



**RTE-A**

**Utilities Manual**

---

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

**Manual Part No. 92077-90004  
E1292**

**Printed in U.S.A. December 1992  
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, 1992 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 which 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 what 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 .....	Jul 1983	.....
Update 1 .....	Dec 1983	..... Update TF, File System Utilities
Update 2 .....	Jun 1984	..... Update and Add utilities
Reprint .....	Jun 1984	..... Update 1 & 2 Incorporated
Update 3 .....	Jan 1985	.....
Reprint .....	Jan 1985	..... Update 3 Incorporated
Update 4 .....	Jul 1985	..... Add FORMA and ERTISH
Reprint .....	Jul 1985	..... Update 4 Incorporated
Update 5 .....	Jan 1986	.....
Reprint .....	Jan 1986	.....
Update 6 .....	Oct 1986	..... Add FST
Reprint .....	Oct 1986	..... Update 6 Incorporated
Third Edition .....	Aug 1987	..... 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)
Update 2 (SCSI) .....	Jul 1990	..... Added SCSI Rev. 5240
Update 3 .....	Jan 1992	..... Rev. 5270
Fifth Edition .....	Dec 1992	..... Rev. 6000 (Software Update 6.0)

**HP Computer Museum**  
**[www.hpmuseum.net](http://www.hpmuseum.net)**

**For research and education purposes only.**



# Preface

This manual describes the scope, format and use of utilities available in the RTE-A Operating System. It is intended to be the primary reference source for programmers using RTE-A utilities. For a complete list of the RTE-A manual set, refer to the *RTE-A Index and Glossary*, part number 92077-90036.

## How This Manual Is Organized

Chapter 1	Provides an overview of the RTE-A utilities.
Chapter 2	Explains the file backup utilities, FST, TF, FC, and LIF. FST and TF are compatible with the UNIX* operating system TAR utility.
Chapter 3	Describes the physical disk image backup utilities, ASAVE, ARSTR, DSAVE, DRSTR, and COPYL.
Chapter 4	Describes file system pack, file space, verify, and system conversion utilities: FPACK, MPACK, FREES, FOWN, FVERI, and FSCON.
Chapter 5	Discusses the formatting utilities, FORMA, FORMC, FORMF, and FORMT.
Chapter 6	Discusses the file manipulation utilities, MERGE, OLDRE, and SCOM.
Chapter 7	Explains the use of system setup utilities, AB2MI, MI2AB, INSTL, FPUT, and CSYS.
Chapter 8	Describes the notification utility, NOTIFY, and the printing utility, PRINT.
Chapter 9	Describes the system status utilities, METER, SAM, and SPORT.
Chapter 10	Describes the multiuser accounting utilities, RINFO, SINFO, KILLSSES, and SESLU.
Chapter 11	Describes the keyword indexed help utilities, CALLS and CALLM.
Appendix A	Describes the FMGR commands for disk/file manipulation, transferring files, and FMGR scheduling.
Appendix B	Describes the CS/80 Exerciser Utility (EXER) used to diagnose and troubleshoot CS/80 disk drives on HP 1000 systems.

---

\* UNIX is a registered trademark of UNIX System Laboratories, Inc. in the U.S.A. and other countries.

## 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)
```

The command syntax conventions are as follows.

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> <u>as testprogram 2 c</u>
[ ]	Optional parameters are shown in square brackets. If you omit a parameter, use a comma as a placeholder before specifying additional parameters.
/	Separates alternate choices. The C/D shown above indicates either C or D can be entered. (Note that the slash is also used in the command string to designate directories and subdirectories.)
, or blank	Use a comma or a space as a delimiter between commands and parameters. Blank spaces are used throughout this manual in all syntax strings.
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>.

# Table of Contents

---

## Chapter 1 Introduction

Backup and File Interchange Utilities .....	1-1
Physical Disk Image Backup Utilities .....	1-1
File System Utilities .....	1-2
Formatting Utilities .....	1-2
File Manipulation Utilities .....	1-3
System Setup Utilities .....	1-3
NOTIFY and PRINT Utilities .....	1-3
System Status Utilities .....	1-3
Multiuser Accounting Utilities .....	1-4
Keyword Indexed Help Utilities .....	1-4
File Manager Control .....	1-4
CS/80 Exerciser Utility (EXER) .....	1-4

## Chapter 2 Backup and File Interchange Utilities

Backup Utilities .....	2-1
File Interchange on RTE-A .....	2-3
File Storage to Tape (FST) .....	2-4
Calling FST .....	2-5
FST Commands .....	2-7
Command Stack (/) .....	2-8
Backup (BA) .....	2-8
Directory File (DF) .....	2-9
List Directory (DL) .....	2-9
Exit (EX) .....	2-10
Begin Backup/Restore (GO) .....	2-10
Help (HE) .....	2-10
List Comment File (LC) .....	2-10
List Header (LH) .....	2-11
List Selected Files (LI) .....	2-11
Select Log Device/File (LL) .....	2-11
List Non-Selected Files (LN) (Restore Only) .....	2-12
Specify Tape LU/Archive File (MT) .....	2-12
Next (NE) .....	2-13
Position (PO) .....	2-13
Previous (PR) .....	2-13
Restore (RE) .....	2-14
Run (RU) .....	2-15
Select Comment File (SC) (Backup Only) .....	2-15
Set Tape Density (SD) (Backup Only) .....	2-15
Secure (SE) .....	2-16
Show (SH) .....	2-16
TAR (TA) .....	2-16
Title (TI) (Backup Only) .....	2-17



Transfer to Command File (TR) .....	2-18
Unselect (UN) .....	2-18
FST Options .....	2-19
Append (A) (Backup Only) .....	2-20
Brief (B) .....	2-21
Clear (C) .....	2-21
Duplicate (D) (Restore Only) .....	2-21
Faulty (F) (Restore Only) .....	2-21
Inhibit (I) .....	2-21
Keep (K) .....	2-21
Lock (L) (Backup Only) .....	2-22
MinDir (M) (Restore Only) .....	2-22
Normal (N) (Backup Only; VC+ Only) .....	2-22
Original (O) (Restore Only) .....	2-22
Purge (P) (Backup Only) .....	2-23
Quiet (Q) .....	2-23
RwndOff (R) .....	2-23
SrchApp (S) (Restore Only) .....	2-23
Update (U) .....	2-23
Verify (V) .....	2-24
Whole (W) (Backup Only) .....	2-24
Yes (Y) .....	2-24
Z (Z) .....	2-24
File Masking and Renaming .....	2-25
D, K, N, and S Qualifiers .....	2-25
Backing Up .....	2-25
Restoring .....	2-27
Incremental Backup .....	2-29
Restoring from Incremental Backups .....	2-31
Appending Data .....	2-33
Consecutive Backups .....	2-33
Multiple Reels .....	2-33
Tape Loading .....	2-34
TF Compatibility .....	2-34
TAR Compatibility .....	2-35
UNIX Compatibility .....	2-35
Rescuing Files from an Overwritten Tape .....	2-36
Disk Directory File .....	2-37
Shareable EMA .....	2-38
Replacing Reserved Characters .....	2-38
Recommended System Usage .....	2-39
Streaming .....	2-39
FST Format .....	2-40
Installing FST .....	2-42
FST Error Handling .....	2-42
FST Error Messages and Warnings .....	2-43
Tape Filer (TF) .....	2-53
Calling TF .....	2-54
TF Commands .....	2-54
Copy (CO) .....	2-55
CO Command Source and Destination Parameters .....	2-56
CO Command Options .....	2-57
Copy Examples without Subdirectories .....	2-60

Copy Examples with Subdirectories .....	2-66
Copy Examples Using DS .....	2-74
Default (DE) .....	2-77
Directory List (DL) .....	2-78
Relation of the DL Command to the CO Command .....	2-80
Exit (EX) .....	2-81
Group Copy Commands (GR, EG, and AG) .....	2-81
Help (HE) .....	2-82
List Header File (LH) .....	2-82
List Device (LL) .....	2-83
Title (TI) .....	2-83
Transfer Command (TR) .....	2-84
Tape Protection and the K Option .....	2-85
Time Stamps .....	2-85
Create Time .....	2-86
Update Time .....	2-86
Access Time .....	2-86
Missing Time Stamps .....	2-86
Maintaining the System Time .....	2-86
Incremental Backup .....	2-87
Backup Bit, B Qualifier, and C Option .....	2-87
Standard Incremental Backup Procedure .....	2-88
Multiple Copies of the Same Backup .....	2-89
Reusing the Backup Bit .....	2-89
Restoring Incremental Backups .....	2-89
Restoring Older Versions from Incremental Backup Tapes .....	2-90
Setting Backup Bits on Restore .....	2-91
Alternatives to Standard Incremental Backup .....	2-91
Replacing Duplicate Files .....	2-91
Automatic Creation of Directories on Restore .....	2-92
Saving and Restoring File Properties .....	2-93
System Backup and Restore .....	2-94
Handling Disk Full Errors .....	2-96
File Access During Backup and Restore Operations .....	2-96
Using TF with FMGR Files .....	2-96
Using TF with FC Tapes .....	2-97
Multi-Tape Backup and Restore .....	2-98
UNIX Compatibility .....	2-99
File Formats on FMP and UNIX .....	2-99
Copying Files Between FMP and UNIX .....	2-100
Summary of Copying Recommendations .....	2-102
Directory File Names .....	2-103
File Name Letter Case .....	2-103
Dots Used in File Names .....	2-104
Special Characters .....	2-105
File Name Length—Moving Files from FMP to UNIX .....	2-105
File Name Length—Moving Files from UNIX to FMP .....	2-105
Linked Files .....	2-105
Appending to Tapes .....	2-106
File Types, Security Codes .....	2-106
Time Stamps .....	2-106
Root Directory .....	2-106
Header and Final Checksum Dummy Files .....	2-106

TF Tape Format .....	2-107
Installing TF .....	2-108
File Copy (FC) .....	2-109
Calling FC .....	2-109
FC Commands .....	2-110
Command Summary Function (?) .....	2-112
Abort (AB) .....	2-113
Name Comment File (CF) .....	2-113
Cartridge List (CL) .....	2-113
Copy (CO) .....	2-114
CO Command Source and Destination Parameter Considerations .....	2-116
CO Command Options .....	2-117
CO Command Examples .....	2-120
Default (DE) .....	2-123
Directory List (DL) .....	2-124
Echo Command (EC) .....	2-126
Exit (EX) .....	2-126
Group Copy Commands (GR, EG, and AG) .....	2-126
List Comment, List Header Files (LC, LH) .....	2-127
List Device (LL) .....	2-127
Scratch Area Definition (SC) .....	2-128
Title (TI) .....	2-128
Transfer (TR) .....	2-129
Tape Handling .....	2-129
Destination Disk Handling .....	2-131
Performance Considerations .....	2-131
Loading FC .....	2-133
Using Globals in Transfer Files .....	2-133
Error Handling in Transfer Files .....	2-135
FC Error Messages .....	2-137
Errors Requiring Operator Action/Response .....	2-137
Information Messages and Warnings .....	2-138
Disk Data I/O Errors .....	2-141
Non-Fatal Tape Read Errors .....	2-141
Errors Affecting a Single File .....	2-142
Loss of Unidentified Files on Copy from Tape .....	2-145
Disk-to-Tape Copy Verify Errors .....	2-145
Errors that Cause Rejection of Current Source Tape Volume .....	2-145
Command Syntax, Parameter Errors (Command is Skipped) .....	2-146
Command Out-of-Sequence Errors (Command is Skipped) .....	2-149
Other Errors that Terminate Current Command .....	2-149
Other Errors .....	2-151
Errors that Terminate FC .....	2-151
HP Computer Systems File Copy (LIF) .....	2-153
Naming Conventions .....	2-153
Calling LIF .....	2-154
LIF Commands .....	2-154
Copy (CO) .....	2-156
Directory List (DL) .....	2-157
Exit (EX) .....	2-158
Help (HE) .....	2-158
Initialize (IN) .....	2-158
List (LI) .....	2-159

Set Logical List Device (LL) .....	2-159
Mount Cartridge (MC) .....	2-159
Pack Cartridge (PK) .....	2-160
Purge (PU) .....	2-160
Rename (RN) .....	2-160
Store (ST) .....	2-161
Severity (SV) .....	2-162
Transfer Control (TR) .....	2-162
LIF Error Handling .....	2-163

## **Chapter 3**

### **Physical Disk Image Backup Utilities**

Using the Utilities .....	3-1
Compatibility with Other Disk Backup Utilities .....	3-1
Compatibility Among Disks .....	3-1
CS/80 Cartridge Tape Drive .....	3-2
ASAVE Physical Backup Utility .....	3-3
Calling ASAVE .....	3-3
ASAVE Commands .....	3-4
Abort, End, and Exit (AB) (EN) (EX) .....	3-5
Help (HE) .....	3-5
List Header (LH) .....	3-5
List Device (LL) .....	3-5
Rewind (RW) .....	3-6
Save (SA) .....	3-6
SA Command Options .....	3-6
Tape LU (TA) .....	3-8
Title (TI) .....	3-8
User Error Handling (UE) .....	3-9
ASAVE Command Examples .....	3-9
The Save Operation .....	3-10
Unit Save LU Checking .....	3-10
Disk Locking .....	3-10
Tape Positioning .....	3-11
Save File Records .....	3-11
Saving the Disks to Tape .....	3-11
Verification .....	3-12
Break Detection .....	3-13
Loading ASAVE .....	3-13
ASAVE Tape and File Formats .....	3-13
ASAVE Tape Records .....	3-14
Header Records .....	3-14
Save Definition Record .....	3-15
Data Record .....	3-15
End-of-Data Record .....	3-16
ASAVE Example .....	3-16
ASAVE Error Messages .....	3-17
ARSTR Physical Restore Utility .....	3-18
Calling ARSTR .....	3-18
ARSTR Commands .....	3-18
Abort, End, and Exit (AB) (EN) (EX) .....	3-19
Help (HE) .....	3-20
List Header (LH) .....	3-20

List Device (LL) .....	3-20
Restore (RE) .....	3-21
RE Command Options .....	3-22
Rewind (RW) .....	3-23
Tape LU (TA) .....	3-23
User Error Handling (UE) .....	3-23
ARSTR Command Examples .....	3-24
The Restore Operation .....	3-24
Checking the Disks .....	3-24
Sectors per Track Must Match on Disks .....	3-25
Total Number of Tracks May Differ .....	3-25
Disk Locking .....	3-26
Order of Disk Restoration .....	3-26
Unit Restore .....	3-26
Save Definition Records .....	3-26
Restoring the Data .....	3-27
Verification .....	3-27
Break Detection .....	3-28
Loading ARSTR .....	3-28
Offline System .....	3-28
ARSTR Example .....	3-29
ASAVE and ARSTR Error Messages and Warnings .....	3-30
ASAVE Messages .....	3-30
ASAVE and ARSTR Shared Error Messages .....	3-31
ASAVE and ARSTR Shared Warning Messages .....	3-33
ARSTR Error Messages .....	3-34
ARSTR Warning Messages .....	3-34
Disk-to-Disk Copy (COPYL) .....	3-35
Calling COPYL .....	3-35
The Copy Operation .....	3-36
COPYL Error Messages .....	3-36
DSAVE Physical Backup Utility .....	3-37
Calling DSAVE .....	3-37
DSAVE Commands .....	3-38
Abort, End, and Exit (AB) (EN) (EX) .....	3-39
Help (HE) .....	3-39
List Header (LH) .....	3-39
List Device (LL) .....	3-41
Media (ME) .....	3-41
Save (SA) .....	3-42
SA Command Options .....	3-42
Title (TI) .....	3-43
User Error Handling (UE) .....	3-44
The Save Operation .....	3-44
Unit Save LU Checking .....	3-45
Disk Locking .....	3-45
Removable Disk Overwriting .....	3-45
Save File Records .....	3-46
Saving to the Removable Disk .....	3-46
Verification .....	3-47
Break Detection .....	3-47
Loading DSAVE .....	3-48
DSAVE Removable Disk Media and File Formats .....	3-49

Removable Disk Media Format .....	3-49
Removable Disk Media Records .....	3-49
Header Records .....	3-49
DSAVE Error Messages .....	3-51
DRSTR Physical Restore Utility .....	3-52
Calling DRSTR .....	3-52
DRSTR Commands .....	3-52
Abort, End, and Exit (AB) (EN) (EX) .....	3-53
Help (HE) .....	3-53
List Header (LH) .....	3-54
List Device (LL) .....	3-54
Media (ME) .....	3-55
Restore (RE) .....	3-55
RE Command Options .....	3-56
User Error Handling (UE) .....	3-57
The Restore Operation .....	3-57
Checking the Disks .....	3-58
Sectors per Track Must Match on Disks .....	3-58
Total Number of Tracks May Differ .....	3-58
Disk Locking .....	3-59
Order of Disk Restoration .....	3-59
Unit Restore .....	3-59
Restoring Fixed Disks from Removable Disks .....	3-60
Header Records .....	3-60
Restoring the Data .....	3-61
Verification .....	3-61
Break Detection .....	3-62
Loading DRSTR .....	3-62
DSAVE and DRSTR Error Messages .....	3-62
DSAVE Error Messages .....	3-62
DSAVE and DRSTR Shared Error Messages .....	3-63
DRSTR Error Messages .....	3-64

## Chapter 4

### File System Utilities

File System Conversion (FSCON) .....	4-2
Calling FSCON .....	4-2
Requirements for Successful Conversion .....	4-2
The Conversion Process .....	4-3
File Renaming .....	4-3
Converted CI Directory Entries .....	4-3
FSCON Error Messages .....	4-5
File System Pack (FPACK) .....	4-6
Calling FPACK .....	4-6
The Packing Process .....	4-6
Moving Directories .....	4-8
Moving Subdirectories .....	4-9
Moving Files .....	4-10
File Compacting and Disk Pack (MPACK) .....	4-11
Calling MPACK .....	4-11
MPACK Options .....	4-11
Compacting and Reporting Options .....	4-12
Packing Options .....	4-15

Logging Option .....	4-16
MPACK Examples .....	4-16
FREES – Indicate Free Space on a Volume .....	4-17
Report File Space by Owner (FOWN) .....	4-20
Calling FOWN .....	4-20
FOWN Examples .....	4-20
File System Verification (FVERI) .....	4-22
Operating Instructions .....	4-24
Error Recovery .....	4-24
Error Messages .....	4-25

## Chapter 5

### Formatting Utilities

General Formatting Information .....	5-2
The Preamble .....	5-3
The Postamble .....	5-3
Initializing and Sparing Hard Disks .....	5-4
Sparing vs Skipping .....	5-4
Interleave Factor Calculation .....	5-4
CS/80 Definitions .....	5-6
CS/80 Disk Formatting (FORMC) .....	5-7
Calling FORMC .....	5-7
Command Execution .....	5-8
Device Driver Status .....	5-8
Break Detection .....	5-8
FORMC Commands .....	5-9
Abort, End, and Exit (AB) (EN) (EX) (/E) .....	5-9
Format (FO) .....	5-10
Help (?) .....	5-10
Spare (SP) .....	5-10
Verify (VE) .....	5-11
The FORMC Formatting Operation .....	5-11
Entering the LU .....	5-11
Disk Formatting .....	5-12
Tape Formatting .....	5-14
The FORMC Sparing Operation .....	5-16
Entering the LU .....	5-17
Specifying the Track .....	5-17
Sparing Defective Disk Blocks .....	5-17
Data Loss During Sparing .....	5-18
The FORMC Verifying Operation .....	5-19
Entering the LU .....	5-19
Readying the Tape .....	5-19
Verifying the Entire Disk LU or Tape .....	5-19
Specifying the Start Track or Block .....	5-20
Specifying the Number of Tracks or Blocks .....	5-20
The Verification Process .....	5-21
Loading FORMC .....	5-21
FORMC Error Messages .....	5-22
Disk Formatting Utility (FORMF) .....	5-23
Calling FORMF .....	5-23
Command Execution .....	5-24
Device Driver Status .....	5-24

Break Detection .....	5-24
FORMF Commands .....	5-25
Abort, End, and Exit (AB) (EN) (EX) (/E) .....	5-25
Format (FO) .....	5-26
Help (?) .....	5-26
Verify (VE) .....	5-26
The FORMF Formatting Operation .....	5-27
Entering the LU .....	5-27
Specifying an Interleave Factor .....	5-27
The Formatting Process .....	5-28
The FORMF Verifying Operation .....	5-28
Entering the LU .....	5-28
Verifying the Disk .....	5-28
FORMF Error Messages .....	5-29
Offline Disk Formatting and Initialization (FORMT) .....	5-31
Calling FORMT .....	5-31
FORMT Commands .....	5-32
End (EN) .....	5-32
Format a Mini-Floppy Disk (FO) .....	5-32
Help (??) .....	5-33
Initialize an LU (IN) .....	5-33
Reformat a Hard Disk (RE) .....	5-33
Spare a Track (SP) .....	5-34
Verify an LU (VE) .....	5-34
Generating a FORMT System .....	5-34
The FORMT Formatting Operation .....	5-35
Entering the LU .....	5-35
Confirming Formatting .....	5-35
Sector Interleaving .....	5-36
The Formatting Process .....	5-36
The FORMT Initializing Operation .....	5-37
Entering the LU .....	5-37
Confirming Initializing .....	5-37
The Initializing Process .....	5-37
The FORMT Reformatting Operation .....	5-38
Before You Use the RE Command .....	5-38
Entering the LU .....	5-38
The Reformatting Process .....	5-39
After You Reformat the Disk .....	5-39
The FORMT Sparing Operation .....	5-40
Entering the Track Number .....	5-40
The Sparing Process .....	5-40
The FORMT Verify Operation .....	5-41
Entering the LU .....	5-41
The Verify Process .....	5-41
FORMT Error Messages .....	5-42
Initialize and Spare Utility (FORMA) .....	5-44
FORMA Definitions .....	5-45
Calling FORMA .....	5-47
FORMA Commands .....	5-47
Help (??) .....	5-47
Initialize Media (IM) .....	5-47
Spare Sector (SS) .....	5-48



SS Command Options .....	5-49
Converting from FORMF to FORMA .....	5-50
Installing a New Disk .....	5-50
FORMA Error Handling .....	5-51
FORMA Error Messages .....	5-51
User Messages .....	5-51
User Errors .....	5-51
Disk Errors .....	5-52
Disk Error Decoding .....	5-53
Error Rate Test Utility (ERTSH) .....	5-54
ERTSH Structure .....	5-54
ERTSH Commands .....	5-56
Help (??) .....	5-57
Build Test Pattern (BP) .....	5-57
Define Cylinder Range (CY) .....	5-57
EXIT (EX) .....	5-58
Define Disk Head Range (HD) .....	5-58
Extra Log (LL) .....	5-58
Selecting Disk LU to Test (LU) .....	5-59
Set Test Pass Count (P#) .....	5-59
Execute Read-Only Error Rate Test (RO) .....	5-60
Define Disk Sector Range (SC) .....	5-60
Spare Sectors (SS) .....	5-61
Select Test Pattern (TP) .....	5-61
Execute Write/Read Error Rate Test (WR) .....	5-62
Using ERTSH .....	5-63
Disks Formatted with FORMF .....	5-63
Initializing and Testing a New Disk .....	5-63
Typical Use of ERTSH .....	5-63
Break Detection .....	5-66
Specialized Use of ERTSH .....	5-66
Installation Procedure Summary .....	5-67
Error Interpretation .....	5-68
Error Messages .....	5-71

## Chapter 6

### File Manipulation Utilities

Concatenate Many Files into One (MERGE) .....	6-1
Calling MERGE .....	6-1
MERGE Options .....	6-2
The MERGE Operation .....	6-2
Break Detection .....	6-3
MERGE Examples .....	6-3
Extended Record Converter (OLDRE) .....	6-4
OLDRE Extended Records .....	6-4
Calling OLDRE .....	6-5
OLDRE Operation .....	6-5
Translation Results .....	6-6
Program Restrictions .....	6-7
Macro .....	6-7
Pascal .....	6-7
FORTRAN .....	6-7
OLDRE Error Messages .....	6-8

File Comparison (SCOM) .....	6-9
Calling SCOM .....	6-9
SCOM Options .....	6-10
F1, F2, BO .....	6-11
NN .....	6-11
NH .....	6-11
NT .....	6-11
TB .....	6-11
IB .....	6-11
D<x> .....	6-11
C<x> .....	6-12
IT .....	6-12
IC .....	6-12
ET, ER .....	6-12
BR, BB .....	6-13
TC .....	6-13
The Compare Operation .....	6-13
Returned Values .....	6-14
Status Interrogation .....	6-14
SCOM Examples .....	6-15
SCOM Error Messages .....	6-22

## Chapter 7 System Setup Utilities

Absolute Binary to Memory Image (AB2MI) .....	7-1
Calling AB2MI .....	7-1
AB2MI Operation .....	7-2
Break Detection .....	7-2
AB2MI Error Messages .....	7-2
Initialize BOOTEX File (INSTL) .....	7-3
INSTL Overview .....	7-3
Calling INSTL .....	7-4
INSTL Operation .....	7-6
Successful completion is indicated by the message: .....	7-6
INSTL Error Messages .....	7-7
Bootable System Installation (FPUT) .....	7-8
Running FPUT .....	7-8
FPUT Operation .....	7-9
Copy System (CSYS) .....	7-10
Calling CSYS .....	7-10
CSYS Operation .....	7-11
CSYS Examples .....	7-12
Loading CSYS .....	7-13
CSYS Error Messages .....	7-13
Memory Image to Absolute Binary (MI2AB) .....	7-14
Calling MI2AB .....	7-14
Break Detection .....	7-14
MI2AB Error Messages .....	7-15

## Chapter 8 NOTIFY and PRINT Utilities

Using the NOTIFY Utility .....	8-1
Calling NOTIFY .....	8-1
Sending a Message .....	8-1
Turning Notification On or Off .....	8-2
Defining Aliases for Notification .....	8-2
Alternate Logons for a User .....	8-3
Group Distribution Lists .....	8-3
Hosts in a DS Network .....	8-3
Alias Definition Format .....	8-3
Using the PRINT Utility .....	8-5
Calling PRINT .....	8-5
PRINT Options .....	8-5
+? .....	8-7
+A .....	8-7
+B .....	8-7
+C .....	8-7
+F .....	8-8
+I .....	8-8
+M .....	8-8
+N .....	8-9
+O .....	8-9
+P .....	8-9
+Q .....	8-9
+S .....	8-10
+W .....	8-10
+X .....	8-10
The PRINT Operation .....	8-11
PRINT Messages .....	8-12
PRINT Examples .....	8-13
Loading PRINT .....	8-15

## Chapter 9 System Status Utilities

Display CPU Usage (METER) .....	9-1
Sorting and Displaying Process Information .....	9-1
Calling METER .....	9-2
Example of METER Output .....	9-4
Loading METER .....	9-4
Show the Status of System Available Memory (SAM) .....	9-5
Calling SAM .....	9-5
Running SAM without the AL Parameter .....	9-5
Running SAM with the AL Parameter .....	9-6
Returned Values .....	9-6
Loading SAM .....	9-7
Serial Port Analyzer (SPORT) .....	9-8
Calling SPORT .....	9-8
SPORT Operation .....	9-8
SPORT Examples .....	9-9
Including SPORT in a User Program .....	9-10

Loading SPORT .....	9-11
---------------------	------

## Chapter 10 Multiuser Accounting Utilities

Reset Multiuser Accounting Information (RINFO) .....	10-1
Calling RINFO .....	10-1
Loading RINFO .....	10-2
RINFO Protection .....	10-2
Returned Values .....	10-2
Show Multiuser Accounting Information (SINFO) .....	10-3
Calling SINFO .....	10-3
Loading SINFO .....	10-3
Returned Values .....	10-3
Terminate a Session (KILLSES) .....	10-4
Calling KILLSES .....	10-4
Loading KILLSES .....	10-4
KILLSES Protection .....	10-4
Returned Values .....	10-4
KILLSES Examples .....	10-5
Modify or List Session LU Access Tables (SESLU) .....	10-6
Calling SESLU .....	10-6
Loading SESLU .....	10-6
SESLU Protection .....	10-6
Returned Values .....	10-7
SESLU Examples .....	10-7



## Chapter 11 CALLS and CALLM Utilities

CALLS Online Help Facility .....	11-1
Invoking the CALLS Facility .....	11-1
CALLS Catalog File .....	11-2
CALLS Directives .....	11-2
Relating Topics to Other Topics .....	11-3
Index File .....	11-5
CALLM Utility .....	11-6
Invoking the CALLM Utility .....	11-6
.Include Directive .....	11-7

## Appendix A File Manager (FMGR)

FMGR Control .....	A-2
List Device .....	A-2
Log Device .....	A-2
Severity Code .....	A-2
FMGR Errors .....	A-2
FMGR Control Commands .....	A-2
Explain Error Codes (??) .....	A-3
?? Command Examples .....	A-3
Display or Change List Device (LL) .....	A-4
LL Command Examples .....	A-4

Change Log Device (LO) .....	A-5
Display or Set Severity Code (SV) .....	A-5
SV Command Examples .....	A-6
Disk Manipulation .....	A-7
Logical vs Physical .....	A-7
Disk Logical Units and LU Numbers .....	A-7
Cartridges and Cartridge Reference Numbers .....	A-8
Configuration of Logical Units/Cartridges .....	A-8
Mounting and Dismounting Cartridges .....	A-9
Cartridge Directory .....	A-10
Cartridge File Directory .....	A-10
Cartridge Initialization .....	A-11
Master Security Code .....	A-12
Re-initializing a Cartridge .....	A-12
Purging Files on a Cartridge During Re-initialization .....	A-13
Packing a File Cartridge .....	A-14
Transferring Files Between Disk Cartridges .....	A-15
Disk Manipulation Commands .....	A-15
List Cartridge Directory (CL) .....	A-16
CL Command Example .....	A-17
Copy Files (CO) .....	A-17
CO Command Options .....	A-18
CO Command Option Examples .....	A-19
CO Command Examples .....	A-20
CO Command Termination .....	A-22
Dismount File Cartridge (DC) .....	A-23
DC Command Error Handling .....	A-23
List File Directory (DL) .....	A-24
DL Command Examples .....	A-25
Initialize File Cartridge (IN) .....	A-27
IN Command Error Handling .....	A-30
Mount File Cartridge (MC) .....	A-30
MC Command Error Handling .....	A-31
Pack File Cartridge (PK) .....	A-31
PK Command Error Handling .....	A-32
File Manipulation .....	A-33
Records and File Types .....	A-33
Scratch Files .....	A-34
Accessing a Disk File .....	A-34
Creating a File .....	A-34
Purging Files .....	A-35
Storing Data on a Device or New File .....	A-36
Listing the Contents of a File .....	A-36
Renaming Files .....	A-36
File Manipulation Commands .....	A-36
Create a File (CR) .....	A-37
CR Command Examples .....	A-39
CR Command Error Handling .....	A-39
Dump Data to a Device or Existing File (DU) .....	A-40
DU Command Examples .....	A-41
List Contents of a File (LI) .....	A-42
LI Command Example .....	A-43
Purge a File (PU) .....	A-44

PU Command Examples .....	A-44
Rename a File (RN) .....	A-45
RN Command Examples .....	A-45
Store Data on a Device or New File (ST) .....	A-45
ST Command Examples .....	A-48
Transfer Files .....	A-51
Creating a Transfer File .....	A-51
G-Type Global Parameters .....	A-51
P-Type Global Parameters .....	A-53
TR and Related Commands .....	A-55
Calculate Global Parameter (CA) .....	A-56
CA Command Examples .....	A-57
Display Parameters (DP) .....	A-58
DP Command Examples .....	A-58
Conditional Skip (IF) .....	A-59
IF Command Examples .....	A-60
Pause and Send Message (PA) .....	A-61
PA Command Example .....	A-61
Set Global Parameter (SE) .....	A-61
SE Command Examples .....	A-62
Transfer Control to a File or Device (TR) .....	A-62
TR Command Examples .....	A-63
Device Manipulation .....	A-64
Calling FMGR and COMND .....	A-64
Device Manipulation Commands .....	A-66
Display/Modify Buffer Limits (BL) .....	A-66
BL Command Examples .....	A-67
Make Device Unavailable (DN) .....	A-68
DN Command Example .....	A-68
Other FMGR Commands .....	A-69
COMND .....	A-70

## Appendix B

### CS/80 Exerciser Utility (EXER)

Introduction .....	B-1
Getting Started .....	B-1
Loading the program .....	B-2
Using the Exerciser .....	B-2
Selected Command Descriptions .....	B-6
Error Handling .....	B-9
EXER and CS80 Tape Drives .....	B-9
EXER and Cache Disks (HP 793xXP) .....	B-9
EXER and SubSet 80 Disks .....	B-10

## List of Illustrations

Figure 2-1	Example Directory Tree Structure .....	2-26
Figure 2-2	FST Format and Header Basics .....	2-41
Figure 2-3	Example of Directory List Format .....	2-79
Figure 2-4	TF Tape Format and Header Basics .....	2-107
Figure 2-5	Example of FC Command Summary Listing .....	2-112
Figure 2-6	Example of Tape Directory List Format .....	2-125
Figure 5-1	Typical Track Format .....	5-46
Figure A-1	Cartridge Structure .....	A-11
Figure A-2	Disk Structure Before and After Packing .....	A-14
Figure A-3	Initialized Cartridge Structure .....	A-29
Figure A-4	Global Parameters .....	A-53

## Tables

Table 2-1	Disk Backup Utilities .....	2-1
Table 2-2	File Interchange Utilities .....	2-3
Table 2-3	FST Commands Summary .....	2-7
Table 2-4	FST Command Options Summary .....	2-20
Table 2-5	Reserved Character Replacements for TAR Archive Files .....	2-36
Table 2-6	Reserved Character Replacements for FMGR Files .....	2-38
Table 2-7	TF Commands Summary .....	2-55
Table 2-8	TF CO Command Options Summary .....	2-58
Table 2-9	FC Commands Summary .....	2-111
Table 2-10	FC CO Command Options Summary .....	2-117
Table 2-11	LIF Commands Summary .....	2-155
Table 3-1	ASAVE Commands Summary .....	3-4
Table 3-2	SA Command Options Summary .....	3-7
Table 3-3	ASAVE Header Record Format .....	3-14
Table 3-4	ASAVE Example .....	3-17
Table 3-5	ARSTR Commands Summary .....	3-19
Table 3-6	RE Command Options Summary .....	3-22
Table 3-7	ARSTR LU Restore Output .....	3-29
Table 3-8	DSAVE Commands Summary .....	3-38
Table 3-9	LH Command Example .....	3-40
Table 3-10	SA Commands Option Summary .....	3-42
Table 3-11	DRSTR Commands Summary .....	3-53
Table 3-12	RE Command Options Summary .....	3-57
Table 4-1	MPACK Options Summary .....	4-12
Table 5-1	RTE-A Formatting Utilities .....	5-1
Table 5-2	FORMC Commands Summary .....	5-9
Table 5-3	FORMF Commands Summary .....	5-25
Table 5-4	FORMT Commands Summary .....	5-32
Table 5-5	Responses to System Prompts .....	5-49
Table 5-6	ERTSH Commands Summary .....	5-56
Table 6-1	MERGE Options Summary .....	6-2
Table 6-2	SCOM Options Summary .....	6-10
Table 8-1	PRINT Options Summary .....	8-6
Table 8-2	&FFL Variables .....	8-15
Table 9-1	METER Commands Summary .....	9-3

Table A-1	FMGR Control Commands .....	A-3
Table A-2	Disk Manipulation Commands Summary .....	A-16
Table A-3	CO Command Options Summary .....	A-18
Table A-4	Record and File Type Equivalences .....	A-33
Table A-5	File Manipulation Commands Summary .....	A-37
Table A-6	Transfer Commands Summary .....	A-55
Table A-7	FMGR/CI Commands .....	A-69
Table A-8	COMND Commands .....	A-70





# Introduction

---

RTE-A provides utilities that perform a variety of functions. This chapter briefly describes the use of each utility. For a detailed description of the utilities, see the appropriate chapter in this manual.

## Backup and File Interchange Utilities

**File Storage to Tape (FST)** is a high performance backup utility that has streaming capability and supports DDS media, magnetic tapes, and CS/80 cartridge tapes. FST also allows the interchange of files between HP 1000 and UNIX-based systems.

**Tape Filer (TF)** is used to copy files between disk and tape media, either disk-to-tape or tape-to-disk. TF is similar to FC (File Copy), but it works with all files, whereas FC works only with FMGR files. TF also allows the interchange of files between HP 1000 and UNIX-based systems.

**File Copy (FC)** can be used to transfer FMGR files between disk cartridges and tape media, either disk-to-disk, disk-to-tape, or tape-to-disk.

**Logical Interchange Format HP Systems File Copy (LIF)** allows interchange of files between HP 1000 and other Hewlett-Packard systems. It translates files from the standard RTE FMP format to a standard logical interface format and vice versa. LIF is similar to the command interpreter (CI); however, it is not intended to be a duplicate of file management capabilities.

All these utilities are described in detail in Chapter 2.

## Physical Disk Image Backup Utilities

**Save (ASAVE) and Restore (ARSTR)** are used to transfer data between disks and magnetic tape or CS/80 cartridge tape drives (CTD). ASAVE allows you to back up large, contiguous areas of a disk to tape, regardless of the disk's content. ARSTR allows you to restore previously saved disks from tape.

**Disk-to-Disk Copy (COPYL)** is an offline utility that copies the entire contents of one disk LU to a second disk LU. This utility lets you make copies of disks without using FMP.

**Save (DSAVE) and Restore (DRSTR)** are used to transfer data between fixed and removable disks. DSAVE saves selected LUs or an entire disk drive unit to removable disk. The removable disk

media include 3, 5, and 8 inch floppies and disk platters such as the HP 7906H disk. DRSTR restores the saved data from the removable media to fixed disk. The physical backup utilities are not compatible with previous RTE physical backup utilities and therefore cannot be used for interchange.

These utilities are all fully discussed in Chapter 3.

## File System Utilities

**File System Conversion (FSCON)** converts the directory structure of an FMGR cartridge to a hierarchical directory entry. Once converted, FMGR files have the characteristics of the hierarchical file system.

**File System Pack (FPACK)** rearranges the files on a file system volume to pack them together more tightly. This allows more or larger files to be created.

**File Compacting and Disk Pack (MPACK)** compacts files and packs a disk, removing unused blocks and extents from the files and arranging all of the disk's free space as one large block.

**Report Disk Free Space (FREES)** scans the free space table on the hierarchical file system disks and reports on the total amount of free space and the largest amount of free space.

**Report File Space by Owner (FOWN)** scans files according to a user-specified mask and displays the total disk space used by each owner. This utility can be used to identify users with excess disk space or files using excess space.

**File System Verification (FVERI)** scans the directories and table structure of a hierarchical file system disk LU and reports inconsistent data. Extent pointers, data pointers, non-negative file types, and so on, are checked. This utility can be used after a system crash to verify the file system is still intact.

All the file system utilities are described in Chapter 4.

## Formatting Utilities

**CS/80 Disk Formatting (FORMC)** is a maintenance program, provided in both online and offline forms, for CS/80 disk and tape drives.

**Disk Formatting (FORMF)** is the online maintenance program for floppy and standalone 5-1/4 inch mini-Winchester disk drives.

**Offline Disk Formatting and Initialization (FORMT)** is an offline utility that formats a floppy disk, identifies bad tracks on the disk, initializes disk LUs, spares bad tracks, verifies a disk, or reformats a disk.

**Initialize and Spare (FORMA)** is an online utility that provides initializing and sparing functions for Micro/1000 systems with internal disks connected via the 12022 disk controller.

**Error Rate Test (ERTSH)** is an online utility that tests Micro/1000 internal hard disks.

The formatting utilities are fully described in Chapter 5.

## File Manipulation Utilities

**Concatenate Files (MERGE)** collects two or more input files and combines them in a single composite output file.

**Extended Record Converter (OLDRE)** is a standalone utility that provides backward compatibility with loaders and generators that do not accept extended record formats.

**Compare Files (SCOM)** compares two input files and identifies their differences.

All of these utilities are discussed in Chapter 6.

## System Setup Utilities

**Absolute Binary to Memory Image (AB2MI) and Memory Image to Absolute Binary (MI2AB)** utilities convert absolute binary files to memory image files and vice versa.

**Initialize BOOTEX (INSTL) File** allows you to initialize the boot extension (BOOTEX) file for an RTE-A disk LU.

**Bootable System Installation (FPUT)** installs bootable systems and diagnostics in the space on the file system volume that was reserved by a CI command.

**Copy System (CSYS)** copies memory image files from a CS/80 disk to a CTD tape in a memory-based system.

The system setup utilities are discussed in Chapter 7.

## NOTIFY and PRINT Utilities

**NOTIFY** sends a one-line message to another user's terminal.

**PRINT** prints one or several files on any supported system printer. This utility simulates spooling, ensuring that no other output lines are interleaved with PRINT jobs.

Information about using the NOTIFY and PRINT utilities is presented in Chapter 8.

## System Status Utilities

**Display CPU Usage (METER)** displays information about system and user processes.

**System Available Memory (SAM)** prints out information about the status of System Available Memory and Extended System Available Memory.

**Serial Port Analyzer (SPORT)** displays the status of serial ports.

These utilities are discussed in detail in Chapter 9.

## Multiuser Accounting Utilities

**Reset Multiuser Accounting Information (RINFO)** resets the multiuser accounting information found in the user configuration file for the specified users.

**Show Multiuser Accounting Information (SINFO)** displays the multiuser accounting information found in the user configuration file for the specified user.

**Terminate a Session (KILLSES)** terminates the session, shuts down all associated programs, logs off the user, closes associated spool files, and releases the user ID entry.

**Modify or List Session Bit Maps (SESLU)** modifies and lists session bit maps.

See Chapter 10 for a full discussion of these utilities.

## Keyword Indexed Help Utilities

**CALLS** 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.

**CALLM** builds a single compressed input text file for use by the CALLS program.

See Chapter 11 for more information about the CALLS and CALLM utilities.

## File Manager Control

**File Manager (FMGR)** is a program that manipulates files on FMGR directories. FMGR only works with FMGR files. Appendix A contains information on FMGR control, commands to change the way FMGR operates; disk manipulation, commands to handle FMGR cartridges; file manipulation, commands to copy, purge, and list FMGR files; transfer files, commands to run FMGR non-interactively; and the COMND program, the compact version of FMGR.

## CS/80 Exerciser Utility (EXER)

**CS/80 Exerciser Utility (EXER)** is an online program used to diagnose and troubleshoot CS/80 disk drives on HP 1000 systems. See Appendix B for more information about the EXER utility.

## Backup and File Interchange Utilities

---

RTE-A provides several utilities for file and disk backup and for file interchange with other Hewlett-Packard and non-Hewlett-Packard systems. This chapter discusses these utilities and their relationships to each other and suggests their most effective use.

### Backup Utilities

You may use several utilities to back up and verify data from a disk drive to a tape drive (either magnetic tape, DDS media, or CS/80 cartridge tape). Backup utilities include FST, TF, FC, ASAVE/ARSTR, and COPYL .

RTE-A supports two modes of backup, file backup and physical disk image backup. Table 2-1 shows the utilities used for each mode.

**Table 2-1. Disk Backup Utilities**

	<b>File Backup from Disk</b>	<b>Physical Image Backup from Disk</b>
to Cartridge Tape to Magnetic Tape to Disk to DDS Media	FST, TF, FC FST, TF, FC CI, FST FST, TF	ASAVE/ARSTR ASAVE/ARSTR COPYL ASAVE/ARSTR

File backup allows online backup and restoration to and from disks. It allows selective file backup and selective file restore from a larger backup. Selective file backup can be relatively fast, since you need not back up the entire disk LU. However, you may find that total file backup of a full disk is slower than physical image backup due to the file system overhead. In addition, offline restoration is not possible since file backup requires a functioning, disk-based operating system.

Physical backup allows offline restoration and, in the case of a total disk backup, is faster than file backup. However, you cannot restore files selectively using physical backup but must restore the entire disk, which wipes out any previous contents.

FST and TF are used to back up files from disk to tape and restore files from tape to disk (some restrictions apply to FMGR files). These utilities 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. FST can also be used to back up and restore files to and from archive files. Refer to the individual sections in this chapter on FST and TF for a complete description of each utility.

FC is used to back up files from disk to tape and restore files from tape to disk. Refer to the section in this chapter on FC for a complete description of this utility.

ASAVE and ARSTR can be used to copy an RTE-A disk LU to magnetic tape or CS/80 cartridge tape, regardless of file structure. ASAVE lets you save to tape; ARSTR does the restoration. ASAVE and ARSTR let you append new disk LU backups to previous backups and can be used in a memory image system for offline restores and saves. See Chapter 3 for a complete description of the ASAVE and ARSTR utilities.

COPYL can be used to copy the data on one disk LU to another disk LU of similar type, regardless of file structure. See Chapter 3 for a complete description of the COPYL utility.

All of the above utilities support a Verify option that lets you check that data was properly saved.

LIF is used primarily to interchange files between Hewlett-Packard systems but can also provide file backup. Refer to the section in this chapter on LIF for a complete description of this utility.

CI is the RTE-A command interpreter and can be used to copy files on disk. See the *RTE-A User's Manual*, part number 92077-90002, for a complete description of CI.

## File Interchange on RTE-A

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 2-2 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 2-2. 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	-
<b>Notes:</b> (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			



## 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, DDS media, 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 256 characters, the limit imposed by CI.

Use a vertical bar (|) to separate commands in the runstring. 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 2-3 summarizes the FST executable commands.

**Table 2-3. FST Commands Summary**

Commands	Description
<b>Information Commands</b>	
HElp (or ?) SHow	Provide a summary of commands and syntaxes Display the DF, MT, TI, SC, LL, and option settings
<b>Backup/Restore and Related Commands</b>	
<b>BA</b> ckup                    mask <b>DF</b> Directory Files        file_desc <b>GO</b> Begin Backup/Restore <b>RE</b> store                    mask <b>SC</b> Select Comment File    filename <b>TA</b> UNIX TAR Format <b>Ti</b> tle                        title <b>UN</b> select                    mask	Select files to back up Specify a non-default directory file Begin executing backup/restore Select files to restore Select the tape's comment file (backup only) Select UNIX TAR archive format Specify a title for the archive (backup only) Unselect files
<b>Listing Commands</b>	
<b>DL</b> Directory List            mask <b>LC</b> List Comment File <b>LH</b> List Header <b>LI</b> st Selected Files        mask <b>LL</b> Select Log Device/Files device/file <b>LN</b> List Non-Selected Files mask	Display the archive's directory file List the archive's comment file List the archive's header List the files selected for backup/restore Set log device or file for FST activities List the non-selected files (restore only)
<b>Tape LU Control Commands</b>	
<b>MT</b> Specify Tape LU        tape_LU/ or archive file        filename <b>NE</b> xt                        append_# <b>PO</b> sition                    append_# <b>PR</b> evious                    append_# <b>SD</b> Set Tape Density        density <b>SE</b> cure	Set the magnetic tape LU or archive file Advance the tape to another append Position the tape to a specific append Rewind the tape to a previous append Set the tape density (HP 7974/7978 only) Lock the tape LU and check the tape status or open archive and check the file's status
<b>Transfer and Exit Commands</b>	
<b>EX</b> it <b>RU</b> n                         program <b>TR</b> Transfer to             filename Command File / Command Stack	Exit FST Run an external program Begin executing a transfer file 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.

Archive files in TAR format are written with a TAR blocking factor of 20.

## **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 /my_files/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 mag tape streaming 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 2-4 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]
```

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 2-4. FST Command Options Summary**

Options	Description
<b>A</b> ppend	Append this backup to the data already on the tape.
<b>B</b> rief	Only show errors and status messages.
<b>C</b> lear	Clear the disk file's backup bits.
<b>D</b> uplicate	Replace duplicate files.
<b>F</b> aulty	Restores files from a partially overwritten tape.
<b>I</b> nhibit	Inhibit the tape rewind between backups.
<b>K</b> eep	Keep tape online when backup/restore is complete.
<b>L</b> ock	Lock any disk LUs used.
<b>M</b> inDir	Minimize the FST directory file size during restores.
<b>N</b> ormal	Backup symbolic links as normal files (follow links).
<b>O</b> riginal	Restore files to their original main size.
<b>P</b> urge	Purge the disk files after backing up the files.
<b>Q</b> uiet	Report messages only to the log device/file.
<b>R</b> wndOff	Rewind and go offline at exit.
<b>S</b> rchApp	Search through appends during RESTORE.
<b>U</b> ppdate	Replace duplicate file if file has been updated.
<b>V</b> erify	Verify the files during the backup/restore.
<b>W</b> hole	Back up all the blocks reserved for the file.
<b>Y</b> es	Write over the tape without asking.
<b>Z</b>	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 mag 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; 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.

### **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 also initializes 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
```

has the effect of reinitializing grouping, and the previous RE commands are 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 is 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 up 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 some 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.)

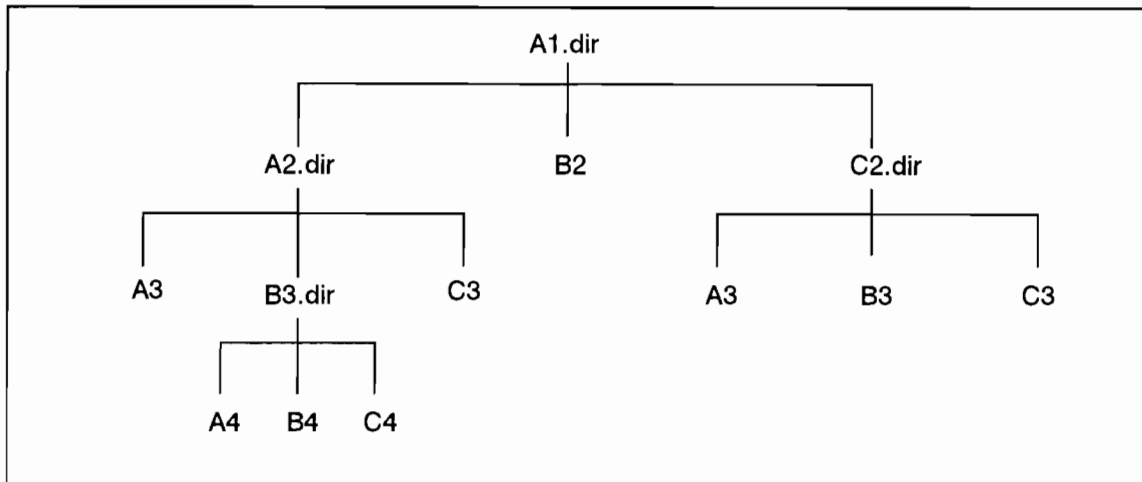
The S qualifier searches down through the entire directory structure. (See Example 3 under Backing Up in the next section.)

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 copy command, CO. 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.



**Figure 2-1. Example Directory Tree Structure**

**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 affect 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 command, CO. 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 bit in a file's directory entry is 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:

```
Friday:  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 the tape movement from one append to another.

The Update option restores a file on the tape only 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 simply 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 a first set of files and copy them to tape using GO, then select a second set, without exiting FST, and copy them to the tape using GO. Thus, the first files are included with the second files on the second copy, because the first files were 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 having to mount 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. The message, “Enter ‘GO’ when a new tape is online/loaded” is then displayed. (Online refers to 1/2 inch magnetic tapes. Loaded refers to CTD and DDS media.)

If Keep is OFF, the tape is taken offline/unloaded after being filled. The message, “FST will continue when the tape is ready (online/loaded)” is then displayed.

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 go 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 and FST 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 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 Table 2-5.



**Table 2-5. Reserved Character Replacements for TAR Archive Files**

Reserved Character	Replaced By
'	—
.	—
,	—
@	*
-	?
:	
>	^
[	(

Note that not all periods are replaced. If the last period in a name results in a legal RTE type extension, it is preserved. File names that begin with numbers are prepended with the underscore character ('\_').

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 displays the original and new names for any file that is 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 whatever files 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 a 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.

## 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 2-6. 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 CS/80 cartridge tape drives, DDS tape drives, 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 10240-byte 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 2-2.

**FST Format:**

Archive Header	Comment File Data (opt)	FST Directory File Header	FST Directory File Data
----------------	-------------------------	---------------------------	-------------------------

Header	Data	Header	Data	...	FileMark	FileMark
--------	------	--------	------	-----	----------	----------

**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 2-2. 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 a FMGR cartridge.

The file >FS000 must be located on the global directory /CATALOGS. If /CATALOGS does not exist, >FS000 can be placed on a 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 FSTRUN, 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 in order 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 SrchApp option enables a group restore and 'RE' selections are stored in a scratch file for use during the append search.

**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 parent program, FST, and the child 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.

**diskerror: 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 SrchApp option enables a group restore and 'RE' selections are stored in a scratch file for use during the append search.

**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 a 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 which 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 in a 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, CS/80 cartridge tape, or DDS media.

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-A 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>
```

<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 2-7 summarizes the commands, which are described in the following sections.

**Table 2-7. TF Commands Summary**

Commands		Description
<b>Information Command</b>		
Help or ?		Provide a summary of commands and command syntaxes
<b>Copy and Related Commands</b>		
<b>CO</b> py Files	filename	Copy files as specified by parameters
<b>DE</b> fault	file_desc	Set default source, destination, and options for subsequent COPY commands
<b>GR</b> oup	commands	Used for grouping more than one copy command
<b>EG</b> End Group		End Group into a single COPY operation
<b>AG</b> Abort Group		Abort Group
<b>TI</b> tle	title	Establish title to be used in tape header file
<b>Listing Commands</b>		
<b>LH</b> List Header		List header file from TF tape
<b>LL</b> Set List Device	device/file	Set list device or file for LH and DL or File
<b>DL</b> Directory List	mask	Compile directory list of TF tape
<b>Transfer and Exit Commands</b>		
<b>TR</b> ansfer	filename	Transfer to/return from TF command file
<b>EX</b> it		Exit TF
<b>Comment Command</b>		
*		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-A 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 2-8 summarizes the CO command options.

**Table 2-8. TF CO Command Options Summary**

<b>Option</b>	<b>Description</b>
<b>Append</b>	Adds files to end of current tape
<b>Brief</b>	Suppresses logging of each file name as it is copied
<b>Clear</b>	Clears the backup bit for each file copied to tape
<b>Duplicate</b>	Replaces existing files with files restored from tape
<b>Ignore</b>	Causes errors in the tape header to be ignored
<b>Keep</b>	Prevents TF from taking the tape unit offline
<b>N (Inhibit)</b>	Stops text file conversion from UNIX to FMP
<b>Update</b>	Replaces duplicate files if the existing file has an earlier update time
<b>Verify</b>	Compares files on disk and tape after they are copied
<b>X (UNIX)</b>	Converts type 4 files to UNIX text file format when files are copied to tape
<b>Yes</b>	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 line-feed 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 a 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. Because 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. Because 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.

If you do not specify a destination when you restore files, they are restored to the same directory from which they were backed up. 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
```

Here 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
```

Here the A mask qualifier is used 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-A 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 node >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>	As defined for the CO command, except that a
<destination>	list of masks may not be used in the source parameter.
<options>	

### 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(s) 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 :>sourcename :[destuser]
de :[sourceuser]>sourcename :>destname
```

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
```

**Figure 2-3. Example of Directory List Format**

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 /PROGRAM.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, otherwise 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-A Programmer’s Reference Manual*, part number 92077-90007.

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 ASAVE/ARSTR 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 do not 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 4, and to the discussion of PK in Appendix A in this 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 line-feed 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 line-feed 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 line-feed characters (UNIX “newlines”) so that the files are usable as text files on a UNIX operating system. If the files contain any line-feed 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 line-feed 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-A Programmer's Reference Manual*, part number 92077-90007, 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: eq
```

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 and 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 thirddtype2} :::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 line-feed 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 line-feed 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/c.c.	aaa/bbb/c.c_
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.

## File 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.

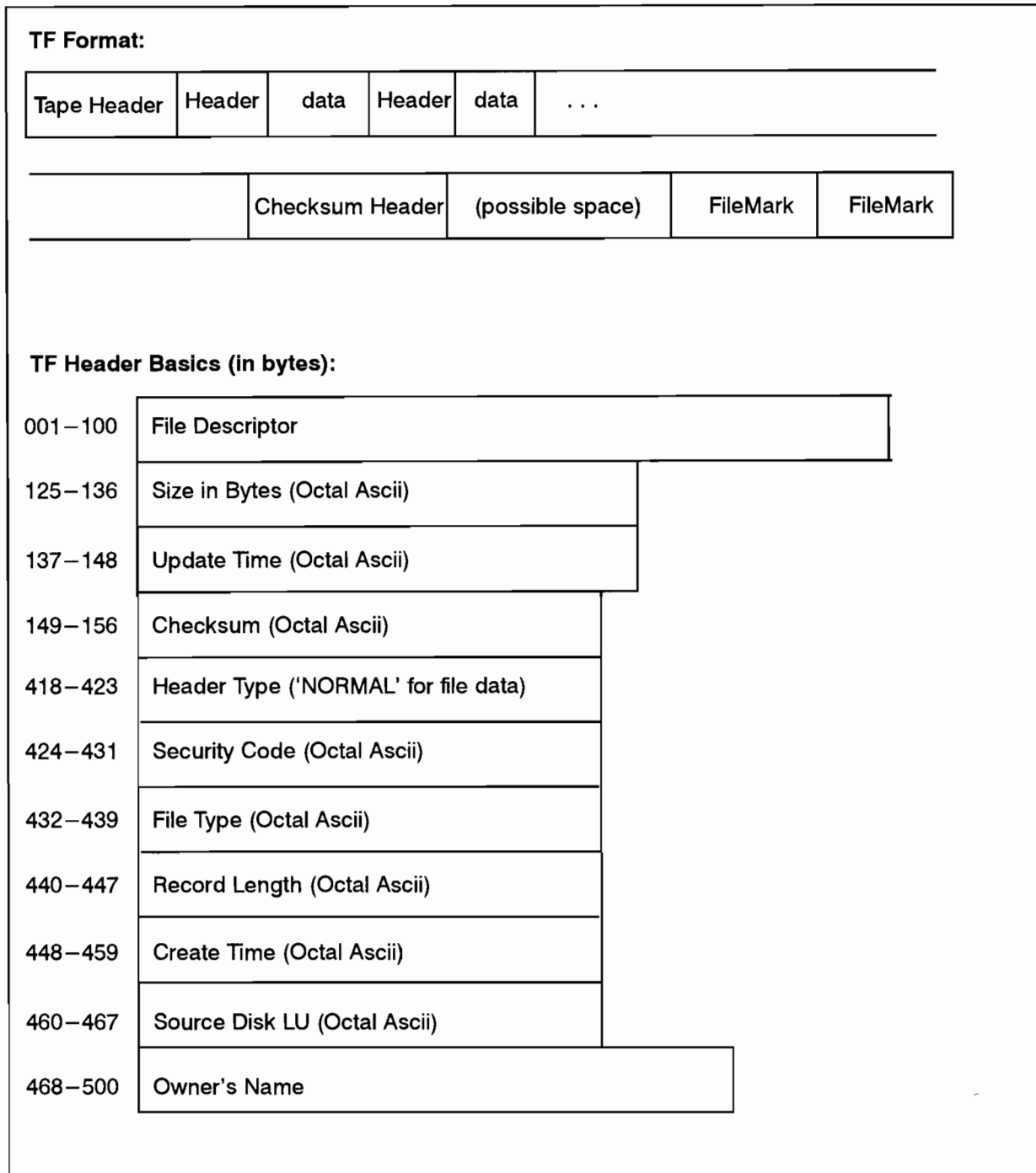
The tape title is used as the path name for the header file. If you do not specify a title for the tape header, the default title (the CO command itself) will contain embedded spaces and could contain slashes. This could result in extraneous files being created by TAR.

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.



**Figure 2-4. TF Tape Format and Header Basics**



## 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 command file. Any single FC command can be entered in the runstring. When you run FC from a command file, the FC transfer (TR) command lets you specify a command 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: <command>
FC: <command>
.
.
.
FC: ex
:
```

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>
```

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 2-9 summarizes the commands, which are described in the following sections.

Table 2-9. FC Commands Summary

Commands	Description
<b>Information Command</b>	
?	Provides a summary of commands and command syntaxes
<b>Configuration Commands</b>	
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
<b>Copy and Related Commands</b>	
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
<b>Listing Commands</b>	
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
<b>Transfer and Exit Commands</b>	
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
<b>Comment Command</b>	
*	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 shown below 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.

```
----- FC commands -----  commands may be abbreviated to 2 chars

COPY, srce, dest, opts, [file1], [file2], [msc] copy files
DEFAULT , srce , dest , opts          set defaults for COPY command
GROUP / EG / AG                       begin / end / abort GROUP of COPY commands
LL , namr                              set list file/device (dash means log device)
DL , srce , [msc] , opts              list tape directory (srce = -tlu or -tlu{namr})
CL , [-tlu] , options                 list local cartridge list or tape cartridge list
CLAL                                  list global cartridge list (RTE-IVB/6 only)
LH , -tlu , opts                      list tape header file
LC , -tlu , opts                      list tape comment file
ECHO [, ON/OFF]                       turn ON/OFF cmd echo to list device (default ON)
TITLE , title                         set tape title (for subsequent COPYs to tape)
CF , comment-file-namr                set comment file (for subsequent COPYs to tape)
TR, namr                              transfer to cmdfile/device(dash means log device)
TR                                    return from command file/device
EXIT                                  exit FC
ABORT                                  abort FC (same as EX except if copy group active)
SCRATCH , cartridge                   set cartridge that FC will use for scratch files
* comment                             command line starting with * treated as comment
  HELP [, key [, lu                  get help, RTE-IVB/6 only (useful for FMGR errors)
? , <command>                         list info about particular command(incl. options)
? , <option>                           list info about particular option (all commands)
```

Figure 2-5. Example of FC Command Summary Listing

## Abort (AB)

Purpose: Stops FC execution immediately.

Syntax: AB

## Name Comment File (CF)

Purpose: To specify 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. Any comment file records longer than 128 words are 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)

Purpose: Displays the list of all cartridges you can access.

Syntax: CL, -tlu[,K]

-tlu Lists the cartridges 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:

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.



file1  
file2

The optional limits on the range of files to be copied from the 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 “Copy 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.

## 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 2-10 summarizes the CO command options, followed by a discussion of each option.

**Table 2-10. FC CO Command Options Summary**

Options	Description
<b>Brief, Full Status</b>	Displays status of namrs as each file is copied, when an error occurs (B) or for successfully copied files as well (F)
<b>Clear Destination Disk</b>	Clears destination disk before the copy begins
<b>Duplicate Files</b>	Purges original file and replaces it with the duplicate
<b>Eliminate Extents</b>	Combines all extents into the main extent
<b>Ignore Data Errors</b>	Overrides checksum or verify errors and allows the file to be copied
<b>Keep Tape online</b>	Keeps tape unit online in all circumstances
<b>Lock, Open</b>	Locks a cartridge or opens individual files for the copy
<b>Purge Source File</b>	Purges a source file after it is copied
<b>Single Volume Copy</b>	Lets you mount a single tape volume set
<b>Tape Length Display</b>	Displays required tape length for copying specified files
<b>Unused Space</b>	Lets FC use more space on the destination medium than required for files being copied, to speed up copying
<b>Verifying Data</b>	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:** DL 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

```

name	crn	LU	type	extent size	record size	#extents log	security phy	code	diskfile 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 Tl 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 2-6. Example of Tape Directory List Format

## Echo Command (EC)

Purpose: To have commands echoed 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 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)

### 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: Lets you transfer control to and return from a command file.

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 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 write 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:

Hdr	Comment	Dir.	EOF	data	data	EOF	...
-----	---------	------	-----	------	------	-----	-----

FC format on mag tape:

Hdr	EOF	Cmt fil	EOF	Dir.	fil	EOF	fil	EOF	...	EOF	EOF	...
-----	-----	---------	-----	------	-----	-----	-----	-----	-----	-----	-----	-----

## 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 may 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 specify 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

Use the LINK command file #FCA to load the utility. The load process also requires copying the FC help file FCHLP 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:

```
:TR,FLOPUP,100,-10,XX
:PA,,Put in floppy and type TR
:TR
:MC,-10
:RU,FC,CO,100,-10,!VF,,,XX
FILE1 (First ten files copied)
.
.
FILE11
FMGR-033 (Cartridge-full error)
copy terminated
:DC,-10
:IF,-1,NE,-1,5
:PA,,Put in next floppy and type TR
:TR
:MC,-10
: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
:DC,-10
:IF,-1,EQ,-1,-5
:PA,,Put in next floppy and type TR
:TR
:MC,-10
:RU,FC,CO,100,-10,!VF,FILE21,,XX (Next file namr in 10G)
FILE21 (Copy remaining files)
.
.
FILE25
:DC,-10
:IF,0,EQ,-1,-5 (Test for cartridge-full error)
:TR
: (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 what 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-A 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>::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>::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 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 filename 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, use the FC command, SC, 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 the sections “Backup Utilities” and “File Interchange on RTE-A” 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 under RTE-A, Rev. 2326 or later, for all supported HP disks using standard HP disk drivers, including mini-floppies, flexible disks, model 7906 disk drive, CS/80 disk drives (models 7908/7911/7912), 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.

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 2-11 summarizes the LIF commands, which are described in the following sections.

**Table 2-11. LIF Commands Summary**

Commands	Description
<b>Information Command</b>	
HElp	Displays available LIF commands and their formats
<b>Configuration Commands</b>	
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
<b>Copy Command</b>	
COpy                lu	Copies all the files from an LIF medium to an RTE disk cartridge
<b>Listing Commands</b>	
DL Directory List	Displays the files that are on the mounted LIF medium
Llst                file	Copies FMP or LIF files to the list file or device
LL Set List Device    namr	Changes the list file or device
<b>Transfer and Exit Commands</b>	
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            As in the CI CO command. Refer to the *RTE-A User's Manual*, part number 92077-90002, for more information.  
mask

### 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 VOL NUM	TYPE	# SCTS	TRAK/SEC	DATE/TIME	STAMP
AZUZA12345 *000001	00001	000017	0037 013	13:38:42	10/12/80
BACDEF1492 *000001	00000	000012	0037 011	09:19:26	07/14/81

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*  
*filedescriptor*      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 *filedescriptor*]  
FMP                      The file or device to which listings and error messages are  
*filedescriptor*      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  
LIF filedescriptor 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 filename). 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-A Programmer's Reference Manual*, part number 92077-90007. Refer to that manual or the *RTE-A User's Manual*, part number 92077-90002, for error codes and definitions.





# Physical Disk Image Backup Utilities

---

RTE-A 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

Physical backup utilities are usually used to save system areas of disks online, so that you can restore these areas offline if a disk becomes corrupt.

ASAVE and ARSTR are used to back up and restore disk LUs to magnetic tape, CS/80 cartridge tape, or DDS media. COPYL is used to back up one disk LU to another. DSAVE and DRSTR are used to transfer data between fixed and removable disks. Transfers are done on continuous tracks because there is less disk head movement; therefore, the utilities tend to run faster than file-oriented backup utilities.

## Compatibility with Other Disk Backup Utilities

The RTE-A physical backup utilities are not compatible with any previous RTE physical backup utilities (for example, PSAVE and PRSTR) or file backup utilities (for example, TF, FC, and LIF). FST is the recommended utility for transporting data between RTE-A systems. See the sections in Chapter 2 called “Backup Utilities” and “File Interchange on RTE-A” for information about when to use FST and other backup and interchange utilities.

## Compatibility Among Disks

RTE-A physical backup utilities do not use any file structure information. This means that the source and destination disks used in save and restore operations must have the same track size (sectors per track), although the total number of tracks in each disk LU need not match.

If you must transfer data between disk LUs with different track sizes, use one of the file backup utilities (FST, TF, FC, or LIF) to transfer the files individually or in groups, rather than on a track-to-track basis.

The physical backup utilities support all disks that are controlled by RTE-A drivers DD\*30, DDM30, DD\*33, ID\*37, DDQ30, and IDQ35.

## **CS/80 Cartridge Tape Drive**

The CS/80 cartridge tape drive (CTD) is a streaming mode, block record device. CS/80 device driver (DD\*33) disk-caching routines, which control transfers to and from the CTD, provide efficient handling of the cartridge tape.

Data to be transferred are buffered on the integrated disk drive and transferred in a steady stream to the CTD, allowing continuous movement of the tape. Refer to the *RTE-A Driver Reference Manual*, part number 92077-90011, for details on the disk caching scheme.

# ASAVE Physical Backup Utility

ASAVE is used to save data from hard disks either to magnetic tape or CS/80 cartridge tape drives. ASAVE lets you back up large, contiguous areas of disks to tape regardless of the content of the disks (such as files). ASAVE saves one disk LU, a group of disk LUs, or an entire disk unit to tape. You may restore the saved disk areas later, with ARSTR.

---

**Warning** ASAVE and ARSTR must be of the same revision. If they are not, you will receive this error message:

```
Warning - Header record checksums
Error - Save definition records checksums
Restore skipped.
```

---

The following table shows the compatible revisions of ASAVE and ARSTR:

<b>AT REV:</b>	5000	5010	5020	6000
<b>ASAVE</b>	5000	5010	5010	6000
<b>ARSTR</b>	5000	5000	5020	6000

## Calling ASAVE

Invoke ASAVE with the following runstring:

```
CI> [RU, ]ASAVE[ , commands ]
```

Alternatively, you may run ASAVE interactively, entering commands at the following prompt:

```
ASAVE:
```

Separate command parameters either with commas or blanks. You may combine multiple commands either in the runstring or in response to the prompt, separated by vertical bars.

Commands are executed in order, from left to right. If any command causes a serious error, execution terminates with that command (all the preceding commands are executed).

## ASAVE Commands

ASAVE uses commands that let you terminate ASAVE, list all the commands and their parameters, list header information, open a list file or list device, rewind and set tape offline, specify a disk-to-tape save operation, specify a tape LU for ASAVE commands to use, set the title to use in save operations, and specify the user error handling mode. Table 3-1 summarizes the commands, which are described in the following sections.

**Table 3-1. ASAVE Commands Summary**

Commands		Description
<b>Information Commands</b>		
HElp		Lists all ASAVE commands and parameters
<b>Configuration Commands</b>		
REwind		Rewinds and sets tape offline
SAve	LU	Specifies a disk-to-tape save operation
TApe	LU	Specifies the tape LU to use
TitlE	text	Sets the title for subsequent save operations
UE User Error Handling	ON/OFF	Selects the user error handling mode or displays current mode
<b>Listing Commands</b>		
LH List Header	file #	Lists header information from the tape
LL List Device	file descriptor/lu	Opens list file or device to which information is to be logged in addition to the terminal
<b>Exit Commands</b>		
ABort ENd EXit		These three commands terminate ASAVE

## Abort, End, and Exit (AB) (EN) (EX)

Purpose: Any one of these commands terminates ASAVE.

Syntax: AB, EN, or EX

Description:

When the program terminates, the list file or device specified is closed, and all locked LUs are released.

## Help (HE)

Purpose: Displays a list of all ASAVE commands and the parameters of each command.

Syntax: HE, ??, or ?

## List Header (LH)

Purpose: Lists the header information for the specified file number from the tape.

Syntax: LH[,file#]

file#      The file for which the header information is being obtained.

Description:

If the specified file does not exist on the current tape, the header record of the first or last file on the tape is listed. If you do not specify a file number, the header record of the next save file is the default. If positioned at the end-of-data (EOD) record, the header record of the last save file is listed.

---

### Note

If there is more than one ASAVE tape, all file numbers specified are relative to the beginning of the first tape. If the save file for a disk LU is on more than one tape, it has the same file number on both tapes.

---

## List Device (LL)

Purpose: Defines a list file or list device to which error messages and terminal displays are to be sent in addition to the terminal.

Syntax: LL[,file descriptor/lu]

file descriptor    The file to be opened.

lu                  The list device.

## Description:

If the list file you define does not already exist, it is created, and a list device is opened. (A previously opened list file or device is closed before the new list file is created or the list device is opened.) If you do not specify a file descriptor/LU, the currently defined list file or device is closed.

A duplicate file error occurs if you specify a file that already exists. In this case, the command terminates with an error if user error handling (described below) is ON; otherwise, the command is skipped.

If you specify a device that is down and user error handling is ON, you are prompted to correct the problem and bring up the device or enter breakmode. If user error handling is OFF, the command is skipped.

If the LU goes down when you use a list device, and user error handling is ON, you are prompted to correct the problem and bring up the device, or enter breakmode. If user error handling is OFF, you must enter breakmode. When a break occurs, the list file or device is closed, and the command continues to execute.

## Rewind (RW)

Purpose: Rewinds and sets the tape offline. A CTD is also unloaded.

Syntax: RW

## Save (SA)

Purpose: Specifies a disk-to-tape save operation.

Syntax: SA, LU[ , LU[ , ... ] ] [ , <options> ]

<options>      The options available for use with the SA command are described in the next section.

## Description:

The save operation is explained in detail in the section, “The Save Operation,” which follows the discussion of the remaining ASAVE commands.

## SA Command Options

The SA command options parameter lets you specify one or more of the options shown below, in any order. Table 3-2 summarizes the SA command options.

**Table 3-2. SA Command Options Summary**

Options	Description
<b>AP</b> pend	Appends save files to current ASAVE tape
Data Records <b>Not Checked</b>	Checksums are computed for the data records, not checked
<b>No Locking</b>	Disk LUs are not locked during save operation
<b>Duplicate</b>	Replaces duplicate files
<b>UN</b> it Save	Specifies a unit save of the disk pointed to by the first LU parameter
Disk-to-Tape <b>VE</b> rify	Performs a disk-to-tape verification of the save operation

### **Append to Tape (AP)**

AP appends the save files to the current ASAVE tape. The tape must be the last tape in an ASAVE tape set.

### **Data Record Checksums Not Checked (NC)**

The checksum included in each record written to tape is a single-word one's complement of the sum of all the words in the data portion of the record.

NC specifies that data record checksums on the tape are not to be checked. Instead, the checksum is computed for the data records. You may turn off the computation for increased performance when the tape device is a 7978A recording with 6250-bpi density.

### **No Locking (NL)**

NL specifies that the disk LUs are not locked during the save operation. The default is to lock the disk LUs during the save to ensure the integrity of the saved disk data.

If you select both NL and VE, errors may occur during the verify pass if any disk data was altered (written) between the save and verify passes while the disk was unlocked.

### **Unit Save (UN)**

UN specifies a unit save of the disk unit pointed to by the first LU parameter. If you do not enter UN, the save is an LU save of all the LUs specified. (See also the section "The Save Operation" later in this chapter.)



## Disk-to-tape Verify (VE)

VE performs a disk-to-tape verification of the save operation. (See the description of the verification method in the section called “Verification,” later in this chapter).

If you specify both NL and VE, errors may occur during the verify pass if any disk data was altered (written) between the save and verify passes while the disk was unlocked.

## Tape LU (TA)

Purpose: Specifies the tape LU to be used by the ASAVE commands.

Syntax: TA[ , LU]

LU                    The tape logical unit to be used.

Description:

If you do not specify an LU, the current tape LU is displayed. Magnetic tape and the CS/80 CTD are the only supported tape devices. The specified tape LU is locked and remains locked until ASAVE terminates or you specify another LU. If you do not specify a tape LU, but attempt an operation that requires tape access, an error occurs.

## Title (TI)

Purpose: Sets the title to use in subsequent save operations.

Syntax: TI[ , text]

text                    The text for the title, up to 40 characters.

Description:

The title you specify is placed in the header record of the save files on tape. Any characters over 40 are truncated. If you do not enter any text, the title consists of blanks.

When you enter TI in the runstring, be aware that CI interprets both blanks and commas as parameter delimiters. To get around this, ASAVE lets you use the underscore ( \_ ) character to replace any blanks in titles in the runstring. (ASAVE then converts the underscores to blanks.)

CI also converts all lowercase characters to uppercase in the runstring before ASAVE retrieves it. For example:

```
CI> asave|ti Unit save of LU 37 without VE
ASAVE : ti
Title = UNIT SAVE OF LU 37 WITHOUT VE
ASAVE :
```

However, when you enter the title interactively, it remains the same as typed. For example:

```
ASAVE: ti Unit save of LU 37 without VE
Title = Unit save of LU 37 without VE
ASAVE :
```

## User Error Handling (UE)

Purpose: Sets the user error handling mode or displays the current mode.

Syntax: UE[ ,ON/OFF]

ON                    You are prompted for corrective action if a recoverable error (such as a typing error) occurs. The initial setting is ON.

OFF                   The default action is taken, as described in the following sections.

Description:

If you correct the error, command execution continues. (Error conditions and the default actions are explained in the section “The Save Operation.”)

You may also use UE to see whether user error handling mode is ON or OFF by entering the command without any parameter. For example:

```
ASAVE: ue (Displays current UE mode)
User Error Handling = ON
ASAVE: ue off (Turns UE OFF)
User Error handling = OFF
ASAVE :
```

## ASAVE Command Examples

**Example 1:** List the help information to LU 6.

```
LL,6|HE
```

**Example 2:** Set the list device to LU 6, tape LU to 25, specify title “Example unit save,” and perform a unit save of the disk unit pointed to by LU 16. Lock the disks during the save, and verify the save.

```
LL,6|TA,25|TI,Example unit save|SA,16,UN,VE
```

**Example 3:** Set the list device to LU 12, tape LU to 24, title to “Example LU save,” and save disk LUs 16,17, and 19. Keep the disks unlocked during the save, and verify the save operation.

```
LL,12|TA,24|TI,Example LU save|SA,16,17,19,VE,NL
```

**Example 4:** Set the list device to LU 6, tape LU to 25, title to “Example unit and LU save.” Do a unit save of the disk unit pointed to by LU 16, then append an LU save of LU 30 to the same tape. Lock the disks during the save, and verify the save operation.

```
LL,6|TA,25|TI,Example unit and LU save|SA,16,UN,VE|SA,30,VE,AP
```

## The Save Operation

When you use the SA command, ASAVE first verifies that all the disk LUs are legal in the current system. If any LU is illegal, the command terminates with the following error logged:

```
Error - LU x not a supported disk.
```

The sections that follow describe the various elements of a save operation.

### Unit Save LU Checking

If you select the UN option to specify a unit save operation, ASAVE locates all the disk LUs associated with the disk unit pointed to by the disk LU you enter, then proceeds in the same manner as for an LU save.

### Disk Locking

If you do not select the NL option with the SA command, the disk LUs being saved are locked (the default). If you do not select VE, the disk LU is locked just before it is saved and remains locked until the save completes. Selecting VE causes the disk LU to remain locked until after the save is verified.

If the disk cannot be locked (it is currently locked to another program), when user-error handling is ON, you are prompted with the following message:

```
Error - LU x locked to (program name)
Correct problem and type GO to continue, SK to skip or BR to
break.
```

If user error handling is OFF, the SA command is skipped, and the disk LU is not saved.



## Tape Positioning

Saving disks to tape may begin in one of two places on the tape, depending upon whether you select the AP option. If you do not use AP, the save begins at the beginning of the tape and overwrites any data previously written there.

When you select AP, the save files are appended to the current ASAVE tape. First, the current data on the tape is checked to make sure it is an ASAVE tape. If it is not, the command terminates with the following error:

```
Error - Non-ASAVE tape.
```

If it is an ASAVE tape, the EOD record is located (see the section on “ASAVE Tape and File Formats” later in this chapter). The EOD record marks the end of save files on the current tape and specifies whether the save operation continued onto another tape. If it did cross tapes, you cannot use AP in this save operation, and the save terminates with the following error logged:

```
Error - Can only append to last tape.
```

If the EOD record shows this is the end of the tape set, the save begins after the last save file on the tape.

If you are using a CTD, an end-of-file (EOF) mark is written in block 0 of the tape, and the save operation begins at block 1.

## Save File Records

ASAVE save files hold three types of records: An ASCII header record at the beginning of the save file that contains user information about the save; two save definition records, and one or more disk data records that contain images of portions of the disk being saved. All records written to tape contain checksums. A more detailed explanation of these records and the ASAVE tape format is provided in the section called “ASAVE Tape and File Formats,” later in this chapter.

## Saving the Disks to Tape

When the save operation begins, the disk is locked unless you specified NL. The header record, save definition records, and disk data records are written to the tape. The save proceeds either until the end of the disk LU is reached or there is not enough usable tape left to write the next disk data record.

If too little tape remains, the appropriate EOD record is written on the tape, followed by two EOF marks. If you selected the VE option, the portion of the save operation on the tape is verified, the tape is rewound, and the following message is displayed:

```
Mount next tape and type GO to continue or BR to break.
```

The last save file written to the tape is also followed by an EOD record, indicating the end of the save, followed by two EOF marks. This constitutes the end of a save operation on the tape.

If the tape or the disk LU goes down during the save operation, the following message appears:

```
LU x is down.  
Correct problem and UP device or use BR to break.
```

You may also see system error messages about the device that is down.

## Verification

When you select the VE option for a disk-to-tape save, the save becomes a two pass process. During the first pass (save), the header and save definition records are written to tape with checksums, the data blocks are read from disk, an NC option-setting-dependent-checksum is calculated for the data blocks, and the data blocks and checksums are written to the tape.

In the second pass (verify), the tape is repositioned to where the save began on the tape, and the following messages are logged:

```
Verifying tape.  
Verifying save of LU x to file x      (For each save file)
```

For each save file, the checksums written with the header and save definition records are verified. If a checksum error occurs during verification, the appropriate error message is logged. For example:

```
Error - Header record verify failed.
```

or

```
Error - Save definition record verify failed.
```

Data records are verified by reading the data records from tape into one buffer and the corresponding data on the disk into a second buffer; then the two buffers are compared bit-by-bit. If a data record fails verification, the following error is logged:

```
Error - Data record verify failed.
```

File marks and the EOD record are also verified, with the following error logged:

```
Error - EOF verify failed.
```

or

```
Error - EOD verify failed.
```

If a verify error occurs on a save file, the appropriate error is logged and the rest of the file is skipped. The verify continues with the next file.

## Break Detection

When a break is detected, ASAVE terminates any operation and unlocks all locked LUs except tape LUs.

If a break is detected during a save operation, an attempt is made to “clean up” by placing an EOD record after the last save file that successfully completed. If this cannot be done (for example, the tape LU is down or you aborted ASAVE), the result is a corrupt ASAVE tape. If the tape LU is down, the following warning message is logged:

```
Warning - Last file corrupt.
```

The tape is corrupt in that it does not contain an EOD record, and the last file is incomplete. You may not append to the tape, and although previous files may be restored with no problems, if you try to restore the corrupted file, unpredictable results may occur.

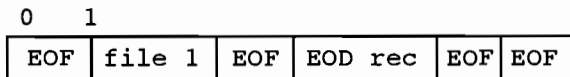
## Loading ASAVE

Load ASAVE online, using the LINK loader and the LINK command file #ASAVE.

## ASAVE Tape and File Formats

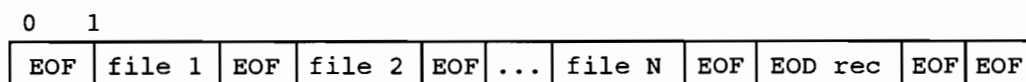
ASAVE places an EOF mark in the first block (block 0) of any CTD tape that contains ASAVE data. This provides protection from possibly overwriting a CS/80 disk if a front panel pushbutton restore is attempted with an ASAVE CTD tape. (In that case, the EOF in the first block is read, and since the EOF signifies the end of the file, the restore operation terminates immediately, and no data is overwritten on the disk.)

The ASAVE CTD tape format for a single LU save is shown below. The ASAVE magnetic tape format is identical, except it does not contain the EOF at the beginning of the tape.



A unit or multiple LU save is a series of single LU saves followed by the EOD record and EOFs. A unit save starts with the lowest numbered LU on the unit and proceeds through the highest numbered LU. In a multiple LU save, the LUs are saved in the order in which they were specified in the SA command.

The ASAVE CTD tape format for unit and multiple LU saves is shown below. The ASAVE magnetic tape format for unit and multiple LU saves is identical, except it does not contain the EOF at the beginning of the tape.



## ASAVE Tape Records

The ASAVE file has the following record format:

Header	Save Def 1	Save Def 2	Data	...	Data	EOF
--------	------------	------------	------	-----	------	-----

### Header Records

The ASCII header record is the first record in an ASAVE file. Table 3-3 shows the format of the header record, followed by descriptions of the various fields in the header.

**Table 3-3. ASAVE Header Record Format**

<pre>Created Using: ASAVE 92077-16586 REV.6000 &lt;920903.1410&gt; Tape Set Date: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Tape #: xxx File #: xxx Options: xx xx xx xx disk LU: xxx Tracks: xxxxx Sec/Trk: xxx Section: xx Trk: xxxxx Sec: xxx Created: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx User: xxxxxxxxxxxxxxxxxxxx Title: xxx</pre>	
Created Using	<i>The version of ASAVE used to save the disk.</i>
Tape Set Date	<i>The time stamp when the first file of the tape set was created; used to identify the various volumes of a tape set.</i>
Tape #	<i>Number of which tape this section of the save file is located.</i>
File #	<i>Save file number.</i>
Options	<i>Options selected for the disk LU save.</i>
disk LU	<i>Disk LU saved.</i>
Tracks	<i>Total number of tracks for the disk LU.</i>
Sec/Trk	<i>Sectors per track for the disk LU (64 word sectors).</i>
Section	<i>Save file section number, used when a save file crosses tapes.</i>
Trk	<i>Disk track where disk data in this section of the save file starts.</i>
Sec	<i>Sector in track described above.</i>
Created	<i>Time stamp when disk was saved.</i>
User	<i>User account name where ASAVE was scheduled.</i>
Title	<i>Title specified for the save operation.</i>

## Save Definition Record

Save definition records are located in the second and third records of the save file. The second save definition record is a copy of the first. Their format is shown below.

Word	Content
1	Unused
2	Checksum
3	Length of save definition record
4	Unused (reserved)
5	Tape number
6	File number (relative to beginning of tape set)
7	File number (relative to beginning of current save operation)
8	Number of saves remaining (for the save operation)
9	Type of save (1=LU, 2=UNIT)
10	Operating system (.OPSY)
11	Size of data records (less 2 word checksum)
12–13	Time stamp
14–15	Block # where previous save file starts (CTD only)
16–17	Tape set ID
18–31	Unused (reserved)
32	Disk LU
33	Disk type
34	Save section number
35	Starting track
36	Starting sector
37	Disk address
38	Disk unit
39	Disk volume
40	Cylinder
41	Head
42	#Surfaces
43	#Tracks
44	#Spares
45	#64 word sectors/track
46	Reserved
47–64	Unused (reserved)

## Data Record

All the save file data records are of a uniform length, except the last data record may be shorter. The first two words of the data record make up a checksum. The third and fourth words are a disk block address where the data starts. Word five is the file number. Words 6–10 are unused, and disk data begins in word 11.



## End-of-Data Record

The EOD record marks the end of the save files on the current tape and contains information that specifies whether the save continues on another tape. Its format is shown below:

Word	Content
1	Unused
2	Checksum
3	EOD record length
4	Type of EOD (described below)
5	File # of last save file (relative to the save operation)
6	File # of last save file (relative to the beginning of tape set)
7	Tape #
8-9	Block # where last save file starts (CTD only)
10-14	'ASAVE EOD'
15-16	'END' or 'CONT'
17-18	Time stamp
19-20	Tape set ID
21-32	Unused (reserved)

The “type of EOD” (word 4) may have one of the three values below:

- 1 = end of save operation
- 2 = saving the current disk LU continues on the next tape
- 3 = save continues on next tape with a new disk LU

## ASAVE Example

Table 3-4 shows the output (to the log device) of an LU save. In this example, a list file save is created, the tape LU is defined as LU 8, the title is set to “Sample LU save,” and a save with verification of disk LUs 12 and 57 is done. After the save operation completes, the tape is rewound and set offline so other users cannot overwrite it.

Table 3-4. ASAVE Example

```
ASAVE - REV.6000 <920903.1410>
Use ? for help.
ASAVE : ll,'save
Opened list file
ASAVE : ta,8
Tape LU = 8
ASAVE : ti, Sample LU save
Title = Sample LU save
ASAVE : sa,12,57,ve
Disk LU(s) to be saved:
12
57

Created Using: ASAVE 92077-16586 REV.6000 <920903.1410>
Tape Set Date: Tue Dec 1, 1992 3:20:41 pm
Tape #: 1      File #: 1      Options: VE      Disk LU: 12
Tracks: 50    Sec/Trk: 96    Section: 1    Trk: 0      Sec: 0
Created: Tue Dec 1, 1992 3:20:41 pm      User: SYSTEM
Title: Sample LU save

Created Using: ASAVE 92072-16586 REV.6000 <920903.1410>
Tape Set Date: Tue Dec 1, 1992 3:20:41 pm
Tape #: 1      File #: 2      Options: VE      Disk LU: 57
Tracks: 48    Sec/Trk: 96    Section: 1    Trk: 0      Sec: 0
Created: Tue Dec 1, 1992 3:21:04 pm      User: SYSTEM
Title: Sample LU save

Verifying tape.
Verifying save of LU 12 to file 1
Verifying save of LU 57 to file 2

      Save Record
-----
File #  Tape #  LU
-----
   1      1     12
   2      1     57

ASAVE : rw
ASAVE : ll
Closed list file.
ASAVE : ex
```

## ASAVE Error Messages

ASAVE error messages, error messages and warnings shared by ASAVE and ARSTR, and error messages generated by ARSTR are all presented following the discussion of ARSTR.

# ARSTR Physical Restore Utility

ARSTR restores one disk LU, a group of disk LUs, or an entire disk unit from magnetic tape or CS/80 CTD.

When you use ARSTR to restore a disk LU, first dismount the disk volume, run ARSTR, and then mount the volume.

## Calling ARSTR

Invoke ARSTR with the following runstring:

```
CI> [RU, ]ARSTR[, commands]
```

If you do not enter a command in the runstring, ARSTR prompts you interactively, as follows:

```
ARSTR:
```

Separate command parameters either with commas or blanks. You may combine multiple commands either in the runstring or in response to the prompt, separated by vertical bars. You may also use the underscore ( `_` ) character to separate parameters in the runstring; ARSTR converts them later to blanks.

Commands are executed in order from left to right. If any command causes a serious error, execution terminates with that command (all the preceding commands are executed).

## ARSTR Commands

ARSTR uses commands that let you terminate ARSTR, list all the commands and their parameters, list header information, open a list file or list device, restore a file, rewind and set tape offline, specify a tape LU for ARSTR commands to use, and specify the user error handling mode. Table 3-5 summarizes the commands, which are described in the following sections.

**Table 3-5. ARSTR Commands Summary**

Commands		Description
<b>Information Command</b>		
<b>HElp</b>		Lists all ARSTR commands and parameters
<b>Configuration Commands</b>		
<b>REstore</b>	file #: LU	Restores files from tape, either an entire LU or a unit restore
<b>RW Rewind</b>		Rewinds and sets the tape offline
<b>TApe</b>	LU	Sets tape LU to the specified LU or displays current LU
<b>UE User Error Handling</b>	ON/OFF	Sets user error handling mode ON or OFF or displays current status
<b>Listing Commands</b>		
<b>LH List Header</b>	file #	Lists header information for the specified file number
<b>LL List Device</b>	file descriptor/ lu	Opens a list file or list device to which information is to be logged in addition to the terminal
<b>Exit Commands</b>		
<b>ABort</b> <b>ENd</b> <b>EXit</b>		These commands all terminate ARSTR

### **Abort, End, and Exit (AB) (EN) (EX)**

**Purpose:** Any one of these commands terminates ARSTR.

**Syntax:** AB, EN, or EX

**Description:**

Upon termination, all disk LUs are released and any list file or device closed.

## Help (HE)

Purpose: Lists all ARSTR commands and parameters.

Syntax: HE, ??, or ?

## List Header (LH)

Purpose: Lists header information from the tape, for the specified file number.

Syntax: LH[,file#]

file#            The file for which the header information is being obtained.

Description:

If the specified file does not exist on the current tape, the header record of the first or last file on the tape is listed. If you do not specify a file number, the header record of the next save file is the default. If positioned at the EOD record, the header record of the last save file is listed.

---

**Note**            All file numbers specified are relative to the beginning of the first ASAVE tape. If the save file for a disk LU crosses tapes, it has the same file number on both tapes.

---

## List Device (LL)

Purpose: Opens a list file or list device to which terminal displays and error messages are sent in addition to the log device (the terminal).

Syntax: LL[,file descriptor/lu]

file descriptor    The list file.

lu                 The list device.

Description:

A defined list file is created if it does not already exist, and a list device is opened. Any previously opened list file or device is closed first. If you do not specify a file descriptor/LU, any currently open list file or device is closed.

A duplicate file error occurs if you specify a file and a duplicate file already exists. In this case, the command terminates with an error if user error handling (described below) is ON; otherwise, the command is skipped.

If you specify a device that is down and user error handling is ON, you are prompted to correct the problem and bring up the device, or enter breakmode. If user error handling is OFF, the command is skipped.

If the LU goes down when you use a list device, and user error handling is ON, you are prompted to correct the problem or enter breakmode. If user error handling is OFF, you must enter breakmode. When a break occurs, the list file or device is closed and the command continues to execute.

## Restore (RE)

The RE command has two formats, one for LU restores and one for unit restores. Each format is described separately below.

### File-by-file (LU) Restore:

Purpose: Specifies an LU restore from tape.

Syntax: `RE, file#:LU[, file#:LU][, ...][, <options>]`

`file#:LU` The file number on tape and the disk LU to which it is to be restored. There are two variations of this parameter:

`disk LU(:LU)` The file number is defaulted to 1 for the first disk specified, 2 for the second, and so forth.

`file number` The destination disk LU (`file#`) is determined by examining the save definition record of the save file for the disk LU that was saved. If you specify a file number for any of the `file#:LU` parameters, then all the `file#:LU` parameters must specify a file number.

`<options>` The options are described in the next section.

---

**Note** All file numbers specified are relative to the beginning of the first ASAVE tape. If the save file for a disk LU crosses tapes, it has the same file number on both tapes.

---

## Unit Restore:

Purpose: To specify a unit restore of the disk unit.

Syntax: RE[ ,<file#>[ ,<options> ]

<file#> The file number on tape and the destination disk LU.

<options> The options are described in the next section.

Description:

The restore operation is explained in detail in the section, “The Restore Operation,” later in this chapter.

## RE Command Options

The RE command options parameter lets you specify one or a combination of the options shown below. Options may be specified in any order. Table 3-6 summarizes the RE command options.

**Table 3-6. RE Command Options Summary**

Options	Description
Data Records Checksums Not Checked	Checksums are computed for the data records, not checked
UNit Restore	Specifies a unit restore beginning at the file number you enter, or file 1 if you do not enter a number
VERify Restored Disks	Specifies verification of the restored disks

### Data Records Checksums Not Checked (NC)

The checksum included in each record written to tape is a one-word complement of the sum of all the words in the data portion of the record.

NC specifies that the data record checksums are not to be checked. Instead, the default is to verify the checksums for the data records that were written with checksums. You may turn off checksum verification for increased performance when the tape device is an HP 7978A reading a 6250-bpi density tape.

### Unit Restore (UN)

The UN option specifies a unit restore beginning at the file number you enter, or file 1 if you do not enter a file number. If the file number you enter specifies a save file, it does not have to be the first save file of a unit save operation. The restore operation only restores the specified save file and any subsequent save files if they are from the same disk unit and were saved in the same operation.

## Verify Restored Disks (VE)

VE specifies a verification of the restored disks (verification is described below).

## Rewind (RW)

Purpose: Rewinds and sets the tape offline. A CTD is also unloaded.

Syntax: RW

## Tape LU (TA)

Purpose: Sets the tape LU to the specified LU, or if no LU is specified, displays the current tape LU.

Syntax: TA[ , LU ]

LU                      The tape logical unit selected.

Description:

If you do not specify an LU, the current tape LU is displayed. Magnetic tape and CTD are the only supported tape devices. The specified tape LU is locked and remains locked until ARSTR terminates or another tape LU is defined.

The tape LU is used by all commands that access the tape. If you do not specify a tape LU, but attempt an operation that requires tape access, an error occurs.

## User Error Handling (UE)

Purpose: Sets the user error handling mode or displays the current mode if no parameter is entered.

Syntax: UE[ , ON/OFF ]

ON                      You are prompted for corrective action if a recoverable error occurs. If you correct the error, command execution continues. The initial setting is ON.

OFF                     The default course of action is taken, as described in the discussion of each command.



## ARSTR Command Examples

**Example 1:** Set the list device to LU 6, tape LU to 25, and list the header from file number 2.

```
LL,6|TA,25|LH,2
```

**Example 2:** Set the list device to LU 12, tape LU to 23, and restore files 1-3 to disk LUs 16, 17, and 19 respectively. Verify the restore operation.

```
LL,12|TA,23|RE,:16,:17,:19,VE
```

**Example 3:** Close the previous list device, set the tape LU to 24, and restore files 1–3 to disk LUs 16, 17, and 19 respectively. Verify the restore operation.

```
LL|TA,24|RE,2:17,1:16,3:19,VE
```

**Example 4:** Set the tape LU to 25, and restore files 2 and 3 to the disk LUs specified in the save definition records of those files. Verify the restore operation.

```
TA,25|RE,2,3,VE
```

**Example 5:** Set the tape LU to 23, and perform a unit restore starting at file 1. Verify the restore operation.

```
TA,23|RE,UN,VE
```

## The Restore Operation

All disk LUs being restored to are first checked to ensure that they are legal. Checking may occur when you enter the RE command or when the appropriate save file is encountered during the restore, as explained below.

The following sections describe the various elements of a restore operation.

### Checking the Disks

Disk LUs entered in the RE command are checked when you enter the command. If any disk LU is illegal, the restore operation terminates at once, and the following error message appears:

```
Error - LU x not a supported disk.
```

Disk LUs that are identified from the data in the save files are checked as the save files are encountered during the restore operation. If any disk LU is illegal for the current system, the following warning message is logged:

```
Warning - LU x not a disk in current system.
```

If user error handling is ON, you are prompted to enter breakmode, skip this restore, or enter the LU of the disk to restore. If user error handling is OFF, the restore is skipped.

### **Sectors per Track Must Match on Disks**

ARSTR only restores a disk save file on tape to a similar disk. For the disk to be similar, the sectors per track of the saved disk must be the same as those of the disk to be restored to. Otherwise, the following error occurs, and the restore of that disk is skipped:

```
Error - Sectors per track mismatch.
```

### **Total Number of Tracks May Differ**

The total number of tracks of the saved disk (source) may differ from the total number of tracks on the target disk (destination). If the number of tracks of the destination disk differs from the source disk, ARSTR issues a warning message. If user error handling is ON, ARSTR prompts you to confirm the restore. For example:

```
Warning - Saved disk has less tracks than target disk LU x
Saved disk tracks = x
Target disk tracks = x
Restore this disk ?
```

If user error handling is OFF (the default), the following message appears, and the restore operation continues:

```
Continuing to restore.
```

If the source disk has fewer tracks than the destination disk, tracks beyond where the restore completes remain unaltered. The destination disk LU can be used as a holding area but cannot be mounted because the bitmap of the source disk LU and the destination LU may not be the same.

If the destination disk has fewer tracks than the source disk, the restored source tracks will be limited to the number of tracks on the destination disk.

## Disk Locking

Each disk LU being restored is locked when the restore operation begins, to ensure the integrity of the restored data. It remains locked until the restore completes (if the VE option was not specified), or until verification completes (if VE was specified). If the disk cannot be locked (for example, it is currently locked to another program) and user error handling is ON, you are prompted with the following message:

```
Error - LU x locked to (program name) Correct problem and type
GO to continue, SK to skip or BR to break.
```

If user error handling is OFF, the restore of that disk is skipped.

## Order of Disk Restoration

Disks are restored in the same order as they were saved to tape, regardless of the order in which they may be specified in the RE command.

## Unit Restore

For a unit restore, the tape is positioned to the file number specified in the RE command. ARSTR restores the save file and any succeeding save files that are on the same disk unit and were saved in the same save operation. Information contained in the save definition record determines if these conditions are met.

## Save Definition Records

Information for the restore of a disk is obtained from the system and the save definition record in the ASAVE file. Data in the save definition record is used to control the restore.

Each tape save file being restored is checked to ensure that it is a legal ASAVE file. If it is not, the restore operation terminates with the following message:

```
Error - Non-ASAVE tape.
```

When the actual restore begins, the header record of the save file is logged. If a checksum error occurs on the header record, the following warning displays, and the restore continues:

```
Warning - Header record checksum.
```

If the checksums of both the save definition records fail to verify, the restore of that file is skipped. When this occurs the following error is logged:

```
Error - Save definition records checksums.
```

## Restoring the Data

If the save file being positioned to (LU or unit restore) is not on the currently mounted tape, if user error handling is ON, you are prompted with this message:

```
Error - First file on tape is x searching for file y. Correct
problem and type GO to continue, SK to skip or BR to break.
```

If user error handling is OFF, the restore of the disk LU is skipped.

For each save file, disk data from the tape is restored to the disk LU until the end of the disk LU, or the end of the disk data in the save file is reached. If checksum errors occur on data records during the restore, the read of that record is retried up to five times before the following error message is logged:

```
Error - Data record checksum.
Data targeted for disks blocks x to y.
Continuing.
```

When the restore of a disk LU completes, the following message is logged:

```
Restore of file x to disk LU y complete.
```

If the end-of-tape condition is detected before the end of the save file is detected, the restores from the current tape are verified (if VE was specified), and you are prompted with the following message:

```
Mount next tape and type GO to continue or BR to break.
```

If a tape LU or disk LU goes down during the restore, the following message displays:

```
LU x is down.
Correct problem and UP device, or use BR to break.
```

The system may also issue messages about the device that is down.

## Verification

Restore verification is similar to save verification. When you select the VE option for a tape-to-disk restore, the restore operation becomes a two pass process. During the first pass (restore), data is read from the tape, checksums (if written with the data) are verified, and the data is written to disk. In the second pass (verify), the data is re-read from tape into one buffer. The corresponding data restored to disk is read into a second buffer, and the data in the two buffers are compared bit-by-bit.

If a mismatch between the tape and disk occurs, the following warning appears, and the verify of that restore continues to the end of the disk or until three verify errors occur:

```
Warning - Possible data loss in disk block range x to y.
```

If three verify errors occur, the verification process aborts.

## Break Detection

If ARSTR detects a break, it terminates all operations and unlocks all LUs except tape LUs. If a break occurs during a restore or during list header command execution, the tape is rewound.

## Loading ARSTR

Load ARSTR online, using the LINK loader and the LINK command file #ARSTR.

## Offline System

Since ASAVE and ARSTR are not segmented, they may be loaded into a memory-based system for offline use. The memory-based, offline system serves three purposes: Installing systems, restoring corrupt system disk LUs, and saving disk LUs from crashed disks.

In general, after you establish a new system, you should use the same system file to create an offline restore system. This should include a startup program (COMND, CI) that enables the scheduling of programs; the restore (ARSTR) utility, and optionally, the save (ASAVE) utility; and the appropriate format (VSCSI, FORMC, FORMT, FORMF) utilities. Using the same system file as the current system eliminates any need to reconfigure the offline system.

---

**Note** COMND is a compact version of File Manager (FMGR) which supports only a subset of FMGR commands. See Appendix A for more information.

---

The RTE-A BUILD utility is used to build the actual memory-based offline system. The following example illustrates a BUILD command file:

```
YES,          (Automatic partitioning)
256,          (Memory size in pages)
RP,ASAVE,,    (Relink and create ID segment for ASAVE)
RP,ARSTR,,    (Relink and create ID segment for ARSTR)
RP,FORMC,,    (Relink and create ID segment for FORMC)
RP,D.RTR,,    (Relink and create ID segment for D.RTR)
RP,COMND,,    (Relink and create ID segment for COMND)
ST,,,,,      (Make COMND the startup program)
/E           (Complete the build process)
```

After you create the system, place it onto bootable media. For detailed instructions on assembling a memory-based, offline system, refer to the *RTE-A System Generation and Installation Manual*, part number 92077-90034.

## ARSTR Example

Table 3-7 shows the output of a LU restore. Here the list file 'RSTR is created, the tape LU is defined as LU 8, and a restore with verify of save files 1 and 2 is performed. You must dismount the disk LUs to be restored before, and mount them after running ARSTR. When the restore completes, the tape is rewound and set offline so it cannot be overwritten.

Table 3-7. ARSTR LU Restore Output

```
CI> DC 12
CI> DC 57
CI> ARSTR

ARSTR - REV.6000 <920903.1620>
Use ? for help.
ARSTR : ll,'rstr
Opened list file.
ARSTR : ta,8
Tape LU = 8
ARSTR : re,1,2,ve

Created Using: ASAVE 92077-16586 REV.6000 <920903.1410>
Tape Set Date: Tue Dec 1, 1992 3:20:41 pm
Tape #: 1      File #: 1      Options: VE      Disk LU: 12
Tracks: 50    Sec/Trk: 96    Section: 1      Trk: 0      Sec: 0
Created: Tue Dec 1, 1992 3:20:41 pm      User: SYSTEM
Title: Sample LU save

Restore of file 1 to disk LU 12 complete.

Created Using: ASAVE 92077-16586 REV.6000 <920903.1410>
Tape Set Date: Tue Dec 1, 1992 3:20:41 pm
Tape #: 1      File #: 2      Options: VE      Disk LU: 57
Tracks: 48    Sec/Trk: 96    Section: 1      Trk: 0      Sec: 0
Created: Tue Dec 1, 1992 3:21:04 pm      User: SYSTEM
Title: Sample LU save

Restore of file 2 to disk LU 57 complete.
Verifying restores from tape.
Verifying restore of file 1 to LU 12
Verifying restore of file 2 to LU 57
ARSTR : rw
ARSTR : ex
Closed list file.

CI> MC 12
CI> MC 57
```

## **ASAVE and ARSTR Error Messages and Warnings**

The sections that follow list 1) all the messages that ASAVE generates, 2) error messages that are shared by ASAVE and ARSTR, 3) shared warning messages, and 4) the error messages that ARSTR generates.

### **ASAVE Messages**

#### **Error – Can only append to last tape.**

You selected AP when saving to a tape that was not the last tape in a tape set. Mount the last tape, and retry the operation.

#### **Error – Corrupt tape, no EOD.**

A corrupt tape resulted when an ASAVE break was unable to clean up. Perhaps the tape LU was down or not ready, or you aborted ASAVE. Only the last file is corrupt. You may not append to this tape.

#### **Error – Data record verify failed.**

A bit-by-bit compare of the tape data record to the disk data detected an error (this also verifies the checksum).

#### **Error – EOD checksum bad, unable to append.**

A checksum error was detected in the EOD record while positioning to it for an append save. Since information in the EOD record is used in the save operation, and the correctness of that data cannot be guaranteed, the append save is not allowed.

#### **Error – EOF verify failed.**

An EOF failed to verify during the verify pass of a save operation.

#### **Error – Header record verify failed.**

A checksum error was detected during verification of the header record of a save file.

#### **Error – Illegal unit save.**

The specification for the unit save contained more than one disk LU.

#### **Error – No disk to save.**

The SA command did not specify any disk LUs.

#### **Error – Save definition record verify failed.**

A checksum error was detected during verification of one of the save definition records of a save file.

#### **Error – Tape LU xx write protected.**

The SA command required write access to a write-protected tape LU.

## ASAVE and ARSTR Shared Error Messages

### **Error – File number.**

You specified an illegal file number in either the RE or LH command, or you may have specified the file number more than once, which is not allowed. In the RE command, if one file number is specified for a <file#:LU> parameter, all <file#:LU> parameters must specify a file number.

### **Error – First file on tape is xx**

or

### **Error – Last file on tape is xx**

The specified file cannot be found on the current tape. This error may be followed by additional information about the error. The corrective action may be to mount the correct tape.

### **Error – <fmp error message>**

An FMP error occurred while accessing the list file or device.

### **Error – Illegal tape LU.**

You used the TA command to enter a tape LU that is illegal in the current system.

### **Error – IOxx on LU xx**

A device I/O error was detected.

### **Error – List LU = x**

You tried to define the tape LU as the currently defined list device.

### **Error – LU xx is a mirrored volume**

The LU you entered is a mirrored volume LU. You must specify the physical paired disk LU associated with the mirrored volume LU. (Refer to the *Data Pair/1000 Reference Manual*, part number 92050-90001, for more information.)

### **Error – LU xx locked to <program name>**

An attempt to lock the LU failed because the LU is locked to another program. This error can occur for both tape and disk LUs.

### **Error – LU xx not a supported disk.**

The LU you entered is not a legally supported disk, or it is not a disk LU in the current system.

### **Error – Non-ASAVE tape.**

The tape being accessed is not an ASAVE format tape.



**Error – Not enough memory (size program).**

The utility does not have enough addressable memory for its buffers. You must size the program.

**Error – Save definition records checksums.**

A checksum error was detected while reading the save definition records of a save file (both save definition records have checksum errors).

**Error – Tape retries exhausted.**

or

**Error – Disk retries exhausted.**

The available retries of a tape or disk access are used up. The tape or disk media may be deteriorating.

**Error – Tape LU undefined.**

A command requiring tape access was entered, but no tape LU was defined. Specify the tape LU with the TA command, and reenter the command.

**Error – Tape LU = x**

You entered the currently defined tape LU as the list device.

**Error – Unknown command.**

An unknown command was entered.

**Error – Unknown parameter.**

An unknown parameter was encountered in a command.

**Error – Uninitialized tape.**

An uninitialized CTD tape was detected. Initialize the tape with the FORMC utility.

## **ASAVE and ARSTR Shared Warning Messages**

### **Warning – CTD may be loading.**

A tape-not-ready error may occur if you attempt to access the CTD tape while the tape is loading. Wait until the tape completes loading, then continue.

### **Warning – Data record checksum. Data targeted for disk blocks xxx to xxx Continuing.**

A checksum error occurred on a data record being restored. This happens only after the read from the tape is retried five times. If you selected the VE option, this disk block range may be smaller during the verify pass. In any case, the restore of that LU continues.

### **Warning – Header record checksum.**

There was a checksum error reading the header record. Since no information in the header is used by the utility, this is not a serious error.

### **Warning – Last file corrupt.**

The break during a save operation could not clean up, because the tape LU could not be accessed (it was down, or not ready).

### **Warning – LU xx is a mirrored volume.**

An LU that was determined from the save file during a restore operation is a mirrored volume LU in the current system. (Refer to the *Data Pair/1000 Reference Manual* for more information.)

### **Warning – LU xx not a disk in current system.**

The LU named is not legal in the current system.

### **Warning – Possible data loss in disk block range xxx to xxx**

The restore verify pass detected the possibility of data loss in the specified disk block range when it did a bit-by-bit comparison of the tape data and disk data.

### **Warning – Saved disk has more tracks than target disk LU xx**

or

### **Warning – Saved disk has less tracks than target disk LU xx**

There is a difference between the total number of tracks of the saved disk on tape and the disk you are restoring to. If user error handling is ON, you must confirm that you want the restore to continue.

## **ARSTR Error Messages**

### **Error – Bad LU.**

A bad LU specification in a restore command was detected.

### **Error – Illegal unit restore.**

You made an illegal specification for a unit restore; perhaps you entered more than one file number.

### **Error – No LU or file number.**

You entered a restore command for an LU without specifying the file number, or specified the LU as a parameter to select a file to be restored.

### **Error – Sectors per track mismatch.**

There is a difference in the sectors per track of the disk saved on tape and the disk to which you are trying to restore. The sectors per track for the saved disk on tape and the disk you are restoring to must be the same.

### **Error – Error msg – Wrong Tape.**

This message appears if you try to restore a series of tapes in the wrong order.

## **ARSTR Warning Messages**

### **Warning – Bad Checksum.**

This message may be ignored if your restore continues and the tape verifies okay.

## Disk-to-Disk Copy (COPYL)

COPYL is an online utility that copies the entire contents of one disk LU to another. It is intended for users who want to make copies of disks without using FMP. COPYL directly addresses the disk driver, rather than using the FMP routines, thus increasing the speed of the copy operation. It ignores the file structure in the copy process, addressing tracks and sectors on the disk, rather than records in the individual files. The speed of the copy, therefore, is independent of the contents of the disk.

Note that the utility can only copy like media; in other words, you can copy an HP 9895 disk to another 9895 disk, but not to an HP 7908 disk. The disk need not be in FMP format; COPYL copies LIF disks, HP 125 compatible disks, or any other disk that has a physically compatible recording format.

The available memory following the last location used by the COPYL program itself becomes the buffer for the copy operation; a minimum of 128 words (equivalent to one sector) must be available. The larger the buffer area, the more efficiently COPYL operates. If the memory buffer is not large enough to copy a full sector at a time, you can increase the size of the partition for COPYL by using the loader SZ command.

See the sections “Backup Utilities” and “File Interchange on RTE-A” in Chapter 2 to learn when to use COPYL and other physical backup utilities.

## Calling COPYL

Before you invoke COPYL, dismount the destination disk LU with the DC,<LU> command. You can then call COPYL interactively, or in the runstring, as follows:

```
CI> [RU,] COPYL [,<source LU>, <destination LU>]
```

The <source LU> and <destination LU> are the logical unit numbers of the disk to copy and the disk to receive the data. If you do not specify the source and destination, COPYL prompts you:

```
Enter "From" LU, "To" LU <EX to Exit>
```

If you specify the source and destination improperly (that is, not as an integer), the following message appears:

```
Bad parameter format.  
Format is from, to; i.e., 32,31
```

## The Copy Operation

When you enter the source and destination parameters correctly, COPYL displays the following message:

```
Contents of LU xx will be destroyed. Enter "GO" to proceed.
```

When you enter GO, COPYL begins the copy operation, overlaying the destination disk with the contents of the source disk, sector by sector. When the copy is complete, this message displays:

```
COPYL exiting; verify that "system" disk is in place  
Enter "GO" when ready.
```

The utility does not exit until you enter GO.

## COPYL Error Messages

The following error messages may be issued by COPYL:

### **Disk is not up and available**

One of the disk LUs (source or destination) is down or busy with another program.

### **Disk sizes do not match**

The number of tracks and the number of sectors per track must be the same for both disks.

### **Insufficient memory in partition. Resize program and try again.**

The program uses the remainder of its partition for the copy buffer. There is not enough memory behind the program for a minimum buffer (128 words).

### **Not a disk LU**

The device type recorded in the DVT indicates that the source or destination devices are not disks or are not the same device type.

### **"To" disk must be dismounted.**

The destination disk must be dismounted before running COPYL. Exit the program, dismount the "to" disk, and try again.

## DSAVE Physical Backup Utility

DSAVE is used to save data from fixed disks to removable disks (for example, 3-1/2, 5-1/4, and 8 inch floppies, and HP 7906H removable platters). DSAVE lets you back up large, contiguous areas of disks to removable disk media regardless of the content of the disks (such as files). You may use DSAVE to save one disk LU, a group of disk LUs, or an entire disk unit to removable disk.

Because DSAVE is used to save data in one or more LUs, file structures are ignored. To save files, use the CI CO command, or use the TF utility with other media.

The size of LUs saved may be a factor when the removable media are micro or mini floppies. Large LUs may take longer to complete, and require many disks. It takes approximately 30 seconds to save without verification on a micro floppy that holds approximately 1055 blocks of data (including one block reserved for header record). With verification, the time is one minute. If you are using old, removable disks (or any used disk of uncertain integrity), verification is recommended even though the save operation takes longer.

### Calling DSAVE

Invoke DSAVE with the following runstring:

```
CI> [RU,]DSAVE[;command[;command;[...]]]
```

Alternatively, you may run DSAVE interactively, entering commands at the following prompt:

```
DSAVE:
```

Separate command parameters either with commas or blanks. You may enter multiple commands either in the runstring or in response to the prompt, separated by semicolons. In the runstring, semicolons must be preceded by a backslash (\) so that CI does not interpret them as CI command delimiters. The underscore ( \_ ) character is also used in the runstring as a parameter separator. For example,

```
CI> dsave\;me32\;lh\;sa_20
```

Commands are executed in order, from left to right. If any command causes a serious error, execution terminates with that command (all the preceding commands are executed).

## DSAVE Commands

DSAVE commands let you terminate DSAVE, display all the commands and their parameters, list header information, define a list file or list device, define removable disk media, specify a save operation, specify a save title, and define the error handling mode. Table 3-8 summarizes the commands, which are described in the following sections.

**Table 3-8. DSAVE Commands Summary**

Commands		Description
<b>Information Command</b>		
HElp		Lists all DSAVE commands and parameters
<b>Configuration Commands</b>		
MEdia	LU	Defines the removable disk LU
SAve	LU	Specifies a save operation from fixed to removable disk
Title	Text	Sets the title to be used in subsequent save operations
UE User Error Handling	ON/OFF	Sets user error handling mode and defines how user errors are handled
<b>Listing Commands</b>		
LH List Header	file #	Lists header information from removable disk
LL List Device	file descriptor/ lu	Defines a list file or device to which error messages are sent in addition to the terminal
<b>Exit Commands</b>		
ABort ENd EXit		These three commands terminate DSAVE

## Abort, End, and Exit (AB) (EN) (EX)

Purpose: Any one of these commands terminates DSAVE.

Syntax: AB, EN, or EX

Description:

When the program terminates, the list file or device specified is closed, and all locked LUs are released.

## Help (HE)

Purpose: Displays a list of all DSAVE commands and the parameters of each command.

Syntax: HE, ??, or ?

## List Header (LH)

Purpose: Lists the header information for a file from the removable disk.

Syntax: LH[ ,<file#>

file#                    The file number on the removable disk.

Description:

If the specified file does not exist on the current removable disk, the header record of the first or last file on the removable disk is listed, depending upon the value of the specified file. If you do not enter a file number, the program lists the header information for the next save file (or the first save file, if this is the first time header information is being listed from the removable disk).

If the file number is smaller than the file number of the first save file on the removable disk, the header record of the first save file is listed. If the number is greater than the last save file, the header record of the last save file is listed.

---

### Note

All file numbers specified are relative to the beginning of the first DSAVE removable disk, if there is more than one. If the save file for a disk LU occupies more than one removable disk, it has the same file number on all the disks.

---

Table 3-9 illustrates LH command use, followed by descriptions of the various fields in the header.



**Table 3-9. LH Command Example**

DSAVE : LH,1

Created Using: DSAVE 92077-16702 REV.6000 <920903.1642>  
Volume Set Date: Wed Dec 9, 1992 3:06:41 pm  
Volume #: 1 File #: 1 Options: Disk LU: 61  
Tracks: 6 SecTrk: 62 Section: 1 Trk: 0 Sec: 0  
Created: Wed Dec 9, 1992 3:06:46 pm User: SYSTEM  
Title: Sample header

Created Using	<i>Information on the version of DSAVE used to save the disk.</i>
Volume Set Date	<i>Time stamp of creation of the first file, used to identify a multi-volume set.</i>
Volume#	<i>The number that identifies a set of removable disks.</i>
File#	<i>Save file number.</i>
Options	<i>Options selected for the save operation.</i>
Disk LU	<i>Disk LU saved.</i>
Tracks	<i>Total number of tracks for the disk LU.</i>
SecTrk	<i>Sectors per track for the disk LU (64-word sectors).</i>
Section	<i>Save file section number, used when a save operation crosses disks.</i>
Trk	<i>Disk track where disk data in this section of the save file starts.</i>
Sec	<i>The sector in track described above.</i>
Created	<i>Time stamp when disk was saved.</i>
User	<i>Name of the user account from which DSAVE was scheduled.</i>
Title	<i>Title specified for the save operation.</i>

## List Device (LL)

**Purpose:** Defines a list file or list device to which error messages and terminal displays are to be sent in addition to the terminal.

**Syntax:** LL[,<file descriptor>/lu]

file descriptor The file to be opened.

lu The list device.

### Description:

If the list file you define does not already exist, it is created, and a list device is opened. (A previously opened list file or device is closed before the new list file is created or the list device is opened.) If you do not specify a file descriptor/LU, the currently defined list file or device is closed.

A duplicate file error occurs if the file you specify already exists. If user error handling is turned ON, the LL command terminates with an error; otherwise, the command is skipped.

If you specify a device that is down or a closed list file, you are prompted to correct the problem or enter breakmode if user error handling is ON. If user error handling is OFF, the default is to break. In this case, the list file or device is closed, and the executing command continues to execute.

## Media (ME)

**Purpose:** Defines the removable disk LU.

**Syntax:** ME[,lu]

lu The logical unit number of the removable disk. If omitted, either the previously defined LU or zero is displayed.

### Description:

This LU number is used by DSAVE to access the removable disk. You can also use ME to examine the LU currently defined.

The specified LU must not be mounted. It is locked and remains locked until DSAVE terminates or another LU is defined. You must enter ME before the SA and LH commands; otherwise, the message, "Removable disk media LU undefined" is displayed.

To find out the range of LUs associated with the removable disks, consult the system generation answer file, or check with a superuser (or your system manager).

---

**Note**

The removable disk media LU must be generated into the operating system, and the entire removable media must be a single LU; otherwise, there may be problems. For example, the two surfaces of an HP 7906 removable platter must be generated in the system as one LU in cylinder mode.

---

**Save (SA)**

Purpose: Specifies a save operation from a fixed disk to a removable disk.

Syntax: `SA, lu[ , lu[ , . . . , lu[ , VE[ , NL] ] ] ]`

or

`SA, lu, UN[ , VE[ , NL] ]`

lu            The logical unit of the disk.

Description:

You may perform a save operation on one contiguous area specified by one LU, or as a unit save of the whole drive unit.

The save operation is described in detail in the section “The Save Operation,” which follows the discussion of the remaining DSAVE commands. The SA command options are described below.

**SA Command Options**

The SA command options parameter lets you specify one or a combination of the options shown below. Options may be specified in any order. Table 3-10 summarizes the options.

**Table 3-10. SA Commands Option Summary**

Options	Description
No Locking	Disk LUs are not locked during a save operation
UNit Save	Specifies a unit save of the disk pointed to by the specified LU
VERify	Instructs DSAVE to verify the save operation



## No Locking (NL)

NL specifies that the disk LUs are not to be locked during the save operation. The default is to lock them during the save to ensure the integrity of the saved disk data.

If you select both the NL and VE options with the SA command, errors may occur during verification if disk data was altered (written) between the save and verify passes while the disk was unlocked.

## Unit Save (UN)

UN specifies a unit save of the disk unit pointed to by the specified LU. If you do not specify UN, the save is considered to be an LU save of all the LUs specified.

## Verify (VE)

The VE option instructs DSAVE to verify the save operation. (The verification method is described later in the section called “Verification”). Verification is recommended to ensure a successful save operation, as it avoids problems such as bad disk media or disk drive errors. If you use old removable disks for a save, VE is highly recommended.

## Title (TI)

Purpose: Sets the title to use in subsequent save operations.

Syntax: TI[ ,text]

text                      The text for the title, up to 40 characters.

Description:

The specified title is placed in the header record of the save files on the removable disk. Any characters over 40 are truncated. If you do not enter any text, the title consists of blanks.

When you enter TI in the runstring, be aware that CI interprets both blanks and commas as parameter delimiters. If you do not want to enter any text for the title, you may enter the underscore ( \_ ) character to replace the default blanks; DSAVE later converts the underscore(s) to blanks.

CI also converts all lowercase characters to uppercase in the runstring before DSAVE retrieves it. For example:

```
CI> dsave\;ti Unit_save_of_LU_37_without_VE
DSAVE : ti
Title = UNIT SAVE OF LU 37 WITHOUT VE
DSAVE :
```

However, when you enter the title interactively, it remains the same as typed. For example:

```
DSAVE : ti Unit save of LU 37 without VE
Title = Unit save of LU 37 without VE
DSAVE :
```

## User Error Handling (UE)

**Purpose:** Sets the user error handling mode and defines how user errors are handled.

**Syntax:** UE[ , <ON/OFF>

- |     |  |
|-----|--|
| ON  | You are prompted for corrective action if a recoverable error occurs. The initial setting is ON. |
| OFF | The default action is taken, as described in the following sections.                             |

**Description:**

If you correct the error, command execution continues.

You may also use UE to see whether user error handling mode is ON or OFF by entering the command without any parameter. For example:

```
DSAVE : ue                                (Displays current UE mode)
User Error handling = ON
DSAVE : ue off                             (Turns UE off)
User Error handling = OFF
DSAVE :
```

## The Save Operation

Before the save operation begins, all the disk LUs to be saved are verified as legal disk LUs. If any LU is illegal, DSAVE displays the following error message:

```
Error - LU xx not a supported disk.
```

If you are running DSAVE interactively, you'll be prompted with additional messages; otherwise, it terminates.

If either the removable disk or the source disk LU goes down during the save operation, DSAVE issues the following message:

```
LU xx is down.
Correct problem and UP device or use BR to break.
```

System error messages about the downed device may also display. The sections that follow describe the various elements of a save operation.

## Unit Save LU Checking

If the SA command specifies a unit save operation, DSAVE locates all the disk LUs associated with the disk unit pointed to by the disk LU you entered, and proceeds in the same way as an LU save.

## Disk Locking

If you do not specify NL for the save, each disk LU being saved is locked just before it is saved, and remains locked until the save completes unless you specified VE, when it remains locked until after the save is verified.

If you do not specify NL, and the disk to be saved cannot be locked (it is currently locked to another program), if user error handling is ON, you are prompted with the following message:

```
Error - LU xx locked to (program name)
Correct problem and type GO to continue, SK to skip or BR to break.
```

If user error handling is OFF, the SA command is skipped and the disk LU is not saved.

## Removable Disk Overwriting

At the beginning of a save operation, the removable disk in the media specified is checked for a valid file system disk header or a DSAVE format media. When a valid CI disk volume header or FMGR disk cartridge header is found, the following message is displayed:

```
Warning - Removable disk media has a valid file system disk header.
Do you want to overwrite the media (Y/N) ?
```

If you respond "Y," the removable disk to be used in the save operation is initialized. Information on that disk stored in CI or FMGR file format is destroyed. If you respond "N," DSAVE prompts for another removable disk media with the following message:

```
Insert another removable disk media and type GO to continue or BR to break.
```

The removable disk is checked for a valid DSAVE format (that is, it does not have a valid file system disk header, and it has a valid DSAVE header record for the first save file on the removable disk). If a valid DSAVE format is present and the disk is from the same media set of the current save operation, DSAVE issues the following message:

```
Error - Cannot overwrite removable disk media in the current media set.
Insert another removable disk media and type GO to continue or BR to break.
```

If the disk has a valid DSAVE format but is from a different media set, the information from the first header record on the removable disk is displayed, along with this message:

```
Warning - Removable disk media is a DSAVE format media.
Do you want to overwrite the media (Y/N) ?
```

When you enter “Y,” the media is used in the save operation. If you enter “N,” the following message appears:

```
Insert another removable disk media and type GO to continue or BR to break.
```

If you enter BReak, the save operation terminates and you are returned to the DSAVE prompt.

## Save File Records

DSAVE files contain two types of records, header and data. A header record at the beginning of each save file contains information about the save. The user information included in the header record is shown below:

```
Version of DSAVE used
Time stamp of save operation
File number (relative to beginning of the media set)
Save options specified
LU of disk area saved
Number of tracks of the LU saved
Section number (used when a file is saved on more than one
removable disc)
Status (end of save, continues on next removable disk with new
fixed disk LU, or continues on next removable disk with current
fixed disk LU)
```

The data record is a variable length record that contains physical images of the portion of the disk being saved. A more detailed explanation of these records and the DSAVE removable disk media format is provided in the section on “DSAVE Removable disk Media Format,” later in this chapter.

## Saving to the Removable Disk

When a save operation begins, the disk is locked unless you selected the NL option. disk data is read from the disk LU and written to the removable disk until the end of a disk LU is reached or until the removable disk is full. The header record is then written.

If the removable disk is full, the portion of the save operation that is on the removable disk is verified if you selected the VE option. Then the following message is issued:

```
Insert next removable disk media and type GO to continue or BR to break.
```

The header for the last section of the last save file is marked with the “end-of-save operation and end-of-media” flag before it is written to the removable disk media. This completes the save operation.

## Verification

When you select the VE option to use with the SA command, the save operation becomes a two pass process. During the first pass (save), disk LUs are saved to the removable disk. In the second pass (verify), the removable disk is repositioned to where the save began on the removable disk, and the following messages are displayed (the second message is displayed for each save file on the media.):

```
Verifying removable disk media.  
Verifying save of LU xx to file x.
```

For each save file, the checksum of the header record is verified. If a checksum error occurs while verifying the header record, the following message is displayed:

```
Error - Header record verify failed.
```

The save operation continues, but you should use the LH command to check the header records of the LU in question and the one immediately after it. If the header records are corrupt, repeat the save operation for these LUs.

The data records are verified by reading the data records from removable disk into one buffer, and the corresponding data on the source disk LU into a second buffer. The two buffers are then compared, bit-by-bit. If a data record comparison fails, the following is displayed:

```
Error - Data record verify failed.
```

If a verify error occurs on a save file, the appropriate error displays and the rest of the section of the file is skipped. The verify continues with the next file.

## Break Detection

DSAVE detects a break whenever you enter the BR command in response to a prompt which has BR as one of the responses, or when you enter a CI, CM, or FMGR break command. When a break occurs, DSAVE terminates the current save operation, unlocks all LUs that were locked (except the removable disk LU), and attempts to clean up the save operation.

If DSAVE is filling a removable disk media with save files when it detects the break, DSAVE stops copying disk data from the disk LU being saved and writes the header record with a status that indicates the save operation was broken and the save file is incomplete.

If the media LU is down, the following message is displayed:

```
Warning - Volume is corrupt.
```

If DSAVE cannot clean up (for example, the removable disk media LU is down, or you aborted DSAVE), a corrupt DSAVE media and media set result. The removable disk is corrupt because the last header on the media is NOT marked as end of media set.



If DSAVE is not filling a removable disk media when it detects a break (for example, when the volume is being verified or when prompting for another media), DSAVE takes the following steps:

1. Checks if the media LU is down; if so, DSAVE displays:

```
Warning - Media set corrupt.
```

2. Checks whether the media is a DSAVE format media; if not, DSAVE displays:

```
Error - Non#DSAVE removable disk media.
```

3. Checks if the media is from the current media set; if not, DSAVE displays:

```
Error - Removable disk media is from the wrong media set.
```

4. Checks if the media is the last volume written; if not, DSAVE displays:

```
Error - Removable disk media is volume xx.
```

In case of an error, the following message is displayed after the error message:

```
Insert volume xx of the current media set and type GO to continue or BR to break.
```

When you enter GO, the program repeats steps 1 through 4 for the new media. If you enter a break, the following message displays:

```
Warning - Media set corrupt.
```

In this case, if DSAVE cannot clean up, the result is a corrupt DSAVE media set, because the status of the last header record on the last DSAVE removable disk media of the set was not marked with "end of media set."

If the last save file on the volume was to be continued on the next removable disk media, it is corrupt because the status of the header record was not marked "save of disk LU incomplete."

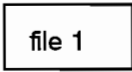
## Loading DSAVE

Use the LINK loader and the LINK command file #DSAVE to load DSAVE online. The DSAVE software and the LINK command file are included in the RTE-A Master Software. No special requirements are needed. DSAVE needs at least 17 pages of memory (including an extra 250 words for buffering). The recommended size is 32 pages.

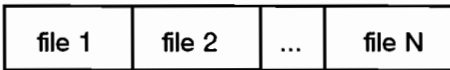
# DSAVE Removable Disk Media and File Formats

## Removable Disk Media Format

The DSAVE removable disk media format for a single LU save is shown below.

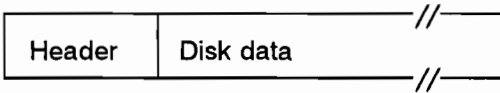


A unit or multiple LU save is a series of single LU saves. A unit save starts with the lowest numbered LU on the unit and proceeds through the highest numbered LU on the unit. In a multiple LU save, the LUs are saved in the order in which they were specified in the SA command. The DSAVE removable disk media format for unit and multiple LU saves is shown below.



## Removable Disk Media Records

The DSAVE file has a fixed length header record followed by a variable length data record.



## Header Records

The format of the header records is shown below.

Word	Content
0-1	Checksum
2	Length of header record
3	Program revision number (numeric)
4-10	Time stamp (ASCII, <000000.0000>)
11-13	Backup utility program name (DSAVE)
14-16	Media set ID (time stamp)
17	Volume number of media
18	Save file number (relative to beginning of media set)
19	Disk LU saved
20	Numeric device type of the disk LU
21	Select code of the disk LU
22	Device address of the disk LU

Word	Content
23	Unit number of the disk LU
24	Volume number of the disk LU
25	Cylinder starting
26	Head block
27	#Surfaces for CS/80
28	#Tracks
29	#Spares
30	#64 word sectors/track
31	Reserved
32	Save file section number
33–34	Disk block number where data for this section starts
35–37	Time stamp when this LU was saved
38	Options selected for the save of the disk LU(s)
39–46	User account name where backup was scheduled
47–66	Title specified for the save operation
67	Total number of tracks for the media LU
68	Sectors per track for the media LU
69–71	Time stamp when the save operation began
72	Header status word (described below)
73–74	Length of the disk data in this section
75	Saved disk data word from “directory”
76–77	Block address of the next header record
78–79	Block address of the previous header record
80	First save file number in save operation
81	Last save file number in save operation
82	First volume number of save file
83	Last volume number of save file (calculated)
84–127	Unused (reserved)

Header record status word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOS	LSK	LIN	SBR	r	r	r	r	r	r	r	r				section status

- EOS – End of the media set.
- LSK – The save of the disk LU was skipped.
- LIN – The save of the disk LU is incomplete.
- SBR – The save operation was broken.
- r – reserved (zero).

Section status:

- 1 – Save operation continues on the next volume with a new disk LU.
- 2 – Save operation continues on the next volume with the current disk LU.
- 3 – Save operation continues on the current volume with a new disk LU.

## **DSAVE Error Messages**

DSAVE error messages, error messages shared by DSAVE and DRSTR, and error messages generated by DRSTR are all presented following the discussion of DRSTR.

# DRSTR Physical Restore Utility

DRSTR is used to restore data from removable disk media saved using DSAVE. DRSTR restores one disk LU, a group of disk LUs, or an entire disk unit.

## Calling DRSTR

Invoke DRSTR with the following runstring:

```
CI> [RU, ]DRSTR[ ;command[ ;... ;command ]]
```

If you do not enter a command in the runstring, DRSTR prompts you interactively, as follows:

```
DRSTR :
```

You may separate parameters within each DRSTR command with commas or blanks, and you may combine multiple commands in the runstring or in response to the prompt, separated by semicolons (;). In the runstring, you must precede each semicolon with a backslash (\) so CI will not interpret the semicolon as a CI command delimiter. You may also use the underscore ( \_ ) character to separate parameters in the runstring; DRSTR converts them later to blanks. For example:

```
CI> drstr\;me_32\;lh\;re_4:20_7
```

Commands are executed in order, from left to right. If any command causes a serious error, execution terminates with that command, and subsequent commands are ignored.

## DRSTR Commands

DRSTR uses commands that let you terminate DRSTR, list all the commands and their parameters, list header information, define a list file or device, define removable disk media, specify a restore operation, and select user error handling mode. Table 3-11 summarizes the DRSTR commands, which are described in the following sections.

**Table 3-11. DRSTR Commands Summary**

Commands		Description
<b>Information Command</b>		
<b>HElp</b>		Lists all DRSTR commands and parameters
<b>Configuration Commands</b>		
<b>MEdia</b>	LU	Defines removable disk media
<b>REstore</b>	file#: lu	Specifies a restore operation
<b>UE</b> User Error Handling	ON/OFF	Selects user error handling mode
<b>Listing Commands</b>		
<b>LH</b> List Header	file#	Lists header information from removable disk
<b>LL</b> List File or Device	file descriptor/lu	Opens a list file or list device to which information is logged in addition to the terminal
<b>Exit Commands</b>		
<b>ABort</b> <b>ENd</b> <b>EXit</b>		These three commands terminate DRSTR

### **Abort, End, and Exit (AB) (EN) (EX)**

Purpose: Any one of these commands terminates DRSTR.

Syntax: AB, EN, or EX

Description:

Upon termination, all disk LUs are released and any list file or device closed.

### **Help (HE)**

Purpose: Lists all DRSTR commands and parameters.

Syntax: HE, ??, or ?

## List Header (LH)

**Purpose:** Lists header information for a file from the removable disk.

**Syntax:** LH[ ,<file #>

file #            The file number on the removable disk media.

**Description:**

If the file number you specify does not exist on the current removable disk, the header record of the first or last file on the disk is listed, depending upon the number specified.

If you do not specify a file number, the default is to list the header information for the next save file (or the first save file if this is the first time header information is being listed from the removable disk).

If the number is smaller than that of the first save file, the header record of the first save file is listed. If the number is greater than that of the last save file on the disk, the header record of the last save file is listed.

---

**Note**            All file numbers specified are relative to the beginning of the first DRSTR removable disk, if there is more than one. If the save file for a disk LU crosses removable disk, it has the same file number on all removable disks.

---

## List Device (LL)

**Purpose:** Opens a list file or list device to which terminal displays and error messages are to be sent in addition to the terminal.

**Syntax:** LL[ ,<file descriptor/lu>

file descriptor    The list file.

lu                 The list device.

**Description:**

A defined list file is created if it does not already exist, and a list device is opened. Any previously opened list file or device is closed first. If you do not specify a file or list device, the currently defined list file or device is closed.

If you specify an existing file, a duplicate file error occurs. In this case, if user error handling is ON, an error message is issued and the LL command terminates; otherwise, the command is skipped.

If you specify a closed list file or a device that is down and user error handling is ON, DRSTR prompts you for corrective action or for a break before continuing. If user error handling is OFF, the executing command default is to break.

If user error handling is OFF and a break is caused by one of the above mentioned errors, the list file or device is closed and the executing command continues to execute.

## Media (ME)

**Purpose:** Defines the removable disk LU number used by DRSTR commands to access the removable disk.

**Syntax:** ME[ ,lu]

lu                    The logical unit number of the removable disk medium. If lu is omitted, either the previously defined LU or zero is displayed.

**Description:**

You may also use ME to examine the currently defined LU. You must enter ME before a restore operation.

The LU you specify must not be mounted. It is locked and remains locked until DRSTR terminates or you define another LU.

An error occurs if DRSTR accesses a removable disk not defined by the ME command. This typically happens when you enter an RE or LH command without or before the ME command.

To find out the LU numbers associated with the removable disks, consult the system generation answer file or check with a superuser (or your system manager).

## Restore (RE)

The RE command restores data from the removable disk to the LU specified, either file-by-file for one or more files (“LU restore”), or a group of files from the same drive unit (“unit restore”). It has two formats, one for each type of restore, described separately below.

### File-by-file (LU) Restore:

**Purpose:** Specifies a file-by-file (LU) restore.

**Syntax:** RE,<file #:lu>[ ,...,<file #:lu>[ ,VE]

file #:lu            The file number on the removable disk and the disk LU where the file is to be restored.

If you specify one file number, then all the parameters must specify a file number. If you do not specify a file number, the default file is 1 for the first disk LU specified, 2 for the second, and so on.

The format for entering the <:lu> parameter is :lu (the colon is required). If you do not enter an LU, the destination disk is determined by examining the header record of the save file for the disk LU saved. If you do not enter anything for <:lu>, the parameter is specified as <file #> (no colon).



Description:

Only one of the above forms is allowed; combinations of file number and LU defaults result in a DRSTR error. For example, the forms

```
re, 3:20, :22
```

or

```
re, 6, :30
```

cause this message to display:

```
Error - Illegal file #:lu parameter combination.
```

The following combination is acceptable:

```
re,4:30,7
```

---

**Note**

All file numbers specified are relative to the beginning of the first removable disk, if there is more than one. If the DSAVE file for a disk LU continues on another disk, it has the same file number on both removable disks.

---

**Unit Restore:**

Purpose: To restore a group of files from the same drive unit.

Syntax: RE[,<file #>UN[,VE]]

file #           The number of the first save file of the group of save files to be restored from the removable medium. If omitted, file 1 is assumed.

Description:

The restore operation only restores the specified DSAVE file and any subsequent files from the same disk unit saved in the same operation.

RE command options are described in the next section. The restore operation is explained in detail in the section, "The Restore Operation," later in this chapter.

**RE Command Options**

The RE command options parameter lets you specify whether you want to have verification of the data when it is restored, and whether you want to specify a unit restore.

Options may be specified in any order. Table 3-12 summarizes the RE command options.

**Table 3-12. RE Command Options Summary**

Options	Description
VErify	Verifies transferred data when a restore operation is performed
UNit Restore	Specifies a unit restore, rather than a file-by-file restore

### **Verify Data (VE)**

You may have transferred data verified when the restore operation is performed. To select verification, simply enter VE in the RE command option parameter, as shown above in the RE format.

### **Unit Restore (UN)**

The UN option specifies a unit restore beginning at the file you enter, or file 1 if you do not enter a file number, as described above.

### **User Error Handling (UE)**

**Purpose:** Sets the user error handling mode or displays the current mode if no parameter is entered.

**Syntax:** UE[ , <ON/OFF>

**ON** You are prompted for corrective action if a recoverable error occurs. The initial setting is ON.

**OFF** The default course of action is taken, as described in the discussion of each command for default information.

## **The Restore Operation**

All disk LUs being restored are first checked for valid disk LU numbers. Checking may occur when you enter the RE command, or when each save file is encountered during the restore.

The sections that follow describe the various elements of a restore operation.

## Checking the Disks

Disk LUs entered in the **RE** command are checked when you enter the command. An illegal disk LU causes the restore to terminate at once, and the following error message is displayed:

```
Error - LU xx not a supported disk.
```

Disk LUs read from the save files are checked when each save file is encountered during the restore. If any LU is illegal in the current system, the following warning message is displayed:

```
Warning - LU xx not a disk in current system.
```

If user error handling is **ON**, **DRSTR** prompts you to break, skip this restore, or enter the disk LU to be restored. If user error handling is **OFF**, the restore of this file is skipped.

## Sectors per Track Must Match on Disks

**DRSTR** only restores a **DSAVE** file on removable disk to a similar disk, one with the same sectors per track configuration. If the sectors per track are different, the following error is displayed and the restore of that LU is skipped:

```
Error - Sectors per track mismatch.
```

## Total Number of Tracks May Differ

The total number of tracks of the saved disk (source) need not be the same as the number of tracks on the disk being restored (destination). If there is a difference, one of the following messages is displayed:

```
Warning - Saved disk has more tracks than target LU xx.
```

or

```
Warning - Saved disk has less tracks than target LU xx.
```

After the warning message is displayed, you will see the following messages:

```
Saved disk tracks = xx  
Target disk tracks = xx
```

If user error handling is **ON**, you are asked:

```
Restore this disk (Y/N) ?
```

If you respond by entering “**N**,” the restoration of the disk is skipped. If you respond “**Y**,” the disk is restored. When you restore the disk, be aware that the destination LU size is not the same as the **DSAVE** file on the removable media. A smaller destination LU results in the restoration of only part of the **DSAVE** file.

A destination LU of a different size restores the save file, but if there is a file system on the LU, it is not accessible, because the file system directory information is at the last track of the LU. The directory information of the restored file system is thus not at the proper location.

If user error handling is OFF, the default is to restore the disk.

## Disk Locking

Each disk LU being restored is locked when the restore operation begins, to ensure the integrity of the restored data. The disk remains locked either until the restoration is complete (if you did not select VE) or until after verification (if VE was specified).

If the destination disk cannot be locked (for example, it is currently locked to another program), and user error handling is ON, the following message is displayed:

```
Error - LU xx locked to (program name) Correct problem and type GO
to continue, SK to skip or BR to break.
```

If user error handling is OFF, the restore of that disk is skipped.

When a disk is locked, it is checked to see if it is mounted; if it is, the following message is displayed:

```
Error - LU xx is mounted.
```

If user error handling is ON, DRSTR prompts you as follows:

```
Correct problem and type GO to continue, SK to skip or BR to break.
```

If user error handling is OFF, the restore of that disk is skipped.

## Order of Disk Restoration

Disks are restored in the order in which they were saved to removable disk media, not as they are specified in the RE command.

## Unit Restore

For a unit restore, you must first enter the removable disk that contains the first section of the save file specified in the RE command; otherwise, DRSTR prompts you with one the following messages:

```
Error - First file is x section y - searching for file z.
```

or

```
Error - Last file is x section y - searching for file z.
```

DRSTR then prompts you as follows:

```
Correct problem and type GO to continue or BR to break.
```

---

**Note**

If the next save file to be restored from is NOT on the current removable disk, you can insert the removable disk volume that contains the first section of the save file. (You may skip over the removable disk volumes that will not be used during this restore operation.)

---

## Restoring Fixed Disks from Removable Disks

Each removable disk must be a valid, DSAVE format removable disk. If it is not, the following message is displayed:

```
Error - Non-DSAVE ^removable disk media.  
Insert another removable disk media and type GO to continue or BR to break.
```

If the save file to be restored is not on the removable disk, one of the following messages is displayed:

```
Error - First file is m section n - searching for file x section y.
```

or

```
Error - First file is m section n - searching for file x section y.
```

After the appropriate error message is displayed, DRSTR prompts you as follows:

```
Correct the problem and type GO to continue, SK to skip or BR to break.
```

## Header Records

Information for the restore of a disk is obtained from the system and the header record. Data from the header record is used to control the restore. If a checksum error occurs on the header record, the restore of that disk is skipped and you are asked for another removable disk.

Because the location of the next header record on the removable disk is in the header record, any save file(s) after the current one on this removable disk media must be skipped. When this occurs, the following error message is displayed:

```
Error - Header record checksum.  
Insert another removable disk media and type GO to continue or BR to break.
```

When the actual restore of the disk LU begins, information from the header record of the save file is logged.

## Restoring the Data

For each save file, disk data from the removable disk is restored to the disk LU until the destination disk is full or the end of the disk data in this section of the save file is reached. If an error occurs while reading a part of the data record from the removable disk during the restore, the following error message is displayed:

```
Error - Data record read error.
```

The range of disk block numbers is determined and reported in the form of the following messages:

```
Data targeted for disk blocks x to y.  
Continuing.
```

In any case, the restore continues. When the end of this section of the save file is reached, or the target disk LU is full, the disk data is verified (if you selected VE). If the restoration is not complete (that is, the target LU is not full and this was not the last section of the save file), DRSTR prompts:

```
Insert next removable disk and type GO to continue or BR to break.
```

If the restoration of the target LU is complete, the next save file to be restored from is searched for on the current removable disk. When it is found, the restore begins. If it is not found, DRSTR prompts you as follows:

```
Insert next removable disk and type GO to continue or BR to break.
```

If the removable disk drive or destination disk goes down during the restore, the following message is displayed:

```
LU xx is down.  
Correct problem and UP device, or use BR to break.
```

You may also see system messages about the device that is down.

When the restore of a disk LU completes, the following message notifies you:

```
Restore of file n to disk LU xx complete.
```

## Verification

When you select VE, the restore operation is a two pass process. During the first pass (restore), data is read from the removable disk and written to the fixed disk. In the second pass (verify), the data is re-read from removable disk into one buffer, and the corresponding data restored to fixed disk is read into a second buffer. The data in the two buffers are compared bit-by-bit.

If a mismatch between the two buffers occurs, the following message displays, and verification continues to the end of the disk or until three verify errors occur on that disk restore, causing the process to abort:

```
Warning - Possible data loss in disk block range n to n.
```

## Break Detection

If DRSTR detects a break, it terminates all ongoing operations and unlocks all LUs except the removable disk drive LU.

## Loading DRSTR

Load DRSTR online, using the LINK loader and the LINK command file #DRSTR. The DRSTR software and the command file are included in the 92077A RTE-A Master Software. DRSTR requires 17 pages of memory (including an extra 250 words for buffering). The recommended program size is 32 pages.

## DSAVE and DRSTR Error Messages

The sections that follow list 1) all the messages that DSAVE generates, 2) error messages that are shared by DSAVE and DRSTR, and 3) the error messages that DRSTR generates.

### DSAVE Error Messages

#### **Error – Cannot overwrite removable disk media in the current media set.**

The removable disk media has a valid DSAVE format and is from the same set of the current save operation.

#### **Error – Data record verify failed.**

Retry the operation. If the error persists, try another media or disk drive unit.

#### **Error – Illegal unit save.**

You entered two or more LUs in a unit save operation. Only one LU is required.

#### **Error – Media LU must not be in the list of LUs to be saved.**

The media LU was specified as one of the LUs to be saved. Retry the operation and specify the correct LUs.

#### **Error – No disks to save.**

The LU parameter was missing from the SA command.

#### **Error – Only xx save file number available.**

The save file number specified is greater than the number of save files available on the removable media.

#### **Error – Too many LUs specified.**

Only 63 LUs are allowed in the save operation. Save the excess LUs in another operation.

**Error – Unknown command; use ?? for help.**

This is probably due to a typing error. Reenter the correct command.

**DSAVE and DRSTR Shared Error Messages**

**Error – Bad header record detected.**

Header record verification failed. Retry operation.

**Error – Bad LU.**

Illegal LU specified. For example, the LU specified in the LL command must be an I/O device LU.

**Error – Disk retries exhausted.**

Unsuccessful disk access. Retry the operation with another disk unit.

**Error – File number expected.**

A file number must be specified in the LH command.

**Error – <FMP error message>.**

A file system error occurred while accessing the list file or device.

**Error – Header record verify failed.**

DSAVE or DRSTR could not verify the header record of an LU, perhaps because of bad media. Use the LH command to check the header in question. If an invalid header is found, this LU and the one after it are not recoverable. To save them, repeat the save operation for those LUs.

**Error – Illegal removable disk media LU.**

The LU specified in the ME command is not a removable disk LU or is not found in the system. Check the system LU assignment and reenter the ME command.

**Error – IOxx on LU xx <error description>**

A device I/O error was detected. The I/O error messages can be found in the *RTE-A Quick Reference Guide*, part number 92077-90020, or the *RTE-A User's Manual*.

**Error – LU xx is a mirrored volume LU.**

The physical paired disk LU associated with the mirrored volume LU must be specified, instead of the mirrored volume LU. (Refer to the *Data Pair/1000 Reference Manual* for more information.)

**Error – LU xx is down.**

The specified I/O device is down. Ensure that the device is operational, and bring it up.



**Error – LU xx is locked to <program name>.**

The specified LU is locked by a program. That program must be terminated before reentering a command.

**Error – LU xx is mounted.**

Dismount the LU to be saved or restored, then retry the operation.

**Error – LU xx not a supported disk.**

The specified LU is not a supported disk in the RTE-A system. Check the system configuration for the proper LU number.

**Error – Non-DSAVE removable disk media.**

Information contained in the removable disk media is not in DSAVE format. Saving LUs on that media will destroy the existing information.

**Error – Not enough memory (size program).**

DSAVE or DRSTR requires at least an extra page of memory. Size up the program one page larger either temporarily, with the SZ command from CI, or permanently, with Link.

**Error – Option expected.**

Usually a typing error. Anything entered after an option must be an option.

**Error – Removable disk media must not be a mirrored volume LU.**

The LU specified in the ME command is a mirrored volume LU. Check the system LU assignment, and reenter the ME command. (Refer to the *Data Pair/1000 Reference Manual* for more information.)

**Error – Removable disk media LU undefined.**

A save or restore operation initiated before a media LU was defined. Specify a removable disk media LU with the ME command, and retry the operation.

**Error – Removable disk media LU = xx.**

The removable disk media LU number is specified in other commands, for example, in the LL command as the list device. Check the system configuration and redefine the LUs.

## **DRSTR Error Messages**

**Error – A duplicate save file number detected.**

Typically, the same file number was specified, such as “re,5:20,6,5.”

**Error – Bad LU specification.**

This error usually occurs when non-numeric characters are entered in the LU parameter field. Reenter the command with the proper LU number.



**Error – Bad save file number specification.**

Typically, a typing error; for example, re,5xx or re, t5.

**Error – Data record read error.**

The verify was unsuccessful during the restore. Retry the operation.

**Error – Illegal file #/lu parameter combination.**

In a file-by-file restore, the file#/lu parameters must be consistent in all parameter specifications. They must be in one of the following forms: : <file #>, <:lu> or <file#:lu>.

**Error – Illegal unit restore.**

Usually, this means the LU parameter is missing from a unit restore operation.

**Error – LU expected after “:”.**

The LU parameter is missing from a multi-file restoration.

**Error – LU must be greater than zero.**

Probably non-numeric characters were entered instead of numbers. Enter the correct LU number.

**Error – Media LU must not be one of the disk LUs to be restored.**

Check the system configuration, and enter the correct LU numbers.

**Error – No LU or save file number specified.**

RE command parameters are missing. Refer to the discussion of the RE command earlier in this chapter for the correct command syntax.

**Error – Save file number must be greater than zero.**

This error is typically caused by a typing mistake. Reenter the command, and specify the proper file number.

**Error – Save of disk LU was skipped during save operation.**

The LU you specified cannot be found in the removable media.

**Error – Sectors per track mismatch.**

The source and destination disks have a different number of sectors per track. The restore operation is performed, but the results are uncertain. If the destination disk has a valid file system, it is no longer accessible to the system.

**Error – Too many save file/lu parameters specified.**

You may enter a maximum of 63 file #/lu parameters. However, the command string is subject to the file system command string length restrictions. Refer to the *RTE-A User's Manual* and to Appendix A in this manual (CI and FMGR respectively) for details.

**Error – Unknown parameter.**

Probably a typing mistake was made, or the wrong command syntax entered. Refer to the DRSTR command description in this chapter for the correct command syntax.

**Error – Wrong removable disk media set.**

A removable disk from the wrong set was inserted. Remove it, and insert the correct disk.

## 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-A System Manager's Manual*, part number 92077-90056, for the format of the file directory entry):

<b>Word</b>		<b>Meaning</b>
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 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.

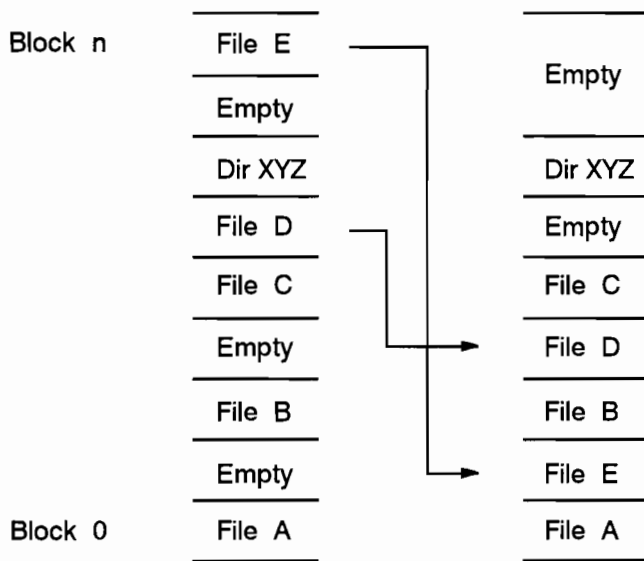
### The Packing Process

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 can 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 2 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
```

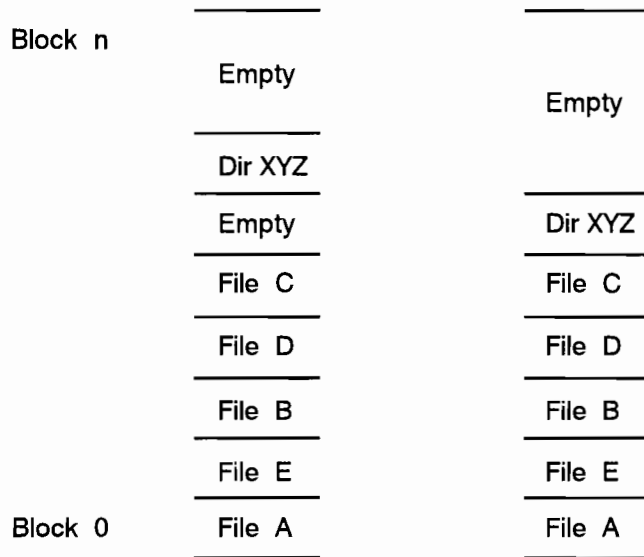
5. Purge the old directory, freeing the disk space. (Note that you must include the type extension .DIR when purging a directory.) If the directory is not empty, it cannot be purged. Move the files back to XYZ, purge TEMP, and terminate the action.

```
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 on "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 4-1 below 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 4-1. MPACK Options Summary**

Options		Description
<b>Compacting and Reporting Options</b>		
+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
<b>Packing Options</b>		
+P		Pack the disk LU
+OK		OK to overlay data during pack
<b>Logging Option</b>		
+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 4-1 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 reporting occurs.

### Remove Extents from Files (+R)

The +R option lets you recreate files that currently have extents, so that the resulting files have 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 the directories.

The format for using +R is:

```
mpack <mask> [options] +r
```

### **Truncate Unused Blocks from Files (+T)**

The +T option lets you truncate types 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 any file 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 any file 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 any file 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 included, 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 the disk. 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 some 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] +l <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,?[?]) causes FREES to display usage information and examples as shown below.

Syntax:

```
frees [-v] [+l:lu] [-g] [+t|+m] [+h|+d] [DiskLu] [DiskLu:DiskLu] ...
```

-v	inhibit the inverse video bar graph.
+l:lu	sends the listing to the given LU.
-g	bar graphs are not relative to the largest disk.
+t	report disk sizes in number of tracks.
+m	report disk sizes in megabytes.
-s	inhibit sort, report will be in cartridge list order.
+h	sort by largest hole size.
+d	sort by disk size.
+q	quiet mode, return the free space information to the scheduling program in \$return_s.
+qd	return the disk size in \$return_s.
+qf	return the free size in \$return_s.
+qh	return the largest hole size in \$return_s.
+qr	return the reserved size in \$return_s.
+q%f	return the percent free in \$return_s.
+q%m	return the percent max in \$return_s.
DiskLu	specifies a disk LU to display; it can be repeated.
DiskLu:DiskLu	specifies a range of disk LUs to display; if no disk LU is specified, all CI volumes will be 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:

```

ru,frees 10 11

```

LU	Total	Resvd	Free	Max Free	%Free	%Max
11	100000	0	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.

```
ru,frees 10 11
```

LU	Total	Resvd	Free	Max Free	%Free	%Max
11	100000	0	80000	40000	80	50
10	100000	1538	80000	40000	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 owners, 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 2 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 owners of remote files, 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 then reporting any inconsistencies. The second function is to fix some of the inconsistencies found by the verification.

Two kinds 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 kinds 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). 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 corrupted 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.

- Bit map corruption. Occasionally a bit map can become corrupted 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:

```
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 – do fixes without asking each time

Running FVERI with a ? or ?? (RU,FVERI,?[?]) causes FVERI to display usage information as shown above.

If neither an LU or 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.

## 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, so that 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 backup. 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.

- 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 be 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 worse 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, to complement 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 will work better if 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 which 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 some other 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.

## Formatting Utilities

---

RTE-A formatting utilities FORMC, FORMF, FORMT, and FORMA are used to format disk volumes or tape, verify the integrity of disks and CS/80 CTDs, and spare out bad areas of media. This chapter describes these utilities, which are shown in Table 5-1 below. It is a good idea to check Table 5-1 before you format any disk, to make sure you are selecting the proper utility for your task. (For SCSI devices, refer to the VSCSI utility in the *HP 12016A SCSI Installation and Reference Manual*, part number 12016-90001.)

**Table 5-1. RTE-A Formatting Utilities**

FORMC Supports	FORMF Supports	FORMT Supports	FORMA Supports
7907 7908 7911 7912 7914 7933 7935 7942 7945 7946 9122C/D 9133H 9144 CS/80 248x hard disk with floppy, with 12122A	9121 S/D 9133A/9134A 9133B/9134B 9895 9130 K (Model 6 built-in disks) 248x Integrated 3 1/2" microfloppy disk with 12022A	7906 H/M 7910 H/M 7920 H/M 7925 H/M 9130 K (Model 6 built-in disks)	248x Integrated hard disk
Runs under these operating systems:			
Online: RTE-6/VM RTE-A  Offline: RTE-A	Online: RTE-A	Online: RTE-6/VM  *Offline: RTE-A	Online: RTE-A
* FORMT is supplied only in relocatable form. The instructions in the section on generating a FORMT system must be followed in order to create an offline version of FORMT.			



## General Formatting Information

A disk cannot be used (normal reads and writes cannot be made to the disk) until it is formatted. Formatting is the simplest form of initialization. (Note that the terms “formatting” and “initializing” are not interchangeable, as there are differences between the two processes.)

When a disk is formatted, the formatting process writes every block on the disk. All information on a disk is written in blocks of 256 bytes (128 words). The block number is usually called a “sector number” in disk literature; however, since an RTE sector is usually 128 bytes (64 words), the term block is used in this discussion.

Each block has three components, the preamble, user data area, and postamble. The disk is written as described below.

preamble	user data – 256 Bytes	postamble
----------	-----------------------	-----------

Each block is tagged with an address (preamble) which initially points to itself, but changes if the block is defective or a spare.

If the disk is normal, the preamble address identifies the block. If the disk is defective, the preamble address is either zero or the address of the next block, depending upon the type of drive. In either case, the block is made functionally invisible.

The formatting process writes test data in the user data area, then reads it back to make sure the block is normal. The postamble is written with error checking information calculated from the test data. If the test passes, the block is normal. Each block is tested several times.

Hard disks supplied by Hewlett-Packard are formatted before shipment. Since a hard disk should be formatted only at the factory to ensure interchangeability of disk packs, hard disk formatting capability is not included in the RTE-A formatting utilities. Hard disk initialization assumes the disk was formatted at least once.

## The Preamble

The preamble has three functions:

- A synchronization field provides timing signals to indicate the start of the block.
- Bits indicate the status of the block. These bits are used differently or not at all depending upon the actual disk type (hard or soft) and the disk design.

The preamble is more complex for a hard disk than for a flexible disk. The status portion of the preamble includes three bits:

S bit	Indicates that the track is a spare for a defective track. Only hard disks supplied by HP permit sparing.
P bit	Indicates a protected track, one that is “read only” unless the FORMAT switch is activated on the disk drive.
D bit	Indicates that the track is defective and should not be used.

For flexible disks supplied by HP, only the D bit is used in the preamble.

- An address gives position information. The address field in the preamble has three basic components:

- Cylinder number
  - Head number
  - Block number

In most cases, the address associated with the block is the address of the block itself; that is, it points to itself. The exceptions are:

Hard Disk	If the D bit is set (track is defective), the address points to the track used as a spare.  If the S bit is set (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 is defective), the address is not relevant (may be anything).

## The Postamble

The postamble provides information that the disk drive needs for error checking during which time it determines the validity of the data in the user data area.

## Initializing and Sparing Hard Disks

Disk initialization uses the track map table in RTE-6/VM or the driver parameters in RTE-A to determine the starting cylinder, head number, number of tracks and number of spare tracks for each LU. Tracks in the spare pool are used only as replacements for the defective tracks and are otherwise unused.

### Sparing vs Skipping

The RTE utility programs declare an entire track bad if one bad block is found on the track during formatting. This method of management results in less head movement (hence faster access) when spared blocks (tracks) are encountered.

On a hard disk, once a track is spared subsequent seeks by the disk controller to that track reveal that the track is spared, and the disk drive automatically accesses the spare track instead of the bad one. This process is transparent to your software.

On a flexible disk, although the technique is not called sparing the effect is almost the same. Once the D bit is set for all defective tracks, they become invisible to your software. This causes the drive to skip bad tracks as if they did not exist on the disk.

For example, the outermost track is normally track zero. If the D bit is set for this track, then a command to access track zero causes this track (and any bad tracks that follow it) to be skipped. The first good track encountered becomes logical track zero.

If no bad tracks occur after formatting is complete, it does not matter whether sparing is provided. Sparing and skipping do not really have any effect on your programs. If a bad track develops when you use the disk, the hard disk sparing capability has the advantage that a simple recovery process is available when you run the FORMT utility. This is not possible with the flexible disk.

### Interleave Factor Calculation

When you format a flexible disk, RTE formatting utilities let you define the order in which consecutive logical blocks are located on the disk by specifying an interleave factor or fill number. The interleave factor is the number of physical blocks to skip between logical blocks.

With an interleave factor of zero, block  $n+1$  immediately follows block  $n$ . If the interleave number is 1, one block is skipped between blocks  $n$  and  $n+1$ . An interleave factor of two causes two physical blocks to be skipped between blocks  $n$  and  $n+1$ , and so on. Once a disk is formatted, the interleave is transparent to your software.

You can calculate the interleave factor using two numbers—the time between disk accesses done by the software, and the rotational speed of the disk. For a program that reads sequential blocks from the disk, the calculation is simple. First the processing time for each block is measured, then the read time per block is calculated. The read time per block equals the time of one disk rotation divided by the number of blocks per track.



For example, suppose 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, the time to move over a single block is found. The interleave number is then computed using the following formula:

$$\text{Processing time} \leq [(\text{interleave} - 1) * \text{sector read time}]$$

Thus, the next block rotates into the proper position for access just when it is needed.

Consider a disk that has the following rotation speed and number of blocks per track:

$$\begin{aligned} 360 \text{ RPM (rotations/minute)} &= 2.77 \text{ milliseconds/rotation} \\ 30 \text{ blocks/track} &\text{ means } 0.09 \text{ milliseconds/block} \end{aligned}$$

Suppose the program takes .15 mS to process each block:

$$.15 \text{ mS per block} / .092 \text{ mS per block} = 1.63$$

The next greater whole integer is 2, so in this case the optimal interleave factor is 2.

If the processing time is less than 0.09 mS, an interleave factor of 2 permits processing to finish before the next block becomes available.

If the processing time is between 0.09 mS and 0.18 mS, an interleave factor of 3 is appropriate.

An interleave factor of 1 means the disk must rotate all the way around before the next block becomes available (2.77 mS vs less than 0.09 mS).

An interleave value of 2 is the smallest value recommended with the current floppy disk controller. An interleave value of 1 causes the disk to wait a full revolution per block when a multiple block transfer is requested, because the transfer rate of the controller does not keep pace with the transfer rate of the disk.

Most user programs that access files define a Data Control Block (DCB) of 144 words, of which 128 words (1 block) are used for disk transfers. Therefore, you can compute a minimum interleave factor in terms of the amount of time to complete one EXEC request to write a block and return to the user program.

Alternatively, you can determine the interleave factor experimentally by trying several interleave numbers. For sequential access to the files with a DCB of 144 words, try an approximate number in the range of 4 to 6.

## CS/80 Definitions

Unless otherwise specified, the following definitions apply to CS/80 disks and cartridge tape drives throughout this chapter.

Block	The smallest unit of information, 128 words in length, that can be addressed by a CS/80 disk. It corresponds to a cylinder, head, or sector address.
Tape	The smallest unit of information, 512 words in length, that can be addressed by a CS/80 tape unit.
Volume	A separately addressable portion of the storage media on a given unit, corresponding to a fixed or removable platter of a disk unit or a tape drive.
Track	A subdivision of a disk LU as defined by the EXEC interface.

## CS/80 Disk Formatting (FORMC)

FORMC is an online maintenance program for CS/80 disk and tape drives. You can use it to format an entire disk volume or tape, verify the integrity of data on the device, and correct any errors detected. FORMC deals only with logical addresses; for offline use or if physical addressing of a device is desired, use the offline version of FORMC (!FORMC) or the CS/80 disk exerciser program, EXER. Refer to the *CS/80 Disk Exerciser Manual*, part number 5955-3462, or Appendix B, for details on the EXER utility.

### Calling FORMC

You may run FORMC interactively or in runstring mode, as described below. FORMC can also be called programmatically by issuing the appropriate program-scheduling EXEC call and entering the parameters in the runstring.

To call FORMC, enter the following runstring:

```
CI> [RU,] FORMC [,list LU], command, LU [,params]
```

- |         |   |
|---------|---|
| list LU | The LU of the list device to which FORMC directs messages generated during the specified operation. |
| command | One of the FORMC commands described below.  |
| LU      | The LU of the CS/80 disk or tape to be accessed by FORMC for the specified operation.               |
| params  | The command descriptions are provided below.  |

If you omit a required parameter or enter one incorrectly, FORMC exits with the messages:

```
/FORMC: ILLEGAL RUNSTRING PARAMETER  
/FORMC: ABORTED
```

If you do not enter any runstring parameters, FORMC runs interactively and prompts you for the commands, as follows:

```
/FORMC: CS80 DISK FORMAT UTILITY  
/FORMC: '?' WILL LIST THE LEGAL COMMANDS  
/FORMC: TASK?
```

When you enter a command, FORMC prompts for parameters.

## 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 Commands

FORMC commands are summarized in Table 5-2 below and described in the sections that follow.

Table 5-2. FORMC Commands Summary

Commands	Description
<b>Information Command</b>	
HElp ?	Displays a list of FORMC commands
<b>Configuration Commands</b>	
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
<b>Exit Commands</b>	
ABort ENd EXit	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.

---



## Format (FO)

Purpose: To format a disk volume or cartridge tape.

Syntax: `FO, media_LU [,interleave]`

`media_LU` The LU of the CS/80 disk or tape to be formatted. FORMC aborts with an `ILLEGAL RUNSTRING PARAMETER` message if you try to format a system disk or a disk that is currently mounted.

`interleave` The interleave factor (a decimal number between 1 and 32) to be assigned. The default is 1.

Description:

Detailed information about FORMC formatting is provided in subsequent sections in this chapter. See also the sections in the beginning of this chapter on “General Formatting Information” and “Interleave Factor Calculation.”

---

**Warning** The `FO` command causes an entire tape or disk volume to be formatted. If you specify a disk LU, the entire disk volume is formatted.

---

## Help (?)

Purpose: Displays all the legal FORMC commands.

Syntax: ?

## Spare (SP)

Purpose: Spares one or more defective blocks within a track of a CS/80 disk.

Syntax: `SP, media LU, track`

`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.

Description:

You may only use the `SP` command with CS/80 disks, including the CTD disk cache memory area. The disk cache is divided into four logical tracks; `SP` can be used to spare one of these cache tracks. (Refer to the *RTE-A Driver Reference Manual*, part number 92077-90011, for a description of the disk cache memory scheme.)

Sparing using `SP` is explained in detail in the section “The Sparing Operation,” later in this chapter.

## Verify (VE)

Purpose: Verifies the integrity of the specified disk or tape.

Syntax: `VE, media LU [,start, numb]`

<code>media LU</code>	The LU of the CS/80 disk or tape to be verified.
<code>start</code>	The number of the disk track or tape block at which verification is to begin.
<code>numb</code>	The decimal number of disk tracks or tape blocks to be verified. If you omit the <code>start</code> and <code>numb</code> parameters in the runstring, the entire disk or tape is verified.

### Description

Verification is a non-destructive process that checks the integrity of a specified disk LU or CTD tape by examining the checksum information for each block of the LU or tape.

## The FORMC Formatting Operation

You must use the CI RP command to RP FORMC before you format a disk or tape, as follows:

```
RP, FORMC
```

Otherwise, the error message

```
INSUFFICIENT CAPABILITY
```

or

```
MUST USE THE PROGRAM NAMED "FORMC" WITH THE FO AND SP COMMANDS
```

displays, and FORMC aborts.

## Entering the LU

When you enter the FO command, FORMC prompts you as follows:

```
/FORMC: CS80 DISK OR TAPE LU?
```

If you are formatting a tape (also called “tape certification” or “tape media initialization”), enter the LU of the CTD drive on which the tape is mounted. You may enter any LU on a disk volume, since the entire volume is formatted by FORMC. If the LU you enter does not correspond to a legal CS/80 disk or tape drive, FORMC displays the following message and prompts you again for the LU number:

```
/FORMC: ILLEGAL CS80 DISK OR TAPE LU
```

FORMC issues a warning and prompts you before a currently mounted disk or the system disks can be formatted.

## Disk Formatting

The FO command applies to the entire disk volume; for most CS/80 disks, this means the entire disk unit. Since CS/80 disks do not let you format RTE subchannels as with previous disk drives, you must specifically spare any defective disk blocks found with the SP command described earlier.

After you enter the correct disk LU, you must indicate whether you are using a single-sided or double-sided floppy disk:

```
/FORMC: SINGLE-SIDED FLOPPY (Y/N)?
```

Enter “Y” if your disk is single-sided or “N” if your disk is double-sided.

The program then asks if you want FORMC to assign a default record interleave factor:

```
/FORMC: INTERLEAVE FACTOR FOR OPTIMAL THROUGHPUT (Y,N)?
```

## Accepting the Default Record Interleave Factor

A “Y” response indicates that you want to use the default interleave factor of 1. This is sufficient for most applications. (Refer also to the earlier section on interleave factor calculation to determine the proper interleave factor for your use.)

---

### Note

If you are using FORMC on double-sided microflops, DO NOT use the default (enter “N” at the prompt). Drives which are both fixed and microfloppy drives also require an interleave factor of 2. When the next prompt is displayed (see below), enter 2 for the interleave factor.

(If you use the default interleave factor of 1, and the timeout is set to 500, timeouts will occur during reads and writes to that LU.)

---

## Specifying a Record Interleave Factor

A response of “N” to the prompt above lets you specify a different record interleave factor. When you enter “N”, you are prompted:

```
/FORMC: INTERLEAVE FACTOR?
```

Your response must be a decimal number in the range of 1 to 32. Otherwise, you are prompted again as follows, and FORMC waits for the correct entry:

```
/FORMC: ILLEGAL INTERLEAVE FACTOR, LEGAL RANGE 1 TO 32  
/FORMC: INTERLEAVE FACTOR?
```

## Dismounting the LUs

FORMC locates all the disk LUs that are on the same volume and thus, are subject to formatting. If any of the LUs are still mounted, FORMC displays the following messages, then exits:

```
/FORMC: DISMOUNT LU  
<lu 1>,<lu 2>.....,<lu n>  
  
/FORMC: DISMOUNT LU'S AND RESTART  
/FORMC: FINISHED
```

Use the FMGR DC command to dismount the LUs listed in the message and restart FORMC.

## The Disk Formatting Process

When the LUs are dismounted, FORMC locks all the LUs to be formatted and displays:

```
/FORMC: DATA WILL BE DESTROYED ON LU  
<lu 1>,<lu 2>.....,<lu n>  
  
/FORMC: OK TO PROCEED (Y,N)?
```

Respond “Y” to confirm that you want FORMC to format the entire disk volume. FORMC then issues the following message and exits:

```
/FORMC: DISK FORMATTING COMPLETED
```

If you enter N to the confirmation prompt above, FORMC unlocks all LUs and exits without formatting any disks.

## Formatting a System Disk

If you are formatting a system disk, FORMC warns you with the following message and waits for your response to the prompt:

```
/FORMC WARNING: SYSTEM WILL BE DESTROYED.  
NOW IS THE TIME TO REPLACE THE SYSTEM CARTRIDGE.  
  
/FORMC: OK TO PROCEED (Y,N)?
```

If you want formatting to proceed, replace the removable system disk with the disk to be formatted and enter “Y”. Be aware that FORMC formats the entire disk volume, including the system subchannels, if you enter “Y” and do not remove the system disk.

If you enter “N”, FORMC terminates.

If you are using FORMC on 3-1/2 inch, double-sided, flexible disks, you may see the following warning message:

```
/FORMC WARNING:  THE DOUBLE-SIDED 3-1/2 INCH FLOPPY THAT YOU JUST
FORMATTED IS NOT USABLE IN THE INTEGRATED MICRO/1000 FLOPPY DRIVE.
FORMATTING THIS FLOPPY IN A MICRO/1000 FLOPPY DRIVE, USING FORMF,
PROVIDES COMPATIBILITY FOR BOTH EXTERNAL AND INTEGRATED FLOPPY
DRIVES.
```

Refer to the *Small Disk User's Guide*, part number 5958-9152, for more details.

When formatting is complete, FORMC issues the message:

```
/FORMC: REPLACE SYSTEM CARTRIDGE TO RETURN TO SYSTEM.
/FORMC: OK TO RETURN (Y,N)?
```

Enter "Y" or "N". After the system disk is replaced, a "Y" response causes a return to the host system; otherwise, FORMC displays the following message and executes an infinite loop:

```
/FORMC: FINISHED
```

When this message displays, you must restore and reboot the system.

## Tape Formatting

---

**Warning** FORMC locks the entire disk when CS/80 cartridge tapes are being certified.

---

## Tape Certifying and Skip Sparing

Formatting a CTD tape is different than formatting a disk. If the tape was not previously certified (the "initialized media" flag on the tape is not set), FORMC performs a tape certification process. This includes writing a known pattern on every block of the tape, then reading it to locate any defective areas. Each 512th block on the tape is allocated as a sparing block, and a table of these blocks is created on the tape. If any defective blocks are discovered, they are flagged and subsequent writes to the tape do not use them. Finally, the initialized media flag is set on the tape.

Skip sparing can take up to twenty minutes for a short tape (DC 150) or more than one hour for a long one (DC 600). RTE-A utilities that access CTD tapes cannot use a tape until the initialized media flag is set.

## Re-certification and Spare Conversion

If the tape was certified and the flag is set, you may either re-certify 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.) Re-certification is identical to initial formatting, and any data on the tape is destroyed in the process.

The spare conversion process flags the tape blocks that were spared in jump sparing, so they are skipped on the next write to the tape. On the next write, the controller simply renumbers the tape blocks starting at the first good block following a defective block. The spare conversion option results in loss of data for any blocks that were jump spared.

You need not re-certify a tape on an HP 1000 system unless you have serious doubts about the tape's quality. Defective areas on a tape discovered after certification by a normal read or verification are automatically flagged as defective, and they are skip spared on any subsequent write. Since all HP 1000 utilities use skip sparing, the spare conversion option for HP 1000 CTD tapes is not necessary, though you may use it to reformat other, non-HP 1000 systems that use jump sparing.

### **Inserting the Cartridge**

When you call FO, FORMC determines whether the cartridge is inserted in the drive; this process requires 10 to 15 seconds to execute. If the cartridge is not present, the following messages are displayed, and FORMC exits:

```
/FORMC: READY TAPE AND RESTART
/FORMC: FINISHED
```

Insert the cartridge into the CTD tape drive and restart the utility.

### **The Tape Formatting Process**

If the cartridge shows an UNINITIALIZED MEDIA status, 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 enter "N", FORMC exits. Otherwise, FORMC proceeds to certify the entire tape, then issues the following message and exits:

```
/FORMC: TAPE FORMATTING COMPLETED
```

If the tape was already initialized, FORMC warns you as follows:

```
/FORMC WARNING: TAPE IS ALREADY INITIALIZED(CERTIFIED)
/FORMC: DO YOU WANT TO RECERTIFY OR CONVERT SPARES(CE,SP)?
```

When you enter "CE", FORMC notifies you:

```
/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 enter "N", FORMC exits. If you enter "Y", FORMC re-certifies the tape, issues the following message, and exits:

```
/FORMC: TAPE FORMATTING COMPLETED
```

### Converting Spares

If you enter "SP" when FORMC asks if you want to re-certify the tape, FORMC prompts you for confirmation before it proceeds:

```
/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 enter "N", FORMC exits. Otherwise, FORMC updates the tape spare table by converting jump spares to skip spares, then issues the following message and exits:

```
/FORMC: TAPE FORMATTING COMPLETED
```

In non-interactive mode, FORMC always performs certification on new tapes or re-certification on previously initialized tapes.

### The FORMC Sparing Operation

You must use the FMGR RP command to RP FORMC before you execute the Spare command, as follows:

```
FMGR: RP, FORMC
```

Otherwise, the error message

```
INSUFFICIENT CAPABILITY
```

or

```
MUST USE THE PROGRAM NAMED "FORMC" WITH THE FO AND SP COMMANDS
```

displays, and FORMC aborts.

## Entering the LU

When you call SP, FORMC prompts you:

```
/FORMC: CS80 DISK OR TAPE LU?
```

Enter the LU of the disk that contains defective blocks to be spared. If you enter a CTD tape LU, the disk cache memory area of the CTD is spared. If you enter an illegal LU, FORMC notifies you and prompts again:

```
/FORMC: ILLEGAL CS80 DISK OR TAPE LU
```

If the LU is a tape LU, and the tape does not have a generated disk cache memory, the following message is displayed and FORMC exits:

```
/FORMC: NO CACHE DEFINED FOR LU <number>  
/FORMC: ABORTED
```

When you enter a legal LU number, FORMC prompts you as follows:

```
/FORMC: TRACK TO BE SPARED?
```

## Specifying the Track

At the prompt, enter the number of the track that contains the bad block or blocks. If you enter a number outside the legal range, FORMC displays the <last track> (the number of tracks on the LU minus one) and prompts you again:

```
/FORMC: ILLEGAL TRACK NUMBER, LEGAL RANGE 0 TO <last track>
```

When you enter a legal number, FORMC checks the track for bad blocks. Note that you must use the SP command for each track to be spared.

## Sparing Defective Disk Blocks

Sparing removes individual defective disk blocks from active service. Defective blocks on CTD tapes do not require sparing because they are flagged when detected during a normal read operation by the format operation. Those blocks are then automatically spared on any subsequent write operation, a process known as “skip sparing.” However, you may also 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. Spare tracks and blocks are known only to the controller (their existence is transparent to users).

When the program spares a defective block within a physical track, it rearranges the track to bypass the defective block and to use the spare block. If it subsequently spares another defective block on the same physical track, it bypasses the entire track and writes the data to another physical track from the spare track pool.



Once the LU number and track number are known, 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 following message is displayed:

```
/FORMC: PHYSICAL BLOCK <address> SPARED FOR <number> BLOCKS
```

The <address> is the physical block address and <number> is the physical number of blocks spared. Since the specified track may cover more than one bad area, the program may issue more than one message. If it must use the next to last spare track, the program issues the following warning:

```
/FORMC WARNING: ONE SPARE PHY. TRACK LEFT. CONTACT SYSTEM MANAGER.
```

Although the program may perform several more sparing operations if the disk only needs one spare block per track, the disk has only one more physical track available for sparing. This 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.

## Data Loss During Sparing

If the data in a bad block may be lost when sparing occur, FORMC issues this warning:

```
/FORMC: DATA MAY BE DESTROYED ON LU <number> TRACK  
<track 1>,<track 2>.....,<track n>
```

```
/FORMC: OK TO PROCEED (Y,N)?
```

If you enter “Y”, sparing proceeds and the data on the tracks indicated by <track 1> through <track n> is not retained. If you enter “N”, the program terminates.

If the disk LU boundary 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 was backed up, you may enter “Y” and have FORMC lock all LUs that might cross physical tracks and perform the sparing operation. If you enter “N”, FORMC terminates.

After a successful sparing operation, FORMC informs you:

```
/FORMC: DISK SPARING COMPLETED
```

If FORMC finds no bad blocks on the specified disk, it issues the following message, then exits:

```
/FORMC: NO BAD BLOCKS - SPARING NOT PERFORMED
```

## The FORMC Verifying Operation

You can use FORMC to verify an entire disk or tape or specific tracks or blocks. A CTD tape block consists of 512 words, and a CS/80 disk block consists of 128 words. Note that all track references are to RTE tracks, not physical disk tracks.

### Entering the LU

When you enter the VE command, FORMC prompts:

```
/FORMC: CS80 DISK OR TAPE LU?
```

Enter the LU of the disk or tape to verify. If you enter an illegal number, the program prompts again:

```
/FORMC: ILLEGAL CS80 DISK OR TAPE LU
```

### Readying the Tape

Before verifying a CTD tape, FORMC makes sure the cartridge is in the drive. This takes 10 to 15 seconds. If the cartridge is not present, you are informed:

```
/FORMC: READY TAPE AND RESTART  
/FORMC: FINISHED
```

FORMC exits after it verifies the CTD disk cache memory area associated with the tape (if the cache exists). You must insert the cartridge in the drive and restart the utility. If the cartridge was not initialized (formatted), FORMC notifies you and returns to the TASK? prompt:

```
/FORMC: UNINITIALIZED TAPE MEDIA - FORMAT TAPE  
/FORMC: TASK?
```

You must use the FO command to format the tape. Refer to the discussion of the FO command and the section on “The Formatting Operation,” both in this chapter, for more information.

### Verifying the Entire Disk LU or Tape

When you insert a formatted cartridge and enter the correct LU, the program prompts you as follows:

```
/FORMC: VERIFY ENTIRE DISK LU (Y,N)?
```

or

```
/FORMC: VERIFY ENTIRE TAPE (Y,N)?
```

If you enter “Y”, FORMC verifies the entire disk LU or tape; otherwise, FORMC prompts with one of the following messages:

/FORMC: START TRACK NUMBER?

or

/FORMC: START TAPE BLOCK NUMBER?

### Specifying the Start Track or Block

If you are not verifying the entire disk LU or tape, enter an integer between zero and the number of tracks or blocks on the medium minus one. For tape verification, the number is between 0 and 65535.

If you enter a negative number, the program uses its complement as the starting block (for example,  $-2 = 65534$ ). If you enter an incorrect number (a number larger than the disk/tape capacity, or in the case of disk verification, a negative number), one of the following messages displays:

/FORMC: ILLEGAL TRACK NUMBER, LEGAL RANGE 0 TO <last track>

or

/FORMC: ILLEGAL BLOCK NUMBER, LEGAL RANGE 0 TO <last block>

The <last track> or <last block> is the number of the last track/block minus one.

### Specifying the Number of Tracks or Blocks

When you enter the correct number, FORMC displays one of the following prompts:

/FORMC: NUMBER OF DISK TRACKS?

or

/FORMC: NUMBER OF TAPE BLOCKS?

You must specify one or more tracks or blocks. If the number you enter is greater than the capacity of the medium, all tracks or blocks following the start number are verified. If you specify a negative number for the number of tape blocks, it is complemented and interpreted as positive (for example,  $-5 = 65531$ ). If you specify zero tracks or blocks, or a negative number of disk tracks, FORMC issues one of the following messages and repeats the prompt:

/FORMC: ILLEGAL NUMBER OF TRACKS

or

/FORMC: ILLEGAL NUMBER OF BLOCKS

## The Verification Process

When you confirm that you want to verify the entire disk LU or tape, or when you enter the correct start track or block and specify the number of tracks and blocks to verify, the program performs the verification.

If you are verifying a CS/80 tape, and it has an associated disk cache memory, the disk cache is also verified.

If FORMC encounters a defective track or block during verification, it notifies you with one of the following messages:

```
/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
```

The <number> is a decimal integer identifying the defective track or block.

When it completes the verification process, FORMC displays:

```
/FORMC: DISKVERIFY COMPLETED <number> BAD TRACK(S)
```

or

```
/FORMC: TAPE VERIFY COMPLETED <number> BAD BLOCK(S)
```

or

```
/FORMC: TAPE CACHE VERIFY COMPLETED <number> BAD TRACK(S)
```

The <number> is either a decimal integer that defines the number of defective tracks or blocks, or "NO" to indicate that no defective tracks or blocks were encountered.

You can use the SP command, as described in earlier sections of this chapter, to spare any bad tracks after verification.

## Loading FORMC

To load FORMC, use the LINK command file #FORMC.

## FORMC 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 messages may be displayed in the following format:

```
/FORMC: eeeee ERROR xxxxxxB
      <mnemonic error code>
IDENTIFICATION FIELD: yyyyyyB QSTAT: q
PARAMETER FIELD P(1) THRU P(10):
ppppppB ppppppB ppppppB ppppppB ppppppB
```

where

- eeeeee The error type: REJECT, FAULT, ACCESS, INFORMATION.
- xxxxxB The error field, in octal.
- yyyyyB The identification field, in octal.
- q The error-reporting message from the device.
- ppppppB The five-word parameter area, in octal, for the error.

If the program encounters an undefined error bit in one of the status words, the following message replaces the <mnemonic error code> in the message:

```
FATAL INTERNAL ERROR
```

Refer to the *RTE-A Driver Reference Manual* for a description of the error reporting scheme and a complete summary of all mnemonic error codes.

## Disk Formatting Utility (FORMF)

FORMF is an online and bootable offline maintenance program for floppy disk drives that you can use to verify the integrity of data on a disk unit and to format disks for future use.

FORMF performs the following functions to format a floppy or mini-Winchester disk drive:

- Writes (or rewrites) the track and sector addresses. The disks are divided into physical sectors of 128 words each.
- Interleaves the disk sectors according to the interleave factor (sometimes called fill number), if requested. Interleaving can help reduce access time.
- Identifies bad tracks and makes them invisible. The faulty tracks present no problem unless there are so many that the disk is not usable.

FORMF supports the following disk drives:

HP 9121 S/D	3-1/2 inch microfloppy
HP 9133 A/B/XV	5 and 10 Mbyte Winchester with microfloppy
HP 9134 A/B/XV	5 and 10 Mbyte Winchester
HP 9895	8 inch floppy drive Model 6 built-in 5 inch floppy drive
HP 248x	Integrated 3-1/2 inch single-sided microfloppy
HP 248x	Integrated 3-1/2 inch double-sided microfloppy

## Calling FORMF

To call FORMF, enter the following runstring:

```
CI> [RU,]FORMF[,list LU][,command,disk LU[,interleave]]
```

where:

list LU	The LU of the list device to which FORMF directs messages.
command	The FORMF commands are described below.
disk LU	The LU on the disk unit to be verified or formatted. (It is important to understand that with some disks, more than one LU can be on one unit. Whichever unit is pointed to by that LU is formatted. The whole unit is formatted, not just the LU.)
interleave	The optional interleave factor (a decimal number between 1 and 32) specified with the FO command. The default is an assigned interleave factor calculated for optimal throughput for the drive type being formatted.

If you omit a required parameter or enter one incorrectly, FORMF exits with the messages:

```
/FORMF: ILLEGAL RUNSTRING PARAMETER  
/FORMF: ABORTED
```

If you do not enter any runstring parameters, FORMF runs interactively, prompting you to enter a command as follows:

```
/FORMF: FLOPPY/MINI-WINCHESTER DISK FORMAT UTILITY  
/FORMF: '?' WILL LIST THE LEGAL COMMANDS  
/FORMF: TASK?
```

When you enter a command, FORMF prompts for parameters, which are described below.

## Command Execution

Before each command is executed, FORMF issues this message:

```
/FORMF: command executing - <command> <parameters>
```

FORMF terminates after successful command execution or after issuing an appropriate error message when it encounters an error.

## Device Driver Status

FORMF tests the disk driver status as it executes commands, and if the driver has timed out, it issues the following messages and exits:

```
/FORMF: DEVICE DRIVER TIMED OUT.  
/FORMF: READY DRIVE ENTER " " RETURN
```

Driver timeout occurs if the disk drive becomes disconnected or if the hardware fails. (Refer to the "FORMF Error Messages" section later in this chapter for a description of the device driver status information message.)

## Break Detection

FORMF checks the break flag during verification (see the discussion of the VE command below) when defective tracks are found. If the flag is set, FORMF exits with these messages:

```
/FORMF: BREAK DETECTED  
/FORMF: ABORTED
```

The program does not check the break flag during the format operation, because formatting must complete without interruption in order to preserve the integrity of the disk. See the discussion of the FO command in this chapter.

## FORMF Commands

FORMF commands are summarized in Table 5-3 and described in the sections that follow.

Table 5-3. FORMF Commands Summary

Commands	Description
<b>Information Command</b>	
HElp ?	Displays a list of FORMF commands
<b>Configuration Commands</b>	
FOrmat VERify disk lu disk lu	Formats a disk Verifies the data on the disk
<b>Exit Commands</b>	
ABort ENd EXit	These three commands terminates program execution

### Abort, End, and Exit (AB) (EN) (EX) (/E)

Purpose: Use any of these commands to terminate FORMF.

Syntax: AB, EN, EX, or /E

Description:

You may enter the exit or abort commands in response to any FORMF prompt and cause the utility to terminate execution.

---

**Warning** Do not abort FORMF during a format operation, as control information on the disk could be damaged or lost, leaving the disk unusable.

---



## Format (FO)

Purpose: To format the specified disk.

Syntax: `FO, disk LU [,interleave]`

disk LU            The LU of the disk to be formatted.

interleave        The interleave factor, a decimal number between 1 and 32. If you do not specify an interleave, FORMF assigns an interleave factor that is calculated for optimal throughput for the drive type to be formatted. (See the earlier section on “Interleave Factor Calculation.”)

Description:

Detailed information about FORMF formatting is provided in subsequent sections in this chapter. See also the sections in the beginning of this chapter on “General Formatting Information” and “Interleave Factor Calculation.”

## Help (?)

Purpose: Displays a list of FORMF commands.

Syntax: ?

## Verify (VE)

Purpose: Verify the integrity of the specified disk.

Syntax: `VE, disk LU`

disk LU            The LU of the disk to be verified.

Description:

Detailed information about verification is provided in subsequent sections on “The Verify Operation.”

## The FORMF Formatting Operation

Before the actual formatting begins, FORMF writes test data and error checking information into the user data and postamble areas as described in the earlier section, “General Formatting Information.” The program tests each block by reading the test data back and calculating the error checking information. If the test passes, the block is normal. If the test fails, the preamble is rewritten to make the block invisible. Each block is tested several times.

FORMF then sends a format code sequence to the disk controller, which performs the actual formatting. In the formatting process, the entire disk unit is formatted, not just the LU specified. Note that all information previously stored on the disk is lost.

If two or more LUs share one unit, part or all of the other LUs are cleared when the unit is formatted. This happens because the controller does not recognize LUs, only the entire unit. This is true for all drives except the multi-volume HP 9133/9134 5Mbyte mini-Winchester disks. Only the formatted unit is cleared; the other units are not affected.

### Entering the LU

You may enter any LU on a disk, but remember that the entire unit is formatted, not just the LU specified. If the LU does not correspond to a valid disk drive, FORMF issues an error message and prompts for the disk LU again:

```
/FORMF: ILLEGAL disk LU
```

When you enter a legal disk LU, if FORMF is a system utility and you have superuser capability, the program proceeds; otherwise, the following message displays:

```
/FORMF: INSUFFICIENT CAPABILITY
```

### Specifying an Interleave Factor

If the disk LU you specified is a floppy drive, FORMF asks if you want to assign a default interleave factor:

```
/FORMF: INTERLEAVE FACTOR FOR OPTIMAL THROUGHPUT (Y,N)?
```

You may define the order in which logical blocks are located on the disk by specifying an interleave factor, the number of physical blocks to be skipped between logical blocks. With an interleave factor of zero, block  $n$  is followed immediately by block  $n+1$ . With an interleave factor of two, two physical blocks are skipped between block  $n$  and block  $n+1$ . Once the disk is formatted, the interleave is transparent to your software.

Refer to the earlier section on “Interleave Factor Calculation” for a more detailed discussion on calculating the interleave factor.

If you respond by entering “Y”, FORMF supplies an interleave factor calculated for optimal throughput for the disk to be formatted. The default interleave factors are listed during execution.

If you enter “N”, FORMF prompts you to enter an integer between 1 and 32 as follows:

```
/FORMF: INTERLEAVE FACTOR?
```

## The Formatting Process

FORMF identifies all the disk LUs that are on the same device as the specified LU and locks them to prevent any interference with the formatting process.

Then FORMF issues the following warning to give you a chance to stop the process before formatting destroys any data that you really want to retain (for example, on the other LUs that share the disk unit):

```
/FORMF: DATA WILL BE DESTROYED ON LU  
<lu 1>,<lu 2>.....,<lu n>  
/FORMF: DO YOU REALLY WANT TO FORMAT THIS DISK <Y,N>?
```

If you enter “N”, FORMF unlocks the LUs and exits without formatting the disk unit. If you enter “Y”, FORMF sends the command to the controller, and the entire disk unit is formatted. When the formatting procedure is complete, FORMF issues the following message and exits:

```
/FORMF: DISK FORMATTING COMPLETED
```

To format the system disk, a memory based FORMF must be used, since FORMT does not allow formatting of mounted LUs.

## The FORMF Verifying Operation

### Entering the LU

When you enter the VE command, FORMF prompts for the required parameter:

```
/FORMF: DISK LU?
```

Enter the LU of the disk to be verified. If you enter an illegal number, FORMF prompts for the disk LU again:

```
/FORMF: ILLEGAL DISK LU  
/FORMF: DISK LU?
```

### Verifying the Disk

After you enter a legal LU, FORMF checks the disk. If a bad track is found, FORMF reports:

```
/FORMF: LU<disk LU> - BAD TRACK <cyl number> <head #>
```

The <number> is the number of the defective track. If no bad tracks are found, the disk is verified, and you are notified upon completion:

```
/FORMF: LU<disk LU> - NO BAD TRACKS FOUND
```

## **FORMF Error Messages**

### **/FORMF: CONTROLLER FAULT**

A controller problem has forced FORMF to halt.

### **/FORMF: DMA PARITY ERROR**

A DMA parity error occurred during a transmission to/from the disk.

### **/FORMF: DRIVE FAULT**

A drive fault occurred on the selected drive.

### **/FORMF: DRIVE NOT CONFIGURED**

The drive selected is not configured on the controller board.

### **/FORMF: FATAL INTERNAL ERROR**

An unidentified error has occurred within FORMF. Try running FORMF again.

### **/FORMF: ILLEGAL ENTRY ENTER “?” FOR HELP!**

FORMF is requesting a command; enter ? if you want to display a list of all the FORMF commands.

### **/FORMF: INSUFFICIENT CAPABILITY**

FORMF must be loaded as a system utility and the user must have superuser capability to format a disk.

### **/FORMF: MAX OF 5 BAD TRACKS EXCEEDED**

The flexible disk being formatted has too many bad tracks. Discard the disk.

### **/FORMF: NO DRIVE ATTACHED**

The controller has sensed that no drive is attached to the selected unit.

### **/FORMF: NO MEDIA IN DRIVE**

FORMT has sensed an empty drive.

### **/FORMF: READY DISK AND ENTER “GO” WHEN READY**

The disk drive is not ready, or no disk is in the drive. Insert a disk or make it ready and type a space followed by a carriage return.

### **/FORMF: REMOVE WRITE PROTECT FROM MEDIA**

Remove the write protection to format a flexible disk.

### **/FORMF: SEEK CHECK DURING OPERATION**

The drive has an unrecoverable seek check.

**/FORMF: UNABLE TO FIND BUS LU**

FORMF requires an HP-IB bus LU to format or verify HP-IB disks.

**/FORMF: UNABLE TO FORMAT DISK**

FORMF could not recover from an error during a format.

**/FORMF: UNABLE TO INITIALIZE DISK**

FORMF could not recover from a disk error while initializing a disk track. A preceding error message may give some indication about the fault, or a media problem exists.

**/FORMF: UNABLE TO VERIFY AFTER TWELVE PASSES. REPLACE MEDIA OR REPAIR DRIVE.**

FORMF was unable to free the media of errors and stabilize it after twelve verify and initialize passes. A media/drive problem exists.

**/FORMF: UNABLE TO VERIFY DISK**

FORMF could not recover from an error while verifying the disk.

**/FORMF: UNFORMATTED MEDIA**

You tried to verify unformatted media. Format the media before trying again.

**/FORMF: UNSUCCESSFUL LU LOCK**

FORMF could not lock all the LUs it needed to maintain bus integrity.

**/FORMF: WARNING! POSSIBLE BAD FLOPPY MEDIUM – RETRY FORMF OR DISCARD FLOPPY**

An error occurred on the fifth and final format/verify combination. Try to format the disk again. FORMF may identify and eliminate the faulty track. If it fails again, discard the disk.



## Offline Disk Formatting and Initialization (FORMT)

FORMT is supplied in relocatable form only. In order to create an offline version of FORMT and to make sure that no other program accesses a disk LU that is being formatted, you must generate a special system in which there is only one ID segment (the one for FORMT). This effectively makes FORMT an offline utility. When FORMT executes, it actually verifies whether or not there is more than one ID segment. If there is, FORMT aborts with an error message.

In order to create a FORMT system under RTE-A, you must relocate %FORMT and put it into the system via either the BOOTEX or BUILD program. Note that whenever you relocate %FORMT, you must search the library \$DKLIB and size the program to at least 19 pages.

The configuration instructions are provided in the subsequent section on “Generating a FORMT System”.

FORMT provides the following functions:

- Formats a flexible (or floppy) disk. All new flexible disks must be formatted by writing track and sector addresses to them.
- Identifies bad tracks on the disk and marks them. Bad tracks are not spared, but are made invisible to your other programs.
- Initializes an HP 7906/7920/7925 disk LU. This removes the protected status of all the tracks in the LU, cleans up the spare track pool, and prepares the LU for RTE use by sparing any bad tracks. (This is different from the IN command in FMGR.)

Note that this implies you are using an HP disk drive and a disk cartridge that was formatted at the factory. A disk cartridge supplied from another source or direct from assembly will not work.

Formatting is done only on test setups where correct alignment and calibration is maintained.

- Spares individual bad tracks. This process recovers data and assigns a spare track to a device track so that the disk controller can access the track correctly in further operations.
- Verifies a hard or flexible disk. The read and verify process checks the tracks of a flexible or hard disk without altering the data on them.
- Reformats a hard disk. This allows you to clear all accumulated bad tracks by clearing the bad track and spare track information on each track. The disk must have been formatted at least once at the factory before reformatting can be done.

### Calling FORMT

FORMT runs in interactive mode, prompting you to specify the task to be performed as follows:

```
FORMT: TASK?
```

When you enter a command, the program prompts you to enter the LU number of the disk to be formatted, initialized, spared, or verified.

# FORMT Commands

Table 5-4 summarizes the available FORMT commands, which are further described in subsequent sections.

**Table 5-4. FORMT Commands Summary**

Commands		Description
<b>Information Command</b>		
HElp	??	Displays a list of FORMT commands
<b>Configuration Commands</b>		
FOrmat	disk	Formats a flexible disk
INitialize	lu	Initializes a hard disk only*
REformat	disk	Reformats a hard disk
SPare	track#	Spares defective tracks on a hard disk only*
VErify	lu	Verifies the data on a disk LU
<b>Exit Command</b>		
ENd		Terminates FORMT when entered in response to TASK?; otherwise, returns system to TASK? prompt.
* The IN and SP commands apply to hard disk operations only		

## End (EN)

**Purpose:** Terminates FORMT when entered in response to the TASK? prompt. Otherwise, entering EN returns the system to the TASK? prompt.

## Format a Mini-Floppy Disk (FO)

**Purpose:** Formats a floppy disk.

**Syntax:** FO [disk lu]

**Description:**

Formatting a floppy disk means writing or rewriting track and sector addresses; these are physical sectors of 128 words each. Sector interleaving can be provided, if desired, to help minimize access time.

Refer to the sections at the beginning of this chapter on “General Formatting Information” and “Interleave Factor Calculation” for more detailed information. See also the discussion of “The FORMT Formatting Operation” in subsequent sections.

## Help (??)

Purpose: Prints a list of all valid responses to the current prompt.

Syntax: ??

## Initialize an LU (IN)

Purpose: Initializes an entire hard disk LU.

Syntax: IN [disk lu]

Description:

The initialization process prepares an entire hard disk LU for use. The previous contents of the disk are overwritten with a pattern of all zeros. Bad tracks are spared to the spare pool at the end of the disk LU.

Refer to the general information sections in the beginning of this chapter to learn more about the differences between initializing and formatting.

See also the subsequent sections under “The FORMT Initializing Operation” for a more detailed discussion of using the IN command.

## Reformat a Hard Disk (RE)

Purpose: Reformats a disk, destroying all the previously written data on the disk.

Syntax: RE [disk lu]

Description:

The RE command clears all the status bits on the designated disk, including the spare track pool. It rewrites the preamble, writes zeros in the data area, and rewrites the postamble. It does not verify this data, nor does it spare bad tracks. This lets you clear all accumulated bad track bits, including factory designated bad tracks, and clears the spare and protect bits, as well.

Disk reformatting uses the track map table to determine the starting cylinder, head number, number of tracks, and number of spare tracks for each LU. All tracks in the LU, including the spare track pool, are reformatted. The program reports any disk errors that occur during reformatting.

For more detailed information about using the RE command, see the subsequent sections on “The FORMT Reformatting Operation.”



## 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:** SP [track#]

**Description:**

Sparing applies to individual bad tracks discovered while the disk is in use. A spare track is substituted for the defective track and as much as possible is copied from the bad track. Offset reads are used in the recovery process, and it is often possible to completely recover the data. In cases where this is not possible, usually only a single block is lost.

General information about sparing bad tracks is presented in sections at the beginning of this chapter. More information about using the SP command is provided in “The FORMT Sparing Operation,” later in this chapter.

## Verify an LU (VE)

**Purpose:** Verifies the specified LU and reports defective tracks.

**Syntax:** VE [disk lu]

**Description:**

Disk verification uses the track map table to determine the starting cylinder, head number, number of tracks, and number of spare tracks for each LU. Tracks in the spare pool are used only as replacements for the defective tracks. They are not verified otherwise.

See also the subsequent sections on “The FORMT Verify Operation.”

## Generating a FORMT System

The FORMT utility was designed as a stand-alone utility in order to reduce the possibility of corrupting your disks. This means that you must generate FORMT into a special system whose sole purpose is formatting.

Follow the steps listed below to generate this system under RTE-A:

1. Make a copy of the current system’s generation answer file. Name it FSYS.ANS or something easy to remember or recognize.
2. Reduce the number of ID segments in the system to 1 (one). There are probably 20-30 ID segments at present. Change the entry from #ID,30“ to #ID,1”.
3. If a MAC disk (HP 7906, 7920, or 7925) is being formatted, follow steps a, b, and c below. Otherwise, skip to step 4.
  - a. Add an entry for %IDM37 in the system relocation phase in the generation answer file.

- b. Remove the entry for %ID\*37 from the driver partition phase. You cannot partition the new MAC disk driver.
  - c. Change all references to %ID\*37 in the answer file to %IDM37. Module %IDM37 is supplied with the Primary System (part number 92077-16700).
4. Run the generator, which creates the type 1 system file.
  5. Run the loader to load %FORMT, searching \$DKLIB. Note that you must size FORMT to at least 19 pages in order to provide sufficient buffer space. The loader produces a type 6 file named FORMT. You can use the link command file #FORMT.
  6. Use the BUILD utility to build the type 6 FORMT file and the type 1 system file together into your final type 1 format system. When you run BUILD, remember to specify FORMT as the startup program by using the ST command.
  7. Place the type 1 format system file that BUILD created on a bootable LU and boot it up.

If your RTE-A FORMT generation is successful, you should see the “TASK?” prompt.

## 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 examining the information defined in the RTE-A driver parameter area. If there is a discrepancy, the program notifies you:

```
INVALID DISK LU
```

You are then prompted again to enter the disk LU.

### Confirming Formatting

Since formatting destroys the contents of the specified disk LU, 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).

## 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, see the section at the beginning of this chapter on “Interleave Factor Calculation.”

---

**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 EDITR, the loader, and so on.

---

## The Formatting Process

FORMAT 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-XX
```

After formatting the disk, the program reports the number of good tracks:

```
# OF GOOD TRACKS = XXXX
```

The program then repeats the request for a task.

## 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 that the LU is a hard disk. Any attempt to initialize a flexible disk or a mini-Winchester causes the following message to display:

```
INVALID DISK LU
```

You are then prompted again to enter the disk LU.

### Confirming Initializing

Since initializing overwrites the previous contents of the disk with a pattern of all zeros, the program asks (except when in batch 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
```

LU XX	LOGICAL	CYL	HEAD	UNIT/ADD
BAD TRACK		XXXX	XXXX	X X/X
SPARED TO		XXXX	XXXX	X X/X

```
XX SPARE TRACKS AVAILABLE
```

The program then repeats the request for the task.

### 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.

## **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.

### **Before You Use the RE Command**

The directory track of the disk LU is also overwritten with zeros when you reformat a disk; therefore, you should take the following steps before using RE:

1. Back up all data from the LU to be reformatted.
2. Dismount the disk LU (use the FMGR DC command).

### **Entering the LU**

After you follow steps 1 and 2 above, enter the RE command and specify the LU. The program checks the disk LU and its type. If you try to reformat a floppy disk LU, the following message appears:

```
INVALID DISK LU
```

You are then prompted again to enter the disk LU.

## The Reformatting Process

Since reformatting destroys all data on the disk, FORMT warns you before continuing:

```
DATA WILL BE DESTROYED, OK TO PROCEED?
```

Enter "YES" or "NO". A YES response causes the reformatting to occur; a NO response causes FORMT to terminate.

## After You Reformat the Disk

After the successful reformatting of the disk, use the IN command to prepare the disk for File Manager use. IN spares any tracks that are defective, according to the system track map table. Then use the SP command to spare factory designated bad tracks.

Follow these steps:

1. Invoke the IN command, which reports and spares all bad tracks.
2. Mount the disk LU (use the FMGR MC command).
3. Restore the data, if desired.
4. Finally, use the SP command to spare factory designated bad tracks, even if the IN command does not report them as bad. Factory designated bad track records are delivered with the disk pack. If your records are misplaced, contact your Hewlett-Packard representative to obtain them from Hewlett-Packard.

## 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 you enter the SP command and specify the LU, the program responds:

```
TRACK TO BE SPARED?
```

Enter the track (logical track) found defective on the disk LU. If the track does not lie within the bounds of the disk LU specified, the following message appears:

```
INVALID TRACK #
```

FORMT then prompts you again for the track to be spared. You may enter ?? when prompted, to see the valid track range for the LU.

### 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, then continues:

```
WARNING! ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED
```

When the program completes execution, the sparing is reported as follows:

```
BAD TRACKS
```

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
```

If not in batch mode, the program repeats the TASK? prompt.

## The FORMT Verify Operation

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 you enter the VE command and specify the LU, the program checks the disk LU and its type. If you try to verify a non-disk LU and are running FORMT interactively, the following message appears:

```
INVALID DISK LU
```

You are then prompted again to enter the disk LU.

### The Verify Process

The program reads each track in the mapped portion of the LU, and if the read is unsuccessful, it reports the track as bad. The directory track of the disk LU is checked, but the spare track pool is not checked.

Bad tracks without a valid spare are reported as follows:

```
BAD TRACKS SUBCHANNEL XX
```

```
LU XX   LOGICAL CYL   HEAD   UNIT/ADDR  
BAD TRACK      XXXXX   XXXX   XX     X/X
```

You can use the SP command to spare any defective tracks reported here.

If you are in interactive mode, the program returns to the TASK? prompt when the verification completes; otherwise, FORMT terminates.



## **FORMAT Error Messages**

The following messages may be issued by FORMAT:

**CYLINDER COMPARE ERROR**  
**LU XX LOGICAL CYL HEAD UNIT/ADDR**  
**TARGET TRACK XXXX XXXX XX XX/XX**

The disk cannot seek to the target track. Make sure the disk is formatted. The current task aborts, and FORMAT returns to the TASK? prompt.

**MORE THAN 1 ID SEGMENT IN SYSTEM.**  
**FORMAT ABORTED**

FORMAT was invoked in a system that was generated with more than one ID segment. This is a fatal error; FORMAT aborts.

**ENTER FO(RMAT), IN(ITIALIZE), SP(ARE), VE(RIFY), RE(FORMAT), EN(D)**

Entry of the first two characters of any of these commands starts that section of the FORMAT program.

**INVALID DISK LU**  
**ENTER DISK LU<256**

The disk type is wrong for the operation (for example, you are trying to use the IN command on a flexible disk) or the LU has zero tracks, or you entered a non-numeric parameter.

**INVALID DISK SPECIFICATIONS XX**

The disk controller detects an out-of-bounds condition on a cylinder, head, sector, or unit based on an invalid mapping of the disk LU. The current task aborts, and FORMAT returns to the "TASK?" prompt. The disk logical unit number is XX.

**LU XX LOGICAL CYL HEAD UNIT/ADDR**  
**BAD SPARE XXXX XXXX XX XX/XX**

A defective spare is encountered in the IN or SP task. The numbers are the same as reported in the CYLINDER COMPARE ERROR, above.

**MAX OF 20 BAD TRACKS EXCEEDED**

This message displays only when you are formatting a flexible disk. The disk cartridge should be discarded.

**NOT ENOUGH ROOM FOR TRACK BUFFER**

The memory bounds specified for FORMAT at load time did not provide sufficient free memory to serve as a track buffer for the task and disk LU requested. FORMAT terminates.

**OUT OF SPARE TRACKS FOR THIS LU**

All spare tracks are used up for this LU. FORMAT returns to the TASK? prompt.

**READY DISK – ENTER “ ”,CR**

The disk drive is not ready, or no disk is in the drive. Ready the disk and type a space followed by a carriage return.

**TURN OFF PROTECT OR READ#ONLY SWITCH – ENTER # “”,CR**

The disk protect or read only switch is on. Or, the message may occur if the floppy in the selected drive has the write notch present. Turn off the switch and enter space, followed by return.

**TURN ON FORMAT SWITCH – ENTER “ ”,CR**

The utility is formatting the disk, but the format switch is off. Turn it on and type a space followed by a carriage return. This message also occurs if a write operation is attempted to a track with its P bit (protected track) set while the Format switch is off.

**WARNING! ALL INFORMATION ON TRACK NOT SUCCESSFULLY RECOVERED**

Spare only. The track about to be spared was not successfully preserved. FORMT continues.

**WARNING! POSSIBLE BAD FLOPPY MEDIUM – RETRY FORMT OR DISCARD FLOPPY**

Format only. An error occurred on the fifth and final format/verify combination. Restarting the FO process may mark the track defective unless the medium is truly bad.

## Initialize and Spare Utility (FORMA)

FORMA provides initialize and spare functions for Micro/1000 systems with internal disks connected via the HP 12022 disk controller. A series of prompts helps you select a function and further configure the program to your specific application.

FORMA is designed to work together with ERTSH, an error rate test program for mini-Winchester hard disks. ERTSH is discussed in detail in the section following FORMA. Note that FORMA and ERTSH require at least 256 KBytes to run.

A major cause of disk problems is minute flaws in the magnetic disk surface that cause some data bits to be unreadable or to be read incorrectly. Because these defects are on the physical disk surface, they may not be centered on a track and may cross logical boundaries. It is common to find a sector with a correctable error next to one that cannot be corrected. You can use FORMA to spare the correctable error before it grows into an error that cannot be fixed.

To spare bad tracks, the disk must be in FORMA format. If your disk is currently in FORMF format, you should re-initialize it, using FORMA, at your earliest convenience (for example, after your next backup). If you wait until major errors appear, the defects could be severe enough to prevent a successful backup.

Using ERTSH should become a regular part of your periodic maintenance procedure for the disk. The ERTSH Read Only (RO) command indicates the general state of the disk without overwriting data. Other combinations of commands discussed in this chapter let you test for and automatically spare defective tracks on disks that are in FORMA format.

## FORMA Definitions

The following definitions apply to formatting and sparing Micro/1000 internal mini-Winchester disks.

Index pulse	A physical marker on the spindle of the mechanism that generates a logical signal once every revolution. It is used to designate the beginning of a track. See Figure 5-1.
Primary spare	A spare that covers a defect that was originally designated by the manufacturer. All other spares are called secondary spares, including any spares found after shipment from HP. Both types of spares are created using the FORMA SS (Spare Sector) command.
Dormant spare	A reserved sector (sector 32) that is used for sparing if a defect appears on that track.
Track format	The logical layout of a section of the disk at one head and cylinder that provides consistent and efficient data transfer.
ID field	A non-data section that contains addressing information for that sector. (See Figure 5-1.) It is read by the disk controller before each data field is read in, thus allowing sectors to be in any order on a track. The process of writing these fields on every sector is part of the IM (Initialize Media) command.

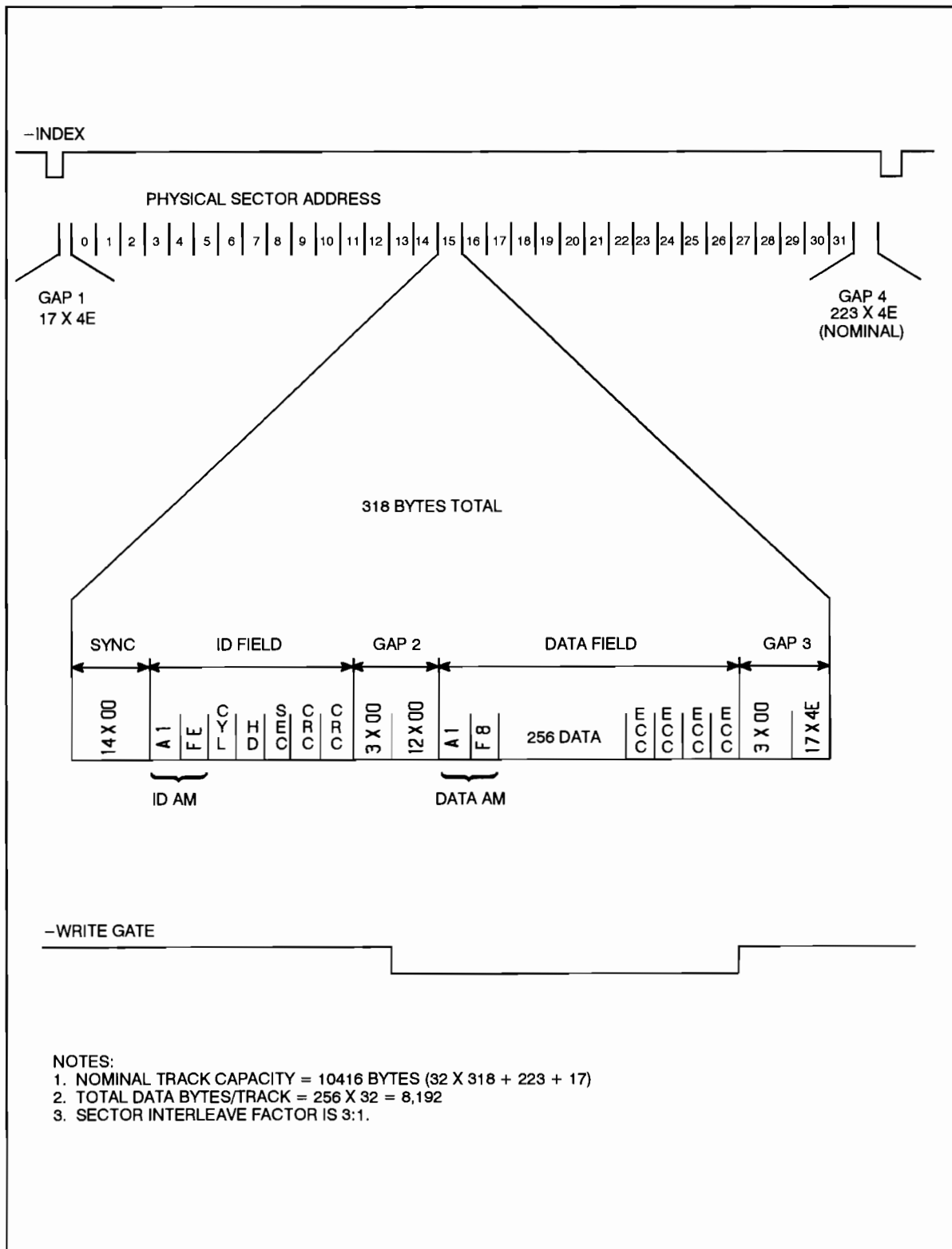


Figure 5-1 . Typical Track Format

## Calling FORMA

To call FORMA, enter the following runstring:

```
CI> RU, FORMA, <command>, <disk lu>, <option>
```

The command and option parameters are discussed in the following sections.

Alternatively, you can run FORMA interactively by entering the following:

```
CI> RU, FORMA
```

The program prompts you for the commands, LU, and parameters.

## FORMA Commands

Formatting and sparing functions are carried out by two FORMA commands, IM (Initialize Media) and SS (Spare Sector). The IM command is used during installation or for recovery from major damage to media. The SS command spares defective sectors by recovering data and assigning a spare track that the disk controller can access.

### Help (??)

Purpose: Enter this command at the TASK? prompt to see a list of available responses.

Syntax: ??

### Initialize Media (IM)

Purpose: Initializes a new disk when it is installed, re-initializes a disk previously initialized with FORMF, and recovers from major disk media damage.

Syntax: IM [<lu>] [<spare option>]

lu                    The disk logical unit.

spare option        The option to save primary (P) spares, secondary and primary (S) spares, or no (N) spares.

Description:

IM provides three options, P, S, and N, which you can choose to preserve primary, secondary and primary, or no spares. Always retain primary spares, unless the media is severely damaged.

---

**Caution**    FORMA initializes an entire disk upon recognition of any valid LU number. Before initializing a disk that is currently in use, be sure to back up the disk.

New disks shipped from HP are already initialized and spared in FORMA format; you need not initialize them again. In this case, proceed to the discussion of ERTSH later in this chapter.

FORMA and FORMF formats are not compatible. Using FORMA to re-initialize disks that were previously initialized with FORMF takes a long time (up to 16 hours). If the floppy disks were initialized with FORMF, do not retain any spares before you re-initialize with FORMA.

Do not use IM in place of a logical debugging process; you should only use it at installation or in case of total media loss.

---

## Spare Sector (SS)

**Purpose:**        Spares a sector to an unused portion of the disk reserved for that purpose. Retention of data from the sectors being spared can be specified.

**Syntax:**        `SS, <disk lu>, <r>, <s/b>, <ccc>, <h>, <ss/bbbbb> [p], [o/u]`

disk lu	The number of the LU, from 1 to 63.
r	The option to retain data (y or n).
s	Logical (sector) addressing.
b	Physical (byte) addressing.
ccc	The cylinder number of the defect.
h	The head number.
ss	The logical sector since index (see index description below).
bbbbbb	The physical byte count since index (see index description below).
p	Save the primary spare.
o	Override the old spare.
u	The "unspare" option.

### Description:

You must enter the first seven parameters or the program aborts. (The options are further described in the next section.)

Note that when you use SS, there is some chance of data loss on the spared sector, depending upon the number of defective bits in that sector.

You must enter only the byte/index address of the defective sector when performing sparing. Logical sectors refer to the address written on the ID field of the sector during formatting. Logical sectors with adjacent addresses are not physically adjacent, due to a 3:1 interleave. Table 5-5 provides information that you need to answer the system prompts.

**Table 5-5. Responses to System Prompts**

	<b>Cylinder</b>	<b>Head</b>	<b>Sector</b>	<b>Byte/Index</b>
10Mb	0 – 305	0 – 3	0 – 30	16 – 10048
15Mb	0 – 305	0 – 5	0 – 30	16 – 10048
20Mb	0 – 611	0 – 3	0 – 30	16 – 10048

Byte/index refers to the number of bytes that passed under the head since the last index pulse (which signifies the physical start of a track). This form of input is useful when you are sparing sectors that the manufacturer has indicated are defective. Use the label on the drive mechanism, as discussed in the section in this chapter on “Installing a New Disk.” The label uses the byte/index format. One spare sector per track is available.

Remember that some defects will fall into gaps between the data fields. FORMA detects these and leaves them unspared, returning this message:

FORMA: REQUESTED SPARE FALLS WITHIN A NON DATA AREA, NO SPARE REQUIRED

## **SS Command Options**

When you enter the P (Primary Spare) option as the seventh parameter, a primary spare is allocated on the track specified. All spares made from the information derived from the label on the drive should be primary spares. (See the section “Installing a New Disk” in this chapter.)

You can use the O (Override Spare) and U (Unspare) options to correct errors made when sparing. Override allows sparing when a sector is already spared on a track. Unspare re-initializes a track to the standard dormant spare state.



## Converting from FORMF to FORMA

Although you can run the error rate test (ERTSH) on disks that are in FORMF format, you must use FORMA to initialize a disk before you can have defective tracks spared. The FORMA initialization procedure is similar to the installation of a new disk, described in the following section, except that references to the manufacturer's label do not apply.

## Installing a New Disk

Follow these steps to install a new internal hard disk:

1. Find the label attached to the top of the disk drive that lists the areas the manufacturer has found to be defective. Record the CYL, HD, and BYTE/INDEX information. This data is used later in the FORMA runstring.
2. If you already installed the disk, your customer engineer (CE) can read the label by removing the three securing screws from the bottom of the peripheral tray and sliding the disk drive out far enough to read the label. If you are working without the aid of an HP CE, do not attempt to remove the disk drive. Use ERTSH to locate the defects. ERTSH is discussed in detail later in this chapter.

## **FORMA Error Handling**

All user errors cause FORMA to abort before execution. The program reports disk errors during execution of a disk operation. The current status of any disk error is always printed to the terminal.

## **FORMA Error Messages**

Error messages fall into three categories: user messages, user errors, and disk errors.

### **User Messages**

#### **FORMA: INITIALIZE COMPLETE**

This message displays when IM execution completes.

#### **FORMA: OVERRIDE SPARE COMPLETE**

This message displays when you specify the Override option.

#### **FORMA: RESTORING DATA**

After sparing, the whole track is re-written to the disk.

#### **FORMA: SAVING SECTOR XX**

This message indicates each sector being spared.

#### **FORMA: SPARING COMPLETE**

The standard spare operation is complete.

#### **FORMA: UNSPARE COMPLETE**

This message displays when you specify the Unspare option.

### **User Errors**

The following messages print at the terminal when FORMA encounters a non-standard condition.

#### **FORMA: ILLEGAL COMMAND**

You entered an unsupported mnemonic.

#### **FORMA: ILLEGAL DISK LU**

Either the logical unit specified is not generated as an integrated hard disk, or the wrong LU was entered. FORMA does not accept any microfloppy disk LUs or non-integrated disk LUs.

**FORMA: ILLEGAL PARAMETER**

A parameter was left out or the wrong values were inserted in the fields.

**FORMA: ILLEGAL RESPONSE**

An interactive mode prompt was answered with an illegal type, either alpha when numeric was requested, or the reverse. The prompt is repeated.

**FORMA: ILLEGAL USE OF OVERRIDE FIELD**

You specified something other than U or O in the last field.

**FORMA: INSUFFICIENT CAPABILITY**

The user is not a superuser.

**FORMA: MISSING PARAMETER**

You entered too few parameters in the runstring.

**FORMA: NOT AN OPTION**

You entered characters other than SS or IM as a command.

**Disk Errors****FORMA: CONTROLLER FAULT**

The controller has passed a status of 11400B, indicating a possible hardware problem.

**FORMA: DRIVE NOT READY**

The disk drive does not respond to the first IO request. The problem may be related to the hardware configuration, the generation IO configuration, or an incorrect configuration of the shunt block on the mechanism.

**FORMA: FATAL INTERNAL ERROR**

The status from the HP 12022 controller does not match any defined status. Check the ribbon cables, processor/memory controller, or other hardware.



## Disk Error Decoding

Because the disk, disk controller, and interface card are all involved in disk reads and writes, running FORMA could result in the disk interface driver (ID\*27) detecting an error. When illegal status is detected, this error code is displayed (when in runstring mode) as follows:

```
FORMA: DVT16 = xxxxxxB DVT17 = xxxxxxB DVT18 = xxxxxxB DVT19 =  
xxxxxxB
```

This is the status returned by the 12022 controller when an error condition occurs. Use the following chart to decode the information.

DVT16	DVT18	DVT19	
1	0	0	Request Rejected by Driver (Set by Reject)
2	0	100002B	Drive Not Ready or Seek Not Done
3	0	0	Timeout on DMA
5	0	0	Parity Error on Transmission
10B	0	0	Corrected Data Error
12B	1400B	0	Configuration Switches Indicate No Drive is Connected, Set by No Drive
12B	11400B	100020B	Busy, Write Fault, TR00 Error, No Handshake Completion
100077B	4400B	0	Bad Block or ECC/CRC Error
100077B	4400B	100004B	ID Not Found or DAM Not Found

DVT17 always contains 0.

For more information on the operation of Driver ID\*27, see the *RTE-A Driver Reference Manual*.

## Error Rate Test Utility (ERTSH)

ERTSH is a disk exerciser utility program that tests the functionality of a disk and its controller, including the disk media and the mechanisms needed to read and write data. It can be used with any of the following disks connected via the HP 12022 disk controller:

- 10MB Winchester, Seagate ST-412
- 15MB Winchester, Seagate ST-419
- 20MB Winchester, Seagate ST-225

The ERTSH utility is generally used with FORMA to find and verify faulty disk sectors so they can be spared. Note that FORMA and ERTSH require at least 256 KBytes to run.

You can perform a series of reads from the disk without erasing data on it, in order to test the entire disk unit or a portion of the disk. A write/read test is more thorough, because more of the disk and disk controller are exercised; however, a write/read test erases data by writing to, then reading from the disk. A variety of test patterns are used in ERTSH's write/read test, thereby increasing the likelihood of detecting errors with the disk media or mechanism.

The number of test passes, error list options, and other user definable options make ERTSH a useful tool in many situations.

### ERTSH Structure

A standard test pass of ERTSH consists of two or three phases. First is the standard read or write/read phase, which uses large disk requests to detect as many problems as possible in a short amount of time. This phase terminates when the desired number of passes have completed or the BR (Break) command is used.

For write-then-read tests, a cleanup phase is included so that the disk is left in a consistent state. ERTSH does not respond to a BR command when cleaning up.

To run ERTSH, enter the following runstring:

```
CI> RU,ERTSH[,commands][<options>]
```

Separate commands that you enter in the runstring with colons. Use blanks or commas to separate command parameters.

Alternatively, you can run ERTSH interactively and enter commands and parameters when you are prompted. If you do not enter commands in the runstring or an EX (Exit) command, ERTSH enters interactive mode.


For example, the runstring

```
CI> RU,ERTSH LU 15:WR
```

executes a write-then-read test on disk LU 15, then enters interactive mode. If you include an EX command in the runstring,

```
CI> RU,ERTSH LU 15:WR:EX
```

ERTSH terminates after the write-then-read test.



If ERTSH detects an error in a command while processing a string of commands, it displays an error message and ignores the rest of the commands. In interactive mode, ERTSH returns with another prompt or terminates if it is processing runstring commands.

## ERTSH Commands

The commands available for use with ERTSH are summarized in Table 5-6 and discussed in the following sections.

**Table 5-6. ERTSH Commands Summary**

Commands		Description
<b>Information Command</b>		
<b>HElp</b>	??	Displays all ERTSH commands
<b>Testing Commands</b>		
<b>BP</b> Build Test Pattern	patterns	Defines or displays the user definable test data pattern
<b>CY</b> Define Cylinder Range	low/high	Defines a subrange of disk cylinders for error rate test
<b>HD</b> Define Disk Head Range	low/high	Defines subrange of disk heads for ERT test
<b>LL</b> Extra Log	lu	Displays the current extra log device or sets up an extra log device
<b>LU</b> Selecting disk Lu to Test	LU_number	Selects the disk LU for testing
<b>P#</b> Set Test Pass	pass count	Defines the number of passes over the disk range for an error rate test
<b>RO</b> Execute Read-Only	sq/rn	Executes a read-only error rate test over the specified disk range
<b>SC</b> Define Disk Sector	low/high	Defines a subrange of disk sectors for error rate test
<b>SS</b> Spare Sectors	hd/al	Spare all defective sectors found during a test pass
<b>TP</b> Select Test Pattern	pattern num	Selects one or more of the 19 defined test data patterns for the write-then-read error rate test
<b>WR</b> Execute Write/Read Error Rate Test	sq/rn	Executes a write-then-read error rate test over the specified disk range
<b>Exit Command</b>		
<b>EX</b> Exit ERTSH		Exits the ERTSH utility

## Help (??)

**Purpose:** Displays a list of all the available commands or specific information about the command entered.

**Syntax:** ?? [command]

**Description:**

If you enter an invalid command, a list of all the commands is displayed.

## Build Test Pattern (BP)

**Purpose:** Defines or displays the user-definable test data pattern.

**Syntax:** BP [word] ... [word]

word                    The test data pattern, from one to six words long.

**Description:**

Each word of the pattern can be decimal, octal, or ASCII data. For example, you can specify a test pattern of all ASCII blanks as 20040B. Since blanks, commas, and semicolons have special meaning in the command syntax, you may not use them as ASCII values.

When you enter the BP command without any parameters, the program displays the test patterns and shows which ones are currently in use.

The following example shows the patterns listed with BP:

```
ERTSH (?? for help) > bp
Active test patterns = 1 2 3 4 5 19
19) 177777B
```

## Define Cylinder Range (CY)

**Purpose:** Defines a subrange of disk cylinders for error rate test.

**Syntax:** CY [low cylinder] [high cylinder]

low/high              The cylinder range.  
cylinder

**Description:**

You can define a sub-area of a disk LU by specifying a cylinder range, a head range, and a sector range. The default cylinder range is all cylinders within the current test LU. You must define a test LU with an LU command prior to setting the cylinder range.



If you enter only one value, the high and low cylinder are set to that value. If you do not enter any parameters, the current cylinder range and the maximum cylinder range are given as follows:

Current cylinder range is 120 to 125  
Maximum cylinder range is 104 to 158

## **EXIT (EX)**

Purpose: Exits ERTSH after any necessary cleanup is performed.

Syntax: EX or /E

Description:

ERTSH does not respond to a BR (Break) command during the cleanup process.

## **Define Disk Head Range (HD)**

Purpose: Defines a subrange of disk heads for ERT test.

Syntax: HD [low head] [high head]  
low/high head      The disk head range.

Description:

You can define a sub-area of a disk LU by specifying a cylinder range, a head range, and a sector range. The default head range is all disk heads within the current test LU. You must define a test LU with the LU command prior to setting the disk head range.

If you enter only one parameter, the head range is defined as that head. If you enter no parameters, the current head range and the maximum head range are given as follows:

Current head range is 2 to 3  
Maximum head range is 0 to 3

## **Extra Log (LL)**

Purpose: Displays the current extra log device or sets up an extra log device.

Syntax: LL [lu]  
lu      The LU of the log device.

Description:

All user messages are written to the extra list device if it exists. Setting the LU to zero terminates the use of the extra log device. Entering the LL command with no parameters causes the current extra log device to be displayed.

If the extra log device is not ready, or an error occurs while writing to the log device, ERTSH discontinues using the device and issues an error message.

### Selecting Disk LU to Test (LU)

**Purpose:** Selects the disk LU for testing.

**Syntax:** LU [lu\_number]  
lu\_number      The disk test LU number.

#### Description:

The LU does not have to be mounted to the file system. It cannot be mounted if a write-then-read error rate test is to be performed, because all disk data is overwritten. You must define a disk test LU before a CY, HD or SC command can change the test range or before an RO or WR command can execute a test.

If you do not specify an LU, a list of all disk LUs in the system that can be tested is displayed, along with the currently selected LU.

The following is a sample list of disk LUs:

```
  Sel. Code  Unit   LUs
    32B      0     10, 15, 36, 37, 38, 39, 55, 56, 57, 58, 59, 60
Current disk test LU is 15
```

### Set Test Pass Count (P#)

**Purpose:** Defines the number of passes over the disk range for an error rate test.

**Syntax:** P# [number]  
number      The number of passes, from 0 to 32767.

#### Description:

Entering a zero as the number parameter specifies an infinite number of passes.

For test passes that use sequential access, one pass is defined as the number of disk accesses required to check each part of the disk test range once. Because random access tests are not guaranteed to access the whole disk range, one (random) pass is defined as the number of disk accesses required to randomly reach a defective track in the middle of the test disk range.

The pass count has an initial value of 12 so that a test can be performed without prior execution of this command.

If you enter the P# command without a parameter, the current pass count is returned.

## Execute Read-Only Error Rate Test (RO)

Purpose: Executes a read-only error rate test over the specified disk range.

Syntax: RO [SQ/RN]

sq/rn                    The specification of a sequential (sq) or random (rn) pass test.

### Description:

The sequential access test covers the entire disk range in a uniform manner. The random access test accesses random disk locations and may not access 100% of the disk media. If you do not enter any parameter, sequential access is used.

Note that you must define a disk test LU prior to executing a read-only test.

Refer to the subsequent section on "Using ERTSH" for more information concerning the use of each type of test.

## Define Disk Sector Range (SC)

Purpose: Defines a subrange of disk sectors for error rate test.

Syntax: SC [low sector] [high sector]

low/high                The specification of the sector range.  
sector

### Description:

You can define a sub-area of a disk LU by specifying a cylinder range, a head range, and a sector range. The default sector range is all disk sectors. You must define a disk test LU before a sector range is defined.

If you only specify one parameter, the sector range consists of the given sector. If you specify no parameters, the current sector range and the maximum sector range are given as follows:

Current Sector Range is 10 to 20  
Maximum Sector Range is 0 to 30

## Spare Sectors (SS)

**Purpose:** Spares all defective sectors found during a test pass.

**Syntax:** SS [HD/AL]

HD/AL            HD only spares those sectors that contain defects that cannot be corrected. AL causes all sectors with defects to be spared.

**Description:**

Errors that cannot be corrected are described as error type 2, 3, or 4 in the error summary. ERTSH schedules FORMA to perform the sparing function, so FORMA must be RP'd prior to running ERTSH. ERTSH cannot spare multiple defective sectors per track or spare when the sector is unknown.

## Select Test Pattern (TP)

**Purpose:** Selects one or more of the 19 defined test data patterns for the write-then-read error rate test.

**Syntax:** TP [pattern number] ... [pattern number]

pattern            The definition of the data pattern to use on each write-then-read test pass.

**Description:**

If more than one test data pattern is selected, a different data pattern is used on each write-then-read test pass. If you enter the TP command with no parameters, the test data patterns are displayed on the system console along with the current pattern choices.

The default set of data patterns consists of patterns 1 through 5, and 19. Patterns 1 through 7 check the controller's data collection capability. Patterns 8 through 18 test the disk media. These 10 patterns are actually rotations of two unique bit patterns designed to check each bit at every word in a sector.

You can redefine the last data pattern, number 19. This is useful if an error is suspected when a specific bit pattern is written on the disk. Refer to the discussion of the BP command for more information on setting pattern number 19.

When you enter a TP command with any parameters, a list of the current test patterns displays, as shown in the following example:

Active test patterns = 1 2 3 4 5 19

```
1)      0B
2)    125252B
3)    111111B    22222B    44444B
4)      66666B      155555B    133333B
5)    22222B    133333B    111111B    155555B    44444B    66666B
6)    165352B
7)    165110B
8)    165656B    42425B    127272B
9)    153535B    72727B    56565B
10)   127272B    144530B    135353B
11)   56565B    156565B    72727B
12)   153535B    127272B    165656B
13)   72727B    56565B    21213B
14)   4102B    10204B    20410B    41020B    102041B
15)   10204B    20410B    41020B    102041B    4102B
16)   20410B    41020B    102041B    4102B    10204B
17)   41020B    102041B    4102B    10204B    20410B
18)   102041B    4102B    10204B    20410B    41020B
19)   177777B
```

### Execute Write/Read Error Rate Test (WR)

Purpose: Execute a write-then-read error rate test over the specified disk range.

Syntax: WR [SQ/RN]

sq/rn                    The specification of sequential (sq) or random (rn) access.

---

**Caution**    The WR command overwrites data on the specified LU. Be sure to back up important data before using this command. You can run a non-destructive, though less efficient, test by selecting the RO (Read Only) command instead, as follows:

```
CI> RU,ERTSH,lu <lu>:wr:ss:ex
```

or

```
CI> RU,ERTSH,lu <lu>:ro:ss:ex
```

---

## Description:

The LU cannot be mounted to the file system when this test is executed. A data pattern is written on the disk and read back. The data read is then compared to the data written. The parameter defines the use of a sequential or random access, as in the RO command. The write-then-read test is more thorough than a read-only test, because more of the disk mechanism is used and worst case data can be placed on the disk.

When you enter the command in response to an interactive prompt, ERTSH asks for permission to overwrite the disk data. For example:

```
ERTSH (?? for help)> wr
```

```
All data on LU 56 will be destroyed, OK to proceed (YES or NO)? > YES
```

## Using ERTSH

You can use ERTSH in several different ways to find and isolate disk errors. The following sections contain suggestions on how to use ERTSH.

### Disks Formatted with FORMF

You can use ERTSH to test disk drives that were formatted using FORMF, but you cannot programmatically correct any defects that are found until you re-initialize the disk with FORMA.

Note that FORMA addresses sectors differently than FORMF. This means that you cannot use FORMA to correct defects found by running ERTSH on a disk formatted with FORMF. See the earlier section on “Converting from FORMF to FORMA” for additional information.

### Initializing and Testing a New Disk

ERTSH is most effective when you use it after you use FORMA to initialize and spare a new disk. See the section “Installing a New Disk” in the earlier discussion of FORMA for the recommended procedure.

## Typical Use of ERTSH

A disk problem is usually suspected when the disk driver returns an error message or when disk data appears to be corrupt. Errors usually appear when a particular disk location is being accessed. The following example illustrates a typical run of ERTSH.

While accessing a file on LU 56, a disk error is reported that you want to correct. You should back up this LU before running ERTSH, because you may need to use a write-before-read test to isolate the error. After backing up the LU, run ERTSH and specify the LU.

For example:

```
CI> RU,ERTSH  
ERTSH Rev. 2526 <850524.1705>  
ERTSH (?? for help) > lu 56
```

Next, select either a write-then-read or read-only test to run. The write-then-read test is more thorough and usually detects more errors. The duration of the test and the test patterns used in a write-then-read test can be defaulted to 1, 2, 3, 4, 5, or 19. Because this test is destructive, ERTSH asks if it can overwrite the current disk data. When you enter "YES", the program continues.

```
ERTSH (?? for help) > wr
```

```
All data on LU 56 will be destroyed, OK to proceed (YES or NO)?> YES
```

As the test executes, it reports the start of each test pass and any errors detected during that pass. The error messages include the sector in error, disk status and error register values, and an English translation of the error and status registers:

```
Write/Read Test on LU 56      Pass #1      Using test pattern #1  
Write/Read Test on LU 56      Pass #2      Using test pattern #2  
Disk Error on LU 56      Cylinder = 137      Head = 2      Sector = 9  
Status = 121B      Error = 100B  
ECC/CRC error in data field on read  
  
Disk Error on LU 56      Cylinder = 154      Head = 4      Sector = 15  
Status = 124B      Error = 0B      Data error corrected by ECC  
  
Write/Read Test on LU 56      Pass #3      Using test pattern #3  
Disk Error on LU 56      Cylinder = 137      Head = 2      Sector = 9  
Status = 121B      Error = 100B  
ECC/CRC error in data field on read  
  
Write/Read Test on LU 56      Pass #4      Using test pattern #4  
Write/Read Test on LU 56      Pass #5      Using test pattern #5  
Write/Read Test on LU 56      Pass #6      Using test pattern #19  
Write/Read Test on LU 56      Pass #7      Using test pattern #1  
Write/Read Test on LU 56      Pass #8      Using test pattern #2  
Disk Error on LU 56      Cylinder = 137      Head = 2      Sector = 9  
Status = 121B      Error = 100B  
ECC/CRC error in data field on read  
  
Disk Error on LU 56      Cylinder = 154      Head = 4      Sector = 15  
Status = 124B      Error = 0B  
Data error corrected by ECC  
  
Write/Read Test on LU 56      Pass #9      Using test pattern #3  
Disk Error on LU 56      Cylinder = 137      Head = 2      Sector = 9  
Status = 121B      Error = 100B  
ECC/CRC error in data field on read  
  
Write/Read Test on LU 56      Pass #10     Using test pattern #4  
Write/Read Test on LU 56      Pass #11     Using test pattern #5  
Write/Read Test on LU 56      Pass #12     Using test pattern #19  
  
Cleaning up disk test area
```

After completing 12 test passes (the default pass count) and cleaning up the test disk area, ERTSH reports a summary of the errors:

Error Summary for LU 28

Table entry format                      Most severe error code - Error count for track

- 1 Soft error: ECC corrected error
- 2 Hard error: ECC/CRC error, ID not found, etc.
- 3 Data miscompare undetected by controller
- 4 Error on EXEC call to the disk

Cleaning up disk test area

Cyl./Sc	Head Number					
	0	1	2	3	4	5
137/ 9			2 4			
154/15					1 2	

An error summary entry contains a general error code (1-4) followed by the total number of times an error was detected at that sector. In the example above, the error summary indicates that there are two defective sectors. The first error (cylinder 137, head 2, sector 9) has an error code 2 or a hard error that occurred 4 times. The error is so severe that the data is probably not recoverable and should therefore be spared.

The second entry (cylinder 154, head 4, sector 15) indicates the error correction code (ECC) is correcting the error, so the data read from that sector may still be valid. Even though data can be written and read correctly (using error correction), you should spare the sector before the defect gets worse and becomes a hard error. For more information, refer to the "Error Interpretation" section, later in this chapter.

You can spare the defective sectors by using the `SS` command from ERTSH, or you can exit the utility and use `FORMA` directly. In the following sequence, the `SS` command is used to have ERTSH schedule `FORMA` to spare the defective sectors. Since the default `SS` command parameter is used, all the defective sectors are spared.

```
ERTSH (?? for help) > ss
```

```
Sparing sector at Cylinder = 137      Head = 2      Sector = 9
FORMA: INTEGRATED DISK FORMATTER, REV 2526 <850530.0911>
FORMA: SPARING COMPLETE
```

```
Sparing sector at Cylinder = 154      Head = 4      Sector = 15
FORMA: INTEGRATED DISK FORMATTER, REV 2526 <850530.0911>
FORMA: SPARING COMPLETE
```

```
ERTSH (?? for help) > ex
```

Note that it takes about two hours for this sequence to test for errors, isolate them, and spare the defective sectors. When the test completes, you can exit ERTSH and restore the files from your backup device.



If ERTSH detects an error in a command while processing a string of commands, it displays an error message and ignores the rest of the commands. In interactive mode, ERTSH returns with another prompt or terminates if it is processing runstring commands.

## Break Detection

You can enter the RTE BR (Break) command to terminate an error rate test before the defined number of test passes complete. Upon detecting the break, ERTSH cleans up the disk and reports the error summary to the terminal. Setting the test pass counter to zero (infinity) and running a test that you terminate with the BR command is an easy way to perform a lengthy test.

## Specialized Use of ERTSH

If a disk error is encountered while a program is attempting to read a certain track from a disk LU, you can limit the disk test area to reduce the test time because you know the general disk location.

First, translate the track address to an absolute cylinder and head address. After selecting an LU with the LU command, use the CY and HD commands to display the physical cylinder and head bounds for the given LU. You can use the following calculations to get a precise disk area to test, or you may be able to approximate the location of the defect and select an appropriate disk test area, as shown in the following example:

```
1st LU track = (1st LU cylinder) * (# of heads on disk)
absolute track = (1st LU track) + (track # within track)
absolute cylinder = (absolute track) Div (# of heads on test LU)
absolute head = (absolute track) Modulo (# of heads on test LU)
```

Some of the above values can be found in the system generation listing. Since some defects cross track boundaries, you should also test the tracks adjacent to the suspected track.

For example, if the suspected disk location is somewhere on the track at cylinder 23, head 2, it is appropriate to test the area from cylinder 20 to cylinder 25. You can test all the heads or just head 2, depending upon how thorough you want the test to be. Testing a small portion of the disk is much faster than testing the entire disk.

In deciding how thorough a test to perform, consider test duration and the selected test pattern (if the test is a write-then-read test). As shown in the sample listing, a disk defect may not be detected on every access to that sector. In general, the smaller the defect and the closer it is to the start or end of a sector, the less likely it is to be detected on each pass of ERTSH.

Another factor related to detecting an error is the bit pattern written on the disk. Some defects appear to be pattern sensitive and may require using several different patterns before the defect is isolated. ERTSH contains 18 standard test patterns and one user-definable test pattern number (19).

For the write-then-read test, a track size buffer is created by duplicating a sector buffer built by repeating the values in a given pattern. For example, test pattern 3 consists of the three values 111111, 22222, and 44444 octal.

The sector buffer built from these values is as follows:

```
111111B 22222B 44444B 111111B 22222B 44444B 111111B
```

and so on. This particular pattern corresponds to a bit pattern of a binary 1 followed by two binary zeros (100100100100...).

It can be very difficult to select a test pattern. There is no way to know ahead of time which pattern will detect a given error. However, it is known that severe errors are usually detected by several test patterns. The test pattern defaults usually detect defects, but you may need to use other patterns if a particular defect is not isolated.

Patterns 6 and 7 stress the controller's ability to distinguish between data bits within a word. Patterns 8-13 and 14-18 are permutations of two basic test patterns that are rotated to check all bit positions. An ideal test would run several passes with each test pattern, but this becomes prohibitive due to the time required.

You can use the user-definable pattern (number 19) if you suspect that data has been corrupted and the disk defect is pattern sensitive. For example, a file is supposed to contain the value 100 in a given word, but the value 101 is returned when the file is read. A test pattern of 100 causes the value of 100 to be written, then read and compared on the whole disk range. The pattern can be defined as follows:

```
ERTSH (?? for help) > bp 100
```

## Installation Procedure Summary

Here is a summary of steps to take when you install a new internal hard disk or run FORMA/ERTSH (except for Step 1) on a disk currently in FORMF format.

1. Find the label attached to the top of the disk drive that lists the areas that the manufacturer has found to be defective. Record the CYL, HD, and BYTE/INDEX information. This data is used in Step 3, in the FORMA runstring.

---

**Caution** The initializing process overwrites data on the entire disk. Be sure to back up your data before initializing disks currently in use.

New disks are shipped from HP already initialized and spared in FORMA format, and need not be initialized again. If you are using new disks, skip to Step 4.

---

2. Initialize the hard disk by running FORMA, using the IM command. If the drive was never formatted (or is currently in FORMF format), use the N option to prevent spares from being retained, to save time. If the drive was already formatted and spared using FORMA, use the P option to save the old spares.

```
CI> RU, FORMA, IM, <lu><n/p>
```

This should take about 40 minutes for a 15MB drive.

3. Spare the defects by using the FORMA SS command and entering the data recorded in Step 1.

```
CI> RU,FORMA,SS,<lu>,N,B,<cyl>,<hd><byte/index>
```

4. Test for new defects by running ERTSH. Using the SS command, ERTSH locates the defects and programmatically calls FORMA to spare them.

## Error Interpretation

ERTSH reports all disk errors (including those that cannot be corrected). Each error is reported when it occurs during an error rate test, and a summary of error locations is listed at the end of each test.

The error messages contain the cylinder, head, and sector where the error was detected along with the status and error code. For example, if an ECC error is detected, the following error is displayed:

```
Disk Error on logical LU 16      Cylinder = 113      Head = 1      Sector = 3
Status = 124B      Error = 0B
Data error corrected by ECC
```

The first line indicates the location of the error. This information is needed if the defective disk area is to be spared later. The second line contains the disk status and error register returned from the controller. Note that this is not the same status that is returned from the driver on a standard request to the disk driver or EXEC call. The last item in the error message is an English translation of the error status and error bytes.

The other type of error message indicates a data miscompare. Again, the disk location is identified, along with the data which caused the miscompare.

```
Disk Error on LU 16      Cylinder = 101      Head = 2      Sector = 13
Miscompare word = 11
Value written = 0000000000000000      0B
Value read    = 00000000000000100      4B
```

The word in the sector that caused the error is identified along with the expected and actual value read from the disk. The values are first given as binary, then octal values.

Occasionally, ERTSH reports an error with a sector number of “?”. This happens when ERTSH detects an error on the track but cannot determine the sector that is at fault. If “?” also appears in the error summary, check the number of times the error occurred. If the total number of errors for that sector is only one or two, the error was probably transient and you can ignore it. If the error count is large, retest the area around the track in question.

Note that ERTSH reports error conditions that the disk controller or drive may mask from you. For example, even if the disks ECC/CRC correct an error in disk data, and the disk data is now correct, ERTSH still reports an error condition.

At the end of a test, an error summary is created to help in error interpretation. Errors are grouped into four categories when placed in the error summary, as shown in the following sample output.

Error Summary for LU 56

Table entry format            Most severe error code - Error count for track

- 1 Soft error: ECC corrected error
- 2 Hard error: ECC/CRC error, ID not found, etc.
- 3 Data miscompare undetected by controller
- 4 Error on EXEC call to the disk

Soft errors are data reads from the disk that were corrected by the disk controller. This ECC correction is a standard mechanism used in many disk drives to compensate for minute defects in the disk media. Even though the data in these sectors is not corrupt, you should spare the sectors before a defect becomes a hard error.

Hard errors are more severe than soft errors, and are usually easier to detect. Hard errors can be caused by a disk defect in the sector data area or sector header area or by a defective HP 12022 interface card. The following errors are all classified as hard errors:

Bad block mark found in ID field  
ECC/CRC error in data field on read  
ID field for given location not found  
Unable to seek to track zero  
Data address mark not found after ID field  
Last command aborted due to error

Card is still busy  
Drive not ready  
Write fault  
Seek not complete  
Disk is write-protected

In the messages above, the first six errors are encoded in the error register value, and the last five are encoded in the status register.

Data miscompare errors indicate that a malfunction of the disk or disk controller was not detected by the controller but was detected by ERTSH. This error usually indicates a defective HP 12022 controller.

An error on EXEC call may be caused by a defective HP 12022 interface or a defective disk; or the problem can be simpler, such as not having the disk cables on tightly or not having the disk powered up.

When you interpret the table as a whole, you must consider other factors. For example, an error table like the following one may indicate one of two likely errors.

Cyl./Sc	Head Number							
	0	1	2	3	4	5	6	7
104/ 0					4	4	4	4
104/ 1					4	4	4	4
104/ 2					4	4	4	4
104/ 3					4	4	4	4
104/ 4					4	4	4	4
104/ 5					4	4	4	4
104/ 6					4	4	4	4
104/ 7					4	4	4	4

The error summary above indicates that every sector on every track using head 4 and 5 is defective. The real problem may be that heads 4 and 5 of the disk drive do not work, or they may not even exist. A 10 MByte disk (which has 4 heads) may have been used instead of a 15 MByte disk (which has 6 heads).

In this case, it does not make sense to spare every defect shown in the table, because the real problem is not defective sectors. Whenever you see a large number of errors in the summary, look for causes other than defective sectors.

## Error Messages

When ERTSH detects an error, it first prints:

```
ERTSH Error: Error in xx command entered
```

This indicates which command in the sequence caused the error and is followed by one of the messages listed below.

### **ERTSH Error: Command requires superuser capabilities**

Commands that cause disk data to be destroyed can only be executed by users who have superuser capability.

### **ERTSH Error: Defects on microfloppy disk cannot be spared**

Defects on microfloppy drives cannot be spared using ERTSH.

### **ERTSH Error: Disk LU x is mounted to file system Command can not be executed.**

Commands that cause disk data to be destroyed cannot be executed while the disk LU is mounted to the file system.

### **ERTSH Error: Error in call to LU xx Error = xxxx**

An error was detected during an I/O request to the given LU. The error code xxxx is a system error code such as IOTO (device timeout).

### **ERTSH Error: Error summary table is full, error not in summary**

The error summary table is full. Errors are still reported, but may not appear in the error summary at the end of the test pass.

### **ERTSH Error: Illegal Disk LU**

The LU you entered is not an integrated disk LU.

### **ERTSH Error: Internal Error xx**

ERTSH encountered an unexpected situation. Please report this error to your local HP Representative.

### **ERTSH Error: Invalid parameter**

The parameter you entered is inappropriate for the command.

### **ERTSH Error: IO error on device LU xx**

There was an error on the I/O request to the given device.

### **ERTSH Error: Lower bound greater than upper bound**

You specified a range in which the lower bound was larger than the upper.

**ERTSH Error: No test LU selected for test**

You must select a test disk LU before a test pass can be executed.

**ERTSH Error: No tracks on test LU lie within cylinder/head range**

The head and cylinder range do not contain any tracks from the test LU. Redefine the cylinder and/or head test range.

**ERTSH Error: Parameter xx is not a valid integer**

The parameter you entered is not a valid integer.

**ERTSH Error: Unable to lock Disk LU xx**

ERTSH must be able to lock the test LU in order to perform the test.

**ERTSH Error: Unable to unlock Disk LU xx**

ERTSH was unable to unlock an LU it had previously locked.

**ERTSH Error: Unknown command, type ?? for help**

You specified an unknown or invalid command mnemonic.

**ERTSH Error: xx is out of range**

The parameter you entered is too large or too small.

# File Manipulation Utilities

---

This chapter discusses the file manipulation utilities MERGE, OLDRE, and SCOM.

## 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, 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:

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 6-1.

**Table 6-1. 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) may 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 identical to the first input file. 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 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
```

## Extended Record Converter (OLDRE)

OLDRE is a stand-alone utility that provides backward compatibility with loaders and generators that do not accept extended record formats. OLDRE translates the 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 that 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 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.
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

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.

## 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> [RU,] 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 6-2. Additional information about some of the options is presented in the following sections.

**Table 6-2. 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. You can specify IB to override TB in ET, ER, BB, and BR.

## **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  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 = the number of the file on which the error occurred:  
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

### Example 1:

```
CI> SCOM,TEST1,TEST2
Scom: comparison performed Sat Nov 21, 1992  4:32:17 pm
  file1 is TEST1
  file2 is TEST2
  options: flf2;  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: flf2nn; 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: flf2bo;  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: flf2bonn;  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>
EOF
/SC
```

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: flf2bod^;  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      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     16     END
18     16     $
19     16     ENDS$
-----
```

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
Scm: comparison performed Sat Nov 21, 1992  4:42:45 pm
  file1 is TEST3
  file2 is TEST4
  options: flf2bo; rematchlns: 1; maxchars: 16
1      1      ASMB,R,L,C
2      2              NAM TEST3
3      3      *
4      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: flf2boc*; 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)

## System Setup Utilities

---

### Absolute Binary to Memory Image (AB2MI)

AB2MI converts an arbitrary type 7 file to a type 1 file in memory image format. Thus, any arbitrary code, whether or not it was produced by the system generator, may be loaded into processor memory. This is useful because memory image format is required by the disk bootstrap program.

AB2MI also patches an existing type 1 file. In this case, the input file overlays selected portions of the output file.

### Calling AB2MI

To call AB2MI, enter the following runstring:

```
CI> [RU,] AB2MI <,input filename> <,output filename>
```

The input file name is the filename of a type 7 file or device. The outputfilename is the filename of a type 1 file.

You must enter either all the parameters or no parameters in the runstring. If you enter none, AB2MI prompts you interactively, as follows:

```
This program copies an absolute binary (type 7)
file to a memory image (type 1) file.
```

```
Please enter the input and output filenames.
Format: input::directory, output::directory::size
```

## AB2MI Operation

If the output file does not exist, it is created by AB2MI (default size, 256 blocks); otherwise, it is overlaid.

At the conclusion of the conversion, the program displays the message:

```
Highest block written: nnn
```

Here nnn is the number of blocks required to contain the input file data.

## Break Detection

To stop execution of the program, you may enter the BReak command as follows:

```
CI> BR,AB2MI
```

When AB2MI detects a break, it issues the following message:

```
Break Flag Set
```

The program then closes the input and output files and terminates. Note that the output file is incomplete if the break flag is set during a conversion.

## AB2MI Error Messages

### Break Flag Set

The break flag is tested in each major loop within the program. When the break flag is detected, the program closes the input and output files and terminates.

### Checksum Error

The checksum is computed on the input record and must match the checksum word stored in the record itself. This error can result from specifying an input file of the wrong type (that is, ASCII).

### Input File Error xxxxxx

### Output File Error xxxxxx

Any error arising from calling a file management subroutine results in this message. The file error code is reported and the files are closed.

## Initialize BOOTEX File (INSTL)

The INSTL utility lets you initialize the boot extension (BOOTEX) file for an RTE-A disk LU. You may run INSTL on an RTE-A or RTE-6/VM system.

### INSTL Overview

When you boot the RTE-A system, enter (either explicitly or by default) the physical address of BOOTEX and the name of a type 1 system file or a type 4 bootup command file. The VCP/loader ROM loads BOOTEX from that address into main memory and starts executing the boot extension. The boot extension either loads the specified system file into main memory and starts executing or refers to the bootup command file and follows the instructions for booting up.

This assumes that the boot extension is properly located. The VCP/loader ROM does not check what it is loading. It goes to the address specified in the bootup command, loads whatever is there into memory, and attempts to execute it. If you are booting from a disk, make sure BOOTEX is in the right place with the INSTL program and the CI (or FMGR) IN command. When you boot from a disk, the file that contains the boot extension must be on physical cylinder 0 and sector 0 of the boot LU. (This implies, of course, that the disk LU starts at physical cylinder 0; otherwise, the system cannot be booted from that LU.)

When you use INSTL to initialize a cartridge that will actually be used on a differently configured disk LU—for example, one set up for a target system other than the host system—INSTL gets the information it requires from the specified snapshot file and system file of the target system. The boot extension code comes from any existing RTE-A boot extension file of the same revision.

The way INSTL and the IN command are used varies according to the type of system you are using, since there are slight differences between the operation of the IN command in RTE-A and RTE-6/VM systems.

The CI IN command allows free space to be reserved at the beginning of a disk LU. This space does not contain BOOTEX code. The FMGR IN command lets you purge all files on a disk LU and set (logical) track 0 as the starting track of the LU. When you do this in an RTE-A system, the FMGR IN command creates a file named BOOTEX (type 1, 768 blocks) at the start of the disk LU. Since this file does not contain boot extension code, you must use INSTL to put the boot extension code into the BOOTEX file. (When installing BOOTEX on a CI volume, INSTL can install it directly into the reserved space if the volume's LU number is the same as the LU number it will have at boot time. Otherwise, you can use INSTL to create a new initialized BOOTEX file which may later be put into the reserved space of any CI volume using the FPUT utility.)

In an RTE-6/VM system, you may use the FMGR IN command to purge all files and set the starting track at 0, but it does not create a boot extension file. Use INSTL to create the file and fill it with the boot extension code.



The following examples compare the use of CI and FMGR:

For CI:

```
INSTL, snap, system, 0, 16, BOOTEX      !LU 16 is a CI volume
                                         !and has the system
                                         !and snap files on it
```

or

```
INSTL, snap, system, BOOTX, 16, BOOTEX
FPUT BOOTX 16
```

For FMGR:

```
PU,BOOTEX:-32767:P1                    !P1 is an empty
INSTL, snap, system, BOOTEX::P1, 16, BOOTEX !FMGR LU. LU 16
                                         !has the system and
                                         !snap files on it
```

## Calling INSTL

To call INSTL, enter the following runstring:

```
CI> [RU,] INSTL, [snap], [system], <boot dest>, <lu>, <boot source>[, option]
```

- |           |   |
|-----------|---|
| snap      | The name of the snapshot file output by the generator. This is the snapshot file of the target system.  |
| system    | The name of the system file output by the generator. This is the system file of the target system.<br><br>If you do not enter the snap and system parameters, the default is to configure the boot extension file according to the specifications of the current (host) system (RTE-A host only).   |
| boot dest | The name of the file to contain the new boot extension (destination file). This is a file in the target system. When using the INSTL command, specify the full path in the boot destination file. Otherwise, it will default to the current working directory.<br><br>If the specified file exists as a type 1 file at least as long as the source file, INSTL puts the boot extension code into it. (If the file exists, but is not type 1 or is too small, INSTL terminates.) If the specified file does not exist, INSTL creates it and fills it with the boot extension code. |

If you are booting from a CI volume, boot dest can be a number. If boot dest is a number, it is assumed to be an offset into the reserved space on the destination LU. Space on a file system volume is reserved by the CI IN command and reported by the FREES utility. This space is used for bootable files (such as BOOTEX and diagnostics).

The offset parameter specifies where in the reserved space to put the file. It corresponds to the file number in the VCP bootstring (see your computer reference manual for information on the VCP bootstring). Given an offset of 0, the file starts at block 0 on the disk. If the disk LU is the first physical LU on the disk and is unit 0 (which CS/80 disks are), HP-IB address 0, and the HP-IB is at select code 27, then this file can be booted using “%BDC0027”. An offset of 3 starts the file at block 768 and it can be booted using “%BDC30027”. Note that there is no directory describing the files in the reserved space.

Be aware that INSTL overwrites any previous information in the reserved space without warning. For example, suppose BOOTEX, which requires 768 blocks, is installed at offset 0 of the disk. If another BOOTEX is later installed at offset 1 (starting at block 256), INSTL overwrites the last two-thirds of the original BOOTEX without warning. This error will not be detected until the next time the system is booted. To prevent this problem, we recommend that you put only BOOTEX, and nothing else, in the reserved space.

- lu This is the LU number in the target system that you boot from and which has the system and snap files on it. BOOTEX mounts it automatically for you. If boot dest is a number, this is also used as the destination LU for the boot extension.
  - boot source The name of an RTE-A boot extension file in the host system. INSTL uses the boot extension code in this file, plus information from the snap and system files, to create the boot extension code that is placed in the destination file. If you do not specify this parameter, INSTL assumes that the destination file is a valid boot extension file and uses that file as the source file (if the destination is a file, not an offset).
  - option N = No Console  
or  
E = Enable CS/80 timeout retry
- If the target system is generated without LU 1 and this option field is “N” (No Console), no system messages will be displayed on the system console when booting the target system. Thus the target system becomes a consoleless system. If the option field is “E”, this will enable the CS/80 timeout retry on the first entry into the driver from BOOTEX.

If you do not enter any parameters in the runstring, INSTL prompts you as follows:

```
ENTER SNAP FILE, SYSTEM FILE, DESTINATION FILE, LU, SOURCE FILE [,OPTION]
```

Enter the parameters requested. The destination file may already exist on the disk; if so, it must be a type 1 file at least as long as the source file. If it does not exist, a file is created using the name you enter.

## **INSTL Operation**

INSTL is not concerned with the physical location of the boot file. It puts the boot extension code in whatever file you specify, whether or not that file begins at physical cylinder 0 and sector 0 of a disk LU.

If you use the snap and system parameter defaults, INSTL configures the destination file according to the specifications of the host system. If the source file is defaulted, INSTL assumes that the destination file is a valid boot extension file and uses the destination file as its source. If the source is defaulted and the destination file is an offset number, INSTL quits.

## **Successful completion is indicated by the message:**

```
INSTL END.  AAAAAA IS YOUR BOOT EXTENSION FILE.  
WARNING: BOOT FILE MUST  
BE AT CYL 0, HEAD 0, SECT 0.
```

AAAAAA indicates the actual filename.



## **INSTL Error Messages**

### **BOOT SOURCE FILE NOT TYPE 1.**

The source file must be type 1. The program terminates after the error is found.

### **DESTINATION FILE TOO SMALL**

The destination file must be at least as large as the source. The program terminates after the error is found.

### **INSTL end. AAAAAA is your boot extension file. warning: boot file must be at cyl 0, sect 0.**

These two messages are given at the successful completion of the program when the destination is a file. The destination file must be at cylinder 0, sector 0 in order to be accessed by the ROM bootstrap code. Note that it must also start at head 0 if the disk is formatted in cylinder mode. The message is given whether or not this condition is found.

### **INSTL end. Your boot extension has been installed at boot block xx on lu yy.**

This message is given at the successful completion of the program when the destination offset is a boot block number.

### **LU XXXX IS NOT A DISK LU.**

The destination file can be created only on a disk. The program terminates after the error is found.

---

### **Warning** Base page links may be overwritten by driver parameters.

This warning is issued if INSTL suspects that locations 1416-1536 of the source BOOTEX are base page links. INSTL writes the driver parameters of the target disk into these locations of the destination BOOTEX. Thus, any base page links in these locations are destroyed. This can only occur if a new BOOTEX is being built with a BTXL supplied by you that requires substantially more page links than the HP-supplied BTXL.

---

# Bootable System Installation (FPUT)

FPUT installs bootable systems and diagnostics in the space on a file system volume that was reserved by the CI IN command. You can use this to install a BOOTEX file that was initialized by INSTL, or to install diagnostics or user-configured memory-based systems in a bootable position on a file system volume.

## Running FPUT

To invoke FPUT, use the following runstring:

```
FPUT, filedescriptor, diskLU [,VCPfilenumber [,BCMoffset]]
```

where:

- filedescriptor** is the name of the file to be copied to the bootable area on the CI file system volume given by diskLU.
- diskLU** is the CI file system volume to which the specified file will be copied.
- VCPfilenumber** is an offset within the bootable area in 256-block units.
- BCMoffset** is an offset within the bootable area in 32-block units.

The VCP bootstring for a SCSI or CS/80 disk is in the form:

```
%BDCffbusc
```

where:

- ff** is a VCPfilenumber that describes an offset into the bootable area in 256-block units. Default is 0. The *ff* parameter provides the option to select one of several bootable systems stored on the same disk LU. This allows you to, for example, place more than one copy of BOOTEX on the LU, such that multiple RTE-A systems that require different revisions of BOOTEX may be booted.
- b** is the bus address of the disk.
- u** is the unit number of the disk (always 0 for SCSI or CS/80).
- sc** is the select code of the interface card.

As an example, assume that your system has LU 10 on a CS/80 disk that is unit 0, bus address 2, and select code 24. To install BOOTEX on that system, you would first run INSTL to make a copy of BOOTEX that is configured properly for the target system (see the INSTL documentation). Assuming that the output file name is "NEWBOOT" and that it is 768 blocks long, you could then run "FPUT NEWBOOT 10" to copy it to block zero of the target disk. The bootstring to boot the resulting system would be "%BDC002024". It could be abbreviated to "%BDC2024" because the *ff* parameter defaults to 0.

You may also wish to have a second copy of BOOTEX for redundancy. The FPUT command would be "FPUT NEWBOOT 10 3". The VCPfilenumber parameter is 3 because "NEWBOOT" is 768 blocks long and 768/256 is 3; so the first copy occupies VCP file numbers 0, 1, and 2. Thus, the next available location is 3. This would be booted with "%BDC032024".

The BCMoffset is used in a similar manner, except that it is in units of 32 blocks. (See the *HP 1000 A/L -Series Computer Diagnostic Operating and Troubleshooting Manual*, part number 24612-90013, for more information about the BCM--Basic Control Module.)

The absolute block number relative to the start of the disk is given by:

$$\text{Block} = (\text{VCPfilenumber} * 256) + (\text{BCMoffset} * 32)$$

## FPUT Operation

FPUT opens the file to make sure it exists. You can install any file, so long as it fits in the reserved space. Note that there is no directory describing the files in the reserved space. FPUT verifies that the passed LU is a file system volume and that you reserved enough space on the volume for this file. Reserve space by using the CI IN command. See the *RTE-A User's Manual*, part number 92077-90002, for more information about the IN command.

If the file does not fit, FPUT exits with an informative message. Files in the normal portion of the volume are not affected.

---

### Warning

**Be aware that FPUT overwrites any previous information in the reserved space without warning. For example, suppose BOOTEX, which requires 768 blocks, is installed at offset 0 of the disk. If a memory-based system is installed at offset 1 (starting at block 256), FPUT overwrites part of BOOTEX without warning.**

**This error is not detected until the next time you try to boot the system, when the cause may easily be forgotten. It is your responsibility to remember what is in this reserved space. The easiest thing to do is put BOOTEX at block 0 (INSTL will do this for you) and not put anything else there.**

---

FPUT does not install files on an FMGR cartridge. You can place files on an FMGR cartridge in a bootable position by controlling their placement in the directory. Put the first bootable file on the cartridge first (this is where BOOTEX is usually found). Additional bootable files can be placed immediately after the first one. Make sure each bootable file is on a 768-block boundary by padding shorter files with empty dummy files.

# Copy System (CSYS)

CSYS copies type 1 (memory image) files from a CS/80 disk to a cartridge tape (CTD) in a memory-based system. Files are written to the CTD in a format compatible with the RTE-A VCP (Virtual Control Panel) boot/loader file system organization. This lets you boot the system from any file you write to the tape into the system. CSYS also invokes the verify feature of the cartridge tape drive to ensure that the data is correctly stored on tape.

---

**Note**      The verify feature does not detect a defective tape. You should boot the tape to determine tape integrity.

---

Refer to the *RTE-A System Generation and Installation Manual* for details of the boot procedure and the boot command.

The CTD drive is a high capacity device that requires data to be transferred in a continuous streaming mode. A 64K byte disk buffer called a cache is created in an area reserved on the associated disk drive. The data is thus buffered and transferred to the CTD in a continuous stream. Refer to the *RTE-A Driver Reference Manual* for details of the disk-caching scheme.

CSYS runs only in an online environment with the HP 7908, 7911, 7912, 7914, 7946 disk drives and CTD drive.

## Calling CSYS

To call CSYS, enter the following runstring:

```
CI> [RU,]CSYS,<filename>,<tape lu>,<file #>,[SA:[<next file #>]],[BCM offset]
```

If you do not enter a required parameter, the runstring format is displayed and the program terminates.

filename	The name of the type 1 file on disk.
tape LU	The CTD to be copied to.
file #	The number of the file the VCP will use (this is the “target”) to boot the system from the tape.
SA	An optional parameter that specifies that the memory-based system will be hidden inside an ASAVE file. This allows ASAVE files and memory-based systems to reside on the same CTD tape. This option uses a 256-block VCP file for the ASAVE header and an additional 256-block VCP file for the trailer. When using this option, file # should not be zero.

**next file #** An optional subparameter of the SA parameter. It specifies where to put the ASAVE end-of-data record. The default value for next file # is immediately after the memory-based system.

To reserve space for more than one memory-based system:

```
next file # = last file # + {truncation to an integer of  
                        ((blk size + 255)/256) }
```

**last file #** The file # of the last file to be put on tape.

**blk size** The number of disk blocks in the last file on cartridge tape. This value is derived from the information contained in the FMGR or CI DL command. Note that a CTD block is 512 words and a disk block is 128 words.

**BCM offset** The BCM file number. The file # and the BCM offset determine the starting block for the file on tape. For example, if the file # is 1 and the BCM offset is 1, then the file starting block is 288 (256 + 32).

## CSYS Operation

CSYS first checks that the file exists and that the tape LU is that of a CTD. If those requirements are satisfied, CSYS clears the CTD tape cache, starts reading the file in 512-word blocks from the disk file, and writes to the cartridge tape.

The reading process continues until the end of the disk file is reached. When the file is copied, CSYS closes the cache and verifies the blocks just written on the tape drive. After verification, the disk file is closed and CSYS issues the message:

```
CSYS COMPLETED.  xxxx BLOCKS WRITTEN TO TAPE.
```

If you selected the SA option, the following message also appears:

```
ADDING ASAVE FORMAT
```

CSYS does not acknowledge the system BReak command; once invoked, the program continues to completion or until aborted by an error.

Note that CSYS will not run if the CTD LU is buffered by RTE-A (use the FMGR, BL command to check this condition).



## CSYS Examples

For the following examples !PBV and !XYZ are type 1 files that are 480 blocks long. The CTD tape LU is 24 with select code 27 and HPIB address 0.

### Example 1: Enter the runstring as shown.

```
CI> CSYS, !PBV, 24, 0
```

CSYS returns the prompt:

```
120 (KB) BLOCKS WRITTEN TO TAPE
```

This puts the program !PBV onto the beginning of the CTD tape. It can be booted by typing:

```
VCP> %BDC0127
```

### Example 2: Enter the following commands.

```
CI> CSYS, !PBV, 24, 1, SA: 5  
CI> CSYS, !XYZ, 24, 3  
CI> ASAVE, TA, 24 | SA, 16, AP, VE | EX
```

The first line puts ASAVE header information into CTD tape file 0, !PBV into tape files 1 and 2 and, ASAVE trailer information into tape file 5.

The second line saves !XYZ into files 3 and 4.

The third line appends an LU immediately after the ASAVE trailer information. Note that there are two ways of looking at files on tape. CSYS and the boot loader consider a file to start every 64 tape blocks. ASAVE and ARSTR consider a file to run until an ASAVE trailer record. This means that CSYS files 1,2,3,4 and 5 of this example will be put into the first ASAVE file. The second ASAVE file contains LU 16.

To boot !PBV enter:

```
VCP> %BDC010127
```

To boot !XYZ enter:

```
VCP> %BDC030127
```

To restore LU 16 enter:

```
CI> ARSTR, TA, 24 | RE, 16: 2, 2, VE | EX
```

Note that LU 16 is restored from ASAVE file number 2. ASAVE file 1 contains !PBV and !XYZ and should not be restored.

## Loading CSYS

To load CSYS, use the load file #CSYS and enter:

```
CI> link #CSYS
```

## CSYS Error Messages

The following error messages can be generated by CSYS. Unless otherwise specified, errors are fatal and immediately terminate the program.

### FILE NOT TYPE 1

The file specified is not a type 1 file; it cannot be copied to the tape in the proper format.

### ILLEGAL LU

The tape drive LU specified is the wrong type; it must be a CTD.

### WARNING. TAPE WRITE DID NOT VERIFY

Warning only. This is not a fatal error; it indicates that tape verification failed, but the file was completely written to the tape.

### TAPE IS NOT LOADED

The tape is not loaded in the tape drive.

### TAPE IS NOT INITIALIZED

The tape in the drive was not initialized (certified) by the utility FORMC.

### TAPE IS WRITE-PROTECTED

The tape is write-protected and thus data cannot be written on the tape.

### INVALID ASAVE FORMAT PARAMETER

Some error was made specifying the SA parameter on <next file#> subparameter.

Standard FMP errors are reported using the common file system error messages (that is, "FILE NOT FOUND XYZ").

## Memory Image to Absolute Binary (MI2AB)

MI2AB converts a type 1 file into a type 7 file. This makes it possible to load an RTE system into memory from a mini-cartridge, since the mini-cartridge bootstrap loader requires absolute binary format. (The system created by the generator is placed in a type 1 file.)

The system size for RTE-A is obtained from words 1 and 2 of the system file: word 1 contains the number of complete 32K blocks in the system files and word 2 contains the remaining number of words after the 32K blocks.

### Calling MI2AB

To call MI2AB, enter the following runstring:

```
CI> [RU,] MI2AB [,<input filename>, <output filename>]
```

The input filename is a type 1 file containing the system. The output filename is a type 7 file to be created or overlaid. It may also be a device LU.

If the output file does not already exist, MI2AB creates it (default size is 256 blocks); if the file exists, it is overlaid.

You must enter either all the parameters or no parameters in the runstring. If you do not enter any, MI2AB runs in interactive mode, and prompts you as follows:

```
This program copies a memory image (type 1)
file to an absolute binary (type 7) file.
```

```
Please enter the input file and output filenames.
Format: input::directory, output::directory
```

### Break Detection

The Break command is entered as follows:

```
CI> BR,MI2AB
```

When you enter BR, the following message is displayed:

```
Break Flag Set
```

When it detects a break, MI2AB closes the input and output files and terminates. Note that the output file is incomplete if the break flag is set during a conversion.

## **MI2AB Error Messages**

If the input file is larger than 32K blocks, MI2AB prints the following warning message:

**Input is greater than 32K; Converting only 32K.**

The output file is created, if necessary, and 32K blocks of the input file are converted.

MI2AB can generate the following error messages:

### **Break Flag Set**

The break flag is tested in each major loop within the program. When the break flag is detected, the program closes the input and output files and terminates.

### **Input File Error XXXXXX**

### **Output File Error XXXXXX**

Any error that results from calling a file management routine in either the input file or output file produces this message. The FMP error code is reported and the files are closed.



# NOTIFY and PRINT Utilities

---

This chapter describes the general purpose utilities, NOTIFY and PRINT.

## Using the NOTIFY Utility

The NOTIFY utility sends a one-line message to another user's terminal, in the format:

```
Message from logon[>node] date-time...  
your-message-here
```

The recipient of the message is specified as either a logon name or an alias name defined in the /system/notify.cf file, which is described below.

---

**Note**        The NOTIFY utility is installed with Mail/1000 by the installmail.cmd file of Mail/1000.

---

## Calling NOTIFY

The runstring syntax for the NOTIFY utility takes one of the following forms:

```
notify [-flags] user message  
notify off  
notify on
```

## Sending a Message

To send out a message use the following syntax format:

```
notify [-a] user message
```

where:

- a Options flag that inhibits notifying alias logons, treats “user” as a logon name only.
- user Specifies who is to receive the message, and may name either:
  - a logon name, in which case all sessions logged on under that name are notified.
  - a session number, in which case only that session is notified.
  - an alias name from /system/notify.cf, in which case the message is sent to all users defined therein.
  - the name “all”, in which case the message is sent to all sessions logged on.
- message Your one-line message to be sent by the NOTIFY utility.

In the example,

```
notify joker 'I'm Batman.'
```

the NOTIFY utility will send that message to every session logged on as JOKER on this system, plus any alias logons for JOKER defined in notify.cf.

## Turning Notification On or Off

To turn notification for your session on or off use the following formats:

```
notify on
```

or

```
notify off
```

If notification is turned off then messages sent to your session are thrown away without being printed.

## Defining Aliases for Notification

NOTIFY may send messages to aliases that are defined for:

- alternate logons for a user
- group distribution lists
- hosts in a DS network

## Alternate Logons for a User

The `/system/notify.cf` file defines alternate logons to also be notified when a message is sent to a user. For instance, if user MAGNOLIA often logs on under user name MANAGER, an entry may be defined that aliases MAGNOLIA to MANAGER, such that all messages destined to MAGNOLIA are also sent to anyone logged on as MANAGER. As shown below, aliases may be created that notify any of several possible logons MAGNOLIA may be using, on any of several hosts in her DS network.

## Group Distribution Lists

Group distribution lists may be defined, which cause messages sent to the group name to be sent to each logon in the group. For instance, group LUNCHERS may be defined to include several users who normally go to lunch together. To notify each person in the group that you are ready for lunch you might enter:

```
notify lunchers 'Lunch, anyone?'
```

## Hosts in a DS Network

For hosts in a DS network, a DS node name or number may also be given in the alias definitions, such that users on other HP 1000 hosts in your DS network may be notified. The remote hosts must be executing the M1KSS RTE-A monitor program to receive the remote requests, and must be able to execute the NOTIFY utility to deliver the message to users on that host. (Like the NOTIFY utility, the M1KSS monitor program is installed as part of Mail/1000 installation.)

## Alias Definition Format

Each line of the `/system/notify.cf` file is either blank, a comment with a star (\*) in the first column, or an alias definition in the form given below.

```
name: alt aliasname [,aliasname...]
```

Entries in the format above define alternate notification logons that should be notified when a message is sent to the 'name' on this host.

'Aliasname' entries are of the following form:

```
logon[>node]
```

where 'node' is the name or number of the host on your DS network where user 'logon' logs on.



For example, the definition:

```
superman: alt clark, kent>dplanet
```

specifies that when a message is sent to Superman on this host, the message should also go out to anyone logged on as CLARK on this host and to anyone logged on as KENT on DS node DPLANET.

## Using the 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.

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**

<b>Option</b>	<b>Description</b>
+?	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 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/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:#`

---

## **Note**

In the case of serial I/O drivers, setting bit 10 in the CN,34B word is required to ensure that form feeds are recognized. This causes the driver to treat CN,11B,-1 as an unconditional page eject. (See Chapter 4, Serial I/O Drivers, in the RTE-A Driver Reference Manual , part number 92077-90011 for more information.)

---

## **+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. The #PRINT, #PRIN0 command files and the &FFL module all reside in the directory /RTE\_A.

---

**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
```



# System Status Utilities

---

## Display CPU Usage (METER)

The METER program displays information about system and user processes.

---

**Note**      Session accounting must be turned on in the boot command file in order for METER to work.

---

## Sorting and Displaying Process Information

When you run METER, your screen displays a list of the top eighteen processes after sorting, with the following information about each process:

- process name
- LU or session number associated with the process
- logon name of the owner session
- current priority
- current state of the process
- total CPU seconds that have been used
- percent of the CPU that the process used since the last update
- a histogram of the percentage of CPU used

METER recognizes several commands that let you specify how you want the processes sorted and displayed. If you do not specify how you want the processes sorted, they are sorted in decreasing order by the percentage of CPU time used. If two percentages are the same, the current state of the processes are used to determine the order.



## Calling METER

To call METER, enter the following runstring:

```
CI> METER [LU] [<commands>]
```

You may enter a command when the cursor is at the bottom of the list on your screen. (If the cursor is anywhere else, command entry is ignored.) Note that you should avoid typing whenever the screen is being refreshed.

When you enter commands, uppercase or lowercase characters do not make a difference to METER except for the H, L, P, and S commands. These commands have both primary and secondary sort values. The primary sort values are indicated by uppercase characters, and the secondary sort values by lowercase characters.

For example, if you want to display programs first in increasing order of priority and then by the value of their histograms, you should enter these commands (not necessarily in this order):

1. Enter a lowercase “h” to specify the histogram value as the secondary sort value.
2. Enter an uppercase “P” to select priority as the primary sort value.
3. Enter lowercase “i” or uppercase “I” to specify that the sort is in increasing order.

If a process is in the marked session, a “>” appears in the left column of the listing. You may use the > command to change the marked session. The session running METER is the default marked session.

The + and - commands control the frequency with which METER refreshes its display. The default is a refresh every four seconds. You may vary the length of time between refreshes from one to 64 seconds. When you press +, the speed of the refresh is twice as fast; when you press -, the refresh is twice as slow.

METER repeatedly updates information until you stop the program by pressing the space bar, entering uppercase “E” or lowercase “e,” or by entering the BR command at the break mode prompt.

Table 9-1 summarizes the commands.

**Table 9-1. METER Commands Summary**

<b>Commands</b>	<b>Description</b>
<b>Information Command</b>	
<b>?</b>	Displays Help screen
<b>Sorting and Displaying Commands</b>	
<b>&gt;</b>	Set marked session
<b>+</b>	Speed up refresh
<b>-</b>	Slow down refresh
<b>A</b>	Display All processes
<b>D</b>	Sort in Decreasing order
<b>H</b>	Sort by Histogram value
<b>I</b>	Sort in Increasing order
<b>L</b>	Sort by LU (session number)
<b>M</b>	Display only Marked processes
<b>N</b>	Sort by Name
<b>O</b>	Sort by Owner
<b>P</b>	Sort by Priority
<b>S</b>	Sort by State
<b>T</b>	Sort by Total CPU
<b>U</b>	Display only User processes (system utility bit not set). METER determines whether a process is a user or system process by checking the system utility bit, which is set when a program is loaded with the LINK command SU.
<b>Exit Commands</b>	
<b>E</b>	Exit
<b>(Press Spacebar)</b>	Exit

## Example of METER Output

```
METER Use E or space to exit, ? for help. Tue Jul 22, 1986 5:41:13 pm
User programs sorted in decreasing order at a 4 second refresh rate.
Primary sort key is the histogram. Secondary sort key is the state.
```

Name	LU	Owner	Priority	Current State	CPU		Histogram												
					Total	%	1	2	3	4	5	6	7	8	9				
LINK	106	DAVE	90	scheduled	5	59													
>METER	1	SCOTT	39	scheduled	4	2													
CI	106	DAVE	51	prog wait ss	15	0													
>CI	1	SCOTT	51	prog wait ss	15	0													

## Loading METER

To load METER, use the load sequence:

```
CI> LINK,#METER
```

## Show the Status of System Available Memory (SAM)

The SAM program prints out information about the status of System Available Memory. If the system contains Extended System Available Memory (XSAM), SAM reports on it as well.

### Calling SAM

To call SAM, enter the following runstring:

```
CI> SAM,[AL],[XS],[lu|QU]
```

The optional parameter AL specifies a full listing. Using SAM with and without AL is described in detail below.

The XS option returns information about XSAM in the return parameters, as described in the section called “Returned Values” below.

The LU is the LU to list output to (the default is 1). To specify LU 0, use the QU option.

QU specifies quiet mode, and causes output to go to the “bit bucket” (LU 0).

### Running SAM without the AL Parameter

When you run SAM without the AL parameter, the utility lists the total amount of memory, the number of free words, the number of free blocks, the average free block size, the largest free block size, and the percentage of total memory that is free space.

If the utility detects any missing or overlapping blocks, it reports the octal address and decimal length of each problem area along with a summary of the number of words missing and overlapping. This process is done once for SAM and if the system contains XSAM, once for XSAM.

A sample of output when a block of SAM is missing from a system without XSAM follows.

```
SAM Analysis
 12      19  ** Unknown Block **
 71      88  ** Unknown Block **

32615 words free of 32763 total (99%)
1 blocks free; largest is 32615 words; average size 32615 words

** 2 errors: 107 unknown words

XSAM Analysis

4021 words free of 4093 total (98%)
1 blocks free; largest is 4021 words; average size 4021 words
```

## Running SAM with the AL Parameter

When run with AL, SAM prints a complete listing of memory blocks; lists the octal address, decimal length, and usage; reports missing or overlapping blocks, and prints summary information. If the system contains XSAM, this is also done for XSAM. A sample of the output follows.

```
SAM Analysis
Octal  Decimal
Address Length      Type

   3     4   Spooling for LU 1 from LU 8 to LU 2
   7     4   Spooling for LU 1 from LU 6
  13    13   String Passage for SAA
  30     7   String Passage for MUSTR
  37    17   Completed Class I/O for MUSTR on class # 32
  60     8   String Passage for SAM
  70    21   Free Space
 115   144   Pending Class I/O on LU 105
 335  32545  Free Space

32566 words free of 32763 total (99%)
2 blocks free; largest is 32545 words; average size 16283 words

XSAM Analysis
Octal  Decimal
Address Length      Type

   3    20   UDSP for session 158
  27    52   UDSP for session 1
 113    52   UDSP for session 105
 177    48   Signal Control Block for PROG
 257     4   Timer Block for PROG
 263   3917  Free Space

3917 words free of 4093 total (95%)
1 block free; largest is 3917 words; average size 3917 words
```

## Returned Values

The SAM utility returns status information to the calling program through five return parameters, as follows:

- \$RETURN1 = # of errors (zero if none)
- \$RETURN2 = Total number of words of SAM
- \$RETURN3 = Total number of free words
- \$RETURN4 = Total number of free blocks
- \$RETURN5 = Size of the largest free block in words

The XS option causes SAM to return information about XSAM in the five return parameters. If XS is not specified, information about SAM is returned. The parameters have the following meaning:

\$RETURN1 = Number of unknown or overlapping blocks found (0 if none)

\$RETURN2 = Total number of words of memory

\$RETURN3 = Number of free words

\$RETURN4 = Number of free blocks

\$RETURN5 = Size of the largest free block in words

## Loading SAM

To load SAM, use the link sequence:

```
CI> LINK,sam.lod
```

# 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 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 5.10 881121
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 manual) 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> ru,sport,31
Status for LU 31:
Device Driver:      DDC01 Rev. 5.00      Driver type = 05
Interface Driver:  ID800 Rev. 5.00
  Firmware:         Rev. 5.15
CN20: Primary Program:  PROMPT (Enabled)
CN40: Secondary Program:      (Disabled)
CN17: 000000B No user defined terminator
CN22:  32767 Timeout = 327.67 seconds
CN30: 010131B Frame=8/1 None BRG1 9600 baud Port
CN31: 000000B
CN33: 000000B
CN34: 000002B HP Protocol
DV20: 000077B Character mode
DVT Address:  44043B; IFT Address:  53233B
```

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

```
CI> ru,sport,32
IFT of LU 32 is busy: Static status report
Status for LU 32:
Device Driver:      DDC0? Rev. 255.255    Driver type = 05
Interface Driver:  ID80? Rev. 255.255
  Firmware:         Rev. 255.255
CN20: Primary Program:  PROMPT (Enabled)
CN40: Secondary Program:      (Disabled)
CN17: 000000B No user defined terminator
CN22:  32767 Timeout = 327.67 seconds
CN30: 010132B Frame=8/1 None BRG1 9600 baud Port 2
CN31: 000000B
CN33: 000000B
CN34: 000002B HP Protocol
DV20: 000077B Character mode
Driver status bits set:  TimeOut
DVT Address:  44227B; IFT Address:  53233B
```

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.



The display format will vary slightly from an RTE-A to an RTE-6/VM system. The format below is an example of a display obtained on an RTE-6/VM system:

```
CI.73> sport
Status for LU 73:
Device Driver:      DV801 Rev. 5.10      Driver type = 05
Firmware:          Rev. 5.18
CN20:  Schedule Program:  PRMPT (Enabled)
CN17:  000000B No user defined terminator
CN22:   32000 Timeout = 320.00 seconds
CN30:  010131B Frame=8/1 None BRG1 9600 baud Port 1
CN31:  000000B
CN33:  000000B
CN34:  000002B HP Protocol
DV20:  000000B Character mode
EQT Address:  02665B
```

## 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 not locked, not down, and not busy. Otherwise, a message is sent 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
```

## Loading SPORT

To load SPORT, invoke LINK on the SPORTLOD command file. The only relocatable needed is SPORT.REL and the assumed libraries are \$BIGLB or \$FMP6.



## Multiuser Accounting Utilities

---

### Reset Multiuser Accounting Information (RINFO)

RINFO resets the multiuser accounting information found in the user configuration file for the specified users. It resets the multiuser accounting information to zero for cumulative connect time (words 95-96 in record one and words 11-12 in the USER.NOGROUP record) and cumulative CPU usage (words 97-98 in record one and words 9-10 in the USER.NOGROUP record).

---

**Note** RINFO was replaced by the RE (Reset) command in the GRoup and User Management Program (GRUMP). For situations where groups are not used, RINFO was updated to use the new multiuser data structures and function as it did before.

---

### Calling RINFO

To call RINFO, enter the following runstring:

```
CI> [RU ]RINFO[ UserMask1[ UserMask2[,...]]]
```

The UserMask is the name or file mask for users whose accounting information you want to set to 0. The default is the current user.

## Loading RINFO

To load RINFO, use the following link command:

```
CI> LINK,#RINFO
```

## RINFO Protection

If SECURITY/1000 is turned on, the system manager can assign a required capability level to run RINFO; otherwise, only superusers can run RINFO.

## Returned Values

RINFO returns the following 5 values through a PRTN call:

- |           |   |
|-----------|---|
| Word 1    | Status (0=successful, -1=unsuccessful).   |
| Words 2,3 | Connect time in seconds for last user. Word 2 is the more significant word of the double integer value.               |
| Words 4,5 | CPU usage in tens of milliseconds for the last user. Word 4 is the more significant word of the double integer value. |

## Show Multiuser Accounting Information (SINFO)

SINFO displays the multiuser accounting information found in the user configuration file for the specified user. The information includes the last logoff time, cumulative connect time, and cumulative CPU usage.

---

**Note** SINFO was replaced by the LI (List) command in the GRoup and User Management Program (GRUMP). In situations where groups are not being used, or if you want the unique user information, SINFO retrieves information from record one of the user configuration file.

---

### Calling SINFO

To call SINFO, enter the following runstring:

```
CI> [RU ]SINFO[ UserMask1[ UserMask2[,...]]]
```

The UserMask is the name or file mask for the users whose accounting information you want displayed. The default is the current user.

### Loading SINFO

To load SINFO, use the following link command:

```
CI> LINK,#SINFO
```

### Returned Values

SINFO returns the following 5 values through a PRTN call:

- |           |   |
|-----------|---|
| Word 1    | Status (0 = successful, -1 = unsuccessful).   |
| Words 2,3 | Connect time in seconds for last user. Word 2 is the more significant word of the double integer value.               |
| Words 4,5 | CPU usage in tens of milliseconds for the last user. Word 4 is the more significant word of the double integer value. |

## Terminate a Session (KILLSES)

KILLSES terminates a session immediately. You are logged off, all programs associated with the session are killed, associated spool files are closed and released, and the user ID entry in the User Table for the session is released. The session can be any type: Background, interactive, or programmatic. The system session, session 0, can never be killed.

### Calling KILLSES

To call KILLSES, enter the following runstring:

```
CI> [RU ]KILLSES[ sessionnumber][ OK]
```

The sessionnumber is the number of the session to be killed if specified, or the user to be prompted for the number of the session to be killed, if not specified.

When you select the OK parameter, KILLSES is executed immediately, without prompting you for verification.

### Loading KILLSES

To load KILLSES, use the following link command:

```
CI> LINK.KILLSES.LOD
```

### KILLSES Protection

If SECURITY/1000 is turned on, the system manager can assign the capability level required to run KILLSES; otherwise, only a superuser can run KILLSES.

### Returned Values

KILLSES returns the following values through a PRTN call:

- |           |   |
|-----------|---|
| Word 1    | Status (0 = successful, -1 = unsuccessful).   |
| Word 2    | If word 1 = 0, the number of the session that was killed. If word 1 = -1, this is not used and will be 0. |
| Words 3-5 | 0, not used.  |

## **KILLSES Examples**

**Example 1: Terminate session 160 immediately.**

```
KILLSES 160 OK
```

**Example 2: Terminate session 150 after user verification.**

```
KILLSES 150
```

**Example 3: Prompt the user for session number and verification.**

```
KILLSES
```



## Modify or List Session LU Access Tables (SESLU)

SESLU modifies and lists session LU access tables. It does not affect the user or group configuration file associated with the session user. If a session user is given access to an LU, the corresponding bit is set in the session LU access table. If a session user's access to an LU is removed, the corresponding bit in the session LU access table is cleared. You can never remove access to the terminal LU of a session or to an LU in the UDSPs of a session.

### Calling SESLU

To call SESLU, enter the following runstring:

```
CI> [RU ]SESLU[ +S:session][ [-]LU[:LU]][ [-]LU[:LU]][ [-]LU[:LU]]
```

+S:session is the session number whose LU access table is to be modified or listed. If you do not specify the number, SESLU defaults to the caller's session.

You may enter a positive or negative [-]LU[:LU] parameter. A positive entry indicates the LU number or range of LUs to set in the session LU access table. A negative entry specifies the LU number or range of LUs to clear in the session LU access table. If you do not enter any LU number or range, the session LU access table is listed.

### Loading SESLU

To load SESLU, use the following link command:

```
CI> LINK,SESLU.LOD
```

### SESLU Protection

If SECURITY/1000 is turned on, the system manager can assign the different capability levels required to perform all the SESLU subfunctions. These include listing and modifying your session's and another session's LU access tables.

If the SECURITY/1000 system is not turned on, any user can list a session LU access table with SESLU, but only superusers can modify a session LU access table.

See the discussion of SECURITY/1000 in the *RTE-A System Manager's Manual* for a detailed description of capability levels and subfunctions.

## Returned Values

SESLU returns the following values through a PRTN call:

Word 1          Status (0 = successful, -1 = unsuccessful).

Words 2-5      0, not used.

## SESLU Examples

**Example 1: List the LU access table for session 102.**

```
SESLU +S:102
```

**Example 2: Add LU 1 to the LU access table for session 102.**

```
SESLU +S:102 1
```

**Example 3: Add LUs 1-32 and 34-50 to the caller's LU access table.**

```
SESLU 1:50 -33
```

**Example 4: Remove LUs 44-60 and add LUs 20-30 to the caller's LU access table.**

```
SESLU -44:60 20:30
```

**Example 5: Add LUs 20-90 to the LU access table for session 116.**

```
SESLU +S:116 20:90
```

**Example 6: Add LUs 3 and 5 and remove LUs 2 and 4 from the caller's LU access table.**

```
SESLU -2 3 -4 5
```



## 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

<code>-flags</code>	is a string of characters preceded by a dash (-). Where an argument is required, the next word in the runstring is consumed, delimited by blanks or a comma. The flags are:
<code>C catalog</code>	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”.
<code>L listfile</code>	Divert the text listing to the named file. By default the text is listed to the terminal.
<code>P pagesize</code>	Set the number of lines per page for “More...” prompting on the terminal. The default size is 22 lines.
<code>B</code>	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 ’’s. 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 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 about the text input format expected.



## File Manager (FMGR)

---

FMGR is a program that manipulates files on FMGR directories. You can use FMGR to perform operations on FMGR cartridges that you cannot perform with CI, such as initializing directories.

FMGR used to be the standard command interpreter just as CI is today. FMGR can run programs and issue control requests in much the same way that CI can, except that FMGR requires commas as parameter separators.

FMGR works only with FMGR files, so it cannot access hierarchical file systems. This appendix discusses only those commands that are useful for manipulating FMGR files. Use CI for other operations.

This appendix covers the following topics:

- **FMGR Control:** Commands that let you perform FMGR operations.
- **Disk Manipulation:** Commands to handle FMGR cartridges.
- **File Manipulation:** Commands to copy, purge, and list FMGR files.
- **Transfer Files:** Commands to run FMGR non-interactively.
- **Device Manipulation:** Commands to down a device and change buffer limits.

A table showing other FMGR commands and their CI equivalents is provided at the end of this appendix, as well as a description of the COMND program.

# FMGR Control

Using FMGR commands, you can designate a list device, send output to it, define a log device for receiving error messages, limit the system messages, and display explanations of error messages. FMGR control definitions are provided below, and the commands are described in subsequent sections.

## List Device

The system list device is the device to which output is directed by default. The list device is often a line printer, but you can use the LL command (described later) to change the designation, directing the output to any device or disk file specified in the command. You may change the assignment of the list device as often as necessary.

The LI (List) command, described later, is used to send output to the system list device.

## Log Device

The system log device receives FMGR error messages. It must support input as well as output, since the error messages may require a reply from the operator. The LO command, described later, lets you reassign the FMGR log device.

## Severity Code

The severity code acts as a filter to screen out unwanted FMGR output (messages from the system). The SV command, described later, is often used in transfer files.

## FMGR Errors

FMGR errors can be either positive or negative. They can be automatically listed to the terminal by specifying a severity code of 1000 or 1001 (see the SV command discussion).

## FMGR Control Commands

Table A-1 provides a summary of the FMGR commands, which are described in the sections that follow the table, along with command examples.

**Table A-1. FMGR Control Commands**

Commands			Description
<b>Information Commands</b>			
<b>??</b>	Explain Error Codes	error#	Provides the meaning of an FMGR error code number
<b>Listing Commands</b>			
<b>LL</b>	Display/Change List Device	namr	Displays or changes list device assignment
<b>LO</b>	Log Device	namr	Assigns a log device to receive FMGR error messages
<b>Configuration Commands</b>			
<b>SV</b>	Display/Set Severity Code	severity	Displays or changes the current FMGR severity code

### Explain Error Codes (??)

**Purpose:** Provides the meaning of an FMGR error code number.

**Syntax:** ?? [,error]  
 ?? [,ALL]

**error** The FMGR error number. If you do not enter the parameter, an explanation is provided for the most recent FMGR error.

**ALL** The error codes and explanations are all listed to the list device.

### ?? Command Examples

The following examples illustrate the command entry and the system response.

**Example 1: Explain the meaning of error code -006.**

```
FMGR : ??,-6
FMGR -006 FILE NOT FOUND
```

**Example 2: Explain the meaning of error code -102.**

```
FMGR : ??,-102
FMGR -102 ILLEGAL D.RTR CALL SEQUENCE.
```

**Example 3: List all the FMGR errors.**

```
FMGR : ??,ALL
```

The system responds to this entry by sending a complete listing of all the FMGR error codes and their meanings to the list device.

## Display or Change List Device (LL)

Purpose: Displays or changes the list device assignment.

Syntax: LL[ ,namr ]

namr                    The new list device. This may be any existing file or LU number.

Description:

Namr should be a device that allows output; if you try to list on an input only device, FMGR terminates and issues the system error code IO07.

Certain FMGR commands (LI,CL,DL,PL,IO,??) direct their output to the list device. The default list LU is your terminal.

## LL Command Examples

**Example 1: Change the list device from logical unit 6 (line printer) to logical unit 4 (user terminal).**

```
LL,4
```

**Example 2: Change the LU back to LU 6.**

```
LL,6
```

**Example 3: Change the list device to list output to existing file LISTF. Specify CRN to ensure that list is sent to the correct file.**

```
LL,LISTF::13
```

## Change Log Device (LO)

**Purpose:** Designates a log device to receive FMGR error messages.

**Syntax:** LO[ ,namr ]

**namr** The device to receive the FMGR messages. This may be any existing file or LU number.

**Description:**

If you do not enter a namr, the LU number of the current log device is displayed.

Namr should be a device that allows input as well as output, as you may need to enter a reply to one or more messages.

## Display or Set Severity Code (SV)

**Purpose:** Displays or changes the severity code currently being used by FMGR.

**Syntax:** SV[ ,severity[ ,global#][ ,IH ]]

**severity** The new severity code, as follows:

- 0 Display error codes and echo commands on log device (default).
- 1 Display error codes on log device, inhibit command echo.
- 2 Display error code displayed only if error requires transfer of control to log device for correction; no command echo.
- 3 Same as 2.
- 4 If a FMGR command error is encountered, command file continues automatically; no command echo, no transfer to log device, and no command file abort occurs.

100x If the above severity codes are added to 100, additional error message information is provided when an error occurs.

**global #** The optional G-type global number from 1-9 in which the severity code is placed.

**IH** The optional parameter that inhibits echo of command entry.



### Description:

If you omit the severity parameter, the current severity code is displayed.

The normal default mode is to echo each command as it is entered on the input device and to log all errors on the same device. During interactive operation, the severity code should be this default value, zero.

When there is no advantage to echoing commands at the console (for example, when commands are entered from a peripheral device or through a file), the severity code can be set to 1.

You can suppress messages that do not require action by setting the code to 2, which terminates any currently executing job when an error occurs.

Whenever you suppress command echo, any command that causes an error is displayed before the error code unless the error display is also suppressed.

### SV Command Examples

**Example 1: Display error codes and echo commands on the log device.**

```
SV,0
```

**Example 2: Display error codes and echo command on the log device, and display additional information about errors encountered.**

```
SV,1000
```

## Disk Manipulation

There are four important points to remember about disk manipulation:

- Disk logical units are logical subdivisions of physical disk drives.
- Cartridges are logical subdivisions of physical disk packs or flexible (floppy) disks.
- The information stored on a disk pack/cartridge is available to the system only when:
  - the disk pack is physically mounted on the disk drive, and
  - the cartridge is logically mounted on the logical unit.
- When a cartridge is mounted on a logical unit, there is an exact correspondence between the cartridge and the logical unit. At other times, the cartridge and the logical unit are independent entities.

The discussion of disk manipulation in this appendix consists of definitions of the various disk manipulation elements, followed by the FMGR commands used in manipulating disks.

### Logical vs Physical

When you are managing the information stored on disk files, consider the difference between “logical” and “physical” entities. In general, logical refers to software, and physical refers to hardware. For example:

- Disk packs and disk drives are hardware, thus physical. Disk files and their structure are software, thus logical (they exist only in the memory of the computer).
- When a disk pack is placed on a disk drive, it is being physically mounted. When the operating system (software) is informed that the storage space on the disk pack is available to the system, the disk pack has been logically mounted.

The disk manipulation commands discussed later in this appendix deal with logical operations only.

### Disk Logical Units and LU Numbers

The operating system divides the storage capability of a disk drive into one or more logical sections, and assigns a logical unit (LU) number to each section. Each logical unit is treated as a separate disk by the system, even though it may share the physical disk drive with several other LUs. Disk logical units can be of different sizes. There must be at least one disk LU for every disk drive on the system.

## Cartridges and Cartridge Reference Numbers

The storage capacity on a disk pack (fixed or removable) or a flexible (floppy) disk can be divided into one or more cartridges. Cartridges are identified by cartridge reference numbers (CRNs), and they are closely related to disk LUs. A CRN is assigned when the cartridge is initialized (initialization is discussed in subsequent sections).

Data stored in disk files can be accessed by referring either to the LU number or to the CRN. The relationship between the disk LU and the cartridge (both of which are logical entities) is the same as that between the disk drive and the disk pack (physical entities). For example:

Suppose you have two flexible disk drives that are assigned the logical unit numbers LU 13 and LU 14. You also have a flexible disk with a cartridge assigned the cartridge reference number CRN 123.

If cartridge CRN 123 is mounted in disk LU 13, you can read a file from the flexible disk by referring to either LU 13 or CRN 123.

If you move the flexible disk to the other drive and want to address it using the LU number, you now refer to it as LU 14. However, the CRN is still 123.

In other words, the cartridge reference number stays with the data on the flexible disk, while the logical unit number stays with the disk drive. This means that when you address files by the CRN, the LU being used should have the same configuration (see the section below on “Configuration of Logical Units/Cartridges”).

CRNs are positive numbers from 1 to 32767. A CRN may also be assigned as two ASCII characters; in this case, the cartridge may be referred to by the numeric equivalent of the ASCII characters, and that number appears as the CRN when you list the cartridge information with the CL or DL commands, described later.

Since LU numbers fall within the range of allowable CRNs, the system needs some way of telling which type of number it is dealing with. In situations that may be unclear, LU numbers are entered as negative numbers and CRNs are entered as positive numbers in the namr parameter of the file descriptor, which is discussed below.

## Configuration of Logical Units/Cartridges

Configuration refers to the way disk logical units or cartridges are arranged to cover the storage capacity of a disk drive, disk pack, or flexible disk.

The configuration of logical units assigned to a disk drive depends upon the type of disk drive involved. For example, since flexible disks hold a relatively small amount of information—about one megabyte on a two-sided disk—a single flexible disk LU is usually configured to cover the entire volume.

Flexible disks are portable, and a standard configuration lets you use a flexible disk in any flexible disk drive on the system. You can also use a flexible disk to transport programs and data between two systems. Because of their portability, flexible disks are frequently mounted and dismounted.

Disk packs (“hard” disks) are much larger than flexible disks both in physical size and storage capacity. Because of their large physical size, disk packs are not as easily removed, transported, and stored as flexible disks. In addition, some disk packs are not removable.

Because of their large storage capacity, in the range of 10 to 50 megabytes, hard disks are almost always divided up into several disk LUs. These LUs seldom have any standard configuration. As a result, hard disks typically stay in one location and are not usually interchanged between drives, either on the same system or on different systems.

Disk LUs are configured during system generation. Cartridges are configured during initialization (initialization is discussed in subsequent sections). Note that a disk pack or a flexible disk can be successfully used on a disk drive only if the configuration of disk LUs on the disk drive is the same as the configuration of cartridges on the disk pack or flexible disk.

## Mounting and Dismounting Cartridges

Mounting and dismounting cartridges are logical operations. Before you use the MC (Mount Cartridge) command (described later) to mount a cartridge logically, you must physically mount the disk pack or flexible disk that contains the cartridge on the disk drive. This makes the cartridge available to the system.

When you invoke MC, the system responds by establishing an exact correspondence between the cartridge and the LU on which it is mounted. That correspondence persists until you use the DC (Dismount Cartridge) command (described later) to logically dismount the cartridge; at that time, the LU and the cartridge regain their separate identities.

Note that you may not mount two cartridges with the same CRN at the same time.

The following example illustrates mounting and dismounting cartridges:

You have a flexible disk with its entire storage volume configured as one cartridge, with a CRN of 1234. You also have a flexible disk drive with its full storage capability configured as LU 17. You physically mount the flexible disk by inserting it into the drive; then you logically mount the disk by invoking the following command:

```
FMGR : MC, -17
```

This tells the system that the cartridge mounted in LU 17 is now available for use. The system searches the file directory of the cartridge to find the CRN, and then it establishes an equivalence between LU 17 and cartridge 1234. Directing a command to LU 17 is now the same as directing the command to cartridge 1234. This relationship continues until the cartridge is dismounted, using the DC command:

```
FMGR : DC, -17
```

or

```
FMGR : DC, 1234
```

The system severs the relationship between LU 17 and cartridge 1234 and once again considers them separate entities. You may now physically dismount the flexible disk.

## Cartridge Directory

The term “cartridge directory” refers to the directory of cartridges that are mounted on the system. The term “file directory” refers to the directory of files on a cartridge. Cartridge directories are described in this section, and file directories are described in the next section, “Cartridge File Directories.”

The cartridge directory is a system table that contains entries for all mounted cartridges and maintains the relationship between mounted cartridges and logical units.

Cartridges are listed in the system table in the order in which they were mounted. The last cartridge mounted is placed at the bottom of the cartridge directory. This is significant if the system is searching for a file but does not know in which cartridge it is located. In this case, the search begins with the first cartridge in the list and continues until the file is found or until all the mounted cartridges are searched.

The order of the cartridge directory is also used if a new file is created but no cartridge was specified to hold it. In this case, the file is placed on the first cartridge in the directory that has enough space to accommodate the new file.

To change the order of a search, you may use the DC and MC commands to dismount and then remount a cartridge. When you dismount the cartridge, it is removed from the directory; when you remount it, the cartridge is placed at the bottom of the directory since it is the newest entry.

Note that you may not dismount a cartridge that contains open files or program files that were restored. If you try to dismount such a disk, an FMGR error message is displayed. However, the cartridge is still moved to the bottom of the cartridge directory.

## Cartridge File Directory

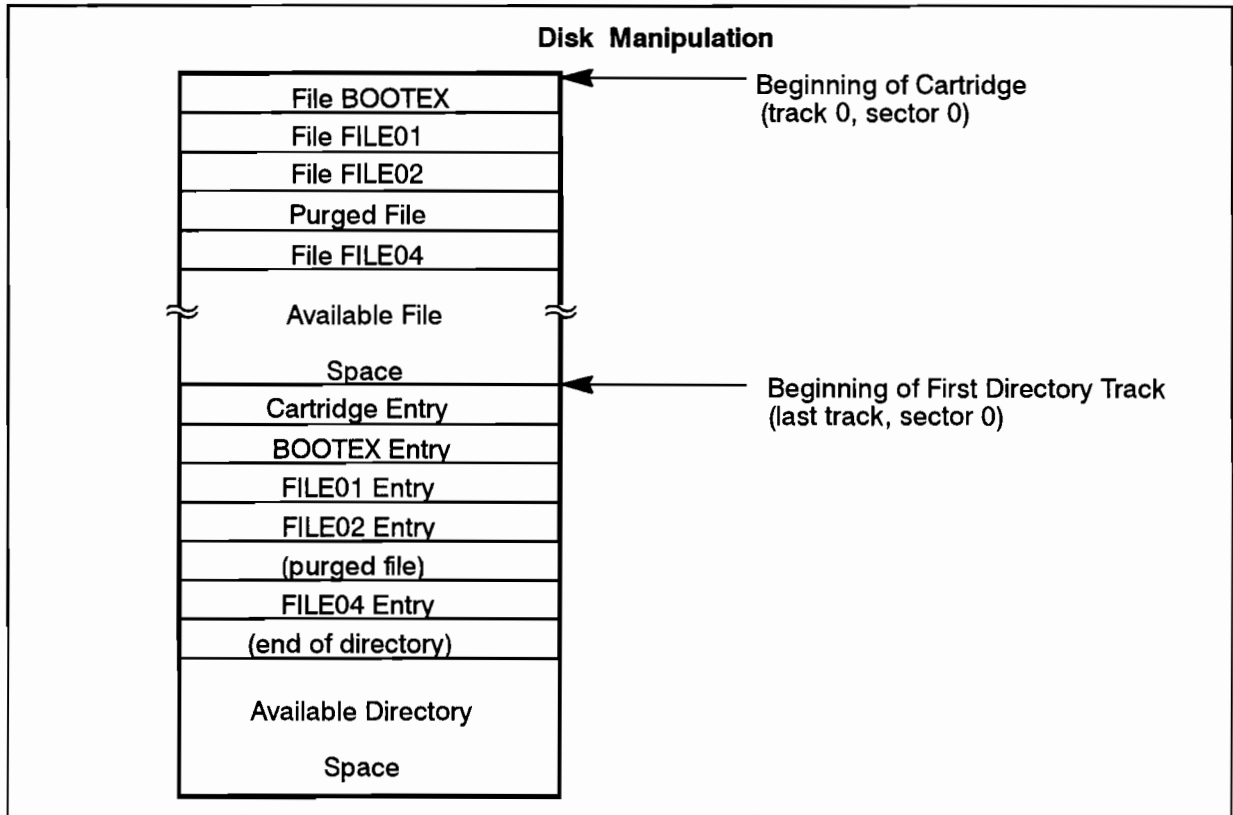
The term “file directory” refers to the directory of files on a cartridge. File directories are described in this section. Cartridge directories are described in the previous section, “Cartridge Directory.”

Files are located at the beginning of cartridges and directory entries at the end of them. Files begin at sector 0 of the logical first track of the cartridge. The first track is set using the IN (Initialize) command (described later), usually at track 0. Figure A-1 illustrates the cartridge structure.

The system always installs the boot extension file, BOOTEX, as the first file on a cartridge. It is a type 1 file, 768 blocks long.

The directory begins at sector 0 of the last track of the cartridge. If there is more than one directory track, the second directory track begins at sector 0 of the next-to-last track, and so on, moving toward the beginning of the cartridge. The first entry in the directory describes the cartridge, and succeeding entries describe files in their order on the cartridge.

When a new file is added to the cartridge, its size is checked. If it is the same size as a file that was purged, it replaces the purged file and its directory entry. If no files of the same size were purged, the new file is added after the last file, and a new entry is created at the end of the directory.



**Figure A-1. Cartridge Structure**

## Cartridge Initialization

Initializing a cartridge builds or modifies the cartridge header information for a mounted disk cartridge. A cartridge must be initialized (using the IN command, described later) before it can be recognized by the file management system.

If you mount an uninitialized cartridge, the system locks the cartridge and it cannot be used with the File Management Package (including FMGR) until you initialize it.

Every cartridge must be initialized at least once, and may be re-initialized as often as necessary.

At initialization, the cartridge takes on the configuration of the LU on which it is mounted. Several of the parameters that describe the cartridge are specified, including the cartridge reference number, the cartridge label, the number of directory tracks, the location of bad tracks, and others. The parameters are stored in the cartridge entry of the file directory on the cartridge.

## Master Security Code

The master security code (msc) controls access to security codes for all the files managed by the file management system. If you specify a master security code at system generation, you must subsequently specify it for new cartridge initialization and then for all re-initializations of the same cartridge. Be aware that this code is never printed or displayed by the system, so you must be particularly careful to remember it. The IN command can be used to change the master security code for the system.

Note that when the system is re-booted, the master security code reverts to the value set during system generation.

## Re-initializing a Cartridge

You can re-initialize a cartridge in order to change the parameters in the cartridge entry, as often as desired.

Since FMGR locks a cartridge during initialization, make sure that all the files on the cartridge are closed and that all temporary (disk-resident) programs have released their ID segments before re-initialization. Otherwise, the cartridge cannot be locked, and the IN command is not executed.

During re-initialization, be careful about changing the first track parameter. If the first track number is increased, the system purges all files on the cartridge. (For more details, see “Purging All Files,” below.) If the first track number is decreased, more track space is added to the cartridge, but the system is unable to use that space until the cartridge is packed with the PK command (described below).

If the starting track is decreased to zero, the system purges all files on the cartridge and installs the BOOTEX file at the beginning of the starting track. The starting track number is not changed unless you want to purge all files, as it is a waste of disk space to assign a starting track other than track zero.

Be careful when you change the number of directory tracks during re-initialization. More space is gained for file entries in the directory by increasing the number of directory tracks, as long as the newly added directory tracks don't use track space that is already assigned to existing files.

If the new directory tracks conflict with existing files, or if you decrease the number of directory tracks, the system purges all of the files on the cartridge. The system does give you another chance to say NO before the files are purged. See the next section for more detailed information about purging files.

## Purging Files on a Cartridge During Re-initialization

A cartridge is purged of all files if you re-initialize the cartridge and any of the following changes occur:

- The first track number is increased.
- The first track is decreased to zero.
- The number of directory tracks is decreased.
- The number of directory tracks is increased and there is not enough room left on the cartridge for the new track(s).

In interactive mode, the file manager issues an FMGR 060 message in response to any of these occurrences. If this message displays, you must respond by entering YES or NO; otherwise, the message simply displays again.

If you answer NO, the IN command aborts. If you answer YES, or if the command is in a transfer file, the files are purged as part of the re-initialization. After the files are purged, the BOOTEX file is created at the beginning of the first track.

The following two command sequences purge all files from a cartridge. Assume that the cartridge was last initialized with the command given in the previous example.

```
FMGR : IN, SC, JS, JS, SORCE3, 1, 2  
FMGR 060 DO YOU REALLY WANT TO PURGE DISK? (YES OR NO). YES
```

```
FMGR : IN, SC, JS, JS, SORCE3, 0, 2
```

The first IN command increases the first track number, causing the system to try to purge all files on the cartridge. Before the files are purged, the system checks to see if you really want to purge the files by issuing the FMGR 060 message. If you respond by entering YES, the system proceeds to purge all the files.

The second IN command makes logical track 0 the first track of the cartridge. BOOTEX is then installed on that track.

```
FMGR : IN, SC, JS, JS, SORCE4  
FMGR 060 DO YOU REALLY WANT TO PURGE DISK? (YES OR NO). YES
```

```
FMGR : IN, SC, JS, JS, SORCE4, , 2
```

In this command sequence, all the files on the cartridge are purged when the number of directory of tracks is decreased. The first IN command uses default values of 0 for the first track and 1 for the number of directory tracks. If you respond YES to the FMGR 060 message, the system purges all the files, reinstalls BOOTEX on track 0, and re-establishes the directory on a single track.

The second IN command increases the number of directory tracks to 2, the original number.



## Packing a File Cartridge

When files are purged from a cartridge, they leave gaps in the disk space. The PK command lets you “pack” a file cartridge, which eliminates such gaps and allows more efficient use of the disk space.

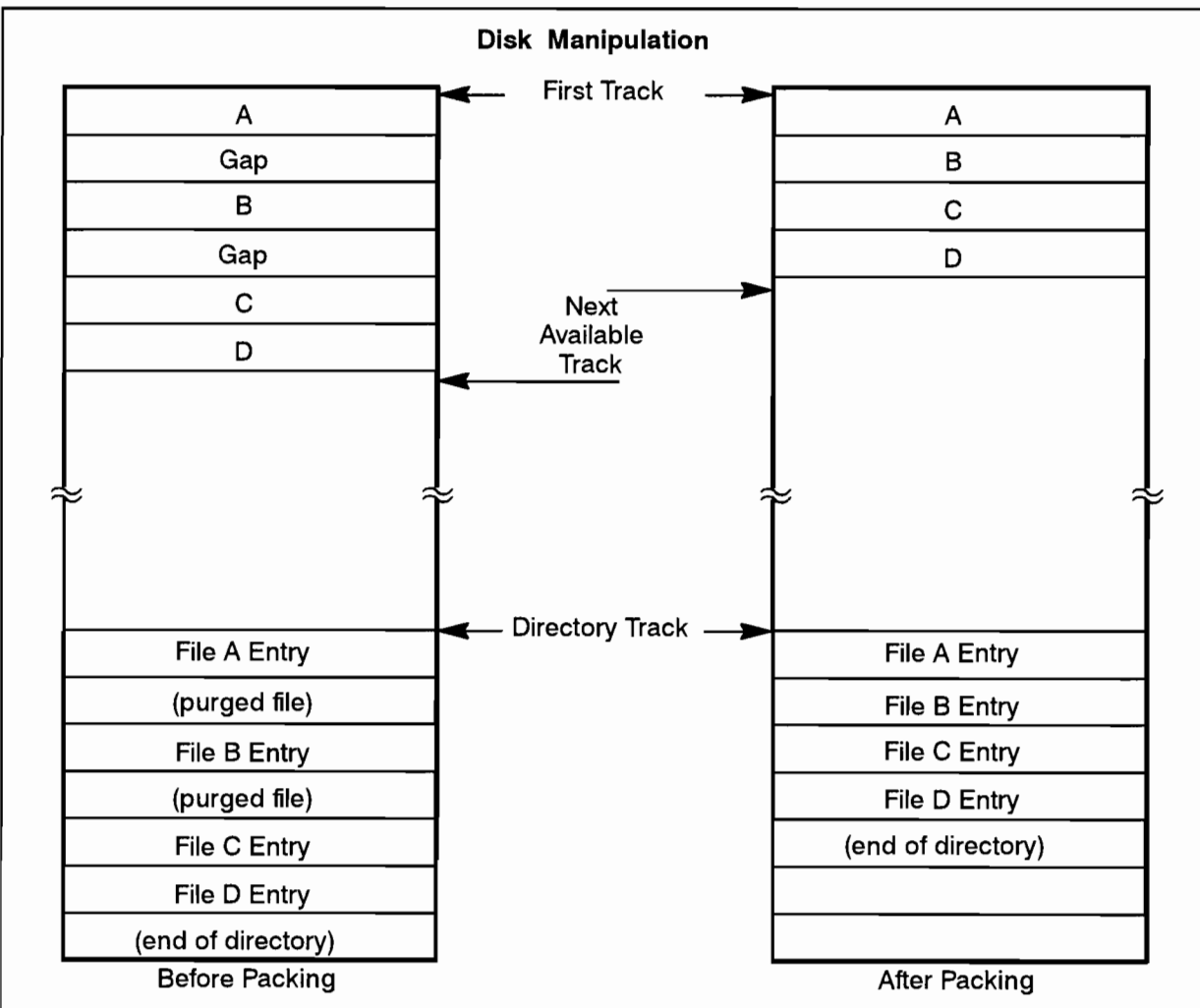
When PK executes, it moves files into the empty spaces left by purged files wherever possible. After a file is packed, its directory entry is updated. After all the files are packed, the directory is packed and the entries for purged files are removed.

Note that the PK command purges all scratch files (scratch files are discussed later in this appendix). If you want to save the information in a scratch file, either rename the file or store the information in another file.

The PK command also purges files that contain tracks reported as bad by the IN command. If you want to save such a file, use the ST command to store it on another cartridge.

For more details, refer to the discussion of the PK command later in this appendix.

Figure A-2 shows the structure of a disk before and after the PK command is executed.



**Figure A-2. Disk Structure Before and After Packing**

## Transferring Files Between Disk Cartridges

All or selected files residing on a mounted disk cartridge can be copied or moved to another mounted disk cartridge with the CO command. Using the CO command, you can perform the following operations:

- Copy or move the contents of an entire disk cartridge to another disk cartridge.
- Copy or move selected files from one disk cartridge to another disk cartridge.
- Back up one disk cartridge to another (or several) disk cartridge(s).
- Merge many cartridges onto one cartridge (restoring from a backup).

The CO command and its options are described in subsequent sections.

## Disk Manipulation Commands

The sections that follow describe the commands you can use for manipulating disks, including listing the cartridge directory, listing the cartridge file directory, mounting and dismounting cartridges, initializing cartridges, packing a file cartridge, and transferring files between mounted cartridges. Table A-2 summarizes the commands.

**Table A-2. Disk Manipulation Commands Summary**

<b>Commands</b>	<b>Description</b>
<b>Listing Commands</b>	
<b>CL</b> List Cartridge Directory	Lists all mounted cartridges in the cartridge directory
<b>DL</b> List File Directory	Lists some or all the entries in any or all of the file directories on a cartridge
<b>Configuration Commands</b>	
<b>DC</b> Dismount File Cartridge	Dismounts a file cartridge logically, removing the entry for the cartridge from the cartridge directory
<b>IN</b> Initialize File Cartridge	Builds or modifies the cartridge header information for a mounted disk cartridge, and used to assign or change the master security code
<b>MC</b> Mount File Cartridge	Mounts a file cartridge logically and creates an entry for the cartridge in the cartridge directory
<b>PK</b> Pack File Cartridge	Moves files together on a cartridge to eliminate gaps left from purged files
<b>File Transfer Command</b>	
<b>CO</b> Transfer Files Between Cartridges	Copies or moves selected files between mounted cartridges

### **List Cartridge Directory (CL)**

**Purpose:** Lists all cartridges in the cartridge directory. Only mounted cartridges appear in the directory.

**Syntax:** CL

**Description:**

When you use the CL command, the cartridge directory list is sent to the designated list device. You may change the list device with the LL command, described earlier in this chapter.

Cartridges are listed in the directory in the same order that is used for all commands that access the dis (such as CL, DL, ST, DU, PU, RN, LI, RU, and XQ commands).

## CL Command Example

The following example illustrates the use of the CL command:



```
FMGR : CL
      LU          LAST TRACK      CR          LOCK
      12          0405             00012
      13          0199             00013
      14          0199             00014
      17          0199             00017
      18          0199             19523
```

LU	The logical unit number of the cartridge.
LAST TRACK	The highest track available on the cartridge.
CR	The cartridge reference number.
LOCK	The name of the program locking the cartridge; blank if not locked.

## Copy Files (CO)

Purpose: To copy files from one disk cartridge to another.

Syntax: `CO,cartridge1,cartridge2[,options[,name1[,name2[,msc]]]]`

or

`CO,namr1,cartridge2[,options[,name1[,name2[,msc]]]]`

cartridge1      The source cartridge identifier (negative LU number or positive CRN).

namr1            The file name and optionally, security code, cartridge identifier (negative LU number or positive CRN), file type, and file size. The namr may contain wildcards. See the discussion of the namr parameter in the section "File Manipulation".

cartridge2      The destination cartridge identifier (negative LU number or positive CRN).

options          The CO command options are described below.

name1	The starting file name (first file to be copied to the destination cartridge). If specified, it must reside on the source cartridge or files are not copied.
name2	The ending file name (last file to be copied to the destination cartridge). If specified, it should reside beyond name1 in the source cartridge file directory. See the comments in the examples below.
msc	The master security code; must be two ASCII characters. The msc is used in conjunction with the Purge and/or Clear options, described below.

## CO Command Options

The five CO command options provide additional control over the transfer of files.

You may enter single or multiple options, in any order, in the third parameter of the CO command. Options are entered without delimiters.

Note that neither the C option nor the P option removes the file BOOTEX from the source or destination cartridge. The D and P options do not purge any type 6 or active file.

Table A-3 summarizes the options, which are discussed below.

**Table A-3. CO Command Options Summary**

Option	Description
C	Clear destination cartridge before copying files
D	Dump mode (store file even if it exists on destination cartridge)
E	Eliminate extents on copied files
P	Purge source files after copy
V	Verify that files are copied correctly

### Clear (C)

This option clears the destination cartridge before any files are copied to it (this is similar to using the IN command, discussed later, except that any existing BOOTEX file is not removed).

If you specify this option from an interactive terminal, an FMGR 060 message requiring a YES or NO response is displayed. (If you enter anything else, the CO command displays the message again.) If you reply NO, the CO command aborts. If you answer YES, the destination cartridge is cleared of all files (except BOOTEX). You must correctly specify the msc parameter to use this option; a missing or incorrect msc causes an FMGR -51 error.

## **Dump (D)**

When you specify the Dump option, the normal copy process still takes place. When the program encounters a duplicate file name on the destination cartridge, it tries to purge that file and replace it with the file on the source cartridge. Purging and replacing the file only occurs if the security code on both files match; otherwise, an FMGR–007 error occurs, leaving the destination file unchanged and the source file uncopied.

## **Eliminate Extents (E)**

The E option eliminates extents from the resulting destination files. A destination file's size is calculated from the source file's size and the number of extents used.

## **Purge (P)**

This option purges selected files from the source cartridge as they are copied to the destination cartridge. This results in “moving” files from the source to the destination cartridge. To purge a file, the security code of the file must match the security code in the subparameter of namr1. Alternatively, you may specify the msc parameter to have files purged regardless of their security codes. The purge option does not affect whether a file is copied or not.

## **Verify (V)**

The Verify option compares the source and destination files after the copy to make sure the transfer was done correctly. If the files differ, CO terminates with an FMGR–049 error.

## **CO Command Option Examples**

**Example 1:** Copy files in dump mode, and verify their transfer.

```
CO,DV
```

**Example 2:** Clear the destination cartridge first, and purge the source files after copying.

```
CO,CP
```

**Example 3:** Clear the destination cartridge first, copy files in dump mode, eliminate any extents, verify their transfer, and purge the files after copying.

```
CO,CDEVP
```

## CO Command Examples

The following four examples of CO command formats show how to select files to copy to the destination cartridge.

**Example 1:** Specify cartridge1. With this form of the command, all the files on the source cartridge are selected (no mask is specified). The starting and ending files are selected by the name1 and name2 parameters (see Example 4).

```
CO,10,20,,&FIRST,&LAST
```

**Example 2:** Specify namr1 without wildcard characters. With this form of the command, only the file specified on the source cartridge is selected. The name1 and name2 parameters are not used with this command format.

```
CO,&PROG::10,20
```

**Example 3:** Specify namr1 with wildcard characters (mask specified). With this form of the command, those files matching namr1 (see the discussion of the namr parameter in the "File Manipulation" section) on the source cartridge are selected. The starting and ending files are selected by the name1 and name2 parameters (see Example 4).

```
CO,&-----:10,20
```

**Example 4:** Specify name1 and/or name2 in conjunction with one of the above forms of the CO command. These parameters restrict the search of the source cartridge. Only those files that reside between name1 and name2 (inclusive) on the source cartridge can be copied. The beginning of the cartridge is assumed if name1 is not specified. The end of the cartridge is assumed if name2 is not specified, or if it doesn't reside on the source cartridge.

```
CO,&-----:10,20,,&FIRST,&LAST
```

If you specify two ASCII characters for the source CRN, the system interprets them as a namr (see Example 2 above). To specify an entire cartridge, you must use the LU number of the cartridge, or use the -----:CRN format (shown in Example 3 above) of namr1.

In the following three examples, files &A, %A, &B, %B, &C, %C, &D, %D reside on cartridge 10; BOOTEX, %C, &C reside on cartridge 20; and cartridge 30 is empty.

**Example 5: Copy all files on cartridge 10 to cartridge 20.**

```
FMGR : CO,10,20
&A (Cartridge1 is 10; cartridge2 is 20)
% A (The file names are printed out as they are copied)
&B
% B
&C
FMGR-002 (Message indicates that these files were not copied)
% C
FMGR-002
&D
% D
FMGR : (Ready for next command)
```

**Example 6: Copy all files on cartridge 10 starting with the file &B and stopping with file &D to cartridge 30. Verify the copies.**

```
FMGR : CO,10,30,V,&B,&D
&B (Cartridge1 is 10; cartridge2 is 30; option is verify; first file is
% B &B; last file is %D.)
&C
% C
&D
% D
FMGR : (Ready for next command)
```

**Example 7: Move all program source files (&) to cartridge 20 and all relocatable files (%) to cartridge 30. Replace files with the same name on the destination cartridge (must supply the master security code if selected files have non-zero security codes).**

```
FMGR : CO,&----::10,20,PDV,,,SC
&A (Namr1 selects all files beginning with "&" on cartridge 10;
&B cartridge2 is 20; options are purge, dump and verify (note
&C that &C was replaced); master security code is SC.)
&D
FMGR :

FMGR : CO,%----::10,20,PDV,,,SC
% A (Namr1 selects all files beginning with "%" on cartridge 10;
% B cartridge2 is 20; options are purge, dump and verify. Ready
% C for next command.)
% D
FMGR :
```



**Example 8: Back up cartridge 40 onto a set of floppy disks (cartridges 10, 20, 30).**

```
FMGR : CO,40.10.CV
FILEA                               (Cartridge1 is 40; cartridge2 is 10; options are clear
FILEB                               destination and verify file integrity.)
.
.
.
FILEK
FMGR-033                             (Disk is full)
FMGR : DP,10G                       (Display next file name)
FILEK
FMGR : CO,40.20.CV,10G             (Repeat above for next floppy)
FILEK
.
.
.
FILEQ
FMGR-033(Disk is full)
FMGR : CO,40.30.CV,10G           (Repeat above for last floppy)
FILEQ
.
.
.
FILEZ
FMGR :                               (Ready for next command)
```

## CO Command Termination

The CO command terminates in the following circumstances:

- The selected files were correctly copied to the destination cartridge. This is a normal termination. The 1P global parameter is set to 0 to indicate a successful termination.
- A file copy was not verified correctly. If you specified the Verify option and a file did not copy correctly, the CO command terminates. An FMGR-049 error is generated, and the file name is placed in the 10G parameter.
- The destination cartridge runs out of file space or directory space. If a selected file cannot be copied to the destination cartridge for lack of space, the CO command terminates with an FMGR-014 or FMGR-033 error. The file name is placed in the 10G global parameter.

If an FMGR-049 error occurs, you may restart the CO command to the same destination cartridge by specifying the 10G global parameter for the starting file name. To display the file name, enter DP,10G from FMGR (shown in Example 8, above).

## Dismount File Cartridge (DC)

**Purpose:** Dismounts a file cartridge logically, removing the entry for the cartridge from the cartridge directory.

**Syntax:** DC, cartridge

cartridge      The cartridge identifier (negative LU number or positive CRN; a CRN may be an integer or two ASCII characters).

### Description:

Dismounting a cartridge makes the cartridge unavailable to the system. It should be done for all file cartridges (logical units) on a physical disk—"hard" disk or flexible ("floppy") disk—before the disk is physically removed from the disk drive.

When you dismount a cartridge, you can reference it by using its LU number, a CRN that is an integer, or a CRN that uses the two-character ASCII equivalent. For example:

FMGR : DC, -10      (Dismounts a cartridge using its LU number)

DC, 21587      (Dismounts a cartridge using an integer CRN)

DC, TS      (Dismounts the same cartridge using the two-character ASCII equivalent)

When you dismount a cartridge, it is locked and its entry is removed from the system. If any program files (type 6) on the cartridge are currently occupying ID segments, the system cannot lock the cartridge and issues an FMGR 038 error (see "DC Error Handling," below).

Dismounting a cartridge can be used to change the order of a cartridge directory. That technique and its usefulness are discussed in the earlier section, "Cartridge Directory."

If FMGR discovers that a cartridge directory is corrupt (the directory is not internally consistent), the cartridge is dismounted as long as it contains no program (type 6) files or swap files. If there are program or swap files on the cartridge, the cartridge remains mounted, it is locked to the copy of FMGR that discovers the corruption, and an FMGR -103 error ("disk directory corrupt") is issued.

## DC Command Error Handling

### FMGR 038 ATTEMPT TO REMOVE ACTIVE TYPE 6 OR SWAP FILE

There are probably some program files (type 6) on your file cartridge that currently occupy ID segments. FMGR lists the offending programs for you. You can release the ID segments by entering the OF, program, DR command for each program, then DC again.

### FMGR -103 DISK DIRECTORY CORRUPT

The file management system found that the information in the directory was not internally consistent. An FMGR -103 error was issued and the cartridge was locked to the copy of FMGR that found the error (that is, your copy). Even though the cartridge is locked, you can still resolve the problem by copying the files to a new cartridge with the CO command.

## FMGR –008 LOCK REJECTED

The cartridge is in use by another program. Reissue the command when it is free.

### List File Directory (DL)

**Purpose:** Lists some or all the entries in any or all of the file directories on a cartridge.

**Syntax:** DL,,[master security]

or

DL,namr[,master security]

or

DL,cartridge[,master security]

**namr** The file name and, optionally, security code and cartridge identifier (negative LU or positive CRN.) Any subparameter may also be specified. You may include wildcards in the namr parameter.

**cartridge** The cartridge identifier (a negative LU number or positive CRN. If you specify a CRN, it must be an integer, not ASCII characters. If you omit the parameter or enter zero, the directories of all the mounted cartridges are listed.

**master security code** The code assigned to the system at system generation. If this is specified correctly, the listing is in the long form; otherwise, the short form is used.

### Description:

Each cartridge has on its last tracks a directory that describes the cartridge and all the files in the cartridge. The DL command can be used to list some or all of the entries in any or all of the file directories. Output goes to the list device.

You may enter

FMGR : DL

to list the directories for all mounted file cartridges; or

FMGR : DL,namr

to list cartridge and file information on file “namr” for any directory that contains that file; or

FMGR : DL,cartridge

to list the directory for the specified cartridge. An optional master security code parameter can be specified for all three command forms.

If two ASCII characters are specified for the CRN, the system interprets them as a namr. To list a specific cartridge, use a negative LU number or a positive CRN in integer form.

## DL Command Examples

### Example 1: Short form listing

```
FMGR : DL,21587
```

```
CR=21587
```

```
ILAB=TRAV NXTR=0005 NXSEC=036 #SEC/TR=060 LAST TR= 0133 #DR  
TR=01
```

```
NAME TYPE #BLKS/LU OPEN TO
```

```
BOOTEX 00001 00001  
&STH1 00003 00002  
&STH2 00003 00002  
%STH1 00005 00002  
%STH2 00005 00002  
TEST1 00006 00030  
TEST2 00006 00030  
JLIST 00003 00001 FMGR -  
JLIST 00003 00001 +001  
JLIST 00003 00001 +002
```

```
CR          = cartridge reference number  
ILAB        = cartridge label  
NXTR        = next available track  
NXSEC       = next available sector  
#SEC/TR     = number of 64-word sectors per track  
LAST TR    = last track on this cartridge  
#DR TR      = number of directory tracks on this cartridge  
NAME        = file name  
TYPE        = file type  
#BLKS/LU    = number of blocks in file or LU number of type 0 file  
OPEN TO     = a file extent number, if preceded by a plus (+) sign; or the  
              name of the program to which the file is open (a minus (-)  
              sign following the program name indicates that the file is  
              open to that program exclusively)
```

**Example 2: Long form listing**

```
FMGR : DL,-5,DX  
CR=21587  
ILAB=TRAV NXTR=0005 NXSEC=040 #SEC/TR=060 LAST TR= 0133 #DR TR=01
```

NAME	TYPE	#BLKS/LU	SCODE	TRACK	SEC	OPEN TO
BOOTEX	00001	00001	-32767	0000	000	
&STH1	00003	00002	00000	0000	002	
&STH2	00003	00002	00000	0000	006	
%STH1	00005	00002	00000	0000	018	
%STH2	00005	00002	00000	0000	022	
TEST1	00006	00030	00000	0001	034	
TEST2	00006	00030	00000	0002	034	
JLIST	00003	00001	00000	0005	034	
JLIST	00003	00001	00000	0005	036	+001
JLIST	00003	00001	00000	0005	038	+002

SCODE = security code of file  
TRACK SEC = track and sector address of start of file

**Example 3: The general forms of the DL command**

DL,222 (List all files on cartridge 222)  
DL,-33 (List all files on disk LU 33)  
DL,A (List all files whose name is A)  
DL,A----- (List all files whose name starts with A)  
DL,--A--A:-5:2 (List all files whose 3rd and 6th name characters are A, whose security code is -5 or 5, and which are in cartridge 2)  
DL,-----:1 (List all files with length of 1 block)  
DL,-----:16 (List all files with record length of 16 words)

## Initialize File Cartridge (IN)

**Purpose:** Builds or modifies the cartridge header information for a mounted disk cartridge. IN may also be used to assign or change the system's master security code.

**Syntax:** `IN,[master security],cartridge,crn,label[,first track [,#dir tracks [,#sectors/track]]]`

or

`IN,master security--new security`

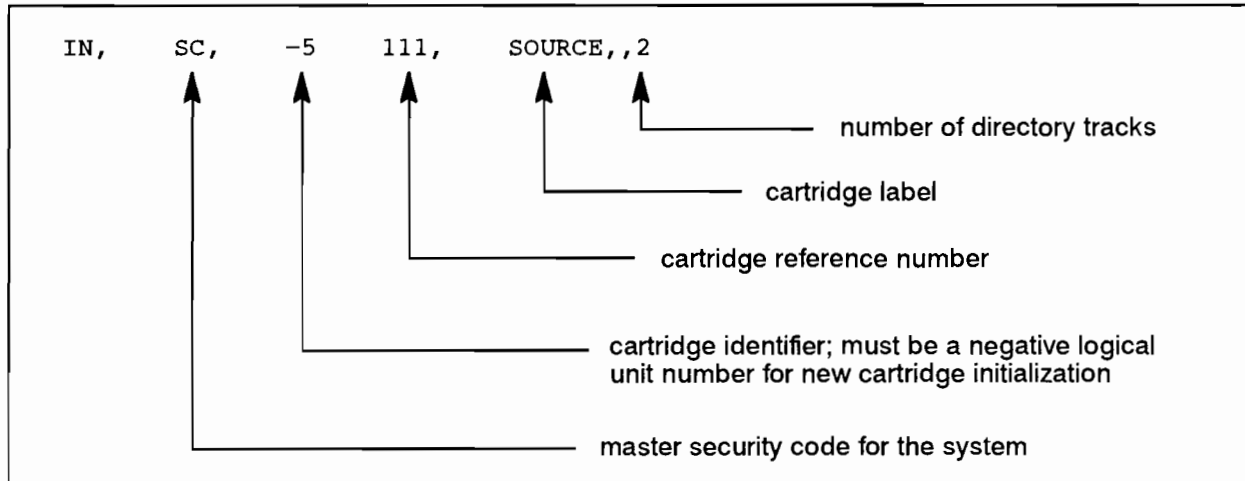
(The first format initializes a cartridge or changes the description of an initialized cartridge. The second format changes the master security code of the file system. This reverts to the original code when the system is re-booted.)

master security code (msc)	The security code that governs access to certain FMGR commands that can adversely affect the file system. This must be two ASCII characters. If they are omitted, directory and file security codes can be accessed with any security code or with none.
cartridge	The cartridge identifier that if positive, specifies the cartridge reference number; if negative, it specifies the logical unit number. It must be negative the first time the cartridge is initialized.
crn	The cartridge reference number that identifies the cartridge. It must be a positive integer from 1 through 32767 or two ASCII characters.
label	The cartridge label is a string of up to six ASCII characters. The same restrictions that apply to file names apply to the cartridge label (see the discussion of the namr parameter in the "File Manipulation" section in this chapter). The cartridge label is used to identify the contents of the cartridge; the system does not use it in any way. This label shows up in the heading for each cartridge listed by the DL command.
first track	The first FMP track on a cartridge, a positive integer. If this is omitted, track 0 is assumed.
#dir tracks	The number of directory tracks used by the file directory on the cartridge, a positive integer from 1 through 48 for a 96 sector/track LU, or 1 through 36 for a 128 sector/track LU. If it is omitted, one track is assumed. Note that using a value that is divisible by 7 will result in directory corruption if the disk is mounted as an FMGR disk. (Refer to the <i>RTE-A System Generation and Installation Manual</i> for more information.)
#sect/track	The number of 64 word sectors per track. If omitted, the driver is called for the value specified at generation.

Description:

The information that you supply when you invoke the IN command is used to build or modify the cartridge entry in the directory for that cartridge. You must use IN to initialize a cartridge before it is recognized by the file management system. Subsequently, you may re-initialize a cartridge, changing the parameters specified as often as necessary.

The command for initializing a new cartridge is illustrated below.



In the example above, the first track parameter is omitted, causing it to be set to zero. This means that the first file in the cartridge starts at logical track zero, sector zero. Except when using the IN command to purge an entire cartridge, there is no reason to specify any first track other than zero; using a non-zero first track wastes disk space.

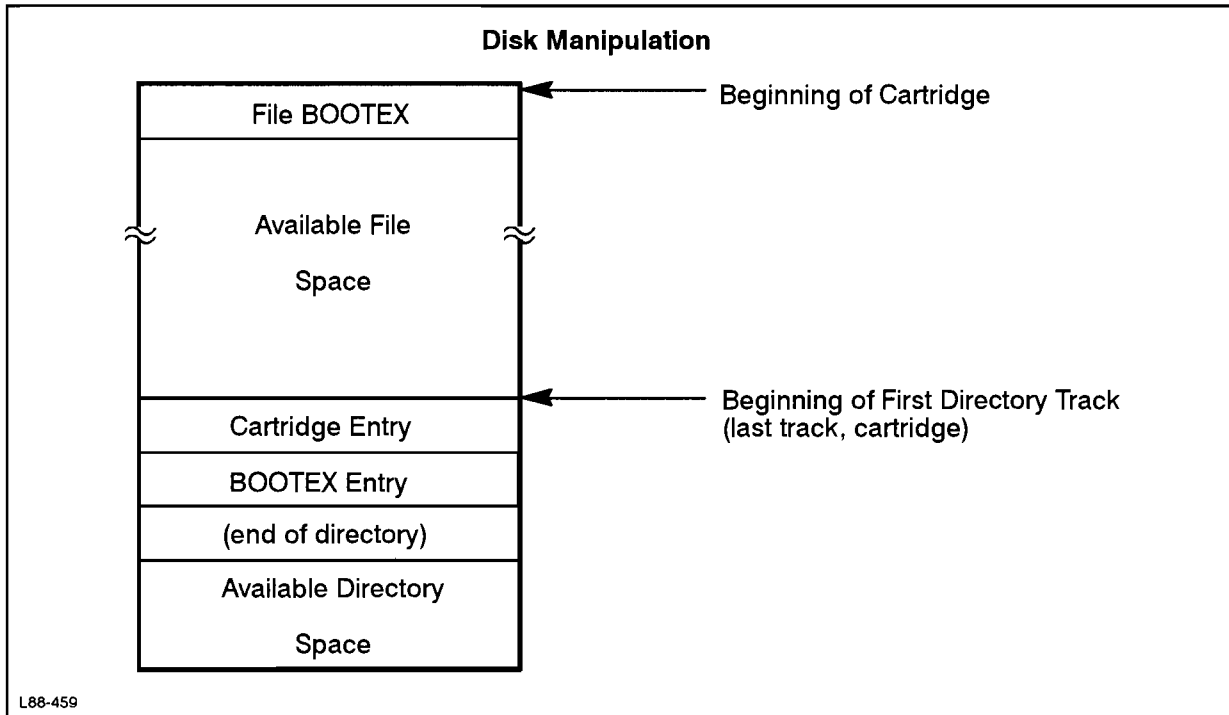
The number of directory tracks specified depends upon the number of files on the cartridge. Floppy (flexible) disks, as used by the HP 7902 disk drive, hold 240 directory entries per track. Among hard disks, HP 7906 and 7920 hold 384 directory entries per track, and HP 7925 holds 512 directory entries per track.

Each directory has a cartridge entry, an end-of-directory entry, plus one entry per file. The first directory track is the last track on the cartridge, the second directory track is the next-to-last track, and so on.

The parameters for the number of sectors per track are also omitted in the example above. You can omit the number of sectors per track and let the system supply the value. Specify 60 for a flexible disk (HP 7902), 96 for HP 7906 and 7920 disks, and 128 for an HP 7925 disk.

As part of the initialization, the boot extension file BOOTEX is created as the first file of the cartridge. BOOTEX is required if the cartridge is to be used for booting up the system. (See the *RTE-A System Generation and Installation Manual* for more details on BOOTEX and its use.) If you do not want BOOTEX on your cartridge, you can use the PU (Purge) command to purge the file.

Figure A-3 shows the structure of the initialized cartridge.



**Figure A-3. Initialized Cartridge Structure**

Note that if FMGR finds that the directory of a cartridge is not internally consistent, the program locks the cartridge to the copy of FMGR that discovered the discrepancy and issues a FMGR -103 error (“disk directory corrupt”).



## IN Command Error Handling

### FMGR 060 DO YOU REALLY WANT TO PURGE THIS DISK? (YES OR NO)

This is not an error, just a question. You entered an IN command that the system interprets as a request to purge all files on the cartridge. If you answer YES, the files are purged; if you answer NO, the command is aborted. For more information about purging files, see the section called “Purging All Files,” later in this chapter.

### FMGR –103 DIRECTORY IS CORRUPT

The file management system found that the information in the directory was not internally consistent. An FMGR –103 error was issued and the cartridge was locked to the copy of FMGR that found the error (that is, your copy). Even though the cartridge is locked, you can still resolve the problem by copying the files to a new cartridge with the CO command.

### FMGR –008 FILE IS ALREADY OPEN

The cartridge is in use by another program. Re-issue the command when it is free.

## Mount File Cartridge (MC)

**Purpose:** Mounts a file cartridge logically, creating an entry for it in the cartridge directory and making it available to the file management system.

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

`lu`                    The logical unit number of a disk. It may be a positive or negative number, but not a cartridge reference number.

`last track`            The last track on the cartridge available to the file management system. The default is the last track defined for the disk LU at system generation time.

### Description:

The `lu` parameter may be positive or negative (see example below), but it must refer to a disk logical unit. It cannot be a cartridge reference number. (The use of a cartridge reference number assumes an existing association between a logical unit and a cartridge reference number; it is the MC command that establishes that association.) For example, either of the following commands makes an entry in the cartridge directory for disk logical unit 10:

```
FMGR : MC,10
FMGR : MC,-10
```

You must both mount and initialize a cartridge (using the IN command, discussed in subsequent sections) before you can use it. Typically, a cartridge may be mounted and dismounted frequently, but initialized only once.

When MC is executed, an entry is established for the cartridge at the bottom of the cartridge directory (see the section “Cartridge Directory,” below). When you mount a cartridge, if FMGR discovers that the directory of a cartridge is not internally consistent, the cartridge is locked to the

copy of FMGR that discovers the discrepancy and the FMGR –103 error (“disk directory corrupt”) is issued.

## **MC Command Error Handling**

### **FMGR 012 DUPLICATE DISK LABEL OR LU**

You probably tried to mount an already mounted cartridge. Either the LU number or the CRN is already entered in the cartridge directory. You can check the contents of the directory with the CL (List Cartridge Directory) command.

### **FMGR –018 ILLEGAL LU. LU NOT ASSIGNED TO SYSTEM.**

You tried to mount a disk LU that is not recognized by the system. Make sure you are not trying to mount a cartridge by its CRN; you must use the LU number. If this is not the problem, check the I/O tables with the IO command to make sure that the LU you want to mount is actually a disk LU.

### **FMGR –103 DISK DIRECTORY CORRUPT**

The file management system found that the information in the directory was not internally consistent. An FMGR –103 error was issued and the cartridge was locked to the copy of FMGR that found the error (that is, your copy). Even though the cartridge is locked, you can still resolve the problem by copying the files to a new cartridge with the CO command.

### **FMGR –008 LOCK REJECTED**

The cartridge is in use by another program. Reissue the command when the cartridge is free.

## **Pack File Cartridge (PK)**

**Purpose:** To move files together on a cartridge to eliminate gaps left from purged files.

**Syntax:** PK[, cartridge]

cartridge      The cartridge identifier (negative LU or positive CRN) of the cartridge to be packed; for example, PK,-10. If you omit the cartridge identifier, all mounted cartridges are packed.

### **Description:**

The PK command locks a file cartridge before packing it. If the cartridge cannot be locked because files on it are open, the cartridge is not packed. If there are active program (programs that have been restored with the RP command) files (type 6) on the cartridge, the cartridge is packed beyond the last active program file.

Be aware that when you issue a PK command within a transfer file, if the transfer file crosses a sector boundary during PK execution, succeeding commands may be read from the wrong area of the disk. To prevent this problem, create a small (under one sector) transfer file that contains a PK command only. Then invoke the small transfer file either by itself or from within another.

If FMGR discovers that the cartridge is corrupt (the directory is not internally consistent), the cartridge is locked, it is not packed, and an FMGR –103 error displays.

## **PK Command Error Handling**

### **FMGR –103 DISK DIRECTORY CORRUPT**

The information in the directory is not internally consistent. An FMGR –103 error is issued and the cartridge is locked to the copy of FMGR that found the error (that is, your copy). Even though the cartridge is locked, you can resolve the problem by copying the files to a new cartridge with the CO command.

### **FMGR –008 LOCK REJECTED**

The cartridge is in use by another program. Reissue the command when the cartridge is free.

# File Manipulation

File manipulation refers to the operations performed on FMGR files such as copying, purging, listing, and so on. The sections that follow provide information about disk files and the various commands that you can use to perform file manipulation.

## Records and File Types

Disk files are composed of records of fixed or variable length, written in binary or ASCII code. The nature of the records determines the file type, as shown in Table A-4 below.

**Table A-4. Record and File Type Equivalences**

Category	Type	Description
Device	0	Non-disk (device) file
Fixed length, random access, not extendable	1	Fixed-length (128 words) records
	2	Fixed-length (user-defined) records
Variable length, sequential access, automatic extents	3	Variable-length records, any data type
	4	Program source file, ASCII
	5	Program object file, relocatable binary
	6	Executable file, memory-image code
	7	Absolute binary
	8 to	User-defined data format; types 8–99
	32767	Reserved

For example, the following frequently used programs produce the listed output file types:

Program	Output File type
RTAGN	1
EDIT	4
FTN7X	5
MACRO	5
LI	6

## Scratch Files

Scratch files are produced by a call to the CRETS routine. They provide temporary disk space for any program that needs a scratch area during its execution. When a program finishes using scratch files, a call to the PURGE routine eliminates them. Scratch files are also purged by the PK command. (The CRETS and PURGE routines are described in the *RTE-A Programmer's Reference Manual*.)

## Accessing a Disk File

Disk files are created and accessed through the file descriptor (FD) namr parameter. When it refers to a disk file, the namr parameter is equivalent to:

```
FD = filename[:sec[:cart[:file type[:file size[:record size]]]]]
```

Refer to the file using the form:

```
filename
```

or

```
filename::cartridge
```

If you use the first form, the cartridges are searched in the order listed in the cartridge directory. The second form restricts the file search to the specified cartridge.

For example,

```
FMGR : LI, SOURCE::AA
```

searches cartridge AA only for the SOURCE file and lists it when it is found.

The command form

```
FMGR : LI, SOURCE
```

starts searching the first cartridge in the cartridge directory and continues through the mounted cartridges in order of the cartridge directory until a file named SOURCE is found. This may be the file you want, or it may be another file named SOURCE on a different cartridge.

Specifying the cartridge along with the file name ensures the correct file is found, and provides the fastest search, since the system goes directly to the file directory of the named cartridge.

## Creating a File

When a disk file is created, an entry is made in the file directory on the cartridge to which the file is allocated. If a file of the same length as the new file was purged, the new directory entry replaces that of the purged file.

The file is created on the cartridge you specify, or, if you do not name a cartridge, on the first mounted cartridge with enough space to hold the file. If a file with the same name already exists on the cartridge, the new file is not created.

If the file is type 3 or greater, an EOF mark is written at the beginning of the file. As data are added to the file (with the DU command, described later), the file mark is moved to the end of the data.

If you create a file using a negative file size subparameter, the file occupies all the remaining available space on the cartridge (up to 16383 blocks).

A non-disk file is created as a type 0 file. In general, a type 0 file can be specified with just the required parameters (file name, LU number, I/O mode). The optional parameters usually follow from this information.

Non-disk (type 0) files are useful because they let you address a device as though it were a file. Programs can address devices using the standard FMP calls.

Use the CR (Create) command, described below, to create a new disk or non-disk (device) file. Note that this command does not store information in the file; it simply creates it.

## Purging Files

You can free up disk space by purging files that you no longer need. Once a file is purged, it cannot be accessed, and its name no longer appears on a file directory listing. A new file with the same name can then be created on the same cartridge.

Disk space allocated to purged files is automatically released to the system if the purged files are at the end of the cartridge. To reclaim disk space from other purged files, pack the cartridge with the PK command (described earlier in this appendix).

For example, assume files A, B, and C are the last three files in the directory. They are purged in order, A, B, and C. When C is purged, its disk space is released by the system. The system then checks file B, which is now the last file. Since B was already purged, its disk space is also released. Then file A is checked, and its disk space is released. This process continues until the system finds a file that was not purged.

One or more files can be purged each time you issue the PU command. Wildcard characters in the namr parameter cause all the files matching the namr to be purged from the disk cartridge.

You may purge a scratch file if you append an "A" to the file number. This is because the PU command does not purge a numbered file, but the addition of a character changes the scratch file name from a numeric to an ASCII character string.

## Storing Data on a Device or New File

You can use the ST (Store Data on a Device or New File) and DU (Dump Data to a Device or Existing File) commands to transfer data from one disk file or logical unit to another. ST is used when the destination is a new disk file (ST creates the file then stores the data), and DU is used for transfers to an existing file.

CS/80 cartridge tape drives (CTDs) should have files stored to them using either the TF or FC utilities, not the ST or DU command. See Chapter 2 of this manual for information on the TF and FC utilities.

The ST and DU commands are very similar in the way they operate. Both commands are described in subsequent sections.

## Listing the Contents of a File

You may list the contents of a file, file directory information, or data stored on a logical unit to a list device using the LI command. The maximum size of a record being listed is 128 words.

When you use the LI command, the output is directed to the list device. You can change the list device with the LL command.

## Renaming Files

At times, you may want to rename an existing disk file. The RN (Rename) command, described below, lets you do this. When you rename a file, the file name is the only characteristic that changes.

A new name must be unique to the cartridge on which the existing file resides.

## File Manipulation Commands

All the commands available for creating, purging, and listing FMGR files are discussed in the sections that follow. Table A-5 summarizes the commands.

**Table A-5. File Manipulation Commands Summary**

Commands	Description
<b>Listing Command</b>	
<b>LI</b> List Contents of a File	Lists a file, file directory information, or data on an LU to the list device
<b>Configuration Commands</b>	
<b>CR</b> Create A File	Creates a disk or non-disk (device) file
<b>DU</b> Dump Data to a Device or Existing File	Transfers data from one disk file to another
<b>PU</b> Purge a File	Removes selected files and their extents from a cartridge
<b>RN</b> Rename a File	Changes the name of an existing disk file
<b>ST</b> Store Data on a Device or New File	Transfers data from one disk file logical unit to another

### **Create a File (CR)**

**Purpose:** Creates a disk or non-disk (device) file.

**Syntax:** **For disk files:**

`CR, fileDes`

`fileDes` The file descriptor parameter, equivalent to:

```
fileName[:security[:cartridge[:fileType[:fileSize
[:recordSize]]]]]
```

This parameter must not be an LU number. Omitted subparameters default to 0. File type must be a positive integer and file size must not be zero. Record size need be specified only for type 2 files. (See Table A-4 for a brief description of file types.)



**For non-disk files:**

`CR, namr, lu, i/o mode[, spacing[, fileMark[, dataType]]]`

<code>namr</code>	The file name and, optionally, security code and cartridge reference; file type defaults to 0 and all remaining subparameters do not apply.
<code>lu</code>	The LU number of the non-disk device; a positive integer.
<code>i/o mode</code>	The required specification of one of the following:  <code>RE</code> Read only, accepts input only, forward spacing.  <code>WR</code> Write only, accepts output only, no forward spacing.  <code>BO</code> Both read and write, accepts input and output, backspacing and forward spacing allowed.
<code>spacing</code>	The type of spacing for the device; if omitted, forward spacing is assumed for read only devices and no spacing is assumed for others. May be specified as:  <code>BS</code> Backspacing is supported.  <code>FS</code> Forward spacing is supported.  <code>BO</code> Both backspacing and forward spacing supported.
<code>fileMark</code>	The type of EOF mark to be written on the device; if omitted, the default depends on the driver type. May be specified as:  <code>EO</code> End of file mark for magnetic tape (default if device has driver type greater than 16 octal, cartridge tape or mass storage device).  <code>PA</code> Page eject for line printer or two line feeds on teleprinter, device type 00 (default if not a punch or if driver type less than 17 octal).  <code>cntrl word</code> Control subfunction (equivalent to function code in FCONT call), supplied if further EOF definition is needed. Specify as octal integer of which only the lowest 5 bits are used.
<code>dataType</code>	The specified type of data on the device; default is ASCII.  <code>BI</code> Binary data  <code>AS</code> ASCII data  <code>cntrl word</code> Subfunction (equivalent to bits 6–10 of IOPTN parameter in OPEN call; supplied if further data definition is needed. Specify as decimal or octal integer of which only the lowest five bits are used.

## CR Command Examples

**Example 1:** Create a type 4 file with a security code of -25 (read and write are restricted to users knowing the code); put the file on cartridge 100, using 10 blocks (20 sectors) of disk space.

```
CR,MYFILE:-25:100:4:10
```

**Example 2:** Create a type 2 file, using 20 blocks, with each record 72 words.

```
CR,URFILE:::2:20:72
```

**Example 3:** Create a type 3 file with a security code of EJ (only write restricted), put on cartridge 100; allocate the remaining unused portion of the cartridge, up to 16K blocks, to the file.

```
CR,MYFILE:EJ:100:3:-1
```

**Example 4:** Create LP as output file on LU 6; defaults are no spacing and ASCII data.

```
CR,LP,6,WR,,PA
```

**Example 5:** Create MT as input/output file on LU 8, both forward and backspacing supported; security code is 32107.

```
CR,MT:32107,8,BO,BO
```

**Example 6:** Create a read only cartridge tape file on cartridge 100.

```
CR,MAG:JT:100,8,RE
```

**Example 7:** Create REALR as input only file on LU 5.

```
CR,REALR,5,RE
```

## CR Command Error Handling

### FMGR -002 File already exists

A file already exists on the specified cartridge with the requested name. Either purge the file, pick another cartridge, or use a different file name.

## Dump Data to a Device or Existing File (DU)

Purpose: Transfers data from one disk file or LU to another.

Syntax: `DU, namr1, namr2[, fmt/cntl[, file#[, #files]]]`

namr1	The source of data. This may be a disk file or an LU number (positive).
namr2	The destination of the data. This may be a disk file or an LU number (positive). If namr2 is a disk file, it must already exist.  Handling of the file type and file size parameters is the same as for the ST command; see the ST command discussion for details.
fmt/cntl	This may be a record format parameter, an EOF control parameter, or one of each (separated by a comma). The parameters and their defaults are the same as for the ST command, provided in the previous section.
file#	This parameter is a positive integer that specifies the starting point on namr2. If file# is n, transfer begins at the nth file, (if namr2 is a non-disk device) or subfile (if namr2 is a disk file). If you omit file#, transfer starts at the beginning of namr2. Transfer is always from the beginning of namr1.  Note that file# here applies to the destination file, whereas in the ST command this parameter applies to the source file.
#files	This parameter is a positive integer that specifies the number of files (if namr1 is a non-disk device) or subfiles (if namr1 is a disk file) to be transferred from namr1. The default is 1, except that if namr1 is a disk file and you omit file#, the default is 9999.

Description:

Note that files are transferred record by record, and records longer than 128 words are truncated. Default values and subfile considerations are the same as those for the ST command, provided in the sections above.



## DU Command Examples

**Example 1:** Dump the contents of file SORC3A to a line printer connected as logical unit 6.

```
DU, SORC3A, 6
```

**Example 2:** Dump the contents of binary relocatable file RELBN6 to a magnetic tape mini-cartridge mounted in logical unit 23.

```
DU, RELBN6, 23
```

**Example 3:** Copy a file from cartridge 333 to cartridge FD.

```
DU, FILEX: : 333, FILEX: : FD
```

**Example 4:** Take the third cartridge tape file on logical unit 8 and the first cartridge tape file on logical unit 23 and concatenate them into one disk file, TAPE12.

```
ST, 8, TAPE12, IH, 3  
DU, 23, TAPE12
```

**Example 5:** Add the contents of file B to file A when both are existing disk files.

```
DU, B, A, , 2
```

## List Contents of a File (LI)

**Purpose:** Lists a file, file directory information, or data on an LU to the list device.

**Syntax:** `LI, namr[, format[, line1[, line2]]]`

**namr** The file name or LU number. If the file is protected by a negative security code, you must specify it. If you include the CRN, only that cartridge is searched; otherwise, the first file with the same file name is listed.

**format** The list format specification:

**null** ASCII source format

**A** ASCII source format

**S** ASCII source format

**B** binary format

**D** directory information only

If you omit this parameter, the file type determines the format as follows:

**S** if file is type 0, 3, or 4

**B** for all other file types

**line1** The first record to list; default is 1.

**line2** The last record to list. If you do not specify line1 or line2, all records of the file are listed. If you specify line1 but not line2, only one record (line1) is listed.

**Description:**

Binary records longer than 128 words are truncated. Source records are truncated to 72 characters.

On a teleprinter, the list starts in column one; on other list devices two blanks precede each list line. If namr is a file, the listing is headed by:

```
fileName T=fileType IS ON CR cartridge USING size BLKS R=recSize
```

In this case, the lowercase words are replaced by the actual values in the file directory for the file (see the LI Command Examples, below). The record size has a value for type 1 and type 2 files only; all other file types have a record size of 0.

If namr is a logical unit, a brief heading is printed with asterisks replacing the file name. Here nn is the logical unit number.

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

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

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

## LI Command Example

```
FMGR : LI,&STH3
&STH3 T=00003 IS ON CR21587 USING 00002 BLKS R=0000
```

```
0001      PROGRAM TEST3
0002      INTEGER P(5)
0003      CALL RMPAR (P)
0004      LU = P(1)
0005      DO 120 J = 401,600
0006      WRITE(LU,116) J
0007 116  FORMAT(1X,"J = ",I3)
0008      IF(IFBRK(IDMY))200,120
0009 120  CONTINUE
0010      WRITE(LU,136)
0011 136  FORMAT(1X,"END STOP 3")
0012      GO TO 300
0013 200  WRITE(LU,216)
0014 216  FORMAT(1X,"BREAK STOP 3")
0015      300  END
```

```
FMGR : LI,&STH3,D
&STH3 T=00003 IS ON CR21587 USING 00002 BLKS R=0000
```

```
FMGR : LI,&STH3
&STH3 T=00005 IS ON CR21587 USING 00002 BLKS R=0000
```

```
REC# 00001
```

```
010400 020000 175566 052105 051524 031440 000127 000000*      TEST3  W
000000 000003 000143 000000 000000 000000 000000 000000*
000000
```

```
REC# 00002
```

```
003400 040001 015277 052105 051524 031440 000005      * @ ?TEST3
```

```
REC# 00003
```

```
014000 100007 153051 027104 044517 027003 027111 044517*      V).DIO. .IIO
027004 027104 052101 027005 042530 042503 020007 041514*. .DTA. EXEC CL
051111 047401 051115 050101 051002 044506 041122 045406*RIO RMPAR IFBRK
```

```
.
.
.
```

...and so on.

## Purge a File (PU)

Purpose: Removes selected files and their extents from a cartridge.

Syntax: `PU, fileDescriptor[, msc]`

`fileDescriptor` The file name (see comments) and, optionally, security code, cartridge identifier (negative LU number or positive CRN, or 2 ASCII characters), file type, and file size.

`msc` The master security code; used to override file security code on wildcard purges.

Description:

If a file is protected by a security code, you cannot purge it unless you enter the correct code or the master security code is supplied.

When you specify a cartridge identifier, only that cartridge is searched for the file to be purged; otherwise, all mounted cartridges are searched and the first file found with the specified name is purged. The search starts with the first cartridge in the cartridge directory. The CL command shows you the order in which cartridges are searched when a cartridge identifier is not specified.

## PU Command Examples

**Example 1: Purge the first file found with a name of DATA7.**

```
FMGR: PU, DATA7 (Namr1 is DATA7 with no security code and no disk id supplied.)
FMGR: (Ready for next command.)
```

**Example 2: Purge a file named A4 with a security code of XX on cartridge number 43.**

```
FMGR: PU, A4:XX:43 (Namr1 is A4 with security code XX on cartridge 43.)
FMGR: (Ready for next command.)
```

**Example 3: Purge all files on cartridge 30 starting with FILE. You must enter the master security code in order to ensure that all files regardless of security code are purged.**

```
FMGR: PU, FILE--: :30, YY (Namr is FILE-- on cartridge 30. The msc is YY.)
FILEA
FILEB
FMGR: (Ready for next command.)
```

## Rename a File (RN)

**Purpose:** Changes the name of an existing disk file.

**Syntax:** `RN, fDesc, newName`

`fDesc` The file name and, optionally, security code and cartridge identifier (negative LU number or positive CRN).

`newName` The new file name to replace existing file name.

**Description:**

When you include a cartridge identifier in `namr`, only that cartridge is searched. If you omit the cartridge identifier, the program searches through all the mounted cartridges and renames the first file it finds. The search starts with the first cartridge in the cartridge directory. You can use the CL command to see the order in which the cartridges are searched.

Note that if the file you are renaming was created with a non-zero security code, you must include the same security code in the `namr`.

## RN Command Examples

**Example 1:** Search the cartridge for the first occurrence of file JD17 and change its name to DATA1. The file is not protected by a security code.

```
RN,JD17,DATA1
```

**Example 2:** Rename file LOG3 on cartridge 100 to Table 2. The file has a security code of XX.

```
RN,LOG3:XX:100,TABLE2
```

## Store Data on a Device or New File (ST)

**Purpose:** Transfers data from one disk file or logical unit to another.

**Syntax:** `ST,namr1,namr2[,fmt/cntl[,file#[,#files]]]`

`namr1` The source of data. This may be a disk file or a positive LU number.

`namr2` The destination of data. This may be a disk file or a positive LU number. If `namr2` is a disk file, it must not already exist.

`fmt/cntl` This may be a record format parameter, an EOF control parameter, or one of each (separated by a comma).

The record format (`fmt`) parameter specifies the format of data in `namr1`, and may be:



- AS ASCII records are transferred.
- BR Binary relocatable records are transferred; checksum is performed.
- BN Binary relocatable records are transferred; no checksum is performed.
- BA Binary absolute records are transferred; checksum is performed.

The EOF control (cntl) parameter specifies the treatment of file marks. The term “file mark” refers to a subfile mark on a disk file or an EOF mark on a non-disk file. As embedded file marks are transferred, they are changed to the form that is appropriate for namr2: EOF marks for non-disk file, subfile marks for disk files. (See the explanation of subfiles, below.)

The EOF control parameter may be:

- IH No file marks, embedded or terminating, are passed to namr2.
- SA All file marks, embedded and terminating, are passed to namr2.

If the EOF control parameter is omitted, embedded file marks are not passed to namr2, but the terminating file mark is passed.

- file# This parameter is a positive integer that specifies the starting point on namr1. If file# is n, transfer begins with the nth file. If you omit file#, transfer starts at the beginning of namr2.
- #files This parameter is a positive integer that specifies the number of files (if namr1 is a non-disk device) or subfiles (if namr1 is a disk) to be transferred from namr1. Default is 1, except that if namr1 is a disk file and file# is omitted, the default is 9999.

**Description:**

Note that files are transferred record by record; records longer than 128 words are truncated.

## Default Values

If namr2 is a disk file and the file type subparameter is not specified, then:

If namr1 is a disk file, the file type of namr1 is assigned to namr2.

If namr1 is a device, the file type of namr2 is:

type 3 if the record format of namr1 is omitted  
type 3 if the record format of namr1 is AS  
type 5 if the record format of namr1 is BR  
type 3 if the record format of namr1 is BN  
type 7 if the record format of namr1 is BA

If namr2 is a disk file and the file size subparameter is not specified, then:

If namr1 is a disk file, the file size of namr1, without extents, is assigned to namr2.

If namr1 is a device, the file size of namr2 is set to 24 blocks.

If the record format is not specified and namr1 is a disk file, the record format defaults to:

AS if namr1 is a type 3 or type 4 file  
BR Binary relocatable records are transferred; checksum is performed  
BN Binary relocatable records are transferred; no checksum is performed  
BA Binary absolute records are transferred; checksum is performed

## Subfiles

Subfile marks are zero-length records that are used to subdivide a disk file. You can use subfile marks to store several non-disk files (say, cartridge tape files) on one disk file, but still maintain them as separate entities. The EOF mark used by a non-disk device depends on the type of device:

- A 264x cartridge tape unit uses a special magnetic tape EOF mark.
- A line printer uses a top-of-form (page feed) control sequence.
- A terminal uses a control D on input and a zero-length record (two consecutive carriage returns) on output.

When files containing these marks are transferred to a disk file, the file marks may be written as EOF or subfile marks, or may be deleted, according to the EOF control parameter. When a disk file is transferred, its subfile marks (if any) may be transferred as subfile marks (to another disk file), as EOF marks, or may be deleted.

## ST Command Examples

Although the ST command can become rather complex in some situations, most of the time it is quite simple to use. The following examples show some typical uses of the ST command.

**Example 1:** Transfer a disk file of FORTRAN source code, &FORT2, to a magnetic tape on logical unit 8.

```
ST, &FORT2, 8
```

**Example 2:** Transport the source file, &FORT4, of a program stored on a model 7908 hard disk (for example, LU 20), to another system on a flexible disk that has a CRN equivalent to FD.

```
ST, &FORT4::-20, &FORT4::FD
```

When you do this, you can then carry the flexible disk to the other system and operate from the flexible disk or transfer the program to a different disk once again with the ST command.

**Example 3:** Transport a file of binary relocatable code, %RELO5, from LU 20 to CRN FD.

```
ST, %RELO5::-20, %RELO5::FD
```

**Example 4:** Transfer the first three files on a cartridge tape (A, B, and C), which contain program source code (ASCII), to disk as separate files, subfiles, or one file.

Tape	Command	Disk
----- A    B    C -----	ST, 8, A   ST, 8, B   ST, 8, C ----->	----- A    B    C -----
3 Separate Files		3 Separate Files
	ST, 8, ABC, SA, 3 ----->	----- A    .    B    .    C .       . -----
		3 Subfiles
	ST, 8, ABC, , , 3 ----->	----- A    B    C -----
		One File

**Example 5:** Transfer a disk file composed of three ASCII subfiles (A, B, and C) to cartridge tape as separate files or as one file.

Disk	Command	Tape
<hr/> A . B . C . . <hr/>	ST,ABC,8,SA <hr/>	<hr/> A B C <hr/>
3 Subfiles		3 Separate Files
	ST,ABC,8 <hr/>	<hr/> A B C <hr/>
		One File

**Example 6:** Concatenate two disk files, D and E, on one magnetic tape file.

ST,D,8,IH <hr/>	<hr/> D <hr/>
	No File Mark
	v
	ST,E,8
	v
	<hr/> D E <hr/>
	One File

**Example 7:** Take the fourth, fifth and sixth files of binary relocatable code on a magnetic tape mounted on LU 8 and copy them to three subfiles on disk file IBINR3 on cartridge CC.

```
ST, 8, !BINR3::CC, BR, SA, 4, 3
```

Note that in this case, you must specify the format parameter (BR). In Example 3 above, the format parameter was not necessary because it could be derived from the file type. In the case of a cartridge tape file, however, the system has no way of knowing the file type or the format type, so you must specify the format type.

## Transfer Files

A transfer file (sometimes called a procedure file) is, in effect, a program written in the language of the operator commands. It lets you execute a series of commands by transferring control to a file that contains those commands. This is very useful when you need to execute a particular sequence of commands repeatedly.

### Creating a Transfer File

Suppose you are using a terminal assigned to LU 13 and are debugging a particularly troublesome FORTRAN program. You make repeated runs of the compiler, the loader, and the loaded program by typing in the following command sequence:

```
FMGR: PU,TEST4                (Purge the old type 6 file)
FMGR: RU,FTN7X,&SORCE,13,%SORCE (Compile the source code)
FMGR: RU,LINK,,%SORCE          (Load the relocatable code)
FMGR: RU,TEST4                (Run the new program)
```

You can make your job easier by creating a transfer file that looks like this:

```
:PU,TEST4
:RU,FTN7X,&SORCE,13,%SORCE
:RU,LINK,,%SORCE
:RU,TEST4
:TR
```

You can create this file with the text editor program. The only difference between this file and the original sequence of commands is that a colon (:) precedes each command, and there is a TR statement at the end of the file. If you name this file TFILE4, you can compile, load and execute your program by typing the following command in response to an FMGR prompt:

```
TR.TFILE4
```

This causes FMGR to transfer control to file TFILE4 and execute the commands contained in the file. FMGR terminates the old program and releases its ID segment, compiles the new program, loads it, executes it, and returns control to your terminal.

This is much easier than retyping the commands each time, and it lets you spend your time debugging the program rather than typing at your terminal.

### G-Type Global Parameters

G-type global parameters are variables that increase the generality of a transfer file. Commonly called G globals, they hold an integer value or an ASCII string up to six characters long. There are nine general and two specialized G globals.

You can directly assign values to globals 1G through 9G. Global 0G is set to the namr value of the input device when FMGR was scheduled; thus, 0G usually contains the LU number of the terminal you are using.

Global 10G is set by a call to library routine PRTN (refer to the *RTE-A Programmer's Reference Manual*, and see also the explanation of P-type globals, below). The most visible use of global 10G is that the on-line loader sets 10G equal to the name of the most recently loaded program or to a loader error code if the load failed.

You can pass values to globals 1G through 9G as part of the TR command or you can set them with the SE (Set) or CA (Calculate) commands in a transfer file. You can also make conditional jumps with the IF (Conditional Skip) command. (The TR, SE, CA, and IF commands are discussed in the following sections.)

The following example shows how you can set up a more general transfer file to compile, load, and execute a FORTRAN program.

```
:PU, 3G
:RU,FTN7X, 1G, 0G, 2G
:RU, LINK, , 2G
:IF, 10G, EQ, 3G, 2
:DP,LOAD FAILED,RETURNING CONTROL TO TERMINAL
:TR
:RU, 10G
:TR
```

You can name this file TESTG and execute it with the following command:

```
FMGR : TR,TESTG,&SorCe,%SORCE,TEST4
```

When values are substituted for globals 0G, 1G, 2G, 3G and 10G, the file becomes:

```
:PU,TEST4
:RU,FTN7X,&SORCE, 13,%SORCE,TEST4
:RU, LINK, ,%SORCE
:IF,TEST4,EQ,TEST4,2
:DP,LOAD FAILED,RETURNING CONTROL TO TERMINAL
:TR
:RU,TEST4
:TR
```

The IF statement tests whether the load is successful. When it is successful, control skips two statements and schedules the program that was just loaded. If the load is unsuccessful, the value of 10G is set to a loader error code and the transfer file prints the message (DP statement) and terminates.

Each G global is composed of four words. The first word, word 0, defines the type of value held by the global; the remaining words, words 1 through 3, hold the actual value. The arrangement is shown below:

word 0 (type)	0 (null)	1 (numeric)	3 (ASCII)
word 1 (value)	0	integer	characters 1 & 2
word 2 (value)	0	0	characters 3 & 4
word 3 (value)	0	0	characters 5 & 6

## P-Type Global Parameters

P-type global parameters, commonly known as P globals, are one-word global parameters numbered from -36P to 7P. They are related to G globals as shown in Figure A-4.

0G	word 0 (type) word 1 word 2 word 3	
1G	word 0 (type) word 1 word 2 word 3	-36P -35P -34P -33P
2G	word 0 (type) word 1 word 2 word 3	-32P -31P -30P -29P
3G	word 0 (type) word 1 word 2 word 3	-28P -27P -26P -25P
4G	word 0 (type) word 1 word 2 word 3	-24P -23P -22P -21P
5G	word 0 (type) word 1 word 2 word 3	-20P -19P -18P -17P
6G	word 0 (type) word 1 word 2 word 3	-16P -15P -14P -13P
7G	word 0 (type) word 1 word 2 word 3	-12P -11P -10P -9P
8G	word 0 (type) word 1 word 2 word 3	-8P -7P -6P -5P
9G	word 0 (type) word 1 word 2 word 3	-4P -3P -2P -1P
10G	word 0 (type) word 1 word 2 word 3	0P 1P 2P 3P 4P 5P 6P 7P

Figure A-4. Global Parameters



You can use P globals to examine any of the words in G globals 1G through 10G. In addition, globals 1P through 5P contain the values passed to and from programs using the RMPAR and PRTN subroutines. (RMPAR and PRTN are discussed in the *RTE-A Programmer's Reference Manual*.)

For example, after executing a program, the value returned in global 10G (actually 1P to 3P) is either zero or the return parameters of the program. Global 6P contains the latest error code, and global 7P contains a copy of the current severity code.

If you are using the contents of global 6P to check for errors, note that the program clears the error code (6P is reset to 0) after the error message is expanded with the ?? command. The error code is also cleared after automatic error expansion (severity code set at 1000 to 1004).

Thus, a check of global 6P always returns a value of zero if automatic error expansion is in effect (because the error code is cleared immediately after it is reported and expanded). Therefore, if your transfer file uses the value in 6P, you should make sure that the severity code is not set for automatic error expansion.

The FORTRAN compiler passes error information via a PRTN call, and the information shows up in globals 1P through 5P. Of particular interest is the value in 1P, which shows the total number of errors, warnings, and disasters encountered during compilation.

The following transfer file expands on the one in the previous example by setting 1P to zero (CA statement) before compilation, then checking to make sure that there were no errors during compilation. The value of the current severity code is saved, and the severity code is reset to 1 (no command echo) while the transfer file executes, and then it is reset to the original value at the end of the file. Two IF statements are used for unconditional skips and the :\* statements are comment lines.

```

:* TRANSFER FILE TESTG      (Comment, not executed)
:*
:CA, 4G, 7P                 (Save severity code in 4G)
:SV, 1                      (Set severity code to 1)
:CA, 1:P, 0                 (# set 1P to 0)
:PU, 3G                     (# purge the old type 6 file)
:RU, FTN7X, 1G, 0G, 2G     (Compile source code)
:IF, 1P, EQ, 0, 2          (Test for compilation errors)
:DP, ** COMPILATION ERROR **
:IF, 1, EQ, 1, 5           (Unconditional skip)
:RU, LINK, , 2G           (Load the relocatable code)
:IF, 10G, EQ, 3G, 2       (Test for load errors)
:DP, ** LOAD ERROR **
:IF, 1, EQ, 1, 1          (Unconditional skip)
:RU, 10G                   (Run the program)
:SV, 4G                     (Reset severity code)
:DP, ** TESTG TERMINATING **
:TR                         (Return)

```

You can execute this transfer file with a command such as:

Loops

You can also perform loops in a transfer file using the CA command and the IF test with a negative skip count, as follows:

```

.
.
.
FMGR : CA,7G,0                (Initialize a counter)
.
.
a
few
other
commands
.
.
FMGR : CA,7,7G+,1             (Increment the counter test; go to top of loop)
FMGR : IF,7G,LT,10,-6
.
.
.

```

(These commands are executed 10 times)

## TR and Related Commands

Table A-6 summarizes the commands for creating and using transfer files.

**Table A-6. Transfer Commands Summary**

Commands	Description
<b>TR</b> Transfer Control to a File or Device	Transfers control to a file or a logical unit
<b>SE</b> Set Global Parameter	Assigns values to the global parameters 1G through 9G
<b>CA</b> Calculate Global Parameter	Assigns individual G-type and P-type global parameter values or nulls them
<b>IF</b> Conditional Skip	Compares two values (usually global parameters) number of commands, depending upon the result of the comparison
<b>PA</b> Pause and Send Message	Suspends execution of a transfer file and transfers control to the log device or other specified device
<b>DP</b> Display Parameters	Displays the current values of up to 14 parameters on the log device, which must be an interactive terminal

## Calculate Global Parameter (CA)

Purpose: Assigns individual G-type and P-type global parameters values or nulls them.

Syntax: `CA,global#[,p1[,op1,p2[op2 ... ,opn,pn+1]]]`

**global#** The integer (1–9) that identifies the global 1G through 9G to be set to the result of the calculation.

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

`n:P`

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

`:CA,6:P,0`

or

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

You can also specify the global type in the first subparameter. “P” identifies P-globals, and “G” identifies G-globals. If you omit the subparameter, G-globals are assumed.

**p1–pn** Values used in calculations. If you omit entering them, the global is nulled.

**op1– opn** Operations performed on operands. These may be:

`+` add two operands

`-` subtract one operand from another

`/` divide one operand by another

`*` multiply one operand by another

`O[R]` inclusive OR two operands

`X[OR]` exclusive OR two operands

`A[ND]` AND two operands

Description:

In its simplest form, CA is used to null an individual global parameter or to set an individual global to the value of p1. The values assigned can be the result of arithmetic or logical calculations.

The type of result depends upon the type of operands you enter. If operand types differ in any one CA statement, the highest type value is used (type 0 = null, type 1 = numeric, and type 3 = ASCII). For example, if one operand is numeric and the other is ASCII, the result is ASCII.

Calculations are performed separately on each word of three-word ASCII globals, except for division and multiplication, in which all three words of the first operand are divided or multiplied by word 1 of the second operand.

When you invoke the command, evaluation proceeds from left to right until a null operation code is detected. Multiple CA statements may change the precedence.

You cannot set all of the globals; some are used to provide system information. The G-globals that cannot be set are 0G and 10G, and the P-globals are 0P and 7P.

## CA Command Examples

FMGR : <u>CA, 6, FTN7X</u>	(Set global 6G to ASCII value "FTN7X")
FMGR : <u>DP, -16P</u> 3	(Display type of 6G is ASCII (3))
FMGR : <u>CA, 6</u>	(Clear global 6G to null value)
FMGR : <u>DP, -16P</u> 0	(Display type of 6G null (0))
FMGR : <u>CA, 2, 15</u>	(Set global 2G to integer value 15)
FMGR : <u>DP, -32P</u> 1	(Display type of 2G numeric (1))
FMGR : <u>CA, 7, 2G</u>	(Set global 7G to current value of 2G)
FMGR : <u>DP, -12P</u> 1	(Display type of 7G assumes type of 2G)
FMGR : <u>DP, 2G, 7G</u> 15, 15	(Display values of 2G, 7G)
FMGR : <u>CA, 1, 2G, *, 14, +, 1</u>	(Set 1G to product of 2G and 14 plus 1)
FMGR : <u>DP, 1G</u> 211	(Display 1G)
FMGR : <u>CA, 7, 7G, -, 1</u>	(Decrement 7G by 1)
FMGR : <u>DP, 7G</u> 14	(Display 7G)
FMGR : <u>CA, 1, 7, OR, 15</u>	(Inclusive OR 7 and 15 (octal 17), assign to 1G)
FMGR : <u>CA, 2, 7, XOR, 15</u>	(Exclusive OR same values, assign to 2G)
FMGR : <u>CA, 3, 7, AND, 15</u>	(AND these values, assign to 3G)
FMGR : <u>DP, 1G, 2G, 3G</u> 15, 8, 7	
FMGR : <u>DP, -23P</u> 0	(Manipulation of P globals)
FMGR : <u>CA, -23:P, -23P, +, 1</u>	
FMGR : <u>DP, -23P</u> 1	

## Display Parameters (DP)

**Purpose:** Displays the current values of up to 14 parameters on the log device, which must be an interactive terminal.

**Syntax:** DP[,p1[,p2...[,p14]...]]

p1 to p14      The parameter values or names of global parameters to be displayed. These may also be a string of text. If you do not enter any values, nothing is displayed.

**Description:**

Although you may use the DP command to display any parameter value, it is typically used to display the values of global parameters (G-globals or P-globals). Null global parameters are displayed as a pair of commas (,,). See also the discussion of global parameters in previous sections in this appendix.

You can use DP to send messages to the log device without causing a transfer of control, as shown in Example 1 below.

Note that the DP command displays parameter values even if the severity code is greater than zero.

## DP Command Examples

The following excerpts from a transfer file illustrate some of the uses of the DP command:

```
:
:
:
:DP,First section (initialization) completed.
:
:
:
:DP,Output file is,1G
:DP,Values of -36P through -33P are, -36P, -35P, -34P, -33P
:
:
:
:DP,Job done!
:
:
:
```

When the transfer file is executed, the log device displays:

```
First section (initialization) completed.  
  
Output file is,FILE23  
Values of -36P through -33P are,3,FI,LE,23  
  
Job done!
```

## Conditional Skip (IF)

**Purpose:** Compares two values (usually global parameters) and skips a specified number of commands, depending upon the result of the comparison.

**Syntax:** IF,p1,operator,p2[,skip]

p1,p2            The values to be compared; one or both may be global parameters.

operator        The relative operator used to compare values of p1 and p2, entered as one of the following two-character keywords:

Keyword	Operation
EQ	p1 = p2
NE	p1 /= p2
LT	p1 < p2
GT	p1 > p2
GE	p1 @>@ p2
LE	p1 @<@ p2

skip            The skip count, a positive or negative integer that specifies the number of commands to skip when the relation between p1 and p2 is true; forward skip if positive, backward if negative. If you omit this parameter, the program skips one command. If the relation is not true, the next sequential command is executed.

**Description:**

Note that you cannot execute IF from an interactive device, only from within a transfer file.

The specified relation between p1 and p2 is examined and if it is true, commands are skipped. If you do not specify a skip count, one command is skipped; otherwise, as many commands as the specified skip number are skipped. A skip of -1 causes the IF command to repeat. To skip back to the preceding command, specify -2.

IF does not skip past the beginning or the end of the transfer file; an attempt to do so causes a skip to the beginning or EOF mark, but does not cause an error.

To use a negative skip, the file must be on a device that recognizes a backspace.

The following relations hold for mixed types:

    null < numeric < ASCII

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

If p1 and p2 are both ASCII, the comparison is based on the ASCII collating sequence.

You may also use the IF command to do an unconditional skip by making a comparison that is always true (see the examples below).

## IF Command Examples

```
:* PERFORM TWO SECTIONS OF COMMANDS
:* DEPENDING ON THE STATE OF 1G
:*
:IF,1G,EQ,1,4
:IF,1G,EQ,2,9
:DP,VALUE OF 1G (,1G,) IS OUT OF RANGE
:SE
::
:* EXECUTE THIS CODE IF 1G = 1
: first command
: second command
: third command
:* NOW UNCONDITIONALLY SKIP THE '1G = 2' CODE
:IF,1,EQ,1,4
:* EXECUTE THIS CODE IF 1G = 2
: first command
: second command
: third command
:* EXECUTION CONTINUES HERE
:
```

To do an unconditional skip, just make sure that the comparison is always true:

```
:** IF 7G IS NEGATIVE, ZERO OUT 4G
:** ELSE (POSITIVE OR ZERO) PUT 7G INTO 4G
:**
:IF,7G,GE,0,2
:CA,4,0
:IF,0,EQ,0,1
:CA,4,7G
:CA,5,4G,*,6G,+,1
:
:
:
```



## Pause and Send Message (PA)

**Purpose:** Suspends execution of a transfer file and transfers control to the log device or other specified device.

**Syntax:** PA, lu

lu                    The LU number of the device to which control is transferred and on which messages are displayed. If you do not specify an LU, the log device is assumed.

### PA Command Example

You can use PA in a transfer file to request and wait for operator action/response. For example, assume a transfer file contains the following command:

```
:DP,,Insert cartridge tape on LU 8, then type TR.  
:PA
```

When the command executes, the log device displays:

```
:Insert cartridge tape on LU 8, then type TR.
```

The PA command gives you a chance to respond. After you mount the tape and type TR on the log device, control is transferred back to the transfer file and execution continues with the next command after PA.

## Set Global Parameter (SE)

**Purpose:** Assigns values to the global parameters 1G through 9G.

**Syntax:** SE[,p1[,p2[...[,p9]]]]

p1 through p9            Values assigned to global parameters 1G through 9G. If you omit all parameters, are nulled. If you omit any one parameter, the corresponding global parameter is unchanged.

**Description:**

The value in p1 is assigned to 1G, that in p2 to 2G, and so on. You can use SE alone (no parameters) to have all the global parameters 1G through 9G cleared (nulled). Individual global parameters can be nulled or set with the CA command.

You may assign any integer value between -32768 and 32767 or an ASCII value up to six characters to p1 through p9. If, however, you enter a global name (0G - 10G, 1P - 5P) as a parameter, the value of the specified global is assigned to the global parameter that corresponds to the parameter position.



## SE Command Examples

**Example 1:** Assign global 5G. Assign values to globals 1G through 4G. Keep 5G through 9G unchanged. Display the globals (DP).

```
SE,,,,,FIVE
SE,256,NEWFIL,AA,0
DP,1G,2G,3G,4G,5G
256,NEWFIL,AA,0,FIVE
```

**Example 2:** Set global 1G to the value of 4G, in this case, 0.

```
SE,4G
DP,1G
0
```

## Transfer Control to a File or Device (TR)

**Purpose:** Transfers control to a file or a logical unit.

**Syntax:** TR[ , ,parameters ]

or

TR,namr[ ,parameters ]

or

TR,-integer[ ,parameters ]

**namr** The namr identifies a file or logical unit to which control is transferred. Pointers to the namrs are maintained in a transfer stack and may be nested up to 8 levels deep.

**-integer** A negative integer. Control is transferred back the specified number of levels in the transfer stack. If the integer is greater than the current level, the stack is reset to the top.

**parameters** The values to be set for the global parameters 1G through 9G. Position determines the global parameter to which the value is passed; omitted global parameters are unchanged.

Note that a colon (:) or comma (,) may replace TR as the command code.

**Description:**

The form TR,namr is like a subroutine call. It transfers control to the specified namr (either a file or a device) and starts executing whatever commands it finds. When all the commands are executed, TR causes transfer of control back to the calling namr (like the RETURN statement of a subroutine), and execution continues with the first statement after the calling TR statement.

When you issue a PK command from within a transfer file, it is important to ensure that the transfer file does not cross a sector boundary, else, commands that follow PK may be read from the wrong area of the disk. It is best to make a file that contains the PK command only and invoke this transfer file from within other transfer files when necessary.

A TR command (with no parameters) is the same as a TR,-1 command. Both commands transfer control back one level in the transfer stack. A TR,-3 transfers control back three levels in the transfer stack, and so on.

When you transfer back to a file on disk, it is possible to backspace and re-execute one or more commands within that file. To do this, specify a negative integer as the security code of a null namr. For example, the command TR,;-4 transfers back to the previous namr, backsapes four commands, and then begins to execute. (There is no way to skip levels and backspace at the same time.)

If an error that requires operator intervention occurs, the system transfers control to the log device. You can use the TR command (either TR or a colon) to transfer control back to the file or device where the error occurred.

## TR Command Examples

**Example 1: Transfer control to file TFILE1.**

```
TR, TFILE1
```

**Example 2: Transfer control to file TFILE2 and set the value of global parameter 1G to FILEA.**

```
TR, TFILE2, FILEA
```

**Example 3: Transfer control to file TFILE3 and set the value of global parameter 4G to 4. Values of all other global parameters are unchanged.**

```
TR, TFILE3, , , , 4
```

# Device Manipulation

Device manipulation includes taking down a device and displaying and modifying buffer limits. The following sections discuss how to call FMGR and COMND, and describe the DN (Make Device Unavailable) and the BL (Display/Modify Buffer Limits) commands. (See end of appendix for more information on the COMND program.)

## Calling FMGR and COMND

To call FMGR and COMND, enter the commands in the runstring as shown below, from a device or from a file.

FMGR: RU,FMGR[,input[,log[,list[,severity]]]] (Commands entered from device)

FMGR: RU,FMGR,fi,le,nm[,severity[,list]] (Commands entered from file)

FMGR: RU,COMND[,INPUT[,LOG]]

- |          |  |
|----------|--|
| input    | The LU number of the input device for FMGR or COMND commands. If you do not enter an LU number, the console log device is assumed.   |
| log      | The LU number of the log device used to log and to correct any diagnosed errors. It must be an interactive device.   |
| list     | The LU number of the device used to list results of FMGR commands. If you do not enter another, LU 6 is assumed.   |
| fi,le,nm | The name of the file that contains command input to FMGR commands. Specify 3 words, each consisting of 2 ASCII characters.   |
| severity | The severity code that defines the action in case of error messages.<br><ol style="list-style-type: none"><li>0 Display error codes and echo commands on log device (the default).</li><li>1 Display error codes on log device, inhibit command echo.</li><li>2 Error code displays only if error requires transfer of control to log device for correction; in this case, the active job terminates. No command echo.</li><li>3 Same as 2, except that the active job does not terminate when an error causes transfer to the log device.</li><li>4 If an FMGR command error is encountered, the job continues automatically. There is no command echo, no transfer to log device, and no job abort occurs.</li></ol> |

Once FMGR is running, you may change the list device using the LL command, and the severity code using the SV command.

Alternatively, you can call FMGR from a program, as follows:

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

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

**ICODE**        This is 23 to schedule FMGR with wait, or 24 to schedule FMGR without wait.

**FMGR**        A 3-word array that contains ASCII file name FMGR.

**INP,LOG,LIST,ISV**  
**FILE,NM,ISV,LIST**

These correspond to the parameters in the RU,FMGR command.

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

**IBUFR,IBUFL**

The buffer address and length containing the command string to be passed to FMGR.

You must begin the command string that you want to pass to FMGR via IBUFR with a colon, or the program ignores the commands. The buffer length that you specify in IBUFL is a positive value for words, or a negative value for characters.

FMGR assumes that the string you enter is a command and executes it before the first command on the specified input device is executed.

## Device Manipulation Commands

The following sections describe the BL and DN commands.

### Display/Modify Buffer Limits (BL)

Purpose: Lets you examine and reset the buffer limits for an I/O device.

Syntax: BL, lu[,buf[,low[,high]]]

lu The logical unit number.

buf BU (buffered) or UN (unbuffered).

low The low buffer limit, > 0.

high The high buffer limit, low < high < available SAM.

Description:

This form of the command always displays the buffer limits for the specified LU:

```
FMGR : BL,lu
```

This form of the command always modifies the buffer limits:

```
FMGR : BL,lu,buf,low,high
```

The BL command always provides one of the following responses:

```
LU#lu buf BL=low,high AC=accum
```

```
LU # lu UNASSIGNED.
```

lu The logical unit number.

buf BU or UN.

low,high The actual limits applied.

accum The current buffer accumulation.

If buf=BU, all output to the named LU is directed to an I/O buffer in System Available Memory (SAM). The LU (peripheral device) retrieves the output from SAM and completes the output processing at its own rate. This allows a program to continue without waiting for the output processing to complete.

To prevent having all of SAM monopolized by slow devices, the system limits the accumulation (AC) of buffer requests. When the accumulation for a buffered device exceeds the high limit, any program that attempts I/O to the device is suspended, and not resumed until the accumulation drops below the low limit.

Low and high buffer limits are truncated to multiples of 16 words. The low limit may never exceed the high limit, and the high limit may never exceed 6112 words. If you specify a high limit that is larger than the maximum, the system reduces the specification to the maximum allowed. The low limit is increased, if necessary, so that it is within 2032 words of the high limit. If you specify low=high=0, buffer limit constraints are not applied to the named LU.

The buffer limits that you set for a device depend upon several factors:

- The output speed of the device.
- The rate of output of your programs.
- The amount of memory available for buffering.
- The extent to which you can afford to have your programs go into I/O suspension.

Whenever you reboot, the buffer limits revert to those set at system generation.

## BL Command Examples

### Example 1: Display buffer limit

```
BL,6
```

```
LU# 6 UN BL=768, 1056 AC= 0
```

### Example 2: Modify buffer limits.

```
BL,6,BU,200,800
```

```
LU# 6 BU BL=192, 800 AC= 0 (Limits are truncated to multiples of 16)
```

## **Make Device Unavailable (DN)**

**Purpose:** Declares an LU or device down and unavailable for use by the system.

**Syntax:** DN, lu

lu            The device LU number.

**Description:**

When you down an I/O device (LU), only that specified device becomes unavailable. Others that use the device's I/O channel are unaffected. The I/O device remains unavailable until you bring it back up with the UP command. Equipment problems or normal maintenance (for example, changing paper or ribbon on a printer) may necessitate taking down a device.

## **DN Command Example**

For example, to take down LU 8, simply enter the command as follows:

```
FMGR : DN, 8
```



## Other FMGR Commands

Table A-7 shows other FMGR commands and their CI equivalents.

**Table A-7. FMGR/CI Commands**

FMGR	CI
AS	AS
BR	BR
CN	CN
DS	IO
GP	GO
IO	IO
IT	AT
OF	OF
ON	AT
PL	WH
PR	PR
PS	PS
RP	RP
SS	SS
SZ	SZ
TM	TM
TO	TO
UL	UL
UP	UP
VS	VS
WS	WS
XQ	XQ



## COMND

The COMND program is a compact version of FMGR that only supports a subset of FMGR commands as listed in Table A-8 below. Reference Table A-7 for CI equivalents.

**Table A-8. COMND Commands**

Command:
BL
CN
IO
PL
TM
TO
IT
ON
EX

## 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 type 6 program file as in the offline version. The offline version for RTE-A runs in a memory-based RTE-A 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-A User's Manual* for command file usage.)

## Loading the program

With RTE-A software, the relocatables EXER.REL and EXER1.REL are supplied, along with link command files #EXER and #EXER1. The following are the contents of the two command files.

```
* #exer 24398-17016 rev.5010 <881110.1451>
* Load command file for RTE-6 and RTE-A disk exercizer program.
* 101288 find pascal_err.rel from /libraries
IF,A,PC,31,31
li $dtclb
if 6 li $dsclb
* re pascal_err.rel
re pascal_err.rel::libraries
li pascal.lib
re %exer
en

* #exer1 24398-17015 rev.5010 <881111.0959>
* Load command file for child disk exercizer program.
* 101288 find pascal_err.rel from /libraries
IF,A,PC,31,31
li $dtclb
* re pascal_err.rel
re pascal_err.rel::libraries
li pascal.lib
re %exer1
en
```

Note the required library files; on some revisions of the Primary System these files may be under /RTE\_A instead of /LIBRARIES. The supplied command file, RTEA2.CMD, which loads all of the RTE programs, also loads EXER and EXER1 automatically.

## 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 or 26B (disk or tape). 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 Primary system:

CS/80 EXTERNAL EXERCISER -- Rev. 5020 02-07-90

```
***** CS80 LU s *****
Sel Code  LU #   HPIB Addr   Unit
=====  =====
          26    12       1         0
          41    41       7         0
          42    42       7         0
          43    43       7         0
          61    61       7         1

          27     9       1         0
          16    16       0         0
          17    17       0         0
          18    18       0         0
          19    19       0         0
          20    20       0         0
          21    21       0         0
          22    22       0         0
          23    23       0         0
          24    24       0         1
          25    25       0         0
          26    26       0         0
          29    29       0         1
          30    30       0         1
          31    31       0         1
          36    36       5         0
          37    37       5         1
          50    50       2         0
          51    51       2         0
          52    52       2         0
          53    53       2         1
          54    54       2         1
```

Input DRIVE LU?

---

**Note** EXER only looks for LUs up to 63. For DataPair systems using LU numbers greater than 63 (which is recommended for DataPair systems) EXER will not find them. You will not be able to run EXER successfully in this case.

---

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 "16" as the LU. Since EXER does not care about LUs, you could use *any* LU 16–26, or 29–31 since they all correspond to the same physical disk drive (HPIB address 0, on select code 27B).

The next thing EXER does is attempt to identify the device type attached, with the result appearing as follows:

```
Input DRIVE LU? 16
```

```
LU 16 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><u>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 16 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 16 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 16 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 18 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 19 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 microprocessor. This may be useful information for field service personnel for troubleshooting purposes.

READ REVISION NUMBER UTILITY  
 LU 16 is a 7914

Current unit = 0

Part number	Revision number
-----	-----
1	5 - 0
2	5 - 1
3	5 - 0
4	5 - 0
5	5 - 0
6	5 - 0

\*NOTE These part numbers refer to the individual firmware ROMs on the processor card.

**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.)

## EXER and Cache Disks (HP 793xXP)

RTE-A supports the Cache version of the HP 793x family of CS/80 disks. If you have a Cache version drive, EXER determines this when it ID's the drive during the initial DESCRIBE command, and will allow access to several additional commands.

These additional commands allow the user to check certain statistics regarding the disk controller cache. They also allow the user to turn caching on and off. This can be useful in determining whether a particular application is really benefitting from the presence of the cache disk. Typical HP 1000 systems *do not* benefit from the cache, they may even slow down overall throughput because of the additional overhead required for the caching feature. Consult your local HP service representative for more information regarding disk caching.

## EXER and SubSet 80 Disks

SubSet 80 refers to a group of disk drives that are a “subset” of the CS/80 family. These include the HP 9133D/H/L and the HP 9153B/C (including the internal disk in the HP 12122A). As far as RTE-A is concerned, these drives are treated just like CS/80 disks. The EXER utility *does not* support these drives. Most commands will return an “Illegal Opcode error”.

# Index

---

## A

Absolute Binary to Memory Image utility (AB2MI), 1-3, 7-1  
break detection, 7-2  
calling AB2MI, 7-1  
error messages, 7-2  
operation, 7-2  
amount of disk space used by owners, 4-20  
arbitrary code, 7-1  
ARSTR physical restore utility, 1-1, 3-18  
ABort, 3-19  
break detection, 3-28  
calling ARSTR, 3-18  
checking the disks, 3-24  
command examples, 3-24  
commands, 3-18  
disk locking, 3-26  
ENd, 3-19  
error messages, 3-30  
example, 3-29  
EXit, 3-19  
HElp, 3-20  
LH List Header, 3-20  
LL List Device, 3-20  
loading ARSTR, 3-28  
offline system, 3-28  
order of disk restoration, 3-26  
RE command options, 3-22  
REstore, 3-21  
restore operation, 3-24  
restoring the data, 3-27  
RW Rewind, 3-23  
save definition records, 3-26  
sectors per track must match, 3-25  
TApe LU, 3-23  
total number of tracks may differ, 3-25  
UE User Error Handling, 3-23  
unit restore, 3-26  
verification, 3-27  
ASAVE physical backup utility, 1-1, 3-3  
ABort, 3-5  
break detection, 3-13  
calling ASAVE, 3-3  
command options, 3-9  
commands, 3-4  
data record, 3-15  
disk locking, 3-10  
ENd, 3-5  
end-of-data record, 3-16  
error messages, 3-17  
example, 3-16  
EXit, 3-5  
header records, 3-14

HElp, 3-5  
LH List Header, 3-5  
LL List Device, 3-5  
loading ASAVE, 3-13  
RW Rewind, 3-6  
SA command options, 3-6  
SAve, 3-6  
save definition record, 3-15  
save file records, 3-11  
save operation, 3-10  
saving disks to tape, 3-11  
tape and file formats, 3-13  
TApe LU, 3-8  
tape positioning, 3-11  
tape records, 3-14  
TTitle, 3-8  
UE User Error Handling, 3-9  
unit save LU checking, 3-10  
verification, 3-12

## B

backing up disk LUs, 3-1, 3-3  
backup and file interchange utilities, 1-1, 2-1, 2-53  
File Copy (FC), 1-1, 2-1, 2-3, 2-109  
File Storage to Tape (FST), 1-1, 2-1, 2-3, 2-4  
Logical Interchange Format HP systems file copy (LIF), 1-1, 2-1, 2-3, 2-153  
Tape Filer (TF), 1-1, 2-1, 2-3  
backup utilities, 2-1  
ARSTR, 2-1, 3-18  
ASAVE, 2-1, 3-3  
COPYL, 2-1, 3-35  
FC, 2-1, 2-109  
FST, 2-1, 2-4  
TF, 2-1, 2-53  
base page links in source BOOTEX, 7-7  
bit map/free space table, 4-22  
bootable system installation utility (FPUT), 7-8  
FPUT operation, 7-9  
running FPUT, 7-8  
BOOTEX file, 7-3, A-10, A-28  
initializing, 7-3  
installing, 7-8  
VCP/loader ROM, 7-3  
buffer limits, displaying and modifying, A-64

## C

CALLM utility, 1-4, 11-1  
.include directive, 11-7  
invoking the CALLM utility, 11-6  
CALLS utility, 1-4, 11-1  
catalog file, 11-2

- directives, 11-2
- index file, 11-5
- invoking the CALLS facility, 11-1
- online help facility, 11-1
- relating topics to other topics, 11-3
- carriage control when printing, 8-5, 8-8
- checking consistency on a CI file system disk LU, 4-22
- CI directory, newly built, 4-3
- COMND program, 1-4, 3-28, A-64
  - commands, A-70
- compacting files, 4-11
- comparing and identifying differences in input files, 6-9
- concatenate many files into one utility (MERGE), 6-1
  - break detection, 6-3
  - calling MERGE, 6-1
  - MERGE examples, 6-3
  - MERGE operations, 6-2
  - MERGE options, 6-2
- concatenating a program and subroutines, 6-1
- continuous streaming mode, 3-2, 7-10
- control file, A-51
- converting the directory structure of an FMGR cartridge, 4-2
- Copy System utility (CSYS), 1-3, 7-10
  - calling CSYS, 7-10
  - error messages, 7-13
  - examples, 7-12
  - loading CSYS, 7-13
  - operation, 7-11
- copying memory image files from CS/80 disk to cartridge tape, 7-10
- COPYL. *See* disk to disk copy utility
- CRETS routine, A-34
- CS/80 cartridge tape drive, 3-2, 7-10
- CS/80 definitions, 5-6
- CS/80 disk formatting utility (FORMC), 5-7
  - ABort, ENd, and EXit commands, 5-9
  - break detection, 5-8
  - calling FORMC, 5-7
  - command execution, 5-8
  - converting spares during tape formatting, 5-16
  - data loss during sparing, 5-18
  - default record interleave factor, 5-12
  - device driver status, 5-8
  - disk formatting information, 5-12
  - disk formatting process, 5-13
  - dismounting LUs, 5-13
  - error messages, 5-22
  - FormaT command, 5-10
  - formatting a system disk, 5-13
  - formatting operation, 5-11
  - FORMC commands, 5-9
  - help command, 5-10
  - inserting the cartridge, 5-15
  - loading FORMC, 5-21
  - re-certification and spare conversion, 5-14
  - readying a tape for verification, 5-19

- SPare command, 5-10
- sparing defective blocks, 5-17
- sparing operation, 5-16
- specifying a start track or block for verification, 5-20
- specifying an interleave factor, 5-12
- specifying the number of tracks or blocks to verify, 5-20
- specifying the track when sparing, 5-17
- tape certifying and skip sparing, 5-14
- tape formatting information, 5-14
- tape formatting process, 5-15
- verification process, 5-21
- Verify command, 5-11
- verifying an entire disk LU or tape, 5-19
- verifying operation, 5-19
- CS/80 Exerciser utility, 1-4
  - See also* EXER utility
- CSYS. *See* Copy System

## D

- data buffering and transfer, 7-10
- device manipulation (FMGR), A-1, A-64
  - BL command examples, A-67
  - BL display/modify buffer limits, A-66
  - calling FMGR and COMND, A-64
  - device manipulation commands, A-66
  - DN command example, A-68
  - DN make device unavailable, A-68
  - other COMND commands, A-70
  - other FMGR commands, A-69
  - taking down a device, A-64
- directories, FMGR, A-1
- disk blocks, 5-2
  - address, 5-2
  - CS/80 definition, 5-6
  - postamble, 5-2
  - preamble, 5-2
  - read/processing time calculated for interleave factor, 5-4
  - sparing defective ones, 5-17
  - status bits, 5-3
  - testing, 5-2
  - user data area, 5-2
- disk caching, 3-2, 7-10
- disk formatting utility (FORMF), 5-23
  - ABort, ENd, EXit commands, 5-25
  - break detection, 5-24
  - calling FORMF, 5-23
  - command execution, 5-24
  - device driver status, 5-24
  - error messages, 5-29
  - FormaT command, 5-26
  - formatting operation, 5-27
  - formatting process, 5-28
  - FORMF commands, 5-25
  - HElP command, 5-26
  - specifying an interleave factor, 5-27
  - supported disk drives, 5-23

- Verify command, 5-26
- verifying operation, 5-28
- disk manipulation (FMGR), A-1, A-7, A-15
  - cartridge directory, A-10
  - cartridge file directory, A-10
  - cartridge initialization, A-11
  - cartridges and cartridge reference numbers, A-8
  - changing the master security code, A-12
  - CL command example, A-17
  - CL list cartridge directory, A-16
  - CO command examples, A-20
  - CO command option examples, A-19
  - CO command options, A-18
  - CO command termination, A-22
  - CO copy files, A-17
  - configuration of logical units/cartridges, A-8
  - DC command error handling, A-23
  - DC dismount file cartridge, A-23
  - disk logical units and LU numbers, A-7
  - disk manipulation commands, A-15
  - DL command examples, A-25
  - DL list file directory, A-24
  - IN command error handling, A-30
  - IN initialize file cartridge, A-27
  - logical vs physical, A-7
  - MC command error handling, A-31
  - MC mount file cartridge, A-30
  - mounting and dismounting cartridges, A-9
  - new cartridge initialization, A-28
  - packing a file cartridge, A-14
  - PK command error handling, A-32
  - PK pack file cartridge, A-31
  - purging files on a cartridge during re-initialization, A-13
  - re-initializing a cartridge, A-12
  - transferring files between disk cartridges, A-15
- disk packing process, 4-6
- disk packs, A-8
- disk-to-disk copy utility (COPYL), 3-1, 3-35
  - calling COPYL, 3-35
  - copy operation, 3-36
  - error messages, 3-36
- display CPU usage utility (METER), 1-3, 9-1
  - calling METER, 9-2
  - commands, 9-3
  - loading METER, 9-4
  - output example, 9-4
  - sorting and displaying process information, 9-1
- DRSTR physical restore utility, 1-1, 3-52
  - ABort, 3-53
  - break detection, 3-62
  - calling DRSTR, 3-52
  - checking the disks, 3-58
  - commands, 3-52
  - disk locking, 3-59
  - ENd, 3-53
  - error messages, 3-62
  - EXit, 3-53
  - file-by-file (LU) restore, 3-55
  - header records, 3-60

- HElp, 3-53
- LH List Header, 3-54
- LL List Device, 3-54
- loading DRSTR, 3-62
- MEdia, 3-55
- order of disk restoration, 3-59
- RE command options, 3-56
- REstore, 3-55
- restore operation, 3-57
- restoring fixed disks from removable disks, 3-60
- restoring the data, 3-61
- sectors per track must match, 3-58
- total number of tracks may differ, 3-58
- UE User Error handling, 3-57
- UNit restore, 3-57
- unit restore, 3-56, 3-59
- verification, 3-61
- VERify data, 3-57
- DSAVE Physical Backup Utility, 1-1
- DSAVE physical backup utility, 3-37
  - ABort, 3-39
  - break detection, 3-47
  - calling DSAVE, 3-37
  - commands, 3-38
  - disk locking, 3-45
  - ENd, 3-39
  - error messages, 3-51
  - EXit, 3-39
  - header records, 3-49
  - HElp, 3-39
  - LH List Header, 3-39
  - LL List Device, 3-41
  - loading DSAVE, 3-48
  - MEdia, 3-41
  - no locking, 3-43
  - removable disk media and file formats, 3-49
  - removable disk overwriting, 3-45
  - SA command options, 3-42
  - SAve, 3-42
  - save file records, 3-46
  - save operation, 3-44
  - saving to the removable disk, 3-46
  - TItle, 3-43
  - UE User Error handling, 3-44
  - UNit save, 3-43
  - unit save LU checking, 3-45
  - verification, 3-47
  - VERify, 3-43

## E

- echoing commands, A-5, A-6
- enlarge free space areas on disks, 4-11
- Error Rate Test utility (ERTSH), 1-2, 5-44, 5-50, 5-54
  - BP Build Test Pattern command, 5-57
  - break detection, 5-66
  - commands, 5-56
  - CY Define Cylinder Range command, 5-57
  - error interpretation, 5-68

- error messages, 5-71
- ERTSH structure, 5-54
- EXit command, 5-58
- HD Define Disk Head Range command, 5-58
- help command, 5-57
- initializing and testing a new disk, 5-63
- installation procedure summary, 5-67
- LL Extra Log Device command, 5-58
- LU Selecting Disk LU to Test command, 5-59
- P# Set Test Pass Count command, 5-59
- RO Execute Read-Only Error Rate Test command, 5-60
- SC Define Disk Sector Range command, 5-60
- specialized use of ERTSH, 5-66
- SS Spare Sectors command, 5-61
- testing disks formatted with FORMF, 5-63
- testing the functionality of a disk and its controller, 5-54
- TP Test Pattern command, 5-61
- typical use of ERTSH, 5-63
- using ERTSH, 5-63
- WR Write/Read Error Rate Test command, 5-62
- ERTSH. *See* Error Rate Test utility
- EXER utility, 1-4, B-1
  - error handling, B-9
  - EXER and cache disks (793xXP), B-9
  - EXER and CS80 tape drives, B-9
  - EXER and subset 80 disks, B-10
  - loading the program, B-2
  - selected command descriptions, B-6
    - CHANGE LU, B-6
    - DESCRIBE, B-6
    - ERT LOG, B-6
    - FAULT LOG, B-7
    - INPUT, B-8
    - OUTPUT, B-9
    - REV, B-8
    - RO ERT, B-8
    - RUN LOG, B-7
    - TABLES, B-7
  - using the Exerciser, B-2
- extended record converter utility (OLDRE), 6-4
  - calling OLDRE, 6-5
  - error messages, 6-8
  - extended records, 6-4
  - FORTTRAN code, 6-7
  - FORTTRAN code processing, 6-5
  - Macro code, 6-7
  - Macro code processing, 6-5
  - operation, 6-5
  - Pascal code, 6-7
  - program restrictions, 6-7
  - translation results, 6-6
- extended record, definition of, 6-4
- extended system available memory (XSAM), 9-5
- extents, file, 2-22, 2-39, 2-56, 2-118, 2-132, 4-6

## F

- FC. *See* File Copy
- file backup utilities, 2-1
- file compacting and disk pack utility (MPACK), 4-11
  - calling MPACK, 4-11
  - compacting options, 4-12
  - examples, 4-16
  - logging option, 4-16
  - MPACK options, 4-11
  - packing options, 4-15
- file comparison utility (SCOM), 6-9
  - calling SCOM, 6-9
  - compare operation, 6-13
  - error messages, 6-22
  - examples, 6-15
  - options, 6-10
  - returned values, 6-14
  - status interrogation, 6-14
- File Copy (FC), 1-1, 2-1, 2-3, 2-109
  - ABort command, 2-113
  - brief, full status display format, 2-117
  - calling FC, 2-109
  - cartridge lock, open, 2-119, 2-132
  - CF name comment file command, 2-113
  - CL Cartridge List command, 2-113
  - clear destination disk, 2-118
  - CO command examples, 2-120
  - CO command options, 2-117
  - CO command source and destination parameters, 2-116
  - command summary function (?), 2-112
  - COpy command, 2-114
  - copy single volume of multi-volume tape set, 2-119
  - DEfault command, 2-123
  - destination disk handling, 2-131
  - display required tape length, 2-119
  - DL Directory List command, 2-124
  - ECHO command, 2-126
  - eliminate extents, 2-118
  - error handling in transfer files, 2-135
  - error messages, 2-137
  - EXit command, 2-126
  - extents, file, 2-118, 2-131
  - FC commands, 2-110
  - globals used in transfer files, 2-133
  - GRoup CO commands, 2-126
  - ignore data errors, 2-118
  - keep tape online, 2-119, 2-127, 2-130
  - LC List Comment files command, 2-127
  - LH List Header files command, 2-127
  - LL List device command, 2-127
  - loading FC, 2-133
  - master security code, 2-115

- performance considerations, file copy operations, 2-131
- purge source file, 2-119
- recover unused space, 2-120
- replace duplicate files, 2-118
- Scratch area definition command, 2-128
- SKip volume option in multi-volume read, 2-130
- tape directory list format, 2-125
- tape handling, 2-129
- TItle command, 2-128
- TTransfer command (to/from command file), 2-129
- verify transferred data integrity, 2-120
- file interchange utilities, 2-3
  - FC, 2-3, 2-109
  - FST, 2-3, 2-4
  - LIF, 2-3, 2-153
  - TF, 2-1, 2-3, 2-53
- File Manager (FMGR) control, 1-4, A-1
  - ?? command examples, A-3
  - ?? explain error codes, A-3
  - commands, A-2
  - COMND program, 1-4
  - errors, A-2
  - list device, A-2
  - LL command examples, A-4
  - LL display or change list device, A-4
  - LO change log device, A-5
  - log device, A-2
  - severity code, A-2
  - SV command examples, A-6
  - SV display or set severity code, A-5
- file manipulation, A-1, A-33
  - accessing a disk file, A-34
  - commands, A-36
  - creating a file, A-34
  - listing the contents of a file, A-36
  - purging files, A-35
  - records and file types, A-33
  - renaming files, A-36
  - scratch files, A-34
  - storing data on a device or new file, A-36
- file manipulation (FMGR), A-36
  - CR command error handling, A-39
  - CR command examples, A-39
  - CR create a file, A-37
  - DU command examples, A-41
  - DU dump data to a device or existing file, A-40
  - LI command examples, A-43
  - LI list contents of a file, A-42
  - PU command examples, A-44
  - PU purge a file, A-44
  - RN command examples, A-45
  - RN rename a file, A-45
  - ST command examples, A-48
  - ST store data on a device or new file, A-45
- file manipulation utilities, 1-3, 6-1
  - compare files (SCOM), 1-3, 6-9
  - concatenate files (MERGE), 1-3, 6-1
  - extended record converter (OLDRE), 1-3, 6-4
  - file ownership reporting, 4-20
  - file renaming during conversion of FMGR cartridge directory structure, 4-3
- File Storage to Tape utility (FST), 2-4
  - append option (A), 2-20
  - appending data, 2-33
  - backing up using file masking, 2-25
  - Backup Bits, 2-21
  - BAckup command, 2-8
    - preserving FMGR security codes, 2-8
  - brief option (B), 2-21
  - building a new directory file, 2-36
  - calling FST, 2-5
  - clear option (C), 2-21
  - command options summary, 2-20
  - command stack, 2-8
  - commands, 2-7
  - consecutive backups, 2-33
  - D, K, N, and S qualifiers, 2-25
  - delta backups, 2-29
  - DF Directory File command, 2-9
  - disk directory file, 2-37
  - DL List Directory command, 2-9
  - duplicate option (D), 2-21
  - end-of-file position ignored with W option, 2-24
  - error handling, 2-42
  - error messages and warnings, 2-43
  - EXit command, 2-10
  - extents, file, 2-22, 2-39
  - faulty option (F), 2-21
  - file masking and renaming, 2-25
  - FST.RC start-up file, 2-5, 2-6
  - full backups, 2-29
  - GO begin backup/restore command, 2-10
  - HElp command, 2-10
  - incremental backup, 2-29
  - inhibit option (I), 2-21
  - installing FST, 2-42
  - keep option (K), 2-21, 2-34
  - LC List Comment file command, 2-10
  - LH List Header command, 2-11
  - LI List selected files sommand, 2-11
  - LL select log device/file command, 2-11
  - LN List Non-selected files command, 2-12
  - lock option (L), 2-22
  - mindir option (M), 2-22
  - MT specify tape LU command, 2-12
  - multiple reels, 2-33
  - NExt command, 2-13
  - normal option (N), 2-22
  - options, 2-19
  - original option (O), 2-22
  - POsition command, 2-13
  - PREvious command, 2-13
  - purge option (P), 2-23
  - quiet option (Q), 2-23
  - recommended system usage, 2-39
  - replacing reserved characters in FMGR file names, 2-38
  - REstore command, 2-14



- restoring files from overwritten tape, 2-21, 2-36
  - restoring from incremental backups, 2-31
  - restoring using file masking, 2-27
  - RUn command, 2-15
  - rwndoff option (R), 2-23
  - SC Select Comment file command, 2-15
  - SD Set tape Density command, 2-15
  - SEcure command, 2-16
  - Shareable EMA (SHEMA), 2-38, 2-42
  - SHow user selected states command, 2-16
  - srchapp option (S), 2-23
  - streaming during verify pass, 2-24
  - streaming may not occur using RE command, 2-14
  - streaming mode, 2-14, 2-24, 2-39
  - tape format, 2-40
  - tape loading, 2-34
  - tape positioning on overwritten tapes, 2-36
  - TAr command, 2-16
  - TF compatibility, 2-34
  - TItle command, 2-17
  - TRansfer to command file command, 2-18
  - UNIX TAR Format, 2-16
  - UNselect command, 2-18
  - update option (U), 2-23, 2-32
  - using GRoup commands for large restores, 2-14
  - verify option (V), 2-24
  - whole option (W), 2-24
  - yes option (Y), 2-24
  - Z option, 2-24
  - File System Conversion utility (FSCON), 4-2
    - calling FSCON, 4-2
    - conversion process, 4-3
    - FSCON error messages, 4-5
    - requirements for conversion, 4-2
  - File System Pack utility (FPACK), 4-6
    - calling FPACK, 4-6
    - moving directories, 4-8
    - moving files, 4-10
    - moving subdirectories, 4-9
    - packing process, 4-6
  - file system utilities, 1-2, 4-1
    - file compacting and disk pack (MPACK), 1-2, 4-11
    - File System Conversion (FSCON), 1-2, 4-2
    - File System Pack (FPACK), 1-2
    - file system pack (FPACK), 4-6
    - file system verification (FVERI), 1-2, 4-22
    - report disk Free Space (FREES), 1-2, 4-17
    - report File Space by Owner (FOWN), 1-2
    - report file space by owner (FOWN), 4-20
  - file system verification utility (FVERI), 4-22
    - error messages, 4-25
    - error recovery, 4-24
    - operating instructions, 4-24
  - files, FMGR, A-1
  - files, printing. *See* PRINT Utility
  - fill number, 5-4
  - FMGR. *See* File Manager (FMGR) control
  - FMGR cartridge conversion, 4-2
  - FORMA. *See* initialize and spare utility
  - formatting information, general, 5-2
  - formatting utilities, 1-2, 5-1
    - CS/80 disk formatting (FORMC), 1-2, 5-7
    - disk formatting (FORMF), 1-2, 5-23
    - Error Rate Test (ERTSH), 1-2, 5-54
    - initialize and spare (FORMA), 1-2, 5-44
    - offline disk formatting and initialization (FORMT), 1-2, 5-31
  - FORMC. *See* CS/80 disk formatting utility
  - FORMF. *See* disk formatting utility
  - FORMT. *See* offline disk formatting and initialization
  - FOWN. *See* report file space by owner
  - FPACK. *See* file system pack
  - FPUT. *See* bootable system installation
  - free space table on an FMGR cartridge, 4-2
  - free space table/bit map, 4-22
  - FREES. *See* file system utilities
  - FSCON. *See* file system conversion
  - FST. *See* file storage to tape utility
  - FVERI. *See* file system verification utility
- ## G
- G-type global parameters, A-5, A-51, A-61
  - GRoup and User Management Program (GRUMP), 10-1, 10-3
- ## H
- HP computer systems file copy utility (LIF), 2-153
    - Calling LIF, 2-154
    - CO command, 2-156
    - DL Directory List command, 2-157
    - EXit command, 2-158
    - HElP command, 2-158
    - INitialize command, 2-158
    - LIF commands, 2-154
    - LIF error handling, 2-163
    - LlIst command, 2-159
    - LL set Logical List device command, 2-159
    - MC Mount Cartridge command, 2-159
    - naming conventions, 2-153
    - PK Pack Cartridge command, 2-160
    - PURge command, 2-160
    - RN rename command, 2-160
    - STore command, 2-161
    - SV severity command, 2-162
    - TR TRansfer control command, 2-162
  - HP Virtual Control Panel (VCP), 7-3, 7-10
    - bootstring for SCSI or CS/80 disk, 7-8
- ## I
- increase disk free space, 4-6
  - initialize and spare utility (FORMA), 5-44
    - byte/index address, 5-48
    - calling FORMA, 5-47
    - converting from FORMF to FORMA, 5-50

- definitions, 5-45
- disk error decoding, 5-53
- error handling, 5-51
- error messages, 5-51
- ERTSH works with FORMA, 5-44, 5-50
- FORMA commands, 5-47
- help command, 5-47
- IM Initialize Media command, 5-47
- installing a new internal hard disk, 5-50
- SS command options, 5-49
- SS Spare Sector command, 5-48
- typical track format, 5-46
- initialize BOOTEX file utility (INSTL), 7-3
  - calling INSTL, 7-4
  - error messages, 7-7
  - operation, 7-6
  - overview, 7-3
- initializing and sparing hard disks, 5-4
- initializing and testing a new disk, 5-63
- installing bootable systems and diagnostics, 7-8
- INSTL. *See* Initialize BOOTEX File
- interleave factor calculation, 5-4
- introduction to RTE-A utilities, 1-1

## K

- keyword indexed help utilities, 1-4
  - See also* CALLS and CALLM utilities
- KILLSSES. *See* terminate a session

## L

- library of binary relocatable files, forming a, 6-1
- LIF. *See* HP Computer Systems File Copy
- LU access tables, 10-6

## M

- M1KSS monitor program, 8-3
- manipulating files on FMGR directories, A-1
- master security code, A-12, A-44
- memory image format, 7-1
- Memory Image to Absolute Binary (MI2AB), 1-3, 7-14
  - break detection, 7-14
  - calling MI2AB, 7-14
  - error messages, 7-15
- MERGE. *See* concatenate many files into one
- METER. *See* System Status Utilities
- mini-cartridge bootstrap loader, 7-14
- modify or list session LU access tables (SESLU), 10-6
  - calling SESLU, 10-6
  - examples, 10-7
  - loading SESLU, 10-6
  - protection, 10-6
  - returned values, 10-7
- MPACK. *See* file compacting and disk pack
- multiuser accounting utilities, 1-4, 10-1

- modify or list session bit maps (SESLU), 1-4, 10-6
- reset multiuser accounting information (RINFO), 1-4, 10-1
- show multiuser accounting information (SINFO), 1-4, 10-3
- terminate a session (KILLSSES), 1-4, 10-4

## N

- NOTIFY utility, 1-3, 8-1
  - and M1KSS monitor program, 8-3
  - calling the NOTIFY utility, 8-1
  - defining aliases for notification, 8-2
    - alias definition format, 8-3
    - alternate logons for a user, 8-3
    - group distribution lists, 8-3
    - hosts in a DS network, 8-3
  - installed by Mail/1000, 8-1
  - message format, 8-1
  - runstring syntax, 8-1
  - sending a message, 8-1
  - turning notification on or off, 8-2
  - using the NOTIFY utility, 8-1

## O

- offline disk formatting and initialization utility (FORMT), 5-31
  - calling FORMT, 5-31
  - ENd command, 5-32
  - error messages, 5-42
  - FOrmat command, 5-32
  - formatting operation, 5-35
  - formatting process, 5-36
  - FORMT commands, 5-32
  - FORMT functions, 5-31
  - generating a FORMT system, 5-31, 5-34
  - help command, 5-33
  - INitialize command, 5-33
  - initializing operation, 5-37
  - REformat command, 5-33
  - reformatting operation, 5-38
  - sector interleaving, 5-36
  - SPare command, 5-34
  - sparing operation, 5-40
  - VERify command, 5-34
  - verify operation, 5-41
- offline restoration, 2-1
- offline system, BUILD utility, 3-28
- offline system (ASAVE, ARSTR), 3-28
- OLDRE. *See* extended record converter
- online backup and restoration to/from disks, 2-1

## P

- P-type global parameters, A-53
- packing a disk, 4-11
- packing files on a volume, 4-6
- physical disk image backup utilities, 1-1, 2-1, 3-1

- compatibility among disks, 3-1
- compatibility with other utilities, 3-1
- disk-to-disk copy utility (COPYL), 1-1, 3-1, 3-35
- restore utility (ARSTR), 1-1, 3-1, 3-18
- restore utility (DRSTR), 1-1, 3-1, 3-52
- save utility (ASAVE), 1-1, 3-1, 3-3
- save utility (DSAVE), 1-1, 3-1, 3-37
- using the utilities, 3-1
- PRINT utility, 1-3, 8-1
  - &FFL module, 8-16
  - &FFL variables, 8-15
  - %FFL Module, 8-8
  - calling PRINT, 8-5
  - carriage control, 8-7, 8-8
  - examples, 8-13
  - form feeds, 8-8
  - form feeds and serial drivers, 8-8
  - loading PRINT, 8-15
  - messages, 8-12
  - operation, 8-11
  - options, 8-5
  - printing files, 8-5
  - using the PRINT utility, 8-5
- PRTN call, 10-2, 10-3, 10-4, 10-7, A-54
- PURGE routine, A-34

## R

- rearranging files on a volume, 4-6
- report disk Free Space (FREES), 4-17
- report file space by owner (FOWN), 4-20
  - calling FOWN, 4-20
  - FOWN examples, 4-20
- reset multiuser accounting information (RINFO), 10-1
  - calling RINFO, 10-1
  - loading RINFO, 10-2
  - returned values, 10-2
  - RINFO protection, 10-2
- RINFO. *See* reset multiuser accounting information
- RMPAR subroutine, A-54

## S

- SAM. *See* System Available Memory
- SCOM. *See* file comparison
- Security/1000, 10-2, 10-4, 10-6
- Serial Port Analyzer (SPORT), 9-8
  - calling SPORT, 9-8
  - including SPORT in a user program, 9-10
  - loading SPORT, 9-11
  - SPORT examples, 9-9
  - SPORT operation, 9-8
- SESLU. *See* modify or list session LU access tables
- show multiuser accounting information (SINFO), 10-3
  - calling SINFO, 10-3
  - loading SINFO, 10-3
  - returned values, 10-3

- Show the Status of System Available Memory (SAM), 1-3
- SINFO. *See* show multiuser accounting information
- sparing vs skipping, 5-4
- spool files closed and released with KILLSSES, 10-4
- SPORT. *See* Serial Port Analyzer
- streaming mode, 3-2, 7-10
- subfile marks (FMGR ST command), A-47
- System Available Memory (SAM), 9-5
- System Available Memory (SAM) status utility, 9-5
  - calling SAM, 9-5
  - loading SAM, 9-7
  - returned values, 9-6
  - running SAM with AL, 9-6
  - running SAM without AL, 9-5
- system setup utilities, 1-3, 7-1
  - Absolute Binary to Memory Image (AB2MI), 1-3, 7-1
  - bootable system installation (FPUT), 1-3, 7-8
  - Copy System (CSYS), 1-3, 7-10
  - initialize BOOTEX (INSTL), 1-3, 7-3
  - Memory Image to Absolute Binary (MI2AB), 1-3, 7-14
- system status utilities, 1-3, 9-1
  - display CPU usage (METER), 1-3, 9-1
  - Serial Port Analyzer (SPORT), 9-8
  - Serial Port analyzer (SPORT), 1-3
  - System Available Memory utility (SAM), 1-3, 9-5

## T

- Tape Filer (TF), 1-1, 2-1, 2-3, 2-53
  - access time for tape files, 2-86
  - alternatives to standard incremental backup, 2-91
  - B qualifier in incremental backup, 2-87
  - backup bits, 2-87, 2-89, 2-91
  - C option used in incremental backup, 2-87
  - calling TF, 2-54
  - CO command, 2-55
  - CO command options, 2-57
    - append to tape (A), 2-58
    - brief logging mode (B), 2-58
    - clear backup bit (C), 2-58
    - ignore errors and file masks (I), 2-59
    - inhibit UNIX to FMP text file conversion (N), 2-59
    - keep tape online (K), 2-59
    - replace duplicate files (D), 2-59
    - Unix compatibility (X), 2-60
    - update (U), 2-59
    - verify files copied (V), 2-59
    - yes option (Y), 2-60
  - CO command source and destination parameters, 2-56
  - copy example without subdirectories, 2-60
  - copy examples using DS, 2-74
  - copy examples with subdirectories, 2-66
  - copying files between FMP/UNIX, 2-100

- create time (tape), 2-86
- DEfault command, 2-77
- delta backups, 2-87
- directory creation and restore, 2-92
- directory file names (FMP/UNIX), 2-103
- disk full errors, 2-96
- DL Directory List command, 2-78
- duplicate files during restoring incremental backups, 2-90, 2-91
- EXit command, 2-81
- file access during backup/restore, 2-96
- file extents, 2-56
- file formats on FMP and UNIX, 2-99
- file properties, saving and restoring, 2-93
- file/tape compatibility, FMP/UNIX, 2-99, 2-108
- full backup, 2-87
- GRoup copy commands, 2-81
- HElp command, 2-82
- incremental backup, 2-87
- incremental backup procedure, 2-88
- installing TF, 2-108
- keep tape online, 2-78, 2-81, 2-82, 2-85
- LH List Header file, 2-82
- LL list device command, 2-83
- maintaining the system time, 2-86
- missing extents, 2-98
- missing time stamps, 2-86
- multi-tape backup/restore, 2-98
- multiple copies of the same backup, 2-89
- relation of DL command to CO command, 2-80
- restoring incremental backups, 2-89
- restoring older versions from incremental backup, 2-90
- sparse files, 2-56, 2-98
- system backup and restore, 2-94
- tape protection and the K (keep) option, 2-85
- TF commands, 2-54
- TF tape format, 2-107
- time stamps, 2-85
- TItle command, 2-83
- TRansfer command, 2-84
- UNIX compatibility, 2-99
- update time (tape), 2-86

- using TF with FC tapes, 2-97
- using TF with FMGR files, 2-96
- terminate a session (KILLSES), 10-4
  - calling KILLSES, 10-4
  - examples, 10-5
  - loading KILLSES, 10-4
  - protection, 10-4
  - returned values, 10-4
- track size of disks, 3-1
- tracks, sparing and skipping, 5-4
- transfer files, A-1, A-51
  - CA calculate global parameter, A-56
  - CA command examples, A-57
  - creating, A-51
  - DP command examples, A-58
  - DP display parameters, A-58
  - G-type global parameters, A-51
  - IF command examples, A-60
  - IF conditional skip, A-59
  - P-type global parameters, A-53
  - PA command example, A-61
  - PA pause and send message, A-61
  - SE command examples, A-62
  - SE set global parameter, A-61
  - TR and related commands, A-55
  - TR command examples, A-63
  - TR transfer control to a file or device, A-62
- translating extended relocatable record formats, 6-4, 6-6

## U

- user configuration file, 10-1, 10-3, 10-6
- user masks, 10-1, 10-3

## V

- validity of disk volume directories and tables, 4-22
- VCP/loader. *See* HP Virtual Control Panel (VCP)

## X

- XSAM, 9-5

