape is taken offline/unloaded after being filled. You'll see a message telling you that "FST will continue when the tape is ready (online/loaded)."

You can use the Keep option in several ways. Use Keep OFF when loading and unloading on the HP 35401A Autochanger. Tapes are unloaded when full, and FST continues when the next tape is loaded. If your terminal is not near the tape drives, you can issue GO from the terminal before the tape is online/loaded. If you have set the Yes option ON, you can then walk to the tape drive, ready the tape, and the backup will begin. You need not return to your terminal to issue another command.

Tape Loading

If FST is ready to begin the backup or restore operation but the tape unit is not (for example, if the tape unit is offline or the tape is not loaded), FST displays one of two messages, depending upon the state of the Keep option, as follows:

- When the Keep option is OFF, the message "Will continue when tape becomes ready" is displayed. FST begins the backup or restore operation when the tape is loaded correctly.
- When the Keep option is ON, a message that gives the state of the tape unit and tells you to "Type 'GO' when ready or 'BR' to terminate" is displayed. FST begins the backup or restore operation when you enter one of the commands.

TF Compatibility

FST can restore from TF formatted tapes with FST command functionality; however, FST has less information about TF formatted tapes than about FST tapes. When FST restores files from an FST tape, it uses the directory file on the tape to restore the files. Since a TF formatted tape does not have a directory file, FST cannot determine the files on the tape as easily. Therefore, when FST restores from a TF formatted tape, it stores the file names or masks entered with each RE command into a directory file, which it then uses to search the tape and restore the selected files to disk.

Because FST uses a larger buffer and a faster process than TF, it restores from TF formatted tapes faster than TF, even though FST does not stream when it restores from TF tapes. FST creates a partial directory file of the TF tape during the restore pass, so that tape positioning during the verify pass (on a selective restore) is also faster.

The following example illustrates an FST restore operation from a TF formatted tape:

```
CI> fst
FST> re @.ftn

FST>re makefile


FST> go

          Tape format: TF
          Title: TapeTitle
          Created : Mon Feb 28, 1986 9:40:00 am

     Copying file1.ftn
     Copying file2.ftn
     Copying makefile

FST>
```

TAR Compatibility

FST can read and write archives in TAR format; however, due to the differences between the TAR and FST formats, FST functions in a slightly different manner. Like the TF format, the TAR format does not include a directory file, so FST therefore cannot immediately determine the files in an archive. For this reason, when FST restores from a TAR archive, the results of the RE command are not reported after each command. Instead, the file names or masks entered with the RE command are saved and used when the tape is searched for the selected files to be restored.

UNIX Compatibility

Generic RTE/UNIX file system differences are discussed in the UNIX compatibility section in the TF section of this manual.

FST cannot restore ASCII files with record lengths greater than 1024 words (2048 bytes) from TAR archives. Records longer than 1024 words will be split into multiple records with a warning issued for each file containing the long records. When creating TAR archives with FST, there is no restriction on the record lengths for type 4 files; however, when a record longer than 1024 words is encountered, a warning is issued stating that the record will be split upon restoration by FST. Note that the record structure of the file is still intact and TAR will be able to restore the file without splitting the record.

Files on TAR archives whose names would be illegal RTE file names are renamed by FST. The output of a DL command of a tar archive displays the original file name and the new name created by FST. File names containing reserved characters will have the reserved characters replaced according to the following table.

Table 4-4 . Reserved Character Replacements for TAR Archive Files

Reserved Character	Replaced By
‘ ’	—
.	—
,	—
@	*
–	?
:	|
>	^
[	(

Note that not all periods will be replaced. If the last period in a name will result in a legal RTE type extension, it will be preserved. File names that begin with numbers will have an underscore character (‘_’) affixed to the front of the file name.

To select a renamed file with the RE command, the new file name should be used as the basis for the mask in the RE command. The FST ‘DL’ command will display the original and new names for any file that will be renamed upon restoration.

Rescuing Files from an Overwritten Tape

It is not uncommon to accidentally overwrite a tape. If the overwritten area is smaller than the original backup, you may use the Faulty option, described earlier in this chapter, to restore the data beyond the overwritten area.

Since overwriting a tape destroys the directory file, FST must try to build a new one from the file headers that can still be found. Once a new directory file is built, the operations associated with a normal restore can proceed.

FST assumes that the overwrite ended with at least one EOF mark. After you set the Faulty option ON, position the tape just past the EOF mark that immediately precedes the point where you want FST to start looking for the data that was not overwritten. Then issue the RE command, with a mask if desired. FST builds the new directory file, displays messages about the status of the process, and provides needed information in case the process fails.

- The following example shows how to restore files that still remain from an FST backup that was partially overwritten by a TF backup on a magnetic tape.

```
FST> mt 8
FST> f v
Faulty ON
Verify ON
FST> ne 2
Positioned at append #2
FST> re
FAULTY option ON:  Assuming tape being restored is partially corrupted.
Searching for a legal tape record
(sometimes some tape errors are seen here because of parity)
Searching for a valid file header
Scanning tape and building directory file
3 files selected; 3 total
FST> li

f:::4:29:36
g:::3:24:59
h:::4:3:10

FST> go
Copying f:::4:29:36
Copying g:::3:24:59
Copying h:::4:3:10

Verifying tape

FST>
```

It may take a few attempts for you to find the correct file position on the tape, depending upon how many EOF marks actually precede the data you want to restore. If FST reports finding an EOF mark but the process stops, either you did not position the tape to the proper place, or FST did not find any file headers before reaching the next EOF mark.

Note This process involves unusual tape positioning. Do not try to use the DL command when you restore from overwritten tapes. DL tries to read the directory file from the tape, which may disturb the current positioning.

If the original backup required multiple tapes, the file that crossed the tape boundary cannot be fully restored.

Disk Directory File

When FST backs up to or restores from an archive, it creates a directory file, which contains the names of the files specified in the BA or RE commands along with information about each file that FST needs to perform the backup or restore. Although you cannot display the directory file itself, you may use the LI command to display a list of the files in it.

The size of the directory file on the disk LU limits the size of the backup/restore. You may use the DF command to specify the name and size of the directory file and place the directory file on an LU that is large enough for the current operation. If you do not use DF, FST creates its own directory file when it becomes necessary.

Each time you use the BA command, the contents of the directory file increase. In a restore, because FST uses the directory file on the archive, the entire directory file is copied to disk; therefore, you must specify all the disk space needed for the restore with the first RE command. The directory file on disk is purged when you exit FST.

You cannot move a directory file after it is created. Therefore, use DF before you use BA or RE if the default file descriptor is not adequate. Refer to the earlier discussion of the DF command for directory file default information.

Shareable EMA

FST uses shareable EMA (SHEMA) for its tape buffering. Internally, FST uses between two and five 25-page buffers. The more pages supplied, the faster the speed at which FST backs up files and maintains streaming. However, FST does not use more than five 25-page buffers (125 pages of SHEMA) for tape buffering during streaming.

SHEMA is also used to buffer FMGR file information temporarily when BA is used. If the current SHEMA is not large enough for an FMGR backup selection, more SHEMA is required. Because FST uses SHEMA, you cannot run two identical copies of FST at the same time. You must specify another SHEMA partition for each copy of FST. Refer to the section “Installing FST” later in this chapter.

FST Format

Each archive that FST creates contains an archive header, followed by the comment file (if one was selected), and a complete directory of the backup. The individually saved files follow, each one preceded by its header.

Each tape of a multiple tape backup has the comment file and the entire directory file at its head. Each directory maintains the tape number for each file saved on previous tapes. This lets you look at the last tape to determine which tape contains which file.

The format for FST contains all the needed extent information for files. This lets you restore files to the exact needed size, leaving no wasted blocks on the disk, or restore files to their original layout with all the extents or extra space present.

Type 1 and type 2 files are always restored to their original disk format. FMP cannot detect wasted blocks for these types of files or manipulate their extents. For the hierarchical (CI) file system, information such as time stamps and access rights are restored as required.

Replacing Reserved Characters

FST replaces reserved characters in FMGR file names with non-reserved characters, and sends a message to the terminal or log device/file that the file was renamed. The reserved characters and their replacements are as follows:

Table 4-5 . Reserved Character Replacements for FMGR Files

Reserved Character	Replaced by
.	*
/	!
@	?
[	(
>	^

The following exceptions apply when FST replaces reserved characters:

- A period (.) in the middle of a file name is not replaced.
- If there are multiple periods (...) in the middle of a file name, all but the first period in the group are replaced by asterisks.
- If "[" or ">" is the first character in a file name, it is not replaced.

Recommended System Usage

FST uses a directory file to handle the list of all files to back up or restore. Although this speeds up operations (such as doing a DL of the tape or selectively restoring files), the directory file does take up space on the disk and on the tape. Also, the larger the size of the directory file, the longer some operations take to complete.

To avoid slowing down operations and running out of space on the disk LU on which the directory file is located (or to which it is being restored), follow these guidelines when you plan your backup strategy:

- To back up multiple disk LUs that require multiple tapes, divide the operation into saves that fit on a single tape.
- If one disk LU has thousands of files to be saved, save the LU by itself (that is, do not try to back up other LUs with it). Note that the saves can be individual appends on the same tape; therefore, the tape is not being wasted, and you do not need to switch tapes.

Streaming

Streaming is supported on the HP 9144 Tape Drive and the HP 7974/7978 and 7979/7980 Magnetic Tape Drives. Streaming is supported only during backups, not during restores. Restore operations, however, imply extraordinary circumstances, such as a corrupt disk or a system that is down, and occur much less frequently than backup operations.

FST is supported on non-streaming tape drives, but its speed will be much slower than what is available with streaming tape drives. However, FST performs backups and restores on non-streaming tape drives faster than other backup utilities.

Multiple buffers in SHEMA help provide the streaming capabilities during FST backups. Note that files that were written to the buffer may not be copied to the tape before the end of the tape is reached (even if the "Copying ..." message for such a file is displayed or written to the log/device file). In this case, the files are copied to the next tape and the "Copying ..." message for those files is displayed again. Thus, occasionally a file may appear to have been copied to the end of the first tape and to the beginning of the second tape, when actually it was copied only to the second tape.

Streaming is affected by your disk organization. During a backup, if FST accesses many small, scattered files, or files with many scattered extents, continuous streaming is less likely. The larger the files and the fewer extents, the better streaming is maintained.

Streaming is not supported when you back up files in TAR format. Since the process for obtaining the data to be stored to tape is much slower, streaming should not be expected.

FST Format

Each FST archive is written as a single file with 10-Kbyte records. Two file marks are written to mark the end of data on the archive. FST appends are separated by file marks on a tape.

The first 512 bytes of each FST archive is an archive header. An optional comment file can follow the first header. After the first header (and optional comment file) the format of the archive consists entirely of header/data pairs. Each header is 512 bytes and contains information describing the data that follows. The header contains all of the directory information for the file whose data follows it.

The first header/data pair describes the FST directory file. This file contains the headers for all of the files that are included in the archive. When restoring files, FST temporarily restores the directory file as a type 2 file with a 512 byte record length. Following the directory file header/data pair are the header/data pairs for each of the files in the archive. (The header for any file on the archive is saved twice, once in the directory file and again preceding the file's data.) No data is saved when a directory is archived, only a file header indicating the directory's attributes is saved.

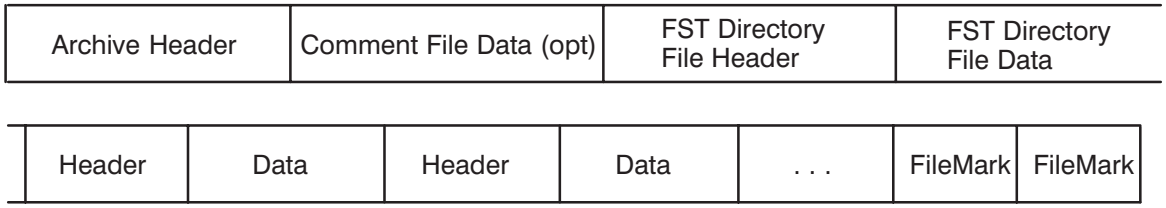
Each tape of a multiple tape backup has the comment file and the entire directory file at its head. Each directory maintains the tape number for each file saved on previous tapes. This lets you look at the last tape to determine which tape contains which file.

The format for FST contains all the needed extent information for files. This lets you restore files to the exact needed size, leaving no wasted blocks on the disk, or restore files to their original layout with all the extents or extra space present.

Type 1 and type 2 files are always restored to their original disk format. FMP cannot detect wasted blocks for these types of files or manipulate their extents. For the hierarchical (CI) file system, information such as time stamps and access rights are restored as required.

The FST format and basic header contents are shown in Figure 4-1.

FST Format:



FST Archive Header Format (in bytes):

001-100	Archive file
149-156	Checksum of archive header (Octal ASCII)
159-222	Revision and time stamp of the version of FST used to create the archive
299-302	Comment file size in blocks (32-bit integer)
381-382	Tape number (16-bit integer)
383-386	Record pointer into the FST directory file to the header for the first file contained in the archive (32-bit integer)
387-390	Starting block number for the data of a file split across a tape boundary (32-bit integer)
391-392	Flag indicating that the first file on this archive was split across a tape boundary (16-bit FORTRAN logical)
418-423	Header type (“FST”)
448-459	Archive create time specified in the number of seconds since 12 AM January 1, 1970 (Octal ASCII)

FST File Header Format (in bytes):

001-100	File descriptor
101-108	File protection bits (file mode in Octal ASCII)
125-136	Size in bytes (Octal ASCII)
149-156	Checksum of header (Octal ASCII)
295-296	Source disk LU (16-bit integer)
297-298	Sectors/track on source disk LU (16-bit integer)
299-302	File size in blocks (32-bit integer)
303-304	Hierarchical file flag (16-bit FORTRAN logical)
305-368	Directory entry (32 element array of 16-bit integers)
369-370	Extent number (16-bit integer)
371-372	More extents flag (16-bit FORTRAN logical)
373-376	Block offset past the archive header to the data for the file (32-bit integer) (1 block = 256 bytes)
418-423	Header type (“FST”)
468-500	Owner’s name (directories only)

Figure 4-1 . FST Format and Header Basics

Installing FST

Two programs, FST and FSTP, are used to facilitate file backup and restore. FST has primary control of all the commands; FSTP handles I/O to and from the archive. For example, when doing a tape backup, FST fills the tape buffers from the disk, while FSTP copies the buffers to tape. In tape backups, FST and FSTP have just enough priority to maintain streaming without preventing other processes from functioning.

To install FST, first link FST and FSTP using the LINK command files #FST and #FSTP. The SHEMA label in #FST must be a legal SHEMA partition label, and the SHEMA size should be set as large as possible to enhance streaming. Up to a maximum of 125 pages of SHEMA is used for streaming.

Use the EM command in the LINK program to specify the SHEMA size. Only #FST must be changed to specify the proper SHEMA labels and size. Place both run files on directory /PROGRAMS or on an FMGR cartridge.

The file >FS000 must be located on the global directory /CATALOGS. If /CATALOGS does not exist, >FS000 can be placed on an FMGR system cartridge.

Because FST uses SHEMA, only one unique FST program can use a particular SHEMA partition at a time. All other FST programs that attempt to access the same SHEMA partition abort with a value of −2 in the CI variable $RETURN1, and the following message is displayed:

```
My SHEMA partition is already in use.
```

For additional copies of FST, relink each new copy of FST.RUN, specifying a different SHEMA label for each copy. You can set up a transfer file to use the −2 value in $RETURN1 to select an unused copy of FST automatically.

FST Error Handling

Tape and disk accesses are monitored so that errors can be captured without aborting FST. When you run FST interactively, most errors cause a return to the FST prompt.

The directory file and all specified options remain intact if an error occurs. When you run FST programmatically, errors return a value of −1 in the CI variable $RETURN1. If the necessary SHEMA partition is already in use, FST aborts with a value of −2 in $RETURN1.

If an error causes FST to abort a backup or restore operation and return to the FST> prompt, directory information is not lost. You can reenter the GO command to restart the operation without reselecting the files. We recommend that you use log files for all backups and restores. Errors may occur even when all the specified files are copied.

FST Error Messages and Warnings

The following error messages and warnings may be displayed when you perform a backup or restore using FST:

Aborting FST

FST is being terminated because of a previously reported error.

Appending is not allowed when creating UNIX TAR tapes

The append option was specified for creating a TAR tape, but appends can be done only to FST tapes.

Appends are only allowed on FST tapes

The tape must be in FST format to append to it.

Appends are not allowed with archive files

Appends can only be made to FST backups on tape.

Archive file is corrupt: <filename>

While reading the archive file, FST encountered an FMP −12 error and could not continue.

Archive file is not type 1: <filename>

Archive files must be type 1 files. TAR archive files transferred from UNIX machines via FTP must be transferred as binary files.

Break command: process aborted

The current process was aborted because the break flag was set.

Cannot access the log file/device: <filename or LU>

The selected log file/device is not available for use.

Cannot append to the log file: <filename>

The append positioning for the log file was unsuccessful.

Cannot backup: <filename>

A selected file could not be saved to tape successfully.

Cannot backup sparse files across DS: <filename>

You cannot back up sparse files (files with missing extents) across a DS link.

Cannot call a transfer file from a transfer file

The current transfer file contains the FST command to access another transfer file. Transfer files cannot be nested.

Cannot clear backup bit: <filename>

FST could not clear the backup bit of the file backed up or restored.

Cannot create group scratch file. Unable to set SrchApp option

The group scratch could not be created. Since the SrchApp option requires a scratch file, it could not be set.

Cannot determine the density

The current tape density cannot be determined with the SD command.

Cannot find the directory file on the tape

The directory file could not be located on the tape during tape backup verification.

Cannot find this tape header: <filename>

Incorrect positioning of the tape occurred while searching for a file header. Note that if this error occurs, there is a potential problem with FST.

Cannot lock the tape LU

FST was not able to lock the tape LU.

Cannot create the directory file

The directory file used for file selection could not be created.

Cannot open the archive file: <filename>

The archive file specified in the MT command could not be opened.

Cannot open the selected comment file

FST is not able to open and use the selected comment file.

Cannot open transfer file

FST could not open the specified transfer file.

Cannot position to beginning of data

A rewind to the beginning of an append on a magnetic tape failed. FST did not return to the beginning of the tape append.

Cannot purge: <filename>

FST could not purge the file that was backed up.

Cannot purge the old <filename> in order to rename its replacement

When FST restores a duplicate file to disk, it first copies the file from tape to a scratch file on disk and gives it a temporary name. If that copy operation is successful, FST purges the original file on disk and renames the scratch file with the original file name. If FST cannot purge the original file, it cannot rename the scratch file with that file name. In other words, the original file on disk cannot be updated until the old contents can be purged.

Cannot restore: <filename>

FST could not restore a selected file from tape successfully.

Cannot restore <filename> from this tape. It is on tape <tape #> (tape <tape #> is mounted)

The mounted tape belongs to a multi-tape backup, and the selected file exists on a previous tape.

Cannot restore linked files: <filename> linked to <filename>

FST does not restore UNIX hard links or symbolic links from TAR archives.

Cannot restore UNIX device files: <filename>

FST does not restore UNIX device files from TAR archives.

Cannot select file for backup: <filename>

The specified file could not be selected for backup.

Cannot set TAR format: the most recent file selection is from a non-TAR archive. Unselect all files and load a TAR archive for TAR file restoring.

The TAR option cannot be turned on when the currently selected files are from a non-TAR archive.

Cannot successfully rename the restored file <scratch file> to <filename>

When FST restores a duplicate file to disk, it copies the file from tape to a scratch file on disk and gives it a temporary name. If the copy from tape to the scratch file is successful, FST purges the original file on disk and renames the scratch file with the original file name.

Cannot turn OFF TAR format: the most recent file selection is from a TAR archive. Unselect all masks and load the correct archive for non-TAR use.

The TAR option cannot be turned OFF when the currently selected files are from a TAR archive.

Class I/O between FST and FSTP confused. Current instruction aborted

There is miscommunication between the father program, FST, and the son program, FSTP, during Class I/O. The current instruction is aborted. If this error occurs, there is a potential problem with FST.

Corrupt comment file on disk

FST cannot obtain the necessary disk information to copy the comment file to the tape.

Corrupt comment file on tape

Corrupt records in the comment file were detected during a listing of the comment file.

Corrupt file. Cannot select: <filename>

The file selected for backup is corrupt.

CTD tape is not initialized

The loaded cartridge tape must be formatted before it can be used.

Directory file failed verify

The directory file failed the verify pass of the backup. This could mean that the directory file became corrupt during the backup operation.

Directory file is corrupt. File selection lost and directory file being purged

The directory file is assumed to be corrupt because the positioning within the directory file failed. The directory file is purged and all previous file selection is lost.

Disk error: Unable to load FST segment

FST was not able to load the needed segment of code.

EMA full. Need to process REstore commands with a GO before continuing.

The number of TF or TAR Restore commands exceeded the EMA limit. Execute the current Restore commands by issuing a GO command; then enter the additional RE commands.

Encountered EOF. Search quitting.

An EOF mark was encountered before any legal file header during an attempt to build a new directory file with the Faulty option.

Erasing the current backup from the tape

The backup in progress is corrupt. A filemark is placed on the tape where the backup began to show that the data following is not valid. The next backup to that tape will begin at that filemark, and the corrupt data will be overwritten.

ERROR: Grouping was never begun.

GR was not entered to begin grouping a set of RE commands, but another grouping command (for example, EG or AG) was entered. You must enter GR before entering other group commands.

Error reading transfer file

FST could not read the next command from the command file successfully.

Error scheduling FSTP:

An FMP error occurred while trying to RP FSTP.

Error scheduling <FSTP rp'd name>: ID segment gone

FSTP was terminated during initialization.

Error scheduling <FSTP rp'd name>: <xxxx> violation

FSTP could not be scheduled successfully.

Error using group scratch file. Cannot process SrchApp option.

The group scratch could not be created. Since the SrchApp option requires a scratch file, it could not be set.

Extent header missing from archive

FST was expecting the next header on the archive to be for a particular extent, but it was not.

FC tape format: unhandled by FST

The mounted tape is in FC format, which FST does not handle.

File cannot be selected for TAR text conversion: <filename>

The specified file is either inaccessible or corrupt for a TAR ASCII backup.

File failed verify: <filename>

The file on the tape failed to verify (compare identically) with the same file on the disk.

File restored, but not with proper TEXT data: <filename>

The restored TAR file was found to be corrupt on the tape. It was probably a binary file and should not have been restored in ASCII format.

FMP ERROR: <error message>

A report of an error returned from FMP.

FMP ERROR: <error message> – <filename>

A report of an error returned from FMP with the related file name.

Grouping not allowed for non-FST archives

You entered GR to group a set of RE commands when restoring a non-FST archive. Grouping is allowed only for FST archives.

Illegal density

An invalid density was specified with the SD command.

Incorrect usage of command

Incorrect parameters were supplied for an FST command.

Insufficient free space available, size up FST

Free space in the program used for internal buffering is not large enough.

LU <#> is already locked

A mask, with the lock option ON, specified a disk LU that is already locked.

Multiple failures. Search quitting.

While attempting to recover data from an overwritten tape, a good record could not be found.

Need at least 50 pages of EMA to run

Not enough shareable EMA space was linked with FST for proper execution.

No comment file exists on this archive

No comment file exists; therefore, none can be listed with the LC command.

No files selected yet

No files can be backed up, restored, or listed because no successful file selection was completed.

No RNs available: Cannot lock LU <#>

No resource numbers are available to lock the disk LU.

No such archive file: <filename>

The archive file specified in the MT command could not be found.

No tape LU has been selected

A command requiring a specified tape LU was entered, but a legal tape LU was not selected.

Non-FST append found on tape: appending cannot continue

An append to a tape with non-FST data was attempted. FST appends are only allowed on tapes that have all their appends in FST format.

Not a legal tape unit

The MT command was used to specify an illegal tape LU.

Not enough disk space for the archive file: <filename>

The archive file specified in the MT command could not be created in the specified directory. Locate the archive file on a disk with more free space.

Not enough room on this tape to hold the backup. Try another tape.

The mounted tape is too short to hold the current backup directory file and/or comment file.

Not updating: <filename>

The specified file is not being restored because it is older than the disk copy and the Update option is ON.

Option or ON/OFF expected: <unrecognized command>

An illegal word was supplied in an option setting command string.

Owner not set for <directory name>

The original ownership could not be successfully restored to the specified directory.

Protection not set for <filename>

The original read/write protection could not be successfully restored for the specified file.

Setting tape density unsuccessful

The density of the tape unit could not successfully be set to the specified value.

SrchApp mode is available only for FST tapes

The SrchApp mode can be used only when restoring FST tapes.

Tape channel error

The CTD returned a channel error.

TAPE ERROR: <error code>

A tape instruction returned on the no-abort/no-suspend path with the error code in the A- and B-Registers.

Tape FAULT error

The CTD returned a fault error.

Tape headers do not match

The wrong tape is mounted for a multiple tape restore.

Tape is not online

The mounted tape is offline.

Tape LU is down

The tape LU is down.

Tape not ready

The cartridge tape drive unit is not ready for tape access (for example, the tape may be unloaded or positioned to the load point).

Tape sequence wrong: Tape <#> loaded, tape <#> expected

The sequence of tapes mounted during a multiple tape restore operation is incorrect. The mounted tape belongs to the current multiple tape backup, but is out of sequence.

Tape status error

The error bit was set in the A-Register when returning from a tape request call.

Tape write-protected

Files cannot be copied to a write-protected tape.

TAR and non-TAR formats cannot be mixed for backup. Current files must be unselected before TAR format can be used.

The TAR option was turned ON for backup after files were already selected without the TAR option. The formats cannot be mixed.

TAR and non-TAR formats cannot be mixed for backup. Current TAR files selected must be unselected before TAR format can be turned OFF.

An attempt was made to turn the TAR option OFF after files were already selected with the TAR option. The formats cannot be mixed.

TAR selection was used, but the archive is not in TAR format.

The TAR option was turned ON, but the archive is not in TAR format.

The current append is not an FST backup

The tape has been positioned to an append that is not in FST format.

The directory file is already open: cannot assign a new one.

The DF command cannot be used once the directory file has been opened. Note that the UN command can be used to purge the current directory file, but all selected file information must be reselected.

This is not the original tape on which the restore was initiated

A different tape was mounted since the restore operation began. For a restore, FST reads the directory file on the tape during the file selection process. When the GO command is entered, the original tape must still be mounted for a successful restore.

Too many appends specified, tape positioned at last one

There are no more appends on the tape. The tape is still positioned at the last append.

Too many files on this cartridge; size up your EMA

The Extended Memory Area (EMA) is too small for the current file selection for backup from an FMGR cartridge. The EMA must be increased in order to execute it.

Unknown command

The command you entered was not recognized by the program.

Unknown tape format

The mounted tape has a tape format that FST does not recognize (that is, it is not in FST, TF, or TAR format).

Unrecoverable data tape error

The CTD returned an unrecoverable data error.

Unsuccessful path creation

The path of a selected file did not exist and FST could not successfully create the path during a restore operation. The file was not restored.

Warning: <filename> contains record lengths > 1024 words.
File saved in TAR mode. (FST will split records during restore.)

A file was saved in TAR format archive that has records with lengths greater than 1024 words. The file was successfully saved in TAR mode; however, a warning is issued because if FST restores this file, the long records will need to be split. (If TAR is used to restore this file, the records will still be intact.)

Warning: <filename> contains record lengths >1024 words.
FST had to split records during restore of TAR text file.

While trying to restore a file from a TAR format archive, FST encountered a text file with records greater than 1024 words in length. FST will split records every 1024 words into a new record.

Warning: Illegal FMP name: <illegal filename>
Renamed to : <new filename>

While reading a TAR format archive, FST encountered a file name that would be an illegal file name. FST will rename the file to a legal file name.

Warning: remainder of command line discarded

Only the commands that precede a TR command, and TR itself, are executed.

Warning: Restore selections cleared

The current append position was changed, so all selected files from there are unselected.

***** Warning *** TAR setting is still on from the restoring. Selected files will be backed up with TAR mode.**

The TAR option is still ON after selecting a TAR archive. Backed up files will be in TAR format.

<xxxx> violation when locking LU <#>

An error occurred when trying to lock a disk LU.

Tape Filer (TF)

TF lets you copy files between disk and tape media, either disk-to-tape or tape-to-disk. You can copy the files to/from a 1600-bpi magnetic tape or a CS/80 cartridge tape drive.

TF is similar to the FC utility, but works with all files, whereas FC works only with FMGR files. TF supports a subset of the FC commands, which provide only tape-related operations. TF does not do disk-to-disk copying. (The CI CO command, described in the *RTE-6/VM CI User's Manual,* can be used for copying files on disk.)

You can select options for special copy functions, including verifying the copy, replacing duplicate files, suppressing the display of the file names being copied, and appending files to a previously written TF tape.

When you select verification, files are verified by a direct comparison of the disk and tape data after the files are copied. Even when verification is not selected, tape checksums are generated and checked.

Incremental backup makes backing up large numbers of files less time consuming. With incremental backup, you back up all the files occasionally, and just the files that were modified since the last backup more frequently. The modified files are usually appended to the same tape used on the previous backup.

The TF tape format is compatible with the TAR format, used by the UNIX TAR (Tape Archive) utility. TAR is the standard tape format for HP implementations of UNIX. The terms UNIX TAR and TF TAR refer to the original format as written by the UNIX TAR utility and TF's adaptation of that format, respectively. See the discussion of UNIX compatibility in the "Copy Command Options" section in this chapter for more information.

Although TF writes tapes only in TF TAR tape format, it can read tapes in FC format (with some limitations), UNIX TAR, or TF TAR format. You do not need to specify the tape format in the command.

You may run TF interactively or from a command file. Both modes are discussed in the following sections.

Be sure to read the section on "Installing TF" that appears later in this chapter before running TF for the first time.

Calling TF

When you run TF interactively, commands are entered at the TF: prompt. TF executes each command before prompting for the next one, continuing until you enter the EXIT command. For example:

```
CI> tf
tf: <command>
tf: <command>
.
.
.
tf: ex
CI>
```

You may enter just the first two characters, or up to the whole name of a command; for example, EX, EXI, and EXIT all cause TF to exit. Uppercase and lowercase are acceptable.

When you enter a command in the TF runstring, as:

```
CI> tf <command>
```

TF executes the command and terminates.

To run TF with a command file, use the TR command, as:

```
CI> tf tr <command file>
```

<command file> can be a device LU or a disk file descriptor. TF executes each of the commands in the command file, then terminates.

TF ignores command lines that begin with an asterisk (*); you can use them to include comments in a command file.

The syntax conventions defined in the Preface of this manual apply to TF, except when file descriptors contain DS locations. In those cases, the "[]" and ">" characters are significant in the entry. Refer to the "Copy Examples" sections in this chapter for more information.

TF Commands

TF uses command sets that let you configure the copy operation, specify names for the tape header and comment files, selectively copy files, group copy commands, transfer to and return from command files, and direct listings to a disk file or list device. Table 4-6 summarizes the commands, which are described in the following sections.

Table 4-6. TF Commands Summary

Commands		Description
Information Command		
Help or ?		Provide a summary of commands and command syntaxes
Copy and Related Commands		
COpy Files	filename	Copy files as specified by parameters
DEfault	file_desc	Set default source, destination, and options for subsequent COPY commands
GRoup	commands	Used for grouping more than one copy command
EG End Group		End Group into a single COPY operation
AG Abort Group		Abort Group
TItle	title	Establish title to be used in tape header file
Listing Commands		
LH List Header		List header file from TF tape
LL Set List Device	device/file	Set list device or file for LH and DL or File
DL Directory List	mask	Compile directory list of TF tape
Transfer and Exit Commands		
TRransfer	filename	Transfer to/return from TF command file
EXit		Exit TF
Comment Command		
*		Identifies following string as comment line

Copy (CO)

Purpose: CO and its related grouping and default commands are used for disk-to-tape (backup) or tape-to-disk (restore) copy operations. The terms “backup” and “disk-to-tape copy” are synonymous in this section, as are the terms “restore” and “tape-to-disk copy”.

Syntax: `co <source> <destination> <options>`

The parameters are discussed in the sections that follow.

Description:

When files are copied, extents are always combined, making a larger main file. Unused space after the end-of-file is removed, except when copying FMGR files.

TF cannot be used to back up “sparse” files (files with missing extents), such as the VMA backing store file. Any attempt to back up a sparse file results in FMP error −104.

CO Command Source and Destination Parameters

Source and destination parameters take several specific forms (described below) depending upon the purpose of the command. Examples are provided in the sections on “Copy Examples” later in this chapter.

Normal backup or disk-to-tape copy

A file mask, or multiple masks, selects the files to be backed up. (Refer to the “File Masks” section of the *RTE-6/VM CI User’s Manual* for more information.) When multiple masks are specified, the list must be enclosed in braces {}. The command’s format is:

```
co <mask> <tape> [<options>]
```

or

```
co {<mask1><mask2> ...<maskn>} <tape> [<options>]
```

Note that if more than one mask selects a file, duplicate copies of the file are written to the tape.

Normal restore or tape-to-disk copy

Restore all files from tape. The destination parameter is omitted, which causes files to be restored with the same names and to the same directories from which they were backed up. The command’s format is:

```
co <tape> [,,<options>]
```

Selective restore

Restore only certain files from the tape specified by the mask or series of masks. Again, the destination parameter is omitted. See the section on "Copy Examples" in this chapter. The command format is:

```
co <tape>{<mask>} [,,<options>]
```

or

```
co <tape>{<mask1> <mask2>...} [,,<options>]
```

Note that unlike backup, if more than one mask selects a file on a restore operation, the file is only copied once. If more than one copy command in a group has a mask matching a given file, only the first matching copy command has any effect and determines the destination and options used for that file.

Restore to different directories or files

You can restore files to directories other than those from which they were saved or give files different names or type extensions when they are restored. Use the destination mask to specify the directory, name, type extension, security code (FMGR files only), and file size for the files copied from tape. You may specify a mask or list of masks to select specific files or sets of files from the tape, as in an ordinary selective restore. For example:

```
co <tape> <destination mask> [<options>]
```

or

```
co <tape>{<mask>} <destination mask> [<options>]
```

or

```
co <tape>{<mask1> <mask2>...} <destination mask> [<options>]
```

Special purpose backup

A disk-to-tape copy can require that the files on tape are given different file descriptors than they had on disk. You may use either a single source mask or a list of masks, as in a normal backup command. File descriptors are assigned to the files on tape based on the destination mask. The format is:

```
co <mask> <tape>{<destination mask>} [<options>]
```

or

```
co {<mask1> <mask2>...} <tape>{<destination mask>} [<options>]
```

Note that a list of destination masks is not allowed. (The section on "Grouping Copy Commands" describes how to specify a different destination mask for each source mask.)

CO Command Options

The CO command options parameter lets you specify any one or a combination of the options shown below. Options may be specified in any order. When the A, I, V, or Y option is used with GR, the option applies to the entire group if it is selected for any command in the group. Once specified, the K option remains in effect until TF terminates. All other options apply only to the command for which the option was specified. Table 4-7 summarizes the CO command options.

Table 4-7. TF CO Command Options Summary

Option	Description
Append	Adds files to end of current tape
Brief	Suppresses logging of each file name as it is copied
Clear	Clears the backup bit for each file copied to tape
Duplicate	Replaces existing files with files restored from tape
Ignore	Causes errors in the tape header to be ignored
Keep	Prevents TF from taking the tape unit offline
N (Inhibit)	Stops text file conversion from UNIX to FMP
Update	Replaces duplicate files if the existing file has an earlier update time
Verify	Compares files on disk and tape after they are copied
X (UNIX)	Converts type 4 files to UNIX text file format when files are copied to tape
Yes	Suppresses questions about overwriting and appending to the current tape

Append to Tape (A)

Append adds the files being copied to the end of the current tape, rather than replacing the previous contents. Files can be added only to TF tapes; you may not append to FC tapes or UNIX TAR tapes.

The time required to append to magnetic tapes is a function of the number of files already on the tape. Since a forward file search is not required for CTD (the end-of-data position is recorded in the tape header), appending is very efficient with CTD tapes. Refer to the section “Tape Protection” later in this chapter and the discussion of the K option below for special precautions to take when you append to tapes.

Brief Logging Mode (B)

This option suppresses the logging of each file name as it is copied, thus reducing the output to the terminal to a small number of messages, plus any error messages.

Clear Backup Bit (C)

The Clear option clears the backup bit for each file copied to tape, thus marking the file as backed up. This is useful when you use incremental backup, in which files are only backed up if they changed since the previous backup (the backup bit is set each time a file is accessed). Refer to the section on “Incremental Backup” in this chapter for more information on how to use the Clear option.

Replace Duplicate Files (D)

The Duplicate option causes existing files to be replaced with files restored from tape if the file names are the same. When you do not use D, a “File already exists” error occurs if you try to restore a file that has the same file name as an existing file, and the file is not restored. Refer to the section “Replacing Duplicate Files” later in this chapter for more information.

Ignore Errors and File Marks (I)

This option is useful in recovering as much data as possible from a partially corrupted (damaged or overwritten) tape. Normally, TF does not continue reading a tape if it does not recognize the header format (“unknown tape format” error) or if a “tape i/o error” occurs when reading the header. The I option causes errors in the tape header to be ignored. TF continues reading the tape even if the header is not valid and ignores any “unexpected eof mark” errors.

This option is valid only for TF and UNIX TAR tapes; it has no effect when reading FC tapes.

Keep Tape Online (K)

The Keep option prevents TF from taking the tape unit offline. While this option is a convenience, it makes it possible for the tape to be overwritten by another program, since the tape unit remains unlocked and online. Refer to the section on “Tape Protection” in this chapter for more information.

Inhibit UNIX to FMP Text File Conversion During Restore (N)

This option is normally used when you restore binary files from UNIX TAR tapes, but can be used whenever you do not want linefeed characters converted to record boundaries. Unless you specify the N option, this conversion is done for all files restored from UNIX TAR tapes and for type 4 files on TF tapes if you specified the X option when you backed up the files. Refer to the discussion of the X option below for more information.

Update (U)

Update is similar to Duplicate, except that when you use the Update option, duplicate files are only replaced if the update time of the file from the tape is later than the update time of the existing file on disk. See the section on “Replacing Duplicate Files” in this chapter for more information.

Verify Files Copied (V)

Verify compares the files on tape and on disk after they are copied for both disk-to-tape and tape-to-disk copy commands. The comparison is done after the entire copy is completed. TF uses a scratch file when it copies files to tape with Verify selected. The scratch file is created in the global directory ::SCRATCH, if possible. If this directory does not exist, you must create the scratch file in the working directory, or, if there is no working directory, on an FMGR cartridge. Refer to the section on “Installing TF” later in this chapter for more information.

UNIX Compatibility (X)

This option causes type 4 files to be converted to the UNIX text file format (in which line feeds are used to separate records) when files are copied to tape. No UNIX compatibility option is necessary when copying files from tape to disk. Refer to the section on “UNIX Compatibility” later in this chapter for more information.

Yes (Y)

Yes suppresses the questions, “Do you want to write over this tape (Y/N)?” and “Do you want to append to this tape (Y/N)?”. TF assumes a Y response when this option is specified.

Copy Examples without Subdirectories

The examples in this section are useful if you do not use subdirectories. All the files in the examples are files in global directories or on FMGR cartridges. Two-character names are used for FMGR cartridges (for example, ::x1) and longer names are used for global directories (for example, ::project1). The tape LU is LU 8.

These examples parallel the discussion in the earlier section, "CO Command Source and Destination Parameters."

Normal Backup Examples

Example 1: Back up all files in global directory PROJECT1, with the Verify (V) option selected.

```
co @.@::project1 8 V
```

You may prefer the following notation, which has the same effect:

```
co /project1/@.@ 8 V
```

When the name and type extension are both @, and no mask qualifiers are used, you may drop the @.@ to save typing and abbreviate the commands as:

```
co ::project1 8 V
```

or

```
co /project1/ 8 V
```

All four commands shown above are identical in effect. In the remaining examples, the "::" notation is used. The abbreviated form (with @.@ dropped) is used whenever possible. If a mask qualifier is used, you cannot drop @.@. For example, you cannot abbreviate @.@.X, where the X mask qualifier is used with @.@ to select files with extents (note that ..X is not allowed).

Example 2: Back up all files on cartridge X1.

```
co ::X1 8
```

Example 3: Back up all FORTRAN source files (files with type extension FTN) in global directory PROJECT1, with the B (Brief) option selected.

```
co @.ftn::project1 8 b
```

Example 4: Back up all files on cartridge X1 whose names begin with '&'.

```
co &@::X1 8
```

Example 5: Back up all type 4 files in global directory PROJECT1.

```
co ::project1:4 8
```

Example 6: Back up all files in global directory PROJECT1 not accessed since May 1983.

```
co @.@.a-8305::project1 8
```

Here the A mask qualifier is used to select files with particular access times (year 83, month 05). You may do this to back up old files before purging them to save disk space.

Example 7: Back up all files in the working directory.

```
co @.@ 8
```

If there is no working directory, this command backs up all files on all cartridges in the cartridge list.

Example 8: Back up two files.

```
co {file1 file2} 8
```

Example 9: Back up all FORTRAN, Pascal and MACRO source files in the directory PROJECT1, with the X (UNIX compatibility) and V (Verify) options selected.

```
co {@.ftn::project1 @.pas::project1 @.mac::project1} 8 X V
```

Alternatively, you may use DE to specify the common source directory, as follows:

```
de ::project1
co {@.ftn @.pas @.mac} 8 X V
```

Note that while DE sets a default source directory, this is NOT the same as setting a default working directory using the CI WD command.

Example 10: Back up files in the global directories PROJECT1, PROJECT2, and SYSTEM.

```
co {::project1 ::project2 ::system}8
```

Example 11: Back up all documentation files (type extension .doc) in global directories PROJECT1, PROJECT2, and PROJECT3 and verify.

```
co {@.doc::project1 @.doc::project2 @.doc::project3}8 V
```

Since the documentation files all have the type extension .DOC, you may use the DE command to save typing as follows:

```
de @.doc
co {::project1 ::project2 ::project3} 8 V
```

You may want to specify more masks than can be typed on a single command line. TF allows 150 characters, but practical considerations usually dictate a maximum of 80 characters. In this case, you can use the GR and EG commands to combine the copy commands into one operation and enter them as follows:

```
de @.doc
gr
co::project1 8 V
co::project2 8 V
co::project3 8 V
eg
```

Since all the directories in the above grouped commands are to be backed up to LU 8 and verified, you may expand the DE command to include the other common properties as follows:

```
de @.doc 8 V
gr
co ::project1
co ::project2
co ::project3
eg
```

Similarly, you may enter the following commands to back up a large number of files. Note that commas are used as placeholders for the omitted source parameter.

```
de,,8 V
gr
co file1
co file2
.
.
.
co lastfile
eg
```

Because DE is so versatile, you can use it to greatly simplify command entry, particularly with GR and EG to copy a large number of files. In the next example, DE is specified several times:

```
gr
de .ftn::dir1 8V
co file1
co file2.mac
.
.
.
co file50
de .ftn::dir2 8 V
co file51
.
.
.
co file100
eg
de
```

The first DE command specifies that the default file type extension is .FTN, the default source directory is ::DIR1, the default destination is LU 8, and the default option is Verify. Since one of the files, FILE2.MAC, is not a FORTRAN file, that file name is explicitly entered in the CO command, which overrides the default type extension for that file only.

The second DE command changes the default directory to ::DIR2. Since each subsequent DE command cancels the previous one, you must enter the entire set of desired default parameters each time you specify DE.

DE is entered a third time, this time with all null parameters, to cancel the defaults set by the second DE command. This is necessary only if you no longer desire defaults. Defaults remain in force until you override them by entering another DE command or until TF exits.

Refer also to the discussions of the Default and Group Copy Commands later in this chapter.

Normal Restore Examples

Example 1: Restore all files from tape LU 8.

```
co 8
```

If you omit the destination parameter, all files are restored to the same directories from which they were backed up (and with the same names and properties).

If the required directories do not exist when you do the restore, TF creates them automatically. You must mount any FMGR cartridges required prior to the restore.

Sometimes a directory is not specified when files are backed up, as in the following command:

```
co @.@ 8
```

In this case, the working directory is assumed. If there is no working directory, the FMGR cartridge list is searched for file1.

If a working directory exists, the files are restored to that directory even if it is different from the one in effect when the files were backed up.

Example 2: Restore all the files from tape LU 8 with the Verify option selected.

```
co 8,,V
```

You must separate the source and options parameters with two commas when you do not specify a destination parameter.

Selective Restore Examples

Example 1: Restore files from tape LU 8, selecting only the FORTRAN source files.

```
co 8{@.ftn}
```

In this example, all files are again restored to their original directories.

Example 2: Restore files from tape LU 8, selecting only those files from global directory PROJECT1.

```
co 8{::project1}
```

The files are restored to the same directory.

Example 3: Restore all FORTRAN, Pascal, and MACRO source files, and verify.

```
co 8{@.ftn @.pas @.mac},,V
```

Restoring to Different Directories or Files

Example 1: Restore all files on LU 8 to the global directory ALLPROJECTS.

```
co 8 ::allprojects
```

All the files are restored to ALLPROJECTS regardless of the directory from which they were backed up.

Example 2: Restore all files on LU 8 to FMGR cartridge X2.

```
co 8 ::X2
```

All file names are truncated to six characters, the maximum allowed under FMGR. Since this is not a function of TF but of the file system, no message is issued if a file name is truncated.

Example 3: Restore all files on LU 8 to the current working directory.

```
co 8 @.@
```

If there is no working directory, the above command restores each file to the first available FMGR cartridge on the cartridge list.

Example 4: Restore all files backed up from global directory PROJECT1 to the global directory PROJECT_XYZ.

```
co 8{::project1} ::project_xyz
```

Example 5: Restore all documentation files backed up from global directories PROJECT1 and PROJECT2 to global directory PROJ_1_AND_2_DOC.

```
co 8{@.doc::project1 @.doc::project2} ::proj_1_and_2_doc
```

Since you are copying the files to a special documentation directory, the .DOC type extensions are redundant, and you may remove the extensions from the restored files with this command:

```
co 8{@.doc::project1 @.doc::project2} @.::proj_1_and_2_doc
```

Example 6: Restore the single file X.DOC from the PROJECT1 directory to the file X_DOCUMENT in the working directory.

```
co 8{x.doc::project1} x_document
```

Example 7: Restore all documentation files to the global directory DOCUMENTATION.

```
co 8{@.doc} ::documentation
```

Example 8: Restore all DOCUMENTATION files to FMGR cartridge DO, removing the .DOC type extensions.

```
co 8{@.doc} @.::do
```

Names longer than six characters are truncated.

Special Purpose Backup Examples

Example 1: Back up all files in global directory PROJECT1, but label the files on tape as if the files came from directory PROJECT2.

```
co ::project1 8{::project2}
```

Example 2: Back up the file X.DOC in the directory PROJECT1, renaming the file INFO_ABOUT_X.

```
co x.doc::project1 8{info_about_x}
```

Example 3: Back up all FORTRAN source files in the working directory, removing the type extensions.

```
co  @.ftn 8{@.}
```

Copy Examples with Subdirectories

The examples in this section illustrate how to use subdirectories to take advantage of the hierarchical features of the file system. Many of the points discussed in the previous section, "Copy Examples without Subdirectories," also apply when you use subdirectories. Features that are common to both discussions (such as examples of how to use the GR command) are handled in more detail in the previous section.

These examples parallel the discussion in the earlier section, "Copy Command Source and Destination Parameters."

The tape LU in all these examples is LU 8.

Normal Backup Examples

Example 1: Back up all files in the global directory MYFILES, including all files in all subdirectories of MYFILES, and all subdirectories of those subdirectories, with the Verify option selected.

```
co /myfiles/@.@ 8 V
```

Alternatively, you may select the following equivalent notation; however, the "::" notation may become confusing when subdirectories are specified, as shown below.

```
co @.@::myfiles 8 V
```

When the name and type extension are both @ and no qualifiers are used, you can drop the @.@ to save typing. The commands shown above can thus be abbreviated as follows:

```
co /myfiles/ 8 V
```

or

```
co ::myfiles 8 V
```

Example 2: Back up files in directory MYFILES, subdirectory PROJECT1.

```
co /myfiles/project1/ 8
```

This command backs up all the files in the PROJECT1 subdirectory of global directory MYFILES, as well as all files found in any subdirectories within the PROJECT1 subdirectory.

Example 3: Back up files in subdirectory sources, within subdirectory PROJECT1, within global directory MYFILES.

```
co /myfiles/project1/sources/ 8
```

Alternatively, you could use the "::" notation in this command, as follows:

```
co project1/sources/::myfiles 8
```

In this notation, the hierarchical structure is less clear than it was in the first command. It is much easier to see that /MYFILES/PROJECT1/SOURCES/ refers to a directory MYFILES, which contains the subdirectory PROJECT1, which in turn contains a subdirectory SOURCES. Therefore, the remaining examples in this section use the hierarchical notation shown in the first command, rather than the "::" notation.

Example 4: Back up all FORTRAN source files in subdirectory SOURCES.

```
co /myfiles/project1/sources/@.ftn 8
```

If you set the working directory to /MYFILES/PROJECT1, you may use the following command to back up the same files:

```
co sources/@.ftn 8
```

If you set the working directory to /MYFILES/PROJECT1/SOURCES, you may use the following command to back up the same files:

```
co @.ftn 8
```

However, when you select files for backup relative to the working directory, the files are identified on the tape with file descriptors such as MODULE1.FTN (or SOURCES/MODULES1.FTN, for the previous example), which do not precisely specify the directory from which the file was backed up.

When you restore files from such tapes without specifying a destination (as in the restore command "co 8"), the file descriptors on the tape are interpreted relative to whatever working directory is in effect at the time of the restore.

This is convenient in some cases, such as when you transfer files that do not need to reside in a particular directory. However, for ordinary backup applications, we recommend that you specify files with full file descriptors, rather than use the working directory. This ensures that the files are restored to the proper directories.

Example 5: Back up all FORTRAN source files found in subdirectory SOURCES, or any subdirectory of that subdirectory, with the S mask qualifier specified.

```
co /myfiles/project1/sources/@.ftn.s 8
```

If you restore files from this tape to a specified destination directory, all the FORTRAN source files on the tape are copied directly into the directory FILES_FROM_8 without any intervening subdirectories, whether they originally came from SOURCES or from a subdirectory of SOURCES. This is due to the S mask qualifier. The command entry is as follows:

```
co 8 /files_from_8/
```

If you restore the files without specifying a destination, all the files are restored to their original directory or subdirectory, as in the following command:

```
co 8
```

Example 6: Back up all files in directory GLOBAL1, subdirectory SUB1, that were not accessed since May 1983.

```
co /global1/sub1/@.@.a-8305 8
```

The A mask qualifier selects files with particular access times (in this case, year 83, month 05).

You can use this command to back up old files before purging them to save disk space.

Example 7: Back up all files in the directory MYFILES, subdirectory PROJECT1, or in any subdirectories contained within the subdirectory, that were not accessed since Jan 4, 1983.

```
co /myfiles/project1/@.@.sna-830104 8
```

The A mask qualifier is used here with the S and N qualifiers. A selects files with particular access times (in this case, year 83, month 01, day 04). The N qualifier (No directories) is needed to keep subdirectory files from being selected, which would automatically cause files within them to be selected regardless of their access time.

Example 8: Back up all files in the working directory and in all subdirectories contained within the working directory.

```
co @.@ 8
```

Example 9: Back up FILE1, contained in the working directory.

```
co file1 8
```

Example 10: Back up all files in SUB1, a subdirectory of the working directory, and all files in subdirectories of that subdirectory.

```
co sub1/ 8
```

Example 11: Back up all MACRO source files in the working directory.

```
co @.mac 8
```

Example 12: Back up two specific files.

```
co {/global1/file1 /global2/sub1/suba/filex} 8
```

Example 13: Back up all FORTRAN and MACRO source files in directory MYFILES, subdirectory PROJECT1, subdirectory SOURCES.

```
co {/myfiles/project1/sources/@.ftn /myfiles/project1/sources/@.mac} 8
```

If the working directory is set to /MYFILES/PROJECT1/SOURCES, you may use the following command to copy the same files to tape:

```
co {@.ftn @.mac} 8
```

However, in this case, the files are identified on the tape with shortened file descriptors that do not specify the directory of origin, rather than full file descriptors such as /MYFILES/PROJECT1/SOURCES/MODULE1.FTN.

To shorten the command syntax without sacrificing full file descriptors, use the DE command to specify a default directory for the source parameter of the copy command, as follows:

```
de /myfiles/project1/sources/
co {@.ftn @.mac} 8
```

DE specifies defaults for source, destination, and option field values that you do not specify in subsequent commands. In the command above, a default is used for the source directory. The final / after SOURCES in the DE command indicates that only the files contained in directory sources are to be backed up, not the directory and its files.

The DE command lets you specify CO commands in abbreviated form. All abbreviations are "expanded" before the command is processed, and do not cause the file names to be abbreviated on the tape. For example, the abbreviated CO command above is expanded into the following command:

```
co {/myfiles/project1/sources/@.ftn /myfiles/project1/sources/@.mac} 8
```

Example 14: Back up all files in the sources and relocatables subdirectories within the PROJECT1 subdirectory.

```
co {/myfiles/project1/sources/ /myfiles/project1/relocatables/} 8
```

Normal Restore Example

Example 1: Restore all files from tape LU 8.

```
co 8
```

If you do not specify a destination parameter, all the files are restored to the same directories from which they were backed up, with the same names and properties.

However, if files are backed up relative to the working directory, the files are restored to the current working directory in effect when they are restored, which may not be the same as the working directory in effect when they were backed up.

Selective Restore Examples

Example 1: Restore all files from tape LU 8 that were backed up from subdirectory /MYFILES/PROJECT1/SOURCES.

```
co 8{/myfiles/project1/sources/}
```

This command does not work if the files were backed up relative to the working directory, because in that case, the files are not identified on the tape with the full file descriptor (starting with /MYFILES). Rather, they are identified with a shorter file descriptor. The file mask used in selecting files from tape must reflect this.

You can use the DL command to determine how the files are identified on the tape, so that you know how to select files being restored. (See the section "Relation of the DL command to the CO command" later in this chapter for details.)

Example 2: Restore all files from tape LU 8, selecting only the FORTRAN source files (type extension .ftn).

```
co 8{@.ftn.e}
```

The E qualifier (look Everywhere) must be specified to select files with type extension .ftn at any directory level. If you specify a particular directory, the E qualifier is not necessary, as shown in the following command:

```
co 8{/myfiles/project1/sources/@.ftn}
```

Example 3: Restore two specific files.

```
co 8{/global1/file1.ftn /global2/suba/filex.ftn}
```

Example 4: Restore all FORTRAN and MACRO source files from the subdirectory /MYFILES/PROJECT1/SOURCES.

```
co 8{/myfiles/project1/sources/@.ftn /myfiles/project1/sources/@.mac}
```

You may use DE to simplify the command, as follows:

```
de /myfiles/project1/sources
co 8{@.ftn @.mac}
```

Example 5: Restore all FORTRAN, Pascal, and MACRO source files, regardless of the directory they came from, using the E mask qualifier.

```
co 8{@.ftn.e @.pas.e @.mac.e},,v
```

Restoring to Different Directories or Files Examples

This type of restore can be selective (specifying a source mask) or non-selective (no source mask). Examples 1-5 illustrate selective restore.

Example 1: Restore all files backed up from global directory /MYFILES to the subdirectory /OLDPROJECTS/CATEGORY1.

```
co 8{/myfiles/} /oldprojects/category1/
```

The CO command above retains the hierarchical structure of the file system. When you copy the contents of global directory MYFILES, all levels of subdirectories are copied intact into the subdirectory CATEGORY1. The subdirectories are automatically created when you restore the files. For example, if you backed up /MYFILES/SUBFILES/FILE1, it is restored with the following path name:

```
/oldprojects/category1/subfiles/file1
```

Since MYFILES itself was not copied (the trailing slash in the source parameter /MYFILES/ says copy only the contents of the directory), that directory is no longer part of the FILE1 path name. Refer to the description of the CO command in the *RTE-6/VM CI User's Manual* for further information.

Example 2: Restore all documentation files (type extension .DOC) that were backed up from subdirectory PROJECT1 of global directory MYFILES, putting all the files in the global directory DOCUMENTATION.

```
co 8{/myfiles/project1/@.doc} /documentation/
```

Example 3: Restore all documentation files to the DOCUMENTATION directory, regardless of what directory or subdirectory they were backed up from, using the E mask qualifier.

```
co 8{@.doc.e} /documentation/
```

This command causes all the selected files to be copied into the specified directory without any intervening subdirectories, regardless of their original directory or subdirectory. In other words, the E qualifier causes the hierarchical structure to be compressed into one level.

However, when you select a directory file, the hierarchy below the selected directory is preserved. (In the example above, specifying the type extension .DOC excludes directory files.)

Levels are similarly compressed when you use the S qualifier, as follows:

```
co 8{/myfiles/@.doc.s} documentation/
```

This command selects all documentation files in global directory MYFILES (on the tape) or any subdirectory contained within MYFILES. All the selected files are copied into the DOCUMENTATION global directory with no intervening subdirectories, regardless of the directory or subdirectory level from which the file originally came.

Example 4: Restore all the same files and remove the .DOC type extension from the resulting files.

```
co 8{/myfiles/@.doc.s} /documentation/@.
```

Example 5: Restore a particular file, /MYFILES/PROGRAM_X.DOC, from tape into a particular destination file on disk, /DOCUMENTATION/INFO_ABOUT_X.

```
co 8{/myfiles/program_x.doc} /documentation/info_about_x
```

In a non-selective restore, the entire contents of the source tape are restored to the named destination. The path names of the restored files are determined by the way in which you specified the initial copy source parameter. Examples 6-8 illustrate non-selective restore.

Example 6: Back up the contents of SUB2 of SUB1 of MAS.

```
co /mas/sub1/sub2/ 8
```

The directory for this backup tape lists the path followed to access the files, then itemizes the files backed up. For example, assume the tape contains file FILEZ and a subdirectory of SUB2 called SUB3, which contains a file FILEX. You can restore this tape to disk with the following command:

```
co 8 /dira/suba/
```

In this case, FILEZ is restored as /DIRA/SUBA/FILEZ, and FILEX is restored as /DIRA/SUBA/SUB3/FILEX.

Example 7: Back up the MAS directory, all its subdirectories, and all the files therein.

```
co /mas.dir 8
```

You may restore this tape to disk with the command:

```
co 8 /dira/suba/
```

In this case, FILEZ is restored as /DIRA/SUBA/MAS/SUB1/SUB2/FILEZ, and FILEX is restored as /DIRA/SUBA/MAS/SUB1/SUB2/SUB3/FILEX.

Note, however, that if you restore the tape with the following command, you delete the directory /MAS from the path name for the restored files:

```
co 8 /dira/suba
```

Example 8: Back up all the files with a .DOC extension in the MAS directory, using the S qualifier.

```
co /mas/@.doc.s 8
```

When you use the S or E qualifiers in selecting the files for backup, TF discards the directories and subdirectories it searched to determine the destination when you restore them with the command:

```
co 8 /dira/suba/
```

the restored files are put in the specified subdirectory with no intervening subdirectories.

You can use the DL command to see the directory, to find out how the files are restored. Refer to the discussion of the DL command in this chapter for more information.

Special Purpose Backup Examples

Example 1: Back up all files in subdirectory /MYFILES/PROJECT1/SOURCES, but label the files on tape as if they had come from a global directory called PROJECTX.

```
co /myfiles/project1/sources/ 8{/projectx/}
```

Example 2: Back up all the files in the DOCUMENTATION directory, but label the files on tape as if they came from a directory called PROJECT1. Give all the files on tape the type extension .DOC, regardless of what type extension they had on disk.

```
co /documentation/ 8{/project1/@.doc}
```

Example 3: Back up all the system files that have type extension .DOC, label them all as if they came from the global directory DOCUMENTATION, and remove all the .DOC type extensions from the files on tape.

```
co @.doc.e 8{/documentation/@.}
```

Copy Examples Using DS

Although TF cannot access a remote tape unit, you can run TF from a system with a local tape unit and then access the backed up or restored disk files over DS. For remote backups and restores of several files, you must specify a superuser logon name in the command. (Being logged on as a superuser locally does not affect remote access.)

Unlike other file attributes, the DS location for a file being restored does not default to the same location from which the file was backed up. If the destination DS location is not specified, a local destination is assumed.

Be sure that the system times are correctly set at both the local and remote systems. Refer to the section "Maintaining the System Time" later in this chapter for details on the system time stamps. Refer also to the section "Saving and Restoring Properties of Files" later in this chapter for additional information on using TF with remote files.

When files are backed up and restored using DS, delimiters are used to specify a remote node and logon. Note that if "]" or ">" occurs at the beginning of a file descriptor or immediately following a slash, it is treated as the first character of a file name, not as a DS delimiter.

When you use a DS delimiter to specify a remote destination, you must precede the DS delimiter by at least one other character.

Example 1: Restore a tape to the same directories from which the files were copied at DS node >mynode.

```
co 8 0>mynode
```

If the delimiter is the first character, precede it with a "0". TF interprets the zero as a blank and correctly interprets the next character as the DS delimiter. The following command is incorrect:

```
co 8 >mynode
```

In this case, TF interprets the > delimiter as the first character of a file name, and it restores the tape to local file >mynode.

Example 2: Restore tape containing a single file to filez at DS note >mynode

```
co 8 filez>mynode
```

In this case, the file name FILEZ precedes the DS delimiter >, and the backup is successful.

When you specify a remote file descriptor in the source or destination, you must not specify the delimiter as the first character following the "/" in the descriptor. Use an "at" sign (@) after the slash to identify the > or [as a DS delimiter. The next example illustrates the use of the @ sign.

Example 3: Back up files in /DIRA at DS location [mylogon]>mynode to LU 8:

```
co /dira/@[mylogon]>mynode 8
```

In this command, the delimiter is preceded with @. This tells TF that the next character is a DS delimiter, rather than the first character of a file name. The following command is incorrect:

```
co /dira/>mynode 8
```

Here TF interprets > as part of the file name and attempts to back up the local /DIRA/>mynode files to tape.

Example 4: Restore tape to /DIRA at remote location [mylogon]>mynode, where [mylogon] is the user logon name at the DS location.

```
co 8 /dira/@[mylogon]>mynode
```

Note that the logon name is used only in multiuser systems. If the DS location is not multiuser, you must not use a logon name. If a password is associated with the logon name (multiuser systems only), you must specify it. For example:

```
co /dira/@[mylogname/mypassword]>mynode 8
```

With the exception of DS delimiter handling, copy operations with TF are identical for both local and remote backups and restores. The following examples summarize the similarities and differences in the TF command syntax. In all cases, the first command in each example is a local TF command, and the second is a remote TF command.

Example 5: Back up all files in global directory MYFILES.

```
co /myfiles/ 8
co /myfiles/@>mynode 8
```

Example 6: Restore all files to global directory THEFILES.

```
co 8 /thefiles/
co 8 /thefiles/@>thenode
```

Example 7: Back up all files in the hierarchical file system (system file backup, generally done only by a superuser).

```
co / 8
co /@[superusername]>thenode 8
```

Example 8: Restore all files to their original directories.

```
co 8
co 8 0[superusername]>thenode
```

Example 9: Back up all files on file system LU 27.

```
co 27 8
co 27[superusername]>thenode 8
```

Example 10: Restore all files to file system LU 54 (except for files in directories that already exist on other LUs).

```
co 8 54
co 8 54[superusername]>thenode
```

In all the examples above, @ is used only in the remote command because it is a requirement when using TF with DS; however, you may specify @ in the local file descriptors if desired. The three commands that follow all have the same effect in a local backup of all files from global directory MYFILES:

```
co /myfiles/ 8
co /myfiles/@ 8
co /myfiles/@.@ 8
```

For remote node operations, only the following two forms of the command work properly:

```
co /myfiles/@[user]>node 8
co /myfiles/@.@[user]>node 8
```

Default (DE)

Purpose: To set up defaults for the CO command source, destination, and options parameters.

Syntax: `de <source> <destination> <options>`

<source> <destination> <options>	As defined for the CO command, except that a list of masks may not be used in the source parameter.

Description:

The DE command is useful when you are entering a number of CO commands in succession and all or most of them have the same parameters, such as the same source or destination tape LU or disk directory, or the same options. DE lets you specify the value once for all the related CO commands.

The DE command is especially useful in conjunction with the GR command (see the section on "Group Copy Commands" and the example of DE command use in a subsequent section).

You may use the DE command source parameter to set up default values for the tape LU, file name, type extension, file mask qualifier, security code, directory (specified in the directory field or in front of the file name), type, size, record length, or DS location items of the CO command source parameter. The default value is subsequently used whenever you do not specify another value in the source parameter of the CO command, except for the qualifier field. Mask qualifiers included in the DE command are merged with any qualifiers specified in subsequent CO commands.

Similarly, the DE command destination parameter can be used to set up default values for the tape LU, name, type extension, security code, directory, size, or DS location items of the CO command destination parameter.

In order to default the DS location only in the source or destination parameter, enter a colon (:) before the "[" or ">" character at the beginning of the DS field. This keeps the "[" or ">" from being interpreted as the first character of the file name. Note that in these examples, "[", "]", and ">" are characters that are actually typed, rather than part of the syntax notation. For example:

```
de :[sourceuser] :[destuser]
de :>sourcenode :[destuser]
de :[sourceuser]>sourcenode :>destnode
```

Note that the user name and node name are both part of the DS location and are not separated when defaults are used. If the CO command source (or destination) parameter specifies either the user name or the node name, the default value is not used for either name.

When you include options in the DE command, they are merged with any options you specify in subsequent CO commands. For example, a DE/CO command set that has the following form:

```
de,,,pv
co <source> <destination> cp
co <source> <destination>
```

specifies options C, P, and V for the first CO command, and P and V for the second.

Each DE command supersedes the preceding default; therefore, you can cancel defaults by entering a DE command with all null parameters.

Directory List (DL)

Purpose: Provides a directory list of files on a TF tape specified by the command.

Syntax:

```
dl <tape> [i][k]
```

or

```
dl <tape>{<mask>} [i][k]
```

or

```
dl <tape>{<mask1> <mask2>...} [i][k]
```

tape The positive or negative tape LU.

mask The specifier of certain files to be listed.

Description:

Since the mask works exactly the same as in the CO command, you can use DL to show what files are to be copied by a particular mask. If you do not specify a mask, all the files on the tape are included in the listing.

The I option causes TF to ignore errors in the tape header and unexpected file marks. Refer to the earlier description of the I option for more details.

You may specify the K option to keep TF from taking the tape unit offline. Be aware that when you use K, the tape unit is unlocked but left online; thus, it is possible for another program to overwrite the tape. Refer to the section in this chapter on “Tape Protection” for more information.

You can use LL to direct the DL listing to another device or disk file.

The following example shows the directory list format:

```
TF:                     dl 8

Tape format:            TF
Title:                  TF:   co {/programs/ /project1/} 8 v
Date:                   Mon Mar 21, 1983  5:21:59 pm

Created Mon Mar 21, 1983  5:21:59 pm:

File from directory /PROGRAMS/      Update Time or Owner

FOWN.RUN:::6:123:128                Tue Mar 15, 1983 11:38:04 am
FREES.RUN:::6:39:128                Tue Mar 15, 1983 11:38:06 am
FPACK.RUN:::6:158:128               Tue Mar 15, 1983 11:38:10 am
CLOSE.RUN:::6:12:128                Tue Mar 15, 1983 11:38:12 am
CI.RUN:::6:229:128                  Tue Mar 15, 1983 11:38:21 am
FVERI.RUN:::6:152:128               Tue Mar 15, 1983 11:38:08 am
DL.RUN:::6:136:128                  Tue Mar 15, 1983 11:38:03 am

File from directory /PROJECT1/      Update Time or Owner

SOURCES.DIR:::2::32                           MYUSERNAME
INCLUDES.DIR:::2::32                MYUSERNAME
SOURCES/MODULE1:::4:31:39           Fri Mar  4, 1983  3:32:32 pm
SOURCES/MODULE3:::4:136:39          Thu Mar  3, 1983  1:52:36 pm
SOURCES/MODULE2:::4:236:39          Fri Mar  3, 1983  3:32:50 pm
SOURCES/MODULE4:::4:20:38           Fri Mar  4, 1983  3:32:51 pm
SOURCES/MODULE6:::4:2:34            Fri Mar  4, 1983  3:32:53 pm
SOURCES/MODULE5:::4:397:39          Fri Mar  4, 1983  3:33:00 pm
INCLUDES/INCLUDE1:::4:1:31          Wed Mar  2, 1983 10:27:12 am
INCLUDES/INCLUDE2:::4:1:17          Wed Mar  2, 1983 11:14:34 am
INCLUDES/INCLUDE3:::4:4:36          Fri Mar  4, 1983  3:33:16 pm
INCLUDES/INCLUDE4:::4:12:35         Tue Mar  1, 1983  4:15:25 pm
```

The directory listing begins with the tape header. The date in the header is the date and time at which the first file was written to the tape.

The “Created” line represents the date and approximate time at which the following list of files was created on the tape by the copy operation; it does not represent the time at which the original file was created. Whenever files are appended to the tape, a new “Created” line precedes the list of the new files and gives the date and time they were appended.

If the “File from directory” line appears, the directory name combines with listed files to form the full file descriptor (for example, /PROGRAMS/FOWN.RUN:::6:123:128). If this line does not appear, the file as listed is the full file descriptor.

Note that the update time for FMGR files is always reported as 1/1/70, since FMGR files do not have an update time in their directory entry.

Relation of the DL Command to the CO Command

The DL command helps you determine how to select files to copy from a tape. You may select a file by its full file descriptor, which includes the name from the directory heading line. Alternatively, you may select one by the reference file descriptor on the file line that follows the directory heading line (if a directory is specified in the line). Referring to the example directory listing shown in the DL command section above, the following two CO commands select the same file from the tape:

```
co 8{/project1/sources/module1}      (Full file descriptor)

co 8{sources/module1}                (Reference file descriptor)
```

(If the header line does not specify a directory, both file descriptors are identical.)

If you do not know which files a given file mask selects, try the mask first with DL before you use it in a CO command. For example:

```
dl 8 {sources/module1}
```

When you issue this command, the files listed are the ones that will be copied if you use the same mask with the CO command.

You can use the reference file descriptor to determine the directory hierarchy created by a non-selective restore to a specified directory, as in the following command:

```
co 8 /global/sub/
```

Assume the tape is the same as the one shown in the example of the DL listing above. The first file that this command restores is /global/sub/fown.run (not /global/sub/programs/fown.run), because the reference file descriptor, rather than the full file descriptor, is used.

Since this command does not copy the programs directory to tape but just the files in it (it does not specify /programs.dir, just /programs/), no programs directory is created when the files are copied from the tape. If you specify /programs.dir in the backup command, rather than /programs/, the reference file descriptor is /programs/fown.run instead of fown.run, and the file is restored as /global/sub/programs/fown.run.

The effect of using this method is that when you back up files and later restore them to a different destination, as in the following two commands:

```
co /programs/ 8
co 8 /global/sub/
```

the result is the same as if you copied the files directly from the source in the first command to the destination in the second command, as in the following CI utility command:

```
CI> co /programs/ /global/sub/
```

The directory header line always shows the name of the global directory (or FMGR cartridge), and the file lines include the name and type extension. Files from such tapes can be selected by any combination of properties, for example:

```
co 8{fown.run}
co 8{fown.run::programs}
co 8{::programs}
co 8{@.ftn::project2}
```

Exit (EX)

Purpose: Terminates TF.

Syntax: `ex [k]`

k — This option keeps the most recently used tape LU from being taken offline. Note that this makes it possible for another program to overwrite the tape, because the tape unit is unlocked, but left online. (Refer to the section on "Tape Protection and the K Option" for more details.)

Group Copy Commands (GR, EG, and AG)

Purpose: To combine multiple CO commands into a single operation.

Syntax:

```
group
co <parameters>
co <parameters>
.
.
.
co <parameters>
eg
```

Description:

Some copy applications require combining multiple CO commands into a single operation. It is useful to group commands, for example, when there are too many source masks to fit on one command line or when different destinations or options are needed for different files.

When TF encounters the GR command, it keeps track of all subsequent CO commands, but does not execute them immediately. Instead, TF executes them as a single operation when it encounters the EG (End Group) command. You can abort the GR operation by using the AG (Abort Group) command, which causes the group to be terminated before any of the CO commands are executed.

If the grouped CO commands have a common source or destination, you may use the DE (Default) command to advantage. For example:

```
de 8                    (Common source tape LU)
gr                      (Start group)
co ::x ::a              (Restore files from directory X to directory A)
co ::y ::b              (Restore files from directory Y to directory B)
co ::z ::c              (Restore files from directory Z to directory C)
eg.                     (End group)
de,,8                   (Common destination tape 8)
gr                      (Start group)
co ABCD WXYZ            (Copy file ABCD to 8 as WXYZ)
co EFGHIJ QRSTU         (Copy file EFGHIJ to 8 as QRSTU)
eg                      (End group)
```

Refer also to the earlier section on the DE command.

If you group many CO commands, you may need to increase the size of EMA for TF, or else the following message may be displayed:

```
EMA size of TF not large enough for a group of this size.
```

See the section on “Installing TF” later in this chapter for more information.

Help (HE)

Purpose: Writes a summary of TF commands to the terminal.

Syntax: `HE` or `?`

List Header File (LH)

Purpose: Lists the tape header file to the list device.

Syntax: `lh <tape> [k]`

tape	The positive or negative tape LU.
k	The option that keeps TF from taking the tape offline. Note that this option makes it possible for another program to overwrite the tape, since the tape unit is unlocked but left online. See the section on “Tape Protection” for more information.

Description:

The following is an example of the LH command listing:

```
Tape Format:        TF

Title:              TF: co /project1.dir 8
Date:               Mon Mar 14, 1983  1:28:05 pm
Capacity:           16352 kilobytes
Used:               725   kilobytes
```

“Capacity” and “used” lines appear only for CTD tapes and indicate the total data capacity of the CTD cartridge and the amount of data that the cartridge currently contains. (A CTD block is one kilobyte.)

For multi-tape backups, tapes that follow the first one also show the tape number in the LH listing after the title and date (for example, “Tape number: 2”).

List Device (LL)

Purpose: To specify a device or file other than the terminal to receive list information.

Syntax: `ll <file>`

file	The file descriptor of the device or disk file to receive the list information. If a disk file is specified, it is created. An existing disk file is not overwritten.

Description:

Initially, the list device for TF is the terminal.

Tape directory listings, header files, comment files, “?” command information, and command echoes are listed with the LL command. Information and error messages are always displayed on the terminal and are not affected by the LL command.

Title (TI)

Purpose: Sets the title to be used on tapes created by subsequent CO commands.

Syntax: `TI <tape title>`

<tape title> The statement of up to 80 characters (extra characters are truncated and a warning is given) that identifies a particular tape.

Description:

The TI command must precede the CO (or GR) command that defines the files to be copied.

You can use the LH command to read the title of a tape. In addition, the title appears at the beginning of a DL command listing and is displayed when you copy files from a tape. The title appears first on a TF tape, and for magnetic tape, can be listed directly from the CI LI (List File) command. The following example shows how to specify a title when you back up files and how to list the title using LH or LI.

```
CI> tf
TF: ti FORTRAN source files for project XYZ.
Title used will be: FORTRAN source files for project XYZ.
TF: co /xyz/@.ftn 8 v
:
:
TF: lh 8
Tape format: TF
Title: FORTRAN source files for project XYZ.
Date: Thu Mar 17, 1983 10:35:41 am
TF: ex k
CI> li 8,,1
FORTRAN source files for project XYZ.
CI>
```

Note that when LI is used, other information may follow the title on subsequent lines. This information can be ignored.

If you enter a CO command but do not specify a title, TF supplies a default title, which consists of the CO command (exactly as typed), preceded by "TF". In the example above, if you omit the TI command, the tape title is:

```
Tape format:      TF
Title:            TF: co /xyz/@.ftn 8 v
Date:             Thu Mar 17, 1983 10:35:41 am
```

"TF:" is included in the title to make the tape easier to identify when it is listed directly, such as when you use the CI LI command.

TI has no effect when you append files to a tape (that is, when you select the Append option to use with the CO command).

Transfer Command (TR)

Purpose: Lets you transfer control to and return from a command file.

Syntax:

`tr <file>` Transfer control to the named command file or input device.

tr Return to the next higher level command file when TR files are nested. (When encountered in the first level command file, this serves the same function as the TF EXIT command.) A return is done automatically when the end of a command file is reached, so this form of the TR command is not usually needed.

Description:

Command files can be nested to four levels, with control passed from file level to file level using the TR command.

Tape Protection and the K Option

TF keeps the tape LU locked when you issue commands that access tape to protect the tape against access by other programs. If the tape is write-protected, TF unlocks the LU at the end of the command because no further protection is necessary.

If the tape is not write-protected, TF keeps the tape LU locked until the utility exits or another tape LU is accessed. When the LU is unlocked, the unit is taken offline (in the case of the CTD, the cartridge is unloaded by TF) to protect the data.

At times, you may need to access the same tape on more than one successive TF command (such as LH followed by DL). In those cases, enter all the commands as a single execution of TF, as follows:

```
CI> tf
TF: lh 8
TF: dl 8
TF: ex
```

In the example above, the tape is not taken offline until the EX command is encountered.

Write-protecting the tape also keeps it from being taken offline.

Another way to keep the tape online is to use the K option. However, K leaves write-enabled tapes unprotected from the possibility of being overwritten by other programs. The K option can be specified with any tape command (CO, DL, or LH) or in the EX command. The option need only be specified once each time TF is run, since it remains in effect until TF terminates.

Be particularly careful to protect a tape being appended, since the data already on the tape must be preserved, yet the tape cannot be write-protected. You should load the tape after you enter the CO command rather than before. (For CTD, all that matters is that the tape does not complete the loading process until after the command is entered). When you enter CO, TF displays the message:

```
Tape not ready.
Type GO when tape is ready, or type BR to terminate.
```

Load the tape, then type GO. This ensures that the tape LU is locked before the tape is ready for access, which protects the previous contents of the tape from being overwritten.

Time Stamps

The file system maintains three time stamps for each file on disk: create time, update time, and access time. Similarly, TF maintains time stamps for files on tape. The meanings of time stamps on tape are analogous to their meanings on disk.

Note that update times are preserved when files are copied (whereas create times and access times are not). This is done so that the update time can be used as an indication of the revision date of the file. It is important that the revision date is not lost when a file is copied, backed up, or restored.

Create Time

This is the approximate time the file was created on the tape. The same time is used for all files copied to tape in the same command (or group of commands) so that separate segments of incremental backups can be easily identified. This time stamp is not present on UNIX TAR tapes. (See "Missing Time Stamps" below.)

Update Time

Since the file on tape is never directly updated, but is always just a copy of a disk file, the update time for the tape file is always the same as the update time of the disk file being copied.

Access Time

No access time is maintained for tape files.

Missing Time Stamps

As indicated above, files on tape do not have access time stamps, and files on UNIX TAR tapes also do not have create time stamps.

For the purposes of selecting files from a tape and doing a DL of a tape, it appears as if these missing time stamps are set to 1/1/70. This is the time that is used by convention whenever the true time is unknown or undefined, such as in FMGR files.

Maintaining the System Time

Confusing and perhaps serious consequences may occur if you do not maintain the system time (or set it backward during a restore). Some of the possibilities are as follows:

- The Update option may produce undesirable results if the system time is not maintained correctly.
- Even if you did not specify the Duplicate or Update option, duplicate files may be purged without warning on a restore.
- Restoring incremental backups may not work correctly if the system time is set backward during a restore. TF may restore out-of-date versions of some files rather than the latest version.

When TF is used to access remote files (that is, over DS), the system time must be correct at both the local and the remote system.

Incremental Backup

Incremental backup is a procedure that involves periodically backing up all the files and doing frequent backups only of files that were changed since the previous backup. (This procedure is not applicable for FMGR files.) In this discussion, a backup of all files is called a "full backup," and a backup of just those files that were changed is called a "delta backup."

An incremental backup consists of one full backup followed by a series of delta backups. Restoring from an incremental backup, therefore, consists of restoring the most recent full backup followed by all of the delta backups done since then.

In the standard incremental backup procedure, a full backup is done at the beginning of a tape, and then subsequent delta backups are appended to the same tape. Each new incremental backup is done on another tape. (An alternative to the standard incremental backup procedure is discussed below.)

Incremental backup has the following advantages over ordinary backup:

- The average backup time is faster, since most of the backups are delta backups, which generally take less time than full backups. As a result, system availability is improved. Since the incremental backup process is faster, other access to the files is restricted for a shorter period of time. (See the section in this chapter on "File Access During Backup and Restore Operations.")
- Less tape is used on the average.
- Fewer tapes are used, since several backups are appended to the same tape.
- Multiple versions of files can be accessed more conveniently on tape (good for archiving applications).

The disadvantages are that incremental backups take longer to restore, and the procedures involved are more difficult to understand.

Backup Bit, B Qualifier, and C Option

Incremental backup is accomplished by making use of the backup bit. The file system provides a backup bit for each file (excluding FMGR files). Since the backup bit is automatically set by the file system whenever a file is modified, it can be used to keep track of which files were changed since the last time they were backed up.

The backup bit is cleared when a file is backed up with the Clear Backup Bit (C) option specified. The Clear option also implies the Verify option. The backup bit for each file is cleared after the file is verified. This ensures that a file's backup bit is not cleared until the file is successfully backed up.

If the B qualifier is specified on a backup (as in the file mask /project1/@.@.b), the backup bit is checked to determine which files are to be backed up. Using this procedure, it is possible to back up only those files that were changed since the previous backup. This does not apply to FMGR files, as specifying the B qualifier for those files has no effect and all the FMGR files are copied.

To ensure that incremental backup works correctly under all circumstances, a fairly complex scheme is used in handling the backup bit for a file and interpreting the B mask qualifier. The file system sets a file's backup bit when any of the following conditions occur:

- The file is created.
- The file's contents are changed.
- The file's name, type extension, protection bits, or owner (directory files only) is changed.
- The file is moved to a different directory (by the CI MO command or by a call to FmpRename).

A file's backup bit is cleared when it is copied to tape by TF with the Clear option specified.

To determine whether a given file is selected by a file mask containing the B qualifier, TF first considers all other fields of the file mask to learn whether the file matches the mask (the B qualifier is ignored). If a directory file matches the mask, all the files in that directory automatically match (due to the implicit D qualifier used in copy commands).

If a file matches the mask this way, either directly or because its parent directory matches, then TF considers the backup bit to determine whether the file will actually be selected. This depends not only upon the backup bit for the file itself, but also upon the backup bits for the directories above the file in the hierarchy up to the level where the mask search started. (For the mask /A/B/@.@.b and the file /A/B/C/D/E, the backup bits in C, D, and E are checked.)

The net result is that a file is selected if the following two conditions are satisfied:

- The file or any of the directories above it match the mask, disregarding the B qualifier.
- The file or any of the directories above it (up to the level where the search began) has its backup bit set.

Standard Incremental Backup Procedure

In terms of options and qualifiers, the standard incremental backup procedure consists of periodically doing a full backup (no B qualifier) at the beginning of a tape (no Append option), and then more frequently (perhaps daily) appending a delta backup (Append option and B qualifier specified). In both cases, the Clear option is specified.

For example, you may perform an incremental backup of the directory /project1/documentation.dir (and all files in the directory and all subdirectories) to tape LU 8 as follows:

1. On the first day of the week start with another tape and use this command to do a full backup:

```
co /project1/documentation.dir 8 c
```

2. On each following day of the week, mount the same tape and use the following command to append a delta backup to the tape (note the addition of the B qualifier and the Append option):

```
co /project1/documentation.dir.b 8 ac
```

It is important to use the same source file mask for every segment of an incremental backup (except for the presence or absence of the B qualifier). Otherwise, you may encounter confusing results when you restore the files.

Selecting files relative to the working directory is also discouraged, since the working directory may be changed. To reduce typing errors, which could have serious consequences, standard backup commands should be kept in TF transfer files.

Before you decide how often to do a delta backup and how often to start a new incremental backup, there are several factors to consider, such as the frequency of file modifications, the size of the tape reel, and so on.

Note that for CTD, the amount of tape used and the total amount on the cartridge is displayed at the end of each backup, and you can use the LH command to display the amount at any time. For magnetic tape, you can determine the tape usage only by observing the tape during backup.

Multiple Copies of the Same Backup

You can repeat the same CO command to make multiple copies of the same backup (on different tapes). However, for incremental backups, you must omit the Clear option except for the final CO command. When you do this, it is important to make sure that none of the files involved in the backup get updated in between two of the copies. Otherwise, the multiple copies will contain different files, and only the last one (which had the Clear option) will be valid.

Reusing the Backup Bit

If you maintain two independent sets of backup tapes that contain files in common, you cannot use incremental backup for both sets of tapes, since there is only one backup bit per file. For example, if everyone backs up his or her own files and, in addition, the system manager backs up everyone's files, incremental backup should NOT be used. Rather than merely providing redundant backups of the same files, in this case incremental backup would result in incomplete backups of all the files.

Restoring Incremental Backups

To restore the entire contents of a backup tape, use the following command:

```
co <tape>
```

This restores all files on the tape as they were at the time of the backup. If the tape is an incremental backup, this restores the files as they were at the time of the last backup; however, some files that were purged, renamed, or moved elsewhere since the first segment on the tape was made may also be restored.

For example, files that were renamed may be restored both with the old and the new name. Although some old files that no longer exist may be restored, it is always guaranteed that all the files that did exist at the time of the last backup segment are restored correctly.

Incremental backup tapes may contain multiple versions of a given file. When you restore from an incremental backup tape, the latest version of each file is restored. The latest version is the last copy of the file on the tape, not necessarily the version with the latest update time. (This is because it is possible to discard a new version and revive an old version of a file.) However, in the process of restoring the tape, each version of the file is restored and is replaced by the next version when it is encountered. Therefore, once the entire tape has been restored, only the last version of each file remains.

When you restore incremental backup tapes, it is normal to see the message "Replacing <name> (created during this copy command)" as each version of a file is replaced by the next version. If you specified the Verify option, it is also normal to see the message "Update times don't match. Not Verifying <file>" as each older version of a file is encountered during the verify process.

The replacement of each version by the next is automatic, and does not require you to specify the Duplicate option. In TF, Duplicate determines whether a file from the tape replaces a duplicate disk file that already exists when the restore operation starts. If a duplicate file does exist, TF assumes that it was created by TF earlier in the restore, and thus is an earlier version of the same file; therefore, TF replaces it without requiring any options.

Although this assumption may not always be true, the results are always correct unless another program creates a file that duplicates a file TF is in the process of restoring. If this happened, TF would inadvertently purge and replace a file created by the other program, even if Duplicate was not specified. To avoid this problem, make sure no other programs access directories that are in use during a TF restore operation.

If a restore operation from an incremental backup tape does not reach completion for any reason, even the files that appear to have been restored correctly may not be up to date, since the latest version of each file is the one that occurs last on the tape.

Restoring Older Versions from Incremental Backup Tapes

To restore files to the state they were in (not as of the most recent backup segment, but as of some previous segment), select files with create times that are less than or equal to the time of that particular segment. For example:

```
CO 8{@.@.C-830704}
```

This is the most reliable way to restore an earlier state of the file environment. If you restore the file based on the update time, you may not achieve the results you expect, because update times are not meaningful for directories.

The DL command can be helpful in deciding what to do in such cases. The DL listing groups files according to backup segments. Each segment has a heading that shows the create time for the files in that segment (the time the files were created on the tape).

Setting Backup Bits on Restore

When files are restored, their backup bits are automatically set. This indicates that the newly restored files were not backed up. (Even though the files were just restored from tape, that is not the same as being backed up because the tape containing the files may not be a backup tape and may not have been saved as a backup.)

Alternatives to Standard Incremental Backup

An alternative way of backing up files is to start a new tape with a delta backup rather than a full backup. The result is an incremental backup that is not confined to a single tape (or multi-tape set). There are a number of variations of this method. One possibility is to put only one backup segment on a tape. This may be desirable if your delta backups tend to be large.

Another possibility is to do the full backup on one tape and all the delta backups on another tape. This ensures that the full backup tape cannot be damaged during the delta backup. It also becomes practical to keep the delta backup tape on a drive over several backups and devise a method for automatic delta backups at frequent intervals. Although there are risks in keeping a tape on the drive, only the delta backups are at risk.

Since the delta backup tapes are not a multi-tape set, each must be restored with a separate CO command. You must use the Duplicate option with every CO command after the first command, since TF automatically replaces duplicate files only if they are encountered during a single CO operation. (Be sure to restore the tapes in the correct sequence so that duplicate files are replaced in the correct order.)

Replacing Duplicate Files

TF uses certain rules to determine whether to replace a duplicate file that it encounters during a restore operation. A duplicate file is one that already exists with the same name and directory as the file being restored from tape. In making the determination, TF considers whether the Duplicate and Update options were specified and whether the duplicate file was created since the restore operation began.

This applies only to ordinary files, as opposed to directory files. A duplicate directory file is not treated as an error. When a duplicate directory is encountered, the existing directory is left there and is not replaced. Files in the directory may be restored to the existing directory, so there is no need to replace the directory.

TF puts a duplicate file into one of two categories, based on how the create time relates to the time the restore operation started. (This does not apply to FMGR files, which are discussed separately.)

- New duplicate file: It was created since the restore operation began.

  TF assumes the file was created earlier in the restore process (by TF itself) and replaces it with the new file unconditionally. This is done so that incremental backups can be restored properly without requiring any special options. The reasons for this and possible side effects are discussed in the earlier section on "Incremental Backup."

  When a new duplicate file is replaced, TF logs the message:

  ```
  Replacing <name> (created during this copy command).
  ```

  The original file is purged before the tape file is restored. If an error occurs when the file is restored from tape, the original file is still purged but is not replaced by anything. This is done so that errors in restoring incremental backup tapes do not cause the wrong version of a file to be restored. (It is better to restore no file at all than to restore the wrong version.)

- Old duplicate file: It was created before the restore operation began.

  Such a file could not have been created by the restore process and is therefore dealt with in the normal way, based upon the D and U options.

  If the Duplicate option is specified, the duplicate file is replaced by the file from the tape.

  If the Update option is specified, the duplicate file is replaced by the file from the tape only if the update time of the file from the tape is NEWER than the update time of the duplicate file on disk. The error "File already exists" is reported for all other files. This is useful for updating software and similar applications.

  When an old duplicate file is replaced, TF logs the message:

  ```
  Replacing <name>
  ```

  When the attempt is made to replace a file because of the Duplicate or Update option, the duplicate disk file is not replaced until after the file from tape is restored successfully to a scratch file, which is then renamed. If an error occurs when the file is restored from tape, the file originally present on the disk remains unchanged.

Because FMGR files have no time stamps, they are never replaced unless the Duplicate option is specified. For FMGR files, you cannot replace duplicate files with security codes, unless the security code of the file you are restoring matches the security code of the existing file. You may use the destination parameter to specify a matching security code to allow replacement to occur.

Automatic Creation of Directories on Restore

If you try to copy a file into a directory that does not exist, TF automatically creates all the directories required for the files being copied. This can happen when the CO command source parameter specifies the files in a directory, but not the directory itself (for example, you specified "/example/" instead of "/example.dir"), or when the CO command destination parameter specifies one or more new parent directories for the files being copied, as shown in the following command:

```
copy 8{/myfiles/@.ftn} /myfiles/sourcefiles/fortran/@.ftn
```

When a directory file is copied explicitly, its various properties (protection, owner, and disk LU) are sometimes copied as well (see the section "Saving and Restoring File Properties," below). However, when a directory file is created automatically, there is no file from which to copy the properties, so default values are used.

Saving and Restoring File Properties

TF does more than just save and restore file descriptors and file data. It also preserves the various properties of each file it copies. The following properties are saved for each file copied to tape (excluding directory files) and restored when the file is copied back to disk:

```
Security code (FMGR files only)

Numeric file type

Record length

Protection information

Update time (See the section on "Time Stamps.")
```

For directory files, the following additional information is saved:

```
Owner name

Disk LU
```

When remote directories are copied to tape, the true owner is not saved. Instead, all remote directory files are considered to be owned by the user name SYSTEM.

Note that DS location (user name—not to be confused with directory owner and node name) is not saved on tape. To prevent confusion, you should not save files from more than one DS node on the same tape.

Properties of a directory file are only restored when you copy the directory file itself. To ensure that the directory file is backed up, use the command:

```
copy /example.dir <tape>
```

rather than

```
copy /example/ <tape>
```

or

```
copy ::example <tape>
```

which just copy the files in the directory.

If you are copying from tape a directory file that already exists on disk, a "File already exists" error does not occur (as it does for non-directory files). Rather, the existing directory file is used as it is, and other files from tape may be copied into the existing directory.

In this case, the properties of the existing directory are left unchanged. Directory properties are only restored when the directory is created by the restore process, not when a previously existing directory is used.

Normally, the owner of each restored directory file defaults to the user who does the restore. The original owner of each directory is only restored when a superuser does the restore. This facilitates system backup and restore by a superuser. Ownership is not restored for other users, because that causes inconvenience for the routine transfer of files between users (via tape).

Ownership is never restored for remote files. The user specified in brackets in the DS location becomes the owner of all directory files created remotely.

When TF restores a directory file as a new global directory, it creates the global directory on the same LU from which the directory from tape was saved, whenever possible. This is not applicable for subdirectories, which must always go on the same LU as their parent.

If the original LU is unknown (for example, for directories created automatically or directories restored from UNIX TAR tapes) or unavailable (for example, not mounted or no valid disk LU), the default LU chosen by the file system is used. The file system defaults the LU for creating directories to the same LU as the current working directory, if any. (More information about the default LU is provided in the documentation for the "FmpCreateDir" call in the *RTE-6/VM CI User's Manual*.)

This discussion assumes that the CO command does not specify a destination disk LU. If it does, all global directories (including "automatically" created directories) are created on the specified LU whenever possible. If a CI directory cannot be created on the specified LU, the default LU is used.

- System Backup and Restore

This section concerns system-wide file backup and does not cover physical backup of the disk LU that contains the operating system and boot extension files, which is also necessary. You may use the PSAVE/PRSTR utilities to back up and restore the system disk LU. These utilities are discussed in Chapter 3.

You may use the following TF command to back up all files on the system, excluding FMGR files:

```
co / <tape>
```

The "/" source file mask is an abbreviation for "/@.@," which selects all global directories. Since selecting a directory implies selecting all the files in it, this mask selects all files in global directories and their subdirectories. Various options can be added to the above command as required.

You can back up all files on a particular disk LU with the command:

```
co <disk LU> <tape>
```

Specifying a disk LU is similar to specifying "/" (in that the root directory is the starting point for the copy), except that only a single disk LU is selected.

You may also perform incremental backup of all non-FMGR files on the system. The section in this chapter on "Incremental Backup" provides a definition of the terms used and complete information about the procedure. The commands to use for full backups and delta backups of all non-FMGR files are presented below for your convenience. Note that you may not select incremental backup for a specific disk LU except by specifying all of the global directories on that LU.

For full backups, start a new tape and use this command:

```
co /@.@ <tape> c
```

For delta backups, remount the same tape and use this command, which appends all modified files to the tape:

```
co /@.@.b <tape> ac
```

Other options can be added to these commands as required.

If you use incremental backup, you must back up FMGR files separately.

If you do not use incremental backup, you may issue the following command to back up all files on the system, including FMGR files:

```
co @.@.e <tape>
```

The E qualifier means search everywhere, including FMGR cartridges. Be aware of the restrictions that apply when using TF with FMGR files (see the section on "Using TF with FMGR Files," later in this chapter.)

Do not use incremental system backup if individual users do incremental backups. (Refer to the earlier section in this chapter on "Reusing the Backup Bit.")

The commands shown above for system backup are not restricted to a superuser, but only a superuser will be able to back up read-protected files owned by someone else.

You can restore a system backup tape made by any of the commands discussed above by using the following TF command:

```
co <tape>
```

You need not specify a destination parameter, because all files are to be restored with the same names and to the same directories from which they were backed up.

Normally, all files are restored to the same LUs from which they were backed up. However, the LUs must be mounted before you do the restore. If you don't need the previous contents of the LUs (which is usually the case when you restore a system), you should clear the LUs prior to the restore (using the CI IN command or the FMGR IN command, as appropriate).

A superuser must do the restore in order to re-establish the original owner of each directory restored.

You may put all the global directories created during a restore on a specific disk LU by issuing the following command:

```
co <tape> <disk LU>
```

If you are entering a directory that already exists on another LU, no error is reported. If a directory cannot be created on the specified LU for any other reason, TF tries to create it on the default LU chosen by the file system. Refer to the earlier section in this chapter, "Saving and Restoring File Properties," for more details. No error is reported when a different LU is used.

Handling Disk Full Errors

When a disk full error occurs, TF reports the error and suspends operation. The error message contains the name of the file being copied when the disk became full; this lets you determine the LU that is full. You can free space on the disk using MPACK, FPACK, or the FMGR command PK, then restart TF (using the system GO command). Refer to the description of MPACK or FPACK in Chapter 6 of this manual and to the discussion of PK in the *RTE-6/VM Terminal User's Manual* for details of disk packing.

This error may recur during the restore operation, but as long as the CO command eventually completes normally, no files are corrupted or omitted from the copy. However, if the disk full error cannot be corrected, you must enter a system BR TF, then GO TF to break the utility.

File Access During Backup and Restore Operations

TF opens files individually as it copies them and does not lock disk volumes. Files that are being backed up are opened non-exclusively; files that are being restored are opened exclusively. Files that are being verified on either backup or restore are opened non-exclusively.

You cannot back up or verify a file (in either a backup or restore) that is opened exclusively to another program while the copy is in progress. You cannot replace a duplicate file upon a restore if it is open (exclusively or non-exclusively) to a program. In these cases, TF issues a "File is already open" message and identifies the file. The rest of the backup, restore, or verify is not affected.

It is possible for a program to create a duplicate file while a TF restore is in progress. This duplicate is then purged and replaced with the file from the backup tape.

A program may alter the contents of a file between the time it is copied then opened for verification. In this case, either a "Data doesn't match source file" error occurs on backup, or an "Update times don't match. Not Verifying" error occurs on restore.

Using TF with FMGR Files

You can use TF to back up and restore FMGR files, with the limitations listed below. Problems may arise when using TF to access a file if any of the following apply:

- The name contains a "/"
- The name contains more than one "."
- The name begins or ends with a "."
- The name ends with ".DIR"
- The name contains a ">" or "[" anywhere after the first character
- The CRN contains a "/", ".", ">", or "["

TF cannot handle such file names correctly because of the special significance the file system attaches to these characters. If you try to back up or restore such files (including restoring from FC tapes), you may encounter the following problems:

- The source file descriptor reported in the "Copying ..." message may not agree with the actual source file.
- The destination file may have part of its name dropped.
- The file type may be lost, causing the type to default to 3. For type 1 or 2 files, this usually makes the file data inaccessible. (This can only happen for files on FC tapes with names containing ">" or "[".)
- The file may be mistaken for a directory file, causing its data to be lost. (This can happen with files whose names end in "/" or ".DIR".)

- Part of the name may be interpreted as a subdirectory name (as in A/B::C).
- An “illegal name” error may be reported.

Some of these problems can occur without TF detecting an error. If you use TF for both backup and restore with the Verify option, the chance of error detection increases.

For backup operations, verification prevents any undetected data loss except for files whose names end in “.DIR,” although file names may be altered without warning. Further, any data loss during backup of files with the “.DIR” extension may cause a verify error on restore, so data loss does not go undetected on both backup and restore.

On a restore, verification prevents undetected data except for possible default to file type 3 (FC tapes only), although file names may be modified or misinterpreted without warning.

- When you use the CO command, you may not use the “@” character to specifically identify file names that contain the “@” character. However, you may copy files whose names contain an “@”.
- There is no incremental backup capability for FMGR files.

Using TF with FC Tapes

In general, you can use TF to read tapes written by the FC utility. This capability may be desirable since TF can restore files to directories, while FC is limited to using FMGR cartridges. To make file conversion convenient, use FC to back up FMGR files, then use TF to restore these files to directories.

When you read FC tapes with TF, the following limitations apply:

- All of the restrictions with regard to special characters in file names and CRNs are applicable (see the preceding section, “Using TF with FMGR Files,” for details).
- The TF LH command can only list tape headers for TF tapes. When you use LH with an FC tape, it indicates that the tape is in FC format but does not list the header. Use the FC LH command to list the FC tape header.
- You must use the FC CL (Cartridge List) command to obtain a cartridge list. TF does not include a comparable command.
- TF cannot restore sparse files (files with missing extents).
- TF can restore files from multi-tape FC backups, but you must issue a separate copy command for each tape, and individual files that cross FC tapes cannot be restored. For each tape in a multi-tape set, TF reports an “FC file error <file>” for every file in the entire backup set that is not stored on the tape you are currently restoring. Spurious errors such as “Tape runaway” or “tape i/o error” may be reported at the end of each 9-track tape, except for the last tape in the set.

- TF does not include the special FC features provided for FMGR cartridges, such as automatic cartridge allocation and the pre-restore check, to determine whether the cartridge can hold all the files to be restored. (TF creates global directories, rather than allocating cartridges.)

When you use TF to restore FC from tapes, the error message "FC file error <file>" indicates a problem with the file, which may be caused by any of the following:

- A previously reported tape I/O error that, while not directly associated with a particular file, affects the file reported in the message.
- A tape format error (corrupt tape).
- The file reported in the message is a sparse file.
- A disk error occurred when the file was backed up (the error is recorded on the tape).
- The file is continued across tapes.

It may be possible to get more specific information on any error that occurs by restoring the file with FC instead of TF. Improved error recovery may be possible using the FC I option.

Multi-Tape Backup and Restore

When a file cannot fully fit on a tape, the incomplete portion is removed from the tape and the remaining contents are verified (if you selected the Verify option). TF then prompts you to mount the next tape and continues the copy, beginning with the file that did not fit on the previous tape. If a single file is too large to fit on one tape, use a larger tape or back up the file by some other means.

In a multi-tape backup, each tape in the set bears the same title and date in the tape header. The header also includes a tape number to identify the tapes in order, from tape 2 to the end of the set (the first tape in the set is not assigned a number). The title and date from the first tape are always repeated on subsequent tapes in the set, even when new tapes are appended. You can use the LH command to examine the tape header.

When you restore from a multi-tape backup, TF prompts for each tape in succession, restoring and verifying each tape before prompting for the next. TF issues a message if a tape is mounted out of sequence, and it includes the option to remount in the correct sequence or to skip one or more tapes in the set.

UNIX Compatibility

The tape format that TF writes is fully UNIX compatible. However, there are incompatibilities between FMP files and UNIX files that have nothing to do with the tape format itself. TF provides complete or partial solutions for some of these problems, but in some cases, the only course to take is to avoid the problem.

Multi-tape TF backups can be read by TAR, but not in a single command. You must invoke TAR separately for each tape.

Compatibility problems between UNIX files and FMP files primarily involve the format of the data in files; however, there are significant differences in the way that FMP and UNIX name files. Compatibility issues are discussed in the sections that follow.

File Formats on FMP and UNIX

FMP files consist of a series of records, each of which contains a series of bytes (or characters). UNIX files simply consist of a series of bytes. Bytes are not grouped into records on UNIX files.

For text files, FMP puts one line of text in each record. On UNIX files, since there are no records, a special character called “newline” is used to separate lines of text. Note that this character cannot be contained within a line of text. The UNIX “newline” character is a byte with a decimal value of 10 and is usually referred to as the linefeed character.

If appropriate options are specified, TF converts text files to UNIX format when copying to tape and converts them back when reading tapes. TF converts text files as it copies them by changing the divisions between records into linefeed characters, or vice-versa.

The presence of tabs in text files that originate on a UNIX operating system may require additional conversion. (See the suggestions given below under “Summary of Recommendations.”)

For our purposes, anything other than a text file is considered a binary file. Binary files do not generally require conversion in order to be copied between FMP and UNIX systems. The contents of binary files can be treated as a series of bytes on either system. FMP uses leading and trailing length words to separate records in variable-record-length files (type 3 and above). TF considers these length words to be part of the file data for binary files. (TF uses forced type 1 access for binary files to bypass the record processing normally done by FMP. Since a UNIX operating system provides no way to indicate records in files, the raw data underlying the FMP record structure is copied directly.)

Porting binary files assumes a common interpretation of the data format between programs on the two types of systems. Differences between hardware representations of numeric data must also be taken into account. Be aware that on some non-HP computers, bytes are stored in inverse order within words and double words, so that bytes may have to be reversed before numeric values are interpreted. If variable-record length binary files are ported to or from a UNIX operating system, the programs that access the files on a UNIX system must understand how FMP uses leading and trailing length words to mark records.

For FMP type 1 and 2 files that originated on UNIX, the size of the main and extents may be larger than the file size on a UNIX system. The end of the file is typically padded with undefined data. This may cause problems unless the file data itself includes information that enables the program that reads the file to determine where the data ends. Type 1 and 2 files that originate in the FMP file system normally contain such information and therefore should not cause problems.

Copying Files Between FMP and UNIX

When files are copied between FMP and a UNIX operating system, text files generally need to be converted as described above, whereas binary files are left unchanged. In order to do this correctly, TF must be able to distinguish text files from binary files. How this is done depends upon the situation, as explained below.

Copying from FMP to UNIX-TAR-Compatible TF Tape to a UNIX Operating System

When TF copies files to tape, it assumes that type 4 files are text and files of all other types are binary. If you specify the X (UNIX) option, type 4 files are converted and other file types are left unchanged. However, if you want text files to remain unconverted, thus leaving them in the FMP record format (with leading and trailing length words), you may omit the X option. If you do not specify X, no files are converted.

Text files copied to a UNIX operating system in this way should be treated as binary files if they are copied back to FMP from a UNIX TAR tape. In other words, you should select the N option. See the detailed information below.

Type 4 files copied to tape with the X option are converted by changing record boundaries into linefeed characters (UNIX “newlines”) so that the files are usable as text files on a UNIX operating system. If the files contain any linefeed characters before conversion, the original line feeds will appear the same as the new ones, and both the original and the new ones added by TF are treated as “newlines” on a UNIX operating system. Furthermore, both kinds are changed into record boundaries when read from tape by TF.

The result of copying such a file to tape (with the X option) and back to disk is that the line feeds are deleted and the records are split where the line feeds were. This would seriously corrupt a binary file, since binary files are likely to contain line feeds (bytes with decimal value 10). Therefore, you should not use the X option if any of the type 4 files might contain linefeed characters. If a file that contains line feeds was converted when copying it to tape, you cannot copy it back from tape correctly, regardless of which options you specify.

In the process of converting type 4 files copied to tape with the X option, records longer than 255 characters are truncated. No warning is given when this is done.

Although files other than type 4 may sometimes contain text, especially type 3 files, such files are not converted because they do not always contain text. Any attempt to convert files that do not contain text causes them to be corrupted simply by copying them to tape and back using TF, as explained above. In contrast, type 4 files almost always contain text, so converting them is unlikely to cause problems.

Copying from UNIX-TAR-Compatible TF Tape to FMP

When copying files from TF tapes, TF can tell which files were converted when copying to tape and therefore automatically knows which files to convert back.

Copying from a UNIX Operating System to TAR Format Tape to FMP

Since no distinction is made on UNIX TAR tapes between text and binary files, TF cannot automatically tell which files need to be converted when reading TAR tapes. Because text files are the most typical, TF assumes all files on TAR tapes are text files and converts them accordingly. You can use the N option, which inhibits conversion, to override this default so that binary files can be copied. (Applying the conversion used for text files to a binary file causes the data to be copied incorrectly.)

When TF converts files that are being copied from a TAR tape, lines are split into separate lines after every 256 characters. When this record splitting occurs, it will cause verification errors.

UNIX TAR tapes have no information to indicate the proper FMP file type. When TF copies from TAR tapes, it assumes type 4 unless you specify another file type in the destination parameter. For type 2 files, you must also specify the proper record length. If you do not specify the appropriate file type (and record length, if applicable), FMP may not be able to access the file correctly.

Normally, you should copy files that originated on FMP back to their original file type. File type 1 is usually appropriate for binary files that originated on a UNIX operating system. An understanding of FMP file types may be necessary to determine what file type is appropriate in a given situation. (Refer to the *RTE-6/VM CI User's Manual* for a description of file types.)

To restore files from a UNIX TAR tape that contains mixtures of binary and text files, or mixtures of various file types, you must specify the N option for some files and not for others, and you must specify different file types for different files. You may use the GR command to do this by specifying different destinations and options for each command in the group. Using a standard naming convention to distinguish different kinds of files facilitates this, as shown in the following CO command group:

```
TF: gr
TF: co 8{@.txt}
TF: co 8{@.bin} :::1 n
TF: eg
```

This command uses the default action for files matching @.txt, treating them as text files, converting them accordingly, and copying them into type 4 files. However, files matching @.bin are treated as binary files, are not converted (due to the N option), and are copied into type 1 files (as specified in the destination parameter).

The above example assumes that files on the tape were in the working directory and were named relative to the working directory when they were copied to tape. Otherwise, you must specify a directory or specify the E qualifier (@.txt.e) to indicate that all directories are to be searched.

If you do not use a standard naming convention, you must expand the group to name each file individually. However, if a tape contains files that are mostly of one type, with a relatively small

number of files that are exceptions, it is possible to name only the exceptions at the beginning of the group and then end the group with a single blanket command to cover all the other files.

For example:

```
TF: gr
TF: co 8{firsttype1 secondtype1} :::1 n
TF: co 8{firsttype2 secondtype2 thirdtype2} :::2:200 n
TF: co 8{fourthtype2 fifthtype2} :::2:200 n
TF: co 8
TF: eg
```

This command works because when files are copied from tape, only the first of the matching commands is used, even though a file may match the source mask of more than one CO command.

This means the destination and options used for a particular file are the ones specified in the first command with a matching source mask. Thus, files selected by each CO command in a group are automatically excluded from subsequent commands in the group.

In the grouped command shown above, each file specifically named in the first three commands in the group is copied with the N option and the specified file type of 1 or 2. All the other files are copied into type 4 files (the default) with no options. The final command specifies no mask and thus matches all the files not matched by the previous commands.

You can also use the DE command effectively to save typing long lists of files.

Summary of Copying Recommendations

Copying Files to Tape with TF

In general, you should use the X option when you copy files to tape with TF if the tape is to be read by the UNIX TAR utility. The X option causes type 4 files to be converted to the UNIX text file format.

For files that are to be copied to a UNIX operating system by using TF with the X option, always keep text in type 4 files and never put binary data in type 4 files. That way TF knows which files to convert.

Do not use the X option with type 4 files that may contain linefeed characters, since this may cause the files to be corrupted by the conversion processes used in going to tape and back again. Type 4 files are usually text files, which do not usually contain line feeds, since they are control characters (control-J). Therefore, this is only a problem if a type 4 file contains binary data (likely to contain linefeed characters, which are simply a byte with decimal value 10) rather than text or if it is a text file that actually contains a control-J.

When type 4 files are copied to tape with the X option specified, records longer than 255 characters are truncated without warning.

You may omit the X option when you move files to a UNIX operating system if you want to leave text files in the FMP record format.

Copying Files from Tape with TF

When you copy text files from UNIX TAR tapes, note that text files from a UNIX operating system frequently contain tabs, but RTE does not support the use of tabs in text files. Files that contain tabs cannot be compiled or printed. However, you may use EDIT/1000 to expand tabs into spaces. Use the EDIT/1000 T command to set tab stops and turn screen mode display functions off (SE DF OF). Then go through the entire file in screen mode, reading each screen and going on to the next using the control-N command (control-Q after the last screen).

When you copy from a UNIX TAR tape, you must specify the N option for binary files to inhibit the conversion process, which is only needed for text files. Specifying the N option for text files, or omitting it for binary files, causes incorrect data in the resulting files. For tapes that contain a mixture of text and binary files, you may specify N for some files but not others by using grouped CO commands.

Text files stored on a UNIX operating system in FMP format are like binary files and need not be converted when they are copied back to FMP. Therefore, you should specify the N option for these files as well. (Such files may result from copying text files to a UNIX operating system without specifying the X option.)

When you copy files from a UNIX TAR tape, the file type defaults to 4. If the files are not to be copied into type 4 files, you must specify the appropriate file type in the destination parameter, to ensure that the files are accessed correctly by FMP. For type 2 files, you must also specify the record length. If several file types are mixed on a single tape, you may use a grouped CO command to specify the different types.

When you copy files from a UNIX TAR tape without using the N option, lines are split into separate lines after every 256 characters. No warning is given when this is done.

When you copy binary files, whether from FMP to a UNIX operating system or from UNIX to FMP, compatibility of the data format must not be taken for granted. The various issues concerning binary files discussed in the section “File Formats on FMP and UNIX” in this chapter should be considered.

Directory File Names

All FMP directories have a type extension of .DIR, whereas a UNIX operating system imposes no such restriction and generally uses ordinary names (with no dots) for directories. Therefore, TF removes the .DIR from names of directory files when copying to tape and appends .DIR to directory file names when reading from tape.

Thus, the files EXAMPLE.DIR and EXAMPLE are both called EXAMPLE when read from a TF tape to a UNIX system, which causes problems.

File name Letter Case

A UNIX operating system allows both uppercase and lowercase letters in file names and treats the two cases differently. FMP allows you to specify names in either case, but uses only uppercase on disk. Thus, the file names ‘myfile’, ‘MYFILE’, and ‘MyFile’ refer to three distinct files on a UNIX system; but on RTE, all three refer to the same file.

When you copy from a UNIX TAR tape to an FMP disk, all file names are converted to uppercase. However, if the tape contains multiple files that have the same name except for letter case, this conversion causes duplicate names, and thus, only the last of the multiple files is restored to FMP. The message "Replacing <filename> (created during this copy command)" displays for each duplicate file name.

Because file names in a UNIX system are generally lowercase, TF converts all file names to lowercase when copying to tape. This is transparent to RTE since TF converts file names to uppercase when reading tapes. However, copying files from a UNIX operating system to FMP and back again to UNIX can cause problems, since the effect in this case is to convert all file names to lowercase.

Dots Used in File Names

FMP file names use a dot (.) to separate the file name from the type extension. No other use of a dot is allowed. A UNIX operating system, however, treats the dot as any character in a file name. When TF encounters a dot in an unexpected position in a file name or path name, it converts the dot to an underscore (_). This includes any of the following placements:

- A dot at the beginning of a name
- A dot at the end of a name
- A dot in a directory name
- A dot that precedes other dots (for example, the first dot in file a.b.c)

The following shows the resulting UNIX path names when dots are converted to underscores and the implicit .DIR type extension for directory files is inserted. Note that TF does not recognize the special names "." and ".." used by a UNIX operating system.

Before	After
/aaa/bbb.ccc	aaa/bbb.ccc
/aaa/bbb.ccc (directory file)	aaa/bbb_ccc.dir
/aaa/bbb.ccc/ddd	aaa/bbb_ccc/ddd
aaa/bbb/.ccc	aaa/bbb/_ccc
aaa/bbb/ccc.	aaa/bbb/ccc_
aaa/bbb.ccc.ddd	aaa/bbb_ccc.ddd
aaa/bbb.ccc.ddd (directory file)	aaa/bbb_ccc_ddd.dir
.aaa	_aaa
./aaa	_/aaa
bbb/.. (directory file)	bbb/__.dir
bbb/../aaa	bbb/__/aaa

UNIX users who do a "tar cv ." get files on the TAR tape whose names begin with "./". If you restore such tapes with TF, the command

```
tf co <tape>
```

creates a directory called "_" in the working directory and copies all the files into that directory (TF changes "./example" to "/example"). To avoid this problem, use "tar cv *" rather than "tar cv," or use the following command instead of the one above:

```
tf co <tape>{_/@.@} @.@
```

Special Characters

File names that contain blanks or the characters "[", ">", and ":" cannot be handled correctly by FMP because they are used as delimiters.

The "@" character can cause problems in some contexts since it is the FMP wildcard character. (In a UNIX operating system, the "*" is the wildcard character.)

File Name Length—Moving Files from FMP to UNIX

FMP files can have up to 21 characters: 16-character names and a dot followed by a 4-character type extension. UNIX file names are limited to a total of 14 characters. Therefore, FMP files that have more than 14 characters (including the name, dot, and type extension) cannot be moved to a UNIX operating system. What happens if you attempt to do so is a function of the TAR utility. (Some newer UNIX versions do allow file names greater than 14 characters.)

File Name Length—Moving Files from UNIX to FMP

UNIX file names may contain a dot followed by more than four characters. Since FMP interprets the last dot as the type extension delimiter for the file name, UNIX file names are truncated to four characters following the last dot. No warning is given when this occurs.

TAR allows path names of up to 99 characters; FMP allows only 63 characters in a path name. If a path name from a TAR tape is longer than 63 characters (after any required conversion), TF returns the message

```
Name too long.  Skipping <filename>.
```

Linked Files

Linked files are not supported by FMP and cannot be restored.

Appending to Tapes

Tapes created by one utility should not be appended using the other utility, since TF and TAR do not append to tape in the same way.

Files Types, Security Codes

A UNIX operating system does not recognize file types or security codes, which are lost when FMP files are moved to a UNIX system. Files moved from UNIX to FMP default to type 4, security code 0.

Time Stamps

UNIX TAR tapes do not have a create time for each file. (Refer to the earlier section on "Time Stamps" for details.)

Root Directory

The FMP root directory may only contain directory files; the UNIX root directory may also contain ordinary files. You can restore ordinary files from the root directory by specifying a directory other than the root in the destination parameter.

When you select these files from a UNIX TAR tape, you must specify the type extension or use a wildcard. Otherwise, TF assumes a DIR type extension and just selects the directory files from the root directory.

Header and Final Checksum Dummy Files

TF includes a header file at the beginning of each tape and a dummy checksum file after the last file on the tape. These are special files that are not restored. To TAR, however, these are ordinary files and they are extracted with the rest of the files.

The tape title is used as the path name for the header file. If you do not specify a title for the tape header, the default title (the CO command itself) will contain embedded spaces and could contain slashes. This could result in extraneous files being created by TAR.

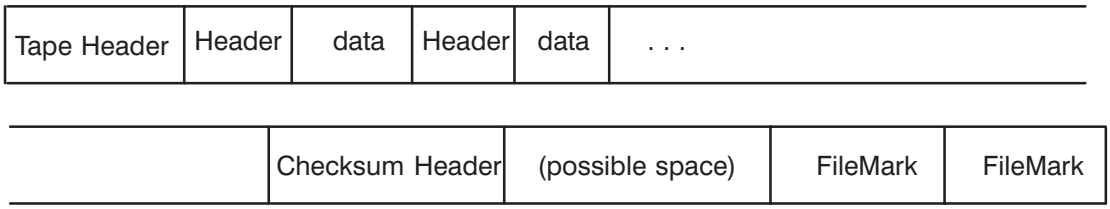
The checksum file always has the path name ".CHECKSUM."

Since neither the header nor the checksum file contains any data, the use of extra disk space is not a problem.

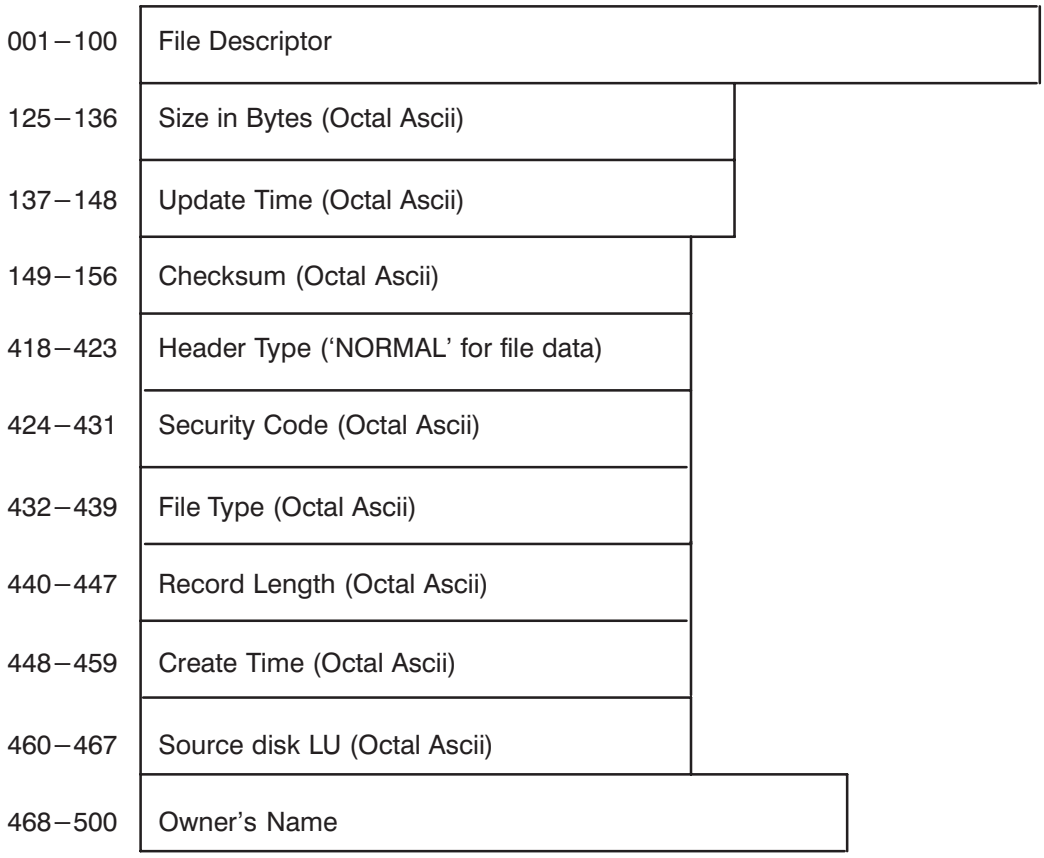
TF Tape Format

The TF tape format and header basics are shown below.

TF Format:



TF Header Basics (in bytes):



Installing TF

Load the TF utility using the LINK command file #TF.

The global directory ::SCRATCH should be created during system installation and put on an LU with a lot of free space. This ensures trouble-free operation of TF and other system utilities that use the ::SCRATCH directory for scratch files.

The file >TF000 must be located in one of two places. If the global directory CATALOGS exists, place >TF000 within it. If /CATALOGS does not exist, you can place >TF000 on an FMGR system cartridge; however, if global directory CATALOGS is later created, you must move >TF000 to /CATALOGS for TF to run.

You may want to increase TF's EMA size when you install TF. EMA size should be increased if you want to use the TF GR command to group more than a few CO commands. If the EMA size is not large enough for all the CO commands in a group, TF displays the following message and exits:

```
EMA size of TF not large enough for a group of this size.
```

Each additional page of EMA lets you include at least 20, and possibly as many as 100, more CO commands in a group, depending upon the complexity of the individual commands.

You can increase the EMA size when TF is loaded by increasing the number of pages specified in the EM command in the LINK command file, #TF. Alternatively, you can use LINK to relink an existing TF program file, as:

```
CI> link
link: lk tf::programs
link: em 30
link: en
```

File Copy (FC)

FC lets you copy FMGR files between disk cartridges and tape media, either disk-to-disk, disk-to-tape, or tape-to-disk. Tape-to-tape copying is not supported. FC does not support LUs greater than 63. You can copy the files to/from a 1600-bpi magnetic tape or a CS/80 cartridge tape drive.

You may give a file being copied a different file name, security code, or cartridge by specifying those fields of the destination name. When you copy from disk, file extents are automatically gathered and copied in ascending order following the main extent. Optionally, you may eliminate all extents and copy all sections of a file to the main extent.

You may also select command options that purge source disk files after copying, list or suppress the listing of files copied, replace duplicate files with the last duplicate copied, or verify the copy. (The Verify option is selected in addition to the tape checksums, which are always used.)

Calling FC

You may run FC from a terminal or from a control file. Any single FC command can be entered in the runstring. When you run FC from a control file, the FC transfer (TR) command lets you specify a control file as the FC utility command source.

Interactive mode is entered by invoking FC with no runstring. Commands are entered at the FC: prompt. FC executes each command before prompting for the next one, continuing until you enter the EXIT command. For example:

```
CI> RU,FC
FC.nn: <command>
FC.nn: <command>
.
.
.
FC.nn: ex
:
```

The .nn in the utility prompt identifies your FMGR session. You may enter just the first two characters or up to the whole name of a command; for example, EX, EXI, and EXIT all cause FC to exit. Uppercase and lowercase are acceptable. You may enter a comment at the end of a command string or by entering an asterisk (*) as the first character in the response. If you enter a comment at the end of a command string and do not use all the available optional parameters for the command, you must insert commas at the end of the string as placeholders for the omitted parameters.

When a command is entered in the runstring, as:

```
CI> RU,FC, <command><parameters>
```

FC executes the command and terminates.

To run FC from a control file, enter the TR command in the runstring:

```
CI> RU,FC,TR <control file>
```

The control file parameter can be a device LU or a file name. FC executes each of the commands contained in the control file, then terminates.

FC Commands

FC commands let you configure the copy operation and specify names for the formatted tape files and comment files, selectively copy files and group copy commands, transfer to and return from command files, and direct listings to a log or list device.

Configuration commands set attributes that affect the execution of other FC commands. They let you select the list device, enable/disable command echoing to the list device, set the title and comment file for tapes subsequently written, and set the scratch file disk cartridge.

The Copy (CO) command initiates the copy operation from disk-to-tape, tape to disk, or disk-to-disk. Cartridges and tapes can be written and restored by naming the devices as the source and destination parameters. Files being copied can be selected by name, security code, cartridge, and so on, as desired. Selection by name can make use of wildcard characters.

In copying to or from tapes, there may be applications that require specifying multiple source/destination parameters in a single copy operation. This is true, for example, if you are copying files to a tape and renaming them, or if tape files are to be copied to disks other than those specified in the tape directory. Grouping copy commands lets you combine multiple copy commands into a single copy operation.

Listing commands direct listings of the FMP cartridge list, global FMP cartridge list, tape directory list, and the tape comment and header files to your terminal, a selected list device, or a file.

Transfer and exit commands let you transfer control to and from a control file and stop FC operations.

Table 4-8 summarizes the commands, which are described in the following sections.

Table 4-8. FC Commands Summary

Commands		Description
Information Command		
?		Provides a summary of commands and command syntaxes
Configuration Commands		
CF Name Comment File	namr	Establish namr for tape comment file
ECHO	ON,OFF	Turn echoing of commands to list device ON or OFF
LL Set list device	namr	Set list device or file (destination for listed messages and commands
SCratch Files	cart	Specify disk cartridge to be used for internal scratch files
TITLE	title	Establish title to be used in tape header file
Copy and Related Commands		
COPY	srce	Copy files as specified by parameters
DEFAULT	srce	Set default source, destination, and options for subsequent COPY commands
GROUP CO Commands		Used for grouping more than one copy command
AG Abort Group		Terminates grouped copy commands execution
EG End Group		Causes execution of the copy commands between the preceding GR command and this EG command.
Listing Commands		
CL Cartridge List	−tlu	List FMP cartridge list or cartridges included on tape
DL Directory List	srce	Compile directory list of FC tape
LC List Comment File	−tlu	List comment file from FC tape
LH List Header File	−tlu	List header file from FC tape
Transfer and Exit Commands		
ABort		Abort FC, including any active group copy
EXit		Exit FC. (If a group copy is active, it is processed before FC is aborted.)
TRansfer	file	Transfer to/from FC command file
Comment Command		
*		Identifies following string as comment line

Command Summary Function (?)

Purpose: Writes the FC command summary to the terminal.

Syntax: `?`

Description:

When you enter ? at the prompt, the FC command summary is written to the terminal.

If you enter ? together with another command or with an option (for example, ?,SCRATCH), FC displays additional information about the specified command or option.

Abort (AB)

Purpose: Stops FC execution immediately.

Syntax: `AB`

Name Comment File (CF)

Purpose: Specifies the name of an optional comment file to be copied from disk to tapes created by subsequent copy commands.

Syntax: `CF,namr`

namr The file name of the comment file.

Description:

When the copy is to magnetic tape, the file is copied uncoded. Because CTD tapes are formatted in fixed length blocks, files must be coded for copy to CTD devices.

Comment file records should not end in a backslash. A comment file record longer than 128 words is truncated. Note that a comment file should not contain zero length records, or a checksum error may result when you list the file with the LC command.

Since the comment file is not encoded on magnetic tapes, you can read the file using the FMGR ST, LI, or DU commands.

Cartridge List (CL)(CLAL)

Purpose: CL displays the list of all cartridges you can access. CLAL displays the list of all cartridges in the system.

Syntax: `CL,[-tlu][,K]`

−tlu lists the cartridge files stored on the tape. For example:

```
title: NNNNN
volume: X
date and time of creation: hh:mm ddd., mmmm, yyyy
created under account name: NNNNNN.NNNN

LU CRN LABEL P/G/S USER.GROUP
xx xxx xxxxx   x   xxxx.xxxx
```

K Keeps the tape online.

Description:

Only the CL command can be used with a tape source. The CLAL command is not defined for use with tape.

If the cartridge was renamed as part of the copy operation, only the cartridge CRN is listed.

You can specify the K option to keep the tape drive online under all circumstances. Note that while the tape is online, it may be overwritten by another program, since it is unlocked.

You may use the LL command to direct the CL listings to another list device.

Copy (CO)

Purpose: Initiates the copy operation from disk-to-tape, tape-to-disk, or disk-to-disk.

Warning **Do not interrupt the tape copy process. An attempt to take the tape unit offline or to dismount the tape will destroy the tape's contents. Wait until the "Cleaning Up" message and your prompt appear before touching the tape unit.**

Syntax: `CO[,srce[,dest[,optns[,file1[,file2[,msc]]]]]]`

srce — The file or files to be copied. Srce can be a single namr, a list of namrs enclosed in braces, or a negative tape LU with an optional namr or list of namrs enclosed in braces. Each namr may take the abbreviated form of a CRN or negative disk LU. Wildcard characters are acceptable in srce namrs.

If the srce is a single namr, you can omit the braces; if, however, the srce is a list of namrs, you must enclose the list in braces, as:

```
namr                    *single-namr srce
{namr,namr...,namr}     *list of srce namrs
```

If the source is a tape LU, you may omit the braces. If, however, it contains one or more optional namrs, you must enclose the single namr or list of namrs with braces, as:

```
-tlu
-tlu{namr}
-tlu{namr,namr...,namr}
```

dest — Determines whether the files selected by the source parameter are being copied to disk or to tape, and allows specification of the name, security code, and cartridge for the destination files. The dest can be either a namr (for disk destinations) or a negative tape LU with an optional namr enclosed in braces. The namr may take the abbreviated form of a CRN or negative disk LU. The dest takes one of the forms:

```
namr
-tlu
-tlu{namr}
```

You may specify just one destination and may not use wildcard characters in dest namrs. (Refer to the section later in this chapter on "Group CO Commands" for a method to specify multiple destinations.) If you do not specify the name, security code, or cartridge in the destination namr, that attribute of the destination is the same as on the source file.

optns — The options are described in the "CO Command Options" section.

file1 — The optional limits on the range of files to be copied from the

file2 — source. Within the range specified, files that conform to the source parameter are selected for copying to the destination. Note that disk resident files are referenced by namr and tape resident files are referenced by disk file reference number. (Refer to the DL command.) If only file1 is specified, the range is interpreted as falling between file1 and the last file on the source medium. If only file2 is specified, the range is interpreted as all files between the first file on the source medium and file2.

If file1 is specified but not found, no files are copied. If file2 is specified but not found, all conforming files between the range of file1 and the end of the source medium are selected for copying to the destination.

If you specify multiple disk cartridges in the source parameter, the file1, file2 parameters are applied separately to each cartridge; for tape sources, the file1 and file2 parameters apply to the entire tape or set of tapes.

msc — The system master security code. Msc enables you to specify a security code in the destination even if you do not specify a security code in the source parameters. This parameter also allows you to purge files using the P and D options even if the file security codes do not match. The msc is also required if you specify the C or ! options. (Refer to the section in this chapter on "CO Command Options" for a detailed description of the available options.)

Description:

You can write and restore cartridges and tapes by naming the devices as the source and destination parameters. Select files to be copied by name, security code, cartridge, and so on, as desired. Selection by name can make use of wildcard characters.

CO Command Source and Destination Parameter Considerations

When you form the source and destination parameters, consider the following characteristics of FC:

- A null security code field in the source parameter matches files regardless of the security code. A zero in the security field matches only files with a security code of zero (unprotected files). Note that in FMGR, unprotected files are created by specifying a zero security code or by omitting the security code from the namr.
- For disk sources, a null cartridge field in the source namr is interpreted to mean the first cartridge that contains a file that matches the source file namr (including wildcard characters). You cannot use a given namr to select files from more than one cartridge.
- For tape sources, a null cartridge field in the source namr is interpreted to mean all cartridges on the tape.

- All files selected by the source and the file1 and file2 parameters are copied, except for type 0 files and files with “illegal names”. Files with illegal names include scratch files with numeric names.
- In the destination parameter, you may specify a file security code field only if you specify the security code in the source namr or if you specify the msc.
- When the specified destination cartridge differs from the one in the source, the copy operation is as follows:

  disk-to-disk: The file is copied to the cartridge named in the destination parameter.

  disk-to-tape: The file is labeled on the tape with the cartridge named in the destination parameter.

  tape-to-disk: The file is copied to the cartridge named in the destination parameter.
- When you copy to tape, if the destination cartridge that will be listed in the tape directory is specified by the destination parameter, it must be specified as CRN, not −LU.
- Namrs specified with only one- or two-character file names (all other fields empty) cannot be distinguished from ASCII CRNs written in the abbreviated form (CRN only, rather than ::CRN). When applicable, the FMP cartridge list is searched first and, if a match is found, the namr is interpreted to be a CRN. If no match is found, the namr is interpreted as a file name. In some situations, it is impossible to recognize a CRN as such if it is given in the abbreviated form:

  1. If a CRN on a tape is specified in the source or destination parameter.
  2. If the destination parameter specifies a disk CRN that is to be automatically allocated (it does not yet exist on the cartridge list).

  In both of these cases, the CRN must be explicitly specified as ::CRN.

CO Command Options

You may specify any of the following options with the CO command, in any order. Spaces in the option string are ignored. Table 4-9 summarizes the CO command options, followed by a discussion of each option.

Table 4-9. FC CO Command Options Summary

Options	Description
Brief, **F**ull Status	Displays status of namrs as each file is copied, when an error occurs (B) or for successfully copied files as well (F)
Clear Destination Disk	Clears destination disk before the copy begins
Duplicate Files	Purges original file and replaces it with the duplicate
Eliminate Extents	Combines all extents into the main extent
Ignore Data Errors	Overrides checksum or verify errors and allows the file to be copied
Keep Tape Online	Keeps tape unit online in all circumstances
Lock, **O**pen	Locks a cartridge or opens individual files for the copy
Purge Source File	Purges a source file after it is copied
Single Volume Copy	Lets you mount a single tape volume set
Tape Length Display	Displays required tape length for copying specified files
Unused Space	Lets FC use more space on the destination medium than required for files being copied, to speed up copying
Verifying Data	Reads data and compares checksums to verify that data is correct

Brief, Full Status Display Format (B,F)

You can have the source namr (and sometimes the destination namr) displayed as each file is copied, if desired. If you select Brief, the namrs are displayed only when an error occurs, to identify the affected file. When you select Full, the namrs are displayed for successfully copied files as well as those with errors. (Refer to the "Error Messages" section later in this chapter for the format of the displayed namrs.) Brief mode is the default when you select files only by tape LU and/or disk cartridge; Full mode is the default in all other cases.

Clear Destination Disk (C,!)

When you select this option, the destination disk is cleared before the copy begins. Note that the boot extension file BOOTEX is not cleared if it is the first entry in the disk directory. The Clear option is permitted only if the destination is a disk cartridge and the system master security code (msc) is correctly specified in the FC command string. During the session, any value in the msc parameter is accepted for clearing private and group cartridges. The true system master security code is required only for system and non-session cartridges.

Before clearing the disk, FC issues the message:

```
Do you really want to purge disk LU nn, CRN nnn ?
```

and waits for a Yes or No response.

The ! specification works the same as the C option except that the above message is suppressed.

Replace Duplicate Files (D)

Normally, when a duplicate file is encountered in a copy, a duplicate name error (FMGR-002) is issued and the duplicate file is not copied. However, if you select the Duplicate option, when a duplicate file is encountered, the original file is purged and replaced with the duplicate.

Typically the replacement does not take place unless the security code of the duplicate destination file matches that of the destination file to be replaced (according to the standard FMP definition of matching security codes). However, if the system master security code is correctly specified in a CO command that includes the D option, file security code rules are superseded, and all duplicates are replaced regardless of whether the file security codes match. (The security code of the new destination file comes from the source file except when this is overridden by a security code specified in the destination parameter.)

Duplicate names can occur when more than one source cartridge is copied to a single destination cartridge. In such cases, the file from the first source cartridge normally is copied successfully, and the rest are not copied because of duplicate name errors. If the D option is selected, each successive duplicate replaces the previous one, so that only the last duplicate file is copied.

Eliminate Extents (E)

This option combines all file extents into the main extent. Unused blocks are not truncated. Extents are not eliminated in files that have missing extents (“sparse” files) or if the resulting main extent would exceed 16383 blocks. In these cases, a warning is issued. If subsequent changes to a file result in the creation of a new extent, the extent created is the same block size as the main extent.

Ignore Data Errors (I)

In general, a file is not copied if a checksum or verify error occurs. The I option overrides this feature on tape-to-disk copies, and the file (including data for which checksum errors were detected) is copied to the destination. Files that are copied with errors are not identified as such in the directory entry for the file; however, a message is issued that defines the range of bad blocks copied. The I option is applicable only for tape-to-disk copy operations.

Keep Tape Online (K)

This option keeps the tape unit online in all circumstances. While this option is a convenience, it makes it possible for another program to overwrite the tape, since the tape unit remains unlocked and online. (Refer to the “Tape Handling” section in this chapter for more details.)

Cartridge Lock, Open (L,O)

You can copy files to/from cartridges either by locking the cartridge (Lock mode) or by opening individual files for the copy (Open mode). Lock mode is generally faster, but excludes other programs from accessing the cartridge. (Refer to the "Performance Considerations" section in this chapter for details.) Lock mode cannot be used if other programs have files open on the cartridge, and should not be used if other programs may need access to files on the cartridge during the copy. Once a Lock mode copy is started, other programs cannot open files on the cartridge for the duration of the copy.

In Open mode, source files are opened non-exclusively, and destination files are opened exclusively. Therefore, even the Open mode restricts access to files, although the restrictions are less severe than those in Lock mode. If neither L or O mode is specified, the defaults are as follows:

- If the source name (or names) contains wildcards or is null, or if more than one copy operation is grouped, a cartridge lock is attempted. If it is successful, Lock mode is used.
- If the lock is rejected, or if there are no wildcard or null source names and the GR command is not used, Open mode is selected.

Purge Source File (P)

When you select the P option, FC purges the source file after it is successfully copied to the destination. To purge a file with a non-zero security code, you must correctly specify either the security code field of the source parameter, or the msc parameter; otherwise, the file is not purged. (This does not, however, prevent the file from being copied.)

If the destination is a tape, the source files are purged after all files are copied. Thus, in the event of an error, or if you abort the disk-to-tape copy while it is in process (using the break command), the source files are not purged.

Copy Single Volume of Multi-Volume Tape Set (S)

Usually, when you copy from a multiple volume tape set, all volumes must be read in sequence. When you select the Single option, this feature is suppressed, and a single tape volume set can be mounted. Note that you cannot copy files that cross volumes when you use the S option.

Display Required Tape Length (T)

When you select this option, FC reads the disk directory, calculates the amount of tape required to copy the specified files, and issues the appropriate message.

Recover Unused Space (U)

In order to copy data faster, FC may use more space on the destination medium than is required for the files being copied. This can only happen on disk-to-tape copies and lock mode disk-to-disk copies. Although this does not typically result in a significant waste of space, specifying the U option prevents any extra space from being used.

Verify Transferred Data Integrity (V)

The V option specifies the following data verification operations:

Disk Read	Data is read twice and the checksums are compared to verify the read.
Disk Write	Data is written to the disk and then immediately read to a second buffer. The two data buffers are compared to verify the write.
Tape Read	The checksum is calculated for each buffer read from the tape and compared to the checksum read with the data. (This tape read verification is performed even if the V option is not selected.)
Tape Write	The tape is rewound after each volume is written, and a tape read verification performed as above.

CO Command Examples

The following examples show how to use CO to copy files between disks and tapes.

Example 1: Copy cartridge 10 to tape LU 8.

```
co,10,-8
```

Example 2: Copy all files beginning with & on cartridge 10 to tape LU 8.

```
co,&-----::10,-8
```

The −−−−− characters in the name field define the wildcard characters. The null security code matches any file security code.

Example 3: Copy three files to cartridge 20.

```
co,{filea,fileb,filec},20
```

Example 4: Copy file1 from cartridge 30 and file2 from cartridge 40 to tape LU 8, and verify the data transfer.

```
co,{file1::30,file2::40},-8,v
```

Any files existing on LU 8 are overwritten as the specified files are copied.

Example 5: Copy all type 4 and type 5 files from cartridge 10 to tape LU 8.

```
co,{::10:4,::10:5},-8
```

Example 6: Copy all files that begin with % and & from cartridge 10 to tape LU 8.

```
co,{%-----::10,&-----::10},-8
```

Example 7: Restore all files from tape LU 8 to their original cartridges.

```
co,-8
```

Since the destination parameter is omitted, the source namrs (from the tape) are used for the destination files. The tape directory file identifies the cartridge from which each file was copied to the tape.

Example 8: Restore all files from tape LU 8 to cartridge 10.

```
co,-8,10
```

The tape directory information identifying the cartridge of origin is ignored, and all files are copied to cartridge 10.

Example 9: Restore from tape LU 8 only those files that were copied to the tape from CRN 10.

```
co,-8{::10}
```

Because no destination cartridge is specified in the command, the files are restored to the same CRN.

Example 10: Restore from tape those files that were copied to tape from CRNs 10 and 20.

```
co,-8{::10,::20}
```

Because no destination cartridge is specified in the command, the files are restored to their original CRNs.

Example 11: Copy file1 to file2 on the same cartridge.

```
co, file1,file2
```

Example 12: Copy file A on cartridge 10 to cartridge 20, and name the file "B".

```
co,A::10,B::20
```

Example 13: Copy file A to the same cartridge, name it file B, and change the security code from 2 to 3.

```
co,A:2,B:3
```

Example 14: Copy all files within the range of file2 and file6 from cartridge 10 to cartridge 20.

```
co,::10,::20,,file2,file6
```

If file2 is not found, no files are copied. If file6 is not found, all files in the range of file2 to the end of the source medium are copied. In specifying the file1,file2 parameters, the files must be specified in ascending order of occurrence on the medium. In other words, no files are copied if file2 precedes file6 on the cartridge when you issue the following command:

```
co,-8{fileA},::20,,file6,file2
```

Example 15: Copy all files beginning with % within the range of fileA and %file6 from cartridge 10 to cartridge 20.

```
co,%-----::10,20,,fileA,%file6
```

Example 16: This example is the same as the one above, except that the source is a tape −LU and therefore, the range of files must be specified as file reference numbers obtained from the DL command.

```
co,-8{%-----},20,,2,6
```

Example 17: Copy all files from file 1 through file 5 from tape LU 8 to cartridge 10.

```
co,-8,::10,,,5
```

Default (DE)

Purpose: Used with the CO command to set default values for the source, destination, and/or options parameters.

Syntax: `DE[,srce[,dest[,optns]]]`

srce, dest, optn	Defined for the CO command, except that a list of namrs may not be used in the srce parameter.

Description:

If you do not enter source and destination parameters in the CO command, the values you specified for those fields with the DE command are used. You can override the DE command values by specifying the source and/or destination parameters.

When you include options in the DE command, they are appended to any options specified in succeeding CO commands. For example:

```
DE,,,epv
co,<srce>,<dest>cde
co,<srce>,<dest>d
```

In this command, C, D, E, P, and V are specified for the first CO command, and D, E, P, and V are for the second CO command. If you specify conflicting options with DE and CO, the CO option overrides the one selected with DE.

Each DE command supersedes the preceding DE specification; therefore, you can cancel defaults by entering a DE command with all null fields.

Directory List (DL)

Purpose: Provides a directory list of files on an FC tape specified by the command. (For directory listings of disk cartridges, use the FMGR DL command.)

Syntax: `DL,srce[,msc[,optn]]`

srce — The source FC tape device, with optional namr, that specifies the files to be included in the list. If you omit the namr, all files are listed. The source can only be a tape; use the FMGR DL command for listing cartridge files. If you specify a namr with the negative tape LU, you must enclose the optional namr within braces, as:

```
-tlu {namr}
```

msc — The system master security code. When the msc is specified, the file security code is contained in the directory list.

optn — One or more of the following options:

- F — Selects Full option, which results in including the extent size, record size, and # extents columns in the directory list.
- S — Selects Single option, which compiles a directory of only one volume of a multi-volume tape set.
- K — Keeps the tape unit online under all circumstances. Note that while the tape is online it can be overwritten by another program, since the online tape is unlocked.

Description:

You can direct the DL listing to another list device by using the LL command. The tape directory list format is shown in the following example.

```
title: SAMPLE OF TAPE DIRECTORY FORMAT
volume: 2  date and time of creation: 7:09 PM TUE., 14 JULY, 1992
created under account name: KURT.SYSTEM

                              extent  record    #extents   security  diskfile
name    crn     LU   type       size    size    log     phy   code      ref #

JLMHI   100     25      1        1       128     1       1      HP        1
*       100     25      4        1               1       1                2 '
DVA32   100     25      4       11               1       1                3
*************************************************************************
mounted volume starts here
*************************************************************************
'DVM33  100     25      4       92               3       3       HP       4 CONT
*************************************************************************
mounted volume ends here

*************************************************************************
'DVM33  100     25      4       92               3       3       HP       4 CONT
'CM80L  100     25      4       50               1       1                5
**      100     25      4       1                1       1                6
'VERIF  100     25      4       24               1       1                7
'PBERS  100     25      4       234              1       1                8
'BKUP   100     25      4       55               1       1                9
&BOOTC  100     25      4       18               1       1                10
/BIGL   100     25      4       2                1       1                11
'DISK   100     25      4       7                1       1                12
'R      100     25      4       3                1       1                13
'NUMS   100     25      4       41               1       1                14
-------------------------------------------------------------------------
```

NOTES:
1. title – specified using TI command
2. crn, LU – appear only if multiple cartridges are on a tape
3. extent size, record size – appear only if F option specified
4. # extents – appears only if F option specified:
   log = logical # of extents = last extent + 1
   phy = physical # of extents = total extents allocated to file
5. security code – appears only if msc specified correctly
6. “mounted volume starts/ends here” messages – appear only in multi-volume directory lists
7. CONT – appears if file is continued on next volume or is continued from previous volume

Figure 4-2 . Example of Tape Directory List Format

Echo Command (EC)

Purpose: To echo commands to the list device as the command is processed, or to suppress echoing.

Syntax:
```
EC,ON
```
or
```
EC,OFF
```

Description:

Initially, command echoing to the list device is OFF. The EC command specifies that each command is to be echoed to the list device as the command is processed and, subsequently, to suppress echoing if desired. The default is ON if neither ON or OFF is specified.

Exit (EX)

Purpose: Exits FC. If a group copy is active, it is processed before FC terminates.

Syntax:
```
EX
```

Group Copy Commands (GR, EG, and AG)

Purpose: To combine multiple copy commands into a single operation.

Syntax:
```
gr
co,<parameters>
co,<parameters>
.
.
.
co,<parameters>
eg
```

Description:

When FC encounters the GR command, it keeps track of all subsequent CO commands but does not execute them immediately. Instead, FC executes them as a single operation when it encounters the EG (End Group) command. You can abort the GR operation by using the AG (Abort Group) command, which causes the group to be terminated before any of the CO commands are executed.

If the grouped CO commands have a common source or destination, you may use the DE command to your advantage. See the following example.

```
DE,-8              (Common source tape −LU)
gr
co,::10,40         (Restore CRN 10 from tape to CRN 40)
co,::20,50         (Restore CRN 20 from tape to CRN 50)
co,::30,30         (Restore CRN 30 from tape to itself)
eg                 (End group)
DE                 (Cancel default)
DE,,-8             (Common destination tape −8)
gr
co, abcd, wxyz     (Copy file ABCD to −8 as WXYZ)
co,efghij, qrstu   (Copy file EFGHIJ to −8 as QRSTU)
EG                 (End group)
```

Only a fixed number of files can be group copied from a given LU. The number that can be group copied, approximately 150 to 180, is dependent on the amount of free memory that exists beyond the end of FC's code. If the number of files is exceeded, an out of memory error will occur.

FMGR Error Help Function (HE)

Purpose: You can call HELP, without exiting FC, to define any FMGR errors encountered during execution of the utility.

Syntax: `HE[,key[,LU]]`

Description:

The following example illustrates the use of the HElp function.

```
:
FMGR-032
FC.nn: HE
FMGR-032
CARTRIDGE NOT FOUND
AN ATTEMPT WAS MADE TO ACCESS A CARTRIDGE THAT CANNOT BE
FOUND IN THE CARTRIDGE LIST.  CHECK THE CARTRIDGE NUMBER
FOR CORRECTNESS
FC.nn:
```

List Comment, List Header Files (LC, LH)

Purpose: The LC and LH commands let you list the comment file (LC) and the ASCII header file (LH) to the list device.

Syntax:

```
LC,-tlu[,K]
```

and

```
LH,-tlu[,K]
```

Description:

You may use the LL command to direct the file listing to another list device.

You can specify the K option to keep the tape drive online under all circumstances. Note that while the tape is online, it can be overwritten by another program, since the online tape is unlocked.

List Device (LL)

Purpose: Lets you specify a device or file other than the terminal to receive list information.

Syntax:

```
LL,namr
```

or

```
LL,-
```

namr	The namr of the device or file to receive the list information.
–	Resets the list device to the terminal—used if a preceding LL command named another list device.

Description:

Initially, the list device for FC is the terminal. If you specify a file that does not already exist, it is created. Any data in an existing file is overwritten with the new information.

You may use the LL command to list cartridge and tape directories, header files, comment files, "?" command responses, and command echoes. Copy status messages and error messages are always displayed on the terminal.

Scratch Area Definition (SC)

Purpose: To specify the cartridge on which the files are to be created.

Syntax: `SC, cart`

cart	The CRN or −LU to be used for the scratch file.

Description:

Normally, FC builds each of its internal scratch files on the first cartridge with sufficient space for the initial size of the file. The SC command lets you specify the cartridge on which the files are to be created. (This command is useful as a way to avoid "Cartridge Full" errors in copy operations that require large amounts of scratch file space.)

You must enter the SC command before you perform a CO operation.

Title (TI)

Purpose: Sets the title to be included in the header file of tapes created by subsequent copy commands.

Syntax: `TI,title`

title	The text, up to 72 ASCII characters, that specifies a title (extra characters are truncated).

Description:

Enter the title exactly as it is to appear in the header; lowercase characters are not converted to uppercase.

Transfer (TR)

Purpose: Transfers control to a command file and returns.

Syntax: `tr <file>`

TR	Transfer back to the next higher level command file when TR files are nested. When encountered in the first level command file, serves the same function as the FC Exit command.
TR,namr	Transfer control to the named command file.
TR,−	Transfer control to the terminal.

Description:

FC can be run interactively from a terminal, from a command file, or as a combination of both.

Command files can be nested to four levels, with control passed from file level to file level using the TR,namr command. The current state of any of the configuration commands (Set List Device,

Title Header File, Name Comment File, Echo Commands, Group Commands) and the default values remain in effect through all levels of the nested command files.

Tape Handling

FC creates tapes in a special, unique format (shown at the end of this section) and thus cannot be written or read using the FMGR ST or DU commands. The program prompts you to mount the initial tape on the specified device or to mount the next volume of a multi-volume set during the read or write operation, with one of the following messages:

```
put volume on LU n

put volume number n on LU n
```

FC checks the tape and device status and, if a problem is encountered, issues the appropriate error message:

```
tape LU n not ready

tape LU n media not initialized

tape LU n not write enabled
```

Each of the messages above is followed by one of these messages:

```
when ready to continue type GO, otherwise type BR

when ready to continue type GO, otherwise type BR or SK
```

If a “media not initialized” error occurs, initialize the CTD tape using the FORMC FO command (see Chapter 5 for information on formatting).

When you enter BR in response to the “when ready” prompt, FC terminates cleanly.

The SK option is viable only if the operation is a multi-volume read. The SK response means skip the volume specified by “n” in the “put volume” message. In this case, FC repeats the “put volume” message for the next volume in the set and reports “files lost” or “data lost” error messages for files on the SKipped volume. Refer to the section “FC Error Messages” later in this chapter for a description of the message text and corrective action.

FC automatically rewinds the tape at the beginning of a copy operation and writes over any existing data. For this reason, you cannot modify a tape created by FC; you must copy all files in a single operation. Use LH or DL to check the current contents before copying over a previously used tape. You can create a FMGR transfer file to run the FC LH command that issues a prompt before executing the copy.

For example:

```
:RU,FC,LH,-8,K
:PA,,DO YOU WANT TO OVERWRITE THIS TAPE (type :,YE or :,NO)?
:IF,1G,EQ,YE,2
:IF,1G,NE,NO,-3
:TR
:RUN,FC,CO,XX,-8
:TR
```

FC reads the first record of a tape volume and, if it does not conform to the FC tape format, either logs a tape format error or issues the message below (followed by the two records in ASCII):

```
tape not readable by FC, first two records are ...
```

FC locks the LU before a tape is accessed to prevent other programs from accessing the tape at the same time. If the tape is write-protected, the LU is unlocked at the end of the command. Otherwise, it remains locked until either FC encounters a tape-oriented command that specifies a different LU, a timeout occurs while FC is waiting for a command, or FC finishes executing.

To protect the tape, FC sets the device offline before the lock is released, unless you specified the K option in the last tape command or the command terminated due to an error or a break command. In those cases, the tape remains online. The CTD device is taken offline by unloading the cartridge (as if the UNLOAD button on the drive was pressed).

Since it can be inconvenient to have the device taken offline, you should use the write-protect mechanism or run FC interactively to retain the lock between commands. If you use the K option, which keeps the tape online in all circumstances, be aware that the tape is unprotected from access by other programs while it is unlocked and online.

If you interrupt a copy to tape by a BR (Break) command, the full tape generally is not usable. (In multi-volume sets, of course, volumes copied prior to the BR are intact.) However, if you specify BR during the verify phase, all data copied up to the point of the break is usable.

If you select so many files that one tape cannot hold the entire save, you must use multiple tapes. Large files are not copied across tape boundaries.

FC format on CTD:

FC format on mag tape:

Destination Disk Handling

FC performs a size check on destination cartridges on tape-to-disk copies before it copies any files. If it finds there is insufficient data or directory space for the files to be stored, FC issues the appropriate error message and exits. (Note that in open mode, other processes can create files on the cartridge; thus, it is possible for cartridge-full errors to occur even when the size check indicates sufficient space for the copy operation.)

For disk-to-disk copies, the size check is not performed and if a directory-full or cartridge-full error occurs, FC terminates and stores the status in the FMGR globals. (Refer to the section on "Error Handling in Transfer Files" later in this chapter for a definition of the FMGR global returns.)

Performance Considerations

The following steps can be taken to help increase the speed of the file copy operation.

1. Specify source cartridges explicitly if multiple cartridges are contained in the cartridge list. For example, if the cartridge list includes cartridges 10, 20 and 30 and you want to copy from cartridge 30, use a CO command as follows:

   ```
   CO,{A::30,B::30,C::30,D::30,E::30},#8
   ```

   In this way, FC does not have to search cartridges 10 and 20 for the files to be copied. You can also use DE to specify the cartridge, as follows:

   ```
   DE,::30
   co,{a,b,c,d,e},-8
   ```

   Note that in the command above, you may not put sequences of commands in the runstring.

2. Keep file extents collected. Copying speed is increased if the file main and its extents are adjacent on the source cartridge. To make the extents adjacent on all files on a cartridge, use a CO command, as follows:

   ```
   co,100,,ldf
   ```

   This copies the contents of cartridge 100 to itself (the destination cartridge defaults to the same cartridge as the source). You must specify the L option to ensure that the extents are adjacent. Select the D option so that the new files (with adjacent extents) can replace the old files. The F option logs the name of each file copied.

   If a cartridge-full or directory-full error (FMGR-033 or FMGR-014) occurs during the above operation, pack the cartridge using the FMGR PK command and enter CO again. You may need to repeat this if the FMGR-033/-014 error recurs. If this happens on the same file, however, PK does not help and you must use a larger cartridge for the operation.

If there are many small extents, reduce the number by making file sizes larger. To do this, copy each file with the ST command. Alternatively, you can eliminate all extents on a cartridge by using the CO command as shown and specifying the E option. (Note that it is not always desirable to eliminate extents, because FMP is not able to re-use the space as easily when the files are purged and larger file sizes tend to result in less efficient use of space.)

Extent collection (and elimination, if desired) can be done as a periodic cleanup function, much like a FMGR PK. If the extent collection/elimination process affects a lot of files (as shown by the F option log), issue a FMGR PK command after the CO command.

3. Use Lock and Open modes efficiently. Determine which one works faster for an application and use that mode whenever possible. Select the mode by specifying the L or O option in the CO command. Lock mode is more efficient for most copy operations that involve large numbers of files, because entire cartridges are locked in one operation, rather than by opening individual files. However, Lock mode may introduce other overhead, which could make it slower than Open mode. Open mode is usually more efficient when you are copying a small number of files. The faster mode for a given cartridge may depend on the following factors:
   - The number of files being copied to or from the cartridge.
   - The total number of files on the source disk cartridge.
   - Whether the destination disk cartridge was initially empty.
   - The number of files purged as a result of the D option.
4. Use cartridges efficiently. It is faster to copy to empty cartridges in Lock mode than to copy to cartridges containing other files.

Loading FC

FC must be loaded online since it runs as an extended background program. Load the utility using the LINK command file, #FC6 (part number 92084-17151). The load process also requires copying the FC help file, ”FCHLP (part number 92084-17150), to a cartridge available to all users.

Using Globals in Transfer Files

The information in the FMGR globals is helpful when you write transfer files to copy one source cartridge to a series of destination cartridges, moving to the next cartridge when the current cartridge is full. Such a file is particularly useful for backing up large, fixed disks to a series of flexible disks.

The transfer file makes use of the fact that the CO command file1 parameter specifies the first file to be copied, and the 10G global contains the name of the next file to be copied when a copy terminates due to a cartridge-full error. When you do a series of copies in which the ASCII contents of 10G are used as the file1 parameter, each copy can continue where the previous one stopped.

The following is an example of a transfer file used for this purpose. The source cartridge is specified in the 1G global, the negative flexible disk LU is specified in the 2G global, and the system master security code (msc) is specified in the 3G global. The msc is required for the ! option, which is specified to clear the destination disk without requiring operator intervention.

```
:PA,,Put in floppy and type TR
:MC,2G
:RU,FC,CO,1G,2G,!VF,,,3G
:DC,2G
:IF,5P,NE,-1,5
:PA,,Put in next floppy and type TR
:MC,2G
:RU,FC,CO,1G,2G,!VF,10G,,3G
:DC,2G
:IF,5P,EQ,-1,-5
:TR
```

Assuming a transfer file name FLOPUP, source CRN 100, destination LU 10, and msc XX, the following FMGR TR command causes CRN 100 to be backed up to a series of flexible disks on LU 10:

```
tr,flopup,100,-10,XX
```

If CRN 100 contains 25 files and 10 of these fit on the first flexible disk, 10 on the second, and five remain for the third, the process is:

```
CI> tr,flopup,100,-10,XX
CI> pa,,Put in floppy and type tr
CI> tr
CI> mc,-10
CI> ru,fc,co,100,-10,!vf,,,xx
file1                                   (First ten files copied)
file11
 fmgr-033                               (Cartridge-full error)
copy terminated
CI> dc,-10
CI> if,-1,ne,-1,5
CI> pa,,put in next
CI> tr
CI> mc,-10
CI> ru,fc,co,100,-10,!vf,file11,,xx     (Next file namr in 10G)
file11                                  (Next ten files copied)
file21
fmgr-033                                (Cartridge-full error)
copy terminated
CI> dc,-10
CI> if,-1,eq,-1,-5
CI> pa,,put in next floppy and type TR
CI> tr
CI> mc,-10
CI> ru,fc,co,100,-10,!vf,file21,,xx     (Next file namr in 10G)
file21                                  (Copy remaining files)

file25
CI> dc,-10
CI> if,0,eq,-1,-5                       (Test for cartridge-full error)

CI> tr
CI>                                     (Copy complete; exit)
```

Error Handling in Transfer Files

Certain error conditions can be detected in FMGR transfer files by checking the FMGR globals after running FC, or, if FC is scheduled by a program other than FMGR, the scheduling program can retrieve the same information that appears in the FMGR globals by making a call to RMPAR. Since only one error can be returned in the globals, only the highest error number (the most severe error) is returned, even if multiple errors occur during FC execution. The parameters returned by RMPAR correspond to the FMGR globals as follows:

PARAM(1)	=	1P or characters 1-2 of 10G (next-source-file information)
PARAM(2)	=	2P or characters 3-4 of 10G
PARAM(3)	=	3P or characters 5-6 of 10G
PARAM(4)	=	4P (error category number)
PARAM(5)	=	5P (next-source-file information status)

4P can have the following values:

10000 – program aborted due to OF or system violation. Violations (such as Memory Protect) might result from a bad segment load.

The following errors terminate FC:

1000 – FC internal error
300 – transfer file stack overflow
200 – scratch file error (see SCRATCH command)
100 – segment loading error

The following errors terminate the current command:

90 – general error causing command termination
80 – destination disk cartridge or directory full
70 – break detected (system BR command)

The following errors allow the command to continue, but the purge phase is skipped on disk-to-tape copies:

50 – copying from current source volume aborted (tape-to-disk)
40 – unusual source file access error – file not selected
30 – general error on file being copied – file skipped
25 – tape format error (reading or verifying tape)

The following are warnings that do not affect FC operation:

20 – warning
10 – soft data error detected
0 – no error or warning

Global 5P can have the following values:

1	−	next source file information not applicable
0	−	no error
−1	−	next source file information valid (4P = 80)
−2	−	source is tape, number of next file greater than 32767

Globals 10G and 1P have possible values of:

If 5P	=	−1 (4P = 80) and source is a disk:
10G	=	name of next file to be copied from disk.
If 5P	=	−1 (4P = 80) and source is a tape:
1P	=	reference number of next file to be copied from tape.
If 5P	=	1 and 4P not 10000, or 5P = −2, or 5P = 0:
10G	=	ASCII nulls
1P	=	0
If 4P	=	10000 (5P = 1):
10G and 1P are undefined.		

In some circumstances, the meaning of the FMGR global information may be ambiguous. Since only the most severe error is posted to the FMGR globals, it is not possible to determine whether other errors occurred or to determine the command that caused an error if multiple commands were entered. Furthermore, if a copy command has more than one source cartridge, the cartridge containing the “next source file” cannot be determined. If the GR copy command is used, additional ambiguity may result.

Because of this, the use of FMGR globals to determine the “next source file” is intended only for command files that contain a single CO command and involve a single source cartridge.

FC Error Messages

This section contains a list of all the messages that FC generates. Brackets [] surround those parts of a message that are included only under certain conditions. Angle brackets <> indicate variables that are replaced with the appropriate value when the message is issued.

Errors Requiring Operator Action/Response

Do you really want to purge disk LU <#> CRN <crn> ?

A CO command specified the C option in the options parameter to clear the destination cartridge of all current data. FC requires a response to this prompt to be sure that you really want to destroy the data on the named LU or CRN. Use ! instead of C to clear the destination cartridge if you want to suppress this warning prompt.

LU <#> is down
correct problem and UP device, or use BR to break FC

Prior messages may help you understand the problem. Once the situation is corrected, bring the device up using the system UP command, and FC will continue. If you did not solve the problem before issuing UP, this message displays again. If you cannot correct the problem, issue BR to have FC terminate the current command.

tape LU <#> not ready
when ready to continue type GO, otherwise type BR [or SK]

This error usually means that the magnetic tape device is not online or that the CTD cartridge is not loaded. Be sure that the correct LU was specified in the command. (Refer to the section on "Tape Handling," earlier in this chapter, for an explanation of GO, BR, and SK.)

tape LU <#> media not initialized
when ready to continue type GO, otherwise type BR

You tried to write to an uninitialized CTD cartridge. Use the FORMC FO (Format) command to initialize the cartridge. (Refer to the earlier section on "Tape Handling" for an explanation of GO and BR.)

tape LU <#> not write enabled
when ready to continue type GO, otherwise type BR

For magnetic tape, you should install the write-protect ring to enable writing to the tape. For CTD, change the protect switch on the cartridge so that the arrow points away from the direction labeled "SAFE." (Refer to the section on "Tape Handling" earlier in this chapter for an explanation of the GO and BR responses.)

put volume number <#> on LU <tape LU>
when ready to continue type GO, otherwise type BR [or SK]

The required tape should be mounted and online, or loaded. (Refer to the section on "Tape Handling" in this chapter for an explanation of the GO, BR, and SK responses.)

waiting to lock list device

FC tried to lock the list device, but discovered that either another program has the list device locked or a resource number for the lock is unavailable. In either case, FC waits, retrying the lock every 2 seconds. Since operator intervention is not necessary, you can enter the system BR command to abort the command, rather than wait for the device or resource to become available.

Information Messages and Warnings

<source namr> [<dest namr>]

When a namr or a pair of namrs is logged with no other message, the indicated file was successfully copied from the source to the destination. When only a single namr is logged, it is the namr for the source file. When two namrs are logged, they are for the source and destination files, respectively. (The namrs may consist of the file name only or may include the CRN.) Logging namrs of successfully copied files is done in Full mode only. (Refer to the "CO Command" section earlier in this chapter for a description of the Brief and Full modes.)

tape needed: <#> feet if 800 BPI, <#> feet if 1600 BPI

This message appears on a copy from disk to 1600 BPI magnetic tape if you specified the T option.

tape needed: <#> CTD blocks

This message appears on a copy from disk to CTD if you specified the T option.

scanning directories

FC is searching the cartridge directories to determine which files are being copied to tape.

copying files

FC is about to begin copying files to tape (assuming that the tape unit is ready).

cleaning up

FC finished copying files to tape and is now closing source files or unlocking source cartridges. This process may take some time and should be allowed to complete.

verifying volume

FC finished writing to the current volume and is now reading the volume in order to verify it.

break acknowledged

FC detected a system BR command and is in the process of cleanly terminating the current command.

writing tape at: <system time string>

Displays the date and time that are written in the tape header for this disk-to-tape copy.

beginning group

A GR command was entered, and subsequent CO commands are executed together once the EG (End Group) command is encountered.

group aborted

The AG (Abort Group) or AB (Abort FC) command was entered, causing all CO commands following the GR command to be ignored.

device is up, FC continuing

The device is now UP, and FC is attempting to continue the operation.

FC xxxxx-xxxxx REV.xxxx <xxxxxx.xxxx>
Use ? for help.

This message appears before the first prompt is issued in interactive mode. The first line indicates the FC part number, the revision code, and the date/time code for the last revision.

(timeout)

FC detected a terminal timeout while waiting for input from the terminal. If a tape LU is locked, the lock is released, which may require taking the tape unit offline. Refer to the section on "Tape Handling" for details.

copy terminated

This message displays when a copy operation is terminated before completion for any reason, including an error or operator break.

title: <title>

Confirms the title entered in the TI command.

warning: title truncated
title: <title>

The title specified with TI was too long and was truncated. The truncated title is logged.

warning: no match for:[namr = <namr>][file1 = <?>][file2 = <?>]

No files were found that match both the source namr and the file1 and file2 parameters. Any combination of the three items may be listed, depending upon what was specified. Note that this does not mean that a match was not found for the file1 or file2 parameter itself, but that no files within the specified file1/file2 range matched the namr.

warning: unable to eliminate extents in one or more files

Although the E option was specified, extents could not be eliminated from some files. See the description of the E option for details.

no files selected

The CO command was not executed, because no source files were selected.

disk write required retries:
LU <#> trk <#> sec <#> thru trk <#> sec <#>

The indicated disk write was unsuccessful on the first try, but succeeded after it was retried one or more times. Failure can be due to an error reported by the disk driver or a verify error if the V option was specified.

disk read required retries:
LU <#> trk <#> sec <#> thru trk <#> sec <#>

Same as above, but for disk read.

disk directory write required retries:
LU <#> dir tr <#> thru dir tr <#>

Same as above, but for disk write in the directory area. The location of the read is isolated to a particular track or range of tracks. The track numbers indicated after "dir tr" are the directory track numbers, not the absolute track numbers on the disk LU. The first directory track (the one with the greatest absolute track number) is indicated as "dir tr 1".

disk directory read required retries:
LU <#> dir tr <#> thru dir tr <#>

Same as above, but for disk read in the directory area.

tape format error <positive error code>

The data read from tape deviates from the expected format in some way, but copying from the tape can continue. The deviation is usually the result of a non-fatal tape read error. This message does not mean data is lost, but is provided primarily for diagnostic use. For tape format errors with a negative error code, see the section "Errors that Result in Rejection of Current Source Tape Volume."

bad tape cartridge entry is:
<octal dump>

bad tape diskfile entry is:
<octal dump>

bad chunk header is:
<chunk size> <chunk type> <#entries> <1st file # or 0>

bad microdirectory entry is:
<file#> <file blk> <#blks> <buffer block>

The four messages above may accompany a tape format error message and are for diagnostic use only.

I/O-<error code> on LU <#>[,D][,F]

This message has the same meaning as the operating system error message **I/O-<error code> @LU <#>[,D][,F]. See the *RTE-6/VM CI User's Manual* for more information.

Disk Data I/O Errors

disk write failed: LU <#> trk <#> sec <#> thru trk <#> sec <#>

A disk write failed, even after retrying. The failure may be due to an error reported by the disk driver or a verify error if the V option was specified. FC also reports a disk error (FMGR-001) for each file affected.

disk read failed: LU <#> trk <#> sec <#> thru trk <#> sec <#>

A disk read failed, even after retrying. The failure may be due to an error reported by the disk driver or a verify error, if the V option was specified.

FC also reports a disk error (FMGR-001) for each file affected. On disk-to-disk copies, the affected files are not copied. On disk-to-tape copies, the affected files are copied, but the affected parts of the files are flagged to indicate the disk error so that an appropriate error message can be given when the tape is read.

Non-Fatal Tape Read Errors

tape read error

The driver indicated an unrecoverable error on the tape read. FC does not retry in this case, because the driver would have retried if appropriate.

checksum error (chunk header)

The driver indicated a successful read, but FC's checksum information indicated a problem. FC tries again to recover; however, if retries fail, FC considers the information lost.

checksum error (chunk body)

The driver indicated a successful read, but FC's checksum information indicated a problem. FC tries again to recover; however, if retries fail, FC considers the information lost, unless the I option was specified. In that case, FC may try to use the information in spite of the checksum error, depending upon the context in which the error occurred. Appropriate warnings are displayed. See the "Data Lost" messages below for more information.

bad record length

The driver indicated a successful read (from magnetic tape), but the record length indicated by the transmission log returned from the driver disagreed with the length value specified in the header field of the record. FC tries to recover from this, but if retries fail, FC considers the information lost.

attempting to use data in spite of checksum error

See information above for "checksum error (chunk body)."

retrying tape read

A "checksum error ..." or "read record length" error occurred, and FC is attempting to recover from the error by retrying the tape read.

Errors Affecting a Single File

can't open <name>::<CRN> due to error:
FMGR <nnn>

The indicated source file could not be opened to copy. Typically, this is because the file is open exclusively to another program (FMGR-008) or because the cartridge is locked to another program (FMGR-013).

file <name>::<CRN>
purged or replaced during copy or directory entry corrupt

This file was purged and possibly replaced by a different file when FC was scanning the directory, or a directory entry for the indicated file is corrupt. The file is not copied.

<source namr> [<dest namr>]
FMGR <nnn>

A namr or pair of namrs followed by a FMGR error code means the file was not copied because of the FMGR error. However, for a FMGR-001 error on a disk-to-tape copy, the file is copied to tape but the affected parts of the file are flagged to indicate the disk error, so that an appropriate error message can be given when the tape is read.

When a single namr is logged, it is the namr for the source file. When a pair of namrs are logged, they are for the source and destination files, respectively. Namrs may consist of the file name only or may include the CRN.

<source namr> [<dest namr>]
FMGR <nnn>
warning: above file copied but not purged from source

Similar to the above error, except that the FMGR error only prevented the source file from being purged as requested by the P option; the file was still copied.

<source namr> [<dest namr>]
file too large for this operating system

Similar to the above error, except that the "file too large ..." error occurred, rather than a FMGR error. The file cannot be copied because it is larger than 16383 blocks. This may happen when you copy files from a tape created under another operating system that supports files of that size.

<source namr> [<dest namr>]
data lost

Similar to the above error, except that the "data lost" error occurred rather than an FMGR error. Data lost errors can only occur on tape-to-disk copies and indicate that file data was lost due to a problem reading the source tape. A read problem may be identified in earlier messages (see "Non-fatal Tape Read Errors"), or a record may have been dropped, in which case no problem is detected other than the data lost error itself.

For data lost errors, other messages may intervene between the line with the namrs and the data lost message. The data lost error applies to the most recently logged file namrs, even if the namrs do not immediately precede the data lost message.

If the message "checksum error (chunk body)" appears anywhere before a "data lost" message, you may recover some of the lost data by repeating the tape-to-disk copy with the I option specified. When I is used, destination files are created even though some or all of the data was lost, and "data lost" messages are replaced by some of the more specific messages below, which indicate the exact part of the file affected and whether the data loss is certain.

<source namr> [<dest namr>]
data lost due to disk error when tape was made

Same as above, except the data was lost because of a disk error that occurred when the tape was written, not because of a problem reading the tape. An FMGR-001 (disk error) occurred when this file was copied to tape.

<source namr> [<dest namr>]
data lost: entire file

Same as "data lost," except this message occurs when the I option was used to try to recover part of the file, but the attempt did not succeed. Because I was specified, the destination file is created even though all of its data was lost. The resulting destination file is of no value, except for identifying information that may be obtained from the directory entry.

<source namr> [<dest namr>]
data lost:
[xtnt <x1>] blk <b1> thru [xtnt <xn> blk <bn>]

Same as above, except that only part of the file was lost. The message indicates the range of blocks that were lost. The first and last block in the range are specified by giving the extent number (omitted if the block is in the main extent) and the relative block number within the main or extent. Blocks are numbered from zero within the main or extent.

More than one of these messages may be given for a single file. The file namrs are not repeated for subsequent data lost messages on the same file, even though other messages may intervene, thereby breaking up the sequence of data lost messages.

Because the I option was specified, the destination file is created even though some or all of its data was lost.

<source namr> [<dest namr>]
possible data loss: entire file

Same as "data lost: entire file", except that it is uncertain whether any data was lost. The file should be examined to determine the loss, if any.

<source namr> [<dest namr>]
possible data loss:
[xtnt <x1>] blk <b1> thru [xtnt <xn>] blk <bn>

Same as above, except that possibly only part of the file was lost. The message indicates the range of blocks that may be lost. The first and last block in the range are specified by giving the extent number (omitted if the block is in the main extent) and the relative block number within the main or extent. Blocks are numbered from zero within the main or extent.

More than one of these messages may be given for a single file. The file namrs are not repeated for subsequent data lost messages on the same file, even though other messages may intervene, thereby breaking up the sequence of data lost messages.

<source namr> [<dest namr>]
data lost due to disk error when tape was made: entire file

Same as "data lost due to disk error when tape was made", except this message occurs when the I option was used to try to recover part of the file and the attempt did not succeed.

Because the I option was specified, the destination file is created even though all of its data was lost. The resulting destination file is of no value, except for any identifying information that may be obtained from the directory entry.

<source namr> [<dest namr>]
data lost due to disk error when tape was made:
[xtnt <x1>] blk <b1> thru [xtnt <xn>] blk <bn>

Same as above, except that only part of the file was lost. The message indicates the range of blocks that were lost. The first and last block in the range are specified by giving the extent number (omitted if the block is in the main extent) and the relative block number within the main or extent. Blocks are numbered starting at zero within the main or extent.

More than one of these messages may be given for a single file. The file namrs are not repeated for subsequent data lost messages on the same file, even though other messages may intervene, breaking up the sequence of data lost messages. This message may appear repeatedly for different parts of the same file, even if the different parts are adjacent and even if the net meaning of all the messages is that the whole file was lost.

Because the I option was specified, the destination file is created even though some or all of its data may have been lost.

source file <name>::<CRN>
selected by commands with conflicting parameters

On a tape-to-disk copy specified by a GR command with more than one CO command, the indicated source file was selected by more than one CO command, but those particular commands did not meet the following restrictions:

- The destination parameters must be the same in all the commands.
- The D option must be consistently selected or not selected in all the commands.
- The E option must be consistently selected or not selected in all the commands.
- The msc parameter must be consistent through all the commands.
- You must not specify the L option in some of the commands and the O option in others.

Loss of Unidentified Files on Copy from Tape

files lost, reference numbers <n1> thru <n2> names not available

Indicates that distributed directory information was lost for files with reference numbers n1 through n2. Because this information was lost, the names and other information about the files is not known. You can list the main tape directory with the DL command to determine what files these numbers correspond to.

Disk-to-Tape Copy Verify Errors

files lost, reference numbers <n1> thru <n2>

Indicates that distributed directory information was lost for files with reference numbers n1 through n2. If necessary, you can list the main tape directory with the DL command to determine what files these numbers correspond to.

data lost for file, reference number = <#>

Indicates that data was lost for file reference number n. If necessary, you may list the main tape directory with the DL command to determine what file this number corresponds to.

error: header file verify error

Usually indicates that no data was actually written to the tape, probably due to a defective write head on the tape drive.

Errors that Cause Rejection of Current Source Tape Volume

tape not readable by FC, first two records are:
<record 1>
<record 2>

The tape mounted is apparently not an FC tape. The first two records are logged to help determine just what the tape is.

volume does not match others

On a multi-volume tape-to-disk copy, the tape volume just mounted does not come from the same set as the previous volumes.

wrong volume

The tape volume just mounted does not have the correct volume number. Volumes in a multi-volume tape set must be restored in the correct sequence. When FC prompts for a new volume, it indicates which volume number is to be mounted.

fatal tape read error

The tape driver reported an error when the header or comment file was read. FC makes no attempt to recover from this; however, you can retry the operation manually. On multi-volume tape-to-disk copies, you need not start over with the first volume; just enter GO at the prompt to mount the volume. Refer to the earlier section in this chapter, “Tape Handling,” for an explanation of the options available at this point.

header file checksum error

The information in the header file is questionable, due to a checksum error. It would be unwise for FC to continue reading the volume in this situation. You may retry the operation manually.

tape format error <negative error code>

This error indicates that the data read from tape deviated from the expected format so severely that FC cannot continue reading the volume. For CTD, this usually means that the tape is blank or is not a CTD tape; otherwise, it is probably the result of a non-fatal tape-read error. The negative error code is for diagnostic use only. You may retry the operation manually.

For tape format errors with positive error codes, see the section on “Information and Warning Messages.”

Command Syntax, Parameter Errors (Command is Skipped)

error: no such command (use ? for help)

error: bad or missing parameter

error: braces used where not permitted

These error messages are self-explanatory.

error: bad namr

An incorrectly formed namr appeared in the command.

error: mismatched braces

Self-explanatory.

error: namr list not allowed in this context

A list of namrs delimited by braces was used in a parameter where a list of namrs is not permitted.

error: unrecognized option character

An unrecognized character appeared in the “options” parameter.

error: incompatible options

Contradictory options, such as B and F, or L and O, were specified in the same command.

error: option not defined for this command

An option character that is not meaningful for this command was specified in the “options” parameter.

error: option not applicable

An option character that is not applicable for this type of copy (disk-to-disk, disk-to-tape, or tape-to-disk) was specified in a CO command.

error: bad file1 or file2 param

The value for the file1 or file2 parameter was incorrectly specified. These parameters specify file names when copying from disk and file numbers when copying from tape.

error: bad msc parameter

Self-explanatory.

error: no tape-to-tape copy

Both the source and destination parameter specified a tape device, implying a tape-to-tape copy, which is not supported. Make sure you have not used the DE command incorrectly.

error: source or destination incompatible with group

The source and destination parameters must be either a disk or a tape consistently throughout the grouped commands, and you may only specify one tape LU in a given group.

error: options inconsistent with group

Within a group of commands, you may not specify certain options--V, I, S, F, B, T, and K--in one CO command but omit them from another. DE may be useful to ensure these options are used consistently in all the CO commands within a group.

error: L and O options must be consistent for a given cartridge

The L and O options must be consistent for a given disk cartridge within a group. Therefore, both L and O may not be specified for the same cartridge, even in different CO commands within a group. You may specify L or O in one command and no option in another. (In this case, the specified option affects both commands.)

error: renaming multiple files to same name, or cartridge not found

The source name was omitted, or has wildcards, or the source is a list of namrs, but the destination specifies a file name. A destination file name is permitted only if it is clear that the source parameter can select only one file.

It is possible that the destination parameter was intended to specify an ASCII CRN in the abbreviated form (CRN rather than ::CRN), but the ASCII CRN was interpreted as a file name because no cartridge with that CRN was found on the cartridge list.

error: bad destination name

An illegal file name was specified in the destination parameter.

error: filename BOOTEX reserved for system

The destination parameter specified BOOTEX as the destination file name. This is not permitted, because the name BOOTEX is reserved for the system boot extension file.

error: on disk-to-tape copy, dest cartridge may not be specified as −LU

If you specify a cartridge in the destination parameter on a disk-to-tape copy, it must be a positive CRN, not a negative LU.

error: P option requires explicit source cartridge

You may not specify the P option without also specifying the source disk cartridge explicitly in the source parameter.

error: L and O options require explicit source cartridge

You may not specify the L or O option without also specifying the source disk cartridge explicitly in the source parameter.

error: P option not allowed if source and dest are same cartridge

You may not specify the P option on a disk-to-disk copy in which the source and destination are on the same cartridge.

error: C and ! options require explicit destination cartridge

When you specify the C or ! option, you must also specify the destination disk cartridge explicitly in the destination parameter.

error: dest secu code not allowed w/o source secu code or good msc

A security code was specified in the destination parameter, but no security code was specified in the source parameter and the msc parameter was not specified correctly.

error: bad LU or device not supported

An LU specified in the command is not the LU of a disk or tape device supported by FC.

error: bad tape LU

The tape LU specified in the command is not the LU of a tape device supported by FC.

Command Out-of-Sequence Errors (Command is Skipped)

error: already in group

A GR command was entered before the previous GR ended.

error: not in group

An EG or AG command was entered, but there was no previous GR to end or abort.

Other Errors that Terminate Current Command

error on help file FCHLP: FMGR <nnn>

The indicated error occurred during an attempt to access FC's help file, FCHLP, needed for the ? command.

cartridge lock error on disk LU <#>:
FMGR <nnn>

FC was unable to lock the indicated disk cartridge due to the FMGR error. If the error is FMGR-103 (Corrupt Directory Detected), resolve the problem before trying to access any files on the cartridge, otherwise unnecessary loss of data may result.

cartridge unlock error on disk LU <#>:
FMGR <nnn>

Same as above, but for cartridge unlock.

cartridge clearing error on disk LU <#>:
FMGR <nnn>

Same as above, but for clearing the directory on the disk cartridge.

error in cartridge status on disk LU <#>:
FMGR <nnn>

Same as above, but more general.

disk directory write failed: LU <#> dir tr <#>[thru dir tr <#>] error writing directory on disk LU <#>:
FMGR <nnn>

A directory write to the specified track or tracks failed due to a disk error (FMGR-001) or verify error (FMGR-049), even after retries were attempted. The location of the write is isolated to a particular track or range of tracks. The track numbers indicated after "dir tr" are the directory track numbers, not the absolute track numbers on the disk LU. The first directory track (the one with the greatest absolute track number) is indicated as "dir tr 1".

disk directory read failed: LU <#> dir tr <#> [thru dir tr <#>]
error reading directory on disk LU <#>:
FMGR <nnn>

Same as above, but for directory read instead of write.

comment file error:
FMGR <nnn>

The indicated FMGR error occurred when the comment file (determined by the most recent CF command) was opened or read.

error: number of cartridges to be copied to tape exceeds limit

Cartridges were renamed on a copy to tape, resulting in the tape's cartridge list containing more than 64 distinct cartridges.

error: volume not big enough for header/comment/directory files

The destination tape is too small to be used. There is not enough room for the minimum amount of information that must be written before going on to another volume.

error: fatal i/o error on LU <#> status=<octal A-register status>

FC attempted I/O to the specified LU and the status returned in the A-Register indicates a problem that should never occur. The status value in the message is the value returned in the A-Register from the EXEC call.

error: comment file checksum error

A checksum error was detected when reading the comment file from the tape.

error: not enough memory

FC does not have enough free memory to let the command proceed. If FC was sized to fewer than 32 pages, increasing the size may solve the problem. If you used the GR command, try using fewer CO commands in the group to prevent this problem.

error: tape lock failed

FC cannot lock the tape LU required by the command. Either the tape LU is locked to another program or no resource number is available for the lock.

error: tape DESCRIBE error

The CTD driver reported an error when FC issued a “DESCRIBE” control request.

error: error on tape write

An unrecoverable error was reported by the driver on a tape write request. FC does not retry in this case, because the driver would have retried if appropriate. FC cannot continue copying to tape after this, as the resulting tape would be corrupt.

CRN <CRN> not found

A tape-to-disk CO command with an unspecified destination cartridge was issued, and the required CRN, determined by reading the tape directory, was not found.

CRN <CRN> LU <#> not large enough:
data blocks/dir entries needed: <bl>/<ent> available: <bl>/<ent>

The specified cartridge does not have enough file or directory space to copy the files from the tape.

Other Errors

list file error:
FMGR <nnn>

An error occurred when FC attempted to access the list file specified with the most recent LL command. This causes the list device to revert to the terminal; it does not cause the command to terminate.

command file error:
FMGR <nnn>

An error occurred when FC tried to read a record from the command file (determined by the most recent TR command). This causes an automatic TR to the terminal (FC enters interactive mode).

Errors that Terminate FC

fatal scratch file error:
FMGR <nnn>

An error occurred when FC tried to create or access one of its scratch files. Each scratch file is created on the first cartridge that has room for the initial size of the file, unless you specified a cartridge in the SC command.

If the error code is FMGR-033 (disk cartridge full) or FMGR-014 (directory full), make more room on the cartridge or use SC to have FC create the scratch files on a cartridge with more room.

If the error code is FMGR-012 (EOF or SOF error), you probably included too many CO commands in a group. The maximum number of grouped copy commands is approximately 100. (A command that has more than one namr in the source parameter must be counted as a separate command for each occurrence of a namr.)

An error (FMGR-011) can occur when you copy more than 5000 files (approximately) from disk to tape in Lock mode. The error only occurs, however, if FC creates a scratch file on a cartridge that is being locked for the copy. This error occurs during the "scanning directories" phase of the disk-to-tape copy when no files are as yet copied to tape.

To avoid this problem when you need to copy more than 5000 files to tape, FC's SC command to specify a scratch cartridge other than one from which files are being copied. If that is not possible, copy files to tape using the Open mode, rather than Lock, by specifying the O option in the CO command.

For any other error, if SC is used, make sure it is executed before any CO commands.

can't load segment <segment name>

The indicated segment could not be loaded.

TR stack overflow

The command file stack used for the TR command overflowed. This error occurs if more than four nested TR commands are given (excluding any TR command in the runstring).

internal error at <address> last segment loaded: <segment name>

FC detected an inconsistency that cannot be interpreted. This error should never occur under normal conditions. The address and segment name are for diagnostic use only.

internal error: pascal <error type#> <error#> <line#>

FC is being terminated due to a Pascal run-time error. This error should never occur under normal conditions. The numeric values are for diagnostic use only.

exec error <system error mnemonic> exec params: <params in octal>

The operating system or I/O driver rejected an EXEC call made by FC. This error should never occur under normal conditions. The numeric values are for diagnostic use only.

HP Computer Systems File Copy (LIF)

The LIF utility lets you interchange files between an HP 1000 system and other HP computer systems. It translates files from the RTE FMP format to a standard Logical Interchange Format (LIF), and vice versa. For example, you can read LIF files and translate them into FMP files that RTE can manipulate. Using LIF, you can write LIF format files on some media and RTE (FMP) files on others.

LIF executes commands that are similar to CI commands. You can store the output to any legal RTE file or LU. LIF is not intended to duplicate the file management capabilities of CI, but to manipulate files on LIF media and interface with FMP. (See Table 4-1 earlier in this chapter to learn which utilities to use for backup and file interchange.)

Since LIF conforms fully with the HP standard Logical Interchange Format, the file descriptor (file name and subparameters) used in the standard LIF format differs from RTE's file descriptor. When media is read from other systems, checks are made to verify the legality of the format. LIF conforms to the proper format when writing and, whenever possible, reads data that does not conform to the standard format.

At times, you may want to write non-LIF file names. This may be useful in preserving naming conventions if both the source and destination systems are capable of handling both conventions. You may partially disable legality checking to allow non-LIF file names to be written. In this case, the utility reports significant differences from the format, but continues to read the media if possible. (See the section below on "Naming Conventions.")

Since LIF files are all type 1 (ASCII), all FMP files are copied into type 1 interchange files. Conversely, all files copied from an LIF medium are copied into type 3 FMP files, unless another type is specified in the type subparameter of the file descriptor. Type 1 files need not be ASCII; the type indicates that it is an interchange type file, and thus, has the same record format as other interchange files.

LIF works for all supported HP disks using standard HP disk drivers, including mini-floppies, flexible disks, model 7906 disk drive, CS/80 disk drives, and for cartridge tape drives.

Naming Conventions

Not all legal RTE file names are legal LIF file names, because LIF file names consist only of uppercase letters and digits (0-9). When an LIF file name is not specified, the name of the input FMP file is used (the type extension is not included). If that name is not a legal LIF name, legality checking must be suppressed or an error is reported and the file is not written to the LIF medium.

When files are copied between FMGR (6 character), LIF (10 character), and FMP (16 character) files, excess characters are truncated. Thus, several files with different names under one file system may map into the same file name under another file system.

If there is a naming conflict, it is treated as a duplicate name error and the file is not copied.

There are three different types of file descriptors, as used in the command parameter description given below. The first type is the FMP file descriptor and is precisely the same as the file descriptor used for file accesses.

The second type is the LIF file descriptor, which is similar to the first but with a few significant differences. The file name may be up to 10 characters long. It must begin with a capital letter, and the remaining characters must be either capital letters or digits. There is no type extension. LIF accepts any legal FMP file descriptor, including 10 characters, as an LIF file descriptor; however, it may fail to be a legal LIF name (the type extension is deleted). If it is not a legal name, you must disable error checking. It is possible to write illegally named files (refer to the SV command description).

The third type of file descriptor is referred to simply as a namr. This is either an FMP file descriptor or an LIF file descriptor, whichever the context implies. When the file descriptor is of the third type (not specified as either LIF or FMP), the program determines the appropriate type of file descriptor. Ambiguity is resolved by the convention that the mounted LIF medium (there is no more than one at any given time) is searched first. Files that are not clearly specified as either LIF or FMP are searched for (or created) first on the LIF media. If this fails to satisfy the search, the FMP file system is searched using standard FMP calls.

For both types of name, the file descriptor CRN is defined. If that subparameter is a negative number, LIF checks to determine if it matches the mounted LIF media. If so, it is treated as if the reference were to that medium. Otherwise, it is treated as an FMP CRN or global directory name and handled as a normal FMP file descriptor.

Calling LIF

LIF can be run either interactively or from a transfer file. The entry for interactive operation is as follows:

```
CI> [RU,] LIF
```

The LIF utility displays the program prompt:

```
LIF:
```

Enter the commands for the desired operation.

To run LIF without any operator intervention, create a command file containing all desired LIF program commands and enter the command file descriptor in the runstring. The runstring is:

```
CI> [RU,] LIF [,command FMP file descriptor [,list FMP file descriptor]]
```

You may specify a list LU to report error messages or file/device listings. Both the command file and list file must be FMP files, since LIF will not run from an LIF file. The default for both files is the display terminal. If the commands are from a file or a non-interactive device, the prompt is suppressed.

LIF Commands

The commands executed by LIF are a subset of the commands executed by CI. In general, the commands work in LIF the same as they do in CI. In some cases, however, the LIF commands do not provide the full range of optional parameters.

Table 4-10 summarizes the LIF commands, which are described in the following sections.

Table 4-10. LIF Commands Summary

Commands		Description
Information Command		
HElp		Displays available LIF commands and their formats
Configuration Commands		
INitialize	lu	Writes a volume label and blank directory on LIF medium
MC Mount Cartridge	lu	Mounts an LIF cartridge
PK Pack Cartridge	lu	Packs the files on the LIF medium to recover free space
PUrge	filename	Removes a file from the LIF medium
RN Rename	filename	Changes the name of an LIF file
STore	srcefile	Creates files from existing files on FMP/LIF medium to FMP LF/medium
SV Severity	level	Modifies amount of error checking by LIF
Copy Command		
COpy	lu	Copies all the files from an LIF medium to an RTE disk cartridge
Listing Commands		
DL Directory List		Displays the files that are on the mounted LIF medium
LIst	file	Copies FMP or LIF files to the list file or device
LL Set List Device	namr	Changes the list file or device
Transfer and Exit Commands		
EXit		Terminates LIF
TRansfer	file	Transfers control from one RTE file or LU to another

Copy (CO)

Purpose: Copies all of the files from an LIF medium onto an RTE disk cartridge.

Syntax: `CO,-LIF lu,destination mask`

lu	The list file or device the file is copied to.
destination mask	As in the CI CO command. Refer to the *RTE-6/VM CI User's Manual* for more information.

Description:

As each file is copied, the name is logged to the list file or device. If errors occur, the related error message follows the file name. All copied files become type 3 FMP files. There is no parameter to specify renaming or copying a subset of the files. You may not copy an FMP disk to an LIF medium, because the files on an FMP disk may not be suitable for an LIF medium. Most of the FMP files have illegal names (&,%,',...) and are not of a readily interchangeable type.

LIF file names are truncated to six characters if the destination is a FMGR cartridge. If this results in naming conflicts, files are reported as duplicates and must be renamed.

Directory List (DL)

Purpose: Displays the files that are on the mounted LIF medium and provides some information about them.

Syntax: `DL[,mask[,level]]`

mask — The wildcard characters that match any letter. Thus if "DL,B----" is entered, LIF lists all the file names with five or fewer letters that start with "B". The mask can be up to 10 characters long. Note that this mask is not as complex as the mask CI uses.

level — The level of directory information requested. The default is level 0, which lists the file names in the directory alphabetically, five names per row. LIF uses the memory space behind the program to store the file names before sorting them, and there may not be enough space to hold all of them. In this case, the level defaults to level 1, which lists only the file names, one per line, in the order in which they appear in the directory. Level 1 may also be legally specified.

level 2 — Provides a list that includes the file names in the order they appear in the directory, one per line (only level 0 alphabetizes the names), with their type and size in sectors listed on the same line.

level 3 — Includes all of the information available in the directory about the file. After the type and size, the starting track and sector are listed. The date and time stamp (creation time of the file) are listed with the volume number. The volume number indicates which volume of a multi-volume file this medium holds. For a file that spans several media, the volume number indicates which section of the file the current volume holds. If this is the last volume of this file, an asterisk appears before the volume number. (This is normally not used because LIF does not support multi-volume files.) Specifying a level greater than 3 adds a list of purged files to the level 3 information.

For example:

```
LIF: DL,,4

   NAME          TYPE    # SCTS   TRAK/SEC      DATE/TIME STAMP       VOL NUM

 AZUZA12345    00001    000017    0037 013    13:38:42    10/12/80    *000001
 BACDEF1492    00000    000012    0037 011    09:19:26    07/14/81    *000001
```

In this example, file type 00000 identifies the purged file, BACDEF1492.

Exit (EX)

Purpose: Terminates LIF, closing open files and performing other proper termination functions.

Syntax: `EX (or EN, /E, ex, en, /e)`

Help (HE)

Purpose: Provides a display of all available LIF commands and their formats.

Syntax: `HE (or ?, ??)`

Description:

Optional parameters for the LIF commands are specified by brackets. The default values for each command are defined in each related command description.

Initialize (IN)

Purpose: Writes a volume label and blank directory on an LIF medium.

Syntax: `IN,LU[,vol label[,directory start[,directory length]]]`

LU	The LIF medium.
vol label	The name of the volume, the sector address of the start of the directory, the number of sectors reserved for the directory, and the number of tracks on the medium. The default is "Default".
directory start	The default directory start address is 2 (logical sector 2).
directory length	The default directory length is 14 sectors.

Description:

You can use IN to prepare a blank medium to use with LIF and to clear the medium of all resident files. IN writes a volume label, an end-of-directory mark over the first file on the medium, and other information not specified by you. When the command is executed, it checks that the LU is an appropriate LIF medium, up and available, and not already mounted to the system. IN then asks:

```
Do You Want To Clear LU XX
```

A Yes answer causes LIF to initialize the medium. Any other response displays the message "Not Initialized" and the LIF: prompt.

Any parameters not specified are replaced by their default values. The default number of tracks is determined from the system tables. After this information is written on the medium, the medium is mounted and made available to LIF. You need not mount an LIF medium before initializing it.

List (LI)

Purpose: LI copies files, either FMP or LIF, to the list file or device.

Syntax: `LI,filedescriptor`

file descriptor	The file to be copied.

Description:

LI searches the LIF directory first, unless you specified a CRN in the file descriptor. If it does not find the file in the LIF directory, LI searches the RTE directory. When found, the file is listed to the current list file or device. If the file is not found, an error is reported. Note that the LI command does not print line numbers before each line.

Set Logical List Device (LL)

Purpose: LL changes the list file or device.

Syntax: `LL[,FMP file descriptor}]`

FMP file descriptor	The file or device to which listings and error messages are sent

Description:

If the specified file already exists, it is opened and information is appended to it. If the file does not exist, it is created as default type 3, size 24.

Outputs to the list device are generated by the DL, LI, and HE commands, and other routines (error messages, and so on.) The default list device is the scheduling LU, normally the terminal.

Mount Cartridge (MC)

Purpose: Mounts an LIF cartridge, specifying that it is the cartridge to be referenced by subsequent commands.

Syntax: `MC,LU`

LU The LIF medium.

Description:

The MC command does not mount an FMP cartridge; it checks that the LU being accessed is an appropriate LIF medium, is up and available, and is not mounted to FMP. Then it reads the volume label from the medium and saves the information that specifies the size and type of the medium, the number of directory tracks, and so on. The medium is also checked for conformity to the LIF standard, and to see if severity is positive. Any errors in format are reported.

The mounted LIF medium is used by all subsequent commands that implicitly refer to the LIF medium. Only one LIF medium can be mounted at a time. If one is already mounted, it is dismounted when another medium is mounted.

Pack Cartridge (PK)

Purpose: PK packs all files on the LIF medium to recover free space.

Syntax: `PK`

Description:

The PK command moves the files on the LIF medium next to each other, eliminating the space that was previously occupied by purged files. It also consolidates directory entries, removing the entries for the purged files. Before a PK operation, the purged files are still listed in the directory as type 0 files (shown by a DL,,4). The PK command removes these entries by sliding all subsequent directory entries down. The end-of-directory mark is also moved down to the last active file.

Purge (PU)

Purpose: PU removes a file from the LIF medium.

Syntax: `PU,LIF filedescriptor`

file descriptor The name of the file to purge.

Description:

PU changes the type of a file to 0, which in LIF is a purged file. This is the only action the PU command takes. A purged file cannot be listed; it appears only on a level 4 or higher DL and is physically written over when the media is packed.

Rename (RN)

Purpose: RN changes the name of an LIF file to a new one.

Syntax: `RN,old LIF filedescriptor,new LIF filedescriptor`

old LIF file descriptor The old file name.

new LIF file descriptor The new file name.

Description:

The RN command rewrites the directory entry by inserting the new name for the file. No other information changes. The new file name is first checked, to make sure it is a legal LIF name, unless the SV level is negative (refer to the description of the SV command for details).

Store (ST)

Purpose: ST creates files from existing files on FMP/LIF medium to FMP/LIF medium.

Syntax: `ST,source file descriptor[,destination file descriptor]`

destination file descriptor	The CRN specified by you; otherwise, the file is placed on the LIF medium (since it is at the top of the pseudo cartridge list).

Description:

This command is similar to the FMGR ST command. You may also specify LU numbers. If you specify a CRN as the source file descriptor, it is searched for the file. If CRN is not specified, the LIF medium is searched. If the file is not found, the FMP file system is searched. If the file is still not found, the utility displays:

```
LIF Error: No such file XXX
FMP Error: No such file XXX
```

If you specify a destination file that already exists, ST is not executed and an error is reported (duplicate file name). The same naming restrictions apply as in the CO command.

If you do not specify a destination, when the source file is found, the destination file is given the same name but is placed on the medium of the other format. For example, if the source file is found on an FMP disk, that file is stored on the LIF medium; if it is found on the LIF medium, it is copied to the first FMP disk.

Note that the above example does not imply that stores are done only from LIF to FMP and back. FMP files (or LUs) can also be stored to other FMP files (or LUs). It is also legal for you to store LIF files to other LIF files. Thus, all four combinations are possible with this command:

FMP to FMP,
FMP to LIF,
LIF to FMP,
LIF to LIF.

When you specify an FMP file descriptor, the full file descriptor is used, either as a source or a destination. Thus, you can individually specify the type (default is 3), the size (default is twice the number of sectors occupied on the LIF medium), the CRN (default is the working directory or top of the cartridge list), and the security code (default is 0).

The ST command does not let you specify specialized control, format and file-skipping functions, as in the FMGR ST command. Therefore, ST is not normally used for storing FMP files to different types of FMP files. For example, storing a binary relocatable from cartridge tape into a type 5 file does not create a loadable type 5 file.

Severity (SV)

Purpose: SV modifies the amount of error checking done by the utility. LIF is used to write and read interchange media, checking the media for conformity to the LIF standard. SV allows you to specify how much error checking is required to ensure conformity.

Syntax: `SV[,severity level]`

severity level	The severity level can be either positive or negative. Two positive numbers, 0 and 1, can be entered to ensure conformity to LIF standard. These numbers correspond to the standard LIF revision levels, 0 and 1. A negative severity level turns off checking entirely, allowing illegal names and unusual configurations (such as an unusually placed directory).

Description:

Since SV can be used to store FMP files with names that are illegal in LIF, if the destination system can read non-legal names, it may be easier to maintain the illegal but descriptive FMP names. The utility can read any medium it writes. However, other systems may not be able to read files written with a negative severity level. If no severity level is entered, the LIF utility displays its present value:

```
SV Level is S000001
```

Transfer Control (TR)

Purpose: TR lets you transfer control from one RTE file or LU to another. It lets you specify the source of the commands LIF executes.

Syntax: `TR[,FMP filedescriptor]`

FMP file descriptor	The command file or LU where the LIF commands are to be entered. If commands are not to come from an interactive device, the prompts are suppressed. The default command file is the scheduling LU.

Description:

There is no ":" before each line. Note that there is no command for calculations (as in the FMGR TR command) or to transfer back, except through an explicit TR command to return to the original file. Thus, there is no command file stacking.

LIF Error Handling

There are numerous error checks throughout this utility. When an error is detected, a message is written to the list device indicating the type of error. All LIF errors are matched as closely as possible to an error number from the FMP error codes. The last error reported is in the RMPAR word 1 on program completion.

The error format is:

```
LIF: <message>
FMP: <message>
```

For example:

```
LIF: No Such File XYZ::-41
FMP: Duplicate filename ABC:123:DB:3
```

FMP errors are standard errors described in the *RTE-6/VM CI User's Manual*.

5

Cartridge Backup Utilities

The cartridge save and restore utilities, WRITT and READT, let you back up disk cartridges to magnetic tape. You can back up a single cartridge or, by inhibiting the rewind feature and repeating the WRITT utility, back up multiple cartridges on a single tape. Cartridges can be selectively restored using the FMGR control command CN to position the tape to the desired file and then invoking READT.

Write Tape Utility (WRITT)

Note WRITT and READT are not supported over DS. Both utilities use data buffers that exceed the 512-word buffer limit of remote I/O mapping. For more information on remote I/O mapping, consult the *DS/1000-IV Theory of Operation and Troubleshooting* manual, part number 91750-90014.

WRITT saves a private or group cartridge to tape and generates a tape-file header to identify the saved cartridge by LU or CRN number, cartridge label, and cartridge type. The saved cartridges then can be restored from the tape to disk using the READT utility.

Calling WRITT

To call WRITT, enter the following runstring:

```
:[RU,]WRITT[,sdisk[,MT:lu[,IH[,DC[,VE[,"..."]]]]]]
```

Sdisk is the disk cartridge to be saved. This parameter can be either a negative LU number or a positive CRN number. The defaults are defined below.

MT:lu is the LU number of the magnetic tape unit on which the cartridge is to be saved. The default is to LU 8. If you enter this parameter using the LU number only (positive or negative), it then must be the second parameter in the runstring.

The following optional parameters can be specified in any sequence:

IH	Inhibits tape rewind. The default is to rewind tape before and after each cartridge restoration.
DC	Disables the overlay check. Unless disabled by this command, WRITT checks to see if a previously stored cartridge file will be overlaid by this operation.
VE	Verifies the mag tape, comparing the data on tape with that on disk.

"..." is a comment to be appended to the tape header and displayed by WRITT during the overlays check, 40 characters maximum.

If the source disk parameter is defaulted and you are under Session Monitor, WRITT saves the first private or group cartridge mounted to your session. If the source disk parameter is defaulted in a non-session environment, WRITT saves the first cartridge in the cartridge list. In either environment, if there are only system cartridges mounted, WRITT issues the following message and exits:

```
ONLY THE SYS MNGR MAY SAVE SYSTEM DISKS
/WRITT: STOP
```

When a cartridge is saved on tape, WRITT creates a file header to identify the cartridge, as WRITT automatically rewinds the tape before and after the save unless the IH parameter is specified to inhibit the rewind. If the end of tape is reached before the entire cartridge is saved, WRITT rewinds the tape, issues the message

```
Please mount another tape
After mounting enter "GO"
```

and waits for you to mount the tape. WRITT increments the tape number and writes the number to the continuation tape, then completes the save operation.

Caution Do not attempt to replace a previously saved cartridge file on a tape as this can corrupt the remaining files on the tape.

Multiple saves to a single tape should be performed in a series of WRITT operations to save the cartridges sequentially on the tape. If you want to add a cartridge save to an existing tape, use the FMGR CN command to position the tape past the last file. Use the IH parameter to inhibit the tape rewind before and after each save. With the rewind inhibited, you must use the CN command (CN,lu,RW) to rewind the tape after the last save operation.

WRITT Examples

Example 1:

Group cartridge 32754 (LU 39) is saved to tape. The mag tape is LU 9, and the comment "RTE-6/VM RELOCATABLES" is added to the header.

```
RU,WRITT,32754,DC,VE,MT:9,"RTE-6/VM RELOCATABLES"
     CR 32754 HPDISK SAVED 3:59 PM FRI., 24 SEPT, 1988 GR
     RTE-6/VM RELOCATABLES
     From LU 39
     /WRITT: Verifying tape
     /WRITT: STOP
```

Example 2:

WRITT is invoked to save private cartridge GU to tape. The DC parameter is not specified so WRITT scans the tape and, since an existing cartridge save file will be overlaid, issues a message identifying the tape file to be overlaid and requesting an OK to continue. The response is NO, and WRITT exits without saving the cartridge.

```
RU,WRITT,GU,VE,MT:9,"GROUP UTILITIES CRN (LATEST SORT)"
     ****** CAUTION ******
     Do you want to
     overlay
     CR 32754 HPDISK SAVED 3:59 PM FRI., 24 SEPT, 1988 GR
     RTE-6/VM RELOCATABLES
     with
     CR GU UTILS SAVED 4:07 PM FRI., 24 SEPT, 1988 PR
     GROUP UTILITIES CRN (LATEST SORT)
     (Yes or No)?
     NO
     *** DISK NOT SAVED ****
     /WRITT: STOP
```

WRITT Error Messages

WRITT can generate the following error messages:

WRIT 001 MAG TAPE DOWN

The requested tape LU is not active. UP the appropriate EQT to enable the device.

WRIT 002 ONLY THE SYS MNGR MAY SAVE SYSTEM DISKS

System disk subchannel 2 and auxiliary disk subchannel 3 can only be saved by the System Manager.

WRIT 003 LU LOCKED

The specified tape unit is locked to another process.

WRIT 004 ILLEGAL MAG TAPE LU

The LU was not specified correctly (was not specified as a number between -63 and 63).

WRIT 005 MT OFF LINE

The tape unit is offline. Press the online switch to bring the unit online.

WRIT 006 NO WRITE RING

The write ring must be installed before you can write to tape.

WRIT OO7 PARITY ERROR

A parity error was detected when writing to the tape. If this error recurs, the tape may be irrecoverable.

WRIT 008 END OF TAPE

The EOT was sensed while writing a cartridge to tape. Mount another tape and enter AB to continue.

WRIT 009 FILE OPEN OR WRIT DISK LU LOCK REJECTED

The cartridge to be saved has an open file or is locked to another program.

WRIT 010 DISK NOT FOUND

The specified disk is not mounted.

WRIT 011 ILLEGAL DISK LU

The specified negative LU number is less than -63, the LU is not included in your Session Switch Table (SST), or the LU driver is not a disk driver.

WRIT 012 ONLY THE SYS MNGR MAY SAVE LU 2 OR LU 3

Only the System Manager can save system disk subchannel 2 or auxiliary disk subchannel 3.

WRIT 013 BAD TRANSMISSION– –DISK TO MEMORY TRK xxx

Transmission is irregular in reading data from disk. If this error recurs, the disk may have a bad track.

WRIT 014 BAD TRANSMISSION– –MEMORY TO MAG TAPE REC xxx

Transmission of data from memory to tape may be faulty. If this error recurs, see the System Manager.

WRIT 016 BAD TRANSMISSION – MAG TAPE TO MEMORY,REC xxxxx

An error was detected in transmission of data from the magnetic tape to memory. If this error recurs, the tape may be faulty. See your System Manager.

WRIT 020 VERIFY ERROR ON TRACK xxxxx

A compare error was encountered when verifying the listed track.

Read Tape Utility (READT)

READT restores a cartridge previously saved on tape by WRITT to a selected disk cartridge. Tracks are reformatted if required to satisfy the track/sector value of the saved file, and the directory tracks are moved if the destination cartridge is smaller than the cartridge from which the data was saved.

Note WRITT and READT are not supported over DS. Both utilities use data buffers that exceed the 512-word buffer limit of remote I/O mapping. For more information on remote I/O mapping, consult the *DS/1000-IV Theory of Operation and Troubleshooting* manual, part number 91750-90014.

Calling READT

To call READT enter the following runstring:

```
:RU,READT[,disk[,MT:lu[,type[,SI:nnn[,IH[,VE/CO]]]]]]
```

Disk is the disk cartridge to which the saved FMP cartridge is to be restored. This parameter can be either a negative LU number or a positive CRN number. The defaults are defined below.

MT:lu is the LU of the magnetic tape unit on which the file is stored. The default is LU 8. This parameter also may be given using the LU number only (positive or negative).

Type is the type of cartridge to be restored: P for private cartridge, G for group cartridge. (System cartridges can only be restored by the System Manager.) If the type specified differs from that in the tape header, the cartridge type given in the command runstring is assigned. The default is to use the cartridge type specified in the tape header. This parameter is meaningless in a non-session environment.

SI:nnn is the size (specified in tracks) of the disk to which the magnetic tape contents are to be restored. The default is the first pool cartridge large enough to accept the saved-cartridge file, and the entire cartridge is mounted. This parameter also can be given without the SI designator, using only the decimal number of tracks required.

The following optional parameters are order-independent and can be specified in any sequence.

IH	Inhibits tape rewind. The default is to rewind the tape before and after the disk cartridge restoration.
VE	Verifies the integrity of the data after the restore operation. The VE and CO options are mutually exclusive; only one may be specified.
CO	Performs a word-by-word comparison of the tape file and a previously restored cartridge.

When you are under Session Monitor and the specified disk CRN is not mounted, READT mounts a cartridge from the cartridge pool and assigns it the specified CRN number; if the disk is specified by LU number, READT mounts that LU if necessary. In a non-session environment, the disk specified by CRN must be a mounted cartridge as there is no non-session cartridge pool. If, however, the disk is specified by LU number, READT mounts that LU if necessary. In either environment, if the cartridge has an open file or is locked to another user, READT returns the appropriate error message and exits.

READT locks the cartridge during the restore operation and unlocks it after the successful restore. Following the restore, the tape-file header created by WRITT and the LU or CRN to which the cartridge was restored is displayed at your terminal, as follows:

```
CR 289 MANUF SAVED 10;45 AM FRI.,18 SEP.,1988

RESTORED TO LU 53
READT: STOP
:
```

If you are going to overlay a currently mounted cartridge, be aware that the type (P or G) of the cartridge to be overwritten cannot be changed and the overlaying cartridge must be at most the same size as the overlaid cartridge (the last track of the overlaid cartridge cannot be moved). When the restore will overlay an existing cartridge, READT issues this message and prompt:

```
DO YOU WANT TO OVERLAY CRN nnn ON LU nnn
WITH CRN nnn (YES OR NO)?
```

If the available disk cartridges are not large enough for the cartridge to be restored, and if by moving the file directory the saved cartridge can be restored to an available cartridge, READT issues the following message and prompt:

```
CRN nnn WAS SAVED FROM A xxxxxx TRACK DISK
LAST TRACK USED IS xxxxxx
WOULD LIKE TO RESTORE TO A xxxxxx TRACK DISK
IS IT OKAY TO MOVE DIRECTORY TRACKS (YES OR NO)?
```

A YES response allows READT to restore the saved cartridge to the available disk. A NO response terminates the utility.

If the saved cartridge file extends to more than one tape, READT issues the message:

```
PLEASE MOUNT SUBSEQUENT TAPE
AFTER MOUNTING ENTER "GO" OR "ABORT"
```

READT checks the first record of the continuation tape to determine if it contains the tape count calculated by WRITT. If the tape count is not in the first record, the MOUNT TAPE message is repeated.

The size parameter specifies the number of tracks required to restore the cartridge. For example, if you specify 203 tracks, READT obtains a 203-track disk LU and places the first directory track at track 202.

If a restore is made to a disk LU that has a sector/track value that is different from the sector/track value on the tape file, READT issues a message indicating that reformatting of the directory is necessary to maintain the integrity of the file structure:

```
TRACKS REFORMATTED FROM xxx SEC/TRK TO xxx SEC/TRK
```

READT then proceeds to restore the cartridge.

When the IH parameter is selected to inhibit tape rewind before and after the restore, the FMGR CN command (CN,lu,RW) must be used to rewind the tape. To position a tape to a particular file, repeat the CN,lu,FF command for each file to be skipped on the tape.

If the verify parameter (VE) is selected, the tape is rewound after the restore operation (using the CN command if the IH parameter is selected) and a word-by-word comparison is made between the tape file and the restored cartridge. If a verify error is encountered, READT issues the following message:

```
VERIFY ERROR ON TRACK nnn
```

The compare parameter (CO) does a word-by-word comparison between the tape file and a cartridge that was restored by a prior invocation of READT. This command does not restore a tape file to the cartridge.

READT Examples

Example 1:

An attempt is made to restore a cartridge with the same CRN as a cartridge already mounted.

```
:RU,READT,289

DUPLICATE CRN LABEL OR LU ALREADY MOUNTED
DO YOU WANT TO OVERLAY CRN 289 ON LU 53
WITH CRN 289 (YES OR NO)?NO
/READT: STOP
:
```

Example 2:

READT is called to restore a cartridge to LU 53, which is already mounted. READT issues the OVERLAY prompt and, following the YES response, overlays LU 53 with the CRN 289 cartridge file.

```
:RU,READT,-53

DISK ALREADY MOUNTED
DO YOU WANT TO OVERLAY LU 53
WITH CRN 289 (YES OR NO)?YES
/READT: CONTINUE

CR 289 MANUF SAVED 10:45 AM FRI.,18 SEP.,1988 PR
RESTORED TO LU 53
/READT: STOP
:
```

Example 3:

READT is called to store a disk cartridge and change the CRN number by specifying another CRN number in the runstring.

```
:RU,READT,1000

CRN 289 MANUF SAVED 10:45 AM FRI.,19 JAN.,1989 PR
CRN 289 HAS BEEN CHANGED TO CRN 1000

/READT: STOP
:
```

When restoring to a disk containing previously restored program files, the following message is issued and the tape is not restored:

```
:RU,READT,-2

CR 2 SYS SAVED 4:00 PM FRI., 4 FEB 1989

THE FOLLOWING PROGRAMS HAVE ID SEGMENTS
POINTING TO FMP TRACKS YOU'RE REPLACING.
THESE PROGRAMS MUST BE REMOVED BEFORE READT WILL
REPLACE THESE TRACKS.

FTN4
RT6GN
MERGE

/READT: STOP
:
```

READT Error Messages

READT can generate the following error messages:

READT 001 MAG TAPE DOWN

The requested magnetic tape LU is not active. UP the appropriate EQT to enable the device.

READT 002 BAD TAPE FORMAT

The tape contains information in a format not restorable by READT. The tape may have been constructed with another utility or with the FMGR DU command, or it may not be the correct volume in a multi-volume tape set.

READT 003 LU LOCKED

The tape device is locked to another process.

READT 004 ILLEGAL MAG TAPE LU

The magnetic tape device was incorrectly specified. It was not specified as a number between −63 and 63, or the driver of the specified LU is not a tape device driver.

READT 005 MT OFF LINE

The tape unit is offline. Press the online switch to bring the unit online.

READT 006 ILLEGAL DISK LU

READT has rejected the use of the specified disk LU. The specified number is less than −63, the driver is not in your Session Switch Table (SST), or the driver of the specified LU is not a disk device driver.

READT 007 PARITY ERROR

The driver detected a parity error when reading from the tape. If this error recurs, the tape may not be recoverable.

READT 008 END OF TAPE

Mount the next tape in the set to complete the restoration.

READT 009 FILE OPEN OR READT'S DISK LU LOCK REJECTED

The disk has a file open or is locked to another program.

READT 010 NO SESSION, LU MUST BE NEGATIVE

You are operating in a non-session environment. A negative LU must be specified; there is no non-session disk pool.

READT 011 SIZE ERROR

The size parameter was not specified correctly or does not specify sufficient tracks to restore the cartridge file.

READT 012 MOUNT ERROR, FMGR xxx

The listed FMGR error was encountered:

FMGR 012	Duplicate label or CRN already mounted. Remove disk or CRN and run READT again.
FMGR 056	The size requested is too large for the disk LU specified. Run READT again with a smaller size parameter.
FMGR 063	The maximum number of disk cartridges has already been mounted. Remove one and run READT again.
FMGR 064	There are no more free LUs in the cartridge pool.
FMGR 065	There is a conflict in SST definition. You are trying to mount a disk LU that has a session LU number assigned to a different device. Check your SST for the LU to which the number is assigned, change the specified number, or choose another LU.
FMGR 066	There is no more room in your SST for an entry.

READT 013 SPECIFIED LU OR FREE LU NOT BIG ENOUGH TO MOUNT CRN

The disk LU specified in the runstring and all available pool cartridges are too small to restore the cartridge file.

READT 014 ILLEGAL RESTORE TO LU 2 OR LU 3

Only the System Manager can save system disk subchannel 2 or auxiliary disk subchannel 3.

READT 015 PRIVATE/GROUP OPTION INCOMPATIBLE WITH EXISTING CARTRIDGE

The P/G attribute of the cartridge tape file conflicts with the attribute of the mounted cartridge. Run READT with the correct P/G option, or mount a compatible cartridge.

READT 016 BAD TRANSMISSION – MAG TAPE TO MEMORY, REC xxxxx

An error was detected in transmission of data from the magnetic tape to memory. If this error recurs, the tape may be faulty. See the System Manager.

READT 017 INTERNAL BUFFER TOO SMALL!

The tape records are longer than the READT internal buffer. The buffer size must be increased and READT must be reloaded.

READT 018 ABORTED BY USER

This message is produced when you respond NO to any prompt or when READT is halted using the BReak command.

READT 019 DISK ERROR ON LU xx, TRACK xxxxx

READT encountered an error when reading the listed track of the listed LU.

READT 020 VERIFY ERROR ON TRACK xxxxx

A compare error was encountered when verifying the listed track.

READT 021 INVALID PARAMETER

An invalid parameter was specified in the READT command runstring. Check the runstring and re-enter the parameter.

6

File System Utilities

This chapter describes the file system utilities provided for use in the CI file system: FSCON, FPACK, MPACK, FREES, FOWN, and FVERI. FSCON converts the directory structure of an FMGR cartridge to that of the CI hierarchical directory structure. FPACK packs a CI disk volume to increase disk free space. MPACK compacts files and packs disks to enlarge free space areas. FREES and FOWN report the amount of free space on the disk and the amount of disk space used by file owners. FVERI verifies the validity of the disk volume directories and tables.

File System Conversion (FSCON)

FSCON converts the directory structure of an FMGR cartridge, creating a hierarchical directory entry for each file on the cartridge. After conversion, the files have all the characteristics of the CI file system (time stamps, file type extensions, and so on). Note that unconverted programs may have difficulty accessing the converted files.

Calling FSCON

To call FSCON, enter the following runstring:

```
CI> RU,FSCON,LU
```

LU is the LU of the FMGR cartridge to convert. If you do not enter the LU, FSCON issues a message defining the correct runstring and terminates.

Requirements for Successful Conversion

Before beginning the conversion, FSCON checks to see if all the conditions described below are met; if not, FSCON terminates with an error message.

There must be sufficient free space between the last file and the FMGR directory at the end of the cartridge to create the new directory and free space table. The amount of space required depends upon such factors as the number of files in the directory and the number of extents. You can usually free enough space by purging unneeded files and using the FMGR PK command to pack the disk before calling FSCON. If there is not enough disk space for the conversion, FSCON exits with the following message:

```
Not enough free space on disk
```

The total size of the cartridge cannot exceed 128k blocks. This is due to a limitation on the size of the free space table. If the FMGR disk cartridge exceeds this size, FSCON terminates with the following message:

```
Disk too big to convert
```

The disk must be dismounted before conversion to preclude the possibility of any open files, swap files, or active type 6 files. If the disk cartridge is mounted, FSCON terminates with the following message:

```
Cartridge must be dismounted
```

FSCON only converts FMGR cartridges. Any other cartridge causes the utility to terminate with the following message:

```
Doesn't look like an FMGR disk
```

The Conversion Process

FSCON scans the directory on the given LU and builds a new directory in the unused space at the end of the disk, before the FMGR directory. At the same time, it builds a free space table. The FMGR directory is scanned once to determine whether it is possible to do the conversion and scanned again to build the CI directory.

File data is never moved during the conversion process; only the directory structure changes. The new directory is built in unused space, and the entire conversion is done before any change is made to the FMGR directory structure. Thus, if the conversion fails at any point short of completion, the FMGR directory is still in place and all the files remain unchanged.

The final step in the conversion process is to overwrite the FMGR directory with the hierarchical CI directory. This is done in a single disk write operation. The new directory name is the CRN of the FMGR cartridge. In this way, the name of a file remains unchanged (except as noted in the following section). For example, a reference to &SORC::DB accesses the same file before and after the conversion.

After the successful conversion, FSCON informs you “Cartridge converted”. If an error occurs during conversion, FSCON issues the appropriate error message, followed by “Cartridge not converted”, and terminates.

File Renaming

If a file name includes a slash (/) or a period (.), those characters are changed to “| ” and “ ~ ” respectively. This is necessary because in the CI file system, a slash delimits directories and subdirectories, and a period delimits the type extension and mask qualifier; therefore, a file name containing those characters would not be found. As each file name is changed, the following message is displayed:

```
Renaming <name> to <name>
```

After conversion, you may list the files with the DL command. For example:

```
DL,@~@::dir *list all files with "." changed to "~"

DL,@|@::dir *list all files with "/" changed to "|"
```

Converted CI Directory Entries

When a new CI directory is built, information required for the directory entries is obtained from the FMGR cartridge directory, from scanning the files or from the default values. The entries of a newly converted CI directory contain the following information (refer to the *RTE-6/VM System Manager's Manual*, part number 92084-90009, for the format of the file directory entry):

Word			Meaning
1	–	flag:	protection set to rw/rw for all files and for directory; backup bit (bit 8) set
2	–	type:	taken from FMGR file directory entry
5	–	size:	taken from FMGR file directory entry
6	–	recln:	type 1, 2 files taken from FMGR file directory entry; all other files calculated by scanning file
9–16	–	name:	taken from FMGR file directory entry as modified to change special characters
17–18	–	ext:	extent type, blank
19–24	–	time:	create, access, update set to current time
25–26	–	nblk:	type 1, 2 files taken from FMGR file directory entry; all other files calculated by scanning file
27–28	–	eof:	type 1, 2 files taken from FMGR file directory entry; all other files calculated by scanning file
29–30	–	nrec:	type 1, 2 files taken from FMGR file directory entry; all other files calculated by scanning file

FSCON Error Messages

If any of the following errors occurs, FSCON issues the related message and terminates:

Cartridge not converted

This message appears with a definitive message if an error occurs during conversion. If the cartridge is not converted, the files are all intact and the FMGR directory structure is unaltered.

Bad LU parameter

The LU parameter is not a disk LU.

Cartridge must be dismounted

The cartridge to be converted must be dismounted, which guarantees there are no open files, active type 6 files, or the swap file on this cartridge.

Disk too big to convert

Disk has more than 128K blocks.

Doesn't look like an FMGR disk

FSCON only converts FMGR disks into CI file system disk volumes.

Insufficient memory, size up program

FSCON uses free space for tables and runs faster with more memory. Sufficient memory for at least one track of the disk is required. If the LU contains many type 1 or type 2 files with extents, more free space is required. Resize the program.

Not enough free space on disk

FSCON requires sufficient free tracks between the last file and the FMGR directory to build a new directory and other tables. Correct this condition by purging some files and packing the disk, using the FMGR PK command, before trying the conversion again.

Open file: xxxxxx

This message should never appear, because the cartridge must be dismounted before the conversion begins. However, FSCON checks for open flags on files as it converts them.

File System Pack (FPACK)

FPACK rearranges the files on a disk volume, packing the files together more tightly to increase the size of the largest free space on the volume. When the operation is complete, there is usually free space at the high end of the volume. After the disk volume is packed, you can run the FREES utility to determine the amount of free space and the largest area of free space.

If the disk volume can be dismounted during the packing process, then MPACK should be used instead. Because MPACK dismounts the volume, it can move all files and directories. Thus it can do a more thorough pack and is much faster than FPACK. See the next section for more information on the MPACK utility.

Calling FPACK

To call FPACK, enter the following runstring:

```
CI> RU,FPACK,LU
```

LU is the LU of the volume to be packed. If you do not enter the LU, FPACK issues a message defining the correct runstring, then terminates.

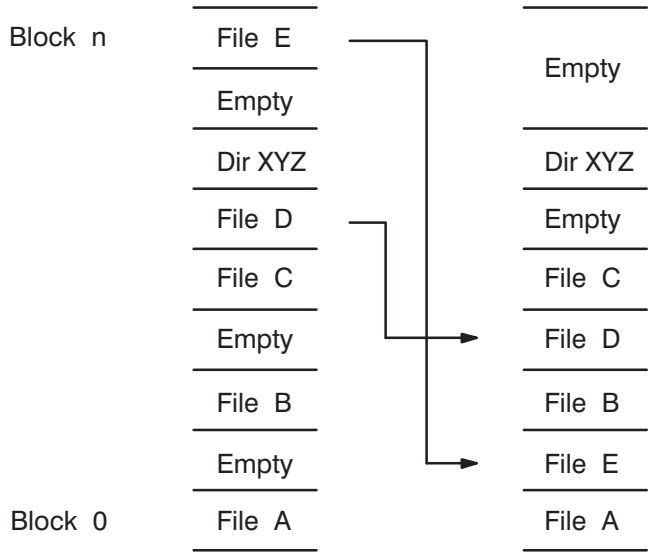
The Packing Process

FPACK first scans the directories and generates a list of files in the order of their location on the disk volume. Then FPACK copies files from higher numbered blocks to free spaces in lower numbered blocks on the disk, purges the original files, and marks the original file blocks as free space. A file is copied only if there is an area of free space below it that is large enough to contain the file and its extents. When a file is copied, any extents are copied into the main. All other attributes of the file (time stamps, protection, and so on) are not changed.

Note that to copy a file, you must have read/write access to the file and to its directory; generally, only a system manager has access to all files and directories.

Because the integrity of a directory cannot be guaranteed if it is moved, FPACK does not copy directories. Open files, type 6 files, and the swap file are also not copied.

The process is illustrated below. Assuming that all files are the same size, FPACK converts the disk volume structure on the left into the one on the right.



File E, the file in the highest numbered block, is copied into the free space in the lowest numbered block. FPACK skips the directory, then copies File D. This process frees one area of space at the top, enlarging the available free space area on the volume. An area of free space still remains below directory XYZ.

After FPACK moves as many files as possible, it scans the LU again and issues an ordered listing of the files that could not be moved, beginning at the highest disk volume address (a maximum of ten files are listed). The files listed are generally those that cannot be copied—type 6 files, open files, directories, and the swap file. The entry for each file includes name, directory, file type, size, and record length.

After printing the list, FPACK exits, and you may run FREES to see the amount of free space that now exists on the disk volume and the size of the largest free space area. (Refer to the description of FREES provided later in this chapter for the utility output format.)

If you still do not have enough free space, you can gain more by moving the appropriate files. For example, you can move down directory XYZ (shown above) to increase the largest free space area. The procedure for moving the directory is provided in the next section. If file C can then be moved, there is even more space in the largest area. (If file C cannot be moved, moving file D does not help.)

Alternatively, you can purge some existing files from the disk volume and run FPACK again to move files into the now empty space.

If you still lack enough space, back up the entire volume on tape, using the TF utility (described in Chapter 4 of this manual); then, reinitialize and restore the volume from tape. The CI file system automatically restores files beginning at the lowest numbered block on the disk volume. Note that if you reinitialize the system volume, you must be able to boot the system from another volume.

Moving Directories

The following sequence of CI commands moves the global directory XYZ to a different disk location. The leading slash in the command argument indicates the named directory is a global directory.

1. Change the working directory to XYZ. This simplifies the command sequence.

```
CI> WD /XYZ
```

2. Create the temporary directory TEMP. The directory is created on the same disk volume as the working directory, at the lowest numbered block available.

```
CI> CRDIR /TEMP
```

3. Move all the file entries from XYZ to TEMP. Only the directory entries are moved; the file data is unaffected. As each directory entry is moved, a message defining the entry is issued to your terminal. A successful move is indicated with the notation [ok]. If an entry cannot be moved, the notation [failed] is given, followed by the reason (often an open file on XYZ). If you cannot correct the failure and move the file, move all the files back to directory XYZ (mo /temp/@.@ /xyz/@.@) and stop the attempt.

```
CI> MO @.@ /TEMP/@.@
```

4. Change the working directory to TEMP.

```
CI> WD /TEMP
```

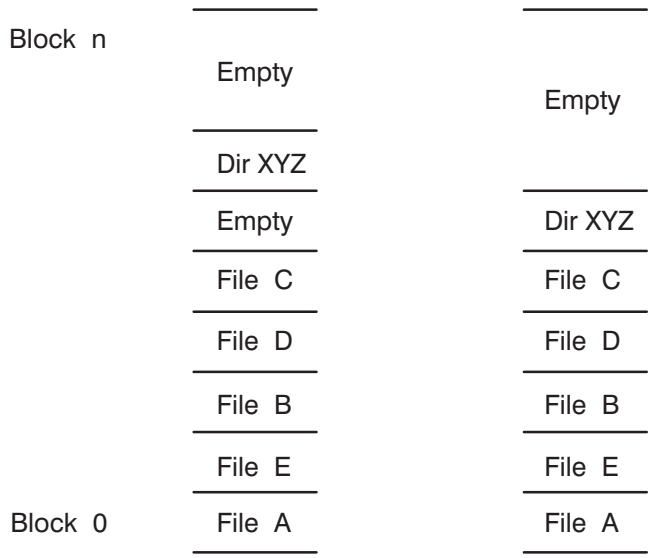
5. Purge the old directory, freeing the disk space. (Note that you must include the type extension .DIR when purging a directory.) If the directory is not empty, it cannot be purged. Move the files back to XYZ, purge TEMP, and terminate the action.

```
CI> PU /XYZ.DIR
```

6. Restore the original name. The files now have the same name as before, but the directory has moved.

```
CI> RN /TEMP /XYZ
```

After this operation, the example disk volume structure is converted, as follows:



Moving Subdirectories

You can move subdirectories the same way; however, you must specify the hierarchical path, as:

```
CRDIR /XYZ/TEMP.DIR
MO /XYZ/SUB/@.@ /XYZ/TEMP/@.@
PU /XYZ/SUB.DIR
RN /XYZ/TEMP.DIR /XYZ/SUB.DIR
```

The concept is the same: Create another subdirectory, move all the files into it, purge the original, and rename the new subdirectory to the original name.

Moving Files

When you use the CO command to copy a file to a disk volume, the file is placed in the lowest free space large enough to hold it. This lets you move files on a disk volume to recover free space. If you copy a file to a different name on the same directory, the new copy is moved to a lower free space.

For example, using the original example in this section, you can move file E to the lowest free space with the following command sequence (assuming file E is on directory XYZ).

1. Change the working directory to XYZ to simplify the command sequence.

```
CI> WD /XYZ
```

2. Copy file E to file X (which does not exist). File X is placed in the lowest free space available.

```
CI> CO E X
```

3. Purge the original file.

```
CI> PU E
```

4. Restore original file name.

```
CI> RN X E
```

When this is done, the example disk volume structure has a larger free space.

File Compacting and Disk Pack (MPACK)

MPACK has three major functions: It compacts files, packs a disk, and reports file block and extent usage. When MPACK compacts files, it removes unused blocks and extents from the files. When it packs a volume, all of the disk's free space becomes one large block. This provides you with faster file data access and more efficient disk space usage. MPACK reports "wasted" blocks (blocks that were allocated to a file but remain unused) and extent usage.

MPACK only works with hierarchical files and hierarchical file volumes.

Calling MPACK

To call MPACK, enter the following runstring:

```
CI> RU,MPACK [options] mask [options]
```

The mask may be any legal FMP mask form (including a standard file descriptor). However, if you want to use the Pack (P) option, the mask must be an LU number.

The MPACK options are discussed below.

MPACK Options

MPACK provides compacting, packing, and logging options. All options start with "+" and may be entered anywhere in the runstring, but only once. The options are summarized in Table 6-1 and discussed in the sections that follow the table.

Note that when you use MPACK, paging output to the screen occurs, but only if you are not doing any file or disk manipulation and you have not selected logging. In other words, you are not prompted every page of output to continue the listing if logging, file compacting, or disk packing is selected.

Table 6-1. MPACK Options Summary

Options		Description
Compacting and Reporting Options		
+R		Remove extents from files
+T	[n[%]]	Truncate wasted blocks from files
+C	[n[%]]	Remove extents and truncate wasted blocks
+E	n	Select files with <n> or more extents
+W	n	Select files with <n> or more wasted blocks
+W	n%	Select files with <n> % or more wasted blocks
+A		AND the +E and +W qualifiers
+D		Include directories in the compacting
+Q		Quiet mode – only report totals
Packing Options		
+P		Pack the disk LU
+OK		OK to overlay data during pack
Logging Option		
+L	file	Log output to a file

Compacting and Reporting Options

The +R, +T, and +C options all specify how a file is to be compacted (refer to Table 6-1 above). These options let you remove unused ("wasted") blocks from files and extents. The +D option causes directories to be compacted.

The +En, +Wn, and +Wn% options use the block and extent information to qualify the selection of files for reporting or compacting and are therefore referred to as "qualifiers."

You must enter a mask in the runstring to let MPACK know which files to select. If a file that matches a specified mask also fits the qualifiers, reporting or compacting is performed. If you do not enter any compacting or packing options in the runstring, only the reporting occurs.

Remove Extents from Files (+R)

The +R option lets you recreate files that currently have extents with no extents. Type 1 and type 2 files cannot have their extents compacted, since extent sizes may contain some significance for those types. The only exception to this rule is for directories; MPACK removes extents from directories whenever possible. A directory's main size never exceeds 64 blocks, so extents may still remain after being compacted. The +D option must be specified for MPACK to remove extents from directories.

The format for using +R is:

```
mpack <mask> [options] +r
```

Truncate Unused Blocks from Files (+T)

The +T option lets you truncate type 3, 4, 5 and 7 files with wasted blocks up to the EOF. When you enter +T, all the unused blocks are truncated from the file. If you enter +Tn, <n> number of unused blocks remain in the file after truncation. If you enter +Tn%, <n> percent of unused blocks remain in the file after truncation.

Since truncating never adds blocks to a file, if the number of unused blocks is smaller than the amount specified to remain, the file is unchanged.

Since truncating never shortens an extent, if truncation occurs within an extent, the entire extent is left unchanged and wasted blocks still exist.

Note that if you specify both +T and +R, +R is performed first so complete truncation can occur.

The format for using +T is:

```
mpack <mask> [options] +t[n[%]]
```

Compact Files (+C)

The +C option behaves like the +R and +T options combined. When you specify +C, you may not use either +R or +T in the same runstring. The +Cn and +Cn% usage is the same as that for the +T option.

The format for using +C is:

```
mpack <mask> [options] +c[n[%]]
```

Extent Count Qualifier (+E)

The +E option lets you select files whose extent count is equal to or greater than <n>.

The format for using +E is:

```
mpack <mask> [options] +En
```

Wasted Block Count Qualifier (+W)

The +W option lets you select files whose unused block count is equal to or greater than <n>.

The format for using +W is:

```
mpack <mask> [options] +Wn
```

Wasted Block Percent Qualifier (+W%)

The +W% option lets you select files whose unused block percent is equal to or greater than <n>.

The format for using +W% is:

```
mpack <mask> [options] +Wn%
```

ANDing Option (+A)

By default, the +E, +W, and +W% qualifiers have an ORing functionality when entered together in a runstring. In other words, a file is selected if it matches any one of the qualifiers. The +A option causes an ANDing functionality. In this case, files must match all of the qualifiers in order to be selected.

The format for using +A is:

```
mpack <mask> [options] +a
```

Include Directories (+D)

By default, when MPACK is reporting file block and extent usage, MPACK does not report information on directory files. Also, when MPACK is compacting files (+C, +T, +R), MPACK does not alter any directory files. If the +D option is specified, MPACK treats the directory files in the same manner as all other files.

The format to report file block and extent usage for files and directories is:

```
mpack +D <mask>
```

The format to compact both files and directories is:

```
mpack +C +D mask
```

Quiet (+Q)

When a file is selected, MPACK produces an informational listing for each file and a total of the results. The +Q option suppresses the individual file information and displays only the totals. This is useful when only the final results are of interest.

The format for using +Q is:

```
mpack <mask> [options] +q
```

Packing Options

Disks almost always develop multiple unused data areas scattered throughout. When this happens, you may find that although the total amount of free space on the disk would be sufficient for your needs, there is no single area large enough for the task you are doing.

When you use the +P option, MPACK "pushes" all the files on a disk together to replace scattered areas of free space with one large, contiguous data area. The only exception is on the last track, where the volume header must remain, so the space following the volume header remains separate. The +P option is described below.

Except for the volume header, MPACK can and may move everything on the disk to a new location. In order to do this, the LU cannot be connected to any paths, working directories, RP'd files, or anything else. Since MPACK must have the disk all to itself, the disk LU being packed will be dismounted and locked by the program. This means that during the pack operation, the disk is unavailable to all other users. When MPACK finishes packing the disk, the program remounts the LU.

Because MPACK is moving around file data and adjusting pointers on the disk, it should never be OF'd while in process. If you need to abort the program, use the BR system break command.

MPACK uses the volume's bit map to determine how the pack operation should proceed. Many precautions are taken to preserve data integrity. Occasionally, when data needs to be moved but there is no free area large enough to accommodate the data, MPACK cannot fully pack a volume. MPACK continues to do what it can, but at the end of its pass through the bit map, all the unused areas may not be successfully combined.

Since at least a partial pack takes place when this happens, a new free area may be created during the packing process that is large enough for the data that was not moved. MPACK makes a second pass to see if a full pack is now possible. Additional passes are taken until a full pack occurs or nothing was moved on the last pass. MPACK reports on each pass to let you know what is happening and why the operation is taking longer than usual.

Normally, MPACK does not copy data onto itself. However, the +OK option enables MPACK to overlay data. When you select +OK, only one pass is ever done or required for a full pack to occur. See the discussion of the +OK option below for more information.

Pack the Volume (+P)

This option causes the volume to be packed. You must enter the LU number as the mask. If you also enter compacting options in the runstring, compacting occurs before packing. This provides for the maximum return of continuous free disk space.

The format for using +P is:

```
mpack <mask> [options] +p
```

OK to Overlay Data (+OK)

The +OK option gives MPACK permission to copy data onto itself if it cannot be moved to another area on the disk. The danger of having this corrupt your data only occurs if the system goes down or if MPACK is aborted during an overlay. To help preserve data integrity in such situations, whenever an overlay occurs, MPACK reports on which blocks are being overlaid and where and when the overlay is done. This lets you know if you need be concerned about data integrity and what steps to take to correct any problem that arises.

The format for using +OK is:

```
mpack <mask> [options] +p +OK
```

Logging Option

MPACK provides a logging option that you can select to direct output to a log file. Compacting and packing operations both support logging.

Logging (+L)

+L causes all the output that is displayed on the screen to go to the log file. When you specify logging, output to the screen is not paged (in other words, the listing does not stop every 22 lines with a “More...” prompt).

If the file name already exists, MPACK fails to run. Logfiles are not overwritten.

The format for using +L is:

```
mpack <mask> [options] +1 <log filename>
```

MPACK Examples

Example 1: **Remove extents from all files on the working directory.**

```
mpack @ +r
```

Example 2: **Truncate wasted blocks from files on disk LU 20.**

```
mpack +t 20
```

Example 3: **Remove extents and truncate wasted blocks, leaving 10 unused blocks remaining, for any file in the working directory or below that has at least two extents or 30 unused blocks.**

```
mpack +c10 @.@.s +e2 +w30
```

Example 4: **Remove extents and truncate wasted blocks from every file on LU 15 that has 100 wasted blocks and the wasted block percent is equal to or greater than 20 percent of its block usage, and then pack the disk.**

```
mpack +c 15 +w100 +w20% +a +p
```

FREES– Indicate Free Space on a Volume

FREES scans the free space table on hierarchical file system disks and reports the amount of free space and the size of the largest free space. The amount of free space on a volume is an indication of how much more data will fit on the volume, while the largest free area defines the largest file that can be created on that volume without first packing the disk. Running FREES with a ? or ?? (RU,FREES,?[?]) displays usage information and examples.

Syntax:

```
[RU] FREES [options] [DiskLu] [DiskLu:DiskLu] ...
```

options one or more of the following:

−v Inhibit the inverse video bar graph.

+l:lu Send the listing to the given LU.

−g Bar graphs are not relative to the largest disk.

disksz where *disksz* is one of the following:

+t Report disk size in number of tracks.
+m Report disk size in Mbytes.

sort where *sort* is one of the following:

−s Inhibit sort, report in cartridge list order.
+h Sort by largest hole size.
+d Sort by disk size.

quiet where *quiet* indicates quiet mode and is one of the following:

+q return all free space information to the scheduling program in $return_s.
+qd return only disk size in $return_s.
+qf return only free size in $return_s.
+qh return only largest hole size in $return_s.
+qr return only reserved size in $return_s.
+q%f return only percent free in $return_s.
+q%m return only percent max in $return_s.

DiskLu Disk LU to display; it can be repeated.

DiskLu:DiskLu
Disk LUs to display; if no disk LU is specified, all CI volumes are listed.

Return values:

$return1 :	= 0	frees executed successfully.
	<> 0	an error was encountered during the scan of a disk LU
$return2 :	the number of disk volumes successfully examined.	

The following return variables are set when quiet mode is enabled.

$return3 :	disk LU (when reporting on a single disk LU).
$return4 :	high order bits (31-16) of the value returned in $return_s.
$return5 :	low order bits (15-0) of the value returned in $return_s.
$return_s :	this string contains all or part of the free space information depending on the "+q" option selected.

Examples:

`frees`	show free space information on all mounted volumes.
`frees 54`	show free space information on volume 54.
`frees 54 22`	show free space information on volumes 54 and 22.
`frees 10:15`	show free space information on volumes 10 through 15.

At the user's option, FREES expresses sizes in number of blocks (the default), number of tracks (+t option), or in megabytes (+m option).

At the user's option, FREES sorts the disks by descending free size (the default), by largest hole size (+h option), or by total disk size (+d option), or the sort can be inhibited to list the disks in cartridge order (−s option).

FREES reports the total free space as an absolute number and also as a percentage of total space on the volume. The only way that this number can be increased is to purge unnecessary files from the disk.

The size of the largest free area (the hole size) is reported as an absolute number and also as a percentage of total free space on the volume. The lower this percentage, the more fragmented the volume is and the more effective the FPACK or MPACK utilities will be in packing the volume.

Following the space table is a short summary that totals all the disk space and free space on the scanned disks.

Unless inhibited by the −v flag, FREES overlays the columnar output with a bar chart to indicate graphically the amount of free space. FREES has two methods of bar chart display, Global (default) and Local (−g option). In the Global mode, the widths of the bars are shown proportional to the largest disk that was scanned. Think of it as a form of autoscaling. In the Local mode, the widths of the bars are computed relative to the individual disks. Note that if only a single disk is displayed there is no difference between Global and Local modes.

In the bar chart, the total width of the bar is proportional to the total free space, while the highlighted portion is proportional to the amount of fragmentation. Here is an example:

```
CI>ru,frees 10 11

LU
---+----+----+----+----+----+----+----+----+----+----
11                          80000    40000    80   50
10   10000       1538       8000     4000     80   50

%Free     = free space as percentage of total space.
%Max      = largest free space as percentage of total free space.
Total storage on all disks scanned is 110000 blocks; 88000 free blocks; 80% free.
```

Note that although there is 80% free space on both disks, the bar for LU 10 is much shorter than the bar for LU 11 because the default Global mode was used. For both LUs, half of the bar chart is highlighted because the largest free chunk is 50% of the total free space.

```
CI>ru,frees 10 11

LU
---+----+----+----+----+----+----+----+----+----+----
11                          80000    40000    80   50
10                          8000     4000     80   50

%Free     = free space as percentage of total space.
%Max      = largest free space as percentage of total free space.
Total storage on all disks scanned is 110000 blocks; 88000 free blocks; 80% free.
```

In this example, both bars are the same width because this is a Local mode display and both LUs have the same percentages of free and fragmented space.

When the +l option is used to send the output to a printer, the −v option is automatically invoked and a form feed is generated at the end of the report. The −v option is also automatic if the driver type for the user's terminal is not type 5 (HP protocol).

Report File Space by Owner (FOWN)

FOWN scans specified files and identifies, by owner, the total disk space used by files that match the mask given in the utility runstring.

Calling FOWN

To call FOWN, enter the following runstring:

```
CI> RU,FOWN[,mask]
```

The mask parameter specifies the class of files to scan. Refer to Chapter 4 in this manual for a description of file masks. The default mask is /@.@.s, which matches all files in the file system. This tells how much disk space is being used by all users who own files.

Occasionally FOWN cannot identify a file's owner. When this happens, FOWN displays the system number that corresponds to the owner. Usually, this means the owner is no longer a user on this system. When FOWN cannot determine the owner of a remote file, they are referred to only by number.

FOWN Examples

Example 1: The default mask is used. User number 7 is no longer known to the system.

```
CI> FOWN
Scanning...    Mask = /@.@.s

  Owner          Disk Blocks   Number of files
DOUG.NOGROUP           6715       1065
DON.NOGROUP            4032        843
MANAGER.SYSTEM         2761        404
DOUGL.NOGROUP           328          3
NAOMI.NOGROUP            69          2
Unknown (#7)            198         58
                       -----       ----
Total                  14103       2375
```

Example 2: All type 6 files are counted and their owners identified. The message "FMGR files not scanned" indicates that some FMGR files matched the mask but were not counted. There is no ownership information available for FMGR files.

```
CI> FOWN,@.@.e:::6
Scanning...    Mask = @.@.E:::6

  Owner          Disk Blocks   Number of files
DOUG.NOGROUP           2943         362
DON.NOGROUP             433          48
DOUGL.NOGROUP           143           3
                       ----         ---
Total                  3524         413

FMGR files not scanned
```

Example 3: Ownership ID requested for directory /DON only, thus the format of the output is changed.

```
CI> FOWN,/DON/
Scanning...    Mask = /DON/
Owner: DON.NOGROUP Total Blocks: 641 Number of files: 86
```

File System Verification (FVERI)

FVERI has two functions. The first function is to verify a hierarchical file system disk LU by scanning the directory and table structures and reporting any inconsistencies. The second function is to fix some of the inconsistencies found by the verification.

Two types of checks are done:

- Internal consistency of the directories and correctness of the bit map (the bit map is a table on the volume that keeps a record of used and free space on that volume).

  FVERI checks directory consistency by looking for legal values within the entries: extent pointers to valid extents, data pointers having legal disk addresses, non-negative file types, and so on.

  FVERI checks the bit map by building its own version of the bit map (based on the files it finds on the volume) and then comparing the two bit maps.

- Consistency between directory entries and their associated files (for example, the number of records in the file and number of words in the file) and internal consistency of the files (valid record length, EOF marks, and so on).

  FVERI checks the consistency between directory entries and their associated files and checks internal consistency of the files by reading through each file it finds (using normal FMP calls). It collects appropriate data as it reads (the number of records in the file, number of words, and so on), then compares what it finds with the directory entry for that file.

Three types of fixes are done:

- Inconsistencies exist between directory entry information and the data actually within the file. For file type 3, 4, 5, 7, and up, directory information may become inaccurate because of misuse of the file system. Note that the user must have appropriate capability to change the directory and file for any of these fixes.

  FVERI can correct an erroneous word count (end-of-file position), record count, and record length that is kept in a file's directory entry.

- Illegal directory entries (entries not recognized by the file system) exist. Unrecognized entries within a directory can confuse particular functions of the file system.

Note Such directory problems may be a sign of a bigger problem within the file system. For example, an overwritten directory may have been caused by a corrupt bit map. Use this fix carefully. Only superusers can use this fix, and the disk must be dismountable.

FVERI can change an illegal directory entry to look like a purged entry so that the file system will not be confused by it.

- The bit map is corrupt. Occasionally a bit map can become corrupt when the directory structure is still intact. For “free space marked used” there will be space on the disk that will never be accessible. For “used space marked free” there is data loss possibility. FVERI cannot correct a duplicate use of space problem.

Note Such directory problems may be a sign of a bigger problem within the file system. Use this fix carefully. Only superusers can use this fix, and the disk must be dismountable.

FVERI can overwrite the bit map on the disk with the bit map that was created by scanning the directory structure.

If an LU is specified in the runstring and the +B option is not given, FVERI will perform both types of verifications. With the +B option, FVERI will only perform the first type of verification. With a given mask, FVERI will verify all files or subdirectories in the directory described by the mask, performing only the second type of verification. Because the bit map covers the entire volume, the bit map verification cannot be done if a specific mask is specified because FVERI may not be looking at all the files on the volume.

Note that because the second type of verification requires that FVERI read through every file, it will be slower than the first. For this reason, if you want to verify the overall integrity of a disk volume but do not need to verify each individual file, it will be faster to use the +B option. On the other hand, if you are concerned about file integrity on a certain area of the volume, specifying a mask can still be relatively fast because you are not asking FVERI to verify the whole volume.

FVERI can be used after a system crash to verify that the file system is still intact. It should also be run from time to time to verify the integrity of the file system. In order to gain access to all read-protected files on the disk, FVERI is best executed by a System Manager (superuser).

FVERI is most efficient when verifying a disk not in use. If other programs are creating, purging, or modifying files while FVERI runs, the tables will appear inconsistent due to synchronization errors. For instance, if a file is purged after being verified but before the bit map is verified, the bit map directories may appear to be invalid. For this reason, when FVERI is asked to fix directories (+FD) or the bit map (+FB), it will first dismount the LU, lock it, and then remount it. This method guarantees that no one else has access to the disk, so FVERI’s data is reliable.

FVERI does not verify or fix FMGR cartridges.

Operating Instructions

To run FVERI, enter the following runstring:

RU FVERI [lu/mask] [options]

where:

lu = LU of volume to be verified.

mask = Mask describing a directory in which all files and/or subdirectories are to be verified.

options = One or more options in any order.
Legal options are:

+L,file|lu − list file or device for errors
+B − verify bit map only; illegal if a mask is also specified
+FB − fix the bit map
+FD − fix illegal directory entries (set purged)
+FF − fix file directory information
+OK − perform fixes without asking each time

If neither an LU nor a mask is given, the default is to verify all mounted volumes. For each volume, the message:

Verifying LU xx

is issued while the verification is taking place. FVERI can be halted before completion with a BR,FVERI. The utility quits immediately without completing a verification in process. When performing a complete verification of a volume, FVERI can take several minutes to run because it reads every variable length record file on the LU.

Help Command (?)

Purpose: Displays information about using FVERI.

Syntax: `RU,FVERI,?[?]`

Running FVERI with a ? or ?? (RU,FVERI,?[?]) displays usage information as shown above.

Error Recovery

There are several possible responses to inconsistencies detected in the file system. Some problems may be related to, or caused by, others so recovery should be done with caution. For example, invalid data pointers will probably cause bit map inconsistencies. In this case, fixing the bit map could allow lost data to be overwritten, and all opportunity for recovery would be gone.

- The fix options in FVERI can correct some of the problems. Directory information about a file, invalid directory entries, and corrupted bit maps can all be corrected.
- Some directory problems can be corrected by creating a new directory on the same disk, moving the troubled directory's files into the new one, purging the old directory, and then renaming the new one.
- Some directory and bit map problems can be corrected by doing a logical backing to tape or disk, initializing the corrupted LU, and restoring the files. This can be time consuming, and some of the logical backup utilities (TF/FST) use the word count (EOF position) to determine how much of a file to back up. Make sure no "number of words" errors exist on the LU before doing the backup.

FVERI can fix inconsistencies related to particular errors using specific options (the number in parentheses preceding each error message indicates the severity of the error).

- Error messages that can be fixed with +FF

  (3) Record length incorrect at block <nnnn>

  (3) Number of words in file incorrect at block <nnnn>

  (3) Number of records in file incorrect at block <nnnn>

- Error messages that can be fixed with +FD:

  (5) Unidentifiable directory entry at block <nnnn>

- Error messages that can fixed with +FB:

  (4) Free space is marked as used space

  (9) Used space is marked as free space

Error Messages

FVERI is susceptible to all FMP errors, which are reported as they occur. It also reports any errors detected in the table structures. Each error message is preceded with a number indicating the relative importance of the error. Most FMP errors are displayed as severity level 4; some, usually protection violations, are reported with a severity level of 0. The higher the number, the greater the severity of the problem. Continued use of an LU with a reported level 9 error can cause loss of data.

The error message usually includes a block number indicating the block on the disk where the bad data was found. This is either the block number of the directory entry for some file or the block on the disk represented by the invalid data in the bit map.

Where possible, the error output will indicate the file whose directory entry is corrupt. This gives some clue to the logical part of the disk that is corrupt and complements the physical location information provided by the block number.

As an example, the following message defines a level 6 error in the directory entry for file FOO.TXT : : JOE. This directory entry is at block 1435 of the disk.

```
(6) Total blocks in the file less than main size at block 1435
    On file FOO.TXT : : JOE
```

The following list of error messages is arranged in ascending order of severity, from 0 to 9. The first four level 0 errors cause FVERI to terminate; all other error conditions are not fatal.

(0) Break detected; verification terminated.

(0) Internal buffer too small, size up program.

(0) Not a hierarchical file system disk.

This error occurs if the volume is not a CI disk or the volume header is corrupt.

(0) Disk volume not mounted.

The volume must be mounted to be verified. If you cannot mount the disk, FVERI cannot give you any further information.

(0) Record Length exceeds 512 bytes at block <nnnn>.

FVERI has an internal buffer of 512 bytes for reading type 3 and above files. A record of more than 512 bytes was read, and further verification of record length or word count will not be done for this file.

(2) Directory Tag field is incorrect at block <nnnn>.

A special 32-bit tag is set for directories as a redundancy check to verify that this is, in fact, a directory. This directory does not have the proper tag value.

(3) Record Length Incorrect at block <nnnnn>

The record length field in the directory does not reflect the length of the longest record in the file. This message can be caused by another program having the file open, the file being created in a non-standard way, or misuse of the FMP calls. Note that FVERI can fix this problem by use of the '+FF' option.

(3) Number of words in file incorrect at block <nnnnn>

The End-Of-File pointer in the directory does not match the End-Of-File mark in the file. This message can be caused by another program having the file open, the file being created in a non-standard way, or misuse of the FMP calls. Note that FVERI can fix this problem by use of the '+FF' option.

(3) Number of records in file incorrect at block <nnnnn>

The record count in the directory does not match the number of records in the file. This message can be caused by another program having the file open, the file being created in a non-standard way, or misuse of the FMP calls. Note that FVERI can fix this problem by use of the '+FF' option.

3) Directory header/trailer flag incorrect at block <nnnnn>

The directory header and trailer (the two ends of a directory extent) should have a particular identifying flag. This directory does not have one.

(4) EOF pointer is beyond last block at block <nnnnn>

The last word of the file is reported to be beyond the last block in the file.

(4) Free space is marked as used space

The bit map has some disk space marked as used. However, the file system does not have any files or directories in that space. The space is wasted and unrecoverable through FMP. This message can be caused by having read-protected directories FVERI could not verify. FVERI works best when run by a system manager. Note that FVERI can fix this problem by use of the '+FB' option.

(5) Unidentifiable directory entry at block <nnnnn>

There is an entry in the directory that is not a file main, an extent, a purged file, or anything else defined for the file system. Note that FVERI can fix this problem by use of the 'FD+' option.

(5) Directory main size does not equal extent size at block <nnnnn>

Each directory extent should be the same size as the main. This is not true for this directory.

(6) Actual blocks in file entry is wrong at block <nnnnn>

The sum of the size of the main and all the extents is not the same as the total number indicated in the file's directory entry.

(6) Extent entry back pointer is wrong at block <nnnnn>

Extent entries have a pointer that points back to the previous extent entry or the main (extent entries are in a doubly linked list). The return pointer in this extent entry does not point back to the right place.

(6) Directory extent back pointer is wrong at block <nnnnn>

Directory extents have a pointer that points back to the previous extent or the main. The pointer does not point back to the right place.

(6) Total blocks in file less than main size at block <nnnnn>

The total number of blocks used by this file is less than the number of blocks in the main.

(6) Illegal file type at block <nnnnn>

This file's type is less than or equal to zero.

(7) Invalid directory extent pointer at block <nnnnn>

The pointer to the next or previous directory extent points to a disk address beyond the valid address space on this disk.

(7) Extent entry flag is wrong at block <nnnnn>

Extent entries in the directory should have a particular identifier flag. This extent does not have the right flag, yet is pointed to as an extent of this file.

(8) Invalid extent data pointer at block <nnnnn>

The pointer to the extent data points to a block address outside the legal address range of this disk.

(8) Invalid extent forward pointer at block <nnnnn>

The pointer to the next extent entry points to a block address outside the legal address range of this disk.

(8) Invalid extent pointer in main at block <nnnnn>

The pointer to the first extent of this file points to a disk address beyond the valid address space of this disk.

(8) Illegal directory size at block <nnnnn>

Each directory extent must be from 1 to 64 blocks long. This directory is not in that range.

(9) Duplicate use of disk block <nnnnn>

Two or more files are stored on the same section of the disk. At least one of the files must be corrupt. This message can occur as a result of other file activity on this disk while FVERI is running.

(9) Blocks per bit value is illegal at block <nnnnn>

The bit map represents up to 128 blocks of disk data per bit in the bit map. The actual number of blocks per bit must be a power of 2. There can be no more than 8192 words in the bit map. Finally, if blocks per bit is larger than 1, the number of words in the bit map should be greater than 4K (otherwise blocks per bit should have been half as large). If this value is wrong, the allocation of space on the disk can be corrupt.

(9) Used space is marked as free space at block

There is a space on the disk pointed to by a directory entry; however, it is not marked used in the bit map. This space is likely to be reclaimed at any time by the file system for another file. This message can occur as a result of other file activity on this disk while FVERI is running.
Note that FVERI can fix this problem by use of the '+FB' option.

7

File Transport Utility

FPORT, the file transport utility, is used in conjunction with the Applications Migration Package (AMP/9000) to transport files between HP 1000 and HP 9000 systems. Files can be transported on flexible disk, CS/80 cartridge tape, or 1600 bpi magnetic tape.

Export and Import Mode

FPORT operates in one of two modes, export or import. In export mode, FPORT writes specified files from disk to any of the transport media mentioned above. In import mode, FPORT writes files from a transport medium to disk. The files to be transported are specified in a user-prepared transport map file that is exported/imported with the files. Files can be in FMGR format or HP-UX format (FPORT does not support the CI hierarchical directory format).

File size, type, and record length information is exported with all FMGR files. HP-UX text files are exported as FMGR type 4 files, and HP-UX binary files are exported as FMGR type 1 files. If an imported file is to be created as a FMGR file, the exported file size, FMGR file type, and record length is used to create the file. If an imported file is to be created in the HP-UX format, the FMGR file information is ignored.

Transport Map

A transport map is required to specify all of the files to be transported and the way they are to be treated (that is, as FMGR or HP-UX files).

FPORT reads this map and then exports/imports the listed files as defined. The transport map is contained in a user-constructed file called the transport map file or transport file.

A transport file is a series of transport specification lines, each consisting of an export file name, an import file name (optional), and flags (optional), separated by blanks. Comments can be included on the specification line by preceding them with "#". (FPORT ignores everything following the # sign.)

The following flags can be used:

−F Treat the export name as an FMGR name.

−f Treat the import name as an FMGR name.

−b Treat the file as a binary file. This flag has meaning only for HP-UX files.

All file names are treated as HP-UX path names unless otherwise identified by a flag. If the import name is not specified, the file is imported with the export name.

FMGR files can be specified with security code and CRN, but the file type and size are ignored. File names cannot contain embedded blanks or the "#" comment character. (Lowercase letters in FMGR file names are shifted to uppercase before the file is referenced.)

As an example, the transport map shown in Figure 7-1 below might be used to move files from an HP 1000 to an HP 9000:

```
# sample transport map
#
#
&prog         prog.f        −F     #FMGR export name.
&sub1         lib/sub1.p    −F
&sub2::m1     lib/sub2.f    −F
data::50                    −fF    #FMGR export, import names;
                                   #default import name
```

Figure 7-1. Transport Map

Calling FPORT

To call FPORT enter one of the following runstrings:

```
:[RU,]FPORT,-E,tmap[,tdev]
```

or

```
:[RU,]FPORT,-I[FM],tmap[,tdev]
```

−E specifies export mode, and −I specifies import mode.

Tmap specifies the transport map file that defines the files to be exported/imported.

Tdev is the device LU of the external medium through which tmap and the transported files are copied. The default is LU 8.

F and M are the mutually exclusive flags, applicable in import mode only:

F	Force the use of tmap specified instead of the transport file from the transport medium.
M	Map only. Import only the transport file from the transport medium into tmap.

In export mode, files specified by tmap are copied to the transport medium specified by tdev. If all files specified in tmap can be opened successfully, the transport file is written to the transport medium and each export file is written to the medium in the order specified.

In import mode, if the FM flag is not specified, the transport map is read from the medium into tmap on the destination disk. Each file specified in the transport file is created if it does not exist or overwritten if it exists on the disk. If all files specified can be created successfully, each file is read from the medium in the order specified in the transport file. If a file cannot be successfully created or opened, FPORT exits.

If tdev is not specified in the command line, the default is LU 8 (normally the magnetic tape device). FPORT automatically rewinds the transport medium before and after executing.

If the medium is a flexible disk, the disk must be mounted and initialized before running FPORT to export files. The command sequence is illustrated in the following example (refer to the *RTE-6/VM Terminal User's Reference Manual* for details of the FMGR commands):

```
:mc,lu                          (Mount flexible disk)
:in,msc,-lu,crn,fport           (Initialize disk with ID FPORT)
:fport,-E,tmap,tfile::crn       (Export files specified in TMAP to TFILE)
:dc,lu                          (Dismount disk)
```

Loading FPORT

FPORT is loaded with the following load command sequence:

```
RE,FPORT
EN
```

8

PRINT Utility

The Print utility (PRINT) instructs the line printer to print files. PRINT has options that allow you to tailor the output to your individual programming and printing needs. PRINT runs in the background, instead of tying up your terminal until the printer has finished. It recognizes file masks and performs carriage control as specified in your file. Optionally, PRINT can indent the printed output, produce numbered listings, and create banners.

Using the PRINT Utility

PRINT puts a file identifying header and a banner headline on the first page of each printed file. The header contains the complete file name and the create, update, access, and print times of your printed file. The banner headline contains your name and file name. You can stop the generation of the header and banner headline at link time.

Calling PRINT

To call PRINT, enter the following runstring:

```
CI> [RU,] PRINT [filename] [LU] [<options>]
```

You may enter more than one file name in the runstring. When it runs, PRINT starts each file on a new page.

PRINT returns a listing of the print queue to your terminal screen if you do not enter any parameters in the runstring.

Normally, PRINT defaults to LU 6. You can set a different default LU at link time, or you can send the file to another LU by specifying an LU number in the runstring. Note that PRINT recognizes the LU specification as the printer regardless of where the LU number appears in the runstring.

You may enter a file mask in the runstring, and you may combine several features into a single runstring, as shown in the section on “PRINT Examples.”

PRINT Options

You can place PRINT options anywhere in the runstring provided a plus sign precedes each one. Options entered once apply to all files entered in a runstring. The +B and +W options can be entered more than once in a runstring. Table 8-1 summarizes the available options.

Table 8-1. PRINT Options Summary

Option		Description
+?		Returns a short explanation of available options
+A	output filename	Appends the output to a specified file instead of an LU
+B	string	Prints the specified string as a banner headline
+C	ON/OFF	Sets the carriage control option ON or OFF
+F	#	Advances paper the specified number of pages after all files are printed
+I	#	Indents each line by the specified number of spaces
+M	#	Merges successive files with the specified number of blank lines between them
+N		Produces a numbered listing for each file in the runstring
+O	output filename	Sends each file listed in the runstring to a specified destination
+P		Purges printed files sent to the line printer
+Q		Suppresses file mask verification
+S		Suppresses the message “Print job supervised by PRINX”
+W	working directory	Interprets file names from a specified working directory instead of current directory
+X	#	Prints all files contained in the runstring a specified number of times

+?

This option returns a short explanation of the PRINT options.

The format of the option is:

```
+?
```

+A

This option appends the output to a specified file instead of an LU. Compare +A to the +O option.

The format of the option is:

```
+A:outputfilename
```

+B

This option prints the specified string as a banner headline.

The format of the option is:

```
+B:string
```

Note that you may use +b more than once in a single runstring. In the following example, you may request up to six headlines per page by extending the option as shown:

```
PRINT +b:pascal +b:stuff file.pas file.lod
```

Here +b is used to combine specified banner headlines with default banners in the same runstring. This runstring prints the following two special headlines before the default banner and header:

```
PASCAL
STUFF
```

The two headlines in this example precede the first page of output for file.pas.

+C

This option sets the carriage control option ON or OFF. When you set it ON, the line printer does not print the first character of each line. Instead, it interprets any character in column one (of an otherwise blank line) as line or page formatting instructions.

When you set the carriage control option OFF, the line printer prints the entire line, including the first character.

If you do not specify this option, PRINT determines the setting of the option by searching for a carriage control (a “1” in column one of an otherwise blank line) in the first 70 lines of the file. If it is not found, PRINT assumes that there is no carriage control information in the file. The search terminates before the 70 lines have been scanned if PRINT finds anything other than a “*”, “+”, “0”, “2”, or “3”.

The format of the option is:

```
+C:ON
```

or

```
+C:OFF
```

+F

This option advances paper the specified number of pages after all files are printed. The form feed option overrides any defaults included in the %FFL module set by the system installer at load time. Specify +F:3, for example, to get three form feeds at the end of a printed file.

The format of the option is:

```
+F:#
```

+I

This option indents each line by the specified number of spaces. The default number of spaces is five.

The format of the option is:

```
+I:#
```

+M

This option merges successive files with the specified number of blank lines between them. The default number of blank lines is zero.

The format of the option is:

```
+M:#
```

+N

This option produces a numbered listing for each file in the runstring.

The format of the option is:

```
+N
```

+O

This option sends each file listed in the runstring to a specified destination. Use the +O option to transmit output to other files, including files on remote systems. This option overwrites destination files if they already exist. Compare this to the +A option.

The format of the option is:

```
+O:outputfilename
```

+P

This option purges the disk file that you have sent to the line printer.

The format of the option is:

```
+P
```

PRINT prompts you to confirm before it purges the files, unless you qualify the command with OK, as shown below:

```
+P:OK
```

In this case, prompting for confirmation is suppressed, and the files are purged.

+Q

This option suppresses file mask verification. If you do not use +Q, PRINT shows you all the files that match your runstring mask and asks you if it is OK to print them.

The format of the option is:

```
+Q
```

+S

This option suppresses the message "Print job supervised by PRINX." It overrides the default included in the %FFL module set by the system installer.

The format of the option is:

```
+S
```

+W

This option interprets file names from a specified working directory instead of your current directory.

The format of the option is:

```
+W:workingdir
```

+X

This option prints all files contained in the runstring a specified number of times.

The format of the option is:

```
+X:#
```

The PRINT Operation

When you call the PRINT utility, the following steps are executed:

1. PRINT expands any file masks, examines runstring arguments, and then stores any numeric argument as the output LU. PRINT defers evaluation of all options until the files are ready to print.
2. PRINT opens all the files to ensure that it can read them.
3. PRINT RPs PRIN0 as PRIN1 (or PRIN2 and so on if that program already exists). PRINT schedules this clone in the background, passing it all options, output, and input file names.

Note Since only the PRINT program should invoke the clone, Hewlett-Packard reserves the right to change the parameter sequence. Invoking PRIN0 or its clones directly may destroy the parameters passed to it.

---

4. The PRIN0 clone takes control of the printing process. The PRINT command terminates, and the CI or FMGR prompt returns to your screen.
5. PRIN0 clone retrieves the working directory and locks the specified output file or device. If the carriage control option is not specified, the PRIN0 clone checks the first 70 lines of each input file for carriage control characters. The character 1 in the first column of an otherwise blank line signifies a carriage control character and directs the line printer to discard the first column of each line in your file. By contrast, alphabetic letters in column 1 force PRINT to copy your file column-for-column to the line printer.

To abort a print job, use the CI command WH to determine the name of the PRIN0 clone doing the actual printing and then use the BR command. For example:

```
CI> BR Prin1
```

PRINT Messages

When you run it without any parameters, PRINT displays the printer queue and reports:

- the name of the file printing slave program
- the status of the file printing slave program
- the line printer LU for that print job
- the print job's priority (lower numbers get printed earlier)
- the terminal LU from which that print job was invoked

When you run it with parameters, PRINT displays a warning message if any input file is a type 1, 2, 5, 6, or 7 file or if FMP errors occur while reading a file.

In these cases, control passes from the first unprintable file in the runstring to the next printable file.

PRINT issues a warning and quits entirely in the following cases:

- if you enter the system BR (Break) command
- if you enter an unrecognizable option
- if you create an unwritable output file or if an FMP error occurs while accessing an output file
- if the printer LU goes down, or the LU number is outside the 0 to 255 range or does not exist
- if PRINT cannot find PRIN0, the master printing clone, or cannot execute PRIN0
- if nine or more PRINT requests are already active

PRINT Examples

Example 1: Print one copy of INPUT.ROFF, two copies of OUTPUT, and one of INDEX.

```
PRINT INPUT.ROFF OUTPUT OUTPUT INDEX
```

Example 2: Send the OUTPUT file to LU 12 instead of LU 6.

```
PRINT OUTPUT 12
```

Example 3: Print every file in the current directory that has a type extension of .FTN.

```
PRINT @.ftn
```

Example 4: Direct several files to LU 24.

```
PRINT 24 kalderesult src/kalde@
```

Example 5: Print MV.HELP from the current directory along with the following files:

```
/HELP/RN                    /HELP/TUTORIALS/CI
/HELP/MO                    /HELP/TUTORIALS/MAKE
/HELP/CO                    /HELP/TUTORIALS/MACRO
/HELP/MASK

PRINT mv.help +w:/help rn mo co mask +w:/help/tutorials ci make macro
```

Example 6: Print the following files:

```
SRC/PLIBF.FTN               SRC/PLIBM.MAC
HELP/PRINT                  HELP/FILES
/SCRATCH/OUTPUT

PRINT +w:src plibf.ftn plibm.mac +w:help print files /scratch/output
```

Note that SRC and HELP are subdirectories of the current directory; SCRATCH is a global directory. Working directories without a leading slash are interpreted as relative to the working directory.

Example 7: Print the file MV.HELP from the current working directory and the first occurrence of &PLIBF found on an FMGR cartridge.

```
PRINT mv.help +w:0 &plibf
```

Example 8: Include C.85 as a banner headline along with the default banner headlines.

```
PRINT +b:C.85 +b:libraries: fnewf.ftn @.mac
```

This example produces a leading page with the following banners:

```
C.85
LIBRARIES:
FNEWF.FTN
```

These are followed by the printed contents of FNEWF.FTN, a page with the following banner:

```
FOLDF.MAC
USERNAME
```

The banner is followed by the contents of the file FOLDF.MAC (assuming that is the only file matching @.MAC).

Example 9: Create the following pages:

- a page with the banner headlines FMP SOURCES and A.FTN followed by the contents of the file A.FTN
- a page with the banner headline B.FTN followed by the contents of the file B.FTN
- a page with the banner headline C.FTN followed by the contents of the file C.FTN
- a page with the banner headline ALIB.MAC followed by the contents of the file ALIB.MAC
- a page with the banner headline BLIB.MAC followed by the contents of the file BLIB.MAC
- a page with the banner headlines MAKEFILES and A.MAKE followed by the contents of the file A.MAKE
- a page with the banner headline MAKEFILE.MAKE followed by the contents of the file MAKEFILE.MAKE

```
PRINT +b:FMP +b:SOURCES @.FTN @.MAC +b:MAKEFILES @.MAKE
```

Note that the current directory contains A.FTN, B.FTN, C.FTN, ALIB.MAC, BLIB.MAC, A.MAKE, and MAKEFILE.MAKE.

Loading PRINT

Install PRINT by linking PRINT and PRIN0 and placing the resulting type 6 files on the directory /PROGRAMS. Use the link command files #PRINT and #PRIN0.

You can configure certain PRINT attributes at link time using the &FFL module. To change the attributes listed in Table 8-2, edit and compile &FFL, then link PRINT and PRIN0 using #PRINT and #PRIN0.

Note The &FFL module requires the HP 92836A FORTRAN 77 product in order to customize it for the PRINT utility. The FORTRAN 77 product is an optional product and is not available on the system unless purchased separately.

Table 8-2 shows the &FFL variables and their defaults. These variables exist in data statements in the &FFL module.

Table 8-2. &FFL Variables

Variable	Explanation	Default Value
defbanner	Should a banner be printed before each file?	yes
header	Should a header be printed before each file?	yes
numffs	How many form feeds belong at the end of the listing?	2
outputlu	Default printer LU	6
printmsgquiet	Should message “Print job supervised by PRINX” be suppressed?	no
The following variables control the width and spacing of banner headlines only:		
printerlength	How many spaces per line sent to the line printer?	80
filelength	How many spaces per line sent to the file?	80
printerbanspace	How many spaces between banner characters?	3
filebanspace	How many spaces between banner characters?	3

A listing of the &FFL module is given below:

```
ftn7x,l,q,s,c
*
*     NAME:   FFL
*     SOURCE: 92077-18067
*     RELOC:  92077-16067
*     PGMR:   sb
*
*  ****************************************************************
*  * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1984.  ALL RIGHTS      *
*  * RESERVED.  NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,       *
*  * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT *
*  * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.        *
*  ****************************************************************
*
      block data ffl
     +,92077-16067 REV.5000 <920807.1030>

*
*     Do not alter these declarations !!
*

      common /defoptions/ defbanner,header
      logical*2           defbanner,header

      common /ff/ numffs
      integer*2   numffs

      common /outputlu/ outputlu
      integer*2         outputlu

      common /lengths/ printerlength,filelength,
     +                 printerbanspace,filebanspace
      integer*2        printerlength,filelength,
     +                 printerbanspace,filebanspace

      common /optprintmsg/ printmsgquiet
      logical*2            printmsgquiet
*
*  Tailor the values within the slashes below according to local  tastes
*


      data defbanner        /.true./  ! print banner before each file?
      data header           /.true./  ! print header before each file?


      data numffs           /2/       ! number of form feeds after last file
```

```
      data outputlu          /6/      ! default printer lu

      data printmsgquiet    /.false./ ! suppress print verification message
*                                           "Print job supervised by PRINx"
*
*     Following variables are used to determine number of banner characters
*     allowable on one line
*

      data printerlength    /80/      ! printer line length
      data filelength       /80/      ! file line length
      data printerbanspace  /3/       ! spaces between banner chars - printer
      data filebanspace     /3/       ! spaces between banner chars - file

      end
```

9

Disk Formatting Utilities

The online disk-formatting utilities, FORMT and FORMC, allow you to format disks and to verify the disk integrity. FORMT formats MAC/ICD disks and flexible disks; FORMC performs the same functions for the CS/80 sealed disks and fixed-block tape cartridges.

This chapter describes these utilities, which are shown in Table 9-1 below. It is a good idea to check Table 9-1 before you format any disk to make sure you are selecting the proper utility for your task.

Table 9-1. RTE-6/VM Formatting Utilities

FORMC Supports	FORMT Supports
7907 7908 7911 7912 7914 7933 7935 7942 7945 7946 9122C/D 9133H 9144 CS/80	7906 H/M 7910 H/M 7920 H/M 7925 H/M 9130 K (Model 6 built-in disks) 9895
Runs under these operating systems:	
RTE-6/VM RTE-A	RTE-6/VM

FORMT

FORMT allows you to format a flexible disk, initialize and reformat a hard disk, and spare hard disk tracks online. A verify function allows you to confirm the data integrity of both the flexible disks and the hard disks. FORMT can be used with all ICD (Integrated Controller Disk) and MAC (Multiple Access Controller) disks except the HP 7900. Only the HP 9895 double-sided, dual-density flexible disk can be formatted using FORMT.

All hard disks are pre-formatted (track and sector addresses and timing surface are written to the disk) before shipment from HP. However, you must format a flexible disk before initial use. The FORMT FO function is provided for this operation and can only be used with the HP 9895 flexible disks.

The pre-formatted hard disks can be initialized, spared, or reformatted with FORMT. Initialization (IN function) is the process of removing the protected status of all tracks in the disk, cleaning up the spare track pool, and sparing any bad tracks found during the initialization process. Bad tracks reported during subsequent use of the disk can be spared individually using the SP function. Reformatting, in which the status bits are cleared and zeroes are written into the data blocks (thus wiping the disk clean), is performed using the RE function. A detailed description of the initializing, formatting, and sparing processes is contained in Appendix A.

Loading FORMT

Note FORMT locks itself into a memory partition. Therefore, the system should be generated to allow background partitions with memory lock.

Load FORMT from the relocatable file %FORMT (part number 92067-16554) during generation or online using LOADR. FORMT also uses $DSCLB, the Disk Utility Library (part number 92067-12002), during the load procedure. In specifying the load, you must override the program page requirements to allow a track buffer for the disk transfer. FORMT itself requires 10 pages; the override program size for each disk type is as follows:

9895 – 11 pages
7905(H) – 16 pages
7906(H) – 16 pages
7920(H) – 16 pages
7925(H) – 18 pages

Load FORMT online with the following command sequence:

```
:RU,LOADR
 /LOADR: SZ,18 ! override program size (7925[H] disk)
 /LOADR: RE,%FORMT ! load FORMT utility
 /LOADR: SE,$DSCLB ! search Disk Utility Library
 /LOADR: END
:
```

FORMT can be loaded as a type 1, 2, 3, or 4 program with no SSGA required. After FORMT is loaded, it should be saved as a type 6 file (executable memory-image code) using the FMGR SP command and restricting access to the System Manager only.

Calling FORMT

Caution No other program should access the disk LU being used by FORMT. The INitialize, SPare, VErify and REformat tasks lock all disk drives with the same EQT as the MAC and ICD disk LU being accessed, thus system performance is downgraded while these tasks are executing.

To call FORMT enter the following runstring:

```
:[RU,]FORMT[,input[,cmd,disk_LU[,n]]]
```

Here input is the LU of the device from which the FORMT parameters are to be read. The default is the log device.

The cmd parameter specifies one of the FORMT function commands, which are described in the following section.

The disk LU must be specified as a positive integer between 0 and 256. Note that the system disk subchannel (LU 2) and the auxiliary disk subchannel (LU 3) cannot be initialized. Be aware, however, that these LUs can be reformatted, which would destroy the system disk. Exercise caution in specifying the disk LU when the operation is to REformat a disk.

If you invoke FORMT using only the utility call or if all parameters are not supplied (only the input LU can be defaulted), FORMT goes into interactive mode, prompts with

```
TASK?
```

and waits for you to enter the desired function. When you enter a configuration command, the program prompts you to enter the LU number of the disk to be formatted, initialized, spared, or verified.

```
DISK LU?
```

FORMT Commands

Table 9-2 summarizes the available FORMT commands that are further described in subsequent sections.

Table 9-2. FORMT Commands Summary

Commands		Description
Information Commands		
HE **??**		Displays a list of FORMT commands in response to TASK?
Configuration Commands		
FOrmat	disk LU,n	Formats a flexible disk using n as the fill sector value
INitialize	disk LU	Initializes the named disk LU
REformat	disk LU	Reformats the named disk LU
SPare	disk LU, n	Spares track n on the named disk LU
VErify	disk LU	Verifies the data on the named disk LU
Exit Commands		
ENd **/E** **EX**		Terminates FORMT when entered in response to TASK?; otherwise, returns system to TASK? prompt.

Help (??)

Purpose: Prints a list of all valid responses to the current prompt.

Syntax: `HE or ??`

End (EN)

Purpose: Terminates FORMT when entered in response to the TASK? prompt. Otherwise, entering EN, /E, or EX returns the system to the TASK? prompt.

Syntax: `EN, /E, or EX`

Format a Flexible Disk (FO)

Purpose: Formats a floppy disk.

Syntax: `[RU,]FORMT,[input], FO[,disk_lu[,n]]`

or

`[RU,]FORMT,,FO`

Description:

Calling FO interactively defaults the input device to your terminal.

The flexible disk physical sectors are 128 words each (64-word logical sectors) plus the sector/block preamble and postamble. The postamble holds information needed for data error checking. Formatting the HP 9895 flexible disk includes writing the track and sector addresses and the interleaving fill number in the preamble and identifying defective tracks.

The interleaving fill number (if supplied in the disk LU parameter) defines the logical order in which consecutive sectors are to be read from the disk. For example, if a fill number = 1 is supplied and sector 1 is read first, the next consecutive physical sector is passed over and the next sector (physical sector 3) is read. The sector is identified in the preamble as sector #2. If the fill number is omitted, the default is to read sectors consecutively (fill number = 0).

If a defective block is found during the formatting process, the status is entered in the preamble. The entire track containing the bad block is skipped and becomes invisible to you during use of the flexible disk. The first good track formatted following the skipped track is assigned the next consecutive track number. For example, if tracks 0, 1, and 2 are good tracks, but the next two tracks (3 and 4) are found to be defective and are skipped, the good track (physical track 5) is identified as track #3 in the sector preamble of that track.

Note In the formatting process, all previous information on the disk LU is lost; binary zeroes are written on every track, including the directory track.

The FORMT Formatting Operation

When formatting occurs, bad tracks are identified and made invisible to your software. Bad tracks are not spared; however, unless the number is excessive, they are no longer of concern.

Note that all information previously written to the disk is lost when the disk is formatted.

Entering the LU

After you enter the FO command and specify the LU, the program verifies that the LU is a flexible disk by checking the System Track Map Table. Under session monitor, FORMT also verifies that the LU is in your SST. If there is a discrepancy, the program notifies you:

```
INVALID DISK LU
ENTER DISK LU <256
```

You are then prompted again to enter the disk LU. If the correct LU was entered, FORMT formats the flexible disk.

Confirming Formatting

Since formatting destroys the contents of the specified disk LU, if you are in interactive mode, FORMT asks you to confirm that you want formatting to proceed:

```
DO YOU REALLY WANT TO FORMAT THIS DISK?
```

Respond by typing YES or NO (the first two characters are sufficient).

Locking Other LUs

Once the disk LU has been determined to be valid and the YE response has been given, FORMT scans the DRT (Device Reference Table) and finds all other disk LUs that point to the flexible disk drives on the controller. All of these LUs are then locked for the duration of the FORMT process. Both the left side and right side drive LUs are locked, therefore it is not possible for other programs to access either side even though only one flexible disk LU is to be formatted.

Sector Interleaving

You are then asked to specify the number of “fill” sectors (the interleave factor):

```
# OF FILL SECTORS?
```

Enter the number of fill sectors desired, between 0 and 28. If you enter zero, sectors are addressed in the same order as they physically appear on the disk. A number other than zero causes consecutive sector addresses to be separated by the number of fill sectors specified. This is known as sector interleaving. For more information on interleaving, see Appendix A.

Note For best performance, a fill number of 1 is recommended for the system disk or any disk accessed by Hewlett-Packard supplied utilities, including EDIT, the loader, and so on.

The Formatting Process

FORMT makes five passes over the disk to detect any defective tracks. Since each pass may take several minutes, the program displays the following message at the beginning of each pass:

```
FORMAT PASS # nn
```

During each pass, FORMT issues commands to the controller hardware that may take several minutes to execute. During execution of these commands, the controller physically locks the door on the drive so that the disk cannot be removed. If you halt FORMT while these commands are being executed, the drive access door remains locked until all hardware commands are executed.

After formatting the disk, FORMT reports the number of good tracks on the medium as

```
# OF GOOD TRACKS = nnn
```

In interactive mode, FORMT issues the TASK? prompt. You may enter another task or enter EN, /E, or EX to exit. If FORMT was called through a runstring, the utility exits after the successful FO operation.

FORMT FO Example

In the following example, the FMGR SL command is first issued to map flexible disk LU 45 to the system LU currently in the user SST. The FMGR DC command then is used to dismount the LU and return the resource to the system (RR parameter). FORMT is then invoked interactively. Following the successful format operation, the flexible disk is mounted to the user session with the FMGR MC command.

```
:SL,45,45               ! map disk LU into SST.
:DC,45,RR               ! dismount disk and make sure no
                        ! one is using LU (RR parameter).

 DISK CRN 45 LU 45 DISMOUNTED FROM
      SYSTEM

:RU,FORMT

 TASK?
 FO

 DISK LU?
 45

 DO YOU REALLY WANT TO FORMAT THIS DISK?
 YES

 # OF FILL SECTORS?
 1


 READY DISK - ENTER " ",CR
 <cr>
 FORMAT PASS # 01
 FORMAT PASS # 02
 FORMAT PASS # 03
 FORMAT PASS # 04
 FORMAT PASS # 05

 # OF GOOD TRACKS = 130

 TASK?
 EN

 FORMT FINISHED
:MC,45,P,,LU 45,1,45                ! mount cartridge to get
```

Initialize an LU (IN)

Purpose: Initializes a disk.

Syntax: `[RU,]FORMT,[input],IN[,disk_lu]`

or

`[RU,]FORMT,,IN`

Description:

Calling IN interactively defaults the input device to the log terminal.

In initializing a pre-formatted ICD or MAC hard disk, the existing contents of the disk (including the directory tracks) are overwritten with a pattern of all zeroes.

The spare track pool is initialized first and the tracks, including the sector preambles and postambles, are set to zero. On verification, bad tracks are flagged by setting the defective track bit (D) in the preamble of the track's sector 0.

The mapped tracks are then initialized. If the track was previously flagged as bad, it is spared and the defective track/spare track addresses are cross-referenced in the sector 0 preamble of both tracks. As part of the initialization process, each track is then verified block by block and bad tracks encountered in the verification also are spared.

Caution The IN command locks all disk drives sharing the EQT of the specified disk LU. The lock is issued on a track-by-track basis for each track on the disk. All loads, swaps, and other accesses to any of the disks sharing the same EQT are severely limited during execution of the IN command. If the system disk shares the EQT, the IN command should be used only when severely degraded system performance can be tolerated.

The FORMT Initializing Operation

Remember that you can only initialize a disk cartridge that was previously formatted at the HP factory.

Entering the LU

After you enter the FO command and specify the LU, the program verifies the disk LU and type. Any attempt to initialize LU2 (system disk), LU3 (auxiliary disk), or a flexible disk causes the following message to display:

```
INVALID DISK LU
ENTER DISK LU < 256
```

You are then prompted again to enter the disk LU. When the correct LU is entered, FORMT initializes the disk.

Confirming Initializing

Since initializing overwrites the previous contents of the disk with a pattern of all zeros, the program asks (if you are in interactive mode) you to confirm that you want initializing to proceed:

```
DATA WILL BE DESTROYED, OK TO PROCEED?
```

When you respond by entering YES, the program performs all the necessary operations to prepare the LU and reports the following information:

```
BAD TRACKS SUBCHANNEL XX

LU XX       LOGICAL     CYL         HEAD        UNIT/ADDR
BAD TRACK   XXXX        XXXX        X           X/X
SPARED TO   XXXX        XXXX        X           X/X

XX SPARE TRACKS AVAILABLE
```

In interactive mode, FORMT then issues the TASK? prompt. You may enter another task or enter EN, /E, or EX to exit. If FORMT was called through a runstring, the utility exits after the successful IN operation.

The Initializing Process

When initializing occurs, the program cleans up the tracks in the spare pool. The full block (including the preamble and postamble) is read and status bits are examined. If the D bit is set (indicating a defective track), the program proceeds to the next track. A track from the spare pool is not spared. If the D bit is not set, the program rewrites the full track with zeros, including the S, P, and D bits. Verification is performed by reading back the track. If the track is found to be bad during verification, it is flagged as defective.

The program then reads each track in the data portion of the LU. If the D bit is set or the read is unsuccessful, a flag indicating that sparing is needed will be set.

If sparing is needed, the preamble is prepared for a defective track by setting the D bit and the address of the spare track in the preamble of the defective track. The preamble for a spare track is prepared by setting the S bit and the address of the defective track in the preamble of the spare.

The program write initializes the first track, using the preamble set above (if sparing). If this is a good track, the address is the address of the good track (it points to itself). If this is a spare track, the address is that of the defective track (a backward pointer). If this is a defective track, the address is that of the spare.

The data buffer is all zeros if processing the IN command. The same procedure is used to spare an individual track when the SP command is used, except that the data buffer is copied (whenever possible) from the defective track on a block-by-block basis. In recovering user data, an offset read is performed to pick up data that cannot normally be read with the head aligned to the center of the track.

The program verifies the whole track. If verification is successful, the program continues with the next track. If verification is unsuccessful (bad verify status), the program gets the next spare and does the sparing procedure above.

FORMT IN Examples

In the following example, the FMGR command SL is first used to map disk LU 54 to the system LU currently within the user SST. The FMGR command DC is then used to dismount the LU and return the resource to the system (RR parameter). Following this, the FORMT utility is invoked interactively. After the successful initialization operation (track sparing is not required in this example), FORMT displays the number of remaining spare tracks and returns to the TASK? prompt. When FORMT exits, the MC command is then used to mount the initialized disk as a group cartridge.

```
:SL,54,54                          !map disk LU into SST
:DC,54,RR                          !dismount disk and make sure
                                   !no one else is using LU 54
                                   !(RR parameter)


 DISK CRN 54 LU 54 DISMOUNTED FROM SYSTEM

:RU,FORMT

 TASK?
 IN

  DISK LU?
  54

  DATA WILL BE DESTROYED, OK TO PROCEED?
  YE

  0004 SPARE TRACKS AVAILABLE

  TASK?
  /E

  FORMT FINISHED
:MC,-54,G,203,CRN54,1,54           ! mount cartridge to get
                                   ! directory entry.
```

In the following example, the FMGR SL, DC, and MC commands are used as above; the utility is called using a runstring, so no operator intervention is required. The results of the track sparing are displayed, the number of remaining spare tracks is listed, and FORMT exits.

```
:SL,14,14                       ! map disk LU into SST
:DC,14,RR                       ! dismount LU and make sure LU is
                                ! not in use (RR parameter)

 DISK CRN 14 LU 14 DISMOUNTED FROM SYSTEM

:RU,FORMT,,IN,14

 BAD TRACKS SUBCHANNEL 05       ! bad tracks found on LU 14,
                                ! Subchannel 5.
                                ! results of track sparing are
                                ! displayed following initialization

 LU 14 LOGICAL CYL HEAD UNIT/ADDR

 BAD TRACK 0002 0210 02 00
 SPARED TO 0198 0406 02 00
 BAD TRACK 0089 0297 02 00
 SPARED TO 0199 0407 02 00

 0003 SPARE TRACKS AVAILABLE

 FORMT FINISHED
:MC,14,P,,LU14,1,14             ! mount cartridge to get
                                ! directory entry.
```

Spare a Track (SP)

Purpose: Substitutes a spare track for a defective track and transfers as much data as possible from the bad track to the spare.

Syntax: `[RU,]FORMT,[input], SP[,disk_lu,n]`

or

`[RU,]FORMT,, SP`

Description:

Calling SP interactively defaults the input device to the log device.

When a bad track is found during a disk read, an error is returned to the calling program, identifying the defective track. The FORMT SPare function is used to spare these bad tracks individually. In track sparing, a known good track is substituted for the defective track and as much data as possible is recovered to the spare track. The defective track address and the spare track address are then cross-referenced in the preamble of both tracks' sector 0.

Caution Be aware that once a track is spared, it is permanently marked as defective. The process is irreversible.

All disks sharing the EQT of the specified disk LU are locked during the SP operation. All loads, swaps, and other accesses to any of the disks on the EQT are not allowed while a track is being spared.

The FORMT Sparing Operation

When you invoke the SP command, the program substitutes a spare track for a defective track and transfers as much data as possible from the bad track to the spare.

Entering the Track Number

After the command and the disk LU are entered, either interactively or by runstring, FORMT verifies the disk LU and type. If you attempt to spare a flexible disk or if the disk LU is not in your SST (under Session Monitor), FORMT issues the message

```
INVALID DISK LU
ENTER DISK LU < 256
```

and waits for you to enter a valid disk LU. When the correct LU is entered, FORMT spares the track specified in the runstring. In interactive mode FORMT prompts with

```
TRACK TO BE SPARED?
```

If you enter ?? to this prompt, FORMT returns the track range of the disk. Enter the logical track number to be spared. Note that if you enter the logical number of a good track, it is spared and permanently labeled as a defective track. If the specified track is not within the bounds of the disk LU specified, FORMT prompts with

```
INVALID TRACK #
ENTER BAD TRACK # x - xxxx
```

and waits for you to enter a logical track number within the specified range.

The Sparing Process

The program copies data, block by block, from the bad track to the spared track. If multiple tries must be made (at each block) with various head offsets, this operation may take several minutes. If all the information on the track cannot be recovered, FORMT issues the following warning message:

```
WARNING!  ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED
```

and continues sparing the defective track. When the sparing is complete, FORMT reports the successful operation as follows:

```
BAD TRACKS SUBCHANNEL xx

LU XX        LOGICAL     CYL         HEAD        UNIT/ADDR
BAD TRACK    XXXX        XXXX        X           X/X
SPARED TO    XXXX        XXXX        X           X/X

XX SPARE TRACKS AVAILABLE
```

In the interactive mode, FORMT then issues the TASK? prompt. You may enter another task or enter EN, /E, or EX to exit. If FORMT was called through a runstring, the utility exits.

FORMT SP Examples

In the following example, the FMGR SL is used to map disk LU to the system LU currently in the user SST. The FMGR command MC is then used to mount LU 14. (If the disk is already mapped and mounted, these commands are not necessary and will cause a FMGR error.) The utility is then called interactively. After the sparing operation, FORMT reports the results and returns to the TASK? prompt.

```
:SL,14,14                          ! map LU 14 into SST.
:MC,14,P                           ! mount LU 14 to your session.
                                   ! as a private cartridge.
:RU,FORMT

 TASK?
 SP

 DISK LU?
 14

 TRACK TO BE SPARED?
 3                                 ! logical bad track on LU 14.

 BAD TRACKS SUBCHANNEL 05          ! LU 14 is on subchannel 05.

 LU 14 LOGICAL CYL HEAD UNIT/ADDR

 BAD TRACK 0003 0211 02 00
 SPARED TO 0202 0410 02 00

 0000 SPARE TRACKS AVAILABLE       ! no more spare tracks left.

 TASK?
 /E                                 ! exit FORMT.

FORMT FINISHED
```

In the following example, the FMGR SL and MC commands are used as above. FORMT is called with a runstring, so no operator intervention is required and FORMT exits after sparing the bad track.

```
:SL,23,23                          ! map LU 23 into SST.
:MC,23,P                           ! mount LU 23 as a private
                                   ! cartridge.

:RU,FORMT,,23,SP,1                 ! spare logical track #1 on
                                   ! LU 23.
 WARNING! ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED

 BAD TRACKS SUBCHANNEL 03

 LU 23 LOGICAL CYL HEAD UNIT/ADDR

 BAD TRACK 0001 0410 01 00
 SPARED TO 0204 0410 00 00

 0002 SPARE TRACKS AVAILABLE

 FORMT FINISHED
```

Reformat a Disk (RE)

Purpose: Reformats the named disk LU.

Syntax: `[RU,]FORMT,[input],RE[,disk_lu]`

or

`[RU,]FORMT,, RE`

Description:

The FORMT REformat function clears the entire disk area designated by the LU, except for the servo and timing data written to the inter-sector gaps when the disk was pre-formatted prior to shipment from HP. The directory track is overwritten with zeroes, zeroes are written to all sector preambles, data areas, and postambles, and the spare track pool is cleared. The reformatted disk is not verified, and bad tracks are not spared during the RE operation.

If the PRSTR VErify option is not specified when restoring the disk or if the disk is restored using READT, the FORMT IN function should be used before restoration to initialize the disk and to spare tracks according to the system track map. HP-designated bad tracks must be spared using the FORMT SP function even if the tracks are not found to be bad during initialization. Bad tracks defined during disk pre-formatting are identified in the HP documentation supplied with the disk. (If the records are not available when the disk is to be reformatted, contact your HP customer engineer for assistance.)

Caution If the system disk (LU 2) and auxiliary disk (LU 3) are reformatted, the operating system is lost and the system will go down. This capability should be used only with the full and complete understanding of how to back up the disk using PSAVE and how to restore it using the offline physical backup utility RE.

Any disk can be formatted, including the system disk and auxiliary disk. However, you must have a user capability of at least 60 (usually reserved for the System Manager or other support personnel) before you can reformat the system disks, and the reformatting can only be done interactively. (Refer to the *RTE-6/VM Terminal User's Reference Manual* for a discussion of user capabilities.)

The FORMT Reformatting Operation

Remember that you can only reformat a disk that was previously reformatted at the HP factory, since the servo and timing data are not reformatted.

In reformatting disks other than LU 2 or LU 3, the following steps are recommended:

1. Use PSAVE or WRITT to back up all data from the disk to be reformatted.
2. Make sure no one is using the disk, then dismount the disk using the FMGR DC command.
3. Reformat the disk using RE.
4. If the data is not to be restored, or if the data is to be restored using READT, use the IN function to spare bad tracks.
5. Use the SP function to individually spare those tracks identified as bad during preformatting prior to shipment from HP.
6. Mount the disk using the FMGR MC command.
7. Restore the data, if desired, using PRSTR or READT. If PRSTR is used with the VErify option, the IN function is not needed: the PRSTR VE option spares the tracks in a prepass before the data is written.

Caution Do not use WRITT to back up LU 2 and LU 3. This utility only saves the FMGR area of the disk, not the system area.

In reformatting LU 2 and LU 3, the following steps are recommended:

1. Log on to an account with a user capability of at least 60. FORMT will issue the message "UNAUTHORIZED LU 2, 3 ACCESS" if your capability is less than 60.
2. Use PSAVE to back up all data from LU 2 and LU 3. Note that you should back up both LU 2 and LU 3 even though reformatting is required for only one of the system disks. After reformatting, the system will attempt to come back and the results will be unpredictable.
3. Halt *all* system activity. Failure to halt all activity when reformatting the system disks could cause FORMT to be swapped to disk between reformats, and the utility would be lost.
4. Reformat LU 2 and LU 3. Note that RE functions only in interactive mode when reformatting the system disks.
5. Use the SP function to individually spare those tracks identified as bad during preformatting prior to shipment from HP.
6. Load the offline physical backup utilities as described in Chapter 6.
7. Restore LU 2 and LU 3 using the offline RE command *with* the VErify option to spare bad tracks in a prepass over the disk prior to the restoration.
8. Re-boot the system.

Caution The RE command locks all disk drives sharing the EQT of the disk to be reformatted. The lock is issued on a track-by-track basis for each track on the specified disk LU. All loads, swaps, and other accesses to any of the disks sharing the same EQT are severely limited during execution of the RE. If the system disk shares the same EQT as the disk being reformatted, the RE function should only be used when severely degraded system performance can be tolerated.

The interactive form of the command defaults to the log device for parameter inputs. If LU 2 and LU 3 are to be reformatted, RE can only be called interactively. After the command is entered, FORMT verifies the disk LU and type. If you attempt to reformat a flexible disk or if the disk is not in your SST (under Session Monitor), FORMT issues the message

```
INVALID DISK LU
ENTER DISK LU < 256
```

and waits for you to enter a valid disk LU. If you attempt to reformat LU 2 and LU 3 without a command capability of at least 60 or without being in interactive mode, FORMT issues the message

```
UNAUTHORIZED LU2,3 ACCESS - (COMMAND IGNORED)
```

and exits or, if in interactive mode, returns to the TASK? prompt. If you are reformatting LU 2 and LU 3, FORMT issues the prompt

```
DO YOU REALLY WANT TO REFORMAT THE SYSTEM DISK?
```

and waits for you to enter a YES or NO response. In interactive mode, FORMT then issues the message

```
DATA WILL BE DESTROYED.  OK TO PROCEED?
```

and waits for you to enter a YES or NO response. If your response is YES, FORMT proceeds to reformat the disk.

In interactive mode, FORMT issues the TASK? prompt following the successful reformat operation. You may enter another task or enter EN, /E, or EX to exit. If FORMT was called through a runstring, the utility exits after the successful RE operation.

Verify a Disk (VE)

Purpose: Verifies the specified LU and reports defective tracks.

Syntax: `[RU,] FORMT,[input],VE[,disk_lu]`

or

`[RU,] FORMT,,VE`

Description:

The FORMT verification process, the VE function, reads and verifies all data on either a flexible disk or a hard disk. While the directory track is checked, unused tracks in the spare-track pool are not verified. If a track has been previously spared, VE will verify that spared track. The results of the verification are reported, but track status is not changed. That is, tracks found to be bad are not spared; the SP function must be used to spare bad tracks individually.

Caution VE locks all disk drives sharing the EQT of the specified disk LU. The lock is issued on a track-by-track basis and all loads, swaps and other accesses to any of the disks sharing the same EQT are severely limited during execution of the VE function. If the system disk shares the same EQT as the disk to be verified, the VE command should only be used when severely degraded system performance can be tolerated.

In verifying a flexible disk, only the LUs associated with a specific disk LU are locked and remain locked until the VE function is completed.

The FORMT Verify Operation

The interactive form of the VE command defaults the input device to your terminal.

FORMT proceeds by reading each track in the mapped portion of the LU. If the read is unsuccessful, the track is reported as bad.

The directory track of the disk LU is checked; the spare track pool is not checked.

Entering the LU

After the command is entered, either interactively or by runstring, FORMT checks the disk LU and type. If you attempt to verify a non-disk LU or a disk not in your SST (under Session Monitor), FORMT issues the message

```
INVALID DISK LU
ENTER DISK LU < 256
```

and waits for you to enter a valid disk LU.

The Verify Process

When the correct LU is entered, FORMT verifies the disk and reports any bad tracks without a valid spare as

```
BAD TRACKS SUBCHANNEL XX

LU XX       LOGICAL     CYL         HEAD        UNIT/ADDR
BAD TRACK   XXXXX       XXXX        XX          X/X
```

In interactive mode, FORMT then issues the TASK? prompt. You may enter another task or enter EN, /E, or EX to exit. If FORMT was called through a runstring, the utility exits after the VE operation.

FORMT Error Messages

The following error messages can be generated by FORMT:

BAD TRACKS SUBCHANNEL XX

LU XX	LOGICAL	CYL	HEAD	UNIT/ADDR
BAD TRACK	XXXX	XXXX	XX	XX
SPARED TO	XXXX	XXXX	XX	XX

XXXX SPARE TRACKS AVAILABLE

A defective track encountered during an IN or SP task is spared.

CYLINDER COMPARE ERROR

LU XX	LOGICAL	CYL	HEAD	UNIT/ADDR
TARGET TRACK	XXXX	XXXX	XX	XX

The disk controller was unable to seek to the target track. Be sure that the disk is formatted. The current task is aborted and FORMT returns to TASK? mode.

INVALID DISK LU

The disk LU is the wrong type, has zero tracks defined in it, or the response was not a numeric parameter. LU 2 and LU 3 cannot be initialized.

INVALID DISK SPECIFICATIONS xx

The disk controller detects an out-of-bounds condition on either a cylinder, head, sector, or unit, based on the invalid subchannel definition specified in the message. This is due to an incorrect track map table or system generation answer file. The current task is aborted and FORMT returns to TASK? mode.

LU XX	LOGICAL	CYL	HEAD	UNIT/ADDR
BAD SPARE	XXXX	XXXX	XX	XX

A defective spare is encountered in either the IN or SP task. This message appears in conjunction with the BAD TRACKS heading.

MAX OF 20 BAD TRACKS EXCEEDED

Flexible disk only. FORMT returns to TASK mode.

NOT ENOUGH ROOM FOR TRACK BUFFER

The memory bounds specified for FORMT at load time did not provide sufficient free memory to serve as a track buffer for the task and disk LU requested. FORMT terminates. Reload FORMT and resize it to the recommended program size.

OUT OF SPARE TRACKS FOR THIS LU

All spare tracks have been used. The sparing operation cannot be performed. FORMT returns to TASK? mode.

READY DISK-ENTER ” ”CR

The disk drive is not ready or there is no disk in the drive. Enter a space and carriage return when the disk is ready or enter /E to abort the task.

TURN OFF PROTECT OR READ-ONLY SWITCH-ENTER ” ”CR

The disk protect (or read-only) switch is on. This error also occurs if the floppy disk in the selected drive has the write-protect notch present. Turn off the switch and enter a space and carriage return to continue or enter /E to abort the task.

TURN ON FORMAT SWITCH-ENTER ” ”CR

The utility is initializing the disk, but the format switch is off. This error also occurs if a write operation is attempted to a track with its P-bit set while the format switch is off. Turn on the format switch and enter a space and carriage return to continue or enter /E to abort the task.

WARNING! ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED

Spare command only. The track about to be spared was not successfully preserved. FORMT continues.

**WARNING! POSSIBLE BAD FLOPPY MEDIA –
RETRY FORMT OR DISCARD FLOPPY**

Flexible disk only. An error occurred on the fifth and final FORMT/Verify pass. Re-starting the FO process may mask the defective track.

FORMC

FORMC, the online maintenance utility for CS/80 disks and tapes, allows you to verify the integrity of data on the device using the VErify option. Errors detected can be corrected using the FORMC SPare and FOrmat options to spare bad disk tracks or to format the entire Cartridge Tape Drive (CTD) tape or disk volume.

If only an offline environment is available or if physical addressing of the disk is required, the CS/80 disk exerciser should be used for disk maintenance. Refer to the *CS/80 External Exerciser Manual* (part number 5955-3462), or Appendix B of this manual, for details on the CS/80 disk exerciser utility (EXER).

Calling FORMC

The utility can be run with or without user interaction. In interactive mode, FORMC prompts for each command and its associated parameters. For non-interactive operation, all parameters required for the specified command must be included in the runstring. FORMC terminates after successful completion of the command or, in the case of an error condition, after issuing the appropriate error message. FORMC also can be called programmatically by issuing the appropriate program-scheduling EXEC call and passing the parameters through the runstring.

To call FORMC, enter the following runstring:

```
:[RU,]FORMC[:IH],[list LU],option,LU[,option params]
```

The :IH parameter inhibits automatic renaming of the program. :IH must be specified when FOrmat or SPare is selected since these options require that FORMC be run without renaming.

The list LU is the LU (Logical Unit number) of the list device to which FORMC will direct messages generated during the specified operation.

Option specifies one of the FORMC commands, described in the next section.

LU is an LU of the CS/80 disk or tape to be accessed by FORMC for the specified operation.

The option params are the parameters applicable to the specified option. Refer to the associated option for a description of the parameters.

When FORMC is called interactively, the following sequence of messages is issued:

```
/FORMC: CS/80 DISK FORMAT UTILITY
/FORMC: '?' WILL LIST THE LEGAL COMMANDS
/FORMC: TASK?
```

and FORMC waits for you to enter the desired function.

Command Execution

Before it executes a command, FORMC issues the message:

```
/FORMC:  COMMAND EXECUTING - <command> <parameters>
```

FORMC terminates after successful command execution or, if an error occurs, after it issues the appropriate error message.

Device Driver Status

During command execution, FORMC tests the driver status and if −1 is present in QSTAT (indicating driver timeout), it issues the following messages and exits:

```
/FORMC: DEVICE DRIVER TIMED OUT. CONTACT SYSTEM MANAGER.
/FORMC: ABORTED
```

Driver timeout can result if the disk drive is disconnected or if a hardware failure occurs. (Refer to the "FORMC Error Messages" section later in this chapter for a description of the CS/80 device driver status information message.)

Break Detection

FORMC checks the break flag only during verification after each locate and read. If the flag is set, FORMC exits with the following messages:

```
/FORMC: BREAK DETECTED
/FORMC: ABORTED
```

The break flag is not checked during formatting and sparing operations. To protect data integrity, these operations should complete without intervention after they are initiated.

FORMC Co mmands

FORMC commands are summarized in Table 9-3 below and described in the sections that follow.

Table 9-3. FORMC Commands Summary

Commands		Description
Information Commands		
HElp **?**		Displays a list of FORMC commands
Configuration Commands		
FOrmat	disk/tape	Formats the specified disk or tape
SPare	disk track	Spares the specified disk track
VErify	disk/tape	Verifies the integrity of the specified disk or tape
Exit Commands		
ABort **EN**d **EX**it /**E**		Any of these commands causes FORMC to terminate

Abort, End, and Exit (AB) (EN) (EX) (/E)

Purpose: To abort or terminate FORMC.

Syntax: `AB, EN, EX, or /E`

Description:

The utility stops execution when you enter any of these commands in response to any FORMC prompt.

Warning **DO NOT abort the FORMC program during a spare or format operation, as you may jeopardize the integrity of the disk data.**

Help (?)

Purpose: To display all the legal FORMC commands.

Syntax: `HE or ?`

Format Command (FO)

Purpose: To format a disk volume or cartridge tape.

Syntax: `[RU,]FORMC :IH,[list LU], FO, media LU[, interleave]`

list LU	The LU of the hard-copy error-logging device. The default is no hard-copy log.
FO	Calls the format option.
media LU	An LU of the CS/80 disk or tape to be formatted.
interleave	The interleave factor (a decimal number between 1 and 32) to be assigned. The default is to a factor of 1. (Refer to Appendix A for a discussion of interleaving.)

Description:

Caution The FOrmat command issues the Initialize Media command to the entire tape or disk volume. If a disk LU is specified, the entire disk volume will be formatted; CS/80 disks do not allow formatting by subchannels.

Note In order to execute the FOrmat command, FORMC must be scheduled without renaming (as FORMC:IH) and you must have a capability of 60 or greater. (Refer to the *RTE-6/VM Terminal User's Reference Manual* for a description of the user capability levels.) If the rename-inhibit (:IH) is not included in the call, FORMC aborts with the messages:

```
/FORMC: MUST USE THE PROGRAM NAMES 'FORMC'
WITH THE FO AND SP COMMANDS
/FORMC: ABORTED
```

If your assigned user capability is less than 60 and you attempt to format a disk or tape, FORMC aborts with the messages:

```
/FORMC: INSUFFICIENT CAPABILITY
/FORMC: ABORTED
```

The FORMC Formatting Operation

Formatting a disk (FO option) consists of writing the preamble for each block of data on the medium. In addition, the data areas are initialized. Note that the format command issues the initialize medium command to the entire disk volume; for most CS/80 disks, this means the entire disk unit. CS/80 disks do not allow the formatting of RTE subchannels, as with previous disk drives. Disk blocks found defective must be specifically spared using the SP command.

Formatting a CTD tape means something different than formatting a disk. If the tape has not been certified (the initialized media flag on the tape has not been set), a tape certification process is performed. This includes writing a known pattern on every block of the tape and then rereading the tape to determine if there are any defective areas on it. In addition, every 512th block on the tape is allocated as a sparing block. A table of these allocated blocks is created on the tape. Any defective blocks discovered are flagged, and subsequent writes to the tape will not use those blocks. This is known as "skip sparing." Finally, the "initialized media" flag is set on the tape. This certification process can take up to 20 minutes for a short tape (DC 150) or more than one hour for a long tape (DC 600). RTE-6/VM utilities that access CTD tapes will not use a tape until the initialized media flag is set.

If the tape was previously certified (the flag is set), you can either recertify the tape or convert jump spares to skip spares. (In jump sparing, a specific defective block is spared to the first spare block following the bad block.) The recertification process is identical to the initial certification, and any data on the tape will be destroyed in the process. The spare conversion process takes the tape blocks that were spared using jump sparing and flags them so that they are skipped on the next write to the tape. On the next write, the controller will simply renumber the tape blocks, starting at the next good block following the defective block. The spare conversion option will result in loss of data for those blocks that were jump-spared.

Previously certified tapes should not need to be recertified on HP 1000 systems unless there are serious doubts as to the quality of the tape. Any defective areas on the tapes discovered after the certification process by either a normal read or verify operation are automatically flagged as defective and automatically skip spared on any subsequent write operation. Since all HP 1000 utilities use skip sparing, you do not need to use the spare conversion option for HP 1000 CTD tapes. This option is included since other non-HP 1000 systems do use jump sparing, and you may want to use this option when reformatting their tapes.

To execute the format command in runstring mode, you must include the FO and media LU parameters in the string; the optional parameters default as defined above. The rename inhibit parameter must be included in the program descriptor, since the FO option does not allow renaming FORMC.

In non-interactive mode, FORMC aborts with an ILLEGAL RUNSTRING PARAMETER message if you attempt to format a system disk (LU 2 or LU 3) or a disk that is currently mounted. In interactive mode, FORMC issues a warning and user action prompt sequence before a currently mounted disk or the system disks can be formatted.

When you invoke the format option as FO in response to the interactive TASK? prompt, FORMC issues the parameter prompt

```
/FORMC: CS/80 DISK OR TAPE LU?
```

Entering the LU

If a tape is to be formatted (also referred to as tape certification or tape media initialization), enter the LU of the CTD drive on which the tape is mounted. Any LU on a disk volume may be entered; the entire volume is formatted by FORMC. If the LU entered does not correspond to a legal CS/80 disk or tape drive, the message:

```
/FORMC: ILLEGAL CS/80 DISK OR TAPE LU
```

is issued and FORMC prompts again for the LU number.

Tape Formatting

Caution When formatting a CS/80 tape, FORMC will lock the entire disk unit. All accesses to any of the disk LUs on the EQT will be severely limited.

If the FO command is issued to the CTD, FORMC determines if the cartridge has been inserted in the drive, this process requires from 10 to 15 seconds to execute. If the cartridge is not present, the message sequence

```
/FORMC: READY TAPE AND RESTART
/FORMC: FINISHED
```

is issued and FORMC exits. You must insert the cartridge into the CTD tape drive and restart the utility.

When the cartridge is in place, FORMC then determines if the cartridge was previously initialized (certified). If the cartridge shows an UNINITIALIZED MEDIA status, FORMC issues the message sequence

```
/FORMC: TAPE INITIALIZATION(CERTIFICATION) MAY TAKE UP TO AN HOUR
/FORMC: DATA WILL BE DESTROYED ON TAPE LU xx
/FORMC: OK TO PROCEED (Y,N)?
```

If you respond with N, FORMC exits. If your response is Y, FORMC proceeds to certify the entire tape, issues the following message:

```
/FORMC: TAPE FORMATTING COMPLETED
```

and exits.

If the tape is already initialized (certified), FORMC issues this warning and question:

```
/FORMC WARNING: TAPE IS ALREADY INITIALIZED(CERTIFIED)
/FORMC: DO YOU WANT TO RECERTIFY OR CONVERT SPARES(CE,SP)?
```

If you respond with CE, FORMC issues these messages:

```
/FORMC: TAPE INITIALIZATION(CERTIFICATION) MAY TAKE UP TO AN HOUR
/FORMC: DATA WILL BE DESTROYED ON TAPE LU xx
/FORMC: OK TO PROCEED (Y,N)?
```

If you respond N, FORMC exits. If your response is Y, FORMC proceeds to recertify the tape, issues the following message:

```
/FORMC: TAPE FORMATTING COMPLETED
```

and exits.

If you respond to the recertification question with SP, FORMC prompts:

```
/FORMC: JUMP SPARES WILL BE CONVERTED TO SKIP SPARES
/FORMC: DATA MAY BE DESTROYED ON TAPE LU xx
/FORMC: OK TO PROCEED (Y,N)?
```

If you respond with N, FORMC exits. If your response is Y, FORMC proceeds to update the tape spare table by converting jump spares to skip spares, issues this message:

```
/FORMC: TAPE FORMATTING COMPLETED
```

and exits.

In non-interactive mode, FORMC will always perform a certification on new tapes or a recertification on previously initialized tapes.

Disk Formatting

After the correct disk LU has been entered, the prompt

```
/FORMC: INTERLEAVE FACTOR FOR OPTIMAL THROUGHPUT (Y,N)?
```

is issued to determine if you want to specify a record interleave factor. (Refer to Appendix A for a discussion of interleaving.) An answer of Yes means that you want FORMC to supply the default optimal throughput factor of 1, which is sufficient for most applications. If you answer No, FORMC expects you to supply the interleave factor, and issues the prompt

```
/FORMC: INTERLEAVE FACTOR?
```

Your response to this question must be a decimal number in the range of 1 to 32. If an incorrect response is entered, the message/prompt set

```
/FORMC: ILLEGAL INTERLEAVE FACTOR, LEGAL RANGE 1 TO 32
/FORMC: INTERLEAVE FACTOR?
```

is issued and FORMC waits for the correct entry.

FORMC then locates all disk LUs that are on the same volume and thus subject to formatting. If any of the disk LUs are still mounted, FORMC issues the message set

```
/FORMC: DISMOUNT LU
   <lu 1>,<lu 2>.....................,<lu n>

/FORMC: DISMOUNT LU'S AND RESTART
/FORMC: FINISHED
```

and exits. The LUs listed in the message must then be dismounted using the FMGR DC command, and FORMC must be restarted. With the LUs dismounted, FORMC locks all LUs to be formatted and issues the message

```
/FORMC: DATA WILL BE DESTROYED ON LU
   <lu 1>,<lu 2>.....................,<lu n>
/FORMC: OK TO PROCEED (Y,N)?
```

A response of Yes causes FORMC to proceed to format the entire disk volume. The message

```
/FORMC: DISK FORMATTING COMPLETED
```

is then issued and FORMC exits. A response of No causes FORMC to unlock all LUs and to exit without formatting any disks. If a system disk (LU 2 or LU 3) is to be formatted, the warning message and prompt

```
/FORMC WARNING: SYSTEM WILL BE DESTROYED.
   NOW IS THE TIME TO REPLACE THE SYSTEM CARTRIDGE.
/FORMC: OK TO PROCEED (Y,N)?
```

FORMC expects a Yes or No answer. The removable system disk should be replaced with the disk to be formatted before you respond to the prompt. Be aware that FORMC will format the entire disk volume, including the system subchannels, if your response is Yes and you do not remove the system disk. FORMC terminates if your response is No.

After the formatting is complete, FORMC issues the message:

```
/FORMC: REPLACE SYSTEM CARTRIDGE TO RETURN TO SYSTEM.
/FORMC: OK TO RETURN (Y,N)?
```

FORMC expects a Yes or No answer. After the system disk is replaced, a Yes response causes a return to the host system. A No response causes FORMC to display the message

```
/FORMC: FINISHED
```

and to execute an infinite loop. The system must then be restored and rebooted.

Spare Command (SP)

Purpose: Spares one or more defective blocks within a track of a CS/80 disk.

Syntax: `[RU,]FORMC :IH,[list_LU], SP, media_LU, track`

list LU	The LU of the optional hard-copy error-logging device. The default is to no hard-copy log.
SP	Calls the spare option.
media LU	The LU of the CS/80 disk or CTD disk cache memory to be spared.
track	The track containing bad blocks to be spared.

Caution In order to execute the SPare command, FORMC must be scheduled without renaming (as FORMC:IH) and you must have a capability of 60 or greater. (Refer to the *RTE-6/VM Terminal User's Reference Manual* for a description of the user capability levels.)

If the rename-inhibit is not included in the call, FORMC aborts with the messages

```
/FORMC: MUST USE THE PROGRAM NAMED 'FORMC'
        WITH THE FO AND SP COMMANDS
/FORMC: ABORTED
```

If your assigned user capability is less than 60 and you attempt to spare a disk track, FORMC aborts with the messages

```
/FORMC: INSUFFICIENT CAPABILITY
/FORMC: ABORTED
```

Description:

This command can be used only with CS/80 disks, including the CTD disk cache memory area. The disk cache is divided into four logical tracks, and SP can be used to spare one of these cache tracks. (Refer to the *RTE-6/VM Driver DVM33/DVN33 Reference Manual*, part number 92084-90025, for a description of the disk cache memory scheme.)

Sparing (SP option) is the process of removing a defective disk block from active service. The process individually spares defective blocks identified by the verify operation. Sparing is not required for CTD tapes because defective blocks are flagged both by the format operation and during a normal read operation when bad tracks are discovered. They are automatically spared on any subsequent write operation, a process known as skip sparing. However, the spare operation can be used to spare out one or more defective blocks within a track of the CTD disk cache memory area.

Each CS/80 disk has a specified number of physical tracks reserved for use as spares when a defective block is detected. Each physical track also contains a spare block. (The spare tracks and blocks are known only to the controller.) When a defective block is spared within a physical track, the track is rearranged to bypass the defective block and to use the spare block. If another defective block is subsequently spared on that physical track, the entire physical track is then bypassed and the data is written to another physical track from the spare-track pool. In either case, the existence of spared tracks/blocks is transparent to the user.

The FORMC Sparing Operation

When you invoke the spare option as SP in response to the interactive TASK? prompt, FORMC issues the parameter prompt

```
/FORMC: CS/80 DISK OR TAPE LU?
```

Enter the LU of the disk that contains defective blocks to be spared. If a CTD tape LU is entered, the disk cache memory area of the CTD is spared. If the specified LU does not correspond to a legal CS/80 disk or tape, the message

```
/FORMC: ILLEGAL CS/80 DISK OR TAPE LU
```

is issued and FORMC reprompts for the LU number. If the LU is that of a tape and the tape does not have a generated disk cache memory, the message

```
/FORMC: NO CACHE DEFINED FOR LU <number>
/FORMC: ABORTED
```

is issued and FORMC exits. When a legal LU number has been entered, FORMC prompts with

```
/FORMC: TRACK TO BE SPARED?
```

Enter the number of the track containing the bad block or blocks. If the number entered is not within the legal range of track numbers for that LU, the message

```
/FORMC: ILLEGAL TRACK NUMBER, LEGAL RANGE 0 TO <last track>
```

is issued, where <last track> is the number of tracks on the LU minus one. FORMC then reprompts for the track number. When a legal track number is entered, FORMC checks the track for bad blocks. Note that the SP option must be invoked separately for each track to be spared.

Once the LU number and track number have been determined, either interactively or through the runstring, FORMC performs an error-rate test on the specified track to determine which blocks, if any, are defective on the track. As each defective block is found, it is removed from active service and the message

```
/FORMC: PHYSICAL BLOCK <address> SPARED FOR <number> BLOCKS.
```

is issued. The <address> is the physical block address and <number> is the physical number of blocks spared. Since there may be more than one bad area covered by the specified track, more than one message may be issued. If the next to last spare track is used, the warning

```
/FORMC WARNING: ONE SPARE PHY. TRACK LEFT. CONTACT SYSTEM MANAGER.
```

Several more sparing operations may be performed if the disk only needs one spare block per track, however, the disk will only be able to use one more physical track for sparing if this need arises. The warning indicates that there is something seriously wrong with the drive or the disk. A complete diagnostic check of the drive and disk should be performed.

If the sparing operation cannot be performed to retain the data in a bad block, FORMC issues the warning

```
/FORMC: DATA MAY BE DESTROYED ON LU <number> TRACK
   <track 1>,<track 2>.............,<track n>
/FORMC: OK TO PROCEED (Y,N)?
```

A response of Yes results in sparing without retaining the data on the disk tracks indicated by <track 1> through <track n>. A No response causes FORMC to terminate.

If the disk LU bound crosses a physical track that may actually cover another disk LU (such as another FMP directory), FORMC issues the warning

```
/FORMC: DATA MAY BE DESTROYED ON LU
   <lu 1>,<lu 2>...................,<lu n>
/FORMC: OK TO PROCEED (Y,N)?
```

If the data has been backed up, a response of Yes can be entered and FORMC will lock all LUs that might cross physical tracks and perform the sparing operation. A response of No causes FORMC to terminate.

After a successful sparing operation, FORMC issues the completion message

```
/FORMC: DISK SPARING COMPLETED
```

If no bad blocks are found on the specified disk, the message

```
/FORMC: NO BAD BLOCKS - SPARING NOT PERFORMED
```

is issued and FORMC exits.

Verify Command (VE)

Purpose: Checks the integrity of the specified disk LU or CTD tape.

Syntax: `[RU,]FORMC,[list_LU],VE,media_LU[,start,numb]`

list LU	The LU of the hard-copy error-logging device. The default is no hard-copy log.
VE	Calls the verify option.
media LU	An LU of the CS/80 disk or tape to be verified.
start	The number of the disk track or tape block at which the verify is to begin.
numb	The decimal number of disk tracks or tape blocks to be verified. If the start and numb parameters are omitted in runstring mode, the entire disk or tape will be verified.

Description:

In runstring mode, you must include the VE and media LU parameters in the string; the optional parameters default as defined above.

Verification (VE option) is a nondestructive process that checks the integrity of a specified disk LU or CTD tape through reading the CRC information (checksum) in each block of the disk or tape. The entire disk/tape can be verified by FORMC, or specific tracks/blocks can be isolated and verified. Note that a block as defined for a CTD tape consists of 512 words, while a CS/80 disk block consists of 128 words. All track references are to RTE tracks, not physical disk tracks.

FORMC Verify Operation

When you invoke the verify option as VE in response to the TASK? prompt, FORMC issues the parameter prompt

```
/FORMC: CS/80 DISK OR TAPE LU?
```

Enter the LU of the disk or tape to be verified. If the number given does not correspond to a legal CS/80 disk or tape LU, the message

```
/FORMC: ILLEGAL CS/80 DISK OR TAPE LU
```

is issued and FORMC reprompts for the number. If a CTD tape is to be verified, FORMC tests to determine if the cartridge to be verified has been inserted into the drive. (The time required to execute this routine is from 10 to 15 seconds.) If the cartridge is not present, the message sequence

```
/FORMC: READY TAPE AND RESTART
/FORMC: FINISHED
```

is issued and FORMC exits after verifying the CTD disk cache memory area associated with the tape (if the cache exists). You must insert the CTD cartridge and restart the utility. If the cartridge has not been initialized (formatted), FORMC issues the messages

```
/FORMC: UNINITIALIZED TAPE MEDIA - FORMAT TAPE
/FORMC: TASK?
```

and returns to the interactive TASK state. You must enter the FO command to format the tape. (Refer to the "Format Command" section for details.)

When the formatted cartridge is in place and the correct LU has been entered, the following sequence of prompts is issued, as appropriate:

```
/FORMC: VERIFY ENTIRE DISK LU (Y,N)?
```

or

```
/FORMC: VERIFY ENTIRE TAPE (Y,N)?
```

If your response is Yes, FORMC proceeds in verifying the entire disk LU or tape. If you respond No, FORMC prompts,

```
/FORMC: START TRACK NUMBER?
```

or

```
/FORMC: START TAPE BLOCK NUMBER?
```

The number entered must be an integer between zero and the number of tracks or blocks on the medium minus one. In the case of tape verification, the number is between 0 and 65535; if a negative number is entered as the starting block, it is interpreted as positive (−2 = 65534). If an incorrect number is entered (a number larger than the disk/tape capacity or, in the case of disk verification, a negative number) the message

```
/FORMC: ILLEGAL TRACK NUMBER, LEGAL RANGE 0 TO <last track>
```

or

```
/FORMC: ILLEGAL BLOCK NUMBER, LEGAL RANGE 0 TO <last block>
```

is issued, where <last track> or <last block> is the number of the last track/block minus one.

When the correct number is issued, FORMC then prompts with

```
/FORMC: NUMBER OF DISK TRACKS?
```

or

```
/FORMC: NUMBER OF TAPE BLOCKS?
```

One or more tracks or blocks must be specified. If the number given is greater than the capacity of the medium, all tracks or blocks following the start number are verified. Note that a negative number may be specified for the number of tape blocks, as it is interpreted as positive (−5 = 65531). If zero tracks or blocks are specified or a negative number of disk tracks is specified, the message

```
/FORMC: ILLEGAL NUMBER OF TRACKS
```

or

```
/FORMC: ILLEGAL NUMBER OF BLOCKS
```

is issued and the prompt is repeated. When the correct number is entered, the verification is performed. If the CS/80 tape to be verified has an associated disk cache memory, the disk cache also is verified. When a defective track or block is encountered, FORMC issues the message

```
/FORMC: LU <disk LU> - BAD TRACK <number>
```

or

```
/FORMC: LU <tape LU> - BAD BLOCK <number>
```

or

```
/FORMC: LU <disk LU> - BAD TRACK <number> IN CACHE
```

where <number> is a decimal integer identifying the defective track or block.

After the verify is complete, FORMC issues the message

```
/FORMC: DISK VERIFY COMPLETED <number> BAD TRACK(S).
```

or

```
/FORMC: TAPE VERIFY COMPLETED <number> BAD BLOCK(S).
```

or

```
/FORMC: TAPE CACHE VERIFY COMPLETED <number> BAD TRACK(S).
```

where <number> is either a decimal integer defining the number of defective tracks or blocks or is NO to indicate that no defective tracks or blocks were encountered. Bad tracks can be spared by passing the disk or tape LU and the track number to the SP command.

If an end-of-file error is encountered during the tape verification operation, FORMC issues the message

```
/FORM: LU <tape LU> - END OF FILE DETECTED IN TAPE BLOCK <number>
```

where <number> is a decimal integer identifying the tape block containing the End of File. FORMC will then resume the verification operation with the next tape block following the end-of-file.

Error Messages

CS/80 devices return 10 words of status information:

word 1 – identification field
word 2 – reject errors (severity #1)
word 3 – fault errors (severity #2)
word 4 – access errors (severity #3)
word 5 – information errors (severity #4)
words 6–10 – parameter area for words 2–5

One or more error message may be displayed, in the following format:

```
/FORMC: eeeeee ERROR xxxxxxB
<mnemonic error code>
IDENTIFICATION FIELD: yyyyyyB QSTAT: q
PARAMETER FIELD P(1) THRU P(10):
ppppppB ppppppB ppppppB ppppppB ppppppB
```

where:

`eeeeee` is the error type: REJECT, FAULT, ACCESS, INFORMATION.

`xxxxxxB` is the error field, in octal.

`yyyyyyB` is the identification field, in octal.

`q` is the error-reporting message from the device.

`ppppppB` is the five-word parameter area, in octal, for the error.

If an undefined error bit is encountered in one of the status words, the message

```
    FATAL INTERNAL ERROR
```

replaces the <mnemonic error code> in the message. Refer to the *RTE-6/VM Driver DVM33/DVN33 Reference Manual* for a description of the error reporting scheme and a complete summary of all mnemonic error codes.

10

Online Driver Replacement Utilities

The driver replacement utilities DRREL and DRRPL allow you to selectively relocate and install device drivers online, thus eliminating the need to regenerate the system each time a driver is changed. DRREL relocates the drivers and DRRPL performs the installation. The actual driver installation (DRRPL DR command) can be issued only by the System Manager or in a non-session environment. Further restrictions pertaining to driver installation are defined in the DRRPL description. Note that a driver will be recognized only if it conforms to all module and entry-point naming conventions defined in the *RTE Operating System Driver Writing Manual*, part number 92200-93005.

A full driver partition or all of the system driver area can be installed at one time, either in memory or permanently on disk. Table entries are automatically constructed for the replacement drivers and are installed in the equipment table (EQT), driver-mapping table, and interrupt table.

Finding Space for Drivers

Space for replacement drivers can be obtained in one of three ways: overlaying an existing driver, assigning a driver partition to unused physical pages, or reserving space via a dummy driver at system generation.

Overlaying an Existing Driver

When you overlay an existing driver, you must be sure that other drivers are not affected in the process. You cannot perform a memory replacement of driver partition number one, which contains the system disk driver. In systems generated with base page linking, you can determine the precise bounds of each driver from the system generation listing. However, if the system is generated with current page linking, the links are placed before and after drivers and can be shared among drivers. In this case, you should replace all drivers in the partition or system driver area or replace only the last driver in the area.

Using Available Pages

By assigning a driver to an unused physical page in the system, you are essentially creating a temporary driver partition. Free physical pages can be identified using the DRREL FR command.

Creating a Dummy Driver

Space for a new driver can be reserved by creating a dummy driver when the system is generated. As an example, you can use the following code to create a two-page driver partition and to reserve base page links:

```
      NAM   DVY77,0                 92084-16602 REV.2340 830719 DUMMY
      EQT1  EQU 1660B               CURRENT EQT WORD 1
      EQT15 EQU 1774B               CURRENT EQT WORD 15
      ENT IY77,CY77

      ORB
      REP 25                        RESERVE 25
      DEC 0                         BASE PAGE LINKS

      ORR
IY77  NOP
      CLB                           CLEAR TRANS LOG
      LDA =B4                       OPERATION WAS IMMEDIATELY COMPLETED
      JMP IY77,I                    EXIT
*
*
CY77  NOP
      CLB                           CLEAR TRANS LOG
      LDA EQT1,I                    CK SPURI INT
      AND =B77777                   MASK LIST PTR
      SZA                           REQ IN PROG?
      JMP CY77,I                    YES, TAKE COMPLETION RETURN

      STA EQT15,I                   CLEAR T.O.
      ISZ CY77                      TAKE CONTINUATION RETURN
      JMP CY77,I

REMSP EQU (2048-(*-IY77))           COMPUTE # WORDS LEFT IN 2 PAGES
      BSS REMSP                     RESERVE THE REST OF THE DRIVER PARTITION
      END
```

At system generation, the response for the EQT prompt to reserve the partition and to reserve a 20-word EQT extension would be (this example assumes EQT table entry 14):

```
EQT 14?
27,DVY77,X=20
```

If you want to place the driver in the system driver area, the response would be:

```
EQT 14?
27,DVY77,S
```

Base Page Links

You should reserve base page only if the replacement driver contains an ORB, if the driver is to fit into a limited logical address space, or if the replacement driver is large.

Base page links for Table Area 1, Table Area 2, SSGA, partitions, and system driver areas are allocated starting from 1644B down. In the generation listing, the BP pointer always gives the address of the next link to be allocated, as:

```
BP LINKAGE 01611

DP 03 <<PAGE 00030>>

DVY77(0099) 06000 11777
   *IY77    06000
   *CY77    06001

BP LINKAGE  01541
```

Base page links used by a driver at generation are shared by all the drivers loaded after the driver that forced the allocation. Therefore, you can use the base page area defined in the generation only if you are replacing the driver last generated or if you are replacing a driver that declared a base page with an ORB. Otherwise, you should use all current page links. The ORB areas should contain NOPs because they are scanned by the generator while it looks for base page links.

Using this example, you would respond to the DRREL prompts as: high base page address, 01611, low base page address (−1), 01541, high logical address, 11777, and low logical address, 06000. (Refer to the DRREL discussion below for the actual prompt format and sequence.)

Loading the Driver Replacement Utilities

Both DRREL and DRRPL require extra memory for building interim tables during the driver relocation/replacement processes. The loading sequence thus requires sizing the programs to allow two pages for table generation. To load the utilities online, use the load sequence:

```
:RU,LOADR
 /LOADR: RE,%DRREL
 /LOADR: EN
:SYSZ,DRREL
 aaaaa ppp
```

or

```
:RU,LOADR
 /LOADR: RE,%DRRPL
 /LOADR: EN
:SYSZ,DRRPL
 aaaaa ppp
```

SYSZ calls the system command SZ from FMGR. The program size is returned, with aaaaa = address of last word +1 and ppp = minimum partition size. Now resize the program as:

```
:SYSZ,DRREL,<ppp+2>
```

or

```
:SYSZ,DRRPL,<ppp+2>
```

For a large number of EQTs, the driver replacement utilities may need three or more pages of table space. In this case, load the driver as large background and resize the program. The following example uses three pages of table space to resize the program:

```
:SYSZ,DRREL,<ppp+3>
```

or

```
:SYSZ,DRRPL,<ppp+3>
```

The driver replacement utilities use the LOADR library, so it is not necessary to specify a library search. Be aware that the driver replacement utilities may not be loaded as extended background.

Driver Relocation Utility (DRREL)

The driver relocation utility DRREL allows you to selectively relocate one or more driver routines, either to a driver partition or to the system driver area, with or without base page links. DRREL can be run interactively from your terminal (the default state) or from a command file. If the input source is a command file, the parameters must be in the same sequence as the interactive prompts.

Calling DRREL

To call DRREL, enter the following runstring:

```
:[RU,]DRREL[,input source]
```

Alternatively, you can run DRREL interactively. DRREL first prompts for the configuration parameters in the sequence listed below, followed by the relocation command prompt. When your response to a prompt is an octal value, the number must be given with a B suffix, as nnnnB.

```
BASE PAGE LINKS (Y,N)?
```

If you have generated a dummy driver to reserve base page links, respond Y and DRREL will allocate the links. (Refer to the previous section “Finding Space for Drivers” for instructions on creating a dummy driver module during system generation.)

If an area was not reserved, respond N and DRREL will attempt to load the driver or drivers using all current page links. (This is the default state.) Note, however, that large drivers or drivers that must fit into a tight logical address may be required to use base page even if the use of base page links is not specified. DRREL always scans base page to see if it can reuse already existing links.

If your response is DRREL, DRREL prompts with

```
HIGH BP ADDRESS?
```

and

```
LOW BP ADDRESS - 1?
```

These addresses may be obtained from the system generation listing, as described in the section “Finding Space for Drivers.” (Note that the low base page address must be specified as address minus 1.) Since the default is to current page linking (unless changed using the relocation command BP), base page links are used only if there is not enough space on the current page. DRREL still scans base page for existing links, but only outside of the allocation area.

If the driver is replaced more than once, the base page links allocated to it may be overwritten, since they are not used by other code.

DRREL then displays the logical address boundaries of the driver partition and the system driver area, as:

```
DRIVER PARTITION    n PAGES          lllll    hhhhh
         SYSTEM DRIVER AREA          lllll    hhhhh
```

where lllll and hhhhh are the low and high logical address of the partition and system driver area. DRREL then prompts for the low and high logical boundaries of the address space in which the driver code and links can be placed.

```
LOW LOGICAL ADDRESS?
```

and

```
HIGH ADDRESS?
```

When more than one driver is to be relocated, the range specified must include sufficient space for all drivers. If the relocated drivers exceed the boundaries, an error message is issued and DRREL exits.

If the system was generated with all base page links, the responses can be taken directly from the generation listing. However, if the system was generated with current page links, it is safest to relocate all of the drivers in the partition or System Driver Area.

Finally, DRREL issues the prompt:

```
OUTPUT FILE?
```

Enter the namr of a disk file to which the memory-image code will be sent. The file will be created as a type 6 file and may be placed on any disk LU. This file is used by DRRPL to install the driver.

DRREL now prompts with:

```
/DRREL:
```

and any of the commands shown in Table 10-1 may be issued. At the end of the successful driver relocation, DRREL lists the number of drivers relocated, the beginning and ending logical addresses, and the base page used.

DRREL Commands

The BPAGE or CPAGE commands, if used, must be given first; all other commands are order-independent. Only the first two characters are required to specify a command.

Table 10-1. DRREL Commands Summary

Command	Description
??	List all available commands with a brief description. The listing format is shown in Figure 10-1.
BPAGE	All linkage is to be through base page. (If no base page was given, links cannot be allocated.)
CPAGE	Linkage is to be through current page as much as possible. If base page area is available, it will be used only if room cannot be found on the current page. This is the default state.
LL,namr	Specifies the LU of the list device or the list file namr. The list file may not already exist, unless defined as a list file (' in the first character position of the file name). Non-interactive LUs will be locked. The default is to your terminal.
ECHO	Echo all prompts, commands, and comment lines to the list device.
RE,namr	Load all modules into the given disk file. The module name, load addresses, entry points, and base page linkages are printed on the list device.
LO	List the current relocation address.
LO,+	Set the current relocation address to the beginning of the next page.
LO,addr	Set the current relocation address to the given address. The requested address must be greater than the current relocation address and must be below the high address boundary.
SEARCH	Search the system library for undefined externals. The module names, load addresses, entry points, and base page linkages are printed on the list device.
DISPLAY	Display all undefined externals on the list device.
END EXIT /E	End command input. DRREL will search the system library, if required to satisfy any undefined externals and will finish loading the driver. The END, EXIT or /E command may not be given unless a driver is relocated. The /A command must be used to abort without relocating a driver.
/A	Abort utility. The /A response can be given to any DRREL prompt and must be used to exit DRREL if a driver is not relocated.
*	Specifies a comment line. DRREL ignores everything on a line beginning with the comment character. If ECHO is specified, all comment lines are echoed to the list device.

DRREL Example

In the following example, two drivers, DVA12 and DVR23, are relocated without using base page linkage, and the memory-image code is placed in output file DP3. The addresses at which each driver initiation and continuation sections will be located in the driver partition are displayed following the RE commands for the drivers. After the user-entered END command, DRREL issues a message defining the number of drivers relocated, the logical address range for the drivers, the output file name and time the file was created, and the utility exit line.

```
BASE PAGE LINKS (Y,N)? NO


  DRIVER PARTITION      2 PAGES      4000   7777
           SYSTEM DRIVER AREA      20000  23672

LOW LOGICAL ADDRESS? 4000B

HIGH ADDRESS? 7777B

OUTPUT FILE? DP3

/DRREL: RE,%DVA12

DVA12               4000 4700 92001-16020 780511 REV 1826
  IA12                4000
  CA12                4215

/DRREL: RE,%DVR23

DVR23               4701 5614 92202-16001 REV.1913 - 790202
  I.23                4701
  C.23                5560

/DRREL:  DI

  NO UNDEFS

/DRREL:  END

******     2  DRIVERS   ******
LOGICAL ADDRESS   4000  5614
    NO BASE PAGE
/DRREL:  FILE DP3   READY AT 8:26 AM TUE., 14 JULY, 1989
/DRREL:  END
```

DRREL Error Messages

The following error messages can be issued by DRREL. Fatal errors that will cause DRREL to abort are flagged with (F) following the error definition. Note that all errors are fatal when the utility is run non-interactively.

D-IL PRM INPUT ERROR

An invalid command or response was given to a prompt.

D-IL ADD OUT OF RANGE

A bad address was given. Logical addresses given must be within the driver partition or the System Driver Area. Base page addresses must be between the base page fence and the System Communication Area. An address given with the LO command must be greater than the current relocation address and within the high address bound.

D-IL CMD ILLEGAL COMMAND

The BP, CP, and LL commands may not be given after the first RE command. The END command may not be given if a driver has not been relocated.

D-IL ALC ILLEGAL ALLOCATION

A module attempted to allocate common, save, pure code, or EMA area. DRREL does not allow any of these allocations. (F)

D-RF EMA or D-ML EMA

Illegal EMA reference. Drivers may not access EMA. (F)

D-IL REL

Assembler or compiler produced an illegal relocatable module. Recompile and try again. (F)

D-RE SEQ RECORD OUT OF SEQUENCE

May be caused by a corrupt relocatable file. Recompile and try again. (F)

D-DU ENT DUPLICATE ENT

Duplicate entry points were encountered. Rename or remove one of the duplicate ents. (F)

D-CM BLK

Common Block error. Drivers may not allocate common. (F)

D-OV SYM SYMBOL TABLE OVERFLOW

DRREL does not have enough room to do relocation. Use SZ operator command to expand the size of DRREL, use the SE command, or reload DRREL as a large background program. (F)

D-OV FIX FIXUP TABLE OVERFLOW

DRREL does not have enough room to do relocation. Use SZ operator command to expand the size of DRREL, use the SE command, or reload DRREL as a large background program. (F)

D-OV MEM MEMORY OVERFLOW

The relocated code exceeds the logical address boundaries. Find more room for the drivers. (F)

D-IL REC ILLEGAL RECORD

A bad relocatable record was encountered. The file given may not have been a relocatable file, or the file has been corrupted. Give correct file name or try recompiling. (F)

D-CK SUM CHECKSUM ERROR

A bad relocatable file was given. Specify the correct file or recompile. (F)

D-OV BSE BASE PAGE OVERFLOW

Not enough base page links were available to the driver. This may be caused by an ORB encountered when no base page is available, or the space available for links on the current page and base page is not sufficient to relocate the driver. Add base page links or expand the logical address space, which increases current page links available. If the module causing the error is a page or more in length, try starting relocation of the module on a page boundary (do LO,+ if necessary). Setting the current location slightly greater using the LO command may help. (F)

D-UN EXT UNDEFINED EXTERNALS

An END command was attempted, and undefined externals were not satisfied by the system library search. Supply the missing entry points.

D-SY LEN WARNING: SYMBOL TRUNCATED TO 5 CHARACTERS

An entry point or external symbol is greater than five characters in length. DRREL truncates these symbols before storing them in its output file (the input file for DRRPL).

Driver Replacement Utility (DRRPL)

DRRPL controls the installation of replacement drivers relocated by DRREL. The actual installation can only be done by the System Manager or in non-session mode from the system console. Since DRREL searches for system entry points and may reuse existing base page links, drivers must be installed in the same system on which they were relocated. The driver installation command DR is rejected if DRRPL is loaded in extended background.

DRRPL can be run interactively from your terminal (the default state) or from a command file. If the input source is a command file, the installation parameters must be in the same sequence as the DR command prompts. Note that the DR command itself need not be included in the file.

Calling DRRPL

To call DRRPL, enter the following runstring:

```
:[RU,]DRRPL[,input source]
```

If run interactively, DRRPL issues the prompt

```
/DRRPL:
```

and any of the commands shown in Table 10-2 below may be issued. In particular, you should issue the DD/MD and FR commands for a listing of the current driver configuration and the number of free pages, since this information is used during the driver installation phase.

Table 10-2. DRRPL Commands Summary

Command	Description
??	List all available commands with a brief description.
LL,namr	The LU of the list device or list file namr. The list file must not already exist, unless it is defined as a list file (' in the first character position of the namr). Non-interactive LUs are locked. When the DR command is issued, all prompts and command inputs are echoed to the list device. The default is your terminal.
/A	Abort utility. The /A may be issued in response to any DRRPL prompt.
*	Specifies a comment line. DRRPL ignores everything on a line beginning with the comment character. Comment lines that are part of the input given with the DR command are echoed to the list device.
DI, <#>,<#> MI <#>,<#>	List the interrupt table/trap cells from the system on disk (DI) or in memory (MI). <#>,<#> represent the beginning and ending EQT numbers. The listing format is shown in Figure 10-1.
DE <#>,<#> ME <#>,<#>	List the equipment table entries from the system on disk (DE) or in memory (ME). Only static information is listed. Note that the EQT extension size listed by this command is taken from the system on disk because this value is wiped out in some memory-resident EQT entries. <#>, <#> represent the beginning and ending EQT numbers. The listing format is shown in Figure 10-2.
MD <#>,<#> DD <#>,<#>	List the driver configuration in memory (MD) or on disk (DD). All EQTs listed are grouped according to the driver partition or the System Driver area they reference. <#>, <#> represent the beginning and ending EQT numbers. The listing format is shown in Figure 10-3.
FR	List the physical pages that are free in the system. Free pages may be obtained by declaring the pages as bad using the reconfigurator (not within SAM extension). Note that pages in high memory are most frequently free.
DR,namr	Install the new drivers. Namr can be an LU or file that contains the output of DRREL. The sequence of prompts described in the section "Replacement Driver Specification" begins when this command is entered. Note that this command is valid only if: 1. You are the System Manager (capability >60) or 2. The command is entered at the system console in a non-session environment, AND 3. DRRPL is invoked as RU,DRRPL:IH, to inhibit renaming of the program.
END EXIT /E	Exit DRRPL. Any of these commands may be used to terminate the utility. The /E command also is used to indicate the end of command prompt groups.

Figure 10-1 below shows the DI/MI listing format.

```
    /DRRPL:  DI,10,25

    S.C.         INTERRUPT TABLE          TRAP CELL       (DISK)
    12                    0               JSB 1553,I
    13                    0               JSB 1544,I
    14                    0               JSB 1544,I
    15                EQT 10              JSB 1544,I
    16                EQT  1              JSB 1544,I
    17                EQT  2              JSB 1544,I
    20                 PRMPT              JSB 1544,I
    21                EQT  8              JSB 1544,I
    22                EQT  8              JSB 1544,I
    23                EQT  6              JSB 1544,I
    24                 PRMPT              JSB 1544,I
    25                 PRMPT              JSB 1544,I
```

```
Notes:  Select code S.C.:  octal value
        Interrupt Table:   decimal EQT entry, or program name,
                              or octal value
         Trap Cell:        JSB,I, or
                           JMP,I to octal address, or octal
                              value
```

Figure 10-1. DRRPL DI/MI Command Listing Format

Figure 10-2 below shows the ME/DE listing format.

```
    /DRRPL:  DE,10,25

                  EQT   (DISK)

    EQT   10  =  15  TYPE  65       T=      3  X=   7  IN= 10307    CC= 11162
    EQT   11  =  40  TYPE  65       T=      3  X=   7  IN= 10307    CC= 11162
    EQT   12  =  41  TYPE  65       T=      3  X=   7  IN= 10307    CC= 11162
    EQT   13  =  42  TYPE  66       T=      0  X=  12  IN=  6150    CC=  6161
    EQT   14  =  42  TYPE  66       T=      0  X=   0  IN=  6150    CC=  6161
    EQT   15  =  43  TYPE  66       T=      0  X=  12  IN=  6150    CC=  6161
    EQT   16  =  43  TYPE  66       T=      0  X=   0  IN=  6150    CC=  6161
    EQT   17  =  44  TYPE  66       T=      0  X=  12  IN=  6150    CC=  6161
    EQT   18  =  44  TYPE  66       T=      0  X=   0  IN=  6150    CC=  6161
    EQT   19  =  45  TYPE  67 D     T=      0  X=   0  IN=  6126    CC=  6072
    EQT   20  =  46  TYPE  37       T=  32767  X=  81  IN=  6122    CC=  6706
    EQT   21  =  24  TYPE  05    B  T=  12000  X=  13  IN=  6102    CC=  6200
    EQT   22  =  25  TYPE  05    B  T=  12000  X=  13  IN=  6102    CC=  6200
    EQT   23  =  26  TYPE  05    B  T=  12000  X=  13  IN=  6102    CC=  6200
    EQT   24  =  27  TYPE  05    B  T=  12000  X=  13  IN=  6102    CC=  6200
    EQT   25  =  30  TYPE  05    B  T=  12000  X=  13  IN=  6102    CC=  6200

Notes:  EQT - entry number, decimal;
        SC  - select code, octal;
        Type  - equipment type code, octal;
        D     - DMA is used;
        B     - automatic output buffering is used;
        S     - system driver - does not do own mapping (not shown);
        M     - system driver - does own mapping (not shown);
        T     - timeout interval, decimal;
        X     - EQT extension length in words, decimal;
        IN  - driver initiation section address, octal;
        CC  - driver continuation/completion section address, octal.
```

Figure 10-2. DRRPL ME/DE Command Listing Format

Figure 10-3 below shows the MD/DD listing format.

```
/DRRPL:  MD

      DRIVER CONFIGURATION    (MEMORY)
  DRIVER PARTITION   2 PAGES       6000   11777
         SYSTEM DRIVER AREA        26000  27774
  DRIVER PARTITION PAGES        3 - 4       42 - 59

 DRIVER PARTITION   1   <<PAGE     3>>
EQT  1 = SC 16 TYPE 32  D    T=  0  X=  0  IN=   7333  CC=   6145
EQT 10 = SC 15 TYPE 65       T=  3  X=  7  IN=  10307  CC=  11162
EQT 11 = SC 40 TYPE 65       T=  3  X=  7  IN=  10307  CC=  11162
EQT 12 = SC 41 TYPE 65       T=  3  X=  7  IN=  10307  CC=  11162
  .              .             .                .            .
  .              .             .                .            .
  .              .             .                .            .
 DRIVER PARTITION  10  <<PAGE    58>>
EQT  7 = SC 37 TYPE 12   B   T=  0  X=  5  IN=   6000  CC=   6737
EQT  8 = SC 21 TYPE 23 D B   T=  0  X=  0  IN=   7611  CC=  10734

 SYSTEM DRIVER AREA    <<PAGE    11>>
EQT  9 = SC 11 TYPE 50    S  T=  0  X=  0  IN=  26000  CC=  26173
EQT 49 = SC 71 TYPE 43    M  T=  0  X= 18  IN=  26660  CC=  26663
EQT 50 = SC 72 TYPE 43    M  T=  0  X= 18  IN=  26660  CC=  26663
EQT 51 = SC 73 TYPE 43    M  T=  0  X= 18  IN=  26660  CC=  26663
```

```
Notes:  Addresses are given in octal;
        Driver partition page numbers are given in decimal.
```

Figure 10-3. DRRPL MD/DD Command Listing Format

Replacement Driver Specification

The installation process for the drivers and the associated I/O tables does not begin until the entire sequence of prompts has been completed. The prompt sequence is summarized in Figure 10-4 and described in detail in the subsequent paragraphs. If your response is an octal number, the number must be entered with the suffix B, as nnnnB; decimal numbers do not require a suffix.

```
MEMORY OR PERMANENT REPLACE (ME,PE)?

PHYSICAL PAGE OR DRIVER PART. (PG,DP)?        (Driver partition only)
    DP:  DRIVER PARTITION NO.?
    PG:  DP STARTING PHYSICAL PAGE?

EQT NUMBER?
EQ TYPE CODE:                                 (Mult. equip. types only)
DMA (Y,N)?
AUTOMATIC OUTPUT BUFFERING (Y,N)?
SELECT CODE?
TIMEOUT INTERVAL?
OWN MAPPING (Y,N)?                            (System Driver Area only)
EQT NUMBER (/E TO END)?

SELECT CODE (/E TO END)?
    ENTRY TYPE (EQT,PRG)                      (Driver Partition only)
    ENTRY TYPE (EQT,PRG,ENT,ABS)              (System Driver Area only)
    ENTRY TYPE (ENT,ABS)                      (Select Code 4 only)
        EQT:   EQT NUMBER?
        PRG:   NAME?
        ENT:   NAME?
        ABS:   VALUE?

MEMORY REPLACE READY (Y,N)?

PERMANENT REPLACE READY (Y,N)?
```

Figure 10-4. Driver Replacement Prompt Sequence

Note The display formats cited in this section are examples only. The displays you see when running the driver replacement utility are those related to your specific system configuration and the DRREL output file you have created.

Issuing the DR,namr command causes DRRPL to enter the driver replacement phase of the utility. The DRREL output file is read into memory and DRRPL displays the number of drivers relocated, the beginning and ending logical addresses, and the beginning and ending base page addresses as defined by the DRREL file. DRRPL then issues the prompt:

```
MEMORY OR PERMANENT REPLACE (ME,PE)?
```

You may install the drivers on the system either in memory or permanently on the disk. Your response to this prompt determines the format of the "REPLACE READY?" prompt that is issued to initiate the installation process.

If the driver has been relocated into the driver partitions, the following sequence of prompts is issued:

```
PHYSICAL PAGE OR DRIVER PART.  (PG,DP)?
```

A driver can be installed either in physical pages allocated for drivers at system generation or in pages unused by the system. If your response is DP to install the driver in the allocated pages, DRRPL issues the prompt:

```
DRIVER PARTITION NO.?
```

Enter the driver partition number. The partition numbers can be found in the generation listing or from the driver configuration listing output of the DD or MD commands if these were issued before you entered the replacement phase. It is advisable to use one of these commands to get the current configuration since DRRPL numbers driver partitions by finding EQTs that reference them. If, for example, all of the EQT entries originally pointing to a particular driver partition are pointed at other partitions, the driver partition is temporarily lost to the utility and can be recovered only by responding with PG to specify the starting page. However, by using the DP response, you can prevent accidental overlay of other drivers. Be sure that the partition contains sufficient pages to hold all of the drivers being installed, or an error will be returned.

If your response is PG, to install the driver or drivers in unused or allocated pages, DRRPL prompts with:

```
DP STARTING PHYSICAL PAGE?
```

The page specified must either be a page allocated for partition drivers at system generation or a page unused by the system. The FR command can be used to list the free pages in the system. Be sure that there are sufficient contiguous pages to hold the replacement drivers, or an error will be returned. Free pages are most often found in high physical memory, or pages may be reserved for this purpose by using the reconfigurator to declare bad pages (not within SAM extension). Be aware that free pages may not be used for permanent replacement.

After the driver location is specified, DRRPL displays all the EQTs that are currently referencing that area, together with the first driver listed in the input file (the DRREL relocation output):

```
DRIVER PARTITION 3 <<PAGE 35>>
 EQT 3 = SC 23 TYPE 23  D B T=  9999 X=    0 IN=  4701  CC= 5560
 EQT 7 = SC 12 TYPE 12    B T=   200 X=    0 IN=  4000  CC= 4215

DVA12
IA12   4000
CA12   4215
```

Be aware that, on replacing a driver, DRRPL does not update the system library to show the addresses of its entry points. DRRPL then prompts with:

```
EQT NUMBER?
```

Enter the EQT number to be modified to reference the listed driver. Select the EQT number from the display list. At least one EQT must be changed for every driver in the input file. Note that any EQT using the replaced driver must be redefined here. The information required is the same as the EQT entry in the system generation.

If the listed driver has multiple initiation/continuation entry-point pairs, DRRPL prompts with:

```
EQ TYPE CODE?
```

Enter the equipment type of the entry points to be referenced by the EQT. The following prompts require a Y or N response:

```
DMA (Y,N)?

AUTOMATIC OUTPUT BUFFERING (Y,N)?
```

DRRPL then requests the select code referenced by the EQT and the timeout interval with:

```
SELECT CODE?

TIMEOUT INTERVAL?
```

If the driver is to be installed in the system driver area, DRRPL prompts with:

```
OWN MAPPING (Y,N)?
```

and then issues the prompt:

```
EQT NUMBER (/E TO END)?
```

to allow you to specify the modification sequence for any additional EQT entries that must reference the given driver. If you have made an error in any of the preceding entries, you can repeat the full sequence for the EQT at this time by entering the EQT number again.

If more than one driver was relocated by DRREL, the next driver and entry points are displayed. The EQT modification prompt sequence is repeated for all relocated drivers.

The information given during system generation for any EQT redefined above must be redefined in this section (that is, respecify what you did for the interrupt table entry in the generation).

After the EQTs have been modified for all the drivers in the input file (/E response to the EQT NUMBER? prompt), DRRPL displays the default interrupt table changes, as:

```
INTERRUPT TABLE CHANGES:
S.C. INTERRUPT TABLE   TRAP CELL (MEMORY)
 12     EQT 7                  JSB 1601,I
 23     EQT 3                  JSB 1601,I
```

The interrupt table entry corresponding to the select code for each modified EQT will, by default, contain the address of that EQT. The trap cells associated with the interrupt table entries will contain a JSB LINK,I where LINK contains the address of $CIC. If the system uses OS microcode, a call to the equivalent microcode routine is placed in the trap cell. DRRPL then issues the prompt:

```
SELECT CODE (/E TO END)?
```

to allow you to modify the interrupt table/trap cells. If you enter a select code, DRRPL responds with one of the following prompts:

```
ENTRY TYPE (EQT,PRG)?              (Partition drivers)

ENTRY TYPE (EQT,PRG,ENT,ABS)?      (System Driver Area)

ENTRY TYPE (ENT,ABS)?              (Select code 4)
```

The action taken by DRRPL for each of the entry type responses is shown in Table 10-3 below.

Table 10-3. DRRPL Entry Type/Action

Entry Type	DRRPL Prompt	Action
EQT	EQT NUMBER?	The address of the given EQT is placed in the interrupt table, and a JSB LINK,I is placed in the trap cell. If the system uses OS microcode, a call to the equivalent microcode routine is placed in the trap cell. Be aware that only modified EQT entries can be specified in response to this prompt.
PRG	NAME	The negated ID segment address is placed in the interrupt table. A JSB LINK,I or OS microcode call is placed in the trap cell.
ENT	NAME	The entry point name must be contained in one of the replacement drivers. Zero is placed in the interrupt table and a JSB,LINK,I is placed in the trap cell. In this case, the LINK must pre-exist or be explicitly placed on base page. This can be accomplished in the driver by the following sequence of instructions: ORB DEF ENT ORR DRRPL will search the allocated base page links and the existing base page links for a link to the given ENT.
ABS	VALUE	Zero is placed in the interrupt table, and the given value is placed in the trap cell.

DRRPL continues to prompt for the SELECT CODE modification sequence until you enter /E to end. If you make an error in any of the select code modifications, the process can be repeated for the code in error.

Driver Replacement

After all system modifications have been made (/E response to the SELECT CODE? prompt), DRRPL displays a summary of the changes to be made, as:

```
SUMMARY OF SYSTEM CHANGES:
*********************************************************
DRIVER PARTITION n <<PAGE nn>>
LOGICAL ADDRESS nnnn nnnn
   NO BASE PAGE
      (or)
   BASE PAGE

EQT CHANGES:                                          (Changes)

INTERRUPT TABLE CHANGES:                              (Changes)
```

If the driver is to be installed in memory, DRRPL issues the following warning and prompts:

```
THIS CAN CRASH YOUR SYSTEM! THE SYSTEM SHOULD BE
INACTIVE.  EQTS TO BE REPLACED MUST NOT HAVE
REQUESTS PENDING.

MEMORY REPLACE READY (Y,N)?
```

If your response is Y, DRRPL turns off the interrupt system and checks to be sure that requests are not pending on any EQT to be replaced. If requests are pending, DRRPL turns on the interrupt system, issues the error message:

```
EQTS BUSY--NO REPLACEMENT DONE
```

and reprompts for the OK to replace.

If the existing driver cannot process the queued requests, you can OF the programs making the requests or you can use the DN command to down the LUs that reference the driver and temporarily redirect these LUs to the bit bucket (System LU command).

When all requests are cleared, DRRPL installs the new drivers and associated I/O tables and exits.

If the driver is to be installed permanently on disk, DRRPL issues the following warning and prompts:

```
THIS CAN DESTROY YOUR SYSTEM! MAKE SURE THE SYSTEM IS
BACKED UP! DOUBLE CHECK YOUR CHANGES TO THE SYSTEM
BEFORE PROCEEDING.  TURN OFF DISK WRITE PROTECT.

PERMANENT REPLACE READY (Y,N)?
```

Since DRRPL checks only to see that the write-protect switch is off, you should make a practice of backing up the system before turning off the write protect.

Driver Replacement Example

In the following example, two drivers relocated by DRREL (DRRPL input file DP3) are installed in memory in the driver partition. The example is limited to the driver replacement phase; refer to the related illustrations for the configuration command output formats.

```
/DRRPL:  DR,DP3

     2  DRIVERS
LOGICAL ADDRESS   4000  5614
   NO BASE PAGE

MEMORY OR PERMANENT REPLACE (ME,PE)? ME

PHYSICAL PAGE OR DRIVER PART.  (PG,DP)? DP

DRIVER PARTITION NO.?  3

DRIVER PARTITION   3   <<PAGE   35>>

  EQT   3 = SC 23 TYPE 23  D B   T = 9999  X = 0 IN = 4701 CC = 5560
  EQT   7 = SC 12 TYPE 12    B   T =  200  X = 0 IN = 4000 CC = 4215

WARNING:  MAY BE OVERLAYING DRIVERS.  TYPE '/A' TO ABORT

DVA12
IA12    4000
CA12    4215

EQT NUMBER?  7

DMA (Y,N)? NO

AUTOMATIC OUTPUT BUFFERING (Y,N)?  YES

SELECT CODE? 12B

TIME OUT INTERVAL? 200

EQT NUMBER (/E TO END)?  /E

DVR23
I.23    4701
C.23    5560

EQT NUMBER?  3

DMA (Y,N)? Y

AUTOMATIC OUTPUT BUFFERING (Y,N)?  Y

SELECT CODE? 23B

TIME OUT INTERVAL? 9999

EQT NUMBER (/E TO END)?  /E

INTERRUPT TABLE CHANGES:
 S.C.  INTERRUPT TABLE   TRAP CELL  (MEMORY)
  12        EQT   7      JSB 1601,I
  23        EQT   3      JSB 1601,I
```

```
INTERRUPT TABLE MODIFY:

SELECT CODE (/E TO END)? 24B

ENTRY TYPE (EQT,PRG)? EQT

EQT NUMBER?  3

SELECT CODE (/E TO END)? /E


SUMMARY OF SYSTEM CHANGES:
**********************************************************************
DRIVER PARTITION   3   <<PAGE   35>>
 LOGICAL ADDRESS   4000  5614
    NO BASE PAGE

EQT CHANGES:
 EQT   7 = SC 12 TYPE 12    B   T =  200  X = 0 IN = 4000 CC = 4215
 EQT   3 = SC 23 TYPE 23  D B   T = 9999  X = 0 IN = 4701 CC = 5560

INTERRUPT TABLE CHANGES:
 S.C.  INTERRUPT TABLE   TRAP CELL  (MEMORY)
  12         EQT   7     JSB 1601,1
  23         EQT   3     JSB 1601,1
  24         EQT   3     JSB 1601,1
**********************************************************************

**************************************************************
**************************************************************

          WARNING!    WARNING!    WARNING!    WARNING!

        THIS CAN CRASH YOUR SYSTEM!  THE SYSTEM SHOULD BE
        INACTIVE.  EQTS TO BE REPLACED MUST NOT HAVE
        REQUESTS PENDING.

**************************************************************
**************************************************************

MEMORY REPLACE READY (Y,N)? _Y
```

DRRPL Error Messages

The following error messages can be issued by DRRPL. Fatal errors that will cause DRRPL to abort are flagged with (F) following the error definition. Note that all errors are fatal when the utility is running non-interactively.

DRRP 001 INPUT ERROR

An invalid command or response was given to a prompt.

DRRP 002 OUT OF RANGE

A non-existing EQT number, driver partition number, physical page number, or select code was given. An invalid timeout interval was given; the range is 0 through 32767 (decimal). Check the system generation listing for the correct response.

DRRP 003 NOT FOUND

An entry point or program was not found. This may be caused by a bad equipment type code or an entry point not in one of the replacement drivers.

DRRP 004 BAD SELECT CODE

A select code may not be between 5 and 7, may not be the select code of the privileged interrupt card, may not be less than the privileged interrupt card for the partition driver, and may not be 4 for partition drivers.

DRRP 005 NOT AVAILABLE

A physical page requested as part of a driver partition is being used by the system. Pages may be reserved by declaring bad pages with the reconfigurator.

DRRP 006 BAD EQT NUMBER

An EQT number that was used with a previous driver was given. An Interrupt Table entry can only be changed to reference one of the modified EQTs. Enter a valid EQT number.

DRRP 007 PARTITION TOO SMALL

The replacement drivers are too big to fit in the requested driver partition without overlaying part of another. Override this check by specifying the driver partition starting physical page. This error may also occur if a requested driver partition will overlay pages used by the system.

DRRP 008 LINK NOT FOUND

No link to the given entry point was found on base page in either the existing links or in the replacement links. You must provide a base page link to the entry point.

DRRP 011 INVALID FILE

An incorrect file name was given, the file is corrupt, or it was created by the wrong version of DRREL. Give the correct file name or recreate the file with DRREL. (F)

DRRP 012 FILE CREATED ON ANOTHER SYSTEM

The input file given was created by DRREL running on a different system. Drivers must be relocated on the same system in which they will be installed. Re-run DRREL on this system.

DRRP 013 TABLE OVERFLOW

DRRPL uses the space between the end of the program and the end of its logical address space to store the input file and to build new I/O tables to be installed in the system. Use the SZ command to increase the space for DRRPL or reload DRRPL as a large background program. (F)

DRRP 014 EQTS BUSY--NO REPLACEMENT DONE

DRRPL will not replace a driver in memory until all pending requests on EQTs to be replaced are cleared. If the existing driver cannot process the requests, you can OFF the programs making the requests, down those LUs that reference the EQT using the system DN command, and temporarily redirect the LUs to the bit bucket with the system LU command. If this does not correct the problem, you may have to reboot the system.

DRRP 015 CORRUPT SYSTEM

There is no link to $CIC on base page, or the replacement driver links will overlay this link. Check your generation listing and specify different base page addresses for the driver. (F)

DRRP 016 NOT SYSTEM MANAGER

An attempt was made to use the DR command running under an account other than the system manager or running non-session from a terminal other than the system console. Log on correctly and call DRPL as RU,DRRPL:IH.

DRRP 017 NOT 'DRRPL'

An attempt was made to use the DR command with a copy of the program. Run the program with the command RU,DRRPL:IH or reload the program with the LOADR command OP,DC.

DRRP 018 SYSTEM DISK WRITE ERROR

An error occurred on an attempt to do a permanent replacement. This error is usually caused because the disk is write protected. Turn off the write protect and try again.

DRRP 019 PROGRAM CANNOT BE EXTENDED BACKGROUND

An attempt was made to use the DR command running DRRPL as an extended background program. Reload DRRPL so that it is not extended background.

11

File Analysis Utilities

File analysis utilities FLAG and EXT work together to identify external references and specific character patterns in a source file. FLAG can use the HP-supplied patterns file, SEP.6, which contains a list of all FMP, EXEC, and system library routines that can be called. EXT can be used to customize or create a patterns file for FLAG.

SEP.6 can be easily edited to include entry point names from other system libraries and subsystems or specific declarations and variables occurring in a source file.

You can also create application-specific patterns files. The words in the patterns file may contain any printable characters to a maximum of 16, with one word to a line. Refer to the EXT discussion for the file format. (Note that unsorted patterns files are compared more efficiently than sorted files.) Comments can be included in the patterns file by entering an asterisk (*) in the first column of the line.

Pattern Matching Utility (FLAG)

FLAG is a pattern-matching utility that searches one or more source files for words listed in the file SEP.6 (or a custom patterns file) and flags all matches. The matches found are listed to your terminal or to an output file if you specify the -O option in the command line.

Calling FLAG

To call FLAG, enter the following runstring:

```
:[RU,]FLAG,pfile[,-options],sfile[,...]
```

The pfile is the name of the patterns file, either SEP.6 or your custom patterns file.

The sfile is the source file name or names to be searched for matches.

Any combination of the options, which are discussed in the next section, can be specified.

FLAG Options

The options available with FLAG are shown in Table 11-1.

All options or option groups must be preceded with a dash. The first argument without a leading dash is interpreted as the patterns file name, and all following arguments without a leading dash are interpreted as the source file names. The options are order independent and can be placed anywhere in the command line. However, the language option must precede the associated source file or files. As an example, the command line

```
FLAG,-KV,SEP.6,-OAFLG,-P,&MUTWN::MR,&RUTWN::MR,-F,&STING::JT
```

specifies that FLAG should ignore comments in the Pascal source files &MUTWN::MR and &RUTWN::MR and the FORTRAN source file &STING::JT. Options −KV specify that case is significant in matching a pattern (K) and that the output is to be all lines of the file with matches flagged (V). The −O option names AFLG as the output file. The output file option O could also be grouped with the other named options, as the last member of the group (for example, −KVOAFLG,).

Table 11-1. FLAG Options Summary

Option	Description
C	For each word matched, print the count of lines that contain a match.
K	Make case significant in determining a match. By default, case is not significant (in other words, a = A or a).
M	Print the source file name at the beginning of each output line. This is the default when more than one source file is specified.
N	Print the line number of each matched line at the beginning of the output line.
V	Verbose mode. Print all lines in the file with line numbers. Flag matches in lines and print a summary of matches for each pattern. (The summary is identical to that given with the C option.)
PBFA	Language options. (Pascal, BASIC, FORTRAN, Assembler.) All source files specified following a language option are assumed to contain code in that language, until the next language option is specified. The purpose of this option is to cause FLAG to ignore all material in the comment fields. FLAG does not verify the actual language of the source; it only checks for the comment character associated with the language specified by the option. If no language option is specified, FLAG searches all material in the source file.
Ofilename	Specifies file as the output file, instead of your terminal. The file name must immediately follow the O option letter; a space may not be used. If the options are grouped, the O option must be the last member of the group.

FLAG Examples

The four examples below illustrate how FLAG and its options are used.

Example 1:

FLAG is called to run in Verbose mode (−V) and to disregard all comment lines in the FORTRAN file &STING::JT (−F). The Verbose mode output is a listing of all the lines of the file to the output file AFLG (−O option) with each matched pattern flagged.

```
5>>         vvvvv
 >>   LU = LOGLU(Z)
 >>
```

A summary count of the matched patterns follows the program file listing.

```
FLAG,-VFOAFLG,SEP.6,&STING::JT
    FLAG/1000              8:56 PM MON., 15 AUG., 1988
    Version 2.2            &STING::JT
          1       PROGRAM STING
          2       IMPLICIT INTEGER (A-Z)
          3       DIMENSION BUFR1  (40)
          4       DIMENSION BUFR2  (40)
          5>>           vvvvv
           >>     LU = LOGLU(Z)
           >>
          6  C   GET THE RUN STRING
          7>>           vvvvv
           >>     CALL EXEC (14,1,BUFR1,40)
           >>
          8       WRITE (LU,1) BUFR1
          9  100  FORMAT ("EXEC14= "40A2)
         10  C   GET THE PARAMETER STRING WITH GETST
         11>>           vvvvv
           >>     CALL GETST (BUFR2,40 TLOG)
           >>
         12       WRITE (LU,2) BUFR2
         13           vvvvv
           >>200  FORMAT("GETST= ",40A2)
           >>
         14       END

Flag/1000 Summary>>4 words flagged in &STING::JT

    EXEC>>1
    GETST>>2
    LOGLU>>1
```

Example 2:

FLAG is called using only the −F language option. The output, defaulted to the terminal, is a summary of the patterns flagged, together with the line of code in which they were found. The comment line reference to GETST is ignored (see the full program listing in the first example).

```
    FLAG,SEP.6,-F,&STING::JT
       LOGLU>>         LU = LOGLU(Z)
       EXEC>>          CALL EXEC (14,1,BUFR1,40)
       GETST>>         CALL GETST (BUFR2,40 TLOG)
       GETST>>200      FORMAT(”GETST= ”,40A2)
```

Example 3:

FLAG is called with the −N option, but without the language option. The output, defaulted to the terminal, is a list of the patterns flagged (including the comment line reference to GETST), the related line of code, and the code line number.

```
FLAG,SEP.6,-N,&STING::JT
   5>LOGLU>>          LU = LOGLU(Z)
   7>EXEC>>           CALL EXCEC (14,1,BUFR1,40)
   10>GETST>>C        GET THE PARAMETER STRING WITH GETST
   11>GETST>>         CALL GETST (BUFR2,40 TLOG)
   13>GETST>>200      FORMAT(”GETST= ”,40A2)
```

Example 4:

FLAG is called with the −F language option to search two source files for matches. Because two source files are specified, the −M (print source file name) is automatically invoked. In this example, the file &NULL::JT does not match any entries in the patterns file SEP.6, as noted in the output.

```
FLAG,-F,SEP.6,&STING::JT,&NULL::JT
   &STING::JT>LOGLU>>         LU = LOGLU(Z)
   &STING::JT>EXEC>>          CALL EXEC (14,1,BUFR1,40)
   &STING::JT>GETST>>         CALL GETST (BUFR2,40 TLOG)
   &STING::JT>GETST>>200      FORMAT(”GETST= ”,40A2)

Flag>>no words matched in &NULL::JT
```

Loading FLAG

Flag is loaded with the following load command sequence:

```
OP,LB
LI,=PLIB
RE,=FLAG
EN
```

Because FLAG uses EMA, it must run as large background in a mother partition. If it is not sized as part of the load command sequence or assigned to a specific partition before running, it will run in the largest available mother partition.

File External References Utility (EXT)

EXT searches a relocatable input file, finds external references, and lists them to your terminal or to an output file if you specified one in the command line. You can use the BREAK key on your terminal to halt the output listing at any time.

Calling EXT

To call EXT, enter the following runstring:

```
:EXT[,-options],srcfile[,outfile]
```

The srcfile is the relocatable (type 5) file to be searched for external references.

The outfile is the file to accept the external references found by EXT. If this file exists, the externals are appended to it unless the replace (−R) option is specified. If the file does not exist, EXT creates it.

You can specify any of the EXT options, which are discussed below.

EXT Options

The options available for use with EXT utility are shown in Table 11-2.

If neither a patterns file nor an ignore file is named, EXT prints all externals in the source file. If both files are specified, EXT prints all externals in the source file that are named in the patterns file, less the externals in the ignore file.

Options are specified with a leading dash and can be placed anywhere in the command line. EXT interprets the first argument it encounters without a leading dash as the source file name; the second, if specified, is interpreted as the output file name. The options can be specified singly or in a group, except that any option requiring a file name must be specified last in the group. The file name must immediately follow the E, F, or I option letter; a space may not be used. For example, either of the two following command lines can be used to run EXT in verbose mode on the source file $BHLIB, scrolling the output 11 lines at a time to your terminal screen and finding patterns from the file IMAGES::JT:

```
EXT,-VS11FIMAGES::JT,$BHLIB

EXT,$BHLIB,-V,-S11,-FIMAGES::JT
```

Table 11-2. EXT Options Summary

Option	Description
C	Condense the output list: separate the pattern words with spaces instead of putting them on separate lines. This option is useful when the output is to your terminal.
Lnnn	Lengthen the output line to nnn characters. The default and minimum is 80, the maximum is 134. This option is useful when the output is to your terminal and forces the C option.
Snn	Scroll the output nn lines at a time. The default is 22. Use −S0 for continuous printing without prompting. This option is useful when the output is to your terminal.
N	Include the name of the routine (the nam record) in which the externals are found.
T	Identify entry point and external names.
V	Verbose mode. Provide all of the information given with the C, N, and T options.
R	Replace (overwrite) the output file if it exists, instead of appending to it.
Efile	Send error messages to the named file. If the file already exists, the error messages are appended to it; if it does not exist, EXT creates it. If this option is omitted, error messages are displayed on your terminal.
Ffile	In searching srcfile, find only the externals in the named patterns file.
Ifile	In searching srcfile, ignore the externals in the named file. This file should have the same structure as the patterns file.

Output Formats

If you are using EXT to create or modify a patterns file for FLAG, run the utility with no output format options (that is, do not specify the C, N, T, or V options).

No Options

Running EXT with no format options produces the following output. Note that the output is not sorted; this is the most efficient form for comparing files for matching (using FLAG).

```
EXT,$BHLIB
   .MPY
   .DIV
   .ENTR
   CBYTE
   LBYTE
   MIN0
   .ENTR
   .MBT
   .SBT
   CRETS
   CLOSE
     :
   WRITF
   LOCF
   PURGE
   RUN
   .ENTR
   IDRPD
```

−C Option

The −C option output is the unsorted listing in condensed form, where information is separated with spaces rather than appearing on separate lines.

```
EXT,-C,$BHLIB
 .MPY     .DIV    .ENTR    CBYTE    LBYTE    MIN0
 .ENTR    .MBT    .SBT     CRETS    CLOSE    WRITF    LOCF    PURGE    RUN

 .ENTR    IDRPD   RP       NAMR     EXEC     IDGET    RMPAR
 .DIV     .ENTR   .GOTO    BLT      FSTAT    RDDIR    LBYTE
 .MPY     .ENTR   EXEC     ISHFT    CLUCR    NAMRT
 .DIV     .ENTR   SBYTE    WTREC
   :
 LOGLU
 .ENTR    EXEC    OPEN     CREAT    CLOSE    WRITF    LOCF     FMGR   .MVW
 LOGLU
 EXEC     .LBT    .SBT
 NAMR     .ENTR   .MVW
 .ENTR    .LBT
 .ENTR    .SBT
```

−T Option

The −T option output is the unsorted listing with each external and entry point identified.

```
ext,$bhlib,-t
  ent: KYWDS
  ent: .MPY
  ext: .DIV
  ext: .ENTR
  ext: CBYTE
  ext: LBYTE
  ext: MIN0
  ext: FMGR
  ext: .ENTR
  ext: .MBT
  ext: .SBT
         :
  ext: LUTRU
  ext: RDACT
  ext: MOD
  ext: ISHFT
  ext: MBYTE
  ext: SBYTE
```

−N Option

The −N option identifies the routine in which each external/entry point is found. The heading is the nam record, which gives the routine type and priority (n,nn), its size, and other information that can vary from routine to routine.

```
EXT,-N,$BHLIB
    KYWDS (7,99) 2024 WHH Keyword table searcher
     149 words
     .MPY
     .DIV
     .ENTR
     CBYTE
     LBYTE
     MIN0

    FMGR (7,99) 2023 WHH FMGR runner
     346 words
     .ENTR
     .MBT
     .SBT
     CRETS
     CLOSE
     WRITF
     LOCF
     PURGE
     RUN

    WILD (7,99) 2023 WHH Wildcard evaluator)
     913 words
     .DIV
     .ENTR
     .GOTO
     BLT
     FSTAT
     RDDIR
     LBYTE
     :
    GETGU (7,99) 2023 WHH Get group and user ID
     26 words
     $ESTB
     .ENTR

    TYPER (7,99) 2017 WHH Text typer
     235 words
     EXEC
     .ENTR
     REIO
```

−V Option

The −V option provides all of the information included in the −N option and the −T option for each routine, using the −C option format.

```
EXT,$BHLIB,-V
    KYWDS (7,99) 2024 WHH Keyword table searcher
       149 words
       ent: KYWDS
       ext: .MPY    .DIV    .ENTR     CBYTE      LBYTE      MIN0

    RUN (7,99) 2023 WHH RUN command processor
       170 words
       ent: RUN
       ext: .ENTR   IDRPD    RP       NAMR       EXEC       IDGET   RMPAR

    WILD (7,99) 2023 WHH Wildcard evaluator
       913 words
       ent: WILD
       ext: .DIV    .ENTR   .GOTO     BLT        FSTAT      RDDIR   LBYTE


    STDIO (7,99) 2023 WHH Standard I/O initialization
       81 words
       ent: STDIO   EXIT
       ext: .ENTR   EXEC    STDIN     STDOU      NAMRN      PRTN
         :

    RDACT (7,99) 2023 WHH Account file reader
       370 words
       ent: RDACT
       ext:.ENTR   OPEN    READF

    NAMRT (7,99) 2023 WHH NAMR typing routine
       20 words
       ent: NAMRT
       ext: .ENTR

    NAMRN (7,99) 2023 WHH Get next NAMR parameter
       105 words
       ent: NAMRN
       ext: .ENTR   BLT     NXPAR     SBYTE      NAMRU
```

Loading EXT

Load EXT with the following load command sequence:

```
OP,LB
SZ,28
LI,=PLIB
LI,$VMCLB                (* Use this command for RTE−IVB systems only)
RE,=EXT
RE,=AVL2
EN
```

If the SZ,28 command does not load EXT, try again with a smaller size.

Error Handling

EXT normally returns the value 0 in the FMGR 1P global. If an error occurs, EXT reports the non-zero value returned in 1P with an error message. A negative value indicates the corresponding FMP error. A positive value is returned with one of the following error messages:

```
Ext: (1) Didn't recognize X,X options.

Ext: (2) Wrong file type for input <output><error><find><ignore>file NNN.

Ext: (3) Size EXT up, or use fewer entries in your find or ignore files.
```

The last message will appear only if EXT was loaded incorrectly, and it indicates an internal table overflow. In all cases, EXT exits after issuing the message.

12

Help Lookup Utilities

This chapter includes information on the HELP, CMD, and GENIX help utilities; and the CALLS and CALLM help lookup utilities.

Help Utilities (HELP, CMD, GENIX)

The help utility group of HELP, CMD, and GENIX supplies descriptive information and corrective action for error conditions, and provides a means for you to generate any desired text messages for display at your terminal or any selected device. Using the HELP, CMD, and GENIX utilities, you can display messages defining any system or FMGR error code.

HELP passes an error code or a key to CMD and schedules that utility to retrieve the associated message. CMD searches for the file CMD.HLP::SYSTEM first, and if this is not found, it searches for the file !CMD::0. CMD then searches the appropriate file for the key and displays the associated message on your terminal or the device specified in the runstring. CMD also can be called directly to display a message retrieved from the disk-resident help files or from a specific message file named in the runstring. GENIX allows you to create files of keyed messages for display through calls to CMD. HELP requires at least a 10-page partition; CMD requires 25 pages, and GENIX needs at least 25 pages (GENIX may be sized up to accommodate more keywords).

HELP

The HELP utility provides a detailed explanation of an error code or message key. If no code/key is included in the command runstring, HELP retrieves the last error code generated. HELP then schedules the CMD utility and passes the code/key to CMD, together with the name of the help file (/SYSTEM/HELP.HLP or !HELP::0) to be searched. CMD searches the files to find an exact match for the error code/key and lists the associated text. HELP terminates when CMD completes the list function. The HELP.HLP file can be created using GENIX with the file “HELP as the input file name.

Calling HELP

To call HELP, enter one of the following runstrings:

```
:HE[,key[,listfile]]                    (from FMGR)
```

or

```
CI.74> HELP[,key[,listfile]]            (from CI)
```

The key is the word or phrase to be searched for in the indexed message file. If no key is given, the default is to the last error posted in the Session Control Block (SCB). If no error was posted, HELP outputs the message

```
NO ERROR IN SCB
```

and exits.

The listfile lu is the file or LU to receive the output. The default is the log terminal.

The key word or phrase can be from 1 to 24 characters long, and leading blanks are ignored. If, however, the key is a single character, it is considered unmatchable and HELP lists ten valid keys alphabetically surrounding the runstring character. If a key cannot be uniquely matched, a

```
KEY NOT UNIQUE
```

message is listed, together with the valid keys (to a maximum of ten) that match, in part, the runstring key. If the key cannot be matched at all, a

```
KEY NOT FOUND
```

message is listed, together with ten valid keys alphabetically surrounding the runstring key. In all cases, HELP exits after the unsuccessful match attempt.

Examples

Example 1:

```
:HE,R                                   (Single-character entry)
                                        (LIST OF KEYS)

OF
PACK
PR
QU
READ02
READ03
READ04
REV
RN
RP
:                                       (HELP exits)
```

Example 2:

```
:he,axx                                 (Invalid key entry)

KEY NOT FOUND 'AXX'
LIST OF KEYS

AC
ACI
:
BL
BR
:                                       (HELP exits)
```

Example 3:

```
:he,read                                (Nondefinitive entry)

KEY NOT UNIQUE 'READ'
LIST OF KEYS

READ 001
READ 002
 :
READ 008
:                                       (HELP exits)
```

When the search is successful, HELP echoes the key word/phrase, then prints the associated text and exits, as shown in the next two examples.

Example 4:

```
:help,re

RE

A RE-ENTRANT SUBROUTINE ATTEMPTED
TO CALL ITSELF.
:
```

Example 5:

```
:HE,FMGR-006

FMGR-006

FILE NOT FOUND
AN ATTEMPT WAS MADE TO ACCESS A FILE THAT CANNOT BE FOUND.
CHECK THE FILE NAME OR THE CARTRIDGE REFERENCE.
:
```

HELP Error Messages

HELP generates the following error messages. (Note that error messages also may be generated by CMD running under HELP. Refer to the CMD utility for a description of CMD error messages.)

NO ERROR IN SCB; HELP KEYWORD MUST BE GIVEN.

No key was given and the SCB has not had an error posted.

HELP KEYWORD MUST BE GIVEN WHEN NOT IN SESSION.

ERROR SCHEDULING CMD, CANNOT GIVE HELP.

CMD

The CMD utility provides the means for retrieving and listing messages for any interactive program on RTE-6/VM. Messages are generated by GENIX as indexed key files; CMD searches the file index for the matching key and lists the related message on your terminal or any LU designated in the runstring. Any message—expanded error description, command syntax, text—can be generated for retrieval. CMD can be called directly or scheduled by HELP to run under that utility.

CMD supports an interactive mode that lets you examine all the keys in a file and call for additional information about a key if the message was configured in multiple blocks.

Calling CMD

To call CMD, enter the following runstring:

```
:[RU,]CMD[,key[,listfile[,helpfile[,NI/SK]]]]
```

where:

key	is the word or phrase to be searched for in the index file. If no key is given, CMD responds with a message describing how to use the utility.

Note If CMD is run from CI, you must enclose the phrase in quotes if embedded blanks in the phase are to be passed to the CMD utility. Refer to the *RTE-6/VM CI User's Manual* for more details.

---

listfile	is the file or LU to receive the output. The default is to your terminal.
helpfile	is the name of the message file to be searched. The file must be resident on disk and must be created by GENIX. If a file is not specified, the default is the file CMD.HLP::SYSTEM or !CMD::0.
NI	directs CMD to run in non-interactive mode. The default is to run in interactive mode.
SK	specifies single key mode: display 1 key with terminal paging ("More..." prompts) and exit.

A key can be from 1 to 24 characters in length and can contain any ASCII character except the comma (,) which is used as a command delimiter. If a key is only one character long, it is considered unmatchable. Leading blanks are ignored; however, they are considered in calculating the 24-character line length. When calling CMD interactively, trailing blanks can be entered directly using the keyboard space bar.

In non-interactive mode, all user prompts are suppressed, and CMD exits after completing the list function associated with the runstring key word or phrase. If the key is matched, the entire message text is output; if the key is not matched or is not unique, the ten keys alphabetically surrounding it are output.

In interactive mode, if a key is not matched or is not unique, the prompt

```
More...
```

is issued following the listing of the first ten alphabetically surrounding keys. Entering a space lists another page of keys; entering "A" aborts the key listing and returns to the "key:" prompt.

You may then enter another key or exit CMD.

When a key is matched and more text is available for that key, the More... prompt is issued following the message if the information is formatted in multiple text blocks. (For example, a command syntax message could be in multiple blocks: the first block could contain only the command runstring parameters; a second message block could contain definitions of the parameters and would only be output in response to the More... prompt.)

In creating the message file, GENIX lets you use the :TR function to specify a transfer to another file to retrieve all or a portion of a message. When CMD encounters the :TR transfer keyword in the message, the new file name is displayed on your terminal with the OK to transfer prompt, as

```
:TRANSFER to MSG1...('a' to abort)
```

This gives you the option of accepting or rejecting the transfer. When a :TR is encountered and you enter the <space> response, the current file is closed and CMD transfers to the new file.

Transfer commands can be chained; that is, the message in the transfer file can itself contain a :TR command to another file and key that could also contain a :TR command. In each case, CMD closes the current file and transfers to the new file. When the last block of the message is displayed, CMD remains in the last open file and prompts for another key. The proper response here is <cr> to exit, since CMD is no longer in the original indexed file.

If a key matches two or more keys that point to different text blocks, a

```
KEY NOT UNIQUE
```

message is output, together with ten valid keys alphabetically surrounding the runstring key. If, however, the runstring key matches two or more keys that point to the same text block, it is considered a match and the text block is output. (Refer to the discussion of GENIX for a description of the multiple key referencing scheme.)

If a key cannot be matched at all, the

```
KEY NOT FOUND
```

message is output, together with the ten valid keys alphabetically surrounding the runstring key.

CMD Examples

Example 1:

```
        :CMD,LIST
        KEY NOT FOUND 'LIST'                    (Invalid key entry)
        LIST OF KEYS

        HE    HELP FCN
        IF    CONDITIONAL
        IN    INIT CARTR
        IT    INTERVAL TIME
        LGTT  SYSTEM TRACKS
        LI    LIST FILE
        LL    SET LIST DEV
        LO    SET LOG DEV
        LOADR LOADER
        LU    SET SYSTEM LU

    More... a

    key: LI                                     (New key entered)

    LI   LIST FILE    FMG          [10]

    LI,namr [,S [,n1 [n2 ]]                     (Text message)
                     ,B
                     ,D

    List file namr to list device               (which can be set with LL command)

    More... a

    key: <cr>                                   (<cr> exits)

    :
```

Example 2:

```
        CI.74>cmd,read,,help.hlp        (Nondefinitive key entry)

        KEY NOT UNIQUE 'READ'
        LIST OF KEYS

        READ 013
        READ 014
        READ 015
        READ 016
        READ 017
        READ 018
        READ 019
        READ 020
        READ 021
        REV

        More... <space>                 (List continuation command)
```

```
RN00
RN01
RN02
RN03
RQ
SC00
SC01
SC02
SC03
SC04

More... a

key: <cr>                                   (Exit)
CI.74>
```

Example 3:

```
CI.74>cmd,‘fmgr 000’,,help.hlp              (Search file specified)
                                            (Backward quotes must be used to
     FMGR 000                               delimit a key with a space in it.)

BREAK
THIS IS AN INFORMATIVE MESSAGE ONLY.  NO ERROR HAS OCCURRED.

     key:  <cr>                             (No continuation block)
     CI.74>
```

Example 4:

```
CI.74>cmd                                   (No key entry)
     NO KEY

     RU,CMD [,key [,listfile [,helpfile [,NI/SK]]]]
                                            (How-to-use-CMD information)
     CMD                                    (A general-purpose help facility
                                            program (this program!)

     More... a

     key: <cr>                              (Exit)
     CI.74>
```

CMD Error Message

CMD can generate the following error message:

Helpfile is corrupt; must be created by GENIX.

The input file may not have been indexed by GENIX, or may have been corrupted by backup from magnetic tape not in binary mode.

GENIX

The GENIX utility allows you to create keyed text files for display through calls to the CMD utility. There are no restrictions on the size or the content of your text; however, each text block must be formatted as described below. For each input text file, GENIX constructs an indexed type 1 file that can be accessed by CMD.

Calling GENIX

To call GENIX, enter the following runstring:

```
:[RU,]GENIX,infile[,listfile],outfile
```

The infile is the input file of text and keys to be indexed.

The listfile is the list file or LU to receive the GENIX messages generated during processing of the file. The default is the log device.

The outfile is the output file name. The output file is disk-resident and may not be an LU. Any size or type parameters in the outfile descriptor are ignored.

The type 1 output file is created in the correct size after GENIX scans the input file to determine the space required. During processing, the input file text is saved in a temporary type 4 scratch area until the output file is generated.

Input File Format

GENIX recognizes the following control characters entered in column 1 of the input; any other characters on the control line are ignored, except in the case of the transfer function (TR), where the record is searched for the transfer command.

`""` A double quote signifies that the following record is a key word or phrase .

`&` An ampersand signifies that the next block of text is a continuation of the preceding keyed message, and that the “More...” prompt is to be issued at this point, before the record is displayed. The “More...” prompt is in the following form:

```
More...
```

If the response is <space> for More, CMD displays the information following the & control character. There is no limit on the number of information blocks that can be created for a key word (however, more text than necessary is generally counterproductive).

```
:TR,name,key
```

A colon TR command initiates a transfer to the named file and key. The command is given as

```
:TR [,name [,key]]
```

where:

name	is the name of the file to which CMD is to transfer. If the name is omitted, the current file is used for the search.
key	is the unique character string to be used in the search. If the key is omitted, the CMD utility NO KEY option is used. (Refer to the previous discussion of the CMD utility for examples of the NO KEY response.)

Transfer commands can be chained; that is, the message in the transfer file can itself contain a :TR command to another file and key that could also contain a :TR command. In each case, CMD closes the current file and transfers to the new file. When the last block of the message is displayed, CMD remains in the last open file.

A key can be from 1 to 24 characters long. Any character may be used except a comma (,), which is used as a command delimiter. All keys are converted by GENIX to uppercase. Leading blanks are ignored, but they are considered in the 24-character length restriction. For example, if the key "descriptors" is physically entered into the input file beginning at column 19, it is truncated to "descri" since the first 19 blank columns plus the key word exceed the 24-character limit.

Note that more than one key can reference the same message. In this case, the form of the entry is

```
""
key1
""
key2
Related text...
```

GENIX text format is entered on the line immediately following the key, in the exact form it is to appear; that is, uppercase and lowercase, paragraph separator lines, and so on. Text may include any terminal escape sequences except NULL or EOT, as these characters are used by CMD. Any line that does not begin with one of the GENIX control characters is considered to be text. Note, however, that unless the text block is preceded by a valid keyword entry, the entire block is ignored.

GENIX Example

```
    ""

    keya

This is the text for the first key.  Note that the key can be given in either
uppercase or lowercase, but not a combination of both.
    ""
    keyb
    ""
    keyc

This is the text for the second and third keys.  Either key will cause this
text to be displayed.
    &

The ampersand will cause CMD to issue the MORE prompt at this point when
either keyb or KEYC is entered in the CMD runstring.  The text is printed only
when the "/" response, requesting further information, is given.
    &

Any number of message blocks can be generated.
    ""
    LOAD

In this example the text message block is listed and a transfer is directed
below.
    :TR,!LOAD,LOADER

When the :TR command is encountered, CMD echoes the command on your terminal,
issues the OK TO TRANSFER prompt and, if the <space> response is given, closes
the current file and opens transfer file !LOAD to search for the key LOADER.
Note that the :TR commands can be chained; each time a :TR is encountered, the
current file is closed and CMD opens the named transfer file.
    ""

    bbbbbbbbbbbbbbbbbbbDESCRIPTORS

This key will be indexed as DESCRI since the physical key word line length
exceeds 24 characters.  Leading blanks are ignored in indexing a key, but are
considered in calculating the 24-character line length.
```

GENIX Error Messages

GENIX can generate the following error messages. Fatal errors that cause GENIX to abort are indicated with (F) following the error description.

HEAP/STACK COLLISION IN LINE xxxx

Too many keys were found to save in the Pascal heap area. Resize the program for a larger memory partition and rerun.

NO KEYWORDS FOUND

The input file contains no key words at all. At least one key word is required for a valid index. (F)

WARNING: BLANK KEYWORD FOUND

A block of text preceded with a blank key word was found. File processing continues and the warning message is inserted in the file listing.

DUPLICATE KEYWORD 'xxx...x'

A duplicate key word was found. Any text associated with the duplicated key word is ignored. File processing continues and the warning message is inserted in the listing.

CALLS and CALLM Utilities

The CALLS and CALLM utilities allow you to build and retrieve keyword indexed help information. The CALLS utility provides a general-purpose help facility, used either as a help subsystem for other programs or as the interface to a "database" of information grouped by keywords. The CALLM utility builds a single compressed input text file for the CALLS program.

CALLS Online Help Facility

The CALLS facility looks up keywords entered by the user in a catalog containing definitions of keywords and associated text and then displays that text. Additionally, the catalog can specify hierarchical groupings of keywords and can suggest related keywords that may be of further interest after the text for a certain keyword is viewed.

Invoking the CALLS Facility

To invoke the CALLS facility use the following runstring:

```
CI> calls [-flags] [keyword]
```

where:

−flags — is a string of characters preceded by a dash (−). Where an argument is required, the next word in the runstring is used, delimited by blanks or a comma. The flags are:

C catalog	The name of the CALLS catalog to use. By default, directory "/catalogs" and type extension ".call" are added to the given name. The default catalog is "/catalogs/calls.call".
L listfile	Divert the text listing to the named file. By default, the text is listed to the terminal.
P pagesize	Set the number of lines per page for "More..." prompting on the terminal. The default size is 22 lines.
B	Build the index file and terminate. See the discussion later in this chapter on index files.

keyword — is the keyword for which the associated text is to be listed. If not given, then the default keyword ("[default]") for the selected catalog is listed.

For example, "calls −c utils −p 5" and "calls −cp utils 5" both use catalog "/catalogs/utils.call" and five lines per page.

At certain times CALLS may ask you to select another topic to display as indicated below:

```
Put cursor on desired name or type new name, press return.
```

This occurs when no topic keyword is given in the runstring or when a mask is given. This also occurs when the topic selected has other topics associated with it, which you may also want to read.

When you press carriage return, CALLS will read the line under the cursor from the screen, isolate the word under or to the left of the cursor, and use that word as the new topic name. If there is no word to the left or under the cursor, CALLS will look to the right of the cursor. If there is no word on the line at all, CALLS will terminate. CALLS isolates the word by looking for blanks, commas, or right parentheses. To terminate CALLS, type carriage return on a blank line.

If an unknown keyword is given, CALLS will list the 16 keywords in alphabetical sequence around the given keyword and then go into interactive mode as above.

CALLS Catalog File

The CALLS catalog is a text file that acts as a database containing keywords and explanatory text. Catalog files may be compressed by the CALLM utility. The default catalog name is actually based on the name by which CALLS is scheduled (that is, the second word in the received runstring). If CALLS is RP'ed under a different name or the .RUN file is renamed, the new name becomes the default catalog name for that copy.

For example, if you issue "rp calls utils" and then execute UTILS, it will use the default file named "/catalogs/utils.call".

CALLS Directives

The CALLS catalog files may be plain text files in the format given below, but more commonly the final catalog is built by the CALLM utility, which merges together plain text files and performs text compression on the result. Additionally, CALLM can extract CALLS catalogs from comments in source code. (See the CALLM section later in this chapter or enter "? callm" from the CI prompt for more information about the CALLM utility.)

A catalog file consists of explanatory text lines and CALLS directives. The CALLS directives must begin in column 1 and are:

```
.topic primarykey[, aliaskey, aliaskey ...]
```

> Begin a new topic. <primarykey> is the official name of this topic, a string of up to 64 characters not containing blanks or commas. <aliaskey> is an alias name following the same syntax; the user can receive help on this topic by specifying either the <primarykey> or any of the <aliaskey>s.
>
> The next line of text (that is, which is not a CALLS directive) will be used as the one-line description for this topic. This description will appear along with the <primarykey> when the topic explanation is read and for any "associated topics" menus.

Any subsequent text lines are printed verbatim by CALLS when this topic is read, until a subsequent ".end" or ".topic" directive is found.

`.group key[, key ...]`

Used within a topic, joins this topic to a group of related topics given by the named keys. Each <key> may name a primary key used for a topic elsewhere or may be used solely in the ".group" directives for the related topics. A discussion on related topics appears below.

`.page` Forces a page break ("More..." prompt) at the current location in the explanatory text if the listing is to the terminal.

`.see key[, key, key ...]`

"See also", relates the named keys to the current topic, such that a menu of the named keys is presented after this topic text is read and the user is invited to select one of these keys for more help. A discussion on related topics appears below.

`.end` Terminates the current topic.

`.include` Directive that is recognized only by the CALLM utility. See the section later in this chapter on the CALLM utility for more information.

Relating Topics to Other Topics

Two of the CALLS directives are used to "relate" topics to other topics:

`.group` for topic groupings.

`.see` for referrals to "see also" topics.

A topic grouping occurs when several topics use the same key in a ".group" directive. When the user requests help on that key, a menu of all the topics that belong to this group is presented.

For instance,

```
.topic help, ?, ??
.group commands
Help!

description of help command
.end

.topic exit, quit
.group commands
Exit this program

description of exit command
.end
```

If the user requests help on “commands”, CALLS answers with:

```
The following topics are associated with "commands":

help -- Help!
exit -- Exit this program

Put cursor on desired name or type new name, press return.
```

If “commands” were used as a primary key for its own topic, that text would be listed before the above menu is given. For example, some general information about the “command” entry could be given before the menu of actual commands is presented.

The “.see” directive is used within a topic to refer the user to other topics that may be of interest. A similar menu of those topics is printed after CALLS lists the current text block. For example,

```
.topic NewProduct
one-line description of NewProduct

NewProduct relies heavily on ProductA and ProductB

.see ProductA, ProductB
.end
.topic ProductA
    .
    .
    .
   etc.
```

Index File

The first time CALLS runs on a catalog, and after subsequent updates of the catalog, CALLS builds a file called the index file. CALLS builds the index file in the same directory and with the same name as the catalog, but with type extension ".indx". More specifically, if the index file is missing or has an update timestamp that is older than the corresponding catalog, CALLS rebuilds the index file. CALLS will also attempt to rebuild the index if it appears that the index is invalid for the catalog, even if the update timestamps are in order.

Note The index file contains FMP internal file position pointers into the catalog file for the various topics, plus the keyword list and associated topic groupings. This means that the first person to run CALLS on a catalog after an update must have write access into the catalog directory for the index file to be successfully created. It is suggested that the system manager installing a new catalog immediately run CALLS on the catalog with the "−b" option to build the index.

CALLM Utility

CALLM merges together a number of text files that contain input to the CALLS program and creates a single compressed file suitable for reading by the CALLS facility.

Invoking the CALLM Utility

To invoke the CALLM utility use the following runstring syntax:

```
CI> callm [-options] commandfile destfile
```

where

-options — is a string of one or more of the following characters preceded by a dash:

l	suppress listing the names of files read
o	overlay an existing destfile
v	verify that an existing destfile should be overlaid
c	inhibit text compression of destfile. The default is to compress the text in destfile.

commandfile — is the name of a file containing a list of text files to be read, one per line. The CALLS input is extracted from each of these files and merged into destfile.

destfile — is the name of the destination file to be used as an input file for CALLS. If compression is performed then this file will be of file type 6004.

Compression is performed via the CompressAsciiRLE routine. As explained in the *RTE-A/RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037, this compression cannot be performed on characters that use the eighth bit of the ASCII code, such as binary data or extended ASCII character sets (Kanji, for example).

The commandfile is a file in format similar to MERGE command files. Each line contains either the name of an input text file to be merged into destfile or the line contains a comment prefixed with "* ". Each input text file may be the compressed output of a previous CALLM execution if it is of file type 6004. The suggested file type extension for CALLM command files is ".CMRG".

The text files may be program source code that contain CALLS input in comments, where the comment must start with character "*" or "{" in column 1 and be followed immediately by the CALLS directive or explanatory text. CALLM will include in the destfile only lines between and including ".topic" and ".end" CALLS directives, leaving out the intervening source code. If a plain text line not within a source code comment begins with either "*" or "{" in column 1, then that character must be doubled, such that the first one is discarded as a comment character.

.Include Directive

The CALLM utility processes an additional directive that includes another text file into the output file at the position where the directive is given. The syntax is:

```
.include <file>
```

These directives cannot be nested.

The example below shows the source code input format in FORTRAN or Macro:

```
*.topic mysubroutine
*.group subroutines
*one-line description of mysubroutine
*
*Calling sequence:
*
*      call mysubroutine(parm1,parm2)
*
*         :   etc.
*
*.see otherroutine
*.end
```

A Pascal example:

```
{.topic myprocedure
{one-line description of myprocedure
{
{Calling sequence:

          :   etc.

{.end
{}
```

Note the closing brace on the last line that terminates the Pascal comment.

See the section earlier in this chapter on the CALLS utility for more information about that program and the text input format expected.

A

Initializing, Formatting, and Sparing ICD/MAC Disks

All data is written on ICD/MAC disks in blocks of 256 bytes (128 words). The term block will be used here, since "sector" is defined in the RTE environment to mean 128 bytes, or 64 words. Each block has three components: the preamble, the data area and the postamble.

Preamble 11 words	User Data – 256 bytes 128 words	Postamble 3 words

The postamble consists of the Cyclic Redundancy Check (CRC) word and one word reserved for future use.

The 11-word preamble consists of four fields:

Sync Field 8 words	Sync Word 1 word	Cylinder Addr 1 word	Head/Block Addr 1 word

The synchronization field provides the timing signals that indicate the start of the block.

The synchronization word is a fixed value for the disk drive type (7905, 7906, 7910, 9895, and so on).

The cylinder address is contained in bits 15−0 of the address word.

The head/block address word contains the block address in bits 7−0, the head address in bits 12−8, and the block status in bits 15−13.

The preamble is more complex for the hard disk than for the flexible disk. (For flexible disks supplied by HP, only the D bit is significant.) The status portion of the preamble includes three bits:

S bit: (bit 15). Indicates that the track is a spare for a defective track. Only hard disks supplied by HP permit sparing.

P bit: (bit 14). Indicates a protected track. A protected track is a read-only track unless the FORMAT switch is activated on the disk drive.

D bit: (bit 13). Indicates that the track is defective and should be spared or not used at all.

In most cases, the address associated with the block will be the address of the block itself; that is, it will point to itself. The exceptions are:

Hard Disk: If the D bit is set (track defective), the address points to the track used as a spare.

If the S bit is set (the track is a spare), the address points to the defective track for which it is a spare.

Flexible Disk: If the D bit is set (track defective), the address is not relevant (may be anything).

The RTE utility programs will declare an entire track bad if one bad block is found on the track. This method of management results in less head movement (hence faster access) when encountering spared blocks (tracks).

For the hard disk, once a track has been spared, the disk controller finds that the track is spared whenever a seek is made to that defective track. The disk drive will automatically access the spare track instead. This process is transparent to the software.

For the flexible disk, although the technique is not called sparing, the effect is virtually the same. Once the D bit has been set for the defective tracks, the utility gives the drive a command to skip bad tracks. For example, the outermost track is normally track zero. But if the D bit has been set for this track, then when commanded to access track zero, this track (and any following bad tracks) will be skipped. The first good track becomes track zero.

If a bad track develops in using the disk, the hard disk sparing capability has the advantage that a simple recovery process is possible by running the SPare function. This is not possible with the flexible disk.

Initializing and Sparing a Hard Disk

Disk initialization uses the track map table in RTE-6/VM to determine the starting cylinder, head number, number of tracks and number of spare tracks for each disk LU. Tracks in the spare pool are used only as replacements for the defective tracks; they are otherwise unused.

The initialization procedure is as follows:

1. Clean up tracks in the spare pool. The full block (including the preamble and postamble) is read and the status bits examined. If the D bit is set (defective), go to next track. A track from the spare pool will not, itself, be spared. If the D bit is not set, rewrite the full track with zeroes, including the S, P, and D bits. Verify by reading back the track. If found bad during verify, set the D bit.

2. Read each track in the mapped portion of the LU. If the D bit is set or if the read is unsuccessful, set the flag indicating sparing is needed.

3. If sparing is needed:

   a. Prepare the preamble of the defective track. Set the D bit and set the address of the spare track in the preamble of the defective track.

   b. Prepare the preamble of the spare track. Set the S bit and set the address of defective track in the preamble of the spare track.

4. Write initialize the track. If sparing is required, use the preambles set in step 3. If sparing is not required (good track found in step 2), the address will be that of the good track, it will point to itself. If this is a spare track, the address will be that of the defective track (a backward pointer). If this is a defective track, then the address will be that of the spare track (a forward pointer).

   The same procedure is used to spare an individual track (SP) except that, where possible, the data buffer is copied from the defective track on a block-by-block basis. In recovering user data, an offset read is performed to pick up data that could not normally be read with the head aligned to the center of the track.

5. Verify the whole track on a block-by-block basis. If successful, move to the next track and repeat the process. If unsuccessful, get the next spare track and repeat the sparing procedure.

Formatting a Flexible Disk

Formatting is the most basic form of initialization. Normal reads and writes cannot be made to a disk until it has been formatted. The initialization process for the hard disk assumes that the disk has been previously formatted at least once. The formatting process writes the full block as described above, tagging the block with an address. Initially this address points to itself, but may be changed if the block is defective or a spare.

A hard disk is always pre-formatted before shipment from HP to ensure the interchangeability of disk packs. Therefore the capability to format a hard disk is not included in the FORMT and FORMC utilities. A flexible disk, however, must be formatted before it can be used initially.

Interleaving (Fill Number)

When formatting a flexible disk, you can define the order in which consecutive logical blocks are accessed on the disk by specifying a "fill number" in the FOrmat function runstring. This value defines the number of physical blocks to be skipped between logical blocks. If the fill number is zero, then the N+1th block is accessed following the Nth block. Otherwise, the number of blocks specified by the fill number are skipped. Once the disk is formatted, the fill number is transparent to the user software.

The fill number may be chosen based upon a knowledge of how the user software is to access the disk and a knowledge of the rotational speed of the disk. For example, suppose that the software takes T milliseconds to read and process one block and it picks up sequential blocks. If the time to rotate once around the disk is divided by the number of blocks per track, then the time to move over a single block is found. The fill number is then computed so that the number times the time to scan one block is slightly greater than the processing time. Thus, the next block will rotate into the proper position for access just when it is needed.

As an example, assume

360 RPM (rotations/minute) = 167 milliseconds/rotation
30 blocks/track = 5.6 milliseconds/block

If the processing time is less than 5.6 ms, then a fill number = 1 can be specified, since processing of the first block will be completed before the next block becomes available. If processing time is between 5.6 ms and 11.2 ms, a fill number = 2 should be specified. If the fill number were 0 or 1, and the processing time is between 5.6 and 11.2 ms, the disk would have to rotate one full revolution before the next block could be picked up (167 ms versus less than 5.6 ms).

Most user programs accessing files will define a Data Control Block (DCB) of 144 words, of which 128 words (1 block) will be used for disk transfers. Therefore, a minimum fill number can be computed in terms of the amount of time to complete one EXEC request to write a block and return to the user program. Or the value can be determined experimentally by trying several fill numbers. For sequential access to the files with a DCB of 144 words, an approximate number to use would be in the range 4 to 6.

Of course, the higher the fill number, the longer the formatting process will take. The performance graph in Figure A-1 plots the formatting time against the fill value.

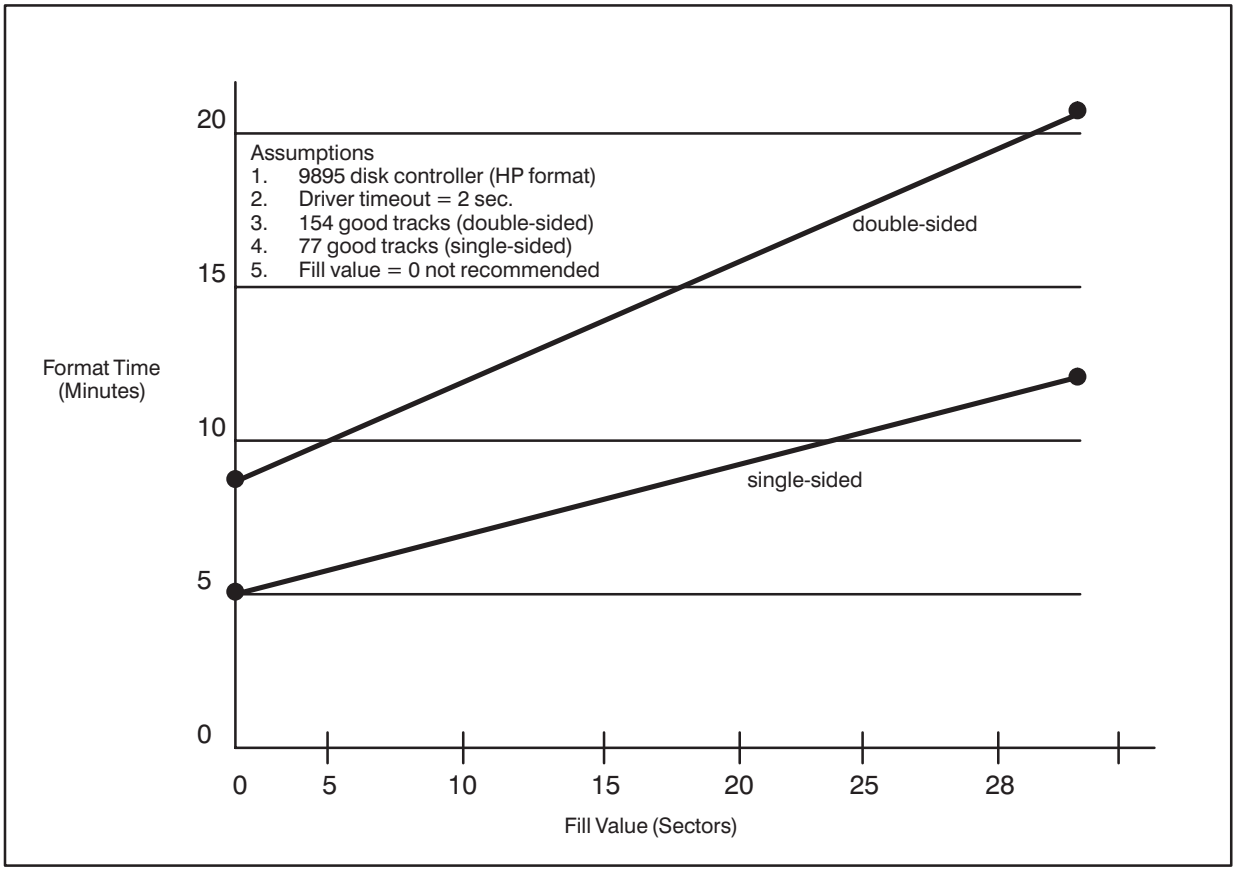


Figure A-1. Formatting Time vs. Fill Value

B

CS/80 Exerciser Utility (EXER)

Introduction

EXER is a CS/80 Exerciser Utility program used to diagnose and troubleshoot CS/80 disk drives on HP 1000 systems. It is available as an offline program and an online program. This appendix covers using the EXER utility in an online environment. The offline version of EXER is documented in the *CS/80 External Exerciser Reference Manual*, part number 5955-3462.

The online version of EXER is essentially the same program file as in the offline version. The offline version for RTE-6 runs in a memory-based RTE-IVE system environment.

Getting Started

If you have installed a Primary system, EXER is already loaded as a program under /PROGRAMS. In actuality, there are two programs required: EXER, the parent, and EXER1, the child, which is scheduled by EXER as needed. Before executing EXER, the child program EXER1 must be RP'ed; otherwise, SC05 errors result.

```
CI> rp,exer1
CI> exer
```

At this point you are in the exerciser. At any point in the program, the command "HELP" will list valid commands. (A command file can be used to RP the EXER1 program and run EXER, simplifying the process. Refer to the *RTE-6/VM CI User's Manual* for command file usage.)

Loading the program

The relocatables EXER.REL and EXER1.REL are supplied with the RTE-6/VM software, along with the link command files #EXER and #EXER1.

Using the Exerciser

When you run EXER, the first thing the program wants to know is what disk LU to test. The program scans through the device reference table looking for all entries with a driver type 33B (disk). It then reports all the LUs that match and asks you to select one. Realize that EXER itself does *not* concern itself with individual LUs, it merely needs to differentiate between multiple disks connected to the system. Once it is given *any* LU on a given disk, EXER looks at the disk as *one physical* volume. All subsequent EXER commands will address the *entire* disk.

The following is a sample of the LU table reported by EXER for a 5.2 RTE−6 system:

```
CS/80 EXTERNAL EXERCISER -- Rev. 5020  02-07-90

*********  CS80 LU s  ************
Sel Code   LU #   HPIB Addr   Unit
========   ====   =========   ====


   14        02        1         0
             03        0         0
             30        0         0
             31        0         0
             32        0         0
             33        0         0


Input DRIVE LU?
```

Note EXER only looks for LUs up to 63.

At this point, you must check the list for the appropriate LU that corresponds to the disk you wish to examine. For example, if you wanted to examine the boot disk for this Primary, you could enter "02" as the LU. Since EXER does not care about LUs, you could use *any* LU 2, 3, or 30−33 since they all correspond to the same physical disk drive (HPIB address 0, on select code 14B).

The next thing EXER does is attempt to identify the device type attached, with the result appearing as follows:

```
Input DRIVE LU? 02

LU 2 is a 7914

Current unit = 0
```

If the identify function fails, for example, if the LU specified is turned off or disconnected, then the following message is displayed:

```
Error on initial describe, please check drive.

Input DRIVE LU?
```

Note that a “broken” disk could also cause this error. EXER then asks for the LU again. Enter a valid (existent LU) .

Sometimes, entering a non-existent LU will cause a “hang” waiting for the CS/80 timeout. Be patient. EXER will come back. OF’ing EXER may not clear the timeout.

After EXER has ID’ed the disk successfully, it will display the prompt:

```
EXER>
```

Typing “HELP” at this point will display all available commands. Note that commands do not need to be entered in their entirety, for instance “H” will suffice for “HELP”.

The following are valid EXER commands:

```
EXER>help

CHANGE LU       - change the lu that you are working on
CANCEL          - cancel transaction
CICLEAR         - channel independent clear
DESCRIBE        - describe selected unit
ERT LOG         - output error rate test log
EXIT            - exit program or command
FAULT LOG       - output fault log
HELP            - output help information
INPUT           - change input file or lu
OUTPUT          - change output file or lu
PRESET          - update device logs
REQSTAT         - request status
REV             - output firmware revision
RF SECTOR       - read full sector
RO ERT          - perform read-only error rate test
RUN LOG         - output run log data
TABLES          - output device tables
TERM            - input/output at terminal
UNIT            - set unit number
```

Selected Command Descriptions

A full description of all the EXER commands is available in the *CS/80 External Exerciser Reference Manual*, part number 5955-3462. The commands most useful to a System Manager or programmer who desires to check the condition of a disk drive are briefly described below.

CHANGE LU This command allows testing another disk drive without exiting EXER.

DESCRIBE As the name implies, this gives a description of the disk drive. For example:

```
DESCRIBE UTILITY
LU 2 is a 7914

Current unit = 0

MODEL: 7914
UNIT: 0
TYPE: DISC
Maximum cylinder address =          1151
Maximum head address =                6
Maximum sector address =             63

Maximum block address =           516095
Current interleave factor = 1
```

ERT LOG This command displays the Error Rate Test log information for the drive. This log is used *only* by the exerciser and does not reflect usage from the system (RTE). See RUN LOG.

```
READ ERT LOG UTILITY
LU 2 is a 7914

Current unit =  0

Input the head (0 - 6) or ALL? 0

Head # = 0
# sectors read = 0
Correctable errors = 0
Uncorrectable errors = 0
 No errors logged
```

FAULT LOG This displays *all* faults logged by the disk during system or exerciser operation.

```
FAULT LOG UTILITY
LU 2 is a 7914

Current unit =  0

No drive faults
```

RUN LOG This displays the error log information from online system usage. This is useful information for the system manager to ascertain disk usage and error rates.

```
READ RUN LOG UTILITY
LU 3 is a 7914

Current unit =  0

Input the head (0 - 6) or ALL? 0

Head # = 0
# sectors read = 429483
Correctable errors = 0
 No errors logged
```

TABLES This will display information on the number of sparing operations that have been performed on the drive with the FORMC utility from the system. By definition, a "secondary" spare is any user-invoked spare operation. (Primary spares are factory done and are not ascertainable.) Note that all sparing is handled by the disk controller, not RTE. For more information on sparing tracks, see the description of FORMC in this manual.

```
READ DRIVE TABLES UTILITY
LU 30 is a 7914

Current unit =  0

SPARE TRACK TABLE

Head number =  0
# of secondary spares =  4
# of tracks used =  2
# of logical tracks spared =  1
```

```
 CYL       TYPE       SCALAR
=====    =========    =======
 491     SECONDARY       50

Head number =  1
# of secondary spares =  0
# of tracks used =  0
# of logical tracks spared =  0
```

REV This reads the revision level of the firmware in the disk microproessor. This may be useful information for field service personnel for troubleshooting purposes.

```
READ REVISION NUMBER UTILITY
LU 2 is a 7914

Current unit =  0

Part    Revision
number   number           *NOTE   These part numbers refer
------ --------                   to the individual firmware
   1     5 - 0                     ROMs on the processor card.
   2     5 - 1
   3     5 - 0
   4     5 - 0
   5     5 - 0
   6     5 - 0
```

RO ERT This command performs a Read Only error rate test on the disk volume. The test area used is determined by user input, regardless of the LU entered when EXER was started.

Caution Executing an RO ERT will lock the disk to EXER for the duration of the test. *Do not* use a loop count of INFinite for the RO ERT command as this will hang the disk and require a reboot of the system.

INPUT Allows use of an “answer” file to supply the responses to EXER, allowing the program to be scheduled automatically from CI. See OUTPUT command.

OUTPUT Allows the output from EXER to be directed to a disk file for later examination by the user. When used with the INPUT command, EXER could be time-scheduled, get its answers from a command file (INPUT) and store the results in a file (OUTPUT) for viewing later.

Error Handling

Many errors occurring during the execution of EXER can be attributed to either user input error (wrong LU number) or certain normal conditions, such as loading a tape in the drive (for example, HP 7914 with built-in CTD). For the most part these can be ignored.

As a rule, the errors seen by looking at the fault and run logs require a much greater knowledge of the hardware than the average user possesses. These types of errors are best left to an HP service representative to analyze.

If you are merely checking the error logs on your disk as a preventative measure, without experiencing any disk related system problems, the majority of any errors logged will be inconsequential. Also, different CS/80 disks have different allowable error rates and certain limits for certain fault conditions. These are best left to trained service personnel to interpret.

The absolute best preventative measure you can take is a regular schedule of system backups. Since the CS/80 drives are relatively easy and fast to repair, your primary concern should be having backups, not downtime.

EXER and CS80 Tape Drives

The online EXER program is not intended to be used for either built-in tape drives in the HP 791x disk family or HP 9144 standalone drives. (The offline version of EXER contains the additional program TAPE, which is not available online.)

Index

E

F

I

J

K

L

M

O

P

R

S

T

V

W