

May 1979

This document describes the procedures used to operate and manage a RSTS/E system. It includes information on software enhancements provided by RSTS/E V7.0.

RSTS/E V7.0
System Manager's Guide

Order No. AA-2762C-TC

SUPERSESSON/UPDATE INFORMATION: This manual contains information on RSTS/E V7.0.

OPERATING SYSTEM AND VERSION: RSTS/E V7.0

SOFTWARE VERSION: RSTS/E V7.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation • maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1975, 1977, 1979 Digital Equipment Corporation

The postage-paid READER'S COMMENTS form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	IAS
DECnet	MASSBUS
DECsystem-10	PDP
DECSYSTEM-20	RSTS
DECtape	RSX
DECUS	UNIBUS
DIBOL	VAX
DIGITAL	VMS
FOCAL	

Contents

	Page
Preface	<i>xiii</i>
Chapter 1 RSTS/E System Structure and System Management	
1.1 Disk Organization	1-1
1.1.1 File Structure	1-1
1.1.2 Disk Optimization	1-2
1.2 System Operation Concepts	1-3
1.3 System Management.	1-4
1.4 Privilege.	1-5
1.4.1 Privileged Capabilities	1-5
1.4.2 Privilege and System Operation.	1-6
1.4.3 Guidelines for Privileged Operation	1-8
Chapter 2 System Start Up, Shutdown, and Automatic Restart	
2.1 Starting Up RSTS/E: Bootstrapping RSTS/E into Memory.	2-1
2.1.1 Bootstrapping RSTS/E using a Hardware Bootstrap Loader	2-1
2.1.2 Bootstrapping RSTS/E after a System Halt	2-2
2.1.2.1 Requesting an Automatic Restart	2-2
2.1.2.2 Requesting an Initialization Option	2-3
2.1.3 Starting Time Sharing	2-3
2.2 Halting the RSTS/E System	2-4
2.3 Automatic Recovery and Restart Facilities	2-5
2.3.1 Catastrophic Errors and System Crashes	2-5
2.3.1.1 Privileged Account Programming Errors	2-5
2.3.1.2 Hardware Malfunctions	2-5
2.3.1.3 System Software Malfunctions.	2-6
2.3.2 Automatic Recovery from Catastrophic Errors	2-6
2.3.3 Automatic Restart Mode Initialization	2-7
Chapter 3 Controlling Time Sharing	
3.1 Controlling System Start Up - INIT	3-1
3.1.1 INIT Program Commands	3-2
3.1.2 Creation and Usage of Control Files.	3-3
3.1.2.1 Detaching and Attaching INIT.	3-7
3.1.2.2 Indirect Command Files.	3-8
3.1.3 Setting Terminal Characteristics - TTY.CMD.	3-9
3.1.4 Spooling and Operator Services - SPOOL.CMD.	3-9
3.1.5 Establishing Auxiliary Run-Time Systems - RTS.CMD	3-10
3.1.6 Defining CCL Commands - CCL.CMD	3-10
3.1.7 System Start-Up Control File - START.CTL	3-11
3.1.8 System Crash Control File - CRASH.CTL	3-12

3.2	Performing System Shutdown - SHUTUP	3-12
3.2.1	Set-up Dialogue Phase	3-14
3.2.2	Warning Message Phase	3-15
3.2.3	Initial Job Killing Phase	3-16
3.2.4	OPSER Shutdown Phase (Optional)	3-17
3.2.5	ERRCPY Shutdown Phase (Optional)	3-18
3.2.6	Final Job Killing Phase (Optional)	3-19
3.2.7	Unload and Remove Run-Time Systems (RTS) and Resident Libraries Phase	3-19
3.2.8	Swap File Removal Phase	3-19
3.2.9	Disk Dismount Phase	3-19
3.2.10	Final Shutdown Phase	3-19
3.2.11	SHUTUP Operation Examples	3-20
3.2.12	Notes on SHUTUP Operation	3-23

Chapter 4 Account Creation and Account Statistics

4.1	Creating and Deleting User Accounts - REACT	4-1
4.1.1	Creating Individual Accounts - ENTER Function	4-2
4.1.2	Deleting Accounts - DELETE Function.	4-3
4.1.3	Automatic Account Creation - STANDARD Function	4-4
4.2	Performing System Accounting Operations - MONEY	4-5

Chapter 5 Operator Services and Spooling

5.1	Overview of Operator Services	5-1
5.2	Operator Services Program - OPSER	5-5
5.2.1	OPSER Operator Commands	5-8
5.2.2	Message Levels and Formats	5-9
5.2.3	Valid Operator and On-Line Job Lists	5-11
5.2.4	Operator INTERRUPT Command	5-12
5.2.5	OPSER Start-Up Procedure	5-12
5.2.6	OPSER Action Under Various Start-Up Conditions	5-13
5.3	Queue Manager Program - QUEMAN	5-14
5.3.1	QUEMAN Start-Up Commands and Options	5-16
5.3.2	QUEMAN Interrupt Commands	5-17
5.3.3	QUEMAN Start-Up Procedure	5-20
5.3.4	QUEMAN Action Under Various Start-Up Conditions	5-20
5.3.5	QUEMAN Consistency Checking	5-20
5.4	Line Printer Spooling Program - SPOOL	5-22
5.4.1	SPOOL Start-Up Options	5-23
5.4.2	Line Printer Spooling	5-28
5.4.3	Keyboard Spooling.	5-28
5.4.4	Start-Up Error Processing	5-29
5.4.5	SPOOL Interrupt Commands.	5-30
5.4.6	SPOOL Start-Up Examples	5-31
5.4.6.1	Line Printer Start Up with All Defaults	5-31
5.4.6.2	Line Printer Start Up with Narrow Width	5-32
5.4.6.3	Keyboard Start Up on an LA36	5-33
5.4.6.4	Keyboard Start Up on an LA180.	5-34

5.4.7	Recovery from Line Printer Errors	5-35
5.4.8	Line Printer Output	5-36
5.4.9	Error Messages During User Output	5-37
5.4.10	Changing and Aligning Forms	5-37
5.5	Batch Processor Program - BATCH	5-38
5.5.1	BATCH Start-Up Options	5-40
5.5.2	BATCH Interrupt Commands	5-41
5.5.3	BATCH Start-Up Procedure	5-42
5.5.4	Operator Action Requests from BATCH.	5-42
5.6	Operator Communication Program - PLEASE	5-43
5.6.1	Running and Terminating PLEASE.	5-44
5.6.2	OPSER Commands through PLEASE.	5-44
5.6.3	PLEASE as a CCL Command	5-45
5.7	Terminating Operator Services and Spooling	5-46
5.7.1	OPSER Shutdown Levels.	5-46
5.7.2	OPSER Manual Shutdown Procedure.	5-46
5.8	BACKUP as an OPSER Controlled Program	5-46

Chapter 6 System Error Package

6.1	Use of the Error Logging Programs - ERRINT and ERRCPY	6-2
6.1.1	Error Logging Initialization - ERRINT	6-2
6.1.2	Examples of ERRINT Dialogue	6-3
6.1.3	Error Logging - ERRCPY	6-3
6.1.4	Description of the Error File (ERRLOG.FIL)	6-3
6.2	Displaying Errors - ERRDIS	6-4
6.2.1	Running ERRDIS	6-5
6.2.1.1	Help Report	6-6
6.2.2	Summary Report.	6-7
6.2.3	Bad Block Report	6-7
6.2.4	Full Report	6-7
6.2.4.1	User Description in Full Report	6-8
6.2.4.2	Disk Error Detailed Description	6-9
6.2.4.3	Nondisk Peripheral Device Error Detailed Description.	6-10
6.2.4.4	Nonperipheral Error Detailed Description.	6-10
6.2.5	PROGRAM LOST-SORRY Errors	6-11
6.2.5.1	Checksum Error on a .BAC File	6-12
6.2.5.2	Unrecoverable Disk Error Reading a .BAC File	6-12
6.2.5.3	Improper .BAC File Size	6-12
6.2.5.4	Mismatch of BASIC-PLUS Version Numbers.	6-13
6.3	Analyzing System Crashes - ANALYS	6-13
6.4	Octal Debugging Tool - ODT	6-15
6.4.1	Running and Terminating ODT.	6-18
6.4.2	Access to Locations in the Address Space (/ and \)	6-19
6.4.2.1	Opening the Preceding Location (↑ or ^)	6-20
6.4.2.2	Opening a PC Relative Location (← or —)	6-21

6.4.2.3	Opening an Absolute Location (@)	6-21
6.4.2.4	Opening a Relative Branch Offset Location (>)	6-22
6.4.2.5	Returning to an Interrupted Sequence (<)	6-22
6.4.3	Printing the Contents of Locations	6-23
6.4.3.1	Printing ASCII Format (")	6-23
6.4.3.2	Printing Radix-50 Format (%)	6-23
6.4.4	Relocation Registers	6-24
6.4.5	Interpretive Address Quantities (Q and .)	6-26
6.4.6	Error Procedures	6-27
6.4.7	Reading a Pack Identification with ODT	6-27

Chapter 7 System Utility Operations

7.1	General Utility Operations - UTILTY	7-1
7.1.1	Running and Terminating UTILTY	7-8
7.1.2	Principles of Disk Management	7-8
7.1.2.1	Preparing a Disk for Use on a Drive	7-8
7.1.2.2	Disk Mounting Options	7-11
7.1.2.3	Removing All Files from an Account	7-12
7.1.2.4	Changing Quota and/or Password of an Account	7-12
7.1.3	Operational Control of the System	7-12
7.1.3.1	Controlling the Number of Logged-In Jobs	7-13
7.1.3.2	Broadcasting Messages to Terminals	7-13
7.1.3.3	Controlling Jobs	7-14
7.1.3.4	Setting Job Priority, Run Burst, and Maximum Size	7-14
7.1.3.5	Controlling Keyboards and Remote Lines	7-16
7.1.3.6	Changing System Date and Time	7-16
7.1.4	Data Caching Control	7-16
7.1.4.1	Size of the Cache	7-17
7.1.4.2	Sequential and Random Caching Modes	7-17
7.1.4.3	Random Mode Caching	7-18
7.1.4.4	Sequential Mode Caching	7-18
7.1.4.5	LIST CACHE Command	7-19
7.1.4.6	ENABLE CACHE Command and Options	7-19
7.1.4.7	DISABLE CACHE Command	7-21
7.1.4.8	FLAG Command and Options	7-21
7.1.4.9	Caching Guidelines	7-22
7.1.5	Run-Time System Control	7-24
7.1.5.1	Adding and Removing Auxiliary Run-Time Systems	7-25
7.1.5.2	Loading and Unloading a Run-Time System	7-28
7.1.5.3	Associating a File with a Run-Time System	7-29
7.1.6	Resident Library Control	7-29
7.1.6.1	Resident Library UTILTY Commands	7-30
7.1.6.2	Loading and Unloading a Resident Library	7-31
7.1.7	System File Control	7-31
7.1.7.1	Adding and Removing Swapping Files	7-32
7.1.7.2	Adding and Removing Overlay and Error Files	7-35

7.1.8	Defining Concise Command Language (CCL) Commands	7-36
7.1.8.1	Adding a CCL Definition	7-37
7.1.8.2	Listing Currently Defined CCL Commands	7-38
7.1.8.3	Removing a CCL Definition	7-38
7.1.9	System Logical Names	7-38
7.1.9.1	Adding New Names	7-39
7.1.9.2	Removing Logical Names	7-39
7.1.9.3	Changing a Disk Logical Name	7-39
7.1.9.4	Listing System Logical Names	7-40
7.2	Monitoring System Status - SYSTAT	7-40
7.3	Dynamic Display of System Status - VT5DPY and VT50PY	7-42
7.3.1	Running and Terminating VT5DPY and VT50PY	7-42
7.3.2	Screen Layout	7-44
7.3.2.1	Header Line	7-47
7.3.2.2	Job Status	7-47
7.3.2.3	Memory Status	7-47
7.3.2.4	Disk Structure	7-48
7.3.2.5	Busy Device Statistics	7-48
7.3.2.6	Message Receiver Statistics	7-48
7.3.2.7	Free Buffer Status	7-48
7.3.2.8	Run-Time System Statistics	7-48
7.3.2.9	Resident Library Statistics	7-48
7.4	Terminal and Remote Line Characteristics - TTYSET	7-49
7.4.1	Terminal Line Speed Characteristics File - TTYSET.SPD	7-49
7.4.2	TTYSET Privileged Feature - KBN: Command	7-51
7.4.3	Automatic Setting of Terminal Characteristics	7-52
7.4.4	Setting Terminal Characteristics of Remote Lines - /RING	7-52
7.5	Reinitializing a Disk During Time Sharing - DSKINT	7-53
7.5.1	Running and Terminating DSKINT	7-54
7.5.2	DSKINT Processing and Unexpected Errors	7-55
7.5.3	Recommended Procedures Following DSKINT	7-56
7.6	Using the ONLCLN Program	7-56
7.6.1	When to use ONLCLN	7-56
7.6.2	Invoking ONLCLN	7-57
7.7	Optimizing Disk Directory Structure - REORDR	7-58
7.7.1	Dialogue Questions and Responses	7-59
7.7.2	Reordering the Public Structure	7-60
7.7.3	Processing and Error Messages	7-61
7.7.4	Sample REORDR Dialogue and Messages	7-62
7.8	Processing User Comments - GRIPE	7-62
7.9	Communicating With Other Terminals - TALK	7-64
7.9.1	Sending a Message to Another Terminal	7-64

Chapter 8 The BACKUP System Package

8.1	BACKUP and System Management	8-1
8.2	How BACKUP Works	8-2

8.2.1	Dialogue.	8-2
8.2.2	File Selection	8-2
8.2.3	Entering Accounts (Optional-Restore only)	8-3
8.2.4	File Transfer.	8-3
8.2.5	File Comparison and Deletion (Optional)	8-4
8.2.6	Building the Listing File	8-4
8.3	Running BACKUP.	8-4
8.3.1	File Specification	8-5
8.3.2	Running BACKUP under BATCH	8-8
8.3.3	The BACKUP Dialogue	8-9
8.3.4	Interruption Commands	8-15
8.3.5	Mounting and Dismounting Volumes	8-16
8.3.6	Writing the BACKUP Structure on Disks	8-18
8.4	BACKUP Error Handling	8-18
8.4.1	Dialogue Command Errors	8-19
8.4.2	Interruption Command Errors	8-21
8.4.3	Volume Mount Errors	8-21
8.4.4	BACKUP Processing Errors	8-22
	8.4.4.1 Selection Errors.	8-23
	8.4.4.2 Transfer, Deletion, and Listing Errors	8-25
	8.4.4.3 Informational Messages	8-25
8.5	Backing Up System Files - Example	8-26
8.5.1	Terminal Printout - Backup	8-26
8.5.2	Listing File - Backup	8-30
8.6	Restoring Files - Example	8-35
8.6.1	Terminal Printout - Restore	8-35
8.6.2	Listing File - Restore	8-38
8.7	Loading the Index File - Example	8-41
8.7.1	Terminal Printout - Loadindex.	8-42
8.7.2	Listing File - Loadindex	8-43
8.8	Listing the Index File - Example.	8-44
8.8.1	Terminal Printout - List.	8-45
8.8.2	Listing File - List	8-45

Chapter 9 SAVE/RESTORE System Program

9.1	When to use SAVE/RESTORE	9-1
9.2	Definition of SAVE/RESTORE Terms	9-2
9.3	Running SAVE/RESTORE.	9-3
9.4	SAVE/RESTORE Options	9-5
9.5	SAVE/RESTORE Dialogue.	9-6
9.5.1	SAVE Volumes	9-7
9.5.2	Device Specifications.	9-8
9.5.3	Checking the Input Volume	9-8
9.5.4	Checking the Output Volume.	9-9
9.5.5	Saving a RSTS/E Disk using the SAVE Dialogue	9-11
9.5.6	Restoring a RSTS/E Disk using the RESTORE Dialogue	9-16

9.5.7	Copying a RSTS/E Disk using the IMAGE Dialogue	9-20
9.5.8	IDENTIFY Dialogue	9-24
9.5.9	Full Function Command Line	9-26
9.6	SAVE/RESTORE and Booting	9-26
9.7	Operator Interface During Processing	9-28
9.7.1	Mounting and Dismounting Volumes	9-28
9.7.2	Re-accessing Devices	9-29
9.7.3	Aborting SAVE/RESTORE	9-29
9.8	SAVE/RESTORE Summary Report.	9-29
9.8.1	Summary Report Format.	9-30
9.8.2	Summary Report Example	9-31
9.8.3	Summary Report Run Statistics	9-31
9.9	SAVE/RESTORE Error Handling.	9-32
9.9.1	General SAVE/RESTORE Error Messages.	9-33
9.9.2	Transfer Errors - Fatal and Non-Fatal	9-37

Chapter 10 Using the Automated Patching Facility

10.1	Patching RSTS/E Binary Code - ONLPAT	10-1
10.1.1	Using ONLPAT in Keyboard Mode	10-1
10.1.2	Using ONLPAT in Command-File Mode	10-2
10.1.3	Patching a Running Monitor with ONLPAT.	10-3
10.2	Building ONLPAT Command Files	10-3
10.3	Patching ASCII Source Code - CPATCH	10-5
10.4	Building CPATCH Command Files	10-5
10.4.1	Building the Patching Command File	10-6
10.4.1.1	File Naming Convention.	10-6
10.4.1.2	Creating the CPATCH Command File	10-6
10.4.2	Building the PBUILD Command File	10-10
10.4.2.1	Using Comments	10-10
10.4.2.2	Using Indirect Command Files.	10-10
10.4.2.3	Using Underscore (___) as a Quote Character	10-10
10.4.2.4	Command File Statements	10-10
10.4.2.5	Sample PBUILD Command File	10-13
10.5	Running the PBUILD Program	10-13
10.6	Error Messages	10-19

Chapter 11 Device Testing Package (DEVTST)

11.1	Introduction to DEVTST.	11-1
11.2	DEVTST Operating Procedures.	11-2
11.3	DEVTST Dialogue Errors	11-4
11.4	CPEXER	11-4
11.4.1	CPEXER Tests	11-5
11.4.2	CPEXER Dialogue.	11-5
11.4.3	CPEXER Errors	11-5

11.5	CPUTST	11-7
11.5.1	CPUTST Dialogue	11-7
11.5.2	CPUTST Errors	11-8
11.6	DSKEXR	11-8
11.6.1	DSKEXR Dialogue	11-8
11.6.2	DSKEXR Errors	11-9
11.7	DSKSEK	11-9
11.7.1	DSKSEK Dialogue	11-10
11.7.2	DSKSEK Errors	11-10
11.8	DTEXER	11-11
11.8.1	DTEXER Dialogue	11-11
11.8.2	DTEXER Errors	11-11
11.9	DXEXER	11-12
11.9.1	DXEXER Dialogue	11-12
11.9.2	DXEXER Errors	11-12
11.10	KBEXER	11-13
11.10.1	KBEXER Dialogue	11-13
11.10.2	KBEXER Errors	11-13
11.11	LPEXER	11-14
11.11.1	LPEXER Dialogue	11-14
11.11.2	LPEXER Errors	11-14
11.12	MTEXER	11-14
11.12.1	MTEXER Dialogue	11-15
11.12.2	MTEXER Errors	11-15
11.13	PPEXER	11-15
11.13.1	PPEXER Dialogue	11-16
11.13.2	PPEXER Errors	11-16
11.14	PREXER	11-16
11.14.1	PREXER Dialogue	11-16
11.14.2	PREXER Errors	11-16

Appendix A Hardware Bootstrap Procedures

A.1	BM873-YA Procedure	A-2
A.2	BM873-YB Procedure	A-3
A.3	H324 Pushbutton Panel Procedure	A-3
A.4	MR11-DB Procedure	A-4
A.5	BM792-YB Procedure	A-4
A.6	M9301-YA and M9301-YB Procedure	A-5
A.7	M9301-YC Procedure	A-6
A.8	M9301-YF Procedure	A-7
A.9	M9312 Procedure	A-7
A.9.1	Using the M9312 Console Emulator	A-8
A.9.2	Determining the M9312 ROM Configuration	A-9
A.9.3	Bootstrapping with the M9312	A-11
A.10	Load Program Bootstrap for the RL01/RL02 and RK07.	A-13

Appendix B System Generation Error Messages

B.1	Initialization Code Error Messages	B-1
B.2	RT11 Run-Time System Error Messages	B-13
B.3	Batch Error Messages	B-15

Appendix C Auxillary System Program Files

C.1	Character Generation File - CHARS.QUE	C-1
C.2	Batch Command Decoding File - BATCH.DCD	C-1
C.3	Backup Prompt File - BACKUP.PRM	C-2
C.4	Error Package Data File - ERRBLD	C-2

Appendix D Number Conversion

Figure

5-1	System Controlled Programs and Operator Interaction	5-3
7-1	Priority Byte Format	7-14
7-2	Monitor Caching Checks	7-20
7-3	Caching Mode Checks	7-22
9-1	Summary Report	9-30
A-1	M9312 ROM Arrangement	A-8

Table

2-1	Initialization Option Summary	2-3
3-1	Control File Commands	3-4
4-1	REACT System Program Functions	4-2
4-2	Responses to ENTER Function Queries	4-3
4-3	Responses to DELETE Function Queries	4-4
4-4	MONEY Program Options	4-6
4-5	MONEY Program Output	4-7
5-1	OPSER Commands	5-8
5-2	OPSER Message and Action Request Contents	5-11
5-3	OPSER On-Line Job List	5-12
5-4	QUEMAN Start-Up Commands	5-16
5-5	QUEMAN Start-Up Options	5-17
5-6	QUEMAN Interrupt Commands	5-18
5-7	SPOOL Start-Up Options	5-24
5-8	SPOOL Syntax Error Messages	5-29
5-9	SPOOL Interrupt Commands	5-30
5-10	SPOOL Error Text in User Requested Output	5-37
5-11	BATCH Start-Up Options	5-40
5-12	BATCH Interrupt Commands	5-41
5-13	BATCH Device Type Designators	5-43
5-14	PLEASE Commands to OPSER	5-45
5-15	BACKUP Commands through OPSER	5-47
6-1	ERRDIS Dialogue Explanation	6-5
6-2	User Description Data	6-8
6-3	Disk Error Detailed Description	6-9
6-4	Nondisk Peripheral Device Format	6-10

6-5	Nonperipheral Error Format	6-11
6-6	System Crash Error Code	6-15
6-7	ODT Characters and Symbols	6-16
6-8	ODT File Question Responses	6-18
7-1	UTILITY Commands	7-1
7-2	Procedures for Using Disk Packs and Cartridges	7-9
7-3	Run-Time System ADD Command Errors	7-27
7-4	ADD SWAPFILE Command Errors	7-33
7-5	ADD OVERLAY and ADD ERROR Command Errors	7-36
7-6	Display Program Options	7-43
7-7	Display Program Commands	7-44
7-8	REORDR Message Text	7-61
8-1	BACKUP File Specification	8-6
8-2	Backup Dialogue Summary	8-9
8-3	Restore Dialogue Summary	8-11
8-4	Loadindex Dialogue Summary	8-12
8-5	List Dialogue Summary	8-13
8-6	Interruption Commands	8-15
8-7	BACKUP Dialogue Error Messages	8-19
8-8	Interruption Command Error Messages	8-21
8-9	BACKUP Volume Mount Error Messages	8-21
9-1	SAVE/RESTORE Functions	9-4
9-2	Response Types for the "SAV/RES Function:" Prompt	9-5
9-3	SAVE/RESTORE Options	9-6
9-4	SAVE/RESTORE Device Specification Options	9-8
9-5	SAVE Dialogue Questions	9-12
9-6	RESTORE Dialogue Questions	9-17
9-7	IMAGE Dialogue Questions	9-21
9-8	IDENTIFY Dialogue Question	9-25
9-9	Bootting RSTS/E and SAVE Volumes	9-27
9-10	Summary Report Run Totals	9-32
9-11	SAVE/RESTORE Non-Fatal Transfer Errors	9-38
10-1	PBUILD Error Messages	10-20
10-2	CPATCH Error Messages	10-21
11-1	DEVTST Programs	11-3
11-2	RSTS/E Device Designator	11-3
A-1	Summary of Hardware Bootstrap Addresses	A-2
A-2	Device Codes for M9301-YA and M9301-YB Bootstraps	A-6
A-3	Device Codes for M9301-YC Bootstrap	A-6
A-4	Device Codes for M9301-YF Bootstrap	A-7
A-5	ROM Locations	A-10
A-6	M9312 Bootstrap ROM Configuration	A-10
A-7	Device Name and Mnemonic	A-11
A-8	Switch Register Codes (SWR)	A-12
A-9	Console Switch Register Settings	A-12
B-1	Initialization Code Error Messages	B-2
B-2	Recoverable RT11 Run-Time System Errors	B-13
B-3	RT11 Run-Time System Fatal Execution Errors	B-14
B-4	BATCH Error Messages	B-15

Commercial Engineering Publications Typeset this manual using DIGITAL's
TMS-11 Text Management System.

834ALL

Preface

This guide tells you how to operate and manage a RSTS/E system. You should be familiar with the structure and programming of RSTS/E and know time sharing software and hardware. The basis for the manual is a program set with privileged status and privileged features. Only the RSTS/E system manager and privileged users should have access to this guide.

For more information on RSTS/E guides and manuals, consult the *RSTS/E Documentation Directory*.

This manual uses the following symbology conventions:

- Ⓕ indicates pressing the LINE FEED key
- Ⓖ indicates pressing the RETURN key
- Ⓔ indicates pressing the ESCAPE key (shown as ALT MODE on some terminals)
- △ represents a required space
- color A contrasting color in examples indicates user input. Unless otherwise noted, user input is terminated by pressing the RETURN key.



Chapter 1

RSTS/E System Structure and System Management

RSTS/E (Resource Sharing Timesharing System/Extended) runs on a PDP-11 computer and allows simultaneous, time shared access to PDP-11 hardware resources and to RSTS/E system software components through either local or remote terminals. Hardware and software supported by RSTS/E are outlined in the RSTS/E Software Product Description.

This chapter introduces the system manager to the structures of RSTS/E and provides references to further descriptions of the philosophy and uses of RSTS/E.

1.1 Disk Organization

1.1.1 File Structure

The RSTS/E Monitor permits all access to system and user data on the RSTS/E system by controlling the logical structure of files. The file structure is flexible enough so that the monitor can control and access any type of information. The file structure design is based on the need to control and access data and code on disk.

The logical disk structure is divided into two types: public and private. The public disk structure consists of a system disk and additional public disk packs or disk cartridges. The system disk must be physically on line and logically mounted whenever the system is running so that it is accessible to all users during time sharing operations. All other public disk packs that you may need to access also should be physically on line and logically mounted, so that all users may access them.

The system disk contains monitor code, initialization code, the default run-time system, the system startup program, and the control files for system start up and crash recovery. The system disk may also be used for storage of active user jobs which are temporarily swapped out of memory. Remaining

space on the system disk and all space on other public disks is used for auxiliary run-time system files, optional system files, and system programs and is available for general storage of user programs and data files.

Any disk drives not devoted to the public structure can accommodate private disk packs or disk cartridges. Private disks can be logically mounted and dismounted and interchanged as needed during time sharing operations. A private disk provides a means to restrict disk storage to a defined set of users. The file structure on a disk, whether it is designated public or private, is the same.

Access to files in the RSTS/E system is accomplished by two structures called a Master File Directory and a User File Directory. A Master File Directory, or MFD, exists on each disk initialized for use on the RSTS/E system. The MFD is treated as an account on the disk and its purpose is to catalogue other accounts on the disk. The MFD on the system disk is a special case, since it maintains a catalogue of the accounts which can be used to log into the system. MFDs on other disks contain entries of accounts which can create files on that disk. Any user can access any file on any disk if the protection code of the file permits. However, only those users whose accounts are entered in the MFD of the private disk can create files on the disk.

One User File Directory, or UFD, exists for each user account on a disk. (The UFD is not actually created until a file is created for the related account.) The UFD catalogues all program and data files under an account and maintains accounting and access information for these files. The UFD contains all retrieval information for the files because each file is pure data and has neither linkage nor structural information.

1.1.2 Disk Optimization

Whenever a file is opened on the public structure, the directories on every public disk are searched. The search either verifies the existence of the file or ensures its nonexistence. The overhead of searching more than one disk can be avoided by placing the file on a private disk.

It is sometimes advantageous to dedicate an entire private disk to a single large production file. This organization ensures an efficient directory structure and minimizes overhead to access file data. When more than one file is on the same private disk, it is best to dedicate a whole account to each production file. This organization minimizes directory search overhead. In general, the system manager should keep volatile files and stable files in separate accounts to avoid directory fragmentation.

In an environment where distinct data files must be accessed by the same program, the optimal organization is to keep each file on a different private disk. If a program must access more than one file on the same disk, overhead is increased because of disk head movement. A large percentage of time, therefore, is spent in moving the disk head back and forth. If, however, each file referenced by the program exists on a distinct private disk, head movement is not required whenever the program references another file. Head movement is restricted to locating the data itself.

A RSTS/E fixed head disk is frequently used as a non-file structured device for swapping storage. It may be used as a file-structured device either in the public structure or as a private disk. When used as a file-structured device, a fixed head disk may contain user data files as well as swapping and other system files. The system manager should consider placing high usage files (such as data base index files) on a fixed head disk.

1.2 System Operation Concepts

Immediately after logging into the system, a user's terminal is under the control of the system default run-time system's keyboard monitor. That is, the terminal is said to be at the system command level because the user may type a system command and the keyboard monitor will process it accordingly. The terminal returns to the system command level when any program execution is completed or whenever a CTRL/C is typed at the terminal. When the terminal is at the system command level, the keyboard monitor examines each ASCII text line entered by the user and determines whether that line is a system command. System commands are executed immediately after being entered as described in the *RSTS/E System User's Guide*. If the system default run-time system is BASIC-PLUS, if the line is a BASIC-PLUS immediate mode statement, or a BASIC-PLUS numbered statement, the system command level is also known as "edit mode" for the BASIC-PLUS Run-Time System, since BASIC-PLUS programs can be edited by retyping numbered statements.

NOTE

The following actions outline the operation of the BASIC-PLUS Run-Time System. RSTS/E supports multiple concurrent run-time systems allowing programming in languages other than BASIC-PLUS. The specific processing of command lines and program operations depends, of course, on the specific run-time system and language processor.

In the BASIC-PLUS Run-Time System, an immediate mode statement is first translated into intermediate code, which is placed in the user's job area and executed immediately by BASIC-PLUS. Program statements preceded by line numbers are analyzed and stored in their ASCII form in a temporary disk file named TEMPnn.TMP stored under the user's account (where nn is the user's job number). Each program statement is also compiled into its intermediate code representation and is placed in the user's area of memory.

Intermediate code created in the user's job area upon entry of numbered statements is not executed automatically. The related program statements can be changed. A copy of the intermediate code of the program can be transferred to disk storage (as a file with a .BAC filename extension) or to an external storage medium.

The user job area is initialized at login time and set to a size of at least 1K words ($K = 1024$). The job area can grow in increments of 1K words to a maximum size set by the system manager at the start of time-sharing operations.

A user leaves edit mode by typing the RUN system command or the CHAIN immediate mode statement. In run mode, BASIC-PLUS interpretively executes the intermediate code stored in the user's job area. Following program execution, the user's terminal is returned to edit mode, signalled by printing of the READY message. The user can interrupt BASIC-PLUS by typing CTRL/C, which also returns the user's terminal to edit mode.

Under the RSTS/E system, jobs run one at a time. A job runs until it either enters an I/O wait state or exhausts the time quantum assigned by either the system or the system manager. When the currently running job ceases to run, the scheduler runs the next job that is ready. Meanwhile, the interrupt-driven I/O device handlers are processing requested data transfers. Upon completion of a transfer, the scheduler marks the job that requested the transfer as ready to run again and starts it from the point at which execution ceased.

RSTS/E attempts to keep as many jobs in memory as possible. When more memory is required to run a job than is available, the system temporarily moves some jobs out of memory and stores them in one of four areas known as swap files. By convention, these files are named SWAP0.SYS, SWAP1.SYS, SWAP2.SYS, and SWAP3.SYS. This operation is called swapping. When a job is again eligible to run, it is swapped back into memory. Jobs waiting for keyboard input and jobs waiting for device I/O completion are most likely stored in the swapping files, while jobs currently running or involved in disk or magnetic tape data transfers are necessarily in memory.

As the system processes each job, it maintains accounting information in memory concerning that job. When the job is logged off the system, this information is used to update the accounting information stored on the system disk for that account.

1.3 System Management

Management of RSTS/E begins with providing properly tailored hardware and software configuration, proceeds through initializing the software at system generation time, and continues with the daily functioning of time sharing. To ensure that these steps of management are efficiently performed, the manager of a RSTS/E system should be familiar with time-sharing concepts and practices or should have a close working relationship with a senior programmer or analyst who is experienced in time sharing.

To begin managing a system well, the person responsible for RSTS/E operation must have knowledge of the processing requirements of the system and also know the capabilities and structure of the system. The information supplied in the *RSTS/E System Generation Manual* explains critical aspects of hardware and software options and provides memory requirements to assist in configuring a RSTS/E system.

After RSTS/E software is generated, many initialization options are available to structure and control system elements. A few of the initialization options are complex and sophisticated and their efficient usage at system generation

time is of paramount importance. Once time sharing begins, certain restructuring capabilities become more difficult to employ. The manager or designate should be careful about the first-time initialization of RSTS/E. Unfortunately, no cookbook approach can attain proper system initialization.

To manage daily time sharing efficiently, an individual must know data processing functions, implement time sharing utility operations, and be familiar with RSTS/E concepts and structure. The set of utility programs described in this manual and in the *RSTS/E System User's Guide* is available to perform necessary tasks.

To ensure that responsible individuals are kept informed, the manager should designate an individual responsible for current documentation of the system, including both locally generated and DIGITAL-supplied procedures and guides. That responsible individual should ensure that delegated members of the staff receive the latest information. In too many cases, improper utilization of resources results from responsible individuals not being aware of published information.

If special facilities are required, a set of privileged programming features is available to the developer at the local installation. The system library can be expanded with tailored utility programs. Auxiliary libraries are available for special groups of users. The descriptions of special features can be located by consulting the *RSTS/E Documentation Directory*.

1.4 Privilege

Privilege is a special condition for a user job. With privilege, a job has capabilities not available to other, non-privileged jobs on the system. The following sections define privilege and explain how privilege is attained and relinquished.

1.4.1 Privileged Capabilities

A privileged job on RSTS/E has the following special capabilities.

1. unlimited access on the system,
2. ability to designate privileged programs,
3. use of privileged aspects of system programs, and
4. use of privileged SYS system functions and the PEEK function.

Unlimited access means that no protection code can protect a file against read and write access. A privileged job can create and delete files under any account number and can access files on locked disks. Such unlimited access by a privileged job does not generate the normal ?PROTECTION VIOLATION error (ERR=10).

NOTE

Privilege does not bypass intrinsic protection mechanisms such as locked blocks in updating files.

A privileged job can set the privileged bit in the protection code. The privileged bit, with the compiled file protection bit, designates a program as privileged. Thus, any program with a protection code of <192> (the sum of the privileged protection <128> and the compiled file protection <64>) or greater denotes a privileged program. Both protection code values must be set for a program to have privilege.

When a job has privilege, it can use privileged aspects of system programs, privileged SYS system functions, and the PEEK function.

Many system programs are designated privileged at system generation time because they execute privileged SYS system functions and the PEEK function. Because some system programs execute privileged operations, the related programs and operations are described in manuals separate from those the beginning user normally has access to. This manual describes privileged system programs and privileged aspects of system programs. The *RSTS/E Programming Manual* describes the privileged SYS system functions and the PEEK function.

1.4.2 Privilege and System Operation

A job has privilege under one of the following conditions:

1. it is a logged out job (a job that is not associated with an account),
2. it is running under a privileged account, or
3. it is running a privileged program.

The privilege remains or is dropped depending on how processing continues for the job.

A logged out job has privilege because the system must minimally perform certain privileged operations to log a job into the system. The privilege remains in effect as long as the job remains logged out.

A job running under a privileged account has privilege. A privileged account is one whose project number is 1. The account number thus has the form [1,*] where the asterisk (*) represents a programmer number between 0 and 254. (The privileged account with a programmer number of 1 is the Master File Directory for disks.) The system library account [1,2] is an example of a privileged account.

A job running under a privileged account has permanent privilege until the job is logged out or the job changes to a non-privileged account (an account whose project number is not 1).

The PIP.SAV program can be used to set the privileged bit in the protection code. Privilege is required to set the privileged bit. To designate a privileged program, the privileged job must assign the protection code of <192> or greater to the compiled form of the program. The following sample statement shows the procedure.

```
RUN $PIP.SAV
* FILE.BAC<232>/RE
*
```


The resultant protection code designates the privileged and compiled protection codes (<128> and <64>) with read/write protection against the owner's project (<8>) and read/write protection against all others not in the owner's project (<32>). This protection code combination enables users inside and outside the owner's project to run the privileged program but prohibits those users from reading or writing (and therefore being able to delete or rename) the file.

Note that when a disk file has the privileged <128> bit set, the deletion of that file causes the Monitor to overwrite it with zeroes. The Monitor deletes the file and overwrites it with zeroes when the following conditions are met:

1. The privileged <128> bit is set and
2. you execute an explicit BASIC-PLUS KILL or UNSAVE, or
3. you execute an OPEN FOR OUTPUT with the same filename implicitly deleting the file, or
4. the file is tentative and you execute a reset CLOSE with a negative channel number, or
5. you perform a ZERO operation on the file's account.

Because the Monitor is in a locked state during the overwrite operation, no other programs on the system can perform file processing. Therefore, permanent privilege is required to set the <128> bit. Also, because a file with the <128> bit set in the protection code will be overwritten with zeroes when it is deleted, you should ensure that sensitive data files (such as ACCT.SYS) are protected by setting the <128> bit in the protection code. That is, rename the file to its required protection code plus <128> but do not set the <64> compiled bit. Note that to process a privileged (<128>) file through the QUE system program, you must have permanent privilege. Also, an attempt to delete such a file with the QUE program /DE switch will fail.

The NAME AS statement can be used in BASIC-PLUS immediate mode to designate a compiled file as privileged. The only restriction, however, is that the job under which the statement is executed must itself have permanent privilege.

A job running a privileged program has temporary privilege unless it is running under an account which has permanent privilege. The job gains the temporary privilege when it runs a program, or compiled file, whose privilege bit is set. The privilege remains until the privileged program exits or until the program drops its temporary privilege. This type of privilege is necessarily temporary because users of both privileged and non-privileged accounts may be able to run a privileged program. Accordingly, if the privilege were not temporary, any unexpected halt in the job would leave the system vulnerable to unwarranted tampering. A job running permanently privileged, however, can never drop its privilege unless it changes to a non-privileged account.

A temporarily privileged job can rely on the normal protection mechanisms built into the system. Under programmed control, the program can either permanently or temporarily relinquish (drop) its temporary privilege by executing a SYS system function call. This ability to drop privilege allows a job

to selectively perform privileged operations. For example, a job could initially set itself up using privileged capabilities and then permanently drop its privilege because further processing does not require privilege. Alternatively, a job could selectively drop and later regain its temporary privilege depending upon the type of processing required. Regaining temporary privilege is accomplished by executing the same SYS system function which allows the job to drop privilege. Regaining privilege, however, is possible only if the job temporarily dropped privilege rather than permanently dropped privilege.

Under certain conditions, the run-time system controls whether temporary privilege is retained. For example, when a program runs via CCL command at other than the lowest numbered line, BASIC-PLUS may drop a job's temporary privilege. The conditions under which this action occurs are described in the *RSTS/E Programming Manual*. When a program is run by a CHAIN statement at a line number not the lowest numbered line, BASIC-PLUS unconditionally drops a job's temporary privilege.

Note that to prevent deliberate or inadvertent misuse of privilege, DIGITAL recommends that user-written privileged programs initially drop privilege and then regain privilege only as needed. This prevents inadvertent misuse or deliberate abuse of privilege.

1.4.3 Guidelines for Privileged Operation

Permanent privilege gives to a user the full capability of the RSTS/E system but also requires of the user a distinct responsibility. It must be emphasized that these capabilities can destroy the system. No error messages are generated because privilege bypasses the normal protection mechanisms. To avoid destroying data, privilege must be carefully controlled and used.

Privilege capability begins with the assignment of privileged accounts. At system generation time, two privileged accounts, the system library [1,2], and the MFD account [1,1] are created. By having access to the system library, the system manager has privilege capability. One such capability is the ability to designate other users as privileged by assigning them accounts with a project number of 1. Account [1,1] can be used as a privileged account but such use is strongly discouraged.

System managers should take precautions against unauthorized access to privileged accounts. Passwords to such accounts should be kept confidential, and should be changed from time to time. Passwords should be devised so that they are not too easily guessed. Obviously, the responsibility for maintaining system integrity extends to other privileged users besides the system manager. This is because users with privileged accounts have all the capabilities of the system manager. Specifically, privileged users can design and implement routines which use privileged SYS system functions and the PEEK function. Privileged users can also designate programs as privileged.

If a program is designated as privileged and is not protected against execution, any user can run the program with temporary privilege. This action extends the use of privileged functions to non-privileged users in the same way standard system library programs do. For example, the program TTYSET

allows a user to change the characteristics of his terminal. Such an operation is privileged. With temporary privilege, however, TTYSET executes the normally privileged operation for an owner of a non-privileged account. TTYSET does not generate the expected protection violation error.

The same TTYSET program additionally allows a privileged user to change the characteristics of another user's terminal. But TTYSET is coded to ensure that a user attempting to change the characteristics of another's terminal is indeed a permanently privileged user. As a result, TTYSET makes available to a non-privileged user certain privileged operations but restricts other privileged features to privileged users.



Chapter 2

System Start Up, Shutdown, and Automatic Restart

2.1 Starting up RSTS/E: Bootstrapping RSTS/E Into Memory

Starting up RSTS/E is a two step operation. The first step requires loading the RSTS/E initialization code into memory and the second step involves using one of the RSTS/E initialization options.

The initialization code in RSTS/E exists on the system disk in a system file named INIT.SYS. The code is loaded into the lower 48K words of memory and overlaid after its execution by system and user programs. The following sections describe the ways that initialization code can be bootstrapped into memory. The *RSTS/E System Generation Manual* describes and explains the RSTS/E initialization options. Section 2.1.3 of this manual summarizes the options.

2.1.1 Bootstrapping RSTS/E using a Hardware Bootstrap Loader

The procedures to follow when bootstrapping RSTS/E depend on the type of hardware bootstrap loader and the type of disk used as the system device. The following steps explain the procedure.

1. Make sure that the system disk is physically mounted on a disk unit. This step is unnecessary for fixed head disks because they are already permanently and physically mounted.
2. Make sure that the system disk drive is READY and is in the WRITE ENABLE condition.
3. If any RK05F disks are on the system, ensure that all units are running and READY.

4. Ensure that the console terminal is on line.
5. Refer to Appendix A for the proper bootstrap operation.

When the initialization code is loaded, it prints the system header, installation name, and disk unit bootstrapped followed by the OPTION: message. If no messages appear after performing the bootstrap operation, ensure that the console terminal is on line and retry the bootstrap procedure. If the initialization code prints an error message, refer to Appendix B for the recovery procedure.

If the automatic restart facility is to be enabled, set the CPU switch register to 177777; that is, all switches are in the up position. The automatic restart facility remains enabled as long as the CPU switch register remains set to 177777. If any switch between positions 15 and 0 is set to its 0 position (that is, to its down position), the automatic restart facility is disabled for the duration of this condition. If the CPU has no switch register, the automatic restart facility is always enabled.

2.1.2 Bootstrapping RSTS/E after a System Halt

If the RSTS/E system halts as a result of a catastrophic error or system crash, the halt address is always 54. The address lights indicate 56 in such a case because the program counter contains the address of the next sequential instruction. (See Section 2.3.1 for a description of catastrophic errors, system crashes, and crash dumps.) The manner in which the system can be restarted in such a situation depends upon the results desired. The alternatives are described in the following sections.

2.1.2.1 Requesting an Automatic Restart — If the RSTS/E system halts at address 54, an automatic restart can be manually initiated. The following procedures prescribe the steps to accomplish an automatic restart.

1. Ensure that the CPU is in a halt state as described in Section 2.2.
2. Ensure that the CPU HALT/ENABLE switch is set to its ENABLE position.
3. Set the CPU switch register to 000052.
4. Press the CPU LOAD ADRS switch.
5. Set the CPU switch register to 177777.
6. Press the CPU START switch.

After RSTS/E starts from address 52, it checks to determine whether the CPU switch register is set to 177777. If the switch register is set to 177777, the system bootstraps the RSTS/E initialization code into memory in the special automatic restart mode described in Section 2.3.2.

If the system finds the switch register set to something other than 177777, a halt immediately occurs again at address 54.

2.1.2.2 Requesting an Initialization Option — If the RSTS/E system halts at address 54, the initialization code can be bootstrapped in the normal start up mode. The procedures to request an initialization option are as shown below.

1. Ensure that the CPU is in a halt state as described in Section 2.2.
2. Ensure that the CPU HALT/ENABLE switch is set to its ENABLE position.
3. Set the CPU switch register to 000050.
4. Press the CPU LOAD ADRS switch.
5. Press the CPU START switch.

The initialization code is loaded into memory and prints the system header line followed, on the next line, by the OPTION: question. See Section 2.1.3 for the valid responses to the OPTION: question. If the system halts at address 54 after a system crash, simply pressing the CONT switch bootstraps the initialization code in the normal start up mode.

2.1.3 Starting Time Sharing

When the system disk is bootstrapped, the system heading and OPTION: message is printed. The user has available the options summarized in Table 2-1.

Table 2-1: Initialization Option Summary

Option Name	Abbreviation	Meaning
DSKINT	DS	Initializes and optionally formats a disk. Checks for bad blocks.
COPY	CO	Copies required RSTS/E files from distribution medium (or system tape) to system disk and bootstraps the system disk.
PATCH	PA	Alters RSTS/E system code.
HARDWR	HA	Specifies device controller characteristics.
INSTALL	IN	Installs a RSTS/E Monitor.
REFRESH	RE	Creates or rebuilds the system files in account [0,1] on any initialized RSTS/E disk; rebuilds the storage allocation table for a disk; adds bad blocks to BADB.SYS.
DEFAULT	DE	Establishes or changes a monitor's start-up defaults.
SET	SE	Sets device characteristics on a unit-by-unit basis.

(continued on next page)

Table 2-1: Initialization Option Summary (Cont.)

Option Name	Abbreviation	Meaning
START	ST	Starts the RSTS/E system for SYSGEN or normal time sharing. Can also be initiated by pressing the LINE FEED key.
BOOT	BO	Bootstraps a device.
LOAD	LO	Loads a stand-alone program from account [0,1].
UNISYS	UN	Enables only the console terminal interface. Used as a diagnostic tool with the START option.
FILL	FI	Sets the fill factor of the console terminal for the initialization code.
SAVRES	SA	Backs up (SAVE) and reconstructs (RESTORE) a RSTS/E disk.
HELP	HE	Prints a help message.

The initialization options are described in the *RSTS/E System Generation Manual*. To start time sharing, use the START option.

2.2 Halting the RSTS/E System

A halt in the RSTS/E system is caused by a system crash or by setting the CPU console HALT/ENABLE switch to its HALT position. The PDP-11/70 and PDP-11/45 are considered in a halt state when both the RUN and PAUSE indicators on the CPU console panel are not lit. Otherwise, the CPU is running. For the PDP-11/40, consult the following chart to determine the state of the CPU based on the condition of the console status lights.

Condition		
RUN light	CONSOLE light	Meaning
OFF	OFF	CPU is powered down or bus is hung.
OFF	ON	Bus is hung.
ON	OFF	CPU is running.
ON	ON	CPU is halted.

The SHUTUP program described in Section 3.2 shuts down the RSTS/E system in an orderly fashion. SHUTUP eventually bootstraps the system disk. The program ensures that all files are properly closed and that system accounting information is accurately updated. If the system manager wishes to turn the power off, he first should run the SHUTUP program, wait for the OPTION: question to be printed, and press the HALT/ENABLE switch. Then all disk packs and cartridges should be removed and the power turned off.

Halting RSTS/E by moving the HALT/ENABLE switch on the CPU console panel to its HALT position is dangerous. Clean-up operations may not be completed; disk storage allocation tables and file directories may be left in obsolete states; file data can consequently become corrupted; and accounting information may be lost. The only way to recover from such a disorderly halt and to salvage possibly vital file information is to raise the HALT/ENABLE switch back to its ENABLE position before any other action is taken, to press the CONT switch, and thereby to return the RSTS/E system to the state in which it was before the halt occurred.

2.3 Automatic Recovery and Restart Facilities

2.3.1 Catastrophic Errors and System Crashes

A catastrophic error or a system crash is an unexpected error-trap. (For information on error traps, see the relevant processor handbook.) Such traps can be caused, for example, by referring to a nonexistent (or non-responding) UNIBUS address (bus time-out trap), by referring to an odd address with an instruction that requires a word address, or by attempting to execute a reserved or nonexistent instruction.

Catastrophic errors and system crashes, therefore, can be due to any of three types of problems: 1) privileged account programming errors, 2) hardware malfunctions, and 3) system software malfunctions. Each of these is discussed individually below.

2.3.1.1 Privileged Account Programming Errors — The RSTS/E system software is designed to protect itself against programming errors that occur under nonprivileged accounts. The system itself, upon detecting such an error, aborts execution of the potential error and reports a corresponding error message to the guilty user.

The RSTS/E software is vulnerable, however, to certain types of errors caused under privileged accounts. By intent and design, the system manager (and those to whom any [1,*] accounts are assigned) have been given extensive powers which are not available under nonprivileged accounts. These powers allow privileged users to modify system library programs or to create utility programs in such a fashion that they can access parts of memory. Even though most mistakes will not cause the system to crash, you should use care when programming with privileged SYS system functions.

2.3.1.2 Hardware Malfunctions — Hardware malfunctions can be responsible for crashing the system. If unexplainable and randomly occurring system crashes or catastrophic errors occur (particularly on systems which hitherto have been functioning well), it is likely that a hardware problem has arisen. Hardware problems can be diagnosed by examining the error logging printouts. Refer to Chapter 6 for information on error logging.

2.3.1.3 System Software Malfunctions — Although every attempt has been made to detect and eliminate system software errors, the paths through the RSTS/E software are incalculably numerous. It is possible that, given certain conditions and certain sequences, the RSTS/E software can trap unexpectedly. If a problem of this type is discovered (it should be reproducible in a defined environment and under defined conditions), a DIGITAL software specialist should be contacted. As problems of this type become known, DIGITAL reports them as described in the *RSTS/E Software Dispatch*.

2.3.2 Automatic Recovery from Catastrophic Errors

Whenever an unexpected trap occurs, the system distinguishes the trap as one of two categories: it is either (a) a catastrophic error which affects only one particular user or (b) a system crash for which some software or hardware problem is possibly responsible. The handling of system crashes is treated below.

The handling of catastrophic errors is as follows. The system determines which user was responsible for the error-trap. It flags that user's job with a special code which causes the system to reinitialize that user's job area completely when it is next his turn to run. The system prints on that user's terminal an error message followed by the text ?PROGRAM LOST-SORRY.* The reinitialized user is in the same state as a newly logged in user. The system resumes normal time sharing operations.

When the system detects a condition from which it cannot recover, it does the following:

1. If the system manager enabled the crash dump facility at start up time, the system writes an image of read/write memory and tables to the CRASH.SYS file.
2. If the CPU Switch Register is set to 177777 and at least one minute has passed since start up, the system bootstraps the system disk, loads the initialization code into memory, and executes an automatic restart.

If either of the conditions in item 2 is not true, the system halts at address 54. Note that when the system halts at address 54, a crash dump has been taken as described in item 1 if the system manager enabled the crash dump facility at start up time.

If the system halts at address 54, one of two procedures is possible.

1. The operator depresses the CPU Console CONTInue switch, which causes the system to be bootstrapped into normal system start up mode.
2. The operator starts the CPU at address 52 with CPU Switch Register set to 177777 (see Section 2.1.2.1). This causes the system to be bootstrapped from disk in the special automatic restart mode described below.

*Run-Time Systems other than BASIC-PLUS may issue other error messages.

If the system takes the automatic restart path, no halt occurs. Instead, the system bootstraps itself into memory from the system disk. After the system has been bootstrapped into memory, control is passed to the initialization routines. At this point the system recognizes the fact that it was not activated through a normal system start up but rather through an automatic restart and consequently initializes itself in automatic restart mode. If two system crashes occur within the same minute (more accurately stated, if an error trap occurs within one minute of system start up or if two error traps occur within the same minute), the system halts at address 54. This protects the system against an infinite loop of error-traps and auto restarts caused by some repeating hardware malfunction.

2.3.3 Automatic Restart Mode Initialization

When the system is initialized in automatic restart mode, control by-passes all parts of the initialization code which call for operator intervention and initializes the system using information already stored in memory. The system logs job 1 into the system on KB0: under account [1,2] and causes it to run the system library program INIT beginning at line 100. Since, in automatic restart mode, INIT begins at line 100 (rather than at its lowest line number), it takes directions from the CRASH.CTL system library file (rather than from the operator-designated control file). The CRASH.CTL file must contain INIT commands which perform all operations the system manager considers necessary in the case of an automatic restart. See Section 3.1 concerning the control of system start up.



Chapter 3

Controlling Time Sharing

3.1 Controlling System Start Up – INIT

The system manager can control system start up by means of the compiled system program INIT. System start up occurs when the START option of the initialization code is executed or when automatic restart occurs. At start up, the monitor runs INIT which executes special commands from a control file. The actions which occur can be controlled by tailoring these commands to local requirements.

To control system start up efficiently, the system manager must understand the conditions in effect at start up time.

1. Login attempts are prohibited (the monitor disables the login capability).
2. The monitor logically mounts only the system disk.
3. No output is made to any terminal.
4. The monitor logs the console terminal (KB0:) onto the system under the system library account [1,2].
5. At the console terminal, the monitor executes the command equivalent to CHAIN "INIT" (for manual start up) or CHAIN "INIT" LINE 100 (for automatic restart).

INIT runs as a consequence of condition (5).

If INIT runs as a result of the START option, it executes from the lowest line number and prints a header line and question at the console terminal.

```
INIT V7.0 RSTS V7.0 TIME SHARING  
COMMAND FILE NAME?
```

The name of the control file to execute must be typed. The control file must exist on the system disk (SY0:). If 30 seconds pass without a response or if

only the RETURN key is typed, INIT uses START.CTL on the system disk (SY0:) under account [1,2].

If INIT runs as a result of a system crash and automatic restart, it executes from line 100 and does not print a question. It reads the file CRASH.CTL on the system disk (SY0:) under account [1,2].

Any error encountered when INIT accesses the control file generates messages in the following manner.

```
text      - ERROR IN READING  filename  
  
READY
```

INIT prints the error message text and the name of the file it attempted to read. INIT terminates but the console terminal remains logged into the system. After the error is fixed, initialization can be restarted by typing RUN \$INIT.

To terminate INIT processing prematurely, type CTRL/C. The result of this action depends upon the state of INIT when the CTRL/C is typed.

If INIT is running detached, the following information will be returned to the keyboard device to which INIT was last attached:

```
CTRL/C DETECTED ON TARGET KB:(KBn:)  
<text>  
INIT WILL 'KILL' ITSELF
```

Here KBn: is the device number of the keyboard device that issued the CTRL/C, and the <text> is an information message. The program returns control to the BASIC-PLUS command level on the target keyboard.

If INIT is running attached, the following information will be returned to the keyboard device to which INIT is attached:

```
CTRL/C DETECTED ON TARGET KB:(KBn:)  
<text>  
  
READY
```

where KBn: is the device that issued the CTRL/C and <text> is an information message. INIT returns control to the system command level on the keyboard device to which INIT is attached.

A control file contains special commands for initializing the system for time sharing, recovering system crash information, and performing other locally required operations. The following sections describe the INIT commands and their usage.

3.1.1 INIT Program Commands

The RSTS/E system is not fully initialized until INIT runs and executes commands in the control file. The INIT commands are described in Table 3-1. For example, the system possibly uses other disk devices in addition to the system disk. By means of the MOUNT command, the system manager

can make such devices in the public and in the private structure immediately available to users before they can log into the system. The start up procedures must include making the specified devices physically ready on the proper drive units.

To execute other actions on the system, the system manager can cause INIT to execute BASIC-PLUS commands and programs. The LOGINS command enables further logins on the system. The LOGIN command automatically logs a specified job onto the system at a designated terminal to allow execution of commands and programs. For example, the FORCE command can run the TTYSET system program at the terminal and can subsequently execute commands to establish terminal characteristics of certain keyboards. The SEND command prints text on all on-line terminals.

The INIT program can be run during time sharing to perform any routine operations. Special control files can be created with the proper sequence of commands to execute. The system manager can simply run INIT and specify the proper control file from which INIT will extract the commands to perform the required routine operation.

3.1.2 Creation and Usage of Control Files

The INIT system program control files must contain commands to properly initialize the RSTS/E system. The control files must be stored on the system disk which can be mounted on any drive. The files included in the RSTS/E system generation kit are only samples and, without modification, may not execute properly on a given system. The system manager must ensure that the files include the necessary commands to initialize the local installation. To replace the sample START.CTL file supplied with the RSTS/E kit, run PIP.SAV and proceed as follows:

```
RUN $PIP
*SY0:START.CTL=KB:

      (Enter INIT commands.)

^Z
*
```

The procedure shown ensures that the new file is created on the system disk (SY0:). The SY0: designator is critically important on systems with multiple disks in the public structure. Its use explicitly indicates the system disk. The file must reside on the system disk because, when INIT runs, only the system disk is mounted.

It is important that the start up control file perform the following operations.

1. Mount all public disks,
2. add system files,
3. use indirect command files to perform routine operations like:
 - a. set terminal characteristics,
 - b. establish auxiliary run-time systems,
 - c. define CCL commands, and
 - d. run spooling and operator services programs

4. run the ERRINT program to start error logging,
5. enable logins,
6. broadcast any messages, and
7. end the control file.

Table 3-1: Control File Commands

Command Name and Format ¹	Use
LOGINS	Allows users at both local and remote terminals to enter the system.
SENDΔxxx	Transmits the text xxx to all keyboards currently on line except to the terminal on which INIT is running.
@filespec	Causes INIT to process the indirect command file specified by filespec and stored on a currently mounted disk. The specification can be any valid RSTS/E disk file specification. If an extension is not specified, INIT assumes the extension .CMD. INIT returns to the next command in the current control file when indirect file is completed. Indirect command files can contain @ commands to allow nesting to ten levels.
LOGINΔKBn:Δ[n,m]	Logs the terminal specified by KBn: onto the system using the account indicated by [n,m]. INIT automatically looks up the password. NOTE Unless INIT has been detached, the LOGIN KBn: command cannot be used on the terminal to which INIT is attached.
FORCEΔKBn:Δxxx	Causes the text xxx to be placed in the input buffer of the terminal specified by KBn: and executed as if typed at that terminal. The text can be any BASIC-PLUS command or system command. To force a control character combination to a terminal, place the circumflex (^) or up arrow (↑) character as the first character of the text to force followed only by the proper character and the RETURN character. (That is, no other text should follow the control character combination.) If either the circumflex or up arrow character is the only character of text, INIT forces a CTRL/C to the terminal. If either the circumflex or up arrow character is the first character of more than 2 characters of text, INIT forces a CTRL/C to the terminal ahead of the text. NOTE If an attempt is made to force the CTRL/C combination to the terminal to which INIT is attached, an error is generated and INIT stops. To properly terminate INIT, place either the END or BYE command as the last command in the control file.
¹ The notation Δ indicates that a space character is required. Optional items are within square brackets.	

(continued on next page)

Table 3-1: Control File Commands (Cont.)

Command Name and Format	Use
MOUNT△dev:id MOUNT△dev:id/ONLY	Causes the disk unit specified by the device designator dev: and by the pack identification (id) to be logically recognized by the RSTS/E system. Optionally, the /ONLY switch can be included to mount the disk with read only access. Additionally, the MOUNT command as used in the control file causes a clean operation (if necessary) and an unlock operation. Refer to Section 7.1.2 for information concerning mount, clean, and unlock operations.
ADD△SWAPFILE△n△dev:[file]	Same as UTILITY command of same name. Installs the device specified as a swapping file in position n, where n is 0, 1 or 3. A file name can be given to designate a file on a device rather than the entire device.
ADD△ERROR△dev:file	Same as the UTILITY command of same name. Adds, as the system error message file, the file with the name given and with extension .SYS stored in account [0,1] on the device and unit indicated. The device must currently be mounted. If a device is not specified, the public structure (SY:) is used.
ADD△OVERLAY△dev:file	Same as the UTILITY command of same name. Adds, as the system overlay file, the file with the name given and with extension .SYS stored in account [0,1] on the device and unit indicated. The device must currently be mounted. If a device is not specified, the public structure (SY:) is used.
DETACH	Causes INIT to detach from its terminal and print the message DETACHING. This action allows the terminal to be logged into the system and used to run other programs while INIT is running. INIT is reattached by the ATTACH command.
ATTACH	Attaches INIT to the terminal from which INIT was detached by the DETACH command. The terminal should not be in use by another job because INIT will wait until the terminal is free. The commands preceding ATTACH should logout the job or detach it from the terminal.
BYE or END	Causes execution of the INIT system program to be terminated. BYE causes the job, running ERRINT under account [1,2], to be logged out. This frees the console keyboard for other use and prevents unauthorized use of account [1,2]. END must be used in place of BYE when running ERRINT on the console terminal because END does not logout the job running ERRINT under account [1,2]. See Section 6.1.1 for a description of ERRINT. If neither an END or BYE command is in the control file, INIT encounters the END OF FILE ON DEVICE error and leaves the terminal logged into the system.

The order in which these operations are performed is critical. Mounting all public disks ensures the integrity of the public structure by making available all files on the system. Any private disks can be mounted in a similar manner. By adding system files, such as the swapping, the error message, and overlay files, the system manager establishes the monitor structures for full system file functionality. The routine operations further tailor the system hardware and software according to local installation requirements.

It is important that the ERRINT program be run on all systems. This program sets initial conditions for system error logging and runs the ERRCPY program which transfers to a disk file all errors logged by the RSTS/E Monitor. If ERRINT is the last program to be run and is run as recommended, it will occupy job slot number 1 on the system and thereby be easily monitored. More detailed information on ERRINT and ERRCPY can be found in Chapter 6.

Following the commands to perform all required and routine operations, the command to enable logins should appear. This command activates all swapping space added on the system and allows users to log into the system. Any messages can then be broadcast (notifying the users that time sharing has started, for example). To terminate INIT without an error, the END command is necessary, especially in a control file which specifies ERRINT to be run. The END command leaves the terminal (on which INIT is running) logged into the system. When ERRINT runs, it uses the job slot (usually number 1) freed by the termination of INIT. ERRINT detaches itself and frees the terminal for other usage. The BYE command can be used in other control files which do not run ERRINT and which must log out the terminal running INIT.

It is also important that the crash control file perform operations similar to those done at start-up time. The following additional operations are recommended.

1. Run the ANALYS program to extract system data and error logging data from the CRASH.SYS file,
2. run ONLCLN to clean all disks, and
3. print the files created by ANALYS.

After these important operations are performed, the system can be initialized for normal time sharing operations.

The manner in which the start up and crash control files perform operations is also important. The control files should take advantage of INIT and RSTS/E features by performing operation in the following manner:

1. Do all forces of commands to the system console terminal (KB0:),
2. perform all routine operations in indirect command files, and
3. use INIT's ability to detach itself from the terminal which initiated it and to reattach itself to that terminal.

Because INIT can detach itself from the terminal which initiated it, all commands forced to that terminal are executed while INIT is running. Additionally, on all RSTS/E systems, a job can be logged into the system on the system console terminal regardless of the number of logins currently allowed.* Given

*As long as swapping space is available on the system, a user can log into the system on keyboard unit 0 regardless of the number of logins currently allowed. The system manager can change the terminal with this special privilege by installing a patch.

that the control file adds sufficient swapping space, subsequent commands in the control file can start as many jobs as required on the system console terminal without having logins enabled. When logins are not enabled (one of the start-up conditions), users cannot log into the system while start up and recovery operations are still in progress.

By placing routine operations in indirect command files, the system manager simplifies the main start up and crash control files. Changing the main procedure becomes a simple matter of changing one indirect command in the main control file. Isolating routine operations in a separate file also makes the flow of the main control file easier to follow.

3.1.2.1 Detaching and Attaching INIT — The DETACH command of INIT frees the terminal on which INIT was run and the ATTACH command reattaches INIT to that terminal. Because of the manner in which RSTS/E and INIT handle text forced to a terminal, these commands allow more flexible use of the terminal on which INIT was initialized. Particularly, the commands allow the system console terminal to be used to start system jobs and to perform other routine operations while INIT is still running and logins are not enabled.

The manner in which INIT handles FORCE commands depends on whether INIT itself is attached or detached. If INIT is attached, it checks whether the FORCE command is destined for the terminal on which it is running. If the destination is its own terminal, it ensures that a CTRL/C is not being forced. A CTRL/C forced to INIT's own terminal prematurely and perhaps unintentionally terminates INIT. Therefore, INIT discards all such FORCE commands, prints an error message at the terminal, and stops processing the control file immediately. The control file must be fixed and INIT rerun. When INIT is detached, it need not check the forced text because the detached job is unaffected by a CTRL/C response.

When INIT processes a FORCE command, it first checks the output buffers of the destination terminal. If the terminal is busy performing output, INIT waits until the buffers are cleared before actually executing the force operation. As a result of this procedure, output and input text at a terminal are not intermixed.

FORCE commands are executed according to the state of the destination terminal. A sequence of FORCE commands should be designed to exactly simulate manual input. The commands actually forced are executed exactly as if they were typed at a terminal. Commands forced to a terminal which is not logged into the system must either be valid logged out commands or a sequence of input which logs a terminal into the system. (The LOGIN command of INIT simply forces the proper sequence of text to log a job into the system.) If commands are forced to the terminal which initiated INIT and if INIT is attached to that terminal, the commands are not executed until the terminal is free (that is, at READY level or logged out).

Commands forced to the terminal which initiated INIT are executed according to whether INIT has freed the terminal (the DETACH command was executed) or INIT has terminated (by either the END or BYE command).

Because of these conditions, it is recommended that the routine operations done by INIT be performed on the system console terminal while INIT is detached. This procedure ensures that the operations are performed while INIT is still running and, more importantly, before INIT has executed the LOGINS command to enable users to gain access to the system. It is recommended that one FORCE command, the command to start ERRINT, be executed while INIT is attached. This latter step causes the FORCE command to be executed after INIT terminates. Because this step comes near the end of the control file, little delay exists between the time users receive the message that time sharing has started and the time INIT terminates allowing ERRINT to run. An additional benefit is that ERRINT occupies job slot number 1 on all systems and is therefore more easily visible to persons responsible for maintaining the system.

An additional recommendation is that each indirect command file perform the DETACH and ATTACH operations. This method of using the detach feature makes each indirect command file independent of the main control file. Because INIT must be attached to its terminal to properly terminate and to print error messages, this method also ensures that the main control file is always processed while INIT is attached.

3.1.2.2 Indirect Command Files — An indirect command file is an ASCII file containing INIT program commands which are executed when the file is specified in either a main control file or another indirect command file. The commercial at character (@) followed by a file specification causes INIT to close temporarily the current file it is processing, open the indirect command file, and process its commands. If no account is specified in the command, INIT searches for the indirect command file in the account under which the program is currently running. If a file extension is not given in the file specification, INIT attempts to open the file with an extension of .CMD. Upon completing the commands in the indirect command file, INIT closes the file, reopens the previous file, and continues processing the remaining commands.

An indirect command file need not contain an END or BYE command unless it is to be run as a main control file. If an END or BYE command is found in an indirect command file, INIT discards it and continues processing. This procedure means that INIT can be properly terminated only by an END or BYE command in the main control file.

It is recommended that indirect command files be used to perform the routine operations necessary at system start-up time. Each indirect command file to be run for system start up or crash recovery should perform the following operations:

1. Detach itself from the terminal on which INIT was run (usually the system console terminal).
2. Use the proper sequence of LOGIN and FORCE commands to perform the desired operations at the terminal from which INIT is detached.
3. Reattach INIT to its terminal.

This sequence of steps ensures the most efficient environment for INIT running during system start up. Using the system console terminal to perform

forces while INIT is detached means that logins need not be enabled and that all routine operations are performed while INIT is still running and logins are still disabled. INIT remains attached while commands in the main control file are being executed. Because an indirect command file contains only the LOGIN, FORCE, DETACH and ATTACH commands, less likelihood exists that INIT will encounter an unexpected error, attempt to output a message to its keyboard (while detached) and enter the hibernate state. (INIT prints errors on its console terminal. If INIT is always attached while the main control file is executing and the indirect command file contains only the recommended commands while INIT runs detached, the likelihood of the job entering the hibernate state is greatly diminished.)

3.1.3 Setting Terminal Characteristics – TTY.CMD

Because, at start-up time, the system sets the characteristics of all keyboard lines (except line number 0) to those of an ASR33-type terminal, the TTYSET program should be run to establish characteristics which match the local hardware. For example, if keyboard 1 is an LA36-type terminal running at 300 baud and keyboard 2 is a VT52 running at a split speed of 300/2400, the system manager can create an indirect command file to properly set the characteristics.

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN $TTYSET
FORCE KBO: KB1:; LA36
FORCE KBO: KB2:; VT52;SPLIT SPEED 300/2400
FORCE KBO: EXIT
FORCE KBO: BYE/F
ATTACH
```

The indirect command file detaches INIT, logs the system console terminal into the system under a privileged account, forces the necessary sequence of TTYSET commands, logs the terminal off the system, and reattaches INIT.

3.1.4 Spooling and Operator Services – SPOOL.CMD

It is convenient to start auxiliary jobs on the system by means of an indirect command file. The following sample command file shows the procedure.

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN $OPSER
FORCE KBO: OPE KBO:[2,2]
FORCE KBO: LOG OPSER.LOG;ALL
FORCE KBO: MES ALL
FORCE KBO: DET
LOGIN KBO: [1,2]
FORCE KBO: RUN $QUEMAN
FORCE KBO: DET
LOGIN KBO: [1,2]
FORCE KBO: RUN $SPOOL
FORCE KBO: LPO:/ASS/PRI:0/RUN:6
LOGIN KBO: [1,2]
FORCE KBO: RUN $BATCH
FORCE KBO: BAO:/PRI:0/RUN:6/QUE:LPO:/NODEL/ERROR:FATAL
ATTACH
```

The commands are in the proper sequence to run the spooling jobs. OPSER must be run first, followed by QUEMAN and any spooling programs to be started. For more information on the operator services and spooling programs, refer to Chapter 5.

3.1.5 Establishing Auxiliary Run-Time Systems – RTS.CMD

An indirect command file can be used to add auxiliary run-time systems at the start of time sharing. The following sequence of commands shows the procedure.

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN $UTILITY
FORCE KBO: ADD RT11
FORCE KBO: ADD RSX
FORCE KBO: ADD RMS11
FORCE KBO: ADD BP2COM
FORCE KBO: ADD BASIC2
FORCE KBO: EXIT
FORCE KBO: BYE/F
ATTACH
```

The ADD command of the UTILITY system program is sufficient to create the monitor structures necessary for auxiliary run-time systems. For more information on run-time system control, refer to Section 7.1.5.

3.1.6 Defining CCL Commands – CCL.CMD

Concise command language commands are most conveniently installed on the system at the start of time sharing. The following example shows the commands which the system programs are designed to handle.

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN [1,2]UTILITY
FORCE KBO: CCL ATT-ACH=[1,2]LOGIN.*;PRIV 30000
FORCE KBO: CCL BCK-=[1,2]RMSBCK.TSK;0
FORCE KBO: CCL BYE-=[1,2]LOGOUT.*;PRIV 0
FORCE KBO: CCL CNV-=[1,2]RMSCNV.TSK;0
FORCE KBO: CCL CRE-ATE=[1,2]EDIT.*;30000
FORCE KBO: CCL DEF-=[1,2]RMSDEF.TSK;0
FORCE KBO: CCL DFN-=[1,2]RMSDFN.TSK;0
FORCE KBO: CCL DIS-MOUNT=[1,2]UMOUNT.*;PRIV 30000
FORCE KBO: CCL DI-RECTORY=[1,2]DIRECT.*;PRIV 30000
FORCE KBO: CCL DSP-=[1,2]RMSDSP.TSK;0
FORCE KBO: CCL EDT-=[1,2]EDT.TSK;0
FORCE KBO: CCL ED-IT=[1,2]EDIT.*;30000
FORCE KBO: CCL HELLO-=[1,2]LOGIN.*;PRIV 0
FORCE KBO: CCL LBR-=[1,2]LBR.TSK;0
FORCE KBO: CCL LIB-R=[1,2]LIBR.SAV;8208
FORCE KBO: CCL LIN-K=[1,2]LINK.SAV;8208
FORCE KBO: CCL LOG-IN=[1,2]LOGIN.*;PRIV 0
FORCE KBO: CCL MACR-O=[1,2]MACRO.SAV;8216
FORCE KBO: CCL MAC-=[1,2]MAC.TSK;0
FORCE KBO: CCL MOU-NT=[1,2]UMOUNT.*;PRIV 30000
FORCE KBO: CCL PAT-=[1,2]PAT.TSK;0
FORCE KBO: CCL PIP-=[1,2]PIP.SAV;8208
```

```

FORCE KBO: CCL PL-EASE=[1,2]PLEASE.*;PRIV 30000
FORCE KBO: CCL QU-EUE=[1,2]QUE.*;PRIV 30000
FORCE KBO: CCL RST-=[1,2]RMSRST.TSK;PRIV 0
FORCE KBO: CCL SE-T=[1,2]TTYSET.*;PRIV 30000
FORCE KBO: CCL SRT-=[1,2]SRT.TSK;0
FORCE KBO: CCL SY-STAT=[1,2]SYSTAT.*;PRIV 30000
FORCE KBO: CCL TKB-=[1,2]TKB.TSK;0
FORCE KBO: CCL UT-ILTY=[1,2]JUTILTY.*;30000
FORCE KBO: EXIT
FORCE KBO: BYE/F
ATTACH

```

The abbreviations shown in the definitions of the CCL commands reflect the way the RSTS/E documentation describes the commands. The system manager may redefine the abbreviations but must not alter the fully defined command because each program is coded to recognize the fully expanded command.

NOTE

CCL commands are optional on all systems. Each command definition requires one small buffer. However, one system program, UMOUNT, can run only by CCL commands. To prevent users from running UMOUNT do not include the MOUNT and DISMOUNT commands on the system.

Two CCL commands duplicate commands recognized by the BASIC-PLUS Run-Time System: HELLO, and BYE. The commands are shown here to point out what BASIC-PLUS does by default. Other run-time systems might not recognize the commands HELLO and BYE if they do not exist as CCL commands.

3.1.7 System Start Up Control File – START.CTL

A complete system start-up control file contains the required INIT commands to perform initialization for time sharing and the proper indirect command file specifications to perform routine operations. The following sample shows a complete start-up control file.

```

@RTS.CMD
@TTY.CMD
@SPOOL.CMD
@CCL.CMD
FORCE KBO: RUN $ERRINT
FORCE KBO: 100
FORCE KBO: NO
LOGINS
SEND RSTS/E IS NOW ON THE AIR...
END

```

NOTE

The start up control files must be on the system disk (SY0:, the disk that was bootstrapped).

The sample control file specifies indirect command files that handle routine operations. When all routine operations are completed, the command to run ERRINT is forced to keyboard unit 0 and logins are enabled. The ON THE AIR message notifies users that time sharing has begun. The END command properly terminates INIT, and leaves keyboard unit 0 logged into the system so that the text forced to its input buffers will properly run ERRINT. ERRINT will detach from keyboard unit 0, thus freeing it for other use. ERRCPY will take job slot number 1 previously used by INIT.

3.1.8 System Crash Control File – CRASH.CTL

In performing a start up during crash recovery, the system runs INIT which automatically accesses the file CRASH.CTL in account [1,2] on the system disk (SY0:). The crash control file should look something like the following sample.

```
@RTS.CMD
@ANALYS.CMD
@CLEAN.CMD
@TTY.CMD
@SPOOL.CMD
@CCL.CMD
FORCE KBO: QUE ANALYS.DMP/DE
FORCE KBO: RUN $ERRINT
FORCE KBO: 100
FORCE KBO: YES
LOGINS
SEND RSTS/E IS NOW ON THE AIR...
END
```

The ANALYS program should be run immediately after a system crash to recover valuable diagnostic information. The ANALYS.CMD file should look like the following.

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN $ANALYS
FORCE KBO: [0,1]CRASH.SYS
FORCE KBO: ANALYS.DMP/WIDE
FORCE KBO: $ERRCRS.FIL
FORCE KBO: BYE/F
ATTACH
```

For information on ANALYS, refer to Chapter 6.

3.2 Performing System Shutdown — SHUTUP

The shutdown procedures for the RSTS/E system are critically important. If the system manager is not careful in performing system shutdown, valuable user data can be irretrievably lost. To fully understand shutdown procedures, the system manager must understand other RSTS/E system procedures. The system manager should study the concepts in this manual entitled "Operator Services and Spooling," "Principles of Disk Management," "Operational Control of the System," and "Monitoring System Status."

The system manager should use any of the available system utility programs, such as SYSTAT (described in the *RSTS/E System User's Guide*) or the system status display programs VT5DPY or VT50PY (described elsewhere in this manual), to verify that conditions are suitable for shutting down the RSTS/E system. By running any of these programs, the system manager can ascertain 1) the active jobs on the system and 2) the disks and devices in use.

The SHUTUP program can be run only from the system console keyboard (KB0:). Any attempt to run SHUTUP on another keyboard results in the following message:

```
? 'SHUTUP' can only be run from console terminal (KB0:)
Aborting shutdown run - Please try again later
```

The program is stored in its compiled form, in the system library account [1,2] with a protection code of <124>.

SHUTUP operates in either of two modes: 1) without OPSER and 2) with OPSER. The system manager should be familiar with the OPSER system utility and how it controls various utility programs (such as BACKUP and the spooling package programs).

When used in mode 1 (regardless of whether OPSER is running), the SHUTUP program treats all jobs in the system except itself and ERRCPY alike during the shutdown procedure. In mode 2, SHUTUP allows OPSER to shut down its controlled jobs before SHUTUP continues with the regular shutdown procedures.

As SHUTUP runs, it proceeds through the following ten phases:

1. Set-up dialogue phase
2. Warning message phase
3. Initial job killing phase
4. OPSER shutdown phase (optional)
5. ERRCPY shutdown phase (optional)
6. Final job killing phase (optional)
7. Unload and remove run-time systems and resident libraries phase
8. Swap file removal phase
9. Disk dismount phase
10. Final shutdown phase

The following sections describe these phases, the operator interactions required, and the expected as well as unexpected results of the shutdown operation. Several sample runs of SHUTUP show the results of shutting down the system with and without OPSER.

3.2.1 Set-Up Dialogue Phase

The set-up dialogue phase processes all questions relating to selection of options in running SHUTUP. Additionally, it checks for the presence of OPSER.

The system manager can respond to each question appearing during this phase with an **ESC** (ALTMODE) response. If such a response appears, the program returns to the previous question, which it reissues to the system manager. Thus, before typing the waiting period data, the system manager may reselect previous options.

If OPSER is present and running, the following question appears:

```
Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
```

Responses of the RETURN key, Y, YE, or YES allow SHUTUP to communicate with OPSER during the shutdown procedures.

A response of N or NO causes the following confirming query to appear:

```
Are you sure you don't want to use 'OPSER' (YES/NO) <NO>?
```

A response of the RETURN key, N or NO causes the program to repeat the previous query so that the system manager can properly express his desires. Otherwise, the program ignores OPSER during the shutdown procedures.

If OPSER is to be used, the following additional query appears:

```
Allow utilities to reach logical end point (YES/NO) <YES>?
```

If the response is the RETURN key, Y, YE or YES, the program directs OPSER to tell its on-line jobs to shut down their operation(s) at the next logical breakpoint in their job streams. For a line printer spooler, for example, this breakpoint occurs between queued jobs.

If the response is N or NO, the program directs OPSER to tell its on-line jobs to immediately abort whatever they are doing and cease operation. The system manager should use this latter mode only in cases of emergency; jobs being processed at the time of shutdown may be lost (unless the system manager has previously halted queuing and spooling operations and requeued such jobs).

Regardless of whether OPSER is present, the following question ends the phase:

```
Minutes until system shutdown (0-99) <5>?
```

The system manager now specifies the waiting period (in minutes) after which the actual shutdown procedure begins. This waiting period is called the warning message phase (described below) and may be as short as 0 minutes or as long as 99 minutes. The default value obtained by typing only the RETURN key is 5 minutes.

3.2.2 Warning Message Phase

At the beginning of the warning message phase, as well as in the beginning of all other phases that may be executed, a message appears on the system console terminal (KB0:) in the following format:

```
hh:mm{AM}  
(PM)dd-mmm-yy  
##### <phase title> #####
```

The time and date are printed with each phase title message. The hh and mm in the time format, hh:mm, represent the hour and minute the message was printed. Whether the message appeared before or after noon is indicated by the AM and PM abbreviations. The day, month, and year are included in the date format, dd-mmm-yy.

Immediately after the message for the warning message phase, SHUTUP prints a message on the system console terminal as follows:

```
Further LOGINs are now disabled
```

The RSTS/E shutdown procedure requires that the system limits on LOGINs be set to 1 before the monitor performs the final shutdown operation. SHUTUP lowers the LOGIN limit at the start of the warning message phase to prevent any new users from entering the system.

During the following phases, SHUTUP continuously checks that the LOGINs limit remains 1. If, for any reason, the limit changes, SHUTUP immediately aborts operations with the following message:

```
LOGINS ARE NOT EQUAL TO 1  
ABORTING SHUTDOWN RUN - PLEASE TRY AGAIN LATER
```

The system manager should ascertain the cause of the change, correct the problem, and run SHUTUP again.

During the warning message phase, the program issues warning messages indicating that the system is shutting down. These messages appear on most keyboards in the system, including pseudo keyboards. The following are the exceptions:

The OPSER operator services console (OSC)
The system console terminal (KB0:)

The program issues warnings after an appropriate waiting period, defined by the following formula:

waiting time = (total time left)/5 + 1 minutes

For an initial waiting period of 60 minutes, warning messages appear at 60, 47, 37, 28, 22, 17, 13, 10, 7, 5, 3, 2, and 1 minute before shutdown begins. The following message appears on the terminals at each warning time:

```
hh:mm{AM}  
(PM)dd-mmm-yy  
System going down in <waiting time> minutes, please finish up
```

At the same time, the following message appears on the system console terminal (KB0:):

<waiting time> minute message sent

When no more time is left for waiting (also if the initial waiting period is specified as 0), the final warning message appears:

```
hh:mm{AM}dd-mmm-yy **** FINAL WARNING!!!! System shutting down ****  
      {PM}
```

At this point, the program enters the next phase.

The time and date are printed for both the SYSTEM GOING DOWN and the FINAL WARNING messages. The hh and mm in the time format, hh:mm, represent the hour and minute the message was printed. Whether the message appeared before or after noon is indicated by the AM and PM abbreviations. The day, month, and year are included in the date format, dd-mmm-yy.

3.2.3 Initial Job Killing Phase

In the initial job killing phase, SHUTUP begins to clear the system of currently active jobs. If SHUTUP finds that there is only one job running on the system (itself), it skips this phase then continues the shutdown procedure at the unload and remove run-time systems and resident libraries phase (Section 3.2.7).

If OPSER is to be used in the next phase (OPSER shutdown phase, Section 3.2.4), the program ignores, during this phase, all jobs currently associated with OPSER (as indicated by entries in OPSER's on-line job table). Additionally, it ignores any job whose primary keyboard (KB:) is a pseudo keyboard because it is probably being controlled by one of the BATCH programs.

SHUTUP further divides all other jobs active in the system into two classes, attached and detached. This classification depends on whether they have an attached primary keyboard (KB:). For all attached jobs, SHUTUP now forces the following text string into the keyboard input buffer:

```
^ C ^ C BYE/Y (RET)
```

For all detached jobs, SHUTUP issues the kill SYS call to remove the job.

SHUTUP makes two passes through the current active job table during this phase. In the first pass, the program terminates in the appropriate manner all active jobs not being ignored. At the end of the first pass, it establishes the number of attached jobs that were found and forced to logout. SHUTUP then waits for a specified period to allow the LOGOUT program to complete operations on each of the attached jobs.

After the waiting period (if any) expires, the program makes a second pass through the active job table. During this pass, all jobs not being ignored are removed via the kill SYS call. At the end of the pass, the program checks the table. If any jobs that should have been killed were not, SHUTUP aborts operations with the following error message:

```
'SHUTUP' Failed in initial Job Killing Phase  
Aborting shutdown run - Please try again later
```

The system manager should ascertain the cause of all jobs not being killed, correct the problem, and run SHUTUP again.

If all jobs are successfully killed during the phase, or if at any time during either pass the number of active jobs becomes one (SHUTUP only), the next phase begins.

3.2.4 OPSER Shutdown Phase (Optional)

The program skips the OPSER shutdown phase if 1) OPSER is not running on the system or 2) the system manager elects not to use OPSER during the shutdown procedures.

If OPSER is to be used, the following actions occur. SHUTUP immediately detaches from the system console terminal (KB0:) to enable OPSER to reattach to the terminal for use during OPSER's shutdown procedures. As the detach occurs, the following messages appear:

```
DETACHING...
```

```
'OPSER' ATTACHING  
#
```

SHUTUP sends a message to OPSER directing it to begin its shutdown procedures by selecting the appropriate mode of OPSER shutdown. SHUTUP then waits for 60 seconds to see if OPSER receives the message and/or is properly functioning. If OPSER does not respond properly within that time, SHUTUP attempts to reattach to the system console terminal (KB0:). If it is successful, the following message appears:

```
'OPSER' not active  
Aborting shutdown run - Please try again later
```

If the terminal is attached or assigned, SHUTUP waits for 1 second and tries again. The system manager should take whatever steps are necessary to free the system console terminal (KB0:) for use so SHUTUP can complete its activities.

If OPSER is shutting down its on-line programs in the immediate mode, it sends each program the appropriate message to cease operations. OPSER waits a certain amount of time to ensure that all programs have completed their assignments. OPSER then closes its files, outputs the following message on the system console (KB0:), and kills itself:

```
'OPSER' TERMINATING
```

SHUTUP allows OPSER 120 seconds to complete its shutdown in the immediate shutdown mode. If unsuccessful, SHUTUP aborts its operations, signaling the failure as follows:

```
'OPSER' Shutdown taking too long  
'SHUTUP' Aborting shutdown operation
```

If OPSER is shutting down in the logical end mode, it proceeds to shutdown successive levels in its on-line job table. Appropriate system console messages (see examples) signal each job's completion. When all on-line OPSER jobs are gone, OPSER closes its files and kills itself, freeing the system console for SHUTUP use again. OPSER signals its completion with the message:

```
'OPSER' TERMINATING
```

SHUTUP allows 60 minutes in the logical end mode for OPSER to complete operations. If the procedure is not successful, SHUTUP aborts operations with the error message described previously.

Otherwise, the shutdown of OPSER-related jobs and activities is complete and the program enters the next phase.

3.2.5 ERRCPY Shutdown Phase (Optional)

The program enters the ERRCPY shutdown phase if the system error logging utility ERRCPY is present. SHUTUP sends ERRCPY a special error message (SH error) which causes ERRCPY to log the shutdown occurrence in the system error log, close the file, and kill itself.

When SHUTUP detects that ERRCPY has completed its shutdown and is no longer present, SHUTUP proceeds to the next phase. SHUTUP allows 60 seconds for ERRCPY to complete its tasks, and if failure occurs, SHUTUP aborts operations with the following error message:

```
'ERRCPY' Failed to shutdown  
Aborting shutdown run - Please try again later
```

3.2.6 Final Job Killing Phase (Optional)

If SHUTUP reaches this point, and is not the only job left running in the system, SHUTUP makes one last attempt at killing all other remaining jobs. If SHUTUP is successful, it enters the next phase. If not successful, SHUTUP aborts operations with the following error message:

```
'SHUTUP' Failed in final Job Killing Phase  
Aborting shutdown run - Please try again later
```

3.2.7 Unload and Remove Run-Time Systems (RTS) and Resident Libraries Phase

During the unload and remove Run-Time System and resident libraries phase, the program unloads from memory all Run-Time Systems and resident libraries. It then removes them from the system tables. SHUTUP does not attempt to remove the system default run-time system.

3.2.8 Swap File Removal Phase

During the swap file removal phase, SHUTUP ensures that all open swapping files are closed and released from the system.

3.2.9 Disk Dismount Phase

During the disk dismount phase, the program dismounts from the system, all mounted disks, whether public or private. The system disk always remains mounted.

If the system manager runs SHUTUP in its source form (.BAS) by mistake, SHUTUP may encounter an unexpected error. If the temporary file (TEMPnn.TMP), created by the run-time system and associated with SHUTUP, resides on a public disk that is not the system disk, an unexpected error occurs when SHUTUP tries to dismount the affected disk. SHUTUP aborts and the system manager must compile the program before retrying the shutdown procedure from the beginning.

Unexpected errors (not related to hardware problems) should be reported through the normal SPR mechanism.

3.2.10 Final Shutdown Phase

SHUTUP enters the final shutdown phase to output the last status message, clears the system console terminal (KB0:) buffers, and kills itself. When SHUTUP kills itself, the monitor transfers control to the system initialization code (INIT.SYS). The monitor then bootstraps the system disk. The initialization code is loaded and the OPTION: question is printed.

3.2.11 SHUTUP Operation Examples

The following examples show three sample operation runs of SHUTUP. The first example shows a system shutdown when OPSER is not present. This example, except for the absence of the first OPSER-related question, would apply when OPSER is present but not used during shutdown.

```
RUN $SHUTUP
SHUTUP V7.0 RSTS V7.0 TIME SHARING

12:24 PM 22-Sep-78 ***** Set-up Dialogue Phase *****
Type 'ESC' ('ALT') to any query to backup one (1) step
'OPSER' not running
Minutes until system shutdown (0-99) <5>? 0

12:24 PM 22-Sep-78 ***** Warning Message Phase *****
Further LOGINs are now disabled

12:24 PM 22-Sep-78 ***** Initial Job Killing Phase *****
12:24 PM 22-Sep-78 ***** Remove RTS/RES LIB Phase *****
12:24 PM 22-Sep-78 ***** SWAP File Removal Phase *****
12:24 PM 22-Sep-78 ***** Disk DISMOUNT Phase *****
12:24 PM 22-Sep-78 ***** Final Shutdown Phase *****
Please wait for system to re-boot itself

RSTS V7.0 TIME SHARING (DM1)

Option:
```

The next example shows a sample run with SHUTUP where OPSER is used to shutdown a spooling system of five spoolers.

```
Ready

RUN $SHUTUP
SHUTUP V7.0 RSTS V7.0 TIME SHARING

12:39 PM 21-Sep-78 ***** Set-up Dialogue Phase *****
Type 'ESC' ('ALT') to any query to backup one (1) step
Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
Allow utilities to reach logical end point (YES/NO) <YES>?
Minutes until system shutdown (0-99) <5>?

12:39 PM 21-Sep-78 ***** Warning Message Phase *****
Further LOGINs are now disabled
5 minute warning message sent
```


3 minute warning message sent
2 minute warning message sent
1 minute warning message sent

12:45 PM 21-Sep-78 ***** Initial Job Killing Phase *****

12:45 PM 21-Sept-78 *****
'OPSER' Shutdown Phase *****
Detaching...

'OPSER' ATTACHING

*
JOB #6 'BA1SPL' TAKEN OFFLINE

*
JOB #5 'BA0SPL' TAKEN OFFLINE

*
MESSAGE 28 : 21-Sep-78 12:45 PM JOB:3 DET
QUMRUN[1,10]
BA1SPL (6) REQUESTED OFF-LINE -- TAKEN OFF-LINE

*
MESSAGE 29 : 21-Sep-78 12:45 PM JOB:3 DET
QUMRUN[1,10]
BA0SPL (5) REQUESTED OFF-LINE -- TAKEN OFF-LINE

*
JOB #4 'LPOSPL' TAKEN OFFLINE

*
MESSAGE 30 : 21-Sep-78 12:45 PM JOB:3 DET
QUMRUN[1,10]
LPOSPL (4) REQUESTED OFF-LINE -- TAKEN OFF-LINE

*
JOB #3 'QUEMAN' TAKEN OFFLINE
*'OPSER' TERMINATING

Re-attaching

12:45 PM 21-Sep-78 ***** 'ERRCPY' Shutdown Phase

12:45 PM 21-Sep-78 ***** Remove RTS/RES LIB Phase

12:45 PM 21-Sep-78 ***** SWAP File Removal Phase

12:45 PM 21-Sep-78 ***** Disk DISMOUNT Phase *****

12:45 PM 21-Sep-78 ***** Final Shutdown Phase *****

Please wait for system to re-boot itself

RSTS V7.0 TIME SHARING (DB0)

Option:

In the last example of SHUTUP, OPSER is used to shutdown the five spooler system in the immediate mode.

```
RUN $SHUTUP
SHUTUP  V7.0 RSTS V7.0 TIMESHARING

12:11 PM 22-Sep-78 ***** Set-up Dialogue Phase *****

Type 'ESC'('ALT') to any query to backUp one (1) step

Use 'OPSER' for utilities shutdown (YES/NO) <YES>?
Allow utilities to reach logical end point (YES/NO) <YES>?
Minutes until system shutdown (0-99) <5>?

12:11 PM 22-Sep-78 ***** Warning Message Phase *****
Further LOGINs are now disabled
5 minute warning message sent

3 minute warning message sent
2 minute warning message sent
1 minute warning message sent

12:12 PM 22-Sep-78 ***** Initial Job Killing Phase *****

12:12 PM 22-Sep-78 ***** 'OPSER' Shutdown Phase *****
Detaching...

'OPSER' ATTACHING
*
'OPSER' TERMINATING

Re-attaching...

12:12 PM 22-Sep-78 ***** 'ERRCPY' Shutdown Phase
*****

12:12 PM 22-Sep-78 ***** Final Job Killing Phase
*****

12:12 PM 22-Sep-78 ***** Remove RTS/RES LIB Phase *****
***** SWAP File Removal Phase
*****

***** Disk DISMOUNT Phase *****
***** Final Shutdown Phase *****

Please wait for system to re-boot itself

RSTS V7.0 TIME SHARING (DM1)

Option:
```

3.2.12 Notes on SHUTUP Operation

SHUTUP is designed to shutdown a RSTS/E system in the correct manner under all normal circumstances.

All unaccountable errors in operation (or unexpected errors not included in this description or in the descriptive text of the SHUTUP program source file itself) should be reported immediately to DIGITAL using the SPR mechanism.

The system manager must establish administrative procedures governing RSTS/E system operational hours. All users can be kept informed by means of the NOTICE.TXT file, which is printed when a user logs into the system. If system shutdown times are fixed, users can plan a work load and properly complete their processing within the allotted hours of scheduled time sharing.

SHUTUP must be running under the system default run-time system; otherwise it will abort. This occurs when it tries to remove the run-time system it is running under.



Chapter 4

Account Creation and Account Statistics

4.1 Creating and Deleting User Accounts – REACT

The system manager or a privileged user creates and deletes accounts by use of the REACT system program. The REACT system program enters user accounts on and deletes user accounts from either the system device or individual private disk devices.

REACT is called by using the RUN command as follows.

```
RUN $REACT
```

REACT responds by printing the following message which requests that the user specify a function.

```
REACT V7.0 RSTS V7.0 TIME SHARING  
SYSTEM ACCOUNT MANAGER  
FUNCTION?
```

The three valid functions are described in Table 4-1 and explained in the following sections.

4.1.1 Creating Individual Accounts – ENTER Function

The ENTER function creates individual user accounts. When the system manager runs REACT, he invokes the ENTER function by typing E in response to the request for a function. Upon recognition of the E response, REACT prints a series of questions. The questions are explained in Table 4-2.

Table 4-1: REACT System Program Functions

Function	Abbreviation and Purpose
ENTER	<i>E</i> To enter individual accounts on a disk.
DELETE	<i>D</i> To delete individual accounts from a disk.
STANDARD	<i>S</i> To create standard user accounts on the system disk from the ACCT.SYS file at system generation time.

The following is a sample dialogue for the ENTER function.

```
RUN $REACT
REACT V7.0 RSTS V7.0 TIME SHARING
SYSTEM ACCOUNT MANAGER
FUNCTION? E
PROJ,PROG? 100,100
DISK:PASSWORD? DEMO
QUOTA? 500
CLUSTER SIZE? 0
ACCOUNT NAME? A NON-PRIVILEGED ACCOUNT
PROJ,PROG? CTRL/Z
```

READY

If the system manager enters an account on a private disk, the owner of the account can create files on that disk. The pack must be logically mounted and placed in the unlock state by means of the UTILITY or UMOUNT system program described in Section 7.1.2.

If the system manager enters an account on the system disk, he permits the owner of that account access to the RSTS/E system and use of storage space within the public structure. When a new account is created, REACT also places an entry for the new account in the ACCT.SYS file.

Table 4-2: Responses to ENTER Function Queries

Question	Response Format and Meaning
PROJ,PROG?	<p><i>n,m</i> The user account number to be entered in the MFD, where $1 \leq n \leq 254$ and $0 \leq m \leq 254$.</p> <p><u>CTRL/Z</u> The user terminates the dialogue and REACT by typing the CONTROL key and Z combination simultaneously.</p>
DISK:PASSWORD?	<p><i>passwd</i> To enter a <i>passwd</i> for an account on the system disk MFD, where <i>passwd</i> is from 1 to 6 alphanumeric characters. No value for DISK need be specified since the system disk is assumed.</p> <p><i>dev:</i> To create an account on a private disk, where <i>dev</i> is the device designator, specify the device name and unit number. For example:</p> <p style="text-align: center;">DK1:</p> <p>The disk specified must be logically mounted and placed in the unlock state by means of the UTILTY or UMount program prior to invoking REACT.</p>
QUOTA?	<p><i>n</i> The number of 256-word blocks of disk storage the user account is allowed to retain at LOGOUT time where $0 \leq n \leq 65535$ and 0 means no quota is imposed upon the user's account. Therefore, a value of 0 limits disk storage retention only by the amount imposed by the UFD cluster size and the amount of disk storage space available. LOGOUT is the only DIGITAL-supplied program that checks the quota and monitors storage retention on the public structure.</p>
CLUSTER SIZE?	<p><i>n</i> The account UFD cluster size where <i>n</i> is 0, 1, 2, 4, 8, or 16. If 0 is specified, the pack cluster size is used. If non-zero, the value for <i>n</i> must be at least the pack cluster size. Cluster sizes of 1, 2, or 4 are recommended for most accounts. The maximum number of files a user is allowed to create is approximated by multiplying the UFD cluster size by 72.</p>
ACCOUNT NAME?	<p><i>x</i> Optional. Used by the GRIPE system program to identify account owner.</p>

4.1.2 Deleting Accounts – DELETE Function

The DELETE function removes individual user accounts from the system disk or from a private disk. As in the case of the ENTER function, a private disk must be logically mounted and in the unlock state prior to deletion of an account. In addition, before an account is deleted, the UFD of that account must contain no files. To remove all files from the UFD, the ZERO command of the UTILTY system program or the /ZE switch of the PIP system program must be used. See the description of the UTILTY system program in Chapter 7 of this guide and of PIP in the *RSTS/E System User's Guide*.

The DELETE function is invoked by typing D in response to the request for a function. The REACT program prints a series of questions explained in Table 4-3.

Table 4-3: Responses to DELETE Function Queries

Question	Response Format and Meaning
PROJ,PROG?	<p><i>n,m</i> The user account number to be deleted from the MFD.</p> <p>CTRL/Z The user terminates the dialog and REACT by typing the CONTROL key and Z key combination simultaneously.</p>
DISK?	<p><i>null</i> By typing the RETURN key, the system device is specified.</p> <p><i>dev:</i> The <i>device</i> designator of a private disk. For example:</p> <p style="text-align: center;">DK1:</p> <p>The disk must be logically mounted and placed in the unlock state by means of the UTILITY or UMount system program prior to running REACT.</p>

The following is a sample dialogue for the DELETE function.

```

RUN $REACT
REACT V7.0 RSTS V7.0 TIME SHARING
SYSTEM ACCOUNT MANAGER
FUNCTION? D
PROJ,PROG? 100,100
DISK? DK1:
PROJ,PROG? ^Z

READY

```

4.1.3 Automatic Account Creation – STANDARD Function

The STANDARD function in the REACT system program facilitates automatic creation of a large number of user accounts when the system disk is built. A detailed explanation of the STANDARD function is contained in the *RSTS/E System Generation Manual*. A few ancillary remarks are made here.

The file ACCT.SYS is stored in the system library account [1,2]. ACCT.SYS is an ASCII text file, each line of which contains, separated by commas, the items which would be specified by the user in response to the questions of the ENTER function. Each line of the file represents a single account to be created. The general format is as follows:

proj,prog,passwd,quota,cluster,name

The items proj, prog, passwd, quota, and cluster are described under the ENTER function. The name can contain any additional information about the account such as the owner's name. The name is not used by REACT but is

used by GRIPE. The name must contain no commas, single quotes, or double quotes. The accounts [1,1] and [1,2] can appear in the ACCT.SYS file although they have been created previously during the REFRESH action of the system generation procedure. These account entries in ACCT.SYS are used by the GRIPE system program and by the standard option of REACT.

4.2 Performing System Accounting Operations – MONEY

The MONEY system program enables the system manager to extract system accounting information for all accounts in the system or for selected accounts. MONEY can be run by a nonprivileged user to obtain his own account information (excluding password) as described in the *RSTS/E System User's Guide*.

MONEY is called by typing the following command while logged into the RSTS/E system:

```
RUN $MONEY
MONEY V7.0 RSTS V7.0 TIME SHARING
SYSTEM ACCOUNTING PROGRAM
```

If the caller is a privileged user, a sequence of option questions is printed at the keyboard. The questions and the explanation for each are given in Table 4-4. The account data given as output for each account is described in Table 4-5.

The MONEY system program can be run during normal time sharing. No conflicts arise if the system attempts to update a user's accounting information while the MONEY program is accessing it. When the RESET option is used, MONEY reads and resets to zero the user's accounting information before any system action can update the values being read and reset. Thus, no user accounting information is lost.

Some of the items output can be used to weigh billing or evaluate usage. The item KCT, in effect, reflects system usage more accurately than CPU-TIME. For example, two users may each exhaust one minute of CPU-TIME in an accounting period. However, one user may tie up 2K words of memory while the other may occupy 6K words. The first user's KCT value is incremented by 2 for each tenth of a second the 2K job is running. With the 6K user, each tenth of a second causes the KCT value to be incremented by 6. The 6K user is tying up more system resources and this is reflected in the higher KCT value. Thus, a user's average job size can be roughly calculated by dividing the number of KCT's reflected for the accounting period by the number of tenths of seconds derived from the value of CPU-TIME. Referring to the example of values in Table 4-5, user [100,100] average job size of 3.6K is computed by dividing the number of KCT's, 3000, by the number of tenths of seconds derived from CPU-time, 832.

Table 4-4: MONEY Program Options

Option Question	Reply and Explanation
OUTPUT DEVICE?	<p><i>dev:filnam.ext</i> A file structured or non-file structured device can be specified. Indicating a disk file reduces processing time. For example:</p> <p style="text-align: center;">MONEY.DMP</p> <p>MONEY writes the data to the file which can later be queued for printing.</p> <p><u>RET</u> Output is printed at the terminal.</p>
PRINT PASSWORDS?	<p>YES Typing YES causes passwords to be printed. Typing anything else omits passwords.</p>
RESET?	<p>NO Typing NO causes the accumulated accounting data to be preserved.</p> <p>YES Typing YES causes the following items to be reset to zero.</p> <p style="text-align: center;">CPU-TIME KCT's CONNECT TIME DEVICE TIME</p> <p>The data is reset after the information is dumped.</p>
DISK?	<p><i>dev:</i> Type the specific disk device designator with unit number to select the accounting data from a private pack.*</p> <p><u>RET</u> The accounting data selected is for the public structure.</p>
SELECTIVE?	<p>NO Typing NO causes accounting data for all accounts to be dumped.</p> <p>YES Typing YES causes an additional question ACCOUNT?</p>
ACCOUNT?	<p><i>n,m</i> The ACCOUNT question appears if the reply to the SELECTIVE? question is YES. Accounting data is dumped for the account specified by the project-programmer number [n,m], following which, the question is repeated.</p> <p><u>CTRL/C</u> or <u>CTRL/Z</u> Typing CTRL/C or CTRL/Z terminates the program.</p>
<p>*Meaningful accounting data on a nonsystem disk is account number, number of blocks occupied, disk quota, and UFD cluster size. The system updates only the system disk MFD with CPU time, KCT's, connect time, and device time.</p>	

Table 4-5: MONEY Program Output

Header	Example and Description
ACCT	100,100 Project-programmer number (account).
PASSWORD	DEMO Account password used at login time.
CPU-TIME	1:23.2 (one minute, 23.2 seconds Number of hours:minutes:seconds.tenths of a second of processor time the account has used since the last reset.
KCT's	3000 Memory usage factor (kilo-core-ticks). One KCT is the usage of 1K of memory for one tenth of a second.
CONNECT	2:34 (2 hours, 34 minutes Number of hours and minutes (hh:mm) of terminal connect time.
DEVICE	20 (20 minutes) Number of hours and minutes (hh:mm) of device usage time, excluding disks.
DISK	100 Number of 256-word blocks of disk storage allocated.
QUOTA	500 Number of 256-word blocks the account is allowed to retain at logout time.
UFD	2 UFD cluster size.

The value under the header description DISK reflects the actual number of blocks tied up in file allocation on disk.* It is not necessarily the same value reported by the BASIC-PLUS CATALOG or the DIRECTORY CCL command. A file may occupy 1 block on a disk as reflected by the CATALOG command, but may tie up 3 additional blocks of disk storage if the file cluster size is 4 blocks. In essence, the user is allocated four contiguous blocks although the file is currently occupying only one block.

The information given under the header UFD is the User File Directory cluster size which the system manager specifies when the account is created by REACT. The cluster size is provided in the MONEY output for information purposes only and has no accounting value.

It is advised that the system manager periodically execute the RESET option to prevent overflowing the accounting values stored on disk. The following list

*This is true if the block count is less than 65,535. When the number of blocks allocated is greater than or equal to 65,535 blocks, Money does not report the precise block count but instead prints the value $\geq 65,535$ under the DISK heading.

shows the maximum times that can be stored for each statistic without an overflow.

Time	On Disk	In Memory
Device	1092 hours	1092 hours
Connect	1092 hours	68 hours
CPU	116 hours	29 hours
KCT's	116 hours at 16K words	29 hours at 16K words

At logout time, the system updates the values on the disk with accumulated values from memory. Thus to prevent loss of accounting data, the user must log a job off the system before any of the values in memory overflow. In like manner, the system manager must execute the RESET option before the values on disk overflow. The sizes of the accounting data fields on disk allow approximately one week of continuous system operation without overflow. Therefore, MONEY must be run with the RESET option at least once per week. The MONEY program indicates reset is in effect by adding the text WITH DATA BEING RESET to the accounting printout header line.

Chapter 5

Operator Services and Spooling

This chapter describes operator services programs and operations involving interaction with operator services. The programs, except for BACKUP, are completely described in this chapter. Most of the programs require permanent privilege to run. General aspects of related user programs, such as QUE and BATCH, are mentioned only briefly in this chapter but are described fully in the *RSTS/E System User's Guide*.

5.1 Overview of Operator Services

Operator services on RSTS/E involve the OPSER program and controlled programs in OPSER tables. Controlled (on-line) programs are QUEMAN, SPOOL, BATCH and BACKUP. SPOOL and BATCH are spooling programs executing queued requests for either a line printer device or batch processor. QUEMAN is the queue manager program which passes queued requests to spooling programs and adds, updates, and deletes requests in the various queues. BACKUP is the system data saving and restoring package which can optionally be controlled through OPSER.

The OPSER program establishes interjob communication on which all on-line programs depend. OPSER declares itself a message receiver with a system-wide identification which the controlled programs can recognize. When a controlled program starts, it declares itself a message receiver and supplies OPSER with certain data by means of the system message send/receive SYS call. OPSER places the program in its table of on-line jobs.

The identification with which OPSER communicates with on-line programs is called the message receiver identification, or simply, the receiver identification. Each identification exists in the system message receiver table and must be unique. Because of this uniqueness, only one copy of OPSER can be running on a system. Additionally, because all control and interjob communication of spooling jobs depends on OPSER, the operator services program must be running before any controlled jobs can be run.

After initial interjob communication is established, OPSER provides the means by which an operator can interact with the controlled jobs. OPSER becomes the main interface between the operator and a system controlled program. The interaction among system controlled programs and the operator is summarized in Figure 5-1.

OPSER broadcasts information to the operator through a terminal designated as the Operator Services Console (OSC). For flexibility, the operator is allowed to alter the OSC to any keyboard line on the system. To isolate the keyboard control from unwarranted tampering, OPSER itself need not be connected to the OSC but may broadcast data on the physical keyboard line. Given that intermittent output is generated on the OSC, any other user, including the operator, could be logged into the system on that terminal.

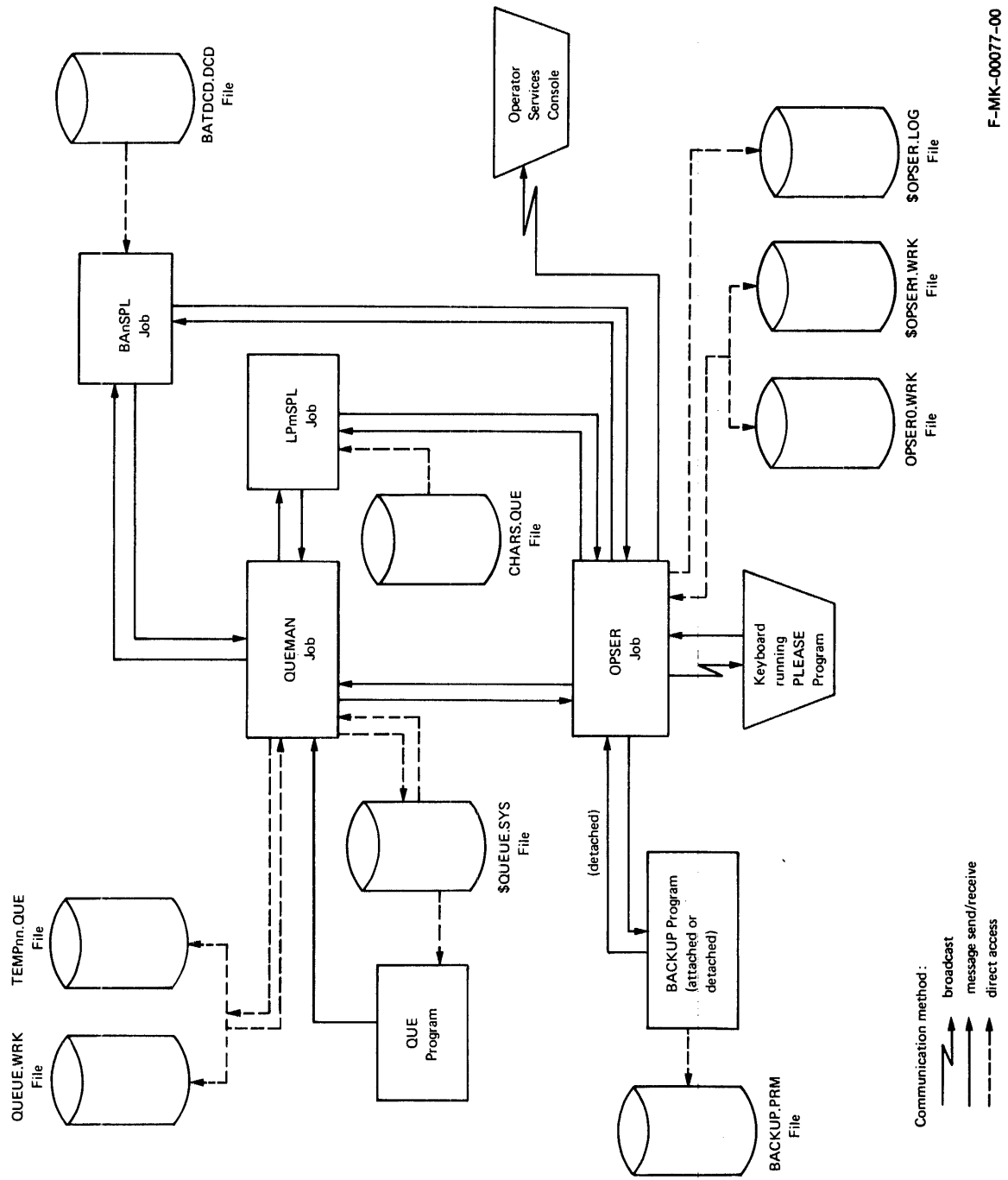
OPSER itself establishes a data base by which it controls on-line jobs. The data base resides in the system library account [1,2] and consists of the work files OPSER0.WRK and OPSER1.WRK and the log file OPSER.LOG. OPSER0.WRK contains the current activities and message control directory tables. OPSER1.WRK has the table of jobs on line to OPSER, the valid operator table, and tables of messages and legal commands. The file OPSER.LOG, which can be optionally renamed, provides a history of operating activity.

To provide a nonvolatile data base during a time-sharing session and to provide continuity of operations between time-sharing sessions, OPSER stores the work file information on disk. Thus, if the system crashes or if OPSER alone unexpectedly terminates, the work files retain the most current processing information. Similarly, these files maintain data continuity from one time-sharing session to another. Whenever OPSER starts, it examines its files to determine whether entries in the on-line job and valid operator tables are still valid. Generally, OPSER retains those active and valid entries and purges those entries which do not satisfy validation requirements. Upon restarting, therefore, OPSER attempts to preserve data from its previous operational state.

The operator most conveniently interacts with OPSER through the PLEASE program. PLEASE transmits commands to the OPSER program. OPSER checks that the sender is a valid operator and, if so, takes the requested action. Because of PLEASE, the operator need never directly run or attach to OPSER to perform operator functions on controlled jobs.

The QUEMAN program manages the system queue file QUEUE.SYS in the system library account. The file retains all pending user requests and can store requests between time-sharing sessions. Users make requests of spooling programs through the QUE system program. To form a request for a user, QUE builds a message and sends it to QUEMAN which updates the QUEUE.SYS file. QUEMAN maintains a table of on-line spooling programs in a work file QUEUE.WRK in the system library account. The program passes a queued request to a destination spooling job not currently busy with a request. As a spooling job completes a request, it notifies QUEMAN, which updates the queue and work files and checks for another request to pass to the spooling job.

Figure 5-1: System Controlled Programs and Operator Interaction



F-MK-00077-00

A spooling program typically handles one request at a time from QUEMAN. If no request is pending for a spooling program, the job enters an indefinite sleep state. The job is not awakened until the system queues a message for it. The message is the result of QUEMAN or OPSER transmitting data to the spooling job by means of the system message send/receive mechanism.

To retain as much data as possible between time-sharing sessions, QUEMAN, like OPSER, examines its files upon restarting. After restarting, QUEMAN should resume processing with no loss of data from its previous run. QUEMAN examines the queue file structure and contents and re-establishes synchronization with any spooling jobs already on line and active.

The SPOOL program handles requests made for line printer output. SPOOL maintains dual communication paths: one with QUEMAN and one with OPSER. QUEMAN merely transmits a queued request to SPOOL. SPOOL interacts with QUEMAN only to notify it that a transmitted request is completed. Under normal circumstances, communication between SPOOL and OPSER is necessary only when an operator requires some special action (such as aligning forms) of the spooling job.

To identify output of a user's queued request, SPOOL prints heading burst pages. The program accesses the character generation file CHARS.QUE in the Spooling Package Library to form the large, easily readable letters necessary to distinguish discrete requests and files within a request. The CHARS.QUE file is generated during the system library build procedure of system generation. It can be rebuilt by following the procedures described in Appendix C.

The BATCH program executes user requests by running a job for a user on a pseudo keyboard. BATCH eliminates the requirement for a physical terminal by using a pseudo keyboard and relieves the user from typing by processing commands from a disk file.

BATCH, like SPOOL, maintains dual communication paths: one with QUEMAN and one with OPSER. QUEMAN sends a message to BATCH to process a job request. BATCH interacts with QUEMAN only to notify it that a transmitted request is completed. BATCH typically depends on OPSER to form requests for operator action. To mount a user volume, for example, BATCH notifies OPSER which, in turn, generates the request for the operator and processes the response from the operator.

To process commands from a user queued disk file, BATCH requires the command decoding file BATCH.DCD in the Spooling Package Library. This file is generated during the system library build procedure of system generation. It can be rebuilt by following the procedures described in Appendix C. The BATCH.DCD file contains all the codes and parameters BATCH requires to decode user-specified control statements, perform syntax checking, and form a special intermediate file (BANJmm.COM) of encoded BATCH commands.

Both the SPOOL and BATCH programs charge processing (execution) time to the account under which the user queues the request. Overhead time, expended to set up a request, is charged to the account under which the program is running.

The BACKUP program can be controlled by operator intervention through OPSER. If BACKUP runs detached, control through OPSER is the only means by which the operator can communicate with a BACKUP job. BACKUP, when detached, transmits all operator requests through OPSER and receives responses only from OPSER.

OPSER provides processing checks for on-line jobs. OPSER ensures that all on-line jobs are still active, are still valid receivers, and are not hibernating. If any job is found in the hibernate state, OPSER notifies the operator who can attach the job to a terminal, remedy the cause of the hibernation, and/or restart the job.

OPSER interaction between a controlled job and the operator is recorded in one of three ways: as a message, as an action request, or as an information line. A message is specifically formatted data to which OPSER assigns a sequence number for operator reference. An action request is a special message which requires operator action and a response before a controlled job can resume processing. A request usually involves both performing an action (such as mounting a volume) and typing an answer (such as a device designator or program interrupt command) in direct response to the request. An information line is a free form report of some internal operation performed. Information lines are typically not important to the operator function but merely provide a history of on-line events.

The entire operator services package is designed to be terminated in a controlled manner using what is called shutdown level. The defined shutdown level for each job determines the order in which OPSER terminates controlled jobs during system shutdown. The SHUTUP system program can be run so that OPSER terminates controlled jobs with the lowest shutdown level first. OPSER informs each job at a certain level to complete processing at a logical end point. Only after OPSER shuts down all jobs at a given level, does it proceed to shut down all jobs at the next highest level. SHUTUP itself does not proceed with final system shutdown until OPSER terminates all its controlled jobs and kills itself. By using the controlled shutdown of OPSER, the system manager ensures continuity of processing user requests from one time sharing session to another.

5.2 Operator Services Program – OPSER

The operator services program OPSER consists of two modules OPSER and OPSRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module establishes the initial conditions on start up, checks initial conditions upon restart, and chains to the second module. The second module actually executes commands and periodically checks status. For simplicity, the documentation refers only to one program, OPSER, which functionally includes the two modules.

The OPSER program must be run before the queue manager and spooling programs can be run. To start OPSER, the following command is typed at a terminal logged into the system under a privileged account.

```
RUN $OPSER
OPSER V7.0 RSTS V7.0 TIME SHARING
*
```

If the account is not privileged, the system prints the ?PROTECTION VIOLATION and ?PROGRAM LOST-SORRY messages. If the account is privileged, an identification line is printed at the terminal to indicate that the program is running.

The OPSER programs must reside in the Spooling Package Library (SPL) in executable form. If the system manager tries to invoke an OPSER program from the SPL account and the program is not in executable form, OPSER prints the following message.

```
?OPSER must be compiled
```

The message warns the system manager that the program was not executable. Programs that are executable usually have a .BAC or .TSK extension. After the message is printed, the system manager must load a compiled version of the OPSER program into the Spooling Package Library account and invoke the program again. The ?OPSER MUST BE COMPILED message can also appear if the system manager invokes OPSER successfully, terminates the program with a CTRL/C, and then tries to invoke OPSER from memory by typing RUN. In the past, the system manager could type RUN then press the RETURN key to invoke a program that was still in memory. This procedure will no longer invoke OPSER. The message, therefore, instructs the system manager to include the account number of the Spooling Package Library in the RUN command. For example, RUN \$OPSER and RUN [1,10] OPSER can be valid commands but RUN cannot. These restrictions are necessary in order to allow the Spooling Package to reside on any account in the system rather than only in account [1,2].

OPSER tests for the presence of its two work files (OPSER0.WRK and OPSER1.WRK) in account [1,2]. If the work files are not present, OPSER prints the following message.

```
'OPSER' FILES NOT FOUND - WILL INITIALIZE
```

OPSER then initializes all its tables. At this point, the valid operator table has an entry which allows all privileged users from any terminal to communicate through OPSER. If ERRCPY is active, the on-line job table has ERRCPY as its only entry.

When OPSER finds the work files present upon starting, it ensures that it has write access to the files. Should another job have write access to the files, OPSER prints the following message and terminates.

```
'OPSER' DOES NOT HAVE WRITE PRIVILEGES TO ITS FILES
```

To allow OPSER to gain write access, the operator must terminate the job currently having write access and rerun OPSER.

The program performs further special processing for currently existing work files. The processing ensures data integrity in case OPSER is being restarted after a system crash. A check is made of the entries in the on-line job and valid operator tables.

For each entry in the on-line job table, the following conditions are checked.

1. The job number indicated in the entry must be active on the system.
2. The receiver identification in the entry must exist in the system message receiver table for that job number.
3. The account in the entry must be a valid account number.

An entry for an inactive or otherwise illegitimate job is removed from the on-line job table. For each active and legitimate job, OPSER requests, when processing begins, retransmission of the job's last message. OPSER generates a command to display (when processing begins) the on-line job table.

OPSER checks the entries in the valid operator table. What further actions take place depend on whether jobs found in the on-line job table are active. If the on-line job table has no active jobs and a meaningless entry is in the valid operator table, OPSER clears the table and sets initial conditions. The resultant operator table has one entry which allows, as valid operators, all privileged users from any keyboard. If the on-line job table had an active job, the program merely removes from the valid operator table those entries that are not meaningful. To signal that a change in the current table has occurred, an internal command is created to print the valid operator list when processing begins.

After processing of the work files is complete, the program attempts to declare OPSER as a receiving job. If the declaration fails, the program prints the following message.

```
'OPSER' CANNOT DECLARE ITSELF A RECEIVER
```

This message indicates that another copy of OPSER has been run and must be properly terminated to allow the current copy to run successfully.

OPSER establishes the current keyboard as the Operator Services Console (OSC) and a file named OPSER.LOG as the default log file. OPSER.LOG is optional, but OPSER uses it until an explicit command closes it or assigns another log file. When OPSER begins normal operation, it executes any internally generated commands to list on-line jobs and valid operators. After executing the commands, OPSER is ready to accept data sent to it by any program. OPSER prints the number sign (#) character to show that it is ready to accept commands. These commands are described in Section 5.2.1.

5.2.1 OPSER Operator Commands

A valid OPSER command may be abbreviated to three or more characters and is separated from any operands by a space character. No embedded spaces are allowed within a command. Delimiters within the text of the operands are the semicolon (;) and colon (:) characters and must be present to delimit elements of text. The commands and formats are summarized in Table 5-1. Throughout the following discussion, optional items are enclosed in square brackets.

Table 5-1: OPSER Commands

Command Name	Syntax and Meaning
ANSWER	ANS[WER]Δmsgnumber:text Conveys text following the colon as a response to an action request denoted by message number. Also deletes the action request.
CHANGECONSOLE	CHA[N]GECONSOLE]ΔKBn: Changes the operator services console to the keyboard unit designated by n. The unit should be on line to the system but need not be free.
DELETE	DEL[ETE]Δmsgnumber Deletes from an OPSER internal table an unanswered action request denoted by its message number. DEL[ETE]Δ#m[:n] Deletes the n oldest (from 1 to 32) action requests for a given job. The command requires the number sign (#), the number (m) of the sending job, a colon, and the number (n) of requests to delete.
DETACH	DET[ACH] Detaches the OPSER program from its keyboard, after which the operator can run PLEASE to communicate with OPSER.
EXIT	EXI[T] Closes the log file and work files, removes OPSER from the system message receiver table, and terminates the program. If the job is attached, the program returns to READY level. If detached, the program broadcasts a message to the system console terminal (KB0:) and kills itself.
INTERRUPT	INT[ERRUPT]Δrcvrid:text INT[ERRUPT]Δ#n:text Sends the unsolicited text to an on-line job specified either by its message receiver identification or by #n (where n is the number of the job under which the on-line program is running). The LIS JO command listing gives job numbers and receiver identifications.
LIST	LIS[T]ΔJO[BS] LIS[T]ΔOP[ERATORS] Prints a listing of all jobs on line to OPSER or a listing of valid operator accounts and keyboards.

(continued on next page)

Table 5-1: OPSER Commands (Cont.)

Command Name	Syntax and Meaning
LOGFILE	<p>LOG[FILE]Δfile;AL[L] RE[REQUESTS] NO[NE]</p> <p>Establishes, as the OPSER log file, the specified file or device and sets the message level to the value specified. Message levels are ALL to record all messages and action requests, REQUESTS to record only action requests, and NONE to record no messages, action requests, or OPSER information lines.</p> <p>LOG[FILE]Δ;msglevel If a file or device is not specified, a semicolon followed by a level simply changes the message level of the log file currently open.</p> <p>LOG[FILE] If neither a file nor message level is specified, OPSER closes the current log file and records nothing thereafter.</p>
MESSAGE	<p>MES[SAGE]Δlevel Sets the level of messages for the operator services console. The level controls what types of messages will be printed at the OSC. If no level is included in the command, an error is generated.</p> <p>MES[SAGE]ΔAL[L] ALL includes all messages, action requests, and OPSER information lines.</p> <p>MES[SAGE]ΔRE[REQUESTS] REQUESTS includes only action requests and information lines and excludes messages.</p> <p>MES[SAGE]ΔNO[NE] NONE means that nothing will be printed on the OSC. (See Section 5.2.2 for a discussion of message level.)</p>
OPERATOR	<p>OPE[RATOR]ΔKBn:[nnn,nnn] For use by privileged operators only. Updates the valid operator list with the designated keyboard unit and account combination. The asterisk (*) can replace the keyboard unit, project number, and programmer specifications to include all of that element.</p> <p>OPE[RATOR]Δ-KBn:[nnn,nnn] If minus (-) precedes the specification, that specific element is deleted from the list.</p>
RETYPE	<p>RET[YPE]Δmsgnumber Reprints, at the requesting terminal, an unanswered action request denoted by its associated message number.</p> <p>RET[YPE]Δ[#n[:n]] Several unanswered action requests can be printed by giving the sending job's job number. If :n is given with the job number, the n oldest (from 1 to 32) action requests are printed for that job.</p>

5.2.2 Message Levels and Formats

The MESSAGE command defines a level which controls what is printed on the Operator Services Console (OSC). As in the LOGFILE command, the possible values for level are ALL, REQUESTS, and NONE. When OPSER starts, the default value is ALL.

When OPSER is running, what it sends to the OSC depends on the current message level. The level should almost always be set to ALL so that the operator can have ready access to the complete processing history. The level can be temporarily set to NONE to suspend printout.

NOTE

When the message level is set to NONE, no action requests or messages are printed. They are stored in a table within OPSER's data base. While the table dimension permits storage of a large number of messages, data may be lost should the table overflow with message level set to NONE.

After the message level is set to ALL, the RETYPE command can be used to print from 1 to 32 of the most recent messages or action requests.

To distinguish between messages and action requests on the Operator Services Console (OSC), output appears in distinct formats. Messages are in the following form.

```
MESSAGE tab tab nnnnn : date time JOB:NN KB:MM jobname [nnn,nnn]  
          tab text
```

If the first character of the message text is a question mark (?) character, the words FATAL MESSAGE replace MESSAGE in the identification line.

Action requests are in the following form.

```
REQUEST tab tab nnnnn : date time JOB:NN jobname [nnn,nnn] rcvrid  
          tab tab text
```

In an action request, the first character of text is a CTRL/G (BEL) character to alert the operator.

The contents of messages and action requests are summarized in Table 5-2.

An action request is the result of a spooling job's requesting interaction with the operator. The spooling job sends a message to OPSER, which forms an action request. The spooling job performs no further processing until operator action is taken or until the condition which generated the action request is satisfied. OPSER broadcasts action requests on the OSC unless the message level is set to NONE.

To respond to an action request, the operator employs the ANSWER command. Because OPSER usually runs detached, PLEASE can be run at valid operator terminals to send the response to OPSER. The ANSWER command requires a message number to identify the action request to which the operator is responding.

OPSER sends the text in the ANSWER command to the spooling program and deletes the action request. If the response satisfies the request, the program resumes processing. If the response does not satisfy the request, the spooling program still expects an appropriate response. The operator must then send such a response by means of the INTERRUPT command described in Section 5.2.4.

Table 5-2: OPSER Message and Action Request Contents

Item	Meaning
nnnnn	Sequence number of message starting at 1 and incremented by 1 during processing; must be used in ANSWER command to respond to a specific action request. Upon restarting with currently existing work files, OPSER rounds the last assigned sequence number up to the next multiple of 10. This number will be the next assigned sequence number.
date	Current system date.
time	Current system time of day.
JOB:nn	Number of job under which sending program is running; can be used, preceded by number sign (#), to refer to job in an INTERRUPT, DELETE, or RETYPE command.
KB:mm	The keyboard number of the job which sent the message. The KB:mm field is displayed as "DET" if the job was detached and as KB:?? if OPSER is unable to determine the keyboard number. Typically, the KB:?? message appears when the job has terminated.
jobname	Name of program which sending on-line job is running; this name is printed for information purposes. Six question marks, ??????, are printed in place of the jobname if OPSER is unable to determine the jobname. Typically, the ?????? message appears when the job has terminated.
[nnn,nnn]	Project-programmer number under which the sending on-line job is running.
rcvrid	For action requests only. The message receiver identification given to the job when it is started. The identification can be standard (LPnSPL for the SPOOL program or BAnSPL for the BATCH program) or can be optional (as specified in the /NAME start-up switch for SPOOL or BATCH). The receiver identification can be used in the INTERRUPT command to refer to the spooling program.
tab	Horizontal tab character(s) to indent text. A single HT character denotes message text; two HT characters denote action request text.

5.2.3 Valid Operator and On-Line Job Lists

The LIST command provides a display of data concerning the valid operator list or the on-line job table maintained by OPSER.

LIST OPERATORS displays keyboard number and project-programmer number combinations currently defined as valid operators. The valid operator list is updated by the OPERATOR command. Only a privileged operator can update the list.

LIST JOBS prints data concerning on-line jobs in the following format preceded by the text ON-LINE JOBS.

```
#n [nnn,nnn] 'rcvrid' SL=n
```

Table 5-3 describes the items in the on-line job list.

Table 5-3: OPSER On-Line Job List

Item	Meaning
#n	Number of job under which the on-line program is running.
[nnn,nnn]	Project-programmer number under which the on-line job is running.
'rcvrid'	Message receiver logical identification used to identify the on-line program in INTERRUPT commands.
SL=n	OPSER shutdown level used when SHUTUP notifies OPSER to terminate spooling operations in an orderly fashion.

The job data listed is valuable to the operator. For example, certain OPSER commands require either a receiver identification or job number to communicate with a spooling program.

5.2.4 Operator INTERRUPT Command

The INTERRUPT command sends unsolicited messages and special commands to spooling programs. OPSER accepts any text in the INTERRUPT command and sends it to the destination job. The destination job alone interprets the text. If a response to an INTERRUPT command is generated, it is displayed on the Operator Services Console. Special text interpreted by spooling programs as commands is described in Sections 5.3 to 5.5.

The INTERRUPT command is the means by which an operator controls and monitors spooling programs. Each spooling program recognizes commands which can be divided into three sets.

1. commands which have no direct effect on how the program itself processes a request queued by a user,
2. commands which do have a direct effect on program operation but are uniform for all spooling programs, and
3. commands which are unique to the individual spooling program.

Included in the first set of commands are PAUSE, CONTINUE, NOTICE, LAST, and STATUS. These commands allow the operator to control the program in general and to gain information about the program.

The second set comprises END, ABORT, and OFFLINE. These commands directly influence the job.

In the third set are unique commands such as those for forms control on a line printer.

5.2.5 OPSER Start-Up Procedure

OPSER should be started by commands in the system start-up control file described in Section 3.1. The following description presents suggestions concerning what commands should be specified in the control file.

The following control file will cause INIT to start OPSER.

```
OPE KB*:(2,2)
LOG $OPSER.LOG;ALL
MES ALL
DET
```

The OPERATOR, LOGFILE, MESSAGE, and DETACH commands establish initial operating conditions for OPSER. The LOGFILE and MESSAGE commands will set conditions normally established as defaults and are shown for clarity.

The OPE command shown updates the valid operator list with a nonprivileged account on any keyboard. It is important to define the operator with a nonprivileged account. This action limits the operator to those privileges allowed by OPSER. Allowing the operator to communicate from any terminal on the system gives flexibility to those persons performing operator functions.

The LOG command keeps the file \$OPSER.LOG with a message level of ALL. As a result, all messages, action requests, OPSER information lines and operator responses are written to the file to provide a complete historical reference.

The MES command shown specifies that all messages, action requests and OPSER information lines will be printed at the OSC. Normally, the operator is concerned only with action requests. If the operator is concerned with any unanswered action request, the RETYPE command can be issued. Additionally, the log file can be closed with the LOG command and printed at any time to recover recorded information.

The DET command causes OPSER to detach itself from the terminal on which it is running. At the start of time sharing, this terminal is usually the system console terminal (KB0:). While OPSER runs detached, it is immune from tampering by unauthorized users. The operator can communicate with OPSER through the PLEASE program.

The terminal on which OPSER starts is automatically defined as the OSC. This definition can be altered by the CHANGECONSOLE command in the start up procedure. Any valid operator may also alter the assignment of the OSC. It is most convenient to keep keyboard unit 0 as the OSC because, regardless of the number of logins currently allowed on the system, the operator can always use the system console terminal to communicate with OPSER.

5.2.6 OPSER Action Under Various Start-Up Conditions

OPSER start-up actions depend on the way OPSER last terminated.

When OPSER runs after the system manager has shut down operator services in an orderly fashion through SHUTUP, a normal start-up procedure occurs. When shutting down in an orderly fashion, OPSER removes all entries from its on-line job table and closes all files properly. Upon restarting, OPSER finds all data valid. The on-line job table is empty and entries in the valid operator table are legal and therefore are retained.

When OPSER starts after a system crash, it may find entries in its on-line job table. Since OPSER starts before other on-line jobs, none of the jobs in the table are active. The program therefore clears the entries from the table. If any entries in the valid operator table are illegal, OPSER clears that table and sets one valid operator entry – all privileged users from any keyboard.

When OPSER starts, after it has terminated unexpectedly, it may find entries in its on-line job table for jobs still active on the system. The program, in this instance, sends a command to each active job having an entry in the on-line job table. The command requests the job to retransmit the last message it sent to OPSER. OPSER does not wait for a response. If any job in the on-line job table does not meet OPSER's integrity checks, OPSER removes its entry from the table (takes the job off line) and generates an internal LIST JOBS command. OPSER also scans the entries in the valid operator table. If any entry is illegal, the program removes it. If OPSER removes any entry, it generates an internal LIST OPERATORS command.

5.3 Queue Manager Program – QUEMAN

The queue manager program QUEMAN maintains the file of queued requests (QUEUE.SYS in the Spooling Package Library account, [1,2] by default) and communicates with spooling programs to execute queued requests. Queue management involves two modules QUEMAN and QUMRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module sets initial conditions on starting and checks initial conditions upon restarting. To begin processing, the first module chains to the second one, which actually executes commands and manages the queues. For simplicity, the documentation refers only to one program, QUEMAN, which functionally includes the two modules.

The QUEMAN program runs only if the job has permanent privilege and the OPSER program is running (that is, only if the name OPSER is in the system table of message receivers). Starting QUEMAN is a prerequisite to starting the spooling programs. To start QUEMAN, use the following command while logged into the system under a privileged account.

```
RUN $QUEMAN
QUEMAN V7.0 RSTS V7.0 TIME SHARING
STARTED AT: 01:52 PM ON 03-OCT-78
```

The QUEMAN programs must reside in the Spooling Package Library (SPL) in executable form. If the system manager tries to invoke a QUEMAN program from the SPL account and the program is not in executable form, QUEMAN prints the following message.

```
?QUEMAN must be compiled
```

The message warns the system manager that the program was not executable. Programs that are executable usually have a .BAC or .TSK extension. After the message is printed, the system manager must load a compiled version of

the QUEMAN program into the Spooling Package Library account and invoke the program again. The ?QUEMAN MUST BE COMPILED message can also appear if the system manager invokes QUEMAN successfully, terminates the program with a CTRL/C, and then tries to invoke QUEMAN from memory by typing RUN. In the past, the system manager could type RUN then press the RETURN key to invoke a program that was still in memory. This procedure will no longer work. The message, therefore, instructs the system manager to include the account number of the Spooling Package Library in the RUN command. For example, RUN \$QUEMAN and RUN [1,10]QUEMAN can be valid commands but RUN cannot. These restrictions are necessary in order to allow the Spooling Package to reside on any account in the system rather than only in account [1,2].

If the account that you log into is not privileged, the system prints the ?PROTECTION VIOLATION and ?PROGRAM LOST-SORRY messages. If job has permanent privilege, the program prints its identification and lines and begins preliminary error checking.

QUEMAN tests for the presence of OPSER. If OPSER is not running, QUEMAN prints the message QUEMAN CANNOT RUN WITHOUT 'OPSER' ACTIVE and terminates. The operator services program OPSER must be started before QUEMAN can start.

QUEMAN opens the files QUEUE.WRK, OPSER1.WRK, and QUEUE.SYS in the system library account. If QUEUE.WRK does not exist, the program prints the following message and creates the file.

```
#QUEUE.WRK NOT FOUND - WILL INITIALIZE
```

If OPSER1.WRK does not exist, the program prints the following message and terminates.

```
#OPSER1.WRK NOT FOUND -- CAN'T RUN
```

To recover, the operator must start the OPSER program before starting QUEMAN. If QUEUE.SYS does not exist, the program creates and initializes it. QUEMAN signals this action by generating the following message.

```
NO QUEUE FILE FOUND -- WILL INITIALIZE
```

If QUEUE.SYS exists, QUEMAN ensures that it has write access to the file. Should another program have write access to the QUEUE.SYS file, QUEMAN GENERATES the following message.

```
QUEUE FILE OPENED BY ANOTHER PROGRAM  
TRY AGAIN (Y/N) <N>?
```

The operator must determine which job has the file QUEUE.SYS open and must terminate that job. Typing Y in response to the question causes QUEMAN to retry the open operation. Typing N terminates QUEMAN.

When the open operation successfully gains write access to QUEUE.SYS, QUEMAN declares itself a message receiver on the system. If QUEMAN is already declared by a different job, the program prints the following message.

QUEMAN CANNOT DECLARE ITSELF AS A RECEIVER --- CAN'T RUN

The operator must terminate the other job so that it will be removed from the message receiver table. Typing the CONT command causes QUEMAN to retry the declaration.

To ensure that the previous QUEMAN job properly closed the currently existing QUEUE.SYS file, the program checks a flag value in the file. If the flag is not properly set, the program prints the following message:

QUEUE FILE NOT CLOSED PROPERLY --
NOW CHECKING DATA FOR CONSISTENCY

A later message QUEUE FILE DATA CHECKED signals that all integrity checks were successful. If the flag is properly set, QUEMAN performs consistency checking but does not print any messages unless it finds problems. If any problems are found, QUEMAN generates the following messages.

QUEUE FILE DATA INCONSISTENT - WILL INITIALIZE
INITIALIZED

The latter message indicates that all entries in the QUEUE.SYS file have been set to their initial conditions and all queued requests have been removed from the queues. Section 5.3.5 describes the queue file consistency checks QUEMAN performs.

After completing all initial checks, QUEMAN prints the number sign character (#) indicating its readiness to accept a start-up command or switch.

5.3.1 QUEMAN Start-Up Commands and Options

QUEMAN recognizes the start-up commands shown in Table 5-4.

Table 5-4: QUEMAN Start-Up Commands

Command	Syntax and Meaning
DETACH	DET[ACH] Detaches QUEMAN from the terminal.
INITIALIZE	INI[TIALIZE] Sets all the entries in the QUEUE.SYS file to their initial conditions. Any currently queued requests are lost.

The INI command clears all the entries from the QUEUE.SYS file and sets initial conditions for the spooling queues. The DET command detaches QUEMAN from the terminal. QUEMAN should be run detached.

Before QUEMAN executes the INITIALIZE command, it checks its on-line spooler table. If the table has an entry for a spooling job, QUEMAN prints the following message:

```
SPOOLERS ONLINE -- CAN'T INIT
```

The program reprints the prompt.

QUEMAN recognizes start up options (switches). One or more options may appear following a command or may be alone on a line in response to the prompt. Table 5-5 lists the start up options allowed.

Table 5-5: QUEMAN Start-Up Options

Option	Syntax and Meaning
PRIORITY	/PRI[ORITY]: <i>nnn</i> Sets the job priority to <i>nnn</i> (where <i>nnn</i> can be from -120 to +120). Without the switch, the program automatically sets the priority to 0.
RUNBURST	/RUN[BURST]: <i>nnn</i> Sets the job run burst to <i>nnn</i> (where <i>nnn</i> can be from 1 to 127). Without the switch, the run burst value currently assigned is used.

If any data entered in response to the prompt are not valid commands or options, QUEMAN prints the following message:

```
INVALID RESPONSE -- text
```

The text is the data entered. The program reprints the prompt.

To cause QUEMAN to detach, the operator types the DET command in response to the prompt. For example:

```
*DET  
DETACHING...
```

As a result, the program prints the DETACHING message and detaches itself from the keyboard. QUEMAN should always be run detached. While detached, the job receives commands from the operator through the operator services program as described in Section 5.3.2.

NOTE

At no time should the operator type CTRL/C to the QUEMAN program. Such action may corrupt the QUEUE.SYS file.

5.3.2 QUEMAN Interrupt Commands

QUEMAN recognizes interrupt commands which allow an operator, through the INTERRUPT command of OPSER, to gain information on queue manager processing and to control program operation. The commands are summarized in Table 5-6.

Table 5-6: QUEMAN Interrupt Commands

Command	Option and Meaning
END	Terminates QUEMAN in an orderly manner. Disables queuing and spooling. If any spooling programs are on line to QUEMAN, sends to OPSER the notification message SPOOLERS STILL ONLINE - WILL CLEAR TABLE and clears the on-line spooler table. Sends a command to OPSER which takes the QUEMAN entry out of OPSER tables. Closes its work files and kills itself.
LAS(T)	Sends to OPSER the most recent message generated by QUEMAN.
NEX(T)	<i>quenam=reqnam</i> Places the pending user request identified by <i>reqnam</i> at the head of the queue specified by <i>quenam</i> . A request name can contain a job name, project-programmer number and sequence number. Request names are found in the listing generated by the QUE program command L dev: (where dev: is the appropriate queue name). For example, valid queue names may be LP:, BA:, or RJ:.
OFF(LINE)	Terminates QUEMAN as the END command does but does not generate notification message
STA(TUS)	Prints a brief report of spooling jobs on-line to QUEMAN
DIS(ABLE)	QUE(UING)<text> SPO(OLING) ALL Disables QUEUING or SPOOLING or both operations with the ALL option. When you specify DISABLE QUEUING, the program QUE stops sending messages to QUEMAN and stops listing the queue. Any jobs already sent to QUEMAN are processed. If you specify DISABLE SPOOLING, QUEMAN prevents any further jobs from being sent to any spooler. Jobs sent to the spooler prior to the DISABLE command are processed to completion. Specify DIS(ABLE) ALL and QUEMAN terminates both QUEUING and SPOOLING. You can include a message to the user while specifying the DISABLE command. Type the command, the option, and then a space followed by the message. After QUEMAN processes the command, it prints the <text> message in the following formats: 1. Further QUEUING DISABLED <text> 2. Further SPOOLING DISABLED <text> 3. Further QUEUING AND SPOOLING DISABLED <text> If you ignore the <text> field, QUEMAN prints the default text, "by operator". The message appears when you try to queue a file.
ENA(BLE)	QUE(UING)<text> SPO(OLING) ALL Enables QUEUING or SPOOLING or both operations with the ALL option. When you specify ENABLE QUEUING, the program QUEUE begins to send messages to QUEMAN and starts listing the queue. If you specify ENABLE SPOOLING, QUEMAN begins to send jobs to the spoolers. Specify ENABLE ALL and QUEMAN activates both the QUEUING and SPOOLING functions. Use the <text> field to send messages to the user. Type the command, the option, and then a space followed by the text of the message. After QUEMAN processes the command, it prints the message in the following formats: 1. QUEUING ENABLED - <text> 2. SPOOLING ENABLED - <text> 3. QUEUING AND SPOOLING ENABLED - <text> The default <text> message is "by operator". The message appears when you try to queue a file.

When a command is received from the operator through OPSER, QUEMAN performs the action requested and either generates a message or an information line. OPSER formats the message or information line and displays it on

the OSC. An example of this interaction between an operator, QUEMAN and the OPSER program is shown by the STATUS command. (The following example assumes the operator is running the PLEASE program and the OSC is the keyboard on which PLEASE is running.)

```
*/INT #6:STATUS
COMMAND SENT TO 'OPSER'
#
MESSAGE 29 : 10-OCT-78 10:48 AM JOB:6 DET QUMRUN[1,2]
          1 SPOOLER(S) ON-LINE -
(25) LP1SPL LP1: FORMS=NORMAL;
```

PLEASE passes the full command line to OPSER and notifies the operator before it reprints the prompt. OPSER recognizes the text INT as a request to send unsolicited text to a spooling job. The characters #6 in the command line are recognized as the number of the destination job to which the text must be sent. QUEMAN receives the text STATUS, generates a response, and sends the response back to OPSER. OPSER formats a message and displays it on the OSC for the operator.

The STATUS command itself is the operator's method for gaining information about jobs which are on line to QUEMAN. QUEMAN'S status printout tells the number of spooling jobs currently on line and supplies data on each job (job number within parentheses, receiver identification, physical unit being spooled, and program default conditions). If the particular spooling job is processing a request, the data in the printout includes the following information about the request: its name, project-programmer number under which it is queued, and its sequence number. The STATUS command will also tell you if SPOOLING or QUEUING is disabled.

The NEXT command allows the operator to place a job request at the head of a queue. Job requests are identified by the name, account, and sequence number under which the request is queued. This format is described in the QUE program discussion in the *RSTS/E System User's Guide*. A request moved to the head of a queue will be the next one QUEMAN sends to a spooling program servicing that queue.

If QUEMAN detects an error in the NEXT command, it generates a message in the following format:

```
'NEXT' CMD: string - text
```

The string is the command typed and text is one of the following phrases:

Text	Meaning
ILLEGAL PARAMETERS	Only a queue name and request name, separated by an equals sign, are allowed in NEXT command.
ILLEGAL SYNTAX	A space or equals sign is missing.
JOB IN PROCESS	The specified job request has already been sent to spooling program.
MULTIPLE JOB SPECIFIED	The specified request name matches two or more jobs in the queue.
NO MATCH FOR JOB	Request name does not match any job in queue.

5.3.3 QUEMAN Start-Up Procedure

QUEMAN should be started by commands in the system start-up control file described in Section 3.1. The description here presents suggestions concerning what QUEMAN commands should be specified in the control file.

The following commands should be used at system start up to start QUEMAN.

```
/PRIORITY:0  
DETACH
```

The priority is set to 0 because most jobs on RSTS/E run at priority -8 and QUEMAN, running at priority 0, can more readily process the queue file.* The DET command is specified because QUEMAN should run detached on all systems.

Because QUEMAN examines the QUEUE.SYS file when it starts, no need exists to explicitly initialize the file. If the file contains bad data, QUEMAN resets all entries to initial states. The INI command, therefore, should be specified only in situations where you want to explicitly initialize the queue file.

5.3.4 QUEMAN Action Under Various Start-Up Conditions

QUEMAN start-up actions depend on the way QUEMAN last terminated.

When QUEMAN runs after the system manager shuts down operator services in an orderly fashion through SHUTUP, a normal start-up procedure occurs. QUEMAN finds that the QUEUE.SYS file was closed properly. The program performs consistency checking but does not print any messages unless it finds problems. If the data is consistent, QUEMAN retains all job requests in the queue file and examines each request for completeness and status. Since QUEMAN normally starts before any spooling programs, there are no entries in the on-line spooler table. QUEMAN ensures that this table is clear of entries.

When QUEMAN starts either after the system crashes or after QUEMAN itself terminates unexpectedly, it finds the queue file improperly closed and generates a message to that effect. The program performs consistency checking on the queue file structure and on the data in the QUEUE.SYS file itself, generates messages telling the results, and re-establishes communication with all spooling jobs.

5.3.5 QUEMAN Consistency Checking

The QUEUE.SYS file has room for 250 queued requests. QUEMAN creates an entry for a queued request from an entry in a free list. A request for a line printer or batch queue requires one entry from the free list. A request queued with an AFTER date and time, however, requires two entries from the free list - one for the proper queue and one for the AFTER queue.

*Without the /PRIORITY option, QUEMAN automatically sets its priority to 0. The option can be explicitly specified to provide a record of what the program does automatically.

In performing consistency checking on the QUEUE.SYS file, QUEMAN initializes all entries if it finds one of the following conditions.

1. an entry in the free list is also in a queue or AFTER list
2. the root of the free list is outside the legal range
3. a request queued with an AFTER date and time which has not expired does not have an entry in the AFTER queue

For other inconsistencies, QUEMAN may either remove a single entry or perform some related action. If an entry in the AFTER queue has no corresponding entry in a spooling queue, QUEMAN removes the AFTER entry. If an entry in either an AFTER or spooling queue is not completely set up, the program removes the entry. (QUEMAN may have terminated while it was processing the request.) To signal this action, QUEMAN prints a message in the following format:

```
%LPn: 'reqnam' [nnn,nnn];QUEUED JOB INCOMPLETE - REMOVED FROM QUEUE
```

If an entry to be killed still remains, QUEMAN removes it and generates text in the following format:

```
%LPn: 'reqnam' [nnn,nnn];QUEUED JOB IN 'KILL' STATUS - REMOVED FROM QUEUE
```

If QUEMAN finds that an entry has been processed (sent to a spooling program) but not yet completed, it places the request in a hold status and generates text in the following format:

```
%LPn: 'reqnam' [nnn,nnn];JOB PREVIOUSLY SENT TO SPOOLER;WILL BE PUT INTO HOLD STATUS
```

The requester can remove the job request from hold status.

QUEMAN checks the on-line spooler table in the QUEUE.WRK file for consistency. If the program finds the table empty, it generates the following message:

```
*SPOOLERS ONLINE=0;WILL CLEAR TABLE
```

If the program finds that the count of on-line spooling jobs is larger than the limit (16), it clears the table and generates the following message:

```
ONLINE SPOOLER TABLE CORRUPT-WILL CLEAR TABLE
```

If QUEMAN clears the on-line spooler table but spooling jobs are still running, the operator must terminate those jobs through OPSER and restart each spooling program. This action allows QUEMAN to put each spooling job properly on-line again.

For each spooling program found in the on-line job table, QUEMAN removes the entry and generates the following message:

```
'rcvrid' (nn) FOUND ON-LINE-TAKEN OFFLINE
```

QUEMAN sends to the job a message requesting that it declare itself on line to QUEMAN again. The spooling job does not respond to the message while it is processing a queued request. The answering message indicating that the job is again on line to QUEMAN does not appear until the job completes the current request. When QUEMAN receives the on-line declaration from a spooling job, it generates text in the following format:

```
'rcvrid' (nn) PUT ONLINE
```

5.4 Line Printer Spooling Program - SPOOL

The line printer spooling program SPOOL runs without operator intervention and executes queued requests to transfer disk files to a line printer or terminal. The program consists of three modules: SPOOL, SPLIDL, and SPLRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module establishes initial conditions on start up and checks initial conditions upon restart. The second module executes when no job is currently being printed while the third module is used to print any spooled file. For simplicity, the documentation refers only to one program, SPOOL, which functionally includes the three modules.

The SPOOL program can run only if the job has permanent privilege and the OPSER and QUEMAN programs are running. To start SPOOL, type the following command while logged into the system under a privileged account.

```
RUN $SPOOL
SPOOL V7.0 RSTS V7.0 TIME SHARING
*
```

The SPOOL programs must reside in the Spooling Package Library (SPL) in executable form. If the system manager tries to invoke a SPOOL program from the SPL account and the program is not in executable form, SPOOL prints the following message.

```
?SPOOL must be compiled
```

The message warns the system manager that the program was not executable. Programs that are executable usually have a .BAC or .TSK extension. After the message is printed, the system manager must load a compiled version of the SPOOL program into the Spooling Package Library account and invoke the program again. The ?SPOOL MUST BE COMPILED message can also appear if the system manager invokes SPOOL successfully, terminates the program with a CTRL/C, and then tries to invoke SPOOL from memory by typing RUN. In the past, the system manager could type RUN then press the RETURN key to invoke a program that was still in memory. This procedure

will no longer work. The message, therefore, instructs the system manager to include the account number of the Spooling Package Library in the RUN command. For example, RUN \$SPOOL and RUN [1,10]SPOOL can be valid commands but RUN cannot. These restrictions are necessary in order to allow the Spooling Package to reside on any account in the system rather than only in account [1,2].

If the account that you log into is not privileged, the system prints the ?PROTECTION VIOLATION and ?PROGRAM LOST-SORRY messages.

If the job has permanent privilege, SPOOL runs and prints its identification line and the number sign (#) prompt. In response to the prompt, a specification in the following form can be given.

```
logical device start up switch(es)
```

The logical device is the name of the queue and the unit number within that queue from which this copy of SPOOL takes job requests to execute. This name ordinarily corresponds to the physical device on which SPOOL prints the requests; the physical device, however, may be changed by the /PHYSICAL start up option described in Section 5.4.1.

The logical device name must have the form LPn: or LP: where n is a unit number from 0 to 7. If the name is LPn:, the spooling job prints only requests queued either to that explicit unit or to the general spooling queue LP:. If LP: is specified, this copy of SPOOL prints requests in the printer queue regardless of the unit to which they were originally queued.* The logical device specified as the queue name must be a line printer; the device name given must not have a logical assignment to some other device.

Any combination of start-up switches described in Section 5.4.1 may be specified to condition the operation of the SPOOL program. A switch is formed by a slash (/) character and an option which may take an operand or other switches.

To guarantee proper forms handling, the system manager must ensure that the SPOOL program is properly conditioned for the device to which it is spooling. The system manager does this conditioning by careful use of the options as discussed in Section 5.4.1.

5.4.1 SPOOL Start-Up Options

Start-up options alter default processing conditions. They can be used to spool output to a different physical device, to change form information, and to control job environment. The options are summarized in Table 5-7. The graphic symbol is an underline character and not a space.

*The general spooling queue LP: is useful only on systems having line printers with similar characteristics.

Table 5-7: SPOOL Start-Up Options

Option	Syntax and Meaning
ASSIGN	<p>/ASS[IGN] Reserves the physical device for this job. Without this option, SPOOL will retry continually whenever it cannot gain access to the unit and is ready to print a job.</p>
FORM	<p>/FOR[M]:name/sws Defines the current form according to a name (NORMAL is the default name) and switches as follows:</p> <p>/ALI[GN] Causes SPOOL to execute a forms alignment procedure before processing any further queued requests</p> <p>/DFL[ENGTH]:nnn Declares the device form length as <i>nnn</i> lines where <i>nnn</i> is a number from 1 to 127. The length is that understood by the hardware of the output device. For example, on a line printer in the U.S., the standard length is usually 66 lines per page.</p> <p>/HEA[DINGS]:n Causes SPOOL to print <i>n</i> (from 0 to 3) heading burst pages preceding each job request. Unless /NH accompanies a file specification in the QUE command, SPOOL also prints <i>n</i> burst pages preceding each file in a job request. The default value is 1.</p> <p>/LEN[GTH]:n Defines length of software form as <i>n</i> lines per page where <i>n</i> is a number from 1 to 127. If this switch is not given, 66 lines per page is assumed.</p> <p>/PAG[E__EJECT]:yes Indicates that the device to which this copy of SPOOL directs output has a hardware top of form capability and that the device interprets the CHR\$(12) CHARACTER as a top of form command. This condition is the default case for line printer devices.</p> <p>/PAG[E__EJECT]:no Conditions the software to translate a form feed character CHR\$(12%) into the proper number of line feed characters because the hardware does not recognize the character as a top of form command. This condition is the default case for SPOOL output directed to terminal devices by the /PHYSICAL option.</p> <p>/WID[TH]:n Defines width of the heading burst page as <i>n</i> characters per line. If this switch is not given, the default value 132 is used.</p>
LPFORM	<p>/LPF[ORM]:YES Indicates that LPFORM characters are interpreted by the system software for the output device. During normal transfer, the program will pass LPFORM characters unmodified to the device. This condition is the default for line printer devices.</p> <p>/LPF[ORM]:NO Indicates that the system software does not interpret LPFORM characters. During normal transfer, the program will convert each LPFORM character to an appropriate number of line feed characters. This condition is the default for terminal devices.</p> <p style="text-align: center;">NOTE</p> <p>This option affects only jobs queued with the /LPFORM switch. The SPOOL program does not automatically invoke LPFORM processing.</p>

(continued on next page)

Table 5-7: SPOOL Start-Up Options (Cont.)

Option	Syntax and Meaning
NAME	<i>/NAM[E]:rcvrid</i> Places the specified identification in the system message receiver table and in OPSER's on-line job table. This identification rather than the default identification is used by the operator in OPSER commands to access this spooling job. This identification must not be in use by some other job.
PHYSICAL	<i>/PHY[SICAL]:dev:</i> Uses this <i>device</i> as the physical device for this spooling program. Requests in the queue specified by the logical device are printed on this <i>device</i> . The default value is the logical device specified as the queue name.
PRIORITY	<i>/PRI[ORITY]:nnn</i> Sets the job priority to <i>nnn</i> where <i>nnn</i> is a number from -120 to 120. Without this option, the priority remains as set by LOGIN.
RUNBURST	<i>/RUN[BURST]:nnn</i> Sets the job run burst to <i>nnn</i> where <i>nnn</i> is a number from 1 to 127. Without this option, the run burst remains as set by LOGIN.

The PHYSICAL option alters the physical device on which queued requests are printed. For example, requests queued for line printer unit 1 could be processed on another line printer unit or on a keyboard unit. The actual device, in any event, differs from the one for which the requests were queued.

The ASSIGN option reserves the actual spooled device to the job. Without this option, SPOOL will periodically try to gain access to the spooled device whenever SPOOL is ready to print and another job has ownership of the device. When SPOOL terminates, it deassigns the device.

The NAME option allows the receiver identification to be different from that normally assigned by SPOOL. The identification should be from 1 to 6 alphanumeric characters and must not currently exist in the system message receiver table. The identification must not begin with a number because the operator uses either this identification or a job number to identify a spooling job.

The FORM option allows the default form information to be altered. Individual characteristics of the NORMAL form can be altered, a new form with a different name can be defined, or a forms alignment can be requested. A forms alignment can be requested by itself or in conjunction with a form alteration. When a form name other than NORMAL is in effect, user's queuing requests must explicitly specify the form name to have files printed with its characteristics.

To alter the default characteristics of the NORMAL form, the operator can specify the appropriate auxiliary switches without the /FORM: switch. For example, to change the number and width of heading burst pages for line printer unit 0, a specification similar to the following can be given.

```
LPO:/HEADINGS:2/WIDTH:80
```

Two heading burst pages with a width of 80 characters per line will be printed for each job request and each file within a request. The width used applies only to the burst pages, not to the data being printed.

Specifying a form with a name other than NORMAL usually means that the operator must load special paper in the output device. The /ALIGN switch with the FORM option requests a forms alignment. The operator will then have to align the form at system start-up time. A SPOOL interrupt command with the /ALIGN switch can request a form alignment during time sharing. Section 5.4.10 describes the forms alignment procedure.

The following message can occur when the line printer queues are empty, but there exist jobs which have nondefault form names:

```
NO JOBS WAITING WITH FORMNAME 'XXXXXX' FOR SPOOLER 'LPn:'  
OTHER JOB(S) WAITING.  
***PLEASE INSPECT QUEUE AND TAKE APPROPRIATE ACTION***
```

The operator should then decide whether or not to use an interruption command to change the form name so that the other jobs can be queued.

The PAGE__EJECT option determines whether the SPOOL program counts lines to effect forms control. (The DFLENGTH option can modify the effect of the PAGE__EJECT option.) A value of NO with the PAGE__EJECT option indicates that the output device (driver) does not understand a form feed character; that is, a form feed character does not cause the device to position itself at top of (paper) form. With NO in effect, SPOOL counts lines to effect forms control. A value of YES with the PAGE__EJECT option indicates that the device (driver) translates the form feed character to an appropriate number of line skips* to place the device at top of form. With YES in effect, SPOOL need not count lines but may be able to pass all forms control data unaltered to the device driver. (The DFLENGTH option can modify the effect of YES.)

If NO is in effect for the PAGE__EJECT option, SPOOL ingores any value specified in a DFLENGTH option.

The operator should specify /PAGE__EJECT:YES for any terminal device which has a top of form capability.

SPOOL uses the value specified in the DFLENGTH option only if /PAGE__EJECT:YES is in effect. The /PAGE__EJECT:YES switch can be in effect either through default (on a line printer) or by explicit specification (for a terminal device having the form feed capability). The operator may specify

*SPOOL issues three forms-control characters: 1) form feed (the CHR\$(12) character, whose ASCII value is 12), 2) carriage return (the CHR\$(13) character which is used for overprinting), and 3) line feed (the CHR\$(10) character). No other forms-control characters are used. SPOOL has no provisions for changing or extending the set of control characters for any special device(s).

The terms line feed and line skip are not synonymous. Line feed refers only to the CHR\$(10) character which either SPOOL or the device driver sends to a device. Line skip refers to the movement of paper which the device itself effects.

the DFLENGTH option either at start-up time in a SPOOL start up switch or during processing in a FORM interrupt command. For proper formatting, the DFLENGTH value must be the maximum number of lines which the output device skips when it receives a form feed command. This number of lines is called the device form length. This number may differ from the length of the paper form being used.

Different devices have different form lengths. Three examples can show how the device form lengths vary. First, the LA180 device has a switch labelled Length of Form which has settings for 4, 8.5, and 11 inches. These settings correspond to 24, 51, and 66 lines per form. (The number of lines per form given for each setting assumes the standard 6 lines per inch on the LA180.) If, when starting the spooling job, the operator sets the switch on the LA180 to 8.5 inches, he should also specify the DFLENGTH option with a value of 51. If, at some later time, the operator changes the setting on the LA180 to 4 inches, he should also specify a new DFLENGTH option with a value of 24 in a FORM interrupt command to the spooling job.

Second, many line printers have only one form length. If the hardware length is 66, the value in the DFLENGTH option should be 66. (The default SPOOL uses in absence of the option is 66.) If the hardware form length is 51, the operator should specify the value 51 with the option when he starts the spooling job.

Third, some printing devices have an adjustable top of form. On these devices, the operator can adjust the device to execute an arbitrary number of line skips when it receives a form feed character. This adjustment is most often on terminal-type devices with high-quality print characteristics. Whenever the operator resets the top of form for this type of device, he should also specify the DFLENGTH and LENGTH options with the new form length information. The operator should also request a forms alignment for each new form.

The effect the DFLENGTH option has on the SPOOL program differs for line printers and terminal devices.*

For a line printer, the DFLENGTH value affects only the procedure SPOOL invokes when it receives an ABORT command while it is processing a job request. If the DFLENGTH value equals the current length of paper form, the program issues a form feed character to position the device at the top of the next form. If the values are not equal, the program automatically invokes the forms alignment procedure to re-establish the top of form position.

The automatic forms alignment procedure requires operator action to align the form and to respond to requests SPOOL makes. Such action is necessary only when the program has no way to ascertain where the top of form lies after an ABORT operation.

*A line printer is a device whose interface to the computer is through a line printer controller and whose designation is in the form LPn:. A terminal is a device whose interface to the computer is through a keyboard line and whose designation is in the form KBn:. This distinction is critical because some devices such as the LA180 are called printers but may be connected to the computer through a keyboard line.

For a terminal, the DFLENGTH value controls the way SPOOL keeps its place on a form and the way the operator handles recovery procedures. If the value of the PAGE EJECT option is YES, the equality of the device form length and the paper form length is important. If the device form length and paper form length values are not equal, the SPOOL program counts lines to maintain a record of its position on a form and issues an appropriate number of line feed characters to simulate form feed. If these values are equal, SPOOL does not count lines and simply sends a form feed character to position the paper at top of form.

If the value in effect for DFLENGTH does not correctly match the setting of the hardware device, two undesirable conditions exist. First, SPOOL must incur the overhead of counting lines. Second, forms alignment may be lost during processing. For these reasons, DIGITAL highly recommends that the operator supply a new value of DFLENGTH whenever he changes the hardware setting on the device.

5.4.2 Line Printer Spooling

If the physical device specified at start-up time is a line printer unit, SPOOL applies the following default values:

Switch	Result
/DFLENGTH:66	The device form length is 66 unless otherwise specified. If the system manager has patched the monitor to apply a different default form length, he must always specify the correct value in a DFLENGTH option.
/FORM:NORMAL /HEADINGS:1 /LENGTH:66 /WIDTH:132	The values shown are used as the form definition.
/LPFORM:YES	For any line printer, spool assumes that the device driver properly processes the special LPFORM characters (refer to the <i>RSTS/E Programming Manual</i>).
/PAGE__EJECT:YES	For any line printer, SPOOL assumes that the device driver properly processes top of form characters.

5.4.3 Keyboard Spooling

If the physical device specified at start up is a keyboard device, SPOOL applies the following default values:

Switch	Result
/DFLENGTH:66	DFLENGTH has no effect unless PAGE__EJECT is explicitly specified as YES.
/FORM:NORMAL /HEADINGS:1 /LENGTH:66 /WIDTH:132	The values shown are used as the form definition.

/LPFORM:NO The terminal driver does not process LPFORM characters. For any file which contains LPFORM characters, SPOOL simulates the LPFORM effect by issuing line feed characters.

/PAGE__EJECT:NO Most terminals do not have a hardware top of form capability. SPOOL simulates top of form by issuing the correct number of line feed characters.

For terminals which do have a hardware top of form capability, the value of **PAGE__EJECT** should be YES. The value for **DFLENGTH** should reflect the actual device form length as discussed in Section 5.4.1.

For terminals with the hardware top of form capability, the system manager should set the proper characteristic of the device which accepts and processes a form feed. The TTYSET command **SET KBn;;FORM** (where n is the keyboard number of the terminal to which SPOOL directs output) performs this action.

5.4.4 Start-Up Error Processing

An error in the response to the prompt causes SPOOL to print a message and the unparsed command line in the following format.

?ERROR MESSAGE
unparsed command line

The unparsed command line is the righthand part of the response beginning at the element causing the error. The messages possible are summarized in Table 5-8.

Table 5-8: SPOOL Syntax Error Messages

Message and Meaning
?CAN'T PARSE REMAINING STRING Something illegal or undefined was found in the command line. An undefined switch or missing slash (/) character causes this error.
?DUPLICATE SWITCH Two occurrences of the same switch were found in the command line.
?ILLEGAL LOGICAL DEVICE The logical device specified was not in the form LP: or LPn: (where n is a number from 0 to 7).
?ILLEGAL OPERAND An illegal operand was found in an option or switch. For example, specifying nonnumeric characters in a switch which requires a number (/WIDTH:n) generates this error.
?ILLEGAL PHYSICAL DEVICE The device specified in the PHYSICAL option is not a line printer or keyboard device.
?MISSING OPERAND A switch requiring an operand was specified without one. For example, if a receiver identification is missing from the NAME option, this error is generated.

If SPOOL encounters no syntax errors in the response, it begins setting initial conditions. Errors in this phase are reported by printing a message in the following format.

ERROR IN SOME OPERATION — RESTARTING
error message

The program reprints its identification line and the prompt.

An error message ?DUPLICATE RECEIVER ID means that the receiver identification, either the default one or the one specified in a NAME option, is already defined for another job. The operator should type the command line again and specify a unique receiver identification in the NAME option.

The message ?NO ROOM IN RECEIVER TABLE means that a small buffer is not available to allow SPOOL to declare itself a receiving job. A later retry with the same command line should succeed.

The message *****OPSER HUNG***** or *****QUEMAN HUNG***** means that SPOOL attempted to send a message to that program and failed. Either the named program does not have an entry in the system message receiver table or the program is not processing its messages and its message limit has been reached. The operator must ascertain the cause of the problem and restart the named program.

5.4.5 SPOOL Interrupt Commands

The operator communicates with a SPOOL job by means of interrupt commands sent through the operator services program OPSER. These commands are summarized in Table 5-9. Note that any responses to an INTERRUPT command are displayed on the Operator Services Console.

Table 5-9: SPOOL Interrupt Commands

Command and Syntax	Meaning
ABO[RT]	Immediately terminates the current process and removes the request from the queue.
CON[TINUE]	Wakes up the spooling job to continue processing after a PAUSE command.
END	Closes out processing upon completing the current request, after which the spooling program must be rerun to process further requests.
FOR[M]△name/sws	Changes the current output form to one identified by name and defined by sws options. Options change current form characteristics; those characteristics not changed by an option remain at their current definitions. See the start-up option FORM description for the allowable options. Alignment is done only if the /ALIGN switch is included.

Table 5-9: SPOOL Interrupt Commands (Cont.)

Command and Syntax	Meaning
FOR[M]	Displays characteristics of the current form if no name or options are in the command.
FOR[M]△/ALIGN	Requests a forms alignment procedure for the current form.
LAS[T]	Prints the most recent message generated by the spooling job.
OFF[LINE]	Immediately terminates all processing by this spooling job (same as ABORT followed by END).
PAU[SE]	Places the spooling job in a sleep state, during which it will respond to most commands and resume normal processing in response to a CONTINUE command.*
REQ[UE]	Stops processing the current request and replaces the request in the queue so that processing will later resume at the terminating point.
RES[TART]	Reprints the current copy of a file from the beginning, including heading burst pages.
RES[TART]△JOB	Reprints the current iteration of the job, including heading burst pages.
RES[TART]:n	Reprints the current copy of the file starting at page n. If n is 0, restarts from the beginning of the file but omits the heading burst pages.
STA[TUS]	Prints a status report for the spooling job.
*Note that processing continues if any job modification command such as ABORT or REQUE is sent.	

5.4.6 SPOOL Start-Up Examples

This section shows four examples of the start-up procedure. These examples represent typical cases.

5.4.6.1 Line Printer Start Up with All Defaults — In response to the program's prompt, the operator types simply a queue name as shown in the following sample:

```
#LPO:
```

If SPOOL encounters no errors, it prints the following message:

```
DETACHING...
```

SPOOL detaches itself and leaves the terminal logged off the system.

The following are the characteristics of this copy of SPOOL:

1. The program prints only job requests which were queued to line printer unit 0 with a form name of NORMAL or queued to the general line printer queue with a form name of NORMAL.

2. The program prints on line printer unit 0.
3. The program prints one heading burst page before each job and, unless a file is queued with /NH, prints one heading burst page before each file. The width of the burst page is 132 columns, and the form length is 66 lines.
4. The program assumes that the device driver handles LPFORM characters.
5. The program assumes that the device form length is 66 lines.
6. The run burst and priority of SPOOL are those in effect when the job starts.
7. When the program receives a job from QUEMAN, it attempts to assign the printer. If the assignment fails, the program retries periodically until it succeeds. When finished processing a job request, the program deassigns the device.

The operator may later change all form characteristics (name, length, width, device form length, and headings count) through appropriate interrupt commands.

5.4.6.2 Line Printer Start Up with Narrow Width — In response to the prompt, the operator types the following command line:

```
*LP1:/FORM:NARROW/WIDTH:80/RUNBURST:12/ASSIGN
```

The following characteristics of this copy of SPOOL result.

1. The program prints only jobs queued with a form name of NARROW and queued to either the unit 1 or general line printer queue.
2. The program prints on the line printer 1 device.
3. The program prints one heading page of width 80 columns; form length is 66.
4. The program assumes the device handles top of form.
5. The program assumes the device driver handles LPFORM characters.
6. The program assumes the device form length is 66.
7. The priority of the job remains unchanged. The program sets the run burst to 12.
8. The program does not begin spooling until it can assign line printer unit 1 and does not deassign the device until it terminates.

In this example, the assumption is that line printer unit 1 is a relatively slow device and that the SPOOL program can generate output data for this unit faster than the device can print. Optimal use of the device results if the unit keeps running at full speed for the entire job request it is printing.

To effect optimal use, it is sometimes necessary to raise either the priority or run burst of the SPOOL program. The priority of a job generally affects how often it runs whereas the run burst affects the length of time the job runs once it starts running. Since the SPOOL program is usually I/O bound and waiting for the device to complete printing, there is little reason for it to run more often. Changing the priority is usually unnecessary and ineffective. Changing the run burst, however, may improve processing.

When SPOOL starts executing, it remains in the run state until one of the following events occurs:

1. The job's run burst expires. That is, the job actually executes for its full run burst.
2. The job requests some type of I/O and the system cannot immediately satisfy the request.

For the spooling program, the second event is much more likely because the system buffers only a fixed amount of data before it refuses to satisfy further requests for output. When the refusal occurs, the system activates another job and the spooler must wait until the system sends at least some of the already buffered data to the output device. (This wait condition shows as an LP or TT STATE on a SYSTAT listing.)

As long as the buffers for the device are never completely emptied, the system keeps the device running at or near full speed. To a certain point, raising the spooling job's run burst makes it more likely that the job will continue running until it completely fills the buffers for its output device. After a certain point, raising the run burst has little benefit because the job can run only until the buffers are full. Conversely, decreasing the run burst causes the system to deactivate the program before the buffers are full. There is more likelihood that the printer can empty the buffers before the system reactivates the job.

If a spooling job spends a significant time in the RN state while it is actually printing, the run burst is too low. DIGITAL recommends that the system manager raise the run burst to a value at which the job spends most of its time in an output wait state.

5.4.6.3 Keyboard Start Up on an LA36 — For this example, the assumption is that keyboard unit 5 is an LA36 terminal. To the prompt, the operator should type a command line shown in the following example:

```
#LP2:/PHYSICAL:KB5:/HEADINGS:0/ASSIGN
```

The following are the characteristics of this copy of SPOOL:

1. The program prints only jobs queued with a form name of NORMAL and queued to the unit 2 or the general line printer queue.
2. The program prints on keyboard unit 5.
3. The program prints no heading burst pages. The form length is 66.

4. The program assumes that keyboard unit 5 does not handle top of form, and, therefore, it counts lines as it prints them to simulate top of form.
5. The program assumes the device driver does not handle LPFORM characters and, therefore, it simulates LPFORM with line feeds.
6. The run burst and priority remain unchanged.
7. The program assigns keyboard unit 5 at start up and keeps it assigned until it terminates.

Since the LA36 is a slow device, processing is not slowed any further with burst pages. The /HEADINGS:0 switch suppresses printing of burst pages.

To retain any forms alignment on the LA36, the device is permanently assigned to the spooling job. This assignment prevents any other job from using the device and leaving the carriage at a position other than the top of form position. When the spooler starts processing a request, it must assume that the device is at its top of form position because the program has no way to force the device to advance to some known position. If the form is not aligned properly when SPOOL starts the output, the entire output will be misaligned.

5.4.6.4 Keyboard Start Up on an LA180 — For this example, keyboard unit 2 is a serial LA180 DECprinter. This terminal has a hardware top of form capability and obeys the XON/XOFF synchronization protocol. Also, the form length selector switch on the terminal is set to 8.5 inches (51 lines per form) and the paper form is also 51 lines per page.

In response to the prompt, the operator should type a command line as shown in the following example.

```
*LP3:/PHYSICAL:KB2:/PAGE_EJECT:YES/DFLENGTH:51/LENGTH:51-
MORE> /WIDTH:40/FORM:SPECIAL
```

The following are the characteristics of the SPOOL copy:

1. The program prints only requests queued with a form name of SPECIAL and queued to either the unit 3 or the general line printer queue.
2. The program prints on keyboard unit 2.
3. The program prints one heading page with a width of 40 columns. The form length is 51 lines.
4. The program assumes the device handles top of form.
5. The program assumes that the device driver does not handle LPFORM characters. The program therefore simulates LPFORM with line feeds.
6. The run burst and priority remain unchanged.
7. The program assigns the keyboard before it prints each request and deassigns it after printing the request.

Since the LA180 is a much faster device than the LA36, SPOOL can print a small number of burst pages to simplify distinguishing jobs and files. With a width of 40, the burst page size is at its minimum. Since the paper form length and the device form length are equal, SPOOL can position the device to top of paper form by issuing a form feed character. Since the program can position to top of form, it need not keep the unit assigned. The ASSIGN option, therefore, does not appear in the example. If the paper form length did not equal the hardware setting, however, SPOOL should keep the device assigned to preserve the top of form position.

5.4.7 Recovery from Line Printer Errors

Control of error handling in SPOOL is by interaction between the spooling job and the OPSER program. For example, if the paper runs out or jams, SPOOL encounters the ?DEVICE HUNG OR WRITE LOCKED error, discontinues processing, and sends OPSER a message. OPSER generates a message for the operator in the following manner.

```
MESSAGE      3 : 28-OCT-78 04:33 PM JOB:21 DET SPLRUN[1,100]
LPO : HUNG - JOB: [220,40]USER1
(PUT DEVICE ONLINE TO CONTINUE)
```

The message tells the operator that the spooling job for line printer unit 0 has a problem and that the device needs to be made ready again before processing can continue.

For a SPOOL program servicing a line printer, certain errors are possibly self-correcting. This self-correction exists because the device handler for the line printer tests the ready status of an off line unit every 10 seconds. If the operator removes the cause of the error (for example, fixes a paper jam) and puts the line printer on line again, the software detects the ready status and continues printing buffered characters. The SPOOL program finds that the error condition is cleared and continues processing.

If a special form is being processed and SPOOL encounters an error, SPOOL may request the operator to realign the forms before it continues processing on its own.

If the operator can not correct the error or wishes to perform some other operation, the spooled device may be left off line. This action prevents the software from restarting printing before the program processes an operator request. The INTERRUPT command is used with the proper text to access the spooling job.

SPOOL allows the operator several ways to control the restart of a queued request. The RESTART command can restart the current copy of a queued request (RESTART JOB), restart the current copy of a file in a queued request (RESTART), or restart the current copy of a file at a specific page (RESTART:nnn). The REQUE command can re-enter the current job in the queue for later processing. Upon receiving a RESTART command, the program waits for any currently buffered data to be printed before actually restarting.

The **REQUE** command causes **SPOOL** to remember the point at which processing was interrupted. Upon resuming the requested request, **SPOOL** scans for the page on which processing terminated and continues printing at the start of that page. An error may occur if a user modifies the requested request before **SPOOL** can resume the printing.

If the operator responds to an error condition with an **ABORT** command which terminates the job, **SPOOL** clears the output buffers on the line printer unit. If the device is a terminal, any buffered data is printed. The program then issues the termination message.

Before terminating a queued request, the **SPOOL** program ensures that all buffers have been emptied successfully.

5.4.8 Line Printer Output

SPOOL generates job header and file header burst pages to identify print requests and files within a print request. Both types of header page contain identification and general accounting information.

The identification consists of large, easily readable, block letters created from the character generation file **CHARS.QUE**. The job identification contains the account number of the user requesting the job and the name of the job. If no job name appeared in the **QUE** command, **SPOOL** prints the name of the first file in the request as the job name. The file identification shows the name and extension of the file.

General accounting information for the job header is on the burst page and is offset from the identification by two rows of special characters. The accounting information contains five lines of data. The first line consists of job name, current date, current time of day, and the requester's account number. The second line comprises the date and time of day of the request and the device for which the request was queued. The third line gives the system name. The fourth line gives the **QUE** options used to process the request. The fifth line gives the job copy number.

The accounting information for the file header burst page appears below the identification and is framed, above and below, by two rows of special characters. The first line of information gives the job name used when **SPOOL** printed the file and shows the current date, time of day, and account as the job header burst page does. The second line gives the copy number, the **QUE** options used on the file, and the complete specification of the file. The third line gives the record type and carriage control format used to print the file.

If **SPOOL** does not print the file because of an error, file header identification and accounting information is replaced by an error message framed above and below by five rows of special characters. The error message is standard **RSTS/E** error text.

5.4.9 Error Messages During User Output

Errors SPOOL encounters during printing are reported in the line printer output. SPOOL differentiates the error message from requested output by framing the text within five rows of special characters. Table 5-10 shows the possible error texts and related meanings.

Table 5-10: SPOOL Error Text in User Requested Output

Text and Meaning
<p>?CAN'T FIND FILE OR ACCOUNT The related file was deleted between the time it was queued and the time SPOOL tried to print it.</p>
<p>?FILE RESTART REQUESTED ON PAGE n The operator requested a restart at the indicated page or placed the job request back in the queue with the REQUE command. If the operator restarted the job request, SPOOL scans the file to find the correct page and continues printing on a new page following the message. If the operator requeued the job, this message appears when SPOOL restarts the job.</p>
<p>?JOB ABORTED Either the operator terminated the job request with an ABORT command or the K command of the QUE program was issued while the job was being printed.</p>
<p>?JOB ABORTED BECAUSE OF FILE ERROR(S) Errors in some file in the job prevented SPOOL from printing the remainder of this job.</p>
<p>?JOB REQUE AT PAGE n The operator placed this job request back in the queue with the REQUE command. SPOOL later continues the job request with a new header burst page and the text FILE RESTART REQUESTED ON PAGE n where n is the same page number shown in the REQUE message.</p>
<p>error text SPOOL encountered the RSTS/E error shown, aborted processing the request, and cleared the request from the queue.</p>

5.4.10 Changing and Aligning Forms

The FORM command can do the following operations: 1) inform the spooler of a change in form on its device, 2) request an alignment operation on the device, or 3) display information about the current form. Options with the command specify the new form definition and request alignment. The command may request alignment when a new form is specified.

To request a form change, the operator sends the FORM command with the form name and the options to define the form, to the related spooling job. If an alignment is desired for the current form, the /ALIGN option may be included with the FORM command. The SPOOL program does not recognize the form change request until it becomes idle. If it is processing a queued request, it completes the processing before it performs the forms change. (To make the change immediately, request that SPOOL requeue the current request. The program requeues the request and becomes idle.)

To perform only an alignment, specify the FORM command without any form name and with only the /ALIGN switch. Again, the command takes effect only when SPOOL becomes idle. The program performs the following actions:

1. prints characters to delineate left and right margins and prints, centered on the same line, the text TOP OF FORM.
2. generates characters to position the paper on the last line of the form.
3. prints characters to delineate the left and right margins, and prints, centered on the same line, the text END OF FORM.
4. positions the paper at top of the next form.

SPOOL generates an action request and awaits a response. The operator may inspect the alignment and type one of two responses to the action request. If the alignment is not correct, the hardware can be adjusted and the response RETRY can be sent back to the spooler. In response to RETRY, the spooler redoes steps 1 through 4 and generates another action request. When the alignment is accurate, the operator can send the text GO as a response to the action request. The spooler returns to normal processing.

Whenever SPOOL processes any FORM command, it sends a message to the operator indicating the new form characteristics in effect. In any FORM command, only those options specified in the command change the form characteristics. Thus, if the operator sends a FORM command alone, without either a form name or any options, the spooler displays the characteristics of the current form on the Operator Services Console (OSC).

5.5 Batch Processor Program – BATCH

The BATCH system program runs without user intervention and executes files of standardized commands queued for either a specific or the general batch processor. The program consists of four modules: BATCH, BATIDL, BATDEC, and BATRUN. Each module is compiled and stored in the Spooling Package Library, has a protection code of <124>, and requires permanent privilege to run. The first module sets initial conditions for batch processing and the second module checks status when no requests are being handled. The third module decodes and performs error checking on control files. The fourth module executes the decoded batch commands.

To run BATCH, type the following command while logged into the system under a privileged account.

```
RUN $BATCH
BATCH V7.0 RSTS V7.0 TIME SHARING
*
```

The BATCH programs must reside in the Spooling Package Library (SPL) in executable form. If the system manager tries to invoke a BATCH program from the SPL account and the program is not in executable form, BATCH prints the following message.

```
?BATCH must be compiled
```

The message warns the system manager that the program was not executable. Programs that are executable usually have a .BAC or .TSK extension. After the message is printed, the system manager must load a compiled version of the BATCH program into the Spooling Package Library account and invoke the program again. The ?BATCH MUST BE COMPILED message can also appear if the system manager invokes BATCH successfully, terminates the program with a CTRL/C, and then tries to invoke BATCH from memory by typing RUN. In the past, the system manager could type RUN then press the RETURN key to invoke a program that was still in memory. This procedure will no longer work. The message, therefore, instructs the system manager to include the account number of the Spooling Package Library in the RUN command. For example, RUN \$BATCH and RUN [1,10]BATCH can be valid commands but RUN cannot. These restrictions are necessary in order to allow the Spooling Package to reside on any account in the system rather than only in account [1,2].

If the account that you log into is not privileged, the system prints the ?PROTECTION VIOLATION and ?PROGRAM LOST-SORRY messages.

If the job has permanent privilege, BATCH runs and prints its identification line and the number sign (#) prompt. In response to the prompt, a specification in the following form can be given.

```
logical device:/start up switch(es)
```

The logical device is the name of the queue and the unit number within that queue from which this copy of BATCH takes job requests. This name does not specify the pseudo keyboard on which BATCH executes requests. The PHYSICAL start-up option described in Section 5.5.1 can specify the pseudo keyboard unit on which BATCH executes. The logical name must have the form BA: or BAn: where n is a unit number from 0 to 7. If BAn: is specified, the batch processor executes requests queued either to that explicit unit or the general batch queue BA:. If BA: is specified, the program executes job requests in the batch queue regardless of the unit to which they are queued.*

*The general spooling queue is useful only on systems with batch processors running with the same default conditions.

The logical device specified as the queue name must be a batch processor; the device name given must not have a logical assignment to some other device.

If only a queue name is given in response to the prompt and BATCH encounters no errors, the program establishes default processing conditions, prints the DETACHING message, and detaches itself from the terminal.

BATCH forms the default receiver identification from the queue name. Other default conditions established by BATCH are shown in the following list.

1. BATCH tolerates no errors and terminates any queued request encountering a fatal or warning error.
2. User log files are queued to the system default line printer with deletion specified.
3. The run burst remains unchanged.
4. The priority remains unchanged.

At start up, BATCH performs the same error checking and processing SPOOL performs. Refer to Section 5.4.4 for the description of SPOOL start-up error processing.

5.5.1 BATCH Start-Up Options

Options can be included with the BATCH processor designator to specialize operations. These options are in the form of switches summarized in Table 5-11.

Table 5-11: BATCH Start-Up Options

Option	Syntax and Meaning
ASSIGN	/ASS[IGN] Reserves a pseudo keyboard device to this job.
ERROR	/ERR[OR]: <i>severity</i> Sets the error <i>severity</i> default for this processor such that errors of that <i>severity</i> will be tolerated and queued requests having that or lesser <i>severity</i> will complete. /ERR[OR]:FAT[AL] Tolerates all errors. /ERR[OR]:WAR[NING] Tolerates only warning errors. /ERR[OR]:NON[E] Tolerates no errors (fatal and warning errors terminate queued requests). This is the default severity.
DELETE	/DEL[ETE] Deletes user log files after printing. This is the default condition unless the NODELETE option is specified.

(continued on next page)

Table 5-11: BATCH Start-Up Options (Cont.)

Option	Syntax and Meaning
NAME	/NAM[E]: <i>rcvrid</i> Places, in the system message receiver table, the specified receiver identification for this job instead of the default identification. The default for a batch processor is BANSP for specific processors and BASPL for the general processor.
NODELETE	/NODEL[ETE] Does not delete user log files after printing. Without this option, all log files are deleted. Requester may override this option by /NOQUE in the \$JOB command.
NOQUEUE	/NOQUE[UE] Does not queue user log files for printing. Without this option, all log files are queued without a queue name and with deletion unless changed by the /QUEUE and /DELETE switches. Requesters can not override this option.
PHYSICAL	/PHY[SICAL]: <i>dev</i> : Uses the pseudo keyboard (for example, PK1:) as the <i>device</i> on which this processor will execute user batch jobs.
PRIORITY	/PRI[ORITY]: <i>nnn</i> Sets the job priority of BATCH to <i>nnn</i> where <i>nnn</i> is a number from -120 to 120. Without this option, the priority remains unchanged.
QUEUE	/QUE[UE]:[<i>quenam</i>] Queues all user log files to the specified spooling queue (LP0: through LP7:) rather than to the general print queue. If no <i>queue name</i> is specified, all jobs are queued to the general print queue. Files are deleted after printing. Requesters may override queuing by the /NOQUE switch in the \$JOB command. Deletion of the files by the SPOOL program may be overridden by including /NODELETE as a start-up switch.
RUNBURST	/RUN[BURST]: <i>nnn</i> Sets the job run burst of BATCH to <i>nnn</i> where <i>nnn</i> is a number from 1 to 127. Without this option, the run burst remains unchanged.

5.5.2 BATCH Interrupt Commands

The operator communicates with a BATCH processor by means of interrupt commands sent through the operator services program OPSER. These commands are summarized in Table 5-12. Note that any responses to an INTERRUPT command are displayed on the Operator Services Console.

Table 5-12: BATCH Interrupt Commands

Command and Syntax	Meaning
ABO[RT]	Immediately terminates the current process and removes the request from the queue.
CON[TINUE]	Wakes up the batch processor to continue normal processing after either a PAUSE command or after a user request for operator action (for example, in the \$MESSAGE/WAIT command).

(continued on next page)

Table 5-12: BATCH Interrupt Commands (Cont.)

Command and Syntax	Meaning
END	Terminates processing upon completing the current request, after which the batch processor must be rerun to process further requests.
LAS[T]	Prints the most recent message generated by this batch processor.
NOT[ICE] <i>text</i>	Inserts the specified <i>text</i> in the user log file and precedes it with the heading NOTICE FROM OPERATOR.
OFF[LINE]	Immediately terminates all processing by this batch processor (same as END followed by ABORT).
PAUSE	PAU[SE] Places the batch processor in a sleep state, during which it will respond to most commands and resume normal processing in response to a CONTINUE command.
STA[TUS]	Prints a status report for this batch processor.

5.5.3 BATCH Start-Up Procedure

BATCH is most conveniently started by commands in the system start-up control file described in Section 3.1.

5.5.4 Operator Action Requests from Batch

The BATCH program, in processing certain commands in a user command file, generates requests for operator action. These requests come through OPSER as action requests which require some operator action and a typed answer before the BATCH processor continues processing. These action requests result from commands requiring the operator either to mount and dismount volumes or to respond to a special message with which a stall is involved. An action request is answered by the ANSWER command through OPSER.

On a mount request from BATCH, the operator is requested to mount a device by an action request with text in the following format:

```
MOUNT xx:'logical id'/'vid'/WRITE/NOWRITE/DEN:nn/PAR:nn
DEVICE?
```

If a specific volume is involved, its identification appears as text replacing 'vid'. This text is the visual identification used to distinguish the volume. The specific device type is given by xx: from one of those shown in Table 5-13.

If a specific volume is indicated in the text of the action request, the operator should mount the medium on a free unit of the device type specified by xx:. If a volume is not involved, the operator should ensure that a unit of the type specified by xx: is ready and online to the system.

When the unit is ready, the operator must type the device specification (device type and unit number with a colon) as text in the ANSWER command for that action request number. BATCH assigns that unit and continues processing the user's queued request.

Table 5-13: BATCH Device Type Designators

Designator	Meaning
CD	Either the CD11 punched card reader or the CM11 marked sense card reader
CR	CR11 high speed punched card reader
DB	RP04, RP05 or RP06 disk pack drive
DK	RK05 or RK05F disk cartridge drive
DM	RK06 disk cartridge drive
DL	RL01/RL02 disk pack drive
DP	RP02 or RP03 disk pack drive
DR	RM03 disk pack drive
DT	TU56 DECTape drive
DX	RX01/RX02 floppy disk drive
LP	Any line printer
MM	TU16, TE16, TU45, or TU77 magnetic tape drive
MS	TS04 magnetic tape drive
MT	Any magnetic tape drive
PP	High speed paper tape punch
PR	High speed paper tape reader
TT	Any keyboard line

When BATCH is done with the device, it generates an action request to dismount the volume or device. If the request involves a volume, the operator should remove the volume from the device specified and place the unit off line. To continue the BATCH processing, the operator should type the CONTINUE command as text in the ANSWER command to OPSER for that action request number. This response tells BATCH to deassign the unit and to continue processing.

5.6 Operator Communication Program – PLEASE

The PLEASE program communicates directly with the operator services program OPSER and, when OPSER is not running, sends text to the system console terminal (KB0:). The program PLEASE is compiled and stored in the system library account [1,2] with protection code of <232>. PLEASE requires privilege to execute, and the protection code allows all users to run it. The system manager can restrict use of PLEASE to permanently privileged users by changing the protection code to <124>.

Operators may run PLEASE to send commands to OPSER. Users who are not valid operators may run PLEASE to send text to the operator services console.

5.6.1 Running and Terminating PLEASE

PLEASE runs at any terminal logged into the system. The following command starts PLEASE.

```
RUN $PLEASE
PLEASE V7.0 RSTS V7.0 TIME SHARING
#
```

PLEASE prints its identification line and the number sign (#) character as a prompt to indicate it is ready to accept text.

PLEASE accepts one line of text terminated with the RETURN key. A line of text exceeding the maximum length causes the program to print the LINE TOO LONG error message and reprint the prompt. If the first character of text is not a slash (/) character, PLEASE attempts to send the text as a message to OPSER. Should the first character of text be a slash, PLEASE attempts to send the text as a command to OPSER. To signal the user that the message or command is being sent to OPSER, PLEASE prints one of the following messages.

```
MESSAGE SENT TO 'OPSER'
or
COMMAND SENT TO 'OPSER'
```

The program reprints the prompt, after which another line of text may be typed. Typing CTRL/Z or CTRL/C in response to the prompt terminates the program.

When OPSER is not active, PLEASE broadcasts messages on the system console terminal but does not send commands. In these situations, the program notifies the user with one of two messages as follows.

```
% 'OPSER' NOT ACTIVE - MESSAGE BROADCAST TO KBO:
or
% 'OPSER' NOT ACTIVE - COMMAND NOT SENT
```

The message printed on keyboard unit 0 is preceded by the text %MSG FOR 'OPSER' BUT ITS NOT ACTIVE:.

5.6.2 OPSER Commands through PLEASE

Any privileged user or valid operator can communicate with OPSER and with jobs on line to OPSER by typing a command through PLEASE. The user denotes an OPSER command by typing the slash (/) character as the first character in the line of text. The commands are summarized in Table 5-14.

Table 5-14: PLEASE Commands to OPSER

Command	Summary
/ANSΔmsgnumber:text	Responds to an action request.
/CHAΔKBN:	Changes operator services console to keyboard unit n.
/DELΔmsgnumber	Deletes outstanding action request.
/DELΔ#nn:n	Deletes n oldest action requests for job number #n.
/DET	Detaches OPSER from the OSC.
/EXI	Terminates OPSER.
/INTΔrcvrid:text	Sends unsolicited text to on-line job.
/LISΔJO	Prints on-line job table.
/LISΔOP	Prints valid operator table.
/LOGΔfile ; msglevel	Creates log file and sets message level.
/LOGΔ;msglevel	Changes message level of current log file.
/LOG	Closes current log file and stops recording.
/MESΔmsglevel	Sets message level for OSC.
/OPEΔKBn:[nnn,nnn]	Adds keyboard and account combination to valid operator table.
/OPEΔ-KBn:[nnn,nnn]	Removes keyboard and account combination from valid operator table.
/RETΔmsgnumber	Prints associated message.
/RETΔm:n	Prints oldest m action requests for job number #n.

When OPSER receives a command from PLEASE, it ensures that the user's keyboard and account numbers are present in the valid operator table. If the user is not a valid operator, OPSER broadcasts the error message ?INVALID OPER on the originating terminal.

Commands sent by a valid operator can be executed by OPSER itself or by a program on line to OPSER. If OPSER itself executes the command, any response from the command is broadcast on the originating terminal. For example, because OPSER itself processes the LIST JOBS command, the on-line job list generated appears on the originating terminal. An INTERRUPT command, however, passes text to another program. Any response from the other program is displayed on the OSC and not on the originating terminal. To see a message or response from an /INTERRUPT command, the operator should change the OSC to the current keyboard (the line on which PLEASE is running) before transmitting the command. Action requests and messages (responses to commands or text sent to OPSER) are displayed only if the currently set message level allows.

5.6.3 PLEASE as a CCL Command

If the system manager installs PLEASE as a CCL command, the operator can type PLEASE followed by a text line to send commands to OPSER. PLE is the standard abbreviation for the CCL command.

5.7 Terminating Operator Services and Spooling

Usually the system manager terminates OPSER and programs on line to OPSER by running the SHUTUP program. SHUTUP, however, terminates time-sharing operations. If it is necessary to terminate operator services and spooling without shutting the system down, the operator can type a sequence of commands to stop spooling operations in an orderly fashion. To understand the sequence, you must know about OPSER shutdown levels.

5.7.1 OPSER Shutdown Levels

When OPSER receives a command from SHUTUP to perform an orderly shutdown in logical end mode, OPSER sends an END command to each on-line job in a fixed, logical sequence. OPSER selects by shutdown level the order in which jobs are ended. On-line jobs at the lowest level terminate first. When all jobs at one level have terminated and have been removed from OPSER'S internal tables, OPSER begins sending END commands to jobs at the next highest level. When all on-line jobs have properly terminated, OPSER kills itself and SHUTUP proceeds with the system shutdown.

5.7.2 OPSER Manual Shutdown Procedure

To terminate operator services in an orderly fashion, the operator issues PLEASE commands to on-line jobs in a sequence determined by the defined shutdown level. For example:

```
PLEASE/INT LPSPL:END
```

This shutdown procedure merely mimics what OPSER does on command from SHUTUP. The operator sends the END command, in turn, to jobs at each shutdown level. The output of the LIST JOBS command gives the shutdown levels for each on-line job. When OPSER displays END messages for all jobs at one shutdown level, the operator can send the END command to all jobs at the next highest shutdown level. After the on-line job table is clear and OPSER has generated END messages for all jobs, the operator can send the EXIT command to OPSER.

5.8 BACKUP as an OPSER Controlled Program

The BACKUP program runs as a job on line to OPSER only after it has once detached from its terminal. When BACKUP is running detached, the operator can send commands to it through OPSER. If, after having been detached, BACKUP is later attached to a terminal again, the operator can communicate with BACKUP either at its terminal or through the OPSER program. The commands which BACKUP recognizes are listed in Table 5-15.

Table 5-15: BACKUP Commands through OPSER

Command and Syntax	Meaning
ABO[RT]	Terminates the BACKUP run immediately and removes BACKUP from OPSER internal tables.
CON[TINUE]	Wakes up the BACKUP program to continue processing after a PAUSE command or continues processing after an operator has performed some requested action.
DET[ACH]	Valid only if BACKUP is attached, OPSER is running, the job running BACKUP has permanent privilege, the BACKUP listing file is not on the terminal to which BACKUP is attached, and no messages are currently pending for the specific BACKUP job. If the above conditions are met, BACKUP detaches and is given a receiver identification of BACKnn where nn is its job number. All further interaction with BACKUP is done through OPSER.
END	Finishes transferring the current file and terminates processing.
LAS[T]	Prints, at the OSC, the most recent message generated by this BACKUP job.
LEG[AL]	Prints, at the OSC, a list of commands which can legally be given for this phase of the BACKUP run.
NOT[ICE]Δtext	Inserts the specified text in the BACKUP listing file. Precedes the text with the heading NOTICE FROM OPERATOR.
PAU[SE]	Places BACKUP in a sleep state, during which it responds to most commands, continues normal processing in response to a CONTINUE command, and terminates processing on an ABORT command.
STA[TUS]	Prints a status report for this BACKUP job.

While BACKUP is running detached, it generates all its requests for operator interaction through OPSER as messages. The operator uses the INTERRUPT command to respond to such requests. The operator must refer to Chapter 8 for the proper responses to BACKUP requests.



Chapter 6

System Error Package

Logging of hardware and system level software errors is an automatic function of the RSTS/E Monitor. To gain the full advantages of this error detection capability, the system manager must properly employ the programs in the system error package. The programs in this package fall into three main categories: (1) extraction and retention of error messages, (2) extraction and retention of error messages not yet retrieved at the time of a system crash, and (3) compilation and formatting of saved error messages.

Upon the occurrence of a monitor detected error, routines save critical error-related data and send a message to the ERRCPY program. The system awakens ERRCPY which then retrieves the saved data, performs minimal error message processing and stores the error message contents in a specially formatted disk file (ERRLOG.FIL). Since the number of messages which may be queued to ERRCPY is limited, the program must be running to prevent the loss of valuable diagnostic information. To minimize the size of ERRCPY, all error logging file initialization and validation functions are performed by a separate program, ERRINT.

When a system crash occurs and the system crash dump facility is enabled, the monitor preserves the contents of certain critical parts of the system in the system file [0,1] CRASH.SYS. The ANALYS program should be run immediately following recovery from a system crash to extract and format key information from the crash file. One of the functions performed by ANALYS is the creation of a separate error logging file (by default ERRCRS.FIL) containing error messages which had not been processed by ERRCPY at the time of the crash, and an ERRDIS report for the errors.

6.1 Use of the Error Logging Programs - ERRINT and ERRCPY

The system program ERRCPY reads error-related information stored in the monitor part of memory and writes it to a special disk file called \$ERRLOG.FIL. ERRCPY may be activated only by running ERRINT, the error file initialization and validation program. ERRINT performs various file checking or generation functions and chains to ERRCPY. The system manager must ensure that the proper commands are present in the START.CTL or CRASH.CTL files so that ERRINT (and therefore ERRCPY) is started and ERRCPY is active during time-sharing operations.

6.1.1 Error Logging Initialization - ERRINT

When the RSTS/E system starts up, commands in either the START.CTL or CRASH.CTL control file are executed by the INIT system program. If the command FORCE KB0: RUN \$ERRINT appears in the control file, the command RUN \$ERRINT is placed in the input buffer of terminal KB0: as if it had been typed at the terminal. When the system executes the command, the ERRINT program runs and attempts to locate the error file ERRLOG.FIL. If the file is found, ERRINT checks certain critical control information in the file. If this information is found to be invalid, ERRINT renames the file ERRLOG.TMP, generates and initializes a new ERRLOG.FIL and outputs a message notifying the user of what has been done. If ERRINT does not find ERRLOG.FIL, it simply generates and initializes the file.

Two other commands, corresponding to the responses to ERRINT questions, must also be present in the control file. Once ERRLOG.FIL has been validated or created, ERRINT prints a message telling what percentage of the file has been used and the maximum number of blocks allowed for the file. The system manager may accept the current size of the file (originally 100 blocks) or change the maximum size as desired. The program then determines whether the system manager wishes to append the contents of the special error crash file ERRCRS.FIL to the contents of ERRLOG.FIL. If the system manager indicates that this should be done, ERRINT attempts to find the file and add its contents to the end of the main error logging file. If the error crash file cannot be found, ERRINT simply ignores the command and continues.

After all its functions have been carried out, ERRINT detaches and chains to ERRCPY. At this point, the terminal on which ERRINT started is free.

The ERRINT program is usually located in the system library account [1,2]. To invoke ERRINT from this account, the system manager types the RUN \$ERRINT command then presses the RETURN key. Alternatively, the system manager can move the ERRINT program to another account and invoke ERRINT from the new location. To initiate the ERRINT program from a non-system library account, the system manager must replace the dollar sign in the RUN \$ERRINT command with the new account number enclosed in square brackets or parentheses.

In this chapter examples demonstrating the operation of system error programs will show all programs invoked from the system library account [1,2].

6.1.2 Examples of ERRINT Dialogue

```
RUN $ERRINT
ERRINT V7.0 RSTS V7.0 TIME SHARING
ERRLOG FILE IS 8% FULL
CHANGE SIZE TO < 100 >? 120
UTILIZE CRASH FILE OUTPUT (YES/NO) [NO]? NO
DETACHING
```

In this example, a valid error log file exists, has a maximum length of 100 blocks and was found to be 8% full. The system manager changed the maximum size to 120 blocks and did not wish to add the special error log file from ANALYS to the error log file. After the dialogue, the program detaches and chains to ERRCPY.

In the example below, ERRINT finds an invalid error log file, renames the invalid file as ERRLOG.TMP, creates a new file, leaves the maximum size at 100 blocks, and appends the special error log file ERRCRS.FIL to the error log file ERRLOG.FIL. Note that the 2% full message indicates a file containing only control information; this is the state of a newly generated error log file.

```
RUN $ERRINT
ERRINT V7.0 RSTS V7.0 TIME SHARING
ERROR FILE WAS FOUND TO BE INVALID
NEW FILE WAS GENERATED AND INVALID FILE RENAMED ERRLOG.TMP
ERRLOG FILE IS 2% FULL
CHANGE SIZE TO < 100 >? (RET)
UTILIZE CRASH FILE OUTPUT (YES/NO) <NO>? YES
DETACHING
```

6.1.3 Error Logging - ERRCPY

The ERRCPY program is executable only by means of a chain from ERRINT.

ERRCPY runs detached, processes incoming error messages, and writes each message as a variable length record to a nonspanned sequential file (ERRLOG.FIL). When its message queue is empty, ERRCPY enters the sleep state and is swapped out.

NOTE

Messages are queued to ERRCPY by the monitor regardless of the presence or absence of ERRCPY. The first message queued is always a "power fail/start up" message and has an automatic sequence number of 1.

Upon receiving a special error message from the SHUTUP program, ERRCPY logs the message, closes the error file and kills itself.

6.1.4 Description of the Error File (ERRLOG.FIL)

The file ERRLOG.FIL consists of a one block header followed by a variable number of data blocks. The header block contains 64 8-byte logical records. The first 62 logical records in the header block contain counters for each of 62

possible error types. The last two logical records in the block contain general control information.

The header contains the following information for each error type:

1. number of error records logged
2. number of errors logged (including repeat counts)
3. number of errors received (including repeat counts)
4. maximum number of error records which may be logged

The general control information consists of the following six total fields:

1. total error records logged (including repeat counts)
2. total errors received (including repeat counts)
3. block number for start of next error record
4. offset within the block for next error record
5. maximum size of error file in blocks
6. bytes reserved for future use

The data portion of this sequential file contains variable length records. Records are packed, but no record spans two disk blocks. A record consists of the error message as received from the monitor preceded by a one-byte field giving the length of the record.

6.2 Displaying Errors - ERRDIS

The error display program ERRDIS provides the system manager with a convenient method of displaying previously logged errors.

ERRDIS provides four functions:

1. Provides a summary report by error type and unit number (where applicable) of all errors logged to the error file.
2. Provides a detailed report of one or all error types logged between any two user-selected date and time pairs.
3. Zeroes the contents of the error file following the generation of a report.
4. Provides a list of potentially bad disk blocks.

ERRDIS consists of two modules ERRDIS and ERRDET. The first module performs functions 1 and 3. To perform the remaining two functions, ERRDIS chains to ERRDET which returns control to ERRDIS upon completion.

NOTE

Before running ERRDIS for the first time, the system manager must run the ERRBLD program which then creates the ERRDAT.FIL error file. Normally, this function is automatically performed when the Error Package is built at the time the system is generated.

To run ERRBLD, type the following:

```
RUN $ERRBLD
```

After ERRBLD prints its header information, it builds ERRDAT.FIL then returns control to the BASIC-PLUS command level (the READY prompt is printed).

6.2.1 Running ERRDIS

The system manager or a privileged user runs the ERRDIS program by typing the following command.

```
RUN $ERRDIS
```

The user then selects optional modes of ERRDIS operation by means of an interactive dialogue. In the dialogue description shown in Table 6-1, prompts are numbered for reference. Default answers to prompts are enclosed in angle brackets and are selected by typing the RETURN key in response to the prompt. Most responses may be abbreviated with 2 characters. Characters not required in a response are enclosed in square brackets []. For example, HE[LP] appearing in a prompt would require the user to type HE, although HEL or HELP would also be acceptable.

Table 6-1: ERRDIS Dialogue Explanation

Step	Prompt and Description
1	ERRDIS V7.0 RSTS V7.0 TIME SHARING ERRDIS prints identification.
2	INPUT FILE <[1,2]ERRLOG.FIL>? Enter specification of file to be processed by ERRDIS. The file must be formatted as an error logging file.
3	OUTPUT TO <KB:ERRDIS.OUT>? Enter specification of file to which ERRDIS should output the report.

(continued on next page)

Table 6-1: ERRDIS Dialogue Explanation (Cont.)

Step	Prompt and Description
4	<p>HE[LP] , BA[D BLOCKS] , SU[MMARY] OR FU[LL] REPORT <SUMMARY>?</p> <p>Select one of four possible reports.</p> <ul style="list-style-type: none"> a. HE[LP]. Prints help file, followed by a list of mnemonics corresponding to all possible error types. b. BA[D BLOCKS]. Outputs a report of possible bad blocks detected from the disk errors logged in the input file entered in Step 2. c. SU[MMARY]. Outputs a report of the number of errors detected for each error type and for each unit number within a specific error type. The next dialogue step is 10. d. FU[LL]. Outputs a report of the detailed contents of each selected error record in the error file. Selection of a specific record depends on the answers to steps 5 through 9 in the ERRDIS dialogue.
5	<p>SPECIFIC ERROR TYPE <ALL>?</p> <p>Enter a two character mnemonic identifying the specific error type to be selected for processing. The default answer selects all error types. Note that if 'ALL' is selected, you may also attach the /NOTAPE switch if you want magnetic tape errors to be omitted from the error listing.</p>
6	<p>STARTING DATE <FIRST ERROR>?</p> <p>Enter the date of the earliest error to be selected for processing. The format is dd-mmm-yy. The default answer is the date of the first error in the error log file. If a default is typed, the next step is 8.</p>
7	<p>STARTING TIME <FIRST ERROR>?</p> <p>Enter the time of the earliest error to be selected for processing. The format is hh:mm. Default is the time of the first error in the error log file on the selected date.</p>
8	<p>ENDING DATE <LAST ERROR>?</p> <p>Enter the date of the latest error to be selected for processing. The format is dd-mmm-yy. Default is the date of the last error in the error log file. If the default is selected then the next dialogue step is 11.</p>
9	<p>ENDING TIME <LAST ERROR>?</p> <p>Enter the time of the latest error to be selected for processing. The format is hh:mm. Default is the time of the last error in the error log file on the selected ending date. Next dialogue step is 11.</p>
10	<p>LIST BAD BLOCKS (YES/NO) <YES>?</p> <p>A Y[ES] answer causes ERRDIS to provide a list of possible bad blocks following the summary report.</p>
11	<p>ZERO ERROR FILE UPON COMPLETION (YES/NO) <NO>?</p> <p>A YES answer causes ERRDIS to zero the previously specified error log file upon successful generation of the desired report. (Type all three characters.)</p>

ERRDIS now processes the selected error records, formats and outputs the selected report, zeroes the error log file (if requested in step 11) and returns to step 2. At that point, typing CTRL/Z in response to any question returns the terminal to the READY state.

6.2.1.1 Help Report — The help report prints information on the use of the ERRDIS program, then prints a list of the error mnemonics used by the other reports.

6.2.2 Summary Report

A summary report supplies general information on all errors logged. The first two columns list the 2-character error mnemonic and full error description. The column titled TOTAL REC/LOG lists the total errors received by the ERRCPY program and the total found in the error file. These two totals may differ due to the limits on the number of errors which may be logged and due to the limit on size of error file. The columns titled UNIT NUMBERS are the total errors generated by each device on a controller.

6.2.3 Bad Block Report

The ERRDET program detects possible bad blocks by checking the error logging file for such errors as unreadable data on the last retry of a disk read operation or unreadable headers encountered during a disk read or write operation. The criteria for a possible bad block depend on the type of disk.

Each possible bad block report includes the logical block number, the logical name or pack identification of the disk, and the physical device name of the disk. Using the logical block number printed by ERRDIS, the system manager may add the possible bad block to the bad block file with the BADS? suboption of the initialization code option REFRESH. Refer to the *RSTS/E System Generation Manual* for information on adding bad blocks.

NOTE

The system manager should exercise caution before adding ERRDIS listed possible bad blocks to the bad block file. Errors may be caused by hardware controller or disk drive problems and not as a result of corrupt disk surfaces.

6.2.4 Full Report

Each error displayed includes a title line containing the error code mnemonic, the error description, a sequence number and the date and time the error occurred. The sequence numbers are maintained by the monitor from startup to shutdown. They are not reset by a crash and subsequent automatic restart. The sequence numbers start at 1.

There are four basic categories for individual errors displayed within the full error report. They are:

1. Nonperipheral errors such as processor traps and memory parity.
2. Disk errors.
3. Nondisk peripheral device errors such as DECtape errors and magnetic tape errors.

4. Other errors such as the shutdown error received from SHUTUP and the missed errors.

The program automatically checks for and reports possible bad blocks for disk errors included in the scan.

NOTE

When the monitor detects a hardware or software error, it sends information about the error to the ERRCPY program. ERRCPY can accept and file error information until the number of queued errors reaches 39. All errors that the monitor detects beyond this are logged as "missed errors". Only after the queue begins to empty and the number of error queued to ERRCPY falls below the maximum of 39, can the monitor begin to send the normal error information. That is, while the queue is full the monitor, rather than sending the usual error information to ERRCPY, reports all subsequent errors as "missed errors". In addition, the monitor reports "missed errors" if it does not have a sufficient number of small buffers available to store error information. When the number of small buffers drops below 75, the monitor reports any detected error to ERRCPY as a "missed error".

6.2.4.1 User Description in Full Report — All errors except shutdown and missed errors usually include a user description. The name fields in the user description relate to a specific job whose number appears in the first field (Job number). Under certain conditions, a job number of 0 is possible. In this case, the user description is not listed. Table 6-2 summarizes the user description data.

Table 6-2: User Description Data

Heading	Meaning
Job Number	The number of the job which requested the I/O in the case of disk errors, the job owning the device in the case of nondisk peripheral errors or the job running at the time of error in the case of non-peripheral errors.
KB Number	The keyboard number of the job, followed by DET if the job is detached.
Account	The project-programmer number of the user who created the job.
Program Name	The name of the program running in the job virtual address space (low segment).
User Job Physical Addr	The current physical memory address of the job.
User Job Size	The size of the user program running in the low segment of the job virtual address space.
Control Parameters*	Monitor control information.
RTS Name	The name of the run-time system mapped into the highest segment of the user's virtual address space.
RTS Physical Address	The address of the location in physical memory where the RTS resides.
*A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.	

6.2.4.2 Disk Error Detailed Description — The fields in the detailed description for disk errors are shown in Table 6-3. The program prints an additional field if the error signalled a possible bad block.

Table 6-3: Disk Error Detailed Description

Heading	Meaning																
I/O Status	See Timeout Indicator below.																
Timeout Indicator	Timeout and I/O Status are considered as a pair. One of the following states can exist: <table><tr><th>Timeout Indicator</th><th>I/O Status</th><th>Meaning</th></tr><tr><td>0</td><td>0</td><td>Idle</td></tr><tr><td>x</td><td>1</td><td>SEEK in progress</td></tr><tr><td>0</td><td>1</td><td>Waiting for Read/Write</td></tr><tr><td>1</td><td>x</td><td>Timeout</td></tr></table> Where x = any nonzero value	Timeout Indicator	I/O Status	Meaning	0	0	Idle	x	1	SEEK in progress	0	1	Waiting for Read/Write	1	x	Timeout	
Timeout Indicator	I/O Status	Meaning															
0	0	Idle															
x	1	SEEK in progress															
0	1	Waiting for Read/Write															
1	x	Timeout															
Offset Position	The contents of Offset Position for disks having offset capabilities follows: <table><tr><td>0</td><td>= centerline position</td></tr><tr><td>20</td><td>= +x offset</td></tr><tr><td>220</td><td>= -x offset</td></tr><tr><td>40</td><td>= +2x offset</td></tr><tr><td>240</td><td>= -2x offset</td></tr><tr><td>60</td><td>= +3x offset</td></tr><tr><td>260</td><td>= -3x offset</td></tr><tr><td>100</td><td>= offset unknown</td></tr></table> Where x = the number of micro inches of offset (a drive type parameter).	0	= centerline position	20	= +x offset	220	= -x offset	40	= +2x offset	240	= -2x offset	60	= +3x offset	260	= -3x offset	100	= offset unknown
0	= centerline position																
20	= +x offset																
220	= -x offset																
40	= +2x offset																
240	= -2x offset																
60	= +3x offset																
260	= -3x offset																
100	= offset unknown																
Overlapped Seek Ind.	377 = overlap seek 000 = non-overlap seek																
Unit Size in DC's	Size (in octal) of the disk expressed as the number of device clusters.																
Device Cluster Size	Device cluster size for this drive.																
Pack Cluster Size	Pack cluster size for this disk.																
Logical Name	The pack identification if no system wide logical name was given at mount time or the system logical name entered at mount time.																
Physical Name	The physical name and unit number.																
Logical Block Number	The block number where the error occurred.																
DSQ*	The disk request queue entry block.																
SCB or WCB*	The SCB (Small Control Block) or the WCB (Window Control Block) is present for disk errors which resulted from a file request by the user job.																

*A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.

(continued on next page)

Table 6-3: Disk Error Detailed Description (Cont.)

Heading	Meaning
CSR Address	The CSR address is the base address of the set of registers in the I/O page belonging to the device which caused the error.
Contents of Registers	<p>This section contains the name of each readable device register, its contents and, possibly, an abbreviated description of various error bits which were found to be set to one in the register. A maximum of 21 error bits are stored for each error type. The register mnemonics, error bit descriptions, and error code descriptions for all error types are contained in a data file ERRDAT.FIL. ERRDIS requires this file in order to run.</p> <p>A detailed description of registers and bit descriptions may be found in the <i>PDP-11 Peripherals Handbook</i>. The register mnemonics displayed by ERRDIS correspond to those given in the manual.</p>

6.2.4.3 Nondisk Peripheral Device Error Detailed Description — Table 6-4 lists the fields in the detailed description for the nondisk peripheral device error report.

Table 6-4: Nondisk Peripheral Device Format

Heading	Meaning
Timeout Indicator	Nonzero indicates that an expected response was not received from the device in the allotted time.
Physical Name	The physical name and unit number of the device.
DDB*	The DDB (Device Data Block) contains various parameters required by the system to control the device.
DDB* Extension	Supplementary DDB information.
CSR Address	See description under disk format.
Contents of Registers	See description under disk format.
*A detailed description of this entry is beyond the scope of this manual. The entry is intended for use by DIGITAL maintenance personnel.	

6.2.4.4 Nonperipheral Error Detailed Description — Detailed descriptions of nonperipheral error fields are available from the related processor handbook. Table 6-5 gives an abbreviated description of the items listed.

Table 6-5: Nonperipheral Error Format

Heading	Meaning
R0-R5	Processor registers 0 through 5. Each register is listed on a separate line.
Virtual PC	The address (within the user virtual address space) of the instruction being executed at the time of error.
Physical PC	As above except the physical memory address of the instruction being executed at the time of the error.
Processor Status	The processor status word.
Stack Pointer	The address of the current top of stack.
(SP) (SP+2)	The contents of the top two words in the stack.
(PC-6) (PC-4) (PC-2) (PC)	The contents of the 3 words preceding the current program PC and the contents of the current PC.
CPU ID	If it exists, the contents of the CPU identification register.
CPU ERR	If it exists, the contents of the CPU error register.
MED X	If CPU has Maintenance Examine/Deposit Instruction.
Contents of Registers	Certain nonperipheral errors include contents of key registers. The format is identical to the contents of registers for disk and nondisk peripheral errors.

NOTE

The PDP-11 computer always executes startup by interrupting through a vector address. The same vector is used for normal start up and for a powerfail recovery. The error log for a normal start up reflects a sequence number of 1 and the contents of R0-R5 are all zeroes. This is not the case for a powerfail restart.

6.2.5 PROGRAM LOST-SORRY Errors

The ?PROGRAM LOST-SORRY message occurs when BASIC-PLUS attempts to run a program and cannot. To avoid damage to other jobs, BASIC-PLUS clears the job image from memory and returns control to the user. If possible, BASIC-PLUS prints a message qualifying the error causing the loss of the program. In several cases, the qualifying error message is not printed and the system manager must check the error log to determine the cause. Note that other run-time systems installed on RSTS/E may also return this message.

6.2.5.1 Checksum Error on a .BAC File — A checksum error is generated when the computed and stored checksums do not agree. The stored checksum is written in the .BAC file when BASIC-PLUS creates a compiled image. Whenever BASIC-PLUS loads a .BAC file, it recomputes the checksum and matches the result with the stored checksum. If the checksums are not equal, BASIC-PLUS generates an error to be logged by the monitor, returns the ?PROGRAM LOST-SORRY error to the user, and aborts execution.

Checksum errors can be caused by one of the following hardware errors:

1. a disk read error undetected by the disk error detection logic.
2. a disk write error caused by a bus transmission error. In this case, incorrectly transmitted data are recorded on the disk but the hardware does not detect any error when the data is read.
3. incorrect data on disk because of undetected memory parity errors. Such errors are caused by defective non-parity memory or by a class of multiple bit errors undetected by parity memory hardware.

Normally, the user can recompile the particular program to correct the checksum problem.

Frequent checksum errors indicate hardware problems, system problems, faulty application software, or malicious destruction of compiled files. Running diagnostic programs can uncover hardware faults. Other causes are not easy to detect because RSTS/E has no specific protection (other than file protection codes) against opening and writing into a compiled file.

6.2.5.2 Unrecoverable Disk Error Reading a .BAC File — An unrecoverable disk error during the loading of a .BAC file results in a ?PROGRAM LOST-SORRY message. Unrecoverable errors are caused by a variety of conditions such as bad disk blocks, dust, drive unsafe conditions, or a transient hardware problem in the disk subsystem. Sometimes these errors generate a warning message such as ?DISK ERROR DURING SWAP. The ERRCPY program logs such errors in the error logging file.

To ascertain the type of error, the system manager can compare the time at which the ?PROGRAM LOST-SORRY message appeared with the time at which the error was logged. If the disk block causing the error is not bad, recompiling the program to rewrite the failing block normally corrects the problem. It is safer, however, to add the failing block to the bad block file and ensure that the failing block is not reused.

6.2.5.3 Improper .BAC File Size — If a .BAC file size is improper, the system generates the ?PROGRAM LOST-SORRY message when attempting to load the file. The error message is prefaced by the ?ILLEGAL BYTE COUNT FOR I/O error message and is repeatable but not logged in the error logging file. The error condition is easily removed by recompiling the program from the .BAS file.

The size of a .BAC file must obey the following relations.

$$8 \leq \text{SIZE} + 1 \leq 64$$

and

$$((\text{SIZE} + 1) \text{ AND } 3) = 0$$

The relations mean that a compiled file must be between 2K and 16K words (inclusive) and the number of blocks in the file (SIZE) must be an integer one less than a multiple of 4.

6.2.5.4 Mismatch of BASIC-PLUS Version Numbers — When BASIC-PLUS runs or chains to a .BAC file, it checks the version number stored in the file with the version number of the run-time system being used. The compiler stores its version number in a .BAC file when the compiled image is created. If the version numbers do not match, the system generates the ?PROGRAM LOST-SORRY message. ERRCPY does not log any error.

The *RSTS/E System Release Notes* tell whether a recompilation of current programs is necessary for a new version of BASIC-PLUS.

6.3 Analyzing System Crashes – ANALYS

When a system crash occurs in RSTS/E, time-sharing operations are halted. If the required conditions described in Section 2.3.2 are met, the critical contents of memory are written into the CRASH.SYS file in the system account [0,1] and the system disk is bootstrapped in automatic restart mode.

The occurrence of a later system crash causes the CRASH.SYS file to be overwritten. Therefore, the system manager is provided with a means of retaining information in the CRASH.SYS file. This means is the ANALYS system program. The use of the ANALYS system program to document system crashes requires that the CRASH.SYS file be created at REFRESH time and that the crash dump feature be enabled at system start-up time.

The ANALYS system program is invoked by the following command:

```
RUN $ANALYS
```

In response, a header line and three successive query lines are printed as follows:

```
ANALYS V7.0 RSTS V7.0 TIME SHARING
INPUT <[0,1]CRASH.SYS>? (RET)
OUTPUT <KB:ANALYS.DMP>?CRASH.DMP/WIDE
CRASH ERROR LOG FILENAME<[1,2]ERRCRS.FIL>? (RET)
READY
```

The first question line requests the name of the file to be analyzed, which by default is CRASH.SYS in account [0,1]. The system manager need only type the RETURN key to apply the default. Because the system manager may wish to keep copies of different crash files, the program can accept a file other than CRASH.SYS. ANALYS uses the currently installed monitor SIL file to extract symbolic references. If the crash file being analyzed is associated with other than the currently installed monitor, the system manager must append the /SIL: switch with the name of the related monitor SIL. This switch allows DIGITAL to analyze crash and .SIL files submitted with an SPR.

The second question line requests a disk file or a device designator for the output medium, which, for example purposes, is CRASH.DMP on the system disk. The /WIDE switch, attached to the CRASH.DMP file specification in the previous example, produces an annotated version of a memory dump. If a dump of memory is not desired, the system manager can append the /NODUMP switch to the response.

The third question allows the system manager to recover certain error information from the crash file and write it to a designated file. ANALYS retrieves error information saved at the time of a system crash but not written to the system error logging file ERRLOG.FIL. If the system manager types only the RETURN key, ANALYS writes this information to the default file ERRCRS.FIL in the system library account. The ERRDIS program accepts this file as input and produces an error log report from its contents. To retain a single continuous error logging file, the system manager may use an ERRINT option.

Normally, ANALYS takes about 5 minutes to run. Upon completion of the output, program execution automatically terminates and READY appears at the terminal.

Output of the ANALYS system program run supplies valuable hardware and software information which can be used by a software specialist to ascertain possible causes of system crashes.

The ANALYS output includes a report similar to SYSTAT in addition to a memory dump of the critical contents of memory. If the system is configured with DECnet/E, the report will provide information similar to that included in a NIP (Network Information Program) report. Refer to the *RSTS/E System User's Guide* for an explanation of SYSTAT, and the *RSTS/E DECnet/E User's Guide* for NIP.

ANALYS reports an error code in the crash dump data. Table 6-6 shows the error codes.

The ANALYS output is supplemented by the error logging printouts. The system manager should compare items in the data labelled VIRTUAL PROGRAM COUNTER and PROCESSOR STATUS in the ANALYS printout with the data labelled Virtual PC and Processor Status in the ERRDIS report. If the values match, the error in the ERRDIS report is the one which caused the crash.

Table 6-6: System Crash Error Code

Error* Code (octal)	Meaning
-1(177777)	Unknown vector
-2(177776)	Jump to 0
41	Trap to 4
42	Trap to 10
43	Trap to 250 (Memory management violation)
44	Kernel SP Stack overflow
46	Trap to 114 (Parity memory error)
0 or other	Forced dump
* The power fail error has no code associated with it.	

The PROCESSOR STATUS data tells the current and previous modes of the processor. RSTS/E does not use supervisor mode. The only valid modes are kernel and user. If bit 11 of the PSW is 1, the processor has two register sets; if bit 11 is 0, the processor has only one register set.

To obtain crash information printouts automatically, include the proper commands in the CRASH.CTL file. The commands can run ANALYS to preserve the crash information and run ERRDIS to create a report.

6.4 Octal Debugging Tool – ODT

The system program ODT opens a file, a peripheral device, or memory as an address space and allows a user to examine and change word or byte locations within the address space. As auxiliary operations, the user can list the contents of certain conventional table locations in the operating system.

The program immediately interprets and executes each character as the user types it. This action is termed ODT submode or delimiterless character input mode and differs from the procedure used by other system programs which interpret input only after the user enters an entire line of characters. Since ODT performs processing based upon single characters typed at the terminal, its language is highly interpretive and interactive. It thus provides a quick and efficient means of finding errors in program and data files and changing data in those files for testing purposes.

NOTE

Because ODT lacks protection against user mistakes, it is advised that only experienced users employ it to perform testing and error correction on a system's data base.

The program accesses and manipulates data in word and byte locations based on octal values. The word is the 16-bit PDP-11 word and can have a value between 0 (octal) and 177777 (octal), the limit imposed by 16 bits. A word has a high order (odd address) and low order (even address) byte. A byte can have a value between 0 (octal) and 377 (octal) – the limit that can be represented by 8 bits. For the purposes of clarity in this chapter, the following symbols express the octal values printed by ODT.

Symbol	Meaning
n	Represents an octal integer between 0 and 17. The use of 8 or 9 generates an error.
k	Represents an octal value up to 6 digits in length. If more than 6 digits are specified or a value greater than 177777 (octal) is specified, ODT truncates the value to the low order (rightmost) 16 bits. If the octal value is preceded by a minus sign, ODT uses the 2's complement value of the number.

For example, ODT interprets the following values as shown.

1	000001
-1	177777 (2's complement)
400	000400
-177730	000050 (2's complement)
1234567	034567 (truncated to low-order 16-bits)

The user can represent a location within the permissible address space by typing an octal value or an expression which reduces to an octal value. The following are the correct forms and the interpretation by ODT.

Symbol	Meaning
k	The 6 digit octal value of k.
n,k	The resultant address is the value of k added to the contents of the relocation register specified by n. Relocation registers are numbered from 0 to 17 (octal). See Section 6.4.4 for more information concerning relocation registers.

The special characters and symbols in Table 6-7 are recognized by RSTS/E ODT and explained in the remainder of the section.

Table 6-7: ODT Characters and Symbols

Character(s) or Symbols	Meaning
/ k/	Opens the previously open location as a word or opens the location designated by k as a word.
\ k\	Opens the previously open location as a byte or opens the location designated by k as a byte.
" k"	Gives the ASCII representation of the currently open or last previously open location or of the location specified by k.
% k%	Gives the ASCII representation of the Radix-50 value in the currently open or last previously open location or in the location specified by k.

(continued on next page)

Table 6-7: ODT Characters and Symbols (Cont.)

Character(s) or Symbols	Meaning
RET key k followed by RET key	Closes the currently open location or modifies the contents of the currently open location with the value k and closes it.
LF key k followed by LF key	Closes the currently open location and opens the next sequential location or modifies the contents of the currently open location with the value of k before closing it and opening the next sequential location.
↑ or ^	Closes the currently open location and opens the preceding sequential location. (On some terminals, the ^ or ↑ character is typed by pressing the SHIFT and N keys simultaneously.)
— or ←	Takes the contents of the currently open location as a PC relative offset and calculates the next location to be opened; closes the currently open location and opens the location thus evaluated.
@ k@	Takes the contents of the currently open location as an absolute address, closes the currently open location, and opens and prints the contents of the location thus evaluated. If @ is preceded by k, the value k replaces the contents of the currently open location before it is closed.
> k>	Takes the low order byte of the currently open location as a relative branch offset and calculates the address of the next location to be opened; closes the currently open location and opens and prints the contents of the relative branch location thus evaluated. If > is preceded by k, the value k replaces the contents of the currently open location before it is closed.
<	Closes the currently open location and opens the last location explicitly open. Returns ODT to the origin of a sequence of relative locations determined by —, @, and > character operations.
,	Separates a relocation register number from an octal value. ODT adds the contents of the specified relocation register to the octal value following the comma and forms a relocatable address.
;	Separates multiple values in a list request using the L character and in a register operation using the R character.
.	Specifies the last explicitly open location similar to that used by the < character operation.
+ space bar	Adds the preceding value and following value and uses the result.
-	Subtracts the following value from the preceding value and uses the result.
R	Resets all relocation registers to -1 (177777).
nR	Resets relocation register n to -1 (177777).
k;nR	Sets relocation register n to the value k.
F	Sets relocation calculation for list requests using L character.
1F	Disables relocation calculation set by F character.

(continued on next page)

Table 6-7: ODT Characters and Symbols (Cont.)

Character(s) or Symbols	Meaning
C	Prints out Monitor table symbolic names and memory addresses.
\$S	Prints out the processor status word.
Q	Uses the last quantity printed by ODT.
k1;k2L	Prints contents of locations k1 through k2 at the terminal.
1;k1;k2L	Prints contents of location k1 through k2 on line printer unit 0.
2;k1;k2L	Prints contents of location k1 through k2 on another device. ODT prints DEVICE question, to which user types the device designator.

6.4.1 Running and Terminating ODT

The user runs ODT by typing the following command.

```
RUN $ODT
ODT V7.0 RSTS V7.0 TIME SHARING
FILE<MEMORY>?
```

ODT runs and prints the FILE question. The user response to this question determines the way ODT runs and what address space ODT accesses. The possible responses are listed and described in Table 6-8.

Table 6-8: ODT File Question Responses

Response	Meaning
Type the RET key only.	Allows read access to memory only if user is privileged.
Type the ALT MODE key only.	Allows read access to the file CRASH.SYS in account [0,1].
Type the LF key only.	Same as the RET key.
Type a file specification followed by the RET key or the ALTMODE key.	Allows read access to the file on the device specified. If no device is specified, the system disk is used.
Type the file specification followed by the LF key.	Allows read and write access to the file specified.

ODT determines the address space by the response to the FILE question and indicates its readiness to accept commands by printing the * character. For example:

```
FILE<MEMORY>? ABC.DAT LF
*
```

• ODT opens for read and write access the file ABC.DAT on the system disk under the current account. To terminate ODT, type CTRL/Z in response to the * character. For example:

• *^Z
READY

ODT closes any file currently open and returns control to the system command level.

6.4.2 Access to Locations in the Address Space (/ and \)

ODT accesses the address space as either a word or a byte. The user indicates the type of access by specifying the slash (/) or back slash (\) character. For example:

*1000/

The user types the address 1000 (octal) followed by the / character. ODT opens the location as a word, generates a space, prints the 6-digit octal contents of the word, generates another space, and leaves the location open for change. The following demonstrates the results.

*1000/ 004100 (RET)
*

ODT prints the contents of location 1000 as 004100. To close the location, type the RETURN key. ODT closes that location, prints an * character on the next line, and does not open a new location.

The user specifies the \ character to open a location as a byte. For example:

*1000\ 100 (RET)
*

ODT opens location 1000 as a byte, generates a space, prints the 3-digit octal contents of the byte, generates another space, and leaves the location open for change. To close the location, type the RETURN key. ODT closes the location, prints an * character on the next line, and does not open a new location.

To change location 1000, type a new 6-digit octal value followed by the RETURN key. For example, if location 1000 is open as a word:

*1000/ 004100 004000 (RET)
*

ODT replaces the current contents 004100 with the specified contents 004000, closes the location, and prints the * character on the next line.

To determine the contents of the current word location, type the / character. For example, if the current location is 1000, the following occurs.

```
* / 004000
```

ODT opens the current location, generates a space, prints the 6-digit contents and generates another space.

If the user types the LINE FEED key while a word location is open, ODT closes the current location and opens the next sequential location. For example, if location 1000 is open as follows:

```
* 1000/ 004000
```

and the user types the LINE FEED key, the following occurs:

```
* 1000/ 004000 (LF)
001002/ 012345
```

ODT generates a carriage return and line feed and prints the address of the next location, followed by the / and space characters and the contents of the word. The new location 1002 is open. Repetitive use of the LINE FEED causes ODT to open and display the contents of sequential locations.

If the user types the LINE FEED key while a byte location is open, ODT performs the same actions as described for a word location except that the next location is treated as a byte.

6.4.2.1 Opening the Preceding Location (↑ or ^) — Typing the up arrow (↑) character or the circumflex (^) character when a location is open causes ODT to close the currently open location and to open the immediately preceding location.

NOTE

On ASR-33 type terminals, type the ^ character by pressing the SHIFT key and the N key simultaneously.

For example, if two sequential locations are successively opened, typing the ^ character opens the immediately preceding location.

```
* 1000/ 002345 (LF)
001002/ 012740 ^
001000/ 002345
```

Typing the ^ character closes location 1002 and opens location 1000 and prints its contents. If a byte location is currently open, typing the ^ character opens the immediately preceding byte location. If the user types a value followed by the ^ character, ODT replaces the current contents with the specified value before closing the location.

If the user types the ^ character and a location is not currently open, ODT opens and prints the contents of the last currently open word or byte location. For example:

```
*1000/ 002345 (RET)
*^
001000/ 002345
```

ODT prints the address and the contents of the word location on the next line.

6.4.2.2 Opening a PC Relative Location (← or —) — Typing the ← character or — character when a location is currently open causes ODT to add 2 to the address of the current location, to add the resultant sum to the contents of the current location, and to open the location specified by the final sum. For example:

```
*1000/ 000040.
001042/ 012345
```

ODT closes location 1000, adds 2 to the address (1000), and adds the resultant 1002 to the contents (40) of the current location. As a result, ODT opens location 1042 and prints its contents. This method of calculating the next location to open is similar to that used in relative addressing by the program counter in the PDP-11 computer. Such a method of address calculation is for position independent code.

If the contents of the current location is an odd value, ODT opens and prints the contents of the low-order byte of the PC relative location. If the user types a value followed by the — character, ODT modifies the current contents and uses the new value to calculate the PC relative address.

6.4.2.3 Opening an Absolute Location (@) — Typing the commercial at (@) character when a location is currently open causes ODT to take the contents of the current location as the address of the next location to open. As a result, ODT closes the current location, opens the calculated location and prints its contents. For example:

```
*1006/ 001024 @
001024/ 000500
```

ODT uses the contents of the current location (1024) as the next location to open.

NOTE

On ASR-33 type terminals, type the @ character by pressing the SHIFT key and the P key simultaneously.

If the user types a value followed by the @ character, ODT changes the contents of the current location to the value and uses the new value to determine the next location to open. The method is equivalent to absolute addressing on the PDP-11 computer where the contents of the location following an

instruction are taken as the address of the operand. The address is absolute since it remains constant regardless of where in memory the assembled instruction is executed.

6.4.2.4 Opening a Relative Branch Offset Location (>) — Typing the greater than (>) character when a location is currently open causes ODT to use the signed value of the low-order byte of the current location to determine an offset from the current location. ODT uses the final sum to open the next location and print its contents. For example:

```
*1032/ 000407 >
001052/ 001456
```

ODT takes the contents of the low order byte (007) and multiplies by 2 to give 16 (octal). Next, ODT adds 2 to the address of the current location (1032) to give 1034. Finally, ODT adds these two quantities (1034 + 16) to give the address of the word (1052) to open. ODT closes the currently open location, opens the calculated location, and prints the address and the contents on the next line.

If the user specifies a value followed by the > character, ODT modifies the contents of the currently open location and uses the low order byte of the new value to calculate the relative branch offset location. For example:

```
*1032/ 000407 301>
000636/ 000010
```

ODT interprets the byte value 301 as a negative value since the high order bit is 1. The absolute value of 301 is 77 (octal) which is multiplied by 2 to give 176 (octal). ODT subtracts the relative branch offset (176) from the address plus 2 of the current location to give 636 as the address of the next location to open. ODT opens the new location and prints its contents.

6.4.2.5 Returning to an Interrupted Sequence (<) — Typing the less than (<) character causes ODT to close the currently open location and open the last explicitly open location. This command is useful when a user has typed the —, @, and > characters, or any sequence thereof, and wishes to open the locations from which ODT calculated subsequent relative locations. For example:

```
*1032/ 000301 >
000636/ 000010 @
000010/ 123456 <
001032/ 000301
```

After typing the > and @ characters, ODT opens location 10. The user returns to the last explicitly open location by typing the < character. ODT opens and prints the contents of location 1032, the last location explicitly opened.

6.4.3 Printing the Contents of Locations

The user can type the L character in three ways to print the contents of locations in the address space open by ODT. For example, to print the contents of a certain range of addresses, specify the start and end addresses with the L command as follows:

```
*0;776L
```

ODT prints at the terminal, the octal contents of each word between address 000000 and 000776. Beginning each line of the printout, ODT prints the address of the first word on the line. The user can turn the printing on and off by typing CTRL/O.

To print a listing on line printer unit 0, type the L command as follows:

```
*1;0;776L
```

ODT prints, on line printer unit 0, the octal contents of each word between addresses 000000 and 000776. To specify another unit, type 2 preceding the command:

```
*2;0;776L  
DEVICE? LP1:
```

Note that if the user types any number other than 1 before the L command, ODT prints the DEVICE question, to which the user types the device designator of the line printer unit to be used.

6.4.3.1 Printing ASCII Format (") — Typing the quotation marks (") character when a location is currently open causes ODT to print the ASCII representation of the word or byte. For example:

```
*1000\ 101 " A (LF)  
001001\ 103 " C (RET)  
*1000/ 41501
```

If the currently open location is open as a word, typing the " character causes ODT to print the 2-character representation of the word. For example:

```
*1032/ 034567 " W9
```

The low order byte contains 167 (octal) which is W and the high order byte contains 071 (octal) which is 9.

If the user types the " character and a location is not currently open, ODT prints the ASCII representation of the previously open location.

6.4.3.2 Printing Radix-50 Format (%) — Typing the percent (%) character when a location is currently open causes ODT to print the 3-character ASCII representation of the Radix-50 word. For example:

```
*1000/ 034567 % IG1
```

If the user types a value preceding the % character, ODT interprets it as the address whose contents are to be interpreted and printed. For example:

```
*1000% IG1
```

ODT interprets 1000 as the address to use. If the user types the % character and a location is not currently open, ODT prints the 3-character ASCII representation of the previously open location.

6.4.4 Relocation Registers

ODT has available 16 relocation registers which the user can employ to specify relative addresses. ODT initially sets the relocation registers to -1 (177777, the highest possible address) to prevent inadvertent errors in address calculation. The user sets a relocation register by typing the relative address, followed by a semicolon and the specification of one of the 16 relocation registers. For example, to set relocation register 0 to 1000, type the following:

```
*1000;0R  
*
```

The user can subsequently use the value in relocation register 0 as an offset or a base address in specifying a location. For example, to open location 1032, as a word, the user types the following:

```
*0,32/ 000010
```

ODT adds the offset 32 to the contents of relocation register 0 to open the location. Since relocation register 0 contains 1000, ODT opens location 1032.

To reset the contents of all relocation registers to -1, the user need type only the R character. For example:

```
*R  
*
```

ODT generates the carriage return and line feed operation and prints the * character again. To reset the contents of any one register to -1, simply specify the register number followed by the R character. For example:

```
*1R  
*
```

The above command resets relocation register 1 to 177777 (octal).

ODT treats registers 0 through 7 differently from 10 (octal) through 17 (octal) when used with disk files. For registers numbered 0 through 7, the leftmost 3 digits specify a block number and the rightmost 3 digits specify a byte location within the block. For example:

```
000017
```

The value designates byte 17 (octal) in block 0 of the file. The following value:

3412

designates byte 412 (octal) in block 3 of the file. (These registers are helpful in accessing data which is partitioned in 512-byte blocks.) For registers numbered 10 (octal) through 17 (octal), the value specifies an absolute block number in the disk file. For example, the value 1000 (octal) specifies block 1000 (octal) of the disk file.

Use relocation registers 10 (octal) through 17 (octal) to access blocks in large files (>65K blocks) that are beyond block number 65,535. To display the contents of an address in this range, type the offset in blocks and a semicolon, followed by one of the relocation registers 10 through 17. ODT accepts your input, performs the line feed operation, and issues the asterisk prompt. At this point, one of the relocation registers contains a block address. You can use this value as a base address to access other locations. To begin accessing addresses beyond the block offset contained in the relocation register, type:

1. the number of the relocation register with the block address (10 - 17),
2. a comma,
3. the offset beyond the block address that must be accessed, and
4. a slash (/) character to terminate the response.

ODT then adds the block address to the offset and displays the contents of the newly calculated address. For example, if you need to display the contents of a large file starting at block number 303240 (which is decimal block number 100,000), enter the following information:

*303240;10R

You must first convert the decimal block number 100,000 to octal and enter this value along with the number of a relocation register, in this case register 10. ODT loads relocation register 10, then prints the asterisk prompt after the information is entered. The relocation register now contains the block offset, 303240. This represents the address (30324000) from which you can access other locations. If you want to examine the contents of address location 303240030, for example, type the following information:

*10R,30/ 000010

Since the increment beyond the base address 303240000 is offset 30 (octal), you must enter this value to access the contents of address location 303240030. ODT adds the octal number 30 to address 303240000 and displays the contents of location 303240030 to the right of the slash character. The contents of the resulting address location is 000010.

To print the contents of locations based on a fixed offset from the relocation registers, type the F character and subsequently use the L character. For example:

```
*F
*1;0;3000L
*
```

The F character conditions ODT to calculate relocated addresses for a print-out. The next command tells ODT to add the offsets from 0 to 3000 to the value of relocation register 0 and print the contents of those resultant locations on line printer unit 0. The procedure is repeated for the values of relocation registers 1 through 7. The resultant printout contains a listing of the contents of addresses n,0 through n,3000 where n is between 0 and 7 (the relocation registers).

To turn off calculation of relocation addresses for a printout, type 1F. For example:

```
*1F
*
```

Subsequent listing requests generate printout for actual addresses rather than relocated addresses.

6.4.5 Interpretive Address Quantities (Q and .)

ODT uses the variable Q to store the last value which it printed at the terminal. The user can type Q to designate the value so stored.

ODT performs any valid operation requested and automatically extracts the value from Q. For example, if the user desired to increase the value in an open location by a certain increment, the following steps could be taken:

```
*1342/ 173214 Q+10 (RET)
*/ 173224
```

The user types 1342/ to open that location as a word. ODT prints the contents of the location and stores that value in Q. The user subsequently types Q+10 followed by the RETURN key. ODT adds 10 to the value in Q and modifies the current location with the sum. The user thus does not have to retype the number or calculate the sum. To verify that ODT has changed the location properly, the user types the / character. ODT opens the last previously open location (1342), prints the contents, and, additionally, updates the value Q with the most recently printed quantity.

The period (.) character indicates the currently open or last explicitly open location and can be typed to indicate the current address for ODT operations. This is the same address used by the < character described in Section 6.4.2.5. In most cases, the . character value is the address used by ODT when the user types the /, \, ", %, and LINE FEED characters. For example, to open as a word a location 16 bytes from the last explicitly open location, the user types the following:

```
*.,+16/ 012345
```

ODT adds 16 (octal) to the address given by the character, opens the resultant address, and prints its contents.

6.4.6 Error Procedures

If the user types an invalid or unrecognized character, ODT prints a question mark (?) character, generates a carriage return and a line feed, and prints the * character. For example:

```
*1008?  
*
```

ODT indicates that the character 8 is an error. The user must retype the number correctly.

If ODT encounters an error while performing output to a device, it prints the message I/O ERROR? followed by the * character. The user must correct the device error and type the command again.

6.4.7 Reading a Pack Identification with ODT

Each RSTS/E file-structured disk has a pack label (or identification) which is required for logically mounting the disk on the RSTS/E system. The pack label consists of one to six alphanumeric characters. This information should be recorded visibly, usually on the pack or cartridge cover. Thus, when you need to mount a disk, you can readily see what its pack identification label is.

If you get a ?PACK IDS DON'T MATCH error, you may have mistyped the pack identification label on the MOUNT request. You may have also mounted the wrong pack or cartridge on the drive. The external label should tell you what the pack label is. If not, then you can use ODT to determine the pack label. Note, however, that it is much easier to use the DSKINT program to tell you what the label is.

ODT can be used to read the pack identification label on a disk. The pack identification is stored in Radix-50 format on a relative, fixed location on all disks - offset 14 (octal) in the first pack cluster. For disks with a device cluster size of 1, the absolute address is 1014 (octal). For disks with a device cluster size of 2, the address is 2014 (octal). For cluster sizes of 4 and 8, the addresses are 4014 (octal) and 10014 (octal). The following sample printout shows the procedure for a disk with a cluster size of 1.

```
RUN $ODT  
ODT V7.0 RSTS V7.0 TIME SHARING  
FILE<MEMORY>? DK1: (RET)  
  
*1014/ xxxxxx % PAC (LF)  
1016/ xxxxxx % KID (RET)  
* (type CTRL/Z; it does not echo.)  
  
READY
```

In response to the file question, type the name and unit of the device on which the pack or cartridge is mounted. Terminate the response with either the RETURN key or the ESC key to open the device for read access in nonfile structured mode. ODT prints the asterisk (*) character when it is ready to accept commands. Typing 1014 followed by the slash (/) character opens location 1014 (octal) as a word. ODT responds by printing the octal contents of the location opened. Typing the percent (%) character causes ODT to print the first three characters of the pack identification, which is in Radix-50 format. Typing the LINE FEED key after the first three characters are printed causes ODT to close the current location and open the next location. Repeating the preceding procedure results in the second three characters of the pack identification, after which the RETURN key is typed to close the location and place ODT in command mode again.

Chapter 7

System Utility Operations

7.1 General Utility Operations — UTILTY

The system manager controls system operation on line using the UTILTY system program.

The commands of the UTILTY system program are presented in Table 7-1 for reference. The following sections explain the commands according to their functional uses. The UTILTY program has a protection code of <124> so that only privileged users can execute its privileged operations.

Table 7-1: UTILTY Commands

Category	Command Format* and Use
Operational Control	<p>LOGINS Raises, to the maximum possible, the number of jobs allowed to be logged into the system at any one time and enables users to log into the system.</p> <p>SETΔLOGINSΔx Sets to x the number of user jobs which can log into the system after the command is executed.</p> <p>NOΔLOGINS Prevents further logins.</p> <p>SENDΔKBn:Δxxx ALL Causes the text string xxx to be printed on the keyboard unit n or all keyboards.</p>
*The notation Δ indicates that a space character is required.	

(continued on next page)

Table 7-1: UTILITY Commands (Cont.)

Category	Command Format and Use
Operational Control (cont.)	<p>FORCEΔKBn:Δxxx ALL Causes the text string xxx to be forced into the input buffers of keyboard unit n or all keyboards as if it had been typed in.</p> <p>To force a control character combination to a terminal, place the circumflex (^) or up arrow (↑) character as the first character of the text to force, followed only by the proper character and the RETURN character. (That is, no other text should follow the control character combination.) If either the circumflex or up arrow character is the only character of text, UTILITY forces the CTRL/C to the terminal.</p> <p>If either the circumflex or up arrow character is the first character of more than 2 characters of text UTILITY forces a CTRL/C to the terminal ahead of the text.</p> <p>KILLΔn Immediately terminates user job specified by n.</p> <p>PRIORITYΔnΔp Sets to p the priority of job n.</p> <p>SUSPENDΔn Sets the priority of job n to -128.</p> <p>RESUMEΔnΔp Resumes job n at priority p. Same as PRIORITY command.</p> <p>RUNBURSTΔnΔQ Changes job n's run burst to Q.</p> <p>SIZEΔnΔQ Changes job n's maximum size to Q.</p> <p>DETACHΔn DETACHΔn/CLOSE Detaches job n and closes all nonzero channels on which console terminal is open.</p> <p>DETACHΔn/NOCLOSE Detaches job n but does not close all nonzero channels on which console terminal is open.</p> <p>HANGUPΔKBn: Disconnects the remote line specified by KBn:</p> <p>DISABLEΔKBn: Disables the terminal interface of keyboard unit n until the start of the next time-sharing session.</p> <p>DATE dd-mmm-yy Sets the RSTS/E system date to the value of day, month and year (for example, 14-NOV-78).</p> <p>TIME hh:mm Sets the RSTS/E 24 hour clock to the value of hours and minutes (for example, 21:52 means 9:52 P.M.).</p>

(continued on next page)

Table 7-1: UTILITY Commands (Cont.)

Category	Command Format and Use
Caching Control	<p>DISABLE CACHE Disables disk caching.</p> <p>LIST△CACHE Prints the current cache settings.</p> <p>ENABLE△CACHE Enables data and directory caching.</p> <p>/ALL Cache all data transfers regardless of UFD entry or OPEN MODE.</p> <p>/FILE Files are cached depending on their UFD entry or OPEN MODE.</p> <p>/NOFILE Files will never be cached.</p> <p>/BUFF Small buffers can be used for directory caching (not legal if data caching is generated).</p> <p>/NOBUFF Small buffers cannot be used for directory caching.</p> <p>/CL:n Specifies cache cluster size, where n is 1, 2, 4, or 8 blocks.</p> <p>/LIMIT:n Specifies the maximum number of cache clusters (based on available memory).</p> <p>/DIR:n Specifies the maximum number of cache clusters used for directory caching.</p> <p>/DATA:n Specifies the maximum number of cache clusters used for data caching.</p> <p>FLAG <filename> Modifies the cache settings for a file.</p> <p>/CACHE The file will always be cached on OPEN.</p> <p>/NOCACHE The file will not automatically be cached on OPEN.</p> <p>/SEQ If cached, the file will be cached sequentially.</p> <p>/RAN Unless otherwise specified, the file will be cached randomly if cached at all.</p> <p>DISABLE△CACHE Disables data and directory caching.</p>
System File Control	<p>ADD△SWAPFILE△n△dev:[filnam] Adds the specified dev:[filnam] as a swapping file in slot n. Use /SIZE:n to create the file at n blocks (contiguous).</p> <p>REMOVE△SWAPFILE△n Disables the swapping file on slot n and closes the file.</p>

(continued on next page)

Table 7-1: UTILITY Commands (Cont.)

Category	Command Format and Use
System File Control (cont.)	<p>ADD△OVERLAY△dev:filnam Uses the specified file for the FIP overlay code.</p> <p>REMOVE△OVERLAY Removes the overlay file. The system resumes using the overlay portion of the installed .SIL file.</p> <p>ADD△ERROR△dev:filnam Adds the specified file as the file from which the system will extract error message text.</p> <p>REMOVE△ERROR Removes the error message file. The system resumes using the established default error message file.</p>
Disk Management	<p>MOUNT△dev:id Logically mounts the disk residing on the specified drive unit with the pack identification label (id) so that data on the disk can be properly accessed by the system. For example:</p> <p>MOUNT DK1:PRIV1 mounts the pack with label PRIV1 on the physical device DK1:.</p> <p>MOUNT△dev:id/option(s)</p> <p>/LOGICAL:xxxxxx Uses xxxxxx as the logical name for the pack in place of pack id.</p> <p>/NOLOGICAL Does not use any logical name.</p> <p>/PRIVATE Mounts this public disk as a private disk.</p> <p>/RONLY Does not allow write access to this disk.</p> <p>DISMOUNT△dev: Logically dismounts a disk pack on the physical drive specified by dev:. Must be used prior to removing the cartridge from the disk drive unit.</p> <p>LOCK△dev: Places the disk mounted on drive unit dev: in a state which prevents files from being OPENed by nonprivileged users.</p> <p>UNLOCK△dev: Allows nonprivileged users to OPEN files on the disk mounted on drive unit dev:.</p> <p>CLEAN△dev: Rebuilds the SAT (Storage Allocation Table) of the pack mounted and locked on disk drive dev:. To be used only when message DEVICE NEEDS CLEANING is printed and the device is locked. Because the ONLCLN program performs various disk cleaning operations in addition to rebuilding the Storage Allocation Table, DIGITAL recommends you use ONLCLN rather than the CLEAN command to affect a more thorough disk cleaning procedure. Refer to Section 7.6 for a description of ONLCLN.</p>

(continued on next page)

Table 7-1: UTILITY Commands (Cont.)

Category	Command Format and Use
Disk Management (cont.)	<p>QUOTAΔ[n,m]Δq Sets the maximum number of 256-word blocks account [n,m] is allowed to retain at logout time to the decimal number q. A zero value for q means unlimited quota. The value for q may be from 0 to 65535.</p> <p>CHANGEΔ[n,m]passwd Alters the password of account [n,m] to the 6-character alphanumeric passwd. If you use the asterisk (*) character as the password, it prevents any user from logging into the system on this account.</p> <p>ZEROΔdev:Δ[n,m] Deletes all files from account [n,m] on the disk drive unit specified by dev:.</p>
Run Time System Control	<p>ADDΔdev:[n,m]name Adds to the run-time system table the run-time system specified by name.</p> <p>ADDΔdev:[n,m]name/option(s) The run-time system in account [n,m] on device dev: defaults to [0,1] if not specified. If you do not specify dev:, your default device is the public disk structure.</p> <p>/EMT[:x] The run-time system uses the RSTS/E special prefix EMT feature. The value x denotes the EMT prefix code from 0 to 255. If x is not specified, the code of 255 is used.</p> <p>/EXT:eee Uses eee as a default extension for files executed under this run-time system.</p> <p>/MIN:s Uses s in K words as minimum size job allowed.</p> <p>/MAX:s Uses s in K words as maximum job size.</p> <p>/ADDR:xxx Uses xxx (1K-word section of memory) as the default load address.</p> <p>/KBM This RTS acts as a keyboard monitor.</p> <p>/1USER This RTS is limited to use by one job at a time.</p> <p>/RW This RTS allows read and write memory access.</p> <p>/LOGERR This RTS logs errors occurring under its control to the system error log.</p> <p>/REMOVE This RTS should be removed immediately from memory when all users have switched to another RTS.</p> <p>/POSITION[:n] Places RTS description block in position n after the default run-time system in the linked list.</p>

(continued on next page)

Table 7-1: UTILITY Commands (Cont.)

Category	Command Format and Use
Run Time System Control (cont.)	<p>/STAY This RTS should be permanently resident in memory.</p> <p>REMOVE△name Removes from the run-time system table the entry for the module specified by name.</p> <p>LOAD△name Loads into memory the run-time system specified by name.</p> <p>/STAY Same as for ADD.</p> <p>/ADDR:xxx The 1K-word section of memory that is to be used as the load address.</p> <p>UNLOAD△name Unloads from memory the run-time system specified by name.</p> <p>NAME△name=filnam Associates the run-time system specified by name with the related file on disk. The designated file can be any valid RSTS/E full disk file specification. When execution of this file is attempted, the system will use the name as the RTS for the program.</p>
Resident Library Control	<p>ADD△LIBRARY△dev:[n,m]name Adds the specified name as a resident library. The resident library in account [n,m] on device dev: defaults to [0,1] if not specified. If you do not specify dev:, your default device is the public disk structure.</p> <p>/ADDR:n Loads the resident library into 1K section of memory at address n. You must specify the /ADDR switch with the ADD LIBRARY option.</p> <p>/STAY Makes the resident library permanently resident.</p> <p>/1USER Allows only one user to access the resident library.</p> <p>/RW Allows the resident library to be mapped read/write rather than read-only.</p> <p>/NOLOGERR Prevents errors that occur within the resident library from being recorded in the system error log.</p> <p>/REMOVE Removes the resident library from memory when all jobs using it switch to another resident library.</p> <p>REMOVE△LIBRARY△name Removes the resident library that is indicated by the specified name.</p> <p>LOAD△LIBRARY△name Loads the resident library specified by name into memory.</p> <p>/ADDR:n Loads the resident library into 1K section of memory at address n.</p>

(continued on next page)

Table 7-1: UTILTY Commands (Cont.)

Category	Command Format and Use
Resident Library Control (cont.)	<p>/STAY Makes the resident library permanently resident.</p> <p>UNLOAD△LIBRARY△name Removes the resident library specified by name from memory.</p>
Concise Command Language	<p>LIST△CCL Lists current valid CCL commands in the order in which their definitions are entered into the CCL command table.</p> <p>CCL△com=prog Defines the CCL com to run the program specified by prog. The prog can be specified in the following format:</p> <p>dev:[ppn]name.ext;PRIV n</p> <p>If device is omitted, the public structure is used. PRIV allows the program to retain temporary privileges for programs entered at a nonzero line number. If PRIV is omitted, any entry, at a nonzero line number, to a program stored with temporary privilege causes the program to drop the temporary privileges. The value n specifies the line number to start execution. The default for n is 0.</p> <p>CCL△com= Deletes the CCL command specified by com.</p>
System Logical Names	<p>ADD△LOGICAL△dev:[nnn,nnn]name Adds, as an additional entry in the system logical name table, the name for the specified device type and unit. If an account is specified, the ADD LOGICAL command also associates that project-programmer number with the logical name.</p> <p>REMOVE△LOGICAL△name Removes from the system logical name table the entire entry for the name specified or removes the name specified for its associated disk entry.</p> <p>CHANGE△LOGICAL△dev:name Changes, for the disk specified, the old logical name to the name specified.</p> <p>LIST△LOGICAL Prints a list of device specifications which currently have a system logical name assigned. Includes any account specification associated with the logical name.</p>
Program Control	<p>HELP Prints a list of valid UTILTY commands and their meanings at the key-board printer.</p> <p>CTRL/C Terminates execution of the current operation and the UTILTY run.</p> <p>CTRL/Z Allows completion of pending operations before termination of the UTILTY run.</p> <p>EXIT Allows completion of pending operations before termination of the UTILTY run.</p>

7.1.1 Running and Terminating UTILTY

The system manager or a privileged user executes the UTILTY system program by typing the following command while logged in at any terminal.

```
RUN $UTILTY
```

The program responds by printing one header and one prompt as follows:

```
UTILTY V7.0 RSTS V7.0 TIME SHARING  
#
```

after which any valid command can be specified. A list of all valid commands is printed if the HELP command is typed. The program prints the # character after each command is executed.

Termination of the UTILTY system program can be properly accomplished by typing either CTRL/Z or the EXIT command. (CTRL/Z is echoed as ^Z on the terminal.) If CTRL/Z or EXIT is typed at any time, the operations currently pending are properly completed, and control is returned to command level. The completion of the UTILTY run is signalled by the message printed at the keyboard by the keyboard monitor (BASIC-PLUS prints READY).

If CTRL/C is typed in order to terminate the program run, any operation in progress is interrupted immediately. Control is returned to command level. (CTRL/C is echoed at the keyboard printer as ^C). Typing CTRL/C to terminate a UTILTY program run is not considered proper termination since the effect of uncompleted internal system operations is unpredictable. The system manager is advised to use CTRL/Z or the EXIT command to terminate UTILTY.

The UTILTY system program is designed to run by the CCL command UTILTY (with an abbreviation UT). If UT-ILTY has been added as a CCL command, typing UT followed by a valid command causes UTILTY to run, execute the command, and return to system command level. Typing UT alone runs UTILTY which prints the prompt character (#).

7.1.2 Principles of Disk Management

Certain commands of the UTILTY system program are used to manage disks on the RSTS/E system. Such commands are presented for reference in Table 7-1 under the category of disk management. To more easily convey the proper use of individual disk management commands, the following description of each command is presented with a discussion of its function and value.

7.1.2.1 Preparing a Disk for Use on a Drive — To be used on the RSTS/E system, a disk pack or cartridge must be made known to the system by being logically mounted. In this process, you supply the system with a physical device and unit number on which the disk resides, and the disk's logical name, the pack label (or identification). The system will check the pack label against the information actually written on the pack. If the pack label you

supply to the system does not match what is on the pack, the PACKID's DON'T MATCH error occurs. Otherwise, the system records the presence of the disk pack in the table of mounted disks.

A new disk must first be formatted and initialized for use on the RSTS/E system by the initialization option DSKINT. A previously formatted disk can be initialized by the DSKINT system program. The pack label, a 6-character alphanumeric name assigned to the disk during the DSKINT dialogue, is required to logically mount the disk. Once a disk is initialized, the REACT program can create accounts on it.

When the RSTS/E system is started, only the system disk is logically mounted. Ensure that other packs in the public structure, and perhaps other private packs, are mounted by commands from the START.CTL or CRASH.CTL file. You can logically mount a private disk on the system by means of the UTILITY system program MOUNT command or use the CCL MOUNT command described in the *RSTS/E System User's Guide*. The procedure to mount a disk is summarized in Table 7-2.

Table 7-2: Procedures for Using Disk Packs and Cartridges

Add a Pack or Cartridge to the System*	Remove a Pack or Cartridge from the System
<ol style="list-style-type: none"> 1. Use the SYSTAT system program to ensure that the drive unit is free. 2. Place the pack in the drive. When the drive is READY, write enable it for normal usage. Write lock the disk if it is to be mounted read only. 3. Invoke the UTILITY system program and use the MOUNT command to mount the new pack logically. 4. CLEAN the pack if necessary. To clean a disk mounted read only, it must be dismounted, mounted write enabled, cleaned, dismounted and mounted read only. DIGITAL recommends that you use the ONLCLN program if you need to perform a thorough disk cleaning operation. Refer to Section 7.6 for a description of ONLCLN. 5. Use UNLOCK command to free device for use. 	<ol style="list-style-type: none"> 1. Invoke UTILITY and use LOCK command on the drive unit containing the pack to be removed. 2. Use the SYSTAT program disk status report to determine the number of OPEN files. If zero, proceed. If non-zero, wait until all files are closed before proceeding. 3. With no files open on the device unit, invoke the UTILITY system program and use the DISMOUNT command to notify the system that the pack is being removed. 4. Remove pack from disk drive unit.
<p>*The disk pack is assumed to have been initialized and formatted using the DSKINT initialization option or the DSKINT system program (Section 7.5). See the <i>RSTS/E System Generation Manual</i> for a description of DSKINT initialization option.</p>	

Unless the disk is mounted for read only access, the error message ?DEVICE HUNG OR WRITE LOCKED indicates that the disk unit is not write enabled. Under no circumstances should a public disk be write protected, mounted read only, or dismounted during normal system operation. Files are created and accessed in the public structure without explicit reference to a device. By removing a public disk during time sharing, the system manager denies users access to files which reside on that device.

Certain other errors can occur during the MOUNT operation. The error message ?NOT A VALID DEVICE indicates that disk type or the unit specified is not configured on the system. If the ?ILLEGAL SYS() USAGE error occurs, one of two conditions is possible. The disk specified either is already mounted or is the system disk. The error ?PACK IDS DON'T MATCH means that the pack identification specified does not correspond to the one on the disk. (The ODT program, as described in Section 6.4.7, can be run to read the label on disk.) If the ?FATAL DISK PACK MOUNT ERROR message is printed, the disk has an unreadable storage allocation table, a cluster size larger than 16, or some other nonrecoverable error condition. The disk may have been written on an operating system other than RSTS/E.

To properly remove a private disk pack or cartridge from a drive unit, actions similar to those of preparing the disk for use must be employed. For example, if the system manager desires to replace a private disk with another private disk, careful procedures must be followed. The first step is to ensure that no files are open on the drive unit. This can be accomplished by requesting a disk status report through the SYSTAT or display system program. The next step is to lock the device unit by using the LOCK command. This action ensures that nonprivileged user programs cannot open any more files on the disk.

When the disk to be removed has no OPEN files and has been LOCKed, the system manager next must logically dismount the device with the DISMOUNT command. If any files are open on the disk, UTILITY prints the ?ACCOUNT OR DEVICE IN USE error message and does not dismount the disk. After the dismounting action is completed, the disk pack can safely be removed from the drive unit. Any pack which is to replace the pack removed must undergo the procedures previously described for proper use of the pack.

If a disk was not dismounted properly after it was last used, UTILITY encounters the ?DISK PACK NEEDS 'CLEANing' error when the MOUNT command is executed. UTILITY prints the warning message ?DISK PACK NEEDS 'CLEANing'. The disk is mounted but locked. The SYSTAT display program prints the text LCK in the disk structure report. The lock condition prevents nonprivileged jobs from accessing the disk but allows the disk to be cleaned.

An improperly dismounted disk requires cleaning because the Storage Allocation Table (SAT) in the SATT.SYS file does not reflect the actual allocation of storage. This condition results because the SAT is manipulated in memory and is not written back to the disk immediately. Proper dismounting procedures ensure that the SAT is updated on disk before the disk is physically removed from the drive.

Because the MOUNT command of UTILTY does not clean disks, the CLEAN command must be used to rebuild the SAT on the disk. The CLEAN routines read all the directory blocks and create a new SAT truly reflecting occupied storage. Additionally, all file access counts are reset to 0 and all files with .TMP extensions are deleted. If the CLEAN command is not executed and the disk is dismounted, the error condition no longer applies but the SAT is still unreliable and requires cleaning. (The DISMOUNT command assumes the proper procedures were followed and clears the bit which indicates that cleaning is required.)

NOTE

The CLEAN suboption of REFRESH and the ONLCLN program perform more extensive verification of the disk structure. DIGITAL recommends that, where possible, the system manager use either of these methods in place of the UTILTY version.

The MOUNT command of UTILTY does not unlock the disk. To allow non-privileged jobs access to the disk, the UNLOCK command must be executed. Nonprivileged jobs can open files on the disk after it is unlocked.

The MOUNT command not only mounts a disk pack or cartridge but also determines whether a logical name should be placed in the system logical name table. If the mount operation succeeds, the system scans the entire name table. If the name is not in use, the system places the pack identification in the table entry for the specific device and unit. A pack identification which duplicates a logical name in use is not entered in the table. UTILTY notifies the system manager in this case by printing the message LOGICAL NAME WAS NOT UNIQUE. The disk is properly mounted but cannot be accessed by a system logical name. The CHANGE LOGICAL command can be issued to establish a unique logical name for that specific disk unit entry in the table.

7.1.2.2 Disk Mounting Options — With the MOUNT command, the system manager can specify options (switches) to modify the logical name for the disk, suppress any logical name, mount a public disk as private and mount a disk for read only access.

The /LOGICAL: and /NOLOGICAL switches control the system logical name for that specific device and unit. By specifying the /NOLOGICAL switch, the system manager prevents the pack identification from being entered in the system logical name table. This action keeps the pack identification from being reported by SYSTAT and the display program and limits access to the disk by its physical name. At a later time, the CHANGE LOGICAL command can establish a logical name for the disk or the ADD LOGICAL command can create an additional logical name entry by which the disk can be accessed.

The /LOGICAL: switch tells UTILTY to replace the pack identification with the logical name specified with the switch. UTILTY first mounts the disk and then determines whether the logical name is unique. If the name is not in use as a system logical name, the system places the specified name, instead of the

pack identification, in the table entry for the specific device and unit. If the name duplicates one currently in use for any device, UTILITY prints the message LOGICAL NAME WAS NOT UNIQUE. The disk is mounted but has no associated logical name. The CHANGE LOGICAL command can be issued to establish a unique logical name for the disk or the ADD LOGICAL command can create an additional logical name entry by which the disk can be accessed.

The system manager can mount a public system disk from another RSTS/E system by simply adding the /PRIVATE and /RONLY switches to the MOUNT command. The former switch causes the system to treat the disk as a private pack and the latter switch prevents any user from creating or altering files on the disk. These actions preserve the integrity of the foreign and host system's public file structures and allow privileged users read access to data files on the public pack or cartridge. The /RONLY switch, moreover, can be specified to restrict write access to any private disk. If /RONLY is used, the disk can (and should) be write locked.

7.1.2.3 Removing All Files from an Account — Before an account can be deleted from the RSTS/E system or deleted from a private disk, the account must contain no files. The ZERO command of UTILITY removes all files from an account on a device. The ZERO command does not remove write-protected files. If the account contains write-protected files, use the PIP RENAME switch or the BASIC-PLUS NAME-AS statement to change the files protection prior to zeroing the account. The REACT program (see Section 4.1.2) is then used to delete the account.

7.1.2.4 Changing Quota and/or Password of an Account — Each user account in the RSTS/E system has associated with it a quota of disk storage that the account can retain at logout time and a password which allows access to the system. The quota and password are specified by the system manager when the account is created.

The system manager can change the quota by use of the QUOTA command of UTILITY. The system manager specifies the account number and the decimal number of 256-word blocks of disk storage the account can retain at logout time. If zero is specified for the quota, the account can retain an unrestricted number of blocks.

The system manager can change the password of an account by using the CHANGE command.

Note that the QUOTA and CHANGE commands do not affect the entries in the ACCT.SYS file.

7.1.3 Operational Control of the System

Certain commands of the UTILITY system program control the operation of the system. Such commands are listed for reference in Table 7-1 under the category of operational control. These commands and examples of their possible usage are described in this section.

7.1.3.1 Controlling the Number of Logged-In Jobs — When the system manager can monitor and control system operation while logged into a RSTS/E system. By use of the SYSTAT or the display system program, the manager observes the number of free small buffers. If, for example, the number of free small buffers drops below 10% of the total buffers available, system efficiency declines. The system manager can remedy the possible degradation of system efficiency by preventing more users from logging into the system. The number of logged-in jobs allowed can be adjusted by the SET LOGINS, NO LOGINS or LOGINS commands.

The NO LOGINS command sets to 1 the number of jobs allowed to be logged into the system. After the NO LOGINS command is executed, further logins are disabled. If a user attempts to log into the system or to execute a logged out command, the monitor prints the ?NO LOGINS message. Jobs already logged into the system can continue running. Jobs such as BATCH which log other jobs into the system cannot, however, successfully process further requests. The SHUTUP system program disables further logins in preparing to shut down time sharing.

To prevent the system manager from being locked out of the system, RSTS/E allows jobs to be logged in at the system console terminal* (KB0:) regardless of the number of logins currently allowed. Of course, this capability remains in effect as long as a swapping slot is available to handle a new job.

The SET LOGINS command adjusts the number of allowed logins to a designated number. This command is useful to limit or extend the load allowed on the system. The number of allowed logins is never allowed to be 0 or greater than the value of JOB MAX set at system start up time. If 0 is specified in the command, the number set is 1. The number of logins allowed can never exceed the capacity of the swapping space installed on the system. If a number greater than the maximum allowed is specified in the command, the system sets the maximum possible. The free buffer status report of SYSTAT or the display programs tells the maximum number of logins currently allowed.

The system manager normally uses the SET LOGINS and LOGINS commands with ADD SWAPFILE commands to adjust the number of logins allowed. After swapping space is added to the system, the LOGINS command recalculates the swapping capacity, factors in the currently allowed maximum swapping size (SWAP MAX), and increases the number of allowed logins. The SET LOGINS command is used to decrease the number of allowed logins to prepare for the removal of swapping space.

7.1.3.2 Broadcasting Messages to Terminals — The system manager can communicate with one user or with all users by the SEND command. The SEND command causes a specified text string to be placed in the output buffer of a terminal or all terminals and, as a result, be printed on the terminal. If a user assigns a peripheral device for an inordinately long time, for example, the

*This feature is a special characteristic of the terminal that you designate as the system console (the default is KB0:). By installing a patch, the system manager can change the terminal having this characteristic. In any event, only one terminal on a system can have this capability.

system manager can transmit a message requesting the user to deassign the device. By specifying ALL in place of the device designator of a single keyboard, the system manager can broadcast the message to each on-line terminal in the RSTS/E system.

7.1.3.3 Controlling Jobs — The system manager can cause a user's terminal (or all users' terminals) to execute a text string by the FORCE command and he can also terminate a user's job by the KILL command. The FORCE command places a text string in the input buffer of a specified terminal as if it had been typed by the user. If the first character of more than 2 characters of the text is the up-arrow character (^), it is replaced by a CTRL/C (^C). The following sequence of two FORCE commands causes a user's terminal to execute two commands which log out the job.

```
#FORCE KB4: ^BYE
#FORCE KB4: YES
```

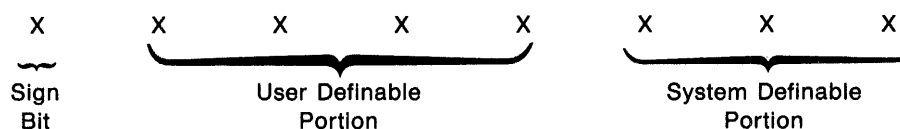
The KILL command terminates the user's job. The user is immediately logged off the system.

The running of a job is controlled by the SUSPEND, PRIORITY, and RESUME commands. The SUSPEND command gives a job a priority of -128. The job will not be eligible to run again unless the RESUME or the PRIORITY command raises its priority to -120 or higher.

7.1.3.4 Setting Job Priority, Run Burst, and Maximum Size — The UTILITY system program may be used to set the priority, run burst, and maximum size of an existing job. The system manager can change any of the current values to increase or decrease the chance of gaining run time in relation to other running jobs, to determine how much CPU time the job can have when it is compute bound and to increase the area a job can occupy.

The system runs jobs on the basis of priority. The higher a job's priority, the better are its chances of obtaining run time in relation to other running jobs. Priority is determined by an 8-bit priority byte, as shown in Figure 7-1.

Figure 7-1: Priority Byte Format



By running UTILITY, a privileged user can set the user definable portion of the priority byte for any job on the system using the PRIORITY command. Since the three system definable bits are normally zero, standard priorities are multiples of 8 between -120 (lowest priority) and +120 (highest priority). Zero is a legal priority. When UTILITY changes a priority, it truncates any value not a multiple of eight to the next lowest legal priority. For example, if you specify a priority of +10, UTILITY sets the value to +8.

All of the system definable bits are normally off (zero). The least significant bit is set when a keyboard delimiter is typed and the job was waiting for keyboard input. Keyboard delimiters are the CR, LF, FF, CTRL/Z, and ESC (or ALT) characters. The next significant bit is set whenever a CTRL/C is typed and can be set at any time. Finally, the most significant bit of the system definable portion is set by SYS system function call. All system definable bits are cleared when another program is chained or when the job is at the system command level.

The system definable portion of the priority byte is always less significant than the user definable portion. Therefore, the system definable bits affect priority only within the user definable priority range. If two jobs are running under priority -8, for example, the user who types a CTRL/C has a higher priority (that is, priority -6 in this case) than the user who does not type that combination. On the other hand, a third user with priority 0 supersedes two users whose priorities are -8 and -6.

When you log into a system, LOGIN is run with priority 0. LOGIN automatically sets your job to priority -8. This is the default priority with which most or all of the jobs are run. Only in unusual circumstances should priorities other than -8 be assigned.

On occasion, you may want to run a program that requires a great deal of computation. If time is not a factor in obtaining results, the privileged user can decrease the job priority to improve efficiency for the users on the system. Conversely, infrequently used detached programs often have higher priorities (typically priority 0) since they must run quickly when needed but do not run compute bound for an extended period and do not run often.

Run burst is the maximum time a job can run compute bound before another job obtains access to the CPU. On systems using the KW11C line frequency clock, each unit of run burst time is equal to 1/60th or 1/50th of a second, depending on the system's power line frequency. Systems running with the KW11P clock at crystal speeds, rather than at line frequency, have a run burst unit of 1/50th of a second. If the system is operating off a 60 Hz power line, one run burst unit equals 1/60th of a second. In that case, six units equal 1/10th of a second, which is the run burst default value.

If a specific job is assigned a run burst of 6 units but does not require that much compute bound time, the system automatically transfers control to the next user before the six units have been used. One tenth of a second is generally considered the best run burst time period to insure efficient overall system operation. If a job is guaranteed to become I/O bound (that is, I/O stalled) after a certain amount of computation, UTILITY can be used to specify a run burst larger than 6. In many cases, a run burst of greater than 6 units has a significant effect on long computational programs.

The maximum size assigned to a job refers to the allowable memory space. You can allow certain jobs to run programs larger than 16K words by assigning a job a maximum size up to 31K words. This assigned limit does not affect privileged, compiled programs. Thus, a user with a small amount of space can still run system programs that would normally exceed the memory limit.

7.1.3.5 Controlling Keyboards and Remote Lines — If the system manager determines that a dataset line is in use but no keyboard activity is taking place (by SYSTAT or display program job status report), he can disconnect the dataset. The HANGUP command causes the remote line specified by KBn: to be disconnected. The hangup capability prevents a user from monopolizing the line without being charged for connect time and frees the line for other remote users.

By the DETACH command the system manager can free a terminal from its job. If the job does not have the terminal open on a nonzero channel, this action frees the terminal for other use. The DETACH command normally forces the system to close all nonzero channels on which the terminal is open. The /NOCLOSE switch suppresses this action. When the job specified is currently detached, UTILITY prints the message JOB NUMBER x IS ALREADY DETACHED.

The DISABLE KBn: command can remove a single line terminal interface from use for the current time sharing session. To disable an interface for subsequent sessions, the SET option of the initialization code can be used. (SET is described in the *RSTS/E System Generation Manual*.) An interface remains disabled until the start of the next time-sharing session, at which time it is reenabled unless it was disabled by the SET option.

7.1.3.6 Changing System Date and Time — The DATE and TIME commands allow the system manager to change the system date and the 24-hour clock. The commands are important on systems where strict accounting data is kept and incorrect values may have been entered at system start-up time.

7.1.4 Data Caching Control

Data caching is used to store blocks from a user file for direct memory access. Directory caching is used to store the MFD, UFD, Monitor overlay code, and other frequently accessed system file directories.

When a user job executes a read request, the RSTS/E Monitor performs a disk access and transfers the requested block(s) of data from the disk to the user job's buffer. With data caching, the monitor stores the most recently read data blocks in an area of memory called the cache. If a user job executes a read request for a data block in the cache, the monitor copies the requested data directly from the cache into the job's buffer. Because the system can retrieve data blocks from memory instead of disk, physical disk accesses are decreased. This can result in improved I/O throughput and faster response time.

Data caching is most useful for read operations in that it can minimize disk transfers. Every write operation causes an actual write to the disk. In a write operation that modifies existing cached data, the data is updated both in the cache and on the disk.

Support for data caching must be selected during system generation (see the *RSTS/E V7.0 System Generation Manual*). You can generate three types of caching: no caching, directory caching only, or data and directory caching. If

you select directory caching, caching is automatically enabled at system start up. If you select data caching, caching is automatically enabled provided that at least 2K words of XBUF are allocated for data caching. Note that data caching cannot be selected without also selecting directory caching.

Data caching can also be enabled by means of the UTILTY command ENABLE CACHE as described in Section 7.1.4.6. You also use UTILTY commands to specify caching for a particular file by marking the file's UFD entry.

Under the BASIC-PLUS or MACRO programming languages, you can use MODE values in the OPEN statement or Monitor directives to open a file for cached data access. Refer to the *RSTS/E Programming Manual* for information on BASIC-PLUS OPEN MODEs and to the *RSTS/E System Directives Manual* for MACRO directives. However, the use of caching MODE values and Monitor directives is privileged. The system manager can use the UTILTY program to designate a file for caching by marking its UFD entry. Once a file's UFD entry is marked for caching, it will be cached on OPEN regardless of the user's privilege as long as caching is enabled on the system.

When caching is generated and enabled, the cache receives all data transfer requests that are otherwise directed to the disk driver. Read operations on data that is in the cache occur without placing a load on the disk driver. The monitor constantly updates the cache such that it contains the most recently requested data for cached files.

7.1.4.1 Size of the Cache — The RSTS/E Monitor allocates space for the cache from the Extended Buffer Pool (XBUF). You specify the amount of this allocation with UTILTY command options that set the size of a cache cluster (/CL:n) and the number of clusters in the cache (/LIMIT:n, /DIR:n, and /DATA:n). These command options are described in Section 7.1.4.6. The size of a cache cluster (1, 2, 4, or 8 blocks) determines the amount of data that is treated as a unit in a read request and, in many cases, the number of read requests that can be resolved in the cache before access to the disk driver is required.

For example, when the cache cluster size is 8 blocks, any read operation that installs data in the cache causes 8 physically contiguous blocks (including the requested blocks) to be installed. Note that the cache cluster size should usually be equal to or less than the pack cluster size (set during disk initialization) of the disk with the most files to be cached, to ensure that only requested data is read into the cache.

7.1.4.2 Sequential and Random Caching Modes — The data in a file can be cached in one of two modes, sequential or random. You can specify either of these modes for a particular file with UTILTY command options as described in Section 7.1.4.8.

If a job executes a read operation on a cached file and the data is not in the cache, that data is installed in the cache if there is free space or if a current cache cluster is eligible for replacement.

A cluster is eligible for replacement if:

1. the last block of the cluster has been read in sequential mode, or
2. the cluster has been in the cache without being read for more than the minimum residency time. An optional patch is available which allows the system manager to change the minimum residency (see the *RSTS/E V7.0 Release Notes*).

7.1.4.3 Random Mode Caching — Caching a file in random mode is recommended if the file is accessed by more than one user or if it is index structured.

As an example of random cache operation, consider a read operation executed on a file whose UFD entry is marked for random caching. When a read on the cached file occurs, the monitor examines the content of the cache to determine if the requested data is present. If the data is in the cache, the data is copied from the cache cluster that contains it. The data is made available to the program and the cache cluster's time of last access is updated.

If the requested data is not in the cache, the monitor examines the list of cache clusters to find one that is eligible for replacement. If one is not found, the requested data cannot be installed in the cache and a normal disk read is automatically performed. If the cache cluster is eligible for replacement, the requested data is installed in that cache cluster and made available to the program.

7.1.4.4 Sequential Mode Caching — Caching a file in sequential mode is recommended if the file is accessed by a single user, more than one user in sequence, or if the file's data is accessed in sequence.

As an example of sequential cache operation, consider a read operation executed on a file whose UFD entry is marked for sequential caching. When a read on the cached file occurs, the monitor examines the content of the cache to determine if the requested data is present. If the data is in the cache, the data is copied from the cache cluster that contains it. The data is made available to the program. If, in the process, the last block of a cache cluster was read, that cluster is made available for replacement in the cache.

Furthermore, when a read operation is performed on any block of a cache cluster (except the last block), the monitor installs a full cluster of data in the cache. That is, if you use the UTILTY command option /CL:n to set a cache cluster size of 8 blocks and then execute a read on the first block of a file whose cluster size is 8 or greater, requested data plus the next 7 blocks from the disk are read into the cache. Thus, the contents of the next 7 blocks can be satisfied from the cache. When the last block of the cache cluster is read, the cluster is immediately made available for the installation of new data as requested by other read operations.

If a read operation from a sequentially cached file causes more than one cache cluster to be read, all of the requested data blocks are made available to the program. However, no data is installed in the cache for any cluster whose last block was read (i.e., only the last cluster can be installed and only if it was

incompletely read). Thus, if the cache cluster size is defined as 1 block and sequential mode is specified, no data blocks will be installed in the cache (i.e., every data block is the last block in a cache cluster).

7.1.4.5 LIST CACHE Command — The LIST CACHE command causes UTILTY to print at your terminal the current caching parameters for your system. The listing includes information on cluster sizes, data caching, and the use of small buffers. Consider the following example:

```
RUN $UTILITY
UTILITY V7.0 RSTS V7.0 Time sharing
#LIST CACHE

Caching enabled
Data caching available

Current settings:
  Cluster size :4
  Data caching :By file
  Small buffers :No

Cluster      All      Dir      Data
Limits :    65535    65535    65535
#
```

In this example, LIST CACHE reports the following information:

1. directory and data caching were installed on your system during system generation and caching was enabled.
2. the current cache cluster size is 4 blocks, only selected files are cached, and small buffers are not used.
3. no limit is set on the use of XBUF for the cache.

7.1.4.6 ENABLE CACHE Command and Options — The ENABLE CACHE command and options allow you to set caching parameters for the system as a whole. The ENABLE CACHE command is used to enable data and directory caching on the system. Note that caching is normally enabled automatically at system start up.

When ENABLE CACHE is specified with no options, the last settings specified are applied. The initial system defaults are as follows:

```
/FILE/LIMIT:65535/DIR:65535/DATA:65535/CL:n
```

where n was specified in the DEFAULT option or was allowed to default to 4 blocks during system initialization.

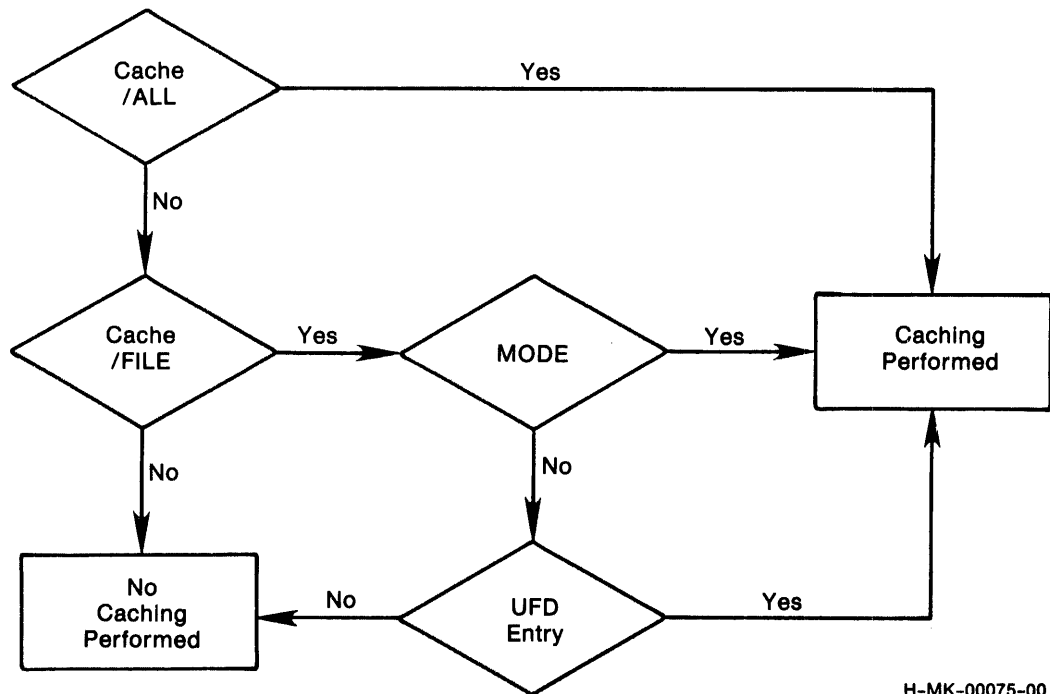
If ENABLE CACHE/ALL is used, all read requests are cached. The type of caching (sequential or random) for a particular file is determined by the file's UFD entry as specified in the FLAG command or by OPEN MODE specification.

If ENABLE CACHE/FILE is used, a file is cached on the basis of its UFD entry and/or the specified OPEN MODE. That is, a file is cached if either the UFD entry or OPEN MODE specifies caching; it is cached sequentially if either the UFD entry or OPEN MODE specifies sequential.

If ENABLE CACHE/NOFILE is used, files are not cached. This option is used to enable directory caching but disable data caching on the system.

Figure 7-2 illustrates the mechanism whereby the monitor decides the amount of caching to be performed. The Figure shows the flow of UTILITY commands and caching specifications examined by the monitor.

Figure 7-2: Monitor Caching Checks



The /BUFF and /NOBUFF options are used to specify the use of small buffers for directory caching. These options are meaningful only if directory caching was selected during system generation but data caching was not selected. If you specify an ENABLE CACHE/BUFF command on a system that has data caching installed, a ?MISSING SPECIAL FEATURE error is returned. If data caching is installed, small buffers cannot be used for caching.

The ENABLE CACHE/CL:n command is used to specify the cache cluster size as 1, 2, 4, or 8 blocks for each cluster. The cache cluster size controls the number of contiguous data blocks that are copied from disk to the cache whenever file data is cached. The cache cluster size should be small enough to contain only the target blocks, but large enough to reduce the number of disk accesses. That is, you must anticipate data requests and ensure that the cache cluster size is equal to the file cluster size of the most often accessed files.

The **ENABLE CACHE/LIMIT:n** command (where n is in the range of 0 to 65535) is used to specify the total number of cache clusters used by both directory and data caching. The default limit is 65535, which means that cache cluster allocation is limited only by the amount of XBUF available. The **ENABLE CACHE/DIR:n** and **ENABLE CACHE/DATA:n** commands (where n is in the range of 0 to 65535) specify the cache cluster allocation for directory and data caching, respectively. The specified allocation is an upper limit. Thus, if a 40K word XBUF were defined at system generation and /DIR and /DATA both specified a number of cache clusters equivalent to 25K, data can use space in the cache up to a maximum of 25K, which leaves a 15K minimum for directory caching. The reverse is also true. In this manner, data and directory caching are guaranteed a minimum allocation and the amount of overlap is controlled, which permits the cache to dynamically adjust to system and program requirements.

The default settings for /CL, /LIMIT, /DIR, and /DATA are: a cache cluster size of 4 blocks (unless altered at system generation), and no limit on directory, data, or total cache allocation.

7.1.4.7 DISABLE CACHE Command — The **DISABLE CACHE** command causes UTILTY to disable all data and directory caching on the system. If you reenables caching following a disable and do not specify caching parameters (/CL, /LIMIT, /DIR, and /DATA), the default parameters are those that were in force prior to the disable.

7.1.4.8 FLAG Command and Options — The **FLAG** Command allows you to specify caching parameters for a specific file. The format of the **FLAG** command is as follows:

```
FLAG <filename>/option[/option]
```

where;

filename is a RSTS/E file specification.

/option is /CACHE, /NOCACHE, /SEQ, or /RAN. At least one of these options must follow the **FLAG** command.

The /CACHE option specifies that the file will automatically be cached when open. The type of caching used on the file's data depends on the file's UFD entry and the specified OPEN MODE. If /CACHE is used alone on a file with no UFD setting or MODE specification, the default is random caching.

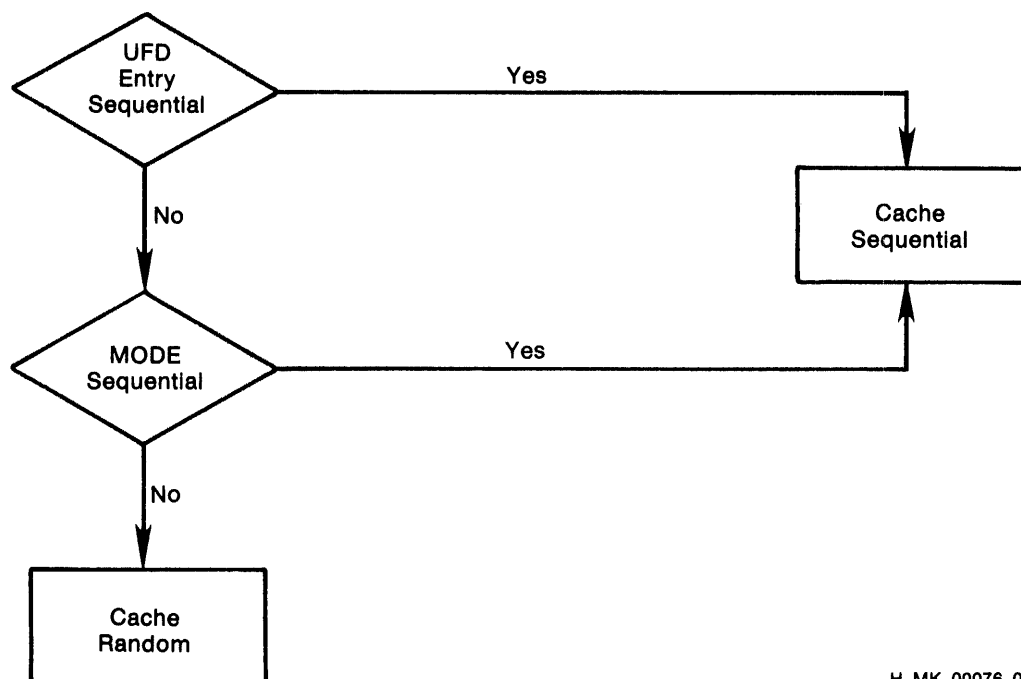
The /NOCACHE option specifies that the file will not automatically be cached when open. To cache such a file, you must specify **ENABLE CACHE/ALL** (see Section 7.1.4.6) or specify a MODE value (see the *RSTS/E Programming Manual*).

The /SEQ option causes UTILTY to mark the file's UFD entry such that if the file is cached, it will be cached sequentially.

The /RAN option causes UTILTY to mark the file's UFD entry such that if the file is cached, it will be cached randomly. Note that you can override the random cache UFD setting with a sequential caching MODE specification (i.e., a sequential caching specification with UTILTY or MODE always overrides random caching). Note also that a file which has never been flagged is equivalent to /NOCACHE/RAN.

Figure 7-3 illustrates the flow of UFD entry and MODE value checks performed by the monitor. The result of these checks determines the type of caching to be used on the file.

Figure 7-3: Caching Mode Checks



H-MK-00076-00

7.1.4.9 Caching Guidelines — The relationship between cache cluster size, which you set with a /CL:n option, and the pack cluster size of the file can have an impact on the efficiency of caching. The cache cluster size determines the number of blocks on disk that are copied into the cache when a cluster is installed. Thus, if the cache and pack cluster sizes are equal, there is a direct correspondence between the data that will probably be read and the data copied from the disk to the cache. If the cache cluster size is larger than the pack cluster size, the data you request plus some possibly unrelated data is copied to the cache, thereby wasting cache clusters. If the cache cluster size is less than the pack cluster size, a read request may require multiple disk accesses to move all of the data into several different cache clusters.

The relationship between the cache cluster size and RMS indexed file bucket size is similar. That is, because RMS always reads an entire bucket at once, efficient caching is obtained when the cache cluster and bucket sizes are equal.

Consider the following guidelines:

1. The amount of XBUF allocated to data and directory caching while setting defaults should be as large as possible while maintaining sufficient memory for user jobs, resident libraries, and run-time systems. However, an exceptionally large allocation can increase swapping activity, especially on systems with less than 128K words of memory.
2. On systems which use RMS heavily, consider the use of the RMS resident library and the library's impact on XBUF allocation. Typically, on systems with less than 128K words of memory, the resident library should receive priority in memory allocation, to the extent of limiting XBUF allocation. In such a case, data caching may become ineffective and probably should be disabled.
3. To optimize directory caching, all UFD cluster sizes should be set to 16 blocks if possible.
4. Files which are frequently accessed should have their UFD entries marked for caching with the UTILTY command FLAG.
5. Files that are frequently accessed sequentially (including RMS sequential and relative files) should have their UFD entries marked for sequential caching with the UTILTY command FLAG/SEQ/CACHE.
6. Files which are frequently accessed randomly (including RMS relative and indexed files) should have their UFD entries marked for random caching with the UTILTY command FLAG/RAN/CACHE. Because of the manner in which RMS indexed files are maintained, indexed files should always be flagged as random even if they are accessed sequentially. Also, where possible, RMS indexed file bucket sizes should be a power of 2 to correspond with caching cluster sizes.
7. Use data caching judiciously on RMS indexed files. You gain the most benefit from caching an indexed file when it is subject to many read requests and is open for long periods of time. Also, you should allocate more XBUF to such files to compensate for the lower caching "hit" rate on indexed file data. That is, the "hit" rate is generally lower on indexed files than it is on sequential files or directory blocks.
8. Where possible, all files should be contiguous to reduce window turning.

Caching is not the most efficient data access method for all files under all circumstances. But, in general, a high-access file would be a logical candidate for caching. A high-access file is a file which meets one or more of the following conditions:

1. accessed by more than one user at a time.
2. opened for long periods of time.
3. frequently accessed.

4. accessed sequentially (indexed files).
5. primarily read.

However, because of the large number of different applications, some experimentation is needed to arrive at the best use of caching on your system.

7.1.5 Run-Time System Control

A run-time system (RTS) on RSTS/E is the common, sharable part of a user job and, in the case of BASIC-PLUS, is the interface between the user's executable code and the monitor. Run-time systems control execution of jobs when the monitor allows the jobs to run. The monitor is responsible for setting up jobs and establishing an environment within which a run-time system can function. The BASIC-PLUS run-time system itself is responsible for interpreting requests made by a user job, translating those requests into a format the monitor understands, and transmitting the requests of the monitor. The monitor processes the requests and passes either data or an error to the job. The run-time system decides whether to interpret the data or log an error.

Other run-time systems are not language interfaces, so that jobs under their control can make direct requests of the monitor. In such cases, the major role of the run-time system is to set up compiled programs for running, handling system errors, and returning control, upon program termination, to the job's default run-time system. Some of these run-time systems can also emulate the action of certain system directives from other operating systems, such as RT11 or RSX.

A run-time system such as BASIC-PLUS interprets user requests when it is a language interface for a user job. A computer language provides a standard means by which many users can define data, process it, and obtain results. The language relieves each user of the need to learn the workings of the monitor. The run-time system under which the language operates is more efficient because multiple users can share the same code. This sharability is more efficient for the system because each user need not have his own copy of the common code.

All RSTS/E installations have a system default run-time system and can optionally have auxiliary run-time systems. The system default run-time system, BASIC-PLUS at most RSTS/E sites, is responsible for allowing users access to system resources, interpreting system command requests, and performing housekeeping chores. An auxiliary run-time system may provide any other shared computer processing function and usually enables processing in a language other than BASIC-PLUS. Such an auxiliary run-time system may interpret and/or execute BASIC-PLUS-2, COBOL, FORTRAN-IV, or other language operations.

All run-time systems are stored as contiguous files with .RTS extensions in the system account [0,1]. The default run-time system must be stored on the system disk because, at the start of time sharing, the system disk is the only disk mounted on the system. Auxiliary run-time systems can be stored on any disk, either public or private. All run-time system files are contiguous because they must be loaded into memory in the fastest time possible.

The system default run-time system is created during system generation. Other run-time systems, each having a distinct name, may exist simultaneously on a system disk. The system manager, through the DEFAULT initialization option, selects one of them as the system default run-time system which is loaded into memory automatically when time sharing starts.

The DEFAULT initialization option creates the monitor parameters which describe the default RTS: the name of its file, its location on the disk, and its characteristics. These parameters are fixed for a given time-sharing session and cannot be altered during a session. Through the DEFAULT option, the system manager can optionally locate the default RTS in memory. This location is also unalterable during a given time-sharing session.

Auxiliary run-time systems are created during system generation or afterwards. The monitor structures which describe these run-time systems do not automatically exist at the start of time sharing. Therefore, the system manager must provide the commands to add those structures for auxiliary run-time systems which he wishes to make available for any given time-sharing session. The structure for an auxiliary RTS is dynamic and can be removed during a time sharing session.

The UTILITY run-time system commands allow a system manager to control operations of auxiliary run-time systems. The RTS commands execute SYS system function calls to effect the particular operations. The UTILITY commands, however, perform additional processing for certain commands. To assist the system programmer, the command descriptions distinguish between the monitor call action and the UTILITY command action. Because run-time systems under development need certain special controls, some features of run-time system control are intended solely for DIGITAL personnel. Nevertheless, all features are described to give the system manager a broader perspective.

It is recommended that the system manager make auxiliary run-time systems available by commands in the START.CTL file. By reading the descriptions in this section the system manager will understand what actions should occur for each run-time system command. The descriptions in Section 3.1 tell the system manager how to organize the run-time system commands so that the desired actions are performed automatically at the start of each time sharing session.

7.1.5.1 Adding and Removing Auxiliary Run-Time Systems — Because the structures which define auxiliary run-time systems are transient, the system manager must create them for each time-sharing session. The ADD command creates the necessary structure to enable an auxiliary run-time system to function. Switches on the ADD command can alter predefined characteristics of the RTS.

The ADD command creates a run-time system description block with a small buffer. A file with the name of the run-time system and an extension of .RTS must exist in the system account [0,1] of the disk specified in the command. A run-time system with the same name can not exist. The system ensures that the file found is contiguous, has the proper format, and has proper parameters.

The system manager can specify switches to override predefined characteristics. The ADD command extracts information from the RTS file to establish the proper entries in the RTS description block. To establish the entries, UTILTY opens the file, reads it, and sets up the predefined characteristics to be placed in the description block. The predefined characteristics are changed by the switches /EMT:n, /REMOVE, /LOGERR, /RW, /1USER, /KBM or the negation of the switches (/NOEMT, /NOREMOVE, /NOLOGERR, /NORW, /NO1USER, and /NOKBM). For example, if a run-time system which has no special command decoding capabilities were being tested, the system manager would use the switches /1USER, /RW, /NOLOGERR, and /REMOVE to add the RTS. The /1USER ensures that no other job will try to share the untested code; /RW is necessary for setting breakpoints with ODT; /NOLOGERR prevents spurious errors generated by the run-time system from cluttering the system error log; and /REMOVE ensures that the run-time system's image will not be left in memory but rather will be reloaded from disk immediately preceding every entry to it. The switches and negations of the switches do not alter the predefined characteristics in the RTS file; they alter only the characteristics defined in the description block. The characteristics remain in effect until the RTS is removed.

The /ADDR:n switch allows the system manager to take advantage of high speed memory on the system or to avoid fragmentation of memory. Without the /ADDR:n switch, the monitor decides where to load the RTS each time residency is required. With the switch, the monitor loads the RTS at the specific 1K-word section of memory. The /ADDR:n switch is required whenever the /RW switch is specified.

Because 1K section numbering begins at 0 and ends at n-1 (where n is the total size of memory), the 1K section number in the /ADDR:n switch is one less than the physical section number. For example, to load the RT11 Run-Time System (4K words in size) into the 61st through 64th 1K sections of memory, specify /ADDR:60. The run-time system is loaded from low memory to high memory at its defined initialized size. To be loaded without error, enough contiguous user space must be available starting at that location. The location specified in the /ADDR:n switch becomes the default location at which the run-time system is loaded during the current time-sharing session. The location need be changed only if the system manager changes the allocation of the section of memory with either the DEFAULT or the START initialization option.

One precaution is necessary when specifying the address at which the run-time system is loaded. The section of memory chosen must not fragment the user job space to prevent the run-time system from executing a job. For example, assume a system has 24K words of user space available between the 36K and 60K sections of memory. Assume also that a job requires 18K words of user space to run and that the run-time system requires 4K words when resident. If the loading address is 36K, the space between 40K and 60K remains available for an 18K job to run. If the loading address is 42K, the user space is fragmented into two sections — one from 36K to 42K and one from 46K to 60K. An 18K-word job area is not available to execute a job using this auxiliary RTS.

The system verifies that the memory section given in the /ADDR:n switch is reasonable. If the entire range of memory starting at the load address is not available, the ?ILLEGAL BYTE COUNT FOR I/O error is returned. The system manager should consult the memory status report of a display program to select an available range of memory. If the range of memory results in fragmenting that would cause a swapping violation, the system returns the ?NO ROOM FOR USER ON DEVICE error. A swapping violation would occur if the memory to be occupied by the RTS would not allow enough contiguous space for a maximum sized job to run. To avoid excessive disk activity on small systems, the system manager should include the ADDR:n switch when adding auxiliary run-time systems.

Errors that can be encountered when a run-time system is being added are summarized in Table 7-3.

The /POSITION:n switch denotes the position in the linked list of blocks in which the RTS block is to be placed. The default RTS block is always first in the list. If the RTS being added is to be accessed frequently, the system manager can reduce system overhead by placing it appropriately in the list. Without the switch, the system adds the block to the end of the list. If n in the /POSITION:n switch is 1, the block is placed immediately after that of the system default RTS. If n is 0 or a value greater than the number of currently defined run-time systems, the block is simply added to the end of the list.

Table 7-3: Run-Time System ADD Command Errors

Text and Meaning
<p>?CAN'T FIND FILE OR ACCOUNT A file with the name specified and with an extension of .RTS cannot be found in account [0,1] on the device given.</p>
<p>?ILLEGAL BYTE COUNT FOR I/O The range of memory starting at the load address given is not available. Refer to the memory status report of a display program to select an available range of memory.</p>
<p>?NAME OR ACCOUNT NOW EXISTS A run-time system with the same name currently exists.</p>
<p>?NO BUFFER SPACE AVAILABLE Adding a run-time system description block requires a small buffer and one is not currently available.</p>
<p>?NO ROOM FOR USER ON DEVICE If the monitor were to load this run-time system at the address specified, memory would be fragmented and a swapping violation would occur. Refer to the discussion of assigning and allocating memory in the <i>RSTS/E System Generation Manual</i> for guidelines on how to avoid fragmenting memory.</p>
<p>?PROTECTION VIOLATION The file to be added as the run-time system has a bad format. For example, the file is not contiguous or has illegal entries in the SIL index.</p>

The position of a run-time system description block in the linked list affects how the system treats a RUN request for a file without an extension. Upon receiving such a RUN request, the system checks the indicated structure for all files with the specified name and a executable extension. For example, if the structure contained three files of the same name and with extensions .BAC, .TSK, and .SAV, the system would check for the run-time system nearest the default run-time system in the linked list of description blocks. If BASIC-PLUS were the system default RTS (whose block is always at the root), RSTS/E runs the .BAC version of the file.

The /EXT:ext switch on the ADD command changes the default extension used for a runnable file. If the switch is not given, UTILITY extracts the extension defined in the RTS file. This default extension is applied when a user types a RUN command for a program and does not specify an extension with the file to be run.

The /MIN:n and /MAX:n switches alter the minimum and maximum job sizes (in K words) defined in the file for the RTS. Generally the minimum value allowed is 1 and the maximum is 31, but a given RTS may not allow this full range of job sizes. For BASIC-PLUS, the minimum and maximum sizes are 2 and 16 respectively.

The /STAY switch makes the RTS permanently resident. Usually, an auxiliary run-time system is temporarily resident. It occupies memory as long as the currently active job is running under its control. The system frees up the memory it occupies when that memory is needed to load another run-time system or to load a job running under another run-time system. (Thus, the RTS may be nonresident if there are active jobs using it.) The system automatically loads the run-time system when a user requires its services. If the /STAY switch is specified, the auxiliary RTS becomes permanently resident as the default RTS is. The only way to make such a RTS nonresident is to use the UTILITY command UNLOAD.

The REMOVE command essentially reverses the steps ADD performs. All auxiliary run-time systems must be removed before the system can be shut down. The SHUTUP program automatically does this removal. The system checks to ensure that no jobs are currently running under control of the RTS to be removed. If the usage count is 0, the description block is removed from the list and the run-time system file in account [0,1] is closed.

7.1.5.2 Loading and Unloading a Run-Time System — Loading and unloading a run-time system are accomplished by the LOAD and UNLOAD commands. The commands perform explicitly the same actions the system performs automatically. The system manager has control over placing the run-time system in memory. This control is normally useful in cases when a run-time system resides in high speed semiconductor memory and the system manager wishes to specify its position.

The /ADDR:n switch on the LOAD command specifies a loading address similarly to the same switch on the ADD command. On the LOAD command, however, the effect of the positioning depends on what was specified in the ADD command. Having specified a permanent load address in the ADD command, the system manager indicates a special need to have the RTS loaded at

a fixed location. With the /ADDR:n switch on the LOAD command, the system changes that special location to the new address. Without the /ADDR:n switch on the LOAD command, the system uses the location defined in the ADD command.

If the system manager did not specify an address in the ADD command, the system assumes there is no need for a special address. An address specified in the LOAD command applies, therefore, only for the current residency. If the monitor must load the run-time system later, it reverts to deciding the load address.

The /STAY switch on the LOAD command makes the RTS permanently resident unless the RTS is currently running a job. If this is the case, a LOAD/STAY is equivalent to a LOAD with no /STAY switch. The /STAY switch is helpful in cases where a special load address was not defined in the ADD command. The /STAY and /ADDR:n switches in the LOAD command ensure that the RTS resides in the desired memory space.

The UNLOAD command frees the memory occupied by the Run-Time System. The system returns the ?ACCOUNT OR DEVICE IN USE error if a job is currently resident or being loaded and intends to use the RTS.

7.1.5.3 Associating a File with a Run-Time System — The NAME command changes the name of the run-time system associated with a file. Every disk file on RSTS/E has stored, in its directory, the name of the RTS under which it was created, except files that are larger than 65,535 blocks. On a RUN request for the file, RSTS/E checks the name to find out what run-time system to use for the job. RSTS/E automatically switches control to that RTS which, in turn, executes the file.

To change the name, UTILITY attempts to open the file to establish write access. If write access is denied, the ?PROTECTION VIOLATION error is generated. The name of the RTS is written in the directory of the file specified. This naming operation is done during the system library build procedures when a file designed to run under an auxiliary RTS is transferred from the distribution medium using PIP under BASIC-PLUS.

7.1.6 Resident Library Control

A resident library is a collection of shareable routines or data that the Task Builder links together into a task image file on disk. The MAKSIL program (see the *RSTS/E Programmer's Utilities Manual*) is used to format this disk file into Save Image Library (SIL) format. The UTILITY system program ADD LIBRARY command is then used to assign the task image portion of the SIL file to a contiguous region of physical memory. Note that you can also use Monitor SYS calls (-18) to assign the task image portion of the SIL file to memory. Once the body of shareable routines or data is linked, formatted, and assigned to memory, it becomes a resident library that is accessible to user tasks as part of their virtual address space.

To add a resident library to the Monitor's list, you must be a privileged user. There are two methods that you can use to add a resident library: Monitor

SYS calls to FIP and UTILTY system program commands. The UTILTY commands are described in this section and the Monitor SYS calls are discussed in the *RSTS/E Programming Manual*.

7.1.6.1 Resident Library UTILTY Commands — You can use the UTILTY program to ADD, REMOVE, LOAD, or UNLOAD a resident library. To distinguish between resident library operations and run-time system operations which use the same command names, UTILTY requires the keyword LIBRARY in its syntax for library operations. You include the keyword LIBRARY in the UTILTY command line between the operation keyword and the target resident library name.

The UTILTY command to add a resident library is as follows:

```
ADD LIBRARY name/ADDR:n</option>
```

The name that you include in the ADD LIBRARY command identifies the resident library. To identify it more specifically you can specify a device, a PPN as well as a protection code in addition to the library name. If you do not specify these variables the device defaults to SY:, the PPN becomes account [0,1], and the library's protection code defaults to <42>. When you add a resident library, UTILTY requires that you use the /ADDR:n switch to locate the library at a specific point (n) in memory. The n variable can be any number from the lowest to the highest available memory locations in increments of 1K word sections. The optional switches </option> that you attach to the ADD LIBRARY command provide the following features.

1. The /STAY switch makes the resident library permanently resident.
2. The /1USER switch allows only one user access to the resident library code.
3. The /RW switch maps the library Read/Write rather than Read-Only and is necessary for setting breakpoints with ODT.
4. The /NOLOGERR prevents spurious errors generated by the resident library from entering the system error log.
5. The /REMOVE switch ensures that the resident library will be removed from memory immediately after the last user has detached from the library.

The /STAY switch allows you to make a resident library permanently resident. Usually, a resident library only temporarily remains in memory. That is, it occupies memory only as long as currently active jobs run under its control. However, when you need to access the library again, the system automatically reloads it. To force the library to remain permanently in memory you must specify the /STAY switch in the ADD LIBRARY command.

If you want to remove a resident library from memory, delete the monitor structure that defines the library, and close the library file. Use the UTILTY command option that follows:

```
REMOVE LIBRARY name
```

Note that the REMOVE option makes the resident library inaccessible for sharing whereas the /REMOVE switch, which can only be used with the ADD LIBRARY command, only temporarily removes the library from memory while it is not in use.

7.1.6.2 Loading and Unloading a Resident Library — The LOAD and UNLOAD options perform the same functions that the system performs automatically. You can use the LOAD option along with the /ADDR:n switch to place it where it can most benefit the system. Loading (LOAD) a resident library into a particular location in memory or unloading (UNLOAD) it to free memory for other system functions may be beneficial. Usually, the option to locate a resident library becomes useful in situations when you want to position the library in high speed memory. Format the LOAD and UNLOAD commands as follows:

```
LOAD LIBRARY name</option>
UNLOAD LIBRARY name
```

UTILITY permits you to specify two switches with the LOAD LIBRARY option, /ADDR:n and /STAY. If you do not attach the /ADDR:n switch, UTILITY places the resident library in the location you specified in the ADD LIBRARY name/ADDR:n command. When you include an address with the LOAD LIBRARY name/ADDR:n command line, you override any location previously set by ADD. Attach the /STAY switch to the LOAD command as follows:

```
LOAD LIBRARY name/ADDR:n/STAY
```

if you want to ensure that the resident library remains in the desired memory space. Note that the /ADDR:n switch forces the system to use the LOAD specified address.

The UNLOAD command removes a resident library from memory and thus frees memory which can then be used to load another library or be available for other system purposes. Note that, the system returns the ?ACCOUNT OR DEVICE IN USE error message if you attempt to unload a resident library that is in the process of being loaded or is in use by the currently running job.

7.1.7 System File Control

By using the REFRESH initialization option, the system manager allocates and positions the system files SWAP0.SYS, SWAP1.SYS, SWAP3.SYS, OVR.SYS, and ERR.SYS on file-structured disks. These optional files optimize system performance by taking advantage of higher speed disks and the characteristics of user jobs running on the system. Because these files are contiguous and contiguous space becomes scarce as user files are created during time sharing, preallocation and prepositioning are necessary. The *RSTS/E System Generation Manual* discusses the planning considerations for preallocating and positioning these system files.

RSTS/E, however, does not automatically access these system files. The system manager, by UTILITY commands, controls RSTS/E access to these files. Thus, the use of these files during time sharing is dynamic. If a hardware problem occurs on a swapping device, the system manager can remove the device without having to stop time sharing. To maintain the same amount of swapping space, a private disk can be designated as the swapping device.

This section discusses the UTILITY commands provided to control system file usage. It is recommended that the system manager implement some of these controls by INIT commands described in Section 3.1. Consequently, access to system files is accomplished at the start of time sharing operations by commands in the INIT.BAC control files. The UTILITY commands are used to alter system file usage established at system start up.

7.1.7.1 Adding and Removing Swapping Files — The swapping files SWAP0.SYS, SWAP1.SYS and SWAP3.SYS in account [0,1] are standardized names to denote the three optional swapping slots — 0,1, and 3. These names are optional but highly recommended because they easily tell the characteristics of the files. The system manager, however, can assign any file name to designate these three swapping slots. The system restricts the extension to .SYS regardless of the file specification used in the UTILITY commands.

The ADD SWAPFILE command tells RSTS/E to access a specific file (if a file structured device is involved) or a specific device (if a nonfile structured disk is involved). A typical command sequence would be as follows:

```
#ADD SWAPFILE 0 DS0:
#ADD SWAPFILE 1 DS1:
#ADD SWAPFILE 3 DB1:SWAP3
#
```

The lowest numbered swapping slots are added (or installed) on the fastest devices, units 0 and 1 of the RS03 or RS04 disk. The highest numbered slot is added on the slowest device, RP04, RP05, or RP06 unit 1. Note that, when a non-file structured device is specified in the ADD SWAPFILE command, any existing file structure on that device is destroyed.

The /SI:n option is provided with the ADD SWAPFILE command for on-line creation of a swapping file during time sharing. It is best used on a private disk with sufficient contiguous space to allow a swapping file to be created. The value n in the /SI:n option designates the number of blocks at which the file is to be created. UTILITY attempts to access the file with an extension of .SYS in account [0,1] on the associated device. If the file exists at a size differing from the size specified in the /SI:n option, UTILITY prints a warning message in the following format:

NOTE — THE SIZE OF file.SYS IS x NOT n.

The value x is the actual size; n is the value given in the /SI:n option. If the file does not exist, the program attempts to create a contiguous file at the size specified. The ?NO ROOM FOR USER ON DEVICE error means that sufficient contiguous space is not available to create the file.

After the ADD SWAPFILE command is typed, the system attempts to install the particular file or device as the swapping slot. If any errors are encountered, UTILTY prints an error message in the following format:

text — in ADD SWAPFILE

Table 7-4 lists the possible errors.

Table 7-4: ADD SWAPFILE Command Errors

Text and Meaning
<p>?ACCOUNT OR DEVICE IN USE A non-file structured disk is being added as a swapping file but the disk is currently mounted (that is, it is being used as a file-structured device).</p>
<p>?CAN'T FIND FILE OR ACCOUNT A file with the name specified and with a .SYS extension does not exist in account [0,1] on the associated device.</p>
<p>?DEVICE NOT AVAILABLE A non-file structured disk is being added as a swapping file but either the disk unit or its controller has been disabled. The system manager must use an initialization option to enable the unit or its controller.</p>
<p>?DEVICE NOT FILE STRUCTURED The device specified is not disk.</p>
<p>?DISK PACK IS NOT MOUNTED A file-structured disk is specified but that disk is not currently mounted. Use the MOUNT command to logically mount the disk before adding the file.</p>
<p>?ILLEGAL FILENAME The file name given in the command contains characters other than alphabetic or numeric characters.</p>
<p>?ILLEGAL VALUE - n The swapping slot number specified is other than 0,1, or 3. (The swapping file for slot 2 must exist on the system disk and need not be added.)</p>
<p>?MISSING DEVICE OR FILE NAME The command must contain either a device specification or a device and file name specification. If the /SI:n option is given, a file name must be present.</p>
<p>?NO ROOM FOR USER ON DEVICE A file (on a file structured device) is being added as a swapping file, but the file is not large enough to store even one job at the current SWAP MAX.</p>
<p>?NOT A VALID DEVICE The device specified is a disk but is not configured on this system.</p>
<p>?PROTECTION VIOLATION A file is being added as a swapping slot. Either the disk is mounted read only or the file is bad (is not contiguous or is currently open). The system must have write access to the device and to the file to successfully add the file.</p>

If no errors occur, the system has installed the swapping file.

The system manager can then activate the added swapping slot by issuing the LOGINS command. The LOGINS command forces the system to examine all swapping file space and to calculate the number of jobs the system is capable of running.

During time sharing the number of jobs capable of running on the system depends on three factors:

1. the job maximum (JOB MAX) set at the start of time sharing,
2. the amount of swapping space added, and
3. the number of logins allowed through the LOGINS, SET LOGINS, and NO LOGINS commands.

At the start of time sharing, the number of logins allowed is 1. One job is necessary for INIT to run. This number can be increased by the LOGINS command but is restricted by the swapping space available and by JOB MAX. When the system manager adds swapping space by ADD SWAPFILE commands, the capacity to handle more logins is increased. The LOGINS command actually sets the number of logins to the maximum allowed. This maximum can never exceed JOB MAX.

Swapping slots 0, 1, and 3 can be dynamically removed as well as added. To remove a swapping file or device, the system manager should first decrease the number of logins. This decrease is done by either the NO LOGINS or SET LOGINS command. The number of logins to be decreased depends on the capacity of the swapping slot to be removed. The decrease in the number of logins allows the system to safely remove swapping space.

The sequence of commands to remove a swapping slot is the reverse of the sequence to add swapping space. For example, assume that a system is running with a JOB MAX of 63 and is using an RS03 fixed head disk (nonfile structured) as swapping slot 1. The disk is capable of swapping 16 jobs with a swapping size of 16K words. The system manager must therefore decrease the number of logins by 16 jobs before removing the device. The following sequence shows the procedure.

```
# SET LOGINS 47
# REMOVE SWAPFILE 1
#
```

In actual practice, it may be necessary to wait for a time while the system adjusts operations. If the number of jobs currently running is greater than 47 or if the number of logins allowed is too large, the program prints the ?PROTECTION VIOLATION error message. The system manager must either wait until jobs are logged off the system or type the SET LOGINS command with a lower number. The SYSTAT or display program, in the free buffer status report, prints the number of jobs currently on the system.

If UTILITY prints the ILLEGAL VALUE - n message in response to the REMOVE SWAPFILE command, the swapping slot number n was a value other than 0,1, or 3.

The ?ACCOUNT OR DEVICE IN USE error is merely a warning by the system. One or more swapped out jobs are in the swapping space to be removed but otherwise the space can be removed properly. The system will lock the file and begin swapping jobs to other space. The system status report can be monitored to find out when the slot is not longer occupied.

A later issuing of the REMOVE SWAPFILE command will finish the operation. The SHUTUP program automatically removes swapping files.

It is possible to have more swapping space added than the system can use. In this case, logins need not be reduced to remove a swapping file. The only requirement is that enough room remain to swap all jobs currently logged in or to swap all jobs which can log in, whichever is greater.

7.1.7.2 Adding and Removing Overlay and Error Files — The overlay and error message files, like the swapping files, can be added and removed during time sharing. OVR and ERR are standardized names to denote the separate file for the overlay code and alternate file for the default error message file. Although other file names can be specified, it is highly recommended that the standardized names be kept. Regardless of the names given to the files, however, the system restricts the extension to .SYS.

The ADD OVERLAY and ADD ERROR commands force the system to access the associated data in the associated files on the related devices. To add these system files, simply specify the commands in a manner similar to the following examples.

```
#ADD OVERLAY DS0:OVR
#ADD ERROR DB0:ERR
```

For the overlay file, the system copies the overlay code from the monitor SIL to the file OVR.SYS on RS03 or RS04 unit 0. For the error message file, the system copies into ERR.SYS on RP04, RP05, or RP06 unit 0 the contents of the error message file established by the DEFAULT initialization option. Note that the disk on which an overlay or error file resides must be properly formatted, initialized and mounted. In particular, a fixed head disk used for overlay or error files must be a file-structured disk.

To add these files, the system ensures that the files exist and are properly formed. If any errors are encountered, UTILITY prints an error message in the following format.

```
text — in ADD OVERLAY
text — In ADD ERROR
```

Table 7-5 lists the possible errors. If no errors occur, the system has successfully added the file and is accessing it. No further commands are required.

To remove these files, the REMOVE OVERLAY and REMOVE ERROR commands can be issued at any time. The system thereafter accesses the original data. The SHUTUP program automatically removes both files.

Table 7-5: ADD OVERLAY and ADD ERROR Command Errors

Text and Meaning	
?CAN'T FIND FILE OR ACCOUNT	The file with the name given and with an extension of .SYS does not exist in account [0,1] on the related device.
?DEVICE NOT FILE STRUCTURED	The device specified is not a disk.
?DISK PACK IS NOT MOUNTED	The disk specified is not mounted.
?ILLEGAL FILE NAME	A name for the file is required and has not been specified; or the name specified contains illegal characters.
?MISSING DEVICE OR FILE NAME	The command must contain either a device specification or a device and file name specification.
?NAME OR ACCOUNT NOW EXISTS	The file being added is already installed and operational.
?NO ROOM FOR USER ON DEVICE	The file being added is not long enough. (The overlay file must be at least 32 blocks and the error file must be at least 16 blocks.)
?NOT A VALID DEVICE	The device specified is a disk but is not configured on this system.
?PROTECTION VIOLATION	Either the disk specified is mounted read only or the file is bad (is not contiguous or is currently open). The system must have write access to the device and to the file to successfully add it as a system file.

7.1.8 Defining Concise Command Language (CCL) Commands

CCL commands on RSTS/E allow users to type system-level commands which load and run programs from disk. The programs must be coded to recognize a CCL entry and to extract any command string passed to them. The *RSTS/E Programming Manual* describes the operation and interpretation of CCL commands.

Certain RSTS/E programs are designed to interpret a standard set of CCL commands. One program, UMount, distributed with RSTS/E, runs only by the CCL commands MOUNT and DISMOUNT. A RSTS/E installation can have any number of unique CCL commands. The system manager makes all CCL commands available to users by the UTILITY CCL command which creates the monitor structure required.

Because the monitor structure which defines a CCL command is transient, CCL commands must be defined at the start of each time-sharing session. It is suggested that the system manager place the proper command definitions in the start and crash control files. By this mechanism, INIT creates the proper structures automatically when it starts the system.

A CCL command definition occupies one small buffer on the system. All definitions are in a linked list of small buffers. UTILITY executes a SYS call to add and remove CCL definitions and provides a means to list all currently defined CCL commands.

7.1.8.1 Adding a CCL Definition — To add the CCL command, type the UTILITY command CCL as follows:

```
# CCL CRE-ATE=[1,2]EDIT.BAC;30000  
#
```

The above command adds the definition for the EDIT system program. The hyphen designates the abbreviation point so that typing CRE as well as CREA, CREAT, and CREATE runs EDIT. If no hyphen is specified, UTILITY places a hyphen at the end of the command and no abbreviation is allowed. Because no explicit device is given, the public structure (SY:) is assumed. Programs run by CCL command can be stored on any disk device, but the device, if not in the public structure, must be specified in the command definition. The account designation [1,2] means that EDIT must reside in the system library. If no account is given in the command, the account under which the job is running is assumed. The number following the semicolon is the line number at which execution starts when the EDIT runs by CCL command. The value 30000 is standard for most BASIC-PLUS programs.

The system performs error checking before adding the CCL command. If the command begins with a number or contains a character other than an alphabetic, numeric or commercial at (@) character, the ?ILLEGAL FILE NAME error is returned. A CCL command may not begin with a number because BASIC-PLUS processes line numbered input as a statement to be compiled. The command can have a maximum of nine characters. If more than 9 characters are present or if the equals (=) character is omitted, UTILITY prints the ILLEGAL FORMAT error message. The system ensures that the command is not currently defined. If a command is currently defined or contains an abbreviation which would cause misinterpretation, the system returns the error ?ACCOUNT OR DEVICE IN USE.

NOTE

When adding two or more commands which begin with the same set of characters, you should define the abbreviation points such that no ambiguity results, as between SHO-U and SHO-UT. However, you can first define a command with a longer abbreviation, then another with a shorter abbreviation. For example, you can first define EDT-, then ED-IT. But you cannot first define ED-IT and then EDT-.

To add a definition for a program which must retain its temporary privilege when it runs at a nonzero line number, include PRIV in the command as shown below.

```
#CCL COM-MAND=DK1:[1,18]FILE.BAC;PRIV 30000  
#
```

By specifying PRIV, the system manager ensures that BASIC-PLUS will retain a program's privilege when the program is run by the CCL command.

7.1.8.2 Listing Currently Defined CCL Commands — The UTILTY command LIST CCL prints a listing of currently defined CCL commands. UTILTY extracts a pointer to the root of the linked list of CCL definition entries, reads information from each CCL definition and prints a listing of the commands in the order in which they were defined. If no CCL commands are currently defined, UTILTY prints the message NO CCL COMMANDS NOW INSTALLED.

7.1.8.3 Removing a CCL Definition — To remove a CCL definition, type the command CCL and the definition followed by the equals (=) character. Either the full definition or a valid abbreviation is sufficient to denote the command to be removed. If the definition does not currently exist, the system generates the ?CAN'T FIND FILE OR ACCOUNT error. The removal occurs immediately.

7.1.9 System Logical Names

RSTS/E allows users to access devices by logical names as well as by physical names. Logical names which apply to all users are termed system logical names. On all systems, users can refer to a disk by its pack identification or by a name which replaces the pack identification. Thus, each disk unit configured on RSTS/E systems has the capability of being accessed by a system logical name. Logical names which apply to a single job are referred to as job-related logical names.

The system manager has the capability to define system logical names for nondisk devices and additional names for disk devices. The number of names allowed is a system generation parameter and varies from system to system.* Such a system logical name can have an account number associated with it. Consequently, use of the name not only refers to the related device but also the account on the device.

The system treats device names in the following manner. It checks the list of job-related logical names first. Each job can have up to 4 assigned logical names. The system then scans the table of disk logical names for a matching pack identification or equivalent. Next, the additional names are checked. Finally, the list of valid physical device names is checked. If individual users

*If the number of names configured is 0, an attempt to define a system logical name always returns the ?NO ROOM FOR USER ON DEVICE error because no table space is available. See the *RSTS/E System Generation Manual*.

have not defined job-related logical names which duplicate currently defined system logical names, all users have access to devices by the logical names the system manager adds.

A system logical name must be unique. Thus, a pack identification or its logical name cannot conflict with a pack identification or other system logical name. The system manager, however, can define multiple system logical names for the same device.

If the pack identification (or a logical name to replace the pack identification) of a disk to be mounted is the same as an existing system logical name, then the disk is mounted with no system logical name. Unless a different, unique name is specified, the disk, at mount time, has no logical name associated with the device and unit being used. (For more information on the disk logical name, see the description of the /LOGICAL option for the MOUNT command.)

System logical names apply for a single time-sharing session but can be removed or changed during the session. It is suggested that the system manager place the proper commands in the system start-up control files to automatically define names at system start-up time. For more details on system logical names, refer to the *RSTS/E Programming Manual*.

7.1.9.1 Adding New Names — The ADD LOGICAL command defines a system logical name for nondisk devices and an additional name for a disk device. If the name duplicates one currently defined, the ?ACCOUNT OR DEVICE IN USE error is returned. To change a currently defined name, issue the REMOVE LOGICAL command and reissue the ADD LOGICAL command again.

The logical name must be a legal file name. (That is, it must contain from one to six alphabetic or numeric characters.) The ?ILLEGAL FILENAME error indicates that the name contains illegal characters. If all entries in the table are occupied, the system returns the ?NO ROOM FOR USER ON DEVICE error. The device with which the name is associated must be configured on the system. If it is not, the ?NOT A VALID DEVICE error is returned.

7.1.9.2 Removing Logical Names — The REMOVE LOGICAL command deletes the association defined for the logical name specified. Because a system logical name must be unique, only the name need be specified in the command. If the name specified is not currently defined, the system generates the ?CAN'T FIND FILE OR ACCOUNT error. If no name or an illegally formed name is specified, the ?ILLEGAL FILENAME error is returned. To delete all logical names for a device, the REMOVE LOGICAL command must be issued for each currently defined name.

7.1.9.3 Changing a Disk Logical Name — The CHANGE LOGICAL command replaces any currently defined pack identification or logical name. This name is the one SYSTAT or the display program includes in its disk status report.

The command does not affect any additional name for a disk. The following sample shows usage of the command:

```
# CHANGE LOGICAL DB1:SYSTST  
#
```

The device specified must be disk or else the system returns the ?NOT A VALID DEVICE error. If the device specified is disk but is not configured on the system, the ?CAN'T FIND FILE OR ACCOUNT error is generated. The system checks the uniqueness of the new logical name. If it duplicates a name already defined, the ?ACCOUNT OR DEVICE IN USE error is returned. A new name must be specified in the command. If the name is omitted or is illegally formed, the ?ILLEGAL FILENAME error occurs.

To change the name of a nondisk device or to change an additional name for a disk device, issue the REMOVE LOGICAL and ADD LOGICAL commands in sequence. To delete all logical names for a disk, the REMOVE LOGICAL command must be issued for each currently defined name.

7.1.9.4 Listing System Logical Names — The LIST LOGICAL command prints, for each device assigned a logical name (or pack identification), the device designation and unit number, any account number associated with the logical name, and the logical name itself. The entries are printed in the order in which they are found in the monitor tables.

7.2 Monitoring System Status — SYSTAT

During normal time-sharing operations, the system manager should monitor the status of the RSTS/E system. Information concerning system status is gained either by use of the system program SYSTAT or by running the VT5DPY or the VT50PY program. The options and output supplied by the SYSTAT system program are described in the *RSTS/E System User's Guide*. The VT5DPY and VT50PY program descriptions appear in Section 7.3. The discussion here gives the system manager guidelines on how and when to use system status information.

Several uses of SYSTAT are described elsewhere in this manual in conjunction with other system manager operations. Those instances are listed here.

1. During preparation for system shutdown, to determine active jobs and disk devices and assignable devices in use.
2. In conjunction with the UTILTY system program command HANGUP, for determining malfunction of a remote line.
3. In conjunction with the UTILTY commands NO LOGINS and SET LOGINS, when the number of free small buffers is less than 10% of the small buffers on the system.

Refer to the discussion relevant to the individual system program or system operation for more information on the above uses of SYSTAT.

Further uses of SYSTAT are listed below and discussed in the ensuing paragraphs.

4. Uncovering malfunctioning keyboards by the HUNG TTY count.
5. Guarding against a disk device filling up by watching the FREE block count.
6. Following the progress of user jobs or detached jobs by the STATE and RUN-TIME items of job status.

Item (4) refers to the HUNG TTY count reported in the SYSTAT buffer status report. A HUNG TTY count of zero is good. A HUNG TTY count of nonzero indicates the presence of a malfunctioning terminal or terminals. The ERRDIS program can identify the device or devices causing the error count. If the HUNG TTY count increases rapidly, a field service representative should be consulted.

The FREE block count mentioned in item (5) reflects the apparent number of free blocks on each disk and is given in the disk status report of SYSTAT. For practical purposes, however, such as for allocation a file on the device, all of the free blocks reported by SYSTAT may not be usable. A ?NO ROOM FOR USER ON DEVICE message may be generated although SYSTAT reports that enough FREE blocks exist. The file cluster size or the number of clusters required can prevent a file from fitting on the device desired. For example, a file whose cluster size is 16 and whose length is 10 blocks possibly does not fit on a device which SYSTAT reports to have 50 free blocks of file space remaining. The cluster size of 16 demands that 16 contiguous blocks of free space must exist on the device before the file can be allocated to the device. In some cases, 16 contiguous blocks simply do not exist on a device. RSTS/E does not allow a file to extend to another physical device.

A further condition exists for showing ?NO ROOM FOR USER on a device. The UFD is perhaps full and cannot accommodate the creation of another file. The UFD cluster size was not made large enough when the account was created with REACT.

The occurrence of jobs being stalled in a resource sharing system is detectable by the means presented in item (6). If the system manager notices that a RUN-TIME value of a job is not increasing (the value is printed out in a job status report), it indicates that the job is stalled, waiting for an I/O device. One user job in the system can ASSIGN a device or keep an assignable device locked by having one file open on it. The system manager can determine the selfish user job by examining the device status report which associates the busy device with the job number of the user controlling that device. The system manager can request that the user free the device or, if that is not viable, can use UTILITY commands to force the job off the system or SEIZE the device.

The status of detached jobs is of interest also. If a detached job is reported by SYSTAT to be in the HB state (hibernate), it is never eligible for run time. The HB state indicates that the detached job is attempting to communicate with its terminal. The problem of a detached job in the HB state is handled by logging into the system at a free terminal, by using the attach capability of LOGIN and attaching the job to a terminal. Once the detached job is attached to a terminal, messages can be printed.

The free status report tells the number of jobs currently logged into the system and the maximum number of jobs allowed to be logged into the system. These numbers are useful when swapping files are being added to and removed from the system. After having added swapping files and raised the logins allowed, the system manager can check this report to confirm the success of the procedure. Before removing a swapping file, this report can be checked to ensure that the number of logins allowed is low enough to enable removal of the swapping file.

7.3 Dynamic Display of System Status — VT5DPY and VT50PY

The VT5DPY and VT50PY system programs display the system status on a VT05, VT50, VT52, or VT100 alphanumeric display terminal and update the status at given intervals. The programs are stored in the system library with protection code <232>. Storing the programs with a protection code of <124> restricts their usage to privileged accounts with an adequate size maximum. Since the programs require a job size of 16K words, it is suggested that they be used only on systems with sufficient memory.

Two programs are provided to display system status: VT5DPY (for the VT05 alphanumeric display terminal) and VT50PY (for the VT50, VT52, and VT100). Both programs are the same except for the output routines. (The hardware commands and screen capacity of the VT05 differ from those of the VT50 and VT52.) Each program is created at system generation time by appending the related output routines (VT05.DPY and VT50.DPY) to a common display program (DISPLY.BAS) and compiling the result.

The operation of each program is automatic or can be varied at user option. When the user starts the program, the interval at which the information on the screen is updated must be specified. The information displayed is similar to that given by the SYSTAT system program. However, the user can type commands to modify the items displayed. For more information on SYSTAT, refer to the *RSTS/E System User's Guide* and Section 7.2 of this guide.

7.3.1 Running and Terminating VT5DPY and VT50PY

The command RUN \$VT5DPY should be used on a VT05 terminal and the command RUN \$VT50PY should be used on a VT50 or VT52 terminal.

The program runs and prints the INTERVAL question to which the user can type the number of seconds between updates and any combination of the options listed in Table 7-6.

Table 7-6: Display Program Options

Option	Description
/DET	Detaches the job from this terminal or from the terminal specified.
/KBn:	Prints the output at keyboard unit n if it is available. If /DET is specified, run the program detached.
/NOFILL	Sets the fill factor to 0. Improves program performance when the VT05 runs at 300 baud or less.
/PRIORITY	Runs the program at special priority rather than at -8 priority.
/24	For the VT52 terminal with the VT50PY program. Tells the VT50PY program to use 24 lines on the screen rather than 12.
/DCA	For the VT52 terminal with the VT50PY program. Performs output to the terminal by direct cursor addressing rather than by spacing and generating line feeds.
/TAB	For the VT52 terminal with the VT50PY program. Uses the hardware tab feature of the terminal rather than generates spaces for horizontal tab positioning.
/ECHO*	Performs output on the terminal by using echo control features. Consequently, echoing is disabled while the program updates the screen and spurious characters do not ruin data displayed.
*Echo control is an optional feature for the RSTS/E Monitor and may not be present on all systems.	

If the user types only the RETURN key or types no number with a modifier in response to the INTERVAL question, the program updates the status every 15 seconds.

Running the program detached allows the user to temporarily interrupt the program and use the terminal to run other programs. To interrupt the display, type CTRL/C which causes the program to print a message telling the user that the terminal is available. When the user releases the terminal by logging off the system, the program automatically displays the status information on the screen again.

To terminate the program, type CTRL/C. If the program is running attached to the keyboard, this action returns the job to command level. If the program is running detached from the keyboard, this action causes the TERMINAL IS ALL YOURS NOW message. The user must attach the job running the program to the terminal. The following sample dialogue shows the procedure.

CTRL/C

THE TERMINAL IS ALL YOURS NOW

ATT 9
PASSWORD:
ATTACHING TO JOB 9
CONTINUE? NO

READY

When the program is attached to the terminal, it prints the CONTINUE? question. Type NO or any string not beginning with Y to terminate the program.

When the program displays the system status, it prints a header line on the top line of the screen, skips a line, and fills the leftmost half of the screen with job status information and the rightmost half of the screen with, in turn, busy device, disk structure, run-time system, free buffer status, and message receiver information. Upon filling the screen, the program moves the blinking cursor to the first character position on the second line of the screen. This indicates that the program is idle.

At the interval specified, system tables are interrogated and the status information on the screen is updated with any changed data. While executing routines to extract update information, the program prints the message WORKING ... and leaves the cursor positioned to the right of the message. After completing the update, the cursor is returned to its idle position.

While the cursor is at its idle position, the user can type commands to modify the contents and arrangement of items on the screen. Any command typed should be terminated by the ALTMODE key. Although any line terminator works, ALTMODE leaves the cursor positioned on the blank line. The program takes no action on invalid commands.

Some commands which add items to the screen can be preceded by a minus sign negates the effect of the command. Table 7-7 lists and describes the commands.

7.3.2 Screen Layout

The program partitions the screen into three parts: the header line, the left-hand half, and the righthand half. The components of these parts are defined in the following subsections.

Table 7-7: Display Program Commands

Command Type	Format and Description
General	<p><i>C</i> Clears the screen and displays new status.</p> <p><i>Sn</i> Displays memory status in place of job status. Starts with the 8K word section less than or equal to n. If n is not given, starts at the beginning of memory.</p> <p><i>J</i> Displays job status in standard manner.</p> <p><i>Jn</i> Displays job status starting with active job n+1. Overcomes physical limitation of the screen.</p>

(continued on next page)

Table 7-7: Display Program Commands (Cont.)

Command Type	Format and Description
General (cont.)	<p><i>Xn</i> Changes the interval to n seconds.</p> <p><i>X0</i> Updates the display with an interval of 0 seconds (that is, runs continuously) but lowers the priority so that other jobs are not stalled.</p>
Job Status	<p><i>O</i> Displays the account number of operator jobs as [OPR]. An operator job has a project number 1 and a programmer number less than 200.</p> <p><i>-O</i> Replaces OPR in operator account designations with the actual project and programmer numbers.</p> <p><i>T</i> Displays total CPU time each job has expended. The time is displayed as number of hours, minutes, seconds, and tenths of seconds under the RUN-TIME column.</p> <p><i>+</i> Displays the increment of CPU time each job has expended since the display program last updated the screen. User can return to total CPU time by typing T.</p> <p><i>%</i> Displays the amount of CPU time each job has expended as a percent of the total CPU time expended. User can return to total or increment of CPU time by typing, respectively, T or +.</p> <p><i>J-O</i> Does not display operator jobs. Operator jobs are those running under project number 1 and programmer number less than 200.</p> <p><i>JO</i> Displays only operator jobs.</p> <p><i>J+O</i> Includes operator jobs in display.</p> <p><i>J-D</i> Does not display detached jobs.</p> <p><i>J+D</i> Includes detached jobs in display.</p> <p><i>JD</i> Displays only detached jobs.</p> <p><i>J-S</i> Does not display sleeping jobs.</p> <p><i>J+S</i> Includes sleeping jobs in display.</p>

(continued on next page)

Table 7-7: Display Program Commands (Cont.)

Command Type	Format and Description
Job Status (cont.)	<p><i>N</i> Indicates program name in WHAT column.</p> <p><i>-N</i> Removes program name from WHAT column and replaces it with name of the RTS under which job is running.</p> <p><i>P</i> Indicates priority of jobs more exactly than + and - characters.</p> <p><i>-P</i> Indicates priority of jobs with + (for higher than normal) and - (for lower than normal) characters.</p> <p><i>W</i> Indicates under STATE column, the last WAIT state rather than actual state.</p> <p><i>-W</i> Removes last WAIT state and indicates actual state of each job.</p> <p><i>K</i> Displays, under SIZE column, the amount of memory occupied by each job.</p> <p><i>-K</i> Displays, under SIZE column, the amount of memory remaining to each job.</p>
Disk Structure	<p><i>D, Dn, -D</i> Displays disk structure statistics. If n is 1, places item first on the screen. Preceding minus sign removes same from the screen.</p> <p><i>L, -L</i> Displays, under COMMENTS column, the logical name of each device. Preceding minus sign replaces logical names with standard PUB, PRI, NFS, or LCK notations.</p>
Busy Devices	<p><i>B, Bn, -B</i> Displays busy device statistics. If n is 1, places item first on the screen. Preceding minus sign removes same from the screen.</p>
Free Buffer	<p><i>F, Fn, -F</i> Displays free buffer statistics. If n is 1, places item first on the screen. Preceding minus sign removes same from the screen.</p>
Message	<p><i>M, Mn, -M</i> Displays message receiver statistics. If n is 1, places item first on the screen. Preceding minus sign removes same from screen.</p>
Run Time System	<p><i>R, Rn, -R</i> Displays run-time system data. If n is 1, places item first on the screen. R preceded by a minus sign removes same from the screen.</p>
Resident Libraries	<p><i>H, Hn, -H</i> Displays resident library data. If n is 1, places item first on the screen. H preceded by a minus sign removes same from the screen.</p>

7.3.2.1 Header Line — The header line looks like the following sample.

```
RSTS V7.0 TIME SHARING STATUS ON 18-SEP-78 11:15 UP 2:22:22
```

The header line contains the system identification, the current date and time of day and the number of hours, minutes and seconds since the start of time-sharing operations. The last item is termed up time.

7.3.2.2 Job Status — The job status section consists of job statistics information similar to that supplied by SYSTAT with the addition of a PR column for running priority.

The PR column can display the following abbreviations.

If -P is in effect:

+	Higher than normal priority
-	Lower than normal priority
S	Special run priority
^	CTRL/C temporary priority
K	Keyboard delimiter temporary priority

If P is in effect:

+n	Positive priority $n * 8$
0	Zero priority
-n	Negative priority $n * 8$

7.3.2.3 Memory Status — The S command causes the program to print a table indicating the usage of each 1K word portion of memory. This memory status report replaces the job status report on the lefthand half of the screen. To display the job information again, use the J command.

Because of the limitations of the screens, all memory status information may not be displayed simultaneously. The Sn command allows the user to determine the starting 8K section. The display program prints the starting section number for the row and 8 abbreviations per row. Each of the 8 abbreviations concerns the status of a 1K section of memory. The number of rows printed, and thus the extent of memory covered, is limited by the terminal. The program indicates the starting 1K section by printing a header line in the following format.

MEMORY USAGE (STARTING AT nK)

The memory status report uses the following abbreviations.

MON	Occupied by RSTS/E Monitor.
n	Occupied by Run-Time System n where n is the position in the run-time system list. (A question mark (?) is printed if the RTS name cannot be determined.)
n	Occupied by job number n.
nLK	Job number n is locked in this 1K portion.
nS	Job number n is being swapped out of memory.
nSI	Job number n is being swapped into memory.
NXM	Memory space is nonexistent.
END	End of physical memory for user jobs.
LCK	Memory is locked.
EBP	Memory is reserved for the extended buffer pool.

7.3.2.4 Disk Structure — The disk structure report produced by the D command is the same as that printed by SYSTAT. The L command replaces abbreviations in the COMMENTS column with the system logical name for the disk.

7.3.2.5 Busy Device Statistics — The busy device statistics are the same as those generated by SYSTAT.

7.3.2.6 Message Receiver Statistics — The message receiver statistics reflect information on jobs using local or network interjob communication. Some of the abbreviations are shorter than those printed by SYSTAT due to limited space on the screen.

7.3.2.7 Free Buffer Status — The free buffer statistics are the same as those generated by the SYSTAT program.

7.3.2.8 Run-Time System Statistics — The statistics for run-time systems are displayed on the righthand portion of the screen. They are the same as those generated by SYSTAT. Some information in the COMMENTS column may be omitted due to limited space on the screen.

7.3.2.9 Resident Library Statistics — The statistics for resident libraries are displayed on the righthand portion of the screen. They are the same as those generated by SYSTAT. Some information in the COMMENTS column may be omitted due to limited space on the screen.

7.4 Terminal and Remote Line Characteristics — TTYSET

The RSTS/E system operates with a variety of terminals. During system generation, the system manager specifies the numbers and types of terminal interfaces as part of the hardware configuration. The types of terminals are intentionally not specified. Since many terminals can operate in Teletype mode at a speed of 110 baud, the system automatically sets the default characteristics of all line interfaces to those of the ASR-33 device. The system manager or privileged user must set the characteristics of other types of terminals.

The TTYSET system program sets terminal characteristics. As described in the *RSTS/E System User's Guide*, a user can set the characteristics of his own terminal by TTYSET commands. The system manager or privileged user can set the characteristics of terminals other than his own by the privileged TTYSET Kbn: command. The standard method of setting characteristics of local lines is by the TTYSET commands in the START.CTL file. This procedure sets up all local lines automatically at the start of each time-sharing session.

Two methods exist to set characteristics of remote lines. First, all remote lines can have the default characteristics of the ASR-33 device. Consequently a remote user must log into the system at a speed of 110 baud and then run TTYSET to set the characteristics of his terminal. If the remote line is connected to a DC11 single line interface or to a DH11 or DZ11 multiplexer line, the user can type the TTYSET SPEED command to change the baud rate. The terminal characteristics revert to those of the ASR-33 device when the user hangs up the line.

The second method for setting characteristics of remote lines involves setting the ring characteristics. The system manager or privileged user can run TTYSET to set the ring characteristics which cause the system to automatically use the characteristics so set every time a call is answered on a particular remote line. By setting ring characteristics for a remote line, the system manager can establish certain lines for alphanumeric display terminals running at 300 baud, for 2741 communications terminals, or for other types of terminals. The ring characteristics remain in effect for the current time-sharing session unless changed again by TTYSET. TTYSET commands in the START.CTL file can also establish ring characteristics.

7.4.1 Terminal Line Speed Characteristics File — TTYSET.SPD

There are two circumstances that require the system manager to create a special terminal line speed characteristics file:

1. The system manager has decided to restrict the valid line speeds at certain terminals to a subset of the line speeds allowed by the terminal interface.
2. The system manager has had installed an interface that has been modified so that non-standard line speeds have been substituted for one or more standard interface line speeds.

To create the file, the system manager gathers information on the restrictions/modifications and the keyboard numbers of the terminals affected. Then, the system manager must create a file named TTYSET.SPD stored in the system library account [1,2], by running PIP or an editor program.

The format of each line in the TTYSET.SPD file is a series of ASCII entries, separated by commas, which specify the keyboard number and the line speeds for the terminal affected. Entries in a line are positional. That is, the first entry must be the keyboard number, the second entry must be the first programmable baud rate allowed by the interface, the third entry the next baud rate allowed, and so on, to a maximum of 16 specified baud rates (the maximum allowed by an interface).

To restrict the valid line speeds of a terminal to a subset of the line speeds allowed by the interface,

1. Determine the line speeds allowed by the interface by referring to the hardware documentation for the interface.
2. Type the keyboard number, then type the speeds you will allow or a -1 (for speeds you disallow) in the exact order presented in the hardware documentation.

For example, assume that a DH11 controls keyboard 11 (among others). The DH11 interface allows 14 baud rates plus two external inputs as shown in the following chart (a DH11 Speed Table excerpt).

Speed Table for Receiver and Transmitter Speeds:

Transmitter	Bits				
	13	12	11	10	
Receiver	9	8	7	6	
	—	—	—	—	
	0	0	0	0	Zero Baud
	0	0	0	1	50 Baud
	0	0	1	0	75 Baud
	0	0	1	1	110 Baud
	0	1	0	0	134.5 Baud
	0	1	0	1	150 Baud
	0	1	1	0	200 Baud
	0	1	1	1	300 Baud
	1	0	0	0	600 Baud
	1	0	0	1	1200 Baud
	1	0	1	0	1800 Baud
	1	0	1	1	2400 Baud
	1	1	0	0	4800 Baud
	1	1	0	1	9600 Baud
	1	1	1	0	External Input A
	1	1	1	1	External Input B

The system manager wishes to restrict valid line speeds for KB11: to the following:

110, 300, 1200 baud

His line in the TTYSET.SPD file for KB11: must be:

11,-1,-1,-1,110,-1,-1,-1,300,-1,1200,-1,-1,-1,-1,-1

Note that every possible entry must be specified in the correct position.

To specify the line speed modifications that have been substituted,

1. Determine the positions in the speed table for the interface line(s) that have been affected by the non-standard line speeds.
2. Type the keyboard number, then the line speeds allowed by the interface, substituting the strings specifying the new speeds in the positions of the replaced line speeds.

For example, assume the use of a DH11 multiplexer and KB11:. The system manager has had a non-standard line speed set on the DH11, that affects the 2400 baud position. The new line speed (2350 baud) replaces the 2400 baud position.

The TTYSET.SPD entry for KB11: (and any other terminal driven from the DH11) would be created as:

11,0,50,75,110,134.5,150,200,300,600,1200,1800,2350,4800,9600,-1,-1

7.4.2 TTYSET Privileged Feature — KBn: Command

The system manager sets the characteristics of other terminals in the RSTS/E system by use of the KBn: command. The system manager, while logged into the system under a privileged account, calls the TTYSET system program as follows.

```
RUN $TTYSET
TTYSET V7.0 RSTS V7.0 TIME SHARING
TERMINAL CHARACTERISTICS PROGRAM
?
```

The program responds with a header line and a question mark character (?), which indicates that the program is ready to accept commands. If the system manager wishes to set the characteristics of a VT52 terminal at keyboard 3, the following commands should be typed.

```
? KB3:
FOR KB3:? VT52
FOR KB3:?
```

In response to the system manager typing the KB3: command, TTYSET prints the FOR KB3: prompting message. This message indicates that the commands typed apply to keyboard number 3. The VT52 command immediately sets the characteristics of the line to those of a VT52 alphanumeric display terminal. TTYSET prints the prompting message again. Another KBn: command can be typed to set characteristics of another keyboard.

The system manager can also change discrete characteristics of a terminal. To limit the line length of the terminal at keyboard unit 4, type the following commands.

```
FOR KB3:? KB4:
FOR KB4:? WIDTH 60
FOR KB4:? EXIT
```

READY

As a result of the execution of the above commands, whenever 60 characters are printed on a line at keyboard unit 4, a carriage return and line-feed operation is performed. The EXIT command terminates TTYSET.

The system manager can also use the TTYSET privileged command /RING. This command is used to specify characteristics on a disabled terminal. For example, /RING can be used to set an initial WIDTH characteristic for a pseudo keyboard.

7.4.3 Automatic Setting of Terminal Characteristics

The setting of terminal characteristics in the RSTS/E system applies only to the current time-sharing session. This condition allows for replacement of an ASR-33 type terminal with another terminal without having to change the system configuration and, thus, rebuild the system. Since it is quite bothersome for the system manager or a user to set terminal characteristics each time the system is initialized, the INIT system program can automatically set both local and remote terminal characteristics by commands in the START.CTL and CRASH.CTL files. Refer to the sample START.CTL and CRASH.CTL files in Section 3.1.

7.4.4 Setting Terminal Characteristics of Remote Lines — /RING

If you are a remote line user whose terminal is other than an ASR-33 type terminal, you must set the characteristics of your terminal each time you log into the system. To ensure recognition of the characteristics of that terminal by the RSTS/E system. The characteristics discussed in Section 7.4.3 remain set until either the telephone is hung up, the line is disconnected or the system shuts down or restarts.

The /RING option relieves the user from the necessity of setting the characteristics each time he or she logs into the system on a certain remote line. For example, to set the characteristics of the remote line on keyboard 14 for the current time-sharing session, the system manager can run TTYSET under a privileged account and type commands as follows.

```
? KB14:/RING
FOR KB14:(RING)? LA36
FOR KB14:(RING)?
```

When the system manager indicates the /RING option with the KBn: command, TTYSET prints the prompting message FOR KBn:(RING) after which

any TTYSET command can be typed. The command takes effect immediately and TTYSET prints the prompting message again. If the system manager is not logged into the system under a privileged account or the characteristics of the line conflict with the command, TTYSET prints an error message and the prompting message. If TTYSET is successful, the characteristics for the remote line are set for the duration of the current time-sharing session. Each time a user dials the particular line, the system uses the characteristics for that terminal.

One caution is in order. The DL11E-type interface and the individual local interfaces (KL11, DL11-A through DL11-D, and LC11) do not have programmable baud rates. Therefore, the user cannot execute commands to change baud rates on a keyboard line having any of those interfaces.

7.5 Reinitializing a Disk During Time Sharing — DSKINT

The system manager or a privileged user can reinitialize a RSTS/E disk pack or cartridge and initialize a nonfile structured disk with the DSKINT system program. Before a disk can be used under the file system of RSTS/E, it must be initialized to the minimal file structure by the DSKINT option of the initialization code. This procedure is described in the *RSTS/E System Generation Manual*. Such a procedure is done while no other users are running on the computer. The DSKINT program, in contrast, runs during time sharing but initializes an already formatted disk.

The DSKINT program performs the following operations on a disk:

1. Creates a new SATT.SYS file in account [0,1],
2. Builds a zero length BADB.SYS file or retains the current BADB.SYS file in account [0,1],
3. Rebuilds account [1,1] to contain directory information for account [0,1] only, and
4. Writes a dummy bootstrap block on block 0 of the disk.

Additionally, DSKINT can reset the pack identification, pack cluster size, MFD password, MFD cluster size, and the public and private designation; it also can position the SATT.SYS file on the disk.

NOTE

The DSKINT system program does not format the disk and does not check for bad blocks as the DSKINT initialization option does. Therefore, the DSKINT program should be used only on disks known to be sound and reliable.

Because the DSKINT program rebuilds the storage allocation table, all user files on the disk are destroyed.

Because the pack cluster size of a disk can be increased, a current BADB.SYS file containing bad blocks may grow larger. This growth occurs if the current

bad blocks are not contiguous and reside in more than one new, larger cluster. The DSKINT program, in reclustering, may have to allocate formerly good blocks to the new cluster containing a bad block.

7.5.1 Running and Terminating DSKINT

DSKINT is called by the RUN command as shown below.

```
RUN $DSKINT
DSKINT V07.0 RSTS V7.0 TIME SHARING

DISK?
```

The DSKINT program responds by printing an identifying header line and the first question. The questions are a subset of the DSKINT initialization option questions. That is, a short form question is automatically printed. If only the RETURN key is typed in response to the short form question, a long form question is printed. Typing the LINE FEED key accepts any default answer to a question. Typing a CTRL/Z or CTRL/C, however, terminates DSKINT. Typing EXIT in response to the DISK? question also terminates DSKINT.

DSKINT does not print the questions concerning the date, time, number of platters, library account, formatting and pattern checking. The following is a sample dialogue showing the questions printed.

```
DISK? DB
UNIT? 0
PACK ID? TEST
PACK CLUSTER SIZE? 2
SATT.SYS BASE? 1999
MFD PASSWORD? RSTS
MFD CLUSTER SIZE? 4
PUB OR PRI? PRI
DATE LAST MODIFIED? Y
NEW FILES FIRST? N
USE PREVIOUS BAD BLOCK INFORMATION? N
PROCEED (Y OR N)? Y
DB0: HAS BEEN SUCCESSFULLY INITIALIZED!
DISK? EXIT

READY
```

After the UNIT? question is answered, DSKINT checks to see whether it has write access to the disk. The disk must be write enabled but must not be logically mounted. If the message DB1: — HUNG OR WRITE LOCKED occurs, DSKINT returns to the DISK question. The disk must be made ready before proceeding. If the disk is logically mounted, the program prints the message SORRY, CAN'T DSKINT A MOUNTED DISK and reprints the DISK question. The disk must be dismounted before the program can initialize.

DSKINT next checks cluster 1 on the disk. If the first cluster is bad, the message CLUSTER 1 (REQUIRED FOR MFD) CONTAINS A BAD

BLOCK. is printed. Because RSTS/E file-structured disks need the first device cluster for the MFD label entry, a new disk must be obtained. The DISK? question is reprinted.

If the response to the questions was a disk type or disk unit that is not configured on the system, the message SORRY, BUT THAT DISK DOESN'T EXIST is printed.

If the disk specified is already a formatted RSTS/E disk, DSKINT will print the pack ID, pack cluster size, and whether it is a public or private disk.

After the preliminary checks are performed, the program proceeds with the dialogue. An incorrect response to a query simply causes DSKINT to repeat the question.

If the response to the question:

USE PREVIOUS BAD BLOCK INFORMATION?

is Y, DSKINT uses the current bad block file information (if present) to create the new bad block file on the disk. If the response is N, DSKINT creates a zero-length bad block file on the disk.

7.5.2 DSKINT Processing and Unexpected Errors

When the PROCEED question is answered with Y, DSKINT begins to extract information to build the disk structure. If DSKINT encounters any errors in the process of extracting information, it prints a message in the following format:

UNEXPECTED ERROR — <text> — AT LINE #n

where <text> is error text and n is the line number at which the error was encountered. For example, if DSKINT is retrieving an existing entry on the disk and encounters a bad block, it terminates with the ?USER DATA ERROR ON DEVICE error. It also terminates when, in reclustering the disk, it finds that the maximum number (161) of bad block clusters is reached. The message EXCESSIVE BAD CLUSTERS is printed. In general, if any unexpected error occurs, the program terminates. The system manager can then use the DSKINT initialization option which can more thoroughly and efficiently examine the disk through pattern checking.

DSKINT assembles the entire file structure for the disk and performs all structural error checking before it writes any of the new structure to the disk. Should an unexpected structural error occur, the disk's original data has not been destroyed.

When the entire structure is assembled, DSKINT actually begins writing the new data to the disk. If a bad block prevents requested positioning of the SATT.SYS file, DSKINT simply places the file at the most convenient location. After the new structure is written to the disk, the program writes the dummy bootstrap block in cluster 0 and mounts and cleans the disk. Cleaning the disk actually creates the storage allocation table. DSKINT then

dismounts the disk to leave it in the same state it was in before the DSKINT procedure started.

To signal that the process has completed, DSKINT prints a message in the following format.

```
DB1: HAS BEEN SUCCESSFULLY INITIALIZED!
```

```
DISK?
```

The program reprints DISK? to allow another disk to be initialized.

7.5.3 Recommended Procedures Following DSKINT

After DSKINT finishes, a newly initialized disk remains dismounted on the drive. The system manager, at this point, should logically mount the disk. The system manager next should run REACT to create user accounts on a private disk because no accounts other than [0,1] and [1,1] exist on the disk.

7.6 Using The ONLCLN Program

The ONLCLN program allows you to repair corrupt disk file structures while you are operating under time sharing. To do this you must physically but not logically mount the corrupt disk, invoke the ONLCLN program with a RUN command, and identify the disk that you want cleaned. ONLCLN then performs several operations to correct the altered file structure.

DIGITAL recommends that you add commands to the CRASH.CTL command file to run ONLCLN. Include in this file the specifications for all disks that will be mounted during the crash/startup procedure.

7.6.1 When to use ONLCLN

There are several RSTS/E system errors that can occur which may be corrected by using the ONLCLN program. Three of these errors are listed here:

1. ?Disk pack needs "CLEANing"
2. ?Bad directory for device
3. ?Corrupted file structure

When you physically and logically mount a disk, the monitor sets a 'dirty' bit located on the mounted volume. If you inadvertently remove the pack without logically dismounting it, the bit remains set. When you attempt to remount the pack on a drive, the monitor will notice the 'dirty' bit is set and issue a message indicating that the disk pack needs to be cleaned. At that point, simply invoke the ONLCLN program to perform the cleaning operation and then logically mount the disk. The disk pack will subsequently be ready for system use.

As part of the cleaning operation, ONLCLN finds and deletes corrupt disk directories. If the monitor prints the ?BAD DIRECTORY FOR DEVICE message, you may be able to correct the error by running ONLCLN. In addition, ONLCLN may correct an error that caused a corrupt file structure condition to occur.

7.6.2 Invoking ONLCLN

To invoke the ONLCLN program, type RUN \$ONLCLN followed by the RETURN key. In response, ONLCLN prints the following dialogue question:

Disk?

Type the mnemonic name, unit number of the drive on which the disk is mounted, a colon (:), then press the RETURN key. DB0: represents, for example, the proper response if you want ONLCLN to clean an RP06 mounted on unit 0. If you decide to specify only the mnemonic name, ONLCLN issues another question to determine the unit number of the mounted drive. Answer the following question:

UNIT?

by typing the unit number and the RETURN key. If you press the RETURN key in response to the DISK question rather than type a device specification, ONLCLN prints a help message:

Type the name of the disk to clean.
The disk must be physically but not logically mounted.

and reissues the question. As soon as you have specified the correct volume, ONLCLN performs the following tasks:

1. Builds a new storage allocation table (SATT.SYS) that reflects all the files on the disk.
2. Scans all directories and deletes all files that have the extension .TMP, are marked for deletion, or have no accounting entry.
3. Finds any doubly-allocated blocks and allows you to specify their correct allocation.
4. Finds and deletes all invalid directories.
5. Marks the disk as having a clean file structure.
6. Allows you to delete files that contain bad blocks.
7. Zeroes all blocks that were in the old storage allocation table (SATT.SYS) but not in the new one.

When the operation is complete, ONLCLN returns you to your default run-time system prompt.

7.7 Optimizing Disk Directory Structure — REORDR

The system catalogues files on a RSTS/E disk in user file directories (UFDs). As users create, delete, and extend files, the file directory entries become scattered across the surface of the disk. This scattering increases both the time to open a file and the time to obtain data blocks.

When RSTS/E creates a new file in an account, it adds the file name at the end of the current list of file names unless the disk was initialized with the NEW FILES FIRST? option. Therefore, the files are ordered in the UFD according to creation date and time. This ordering may not be convenient because, to open a recently created file, the system must step through the entire list.

Additionally, the retrieval pointer information for the data blocks in a file may become scattered throughout the UFD. Therefore, as file processing proceeds, the monitor cannot get sufficient information in a single directory file read operation to allow many data block accesses before the next directory file read operation is needed.

REORDR provides three directory restructuring functions.

1. It places the list of file names in as few physical disk blocks as possible.
2. It attempts to place all directory information concerning the physical location of file data (retrieval pointers) for a file into contiguous disk blocks.
3. It sorts the file name list in one of four user specified orders. (This function is optional).

The first and third functions reduce the number of directory accesses required for a file open operation, while the second function reduces the number of disk directory accesses required to perform disk read and write operations.

The four optional, user-selectable UFD reordering methods are:

1. Forward By Creation Date. This method reorders files in increasing chronological order by creation date and time.
2. Reverse By Creation Date. This method reorders files in decreasing chronological order by creation date and time.
3. Forward By Access Date. This method reorders files in increasing chronological order by access date.
4. Reverse By Access Date. This method reorders files in decreasing chronological order by access date.

NOTE

The DSKINT disk initialization procedures allow the system manager to define the access date on individual disks as being the date of last modification. For disks whose access date is the date of last modification, ordering by access date is not a useful option.

REORDR operates only on a single disk unit and only on UFDs of user accounts and the UFD of the system library account. It does not process the MFD account [1,1] or the UFD of the system account [0,1]. The disk on which REORDR operates must be logically mounted with write access.

7.7.1 Dialogue Questions and Responses

The system manager must run REORDR while logged into the system under a privileged account. REORDR is stored in the system library account with a protection code of <124>. The program prints a dialogue as shown in the following sample printout:

```
RUN $REORDR

REORDR V7.0 RSTS V7.0 TIME SHARING
DIRECTORY REORDERING PROGRAM

SORT DIRECTORY(S) (YES/NO) <NO>? YES

ORDER BY CREATION] OR ACCESS] DATE <CRE>? (RET)

IN FORWARD] OR REVERSE] ORDER<FOR>? (RET)

DEVICE AND UFD SPECIFICATION(S)? DB0:[*,*],DB1:[*,*]
```

REORDR prints its identification lines and the first question asking whether the UFD reordering is desired. If the system manager types YES, the program prints two more questions concerning the sort method desired. Questions concerning the sort and the sort methods show the responses expected. The abbreviation points for responses are signalled by square brackets. The default answer within the less than and greater than symbols applies when only the RETURN key is typed as a response. If the system manager does not request sorting, REORDR omits the questions concerning the sort methods and asks for the device and accounts to be reordered. Without sorting, the name entries of files are kept in their current order.

When sorting is requested, files in an account on the device can be ordered by their creation dates and times or by their dates of last access. Access date can have one of two meanings depending on the way the disk was initialized. If the access date field means date of last access, the date designates when the file was last opened, either for input or output. If the date of last access on that device is set to mean date of last modification, the date designates when the file was last updated. (The abbreviation DLW in the comments part of the disk status report of SYSTAT and the display programs shows that a disk's access date is set to mean date of last modification.)

If the system manager types an incorrect response to any of the sort and order questions, REORDR merely reprints the question.

In response to the specification question, the system manager must give a specific device and unit designation along with the account designation. Although SY: (for the entire public structure) is not a valid device, SY0: (for the

system disk, the device bootstrapped) is acceptable for the device response. An asterisk is valid in either or both of the project and programmer fields in the account designation to denote all project or all programmer numbers on the device. Multiple specifications, separated by commas, are valid.

Valid account designations are shown below.

Account Designations	Meaning
[*,*]	All UFDs on the disk.
[m,*]	All UFDs with project number m.
[:,n]	All UFDs with programmer number n.
[m,n]	Only the UFD with project number m and programmer number n.

Although the accounts [0,1] and [1,1] can be implicit in the account designation, REORDR excludes them from the reorder process.

7.7.2 Reordering the Public Structure

To reorder all disks in the public structure, the system manager must specify each device and unit designation (with the appropriate account designation). For example, to reorder a public structure on RP04 unit 0 and RP04 unit 1, the system manager can type the following response:

```
DEVICE AND UFD SPECIFICATION(S)? DB0:[*,*],DB1:[*,*]
```

REORDR orders all UFDs on RP04 unit 0 followed by all UFDs on RP04 unit 1. The disks must be logically mounted with write access allowed.

To prevent a damaged directory structure, REORDR requires that all files in a UFD must be closed during the entire reordering process.

NOTE

REORDR verifies that no file is open when it begins the process. However, once the reordering begins, REORDR cannot detect subsequent user file operations. Because REORDR can damage a disk file structure if a file operation takes place during reordering, the system manager should take the following precautions:

1. disable logins with NO LOGINS UTILITY command and
2. ensure that no other jobs (spooler, for example) are active on the system.

The system manager can run the program from an indirect command file using the INIT system program. This approach ensures that no users are able to log into the system and access a file while the disk is being reordered.

The following is a sample indirect command file:

```
DETACH
LOGIN KBO: [1,2]
FORCE KBO: RUN $REORDR
FORCE KBO: YES
FORCE KBO: CRE
FORCE KBO: FOR
FORCE KBO: DB0:[*,*],DB1:[*,*]
FORCE KBO: ^Z
ATTACH
```

The CTRL/Z in the file is the response to the sort question which REORDR prints upon completing the current requests for the devices and accounts specified.

NOTE

For clarity, DIGITAL recommends the use of explicit answers rather than defaults in the indirect command file. The above command will not reorder the account on the disk on which the indirect command file is open.

7.7.3 Processing and Error Messages

During the reordering of a UFD, REORDR holds the reordered UFD in a temporary file. At the conclusion of the ordering process, REORDR copies the contents of the temporary file back to the original UFD.

As each UFD is processed, REORDR generates messages giving the results of the attempted UFD reorder. The format of the messages is as follows:

DIRECTORY dev:[m,n] text

Table 7-8 lists and explains the possible text in these messages.

Table 7-8: REORDR Message Text

Text and Meaning
CANNOT BE REORDERED Prints when the system manager tries to reorder accounts [0,1] and [1,1] on a disk. Both these UFDs contain open files during time sharing and cannot be reordered.
DOES NOT EXIST The account specified cannot be found on the device.
HAS BEEN REORDERED Indicates UFD has been successfully reordered.
IS NOT REORDERED - FILE name.ext IS OPEN Failure message for the UFD. The file access count was nonzero. This condition indicates that a user is accessing the UFD. REORDR leaves the UFD intact. REORDR cannot detect all types of UFD access during its operation.
IS NULL The account exists but has no files to reorder.

If the program encounters a fatal error, it prints a message in the following format:

?REORDR FATAL ERROR (e) text AT LINE m

Where e is the error variable, text is the RSTS/E error message, and m is the line number in the REORDR program.

Fatal errors should be reported to DIGITAL by the SPR mechanism.

7.7.4 Sample REORDR Dialogue and Messages

```
RUN $REORDR

REORDR V7.0 RSTS V7.0 TIME SHARING
DIRECTORY REORDERING PROGRAM

SORT DIRECTORY(S) (YES/NO) <NO>? YES

ORDER BY CRE[ATION] OR ACC[ESS] DATE<CRE>?

IN FOR[WARD] OR REV[ERSE] ORDER<FOR>? REV

DEVICE AND UFD SPECIFICATION(S)? DB1:[*,*],DB2:[*,*]
DIRECTORY DB1:[0,1] CANNOT BE REORDERED
DIRECTORY DB1:[1,1] CANNOT BE REORDERED
DIRECTORY DB1:[1,202] HAS BEEN REORDERED
DIRECTORY DB1:[1,3] HAS BEEN REORDERED
DIRECTORY DB1:[1,4] HAS BEEN REORDERED
DIRECTORY DB1:[120,50] HAS BEEN REORDERED
DIRECTORY DB1:[1,13] HAS BEEN REORDERED
DIRECTORY DB1:[1,200] HAS BEEN REORDERED
DIRECTORY DB1:[1,201] HAS BEEN REORDERED
DIRECTORY DB1:[1,203] HAS BEEN REORDERED
DIRECTORY DB2:[1,28] HAS BEEN REORDERED
DIRECTORY DB2:[1,8] HAS BEEN REORDERED
DIRECTORY DB2:[1,44] IS NULL
DIRECTORY DB2:[120,54] IS NULL
DIRECTORY DB2:[251,0] HAS BEEN REORDERED
DIRECTORY DB2:[100,100] HAS BEEN REORDERED
DIRECTORY DB2:[2,240] HAS BEEN REORDERED
DIRECTORY DB2:[210,203] HAS BEEN REORDERED
DIRECTORY DB2:[232,13] HAS BEEN REORDERED
DIRECTORY DB2:[1,77] IS NULL
DIRECTORY DB2:[26,12] HAS BEEN REORDERED
DIRECTORY DB2:[232,15] HAS BEEN REORDERED
DIRECTORY DB2:[2,227] HAS BEEN REORDERED

SORT DIRECTORY(S) (YES/NO) <NO>? ^Z

READY
```

7.8 Processing User Comments — GRIPE

The RSTS/E system includes a program to allow users of the system to communicate comments to the system manager. Comments are entered, under the control of the GRIPE system program, to a common file named GRIPE.TXT. GRIPE.TXT, which retains the user comments for inspection by the system manager, is created, expanded, and deleted on an as-needed basis under the system library account [1,2]. As an aid in identifying the user

who entered the comment, the GRIPE system program uses a name item supplied in the individual user's account information in the ACCT.SYS file, also stored in the system library account [1,2]. However, the name item and entry for the account in the ACCT.SYS file are not required for GRIPE to run. See Section 4.1.3 for the ACCT.SYS file format.

The system manager or a privileged user invokes the GRIPE system program in the same manner as the general user. (GRIPE is also described in the *RSTS/E System User's Guide*.) Once GRIPE prints the YES? question line, the system manager can then examine the contents of GRIPE.TXT or can clear the contents of GRIPE.TXT. Exit from the GRIPE program to system command level by pressing the ESCAPE key or by typing CTRL/Z.

The *LIST command is used in the following manner to examine the contents of the GRIPE.TXT file.

```
RUN $GRIPE
GRIPE V7.0 RSTS V7.0 TIME SHARING
YES? (PRESS THE 'ESCAPE' KEY TO END)
*LIST ☐ OUTPUT? LP:

READY
```

The system manager types *LIST and the ESCAPE or ALT MODE key immediately after the query line. (Typing the ESCAPE or ALT MODE key on a separate line causes *LIST to be entered as text into the GRIPE.TXT file.) If the GRIPE.TXT file is empty, the message NO GRIPES FOUND is printed, followed by the READY message. Otherwise, the GRIPE program requests an output device on which to list the contents of the GRIPE.TXT file. The system manager can type the RETURN key to have the comments listed at the keyboard printer or can type a device designator, such as LP: shown in the example above. The output for each user comment in the GRIPE.TXT file consists of an identification line (including the account entering the comment, the date and time it was entered, and an account name taken from the ACCT.SYS file) and the text of the comment. The program run is automatically terminated upon completion of the output. Control is returned to the system command level. This action is signalled by printing of the READY message or the appropriate prompt for non-BASIC-PLUS Run-Time Systems.

The system manager clears the contents of the GRIPE.TXT file by using the *RESET command after invoking GRIPE. The following example demonstrates the use of *RESET.

```
RUN $GRIPE
GRIPE V7.0 RSTS V7.0 TIME SHARING
Yes? (PRESS THE 'ESCAPE' KEY TO END)
*RESET ☐

READY
```

The system manager must type *RESET immediately followed by the ESCAPE or ALT MODE key. The clearing of the GRIPE.TXT file and termination of the GRIPE run is signalled by the READY message.

7.9 Communicating With Other Terminals — TALK

A user can communicate with the system console terminal or with another user's terminal by using the system program TALK. The program has a protection code of <232>, permitting all users to run it. The system manager can change the protection code to <124> so only privileged users can run it.

7.9.1 Sending a Message to Another Terminal

TALK enables users to broadcast messages, line by line, to each other's terminals. Both the sending and the receiving terminal must be on-line, but no user need be logged in on the receiving terminal.

TALK is called as follows:

```
RUN $TALK  
TALK V7.0 RSTS V7.0 TIME SHARING
```

The first question line printed is:

```
TO WHICH KEYBOARD (KB)?
```

Type the number of the keyboard to which the message is to be sent, then type the RETURN key.

NOTE

As a general rule, do not broadcast to a terminal in use before consulting with that terminal's operator. An unscheduled message can disrupt an elaborate printout, sometimes destroying hours of work.

Typing the number of a nonexistent or off-line terminal does not return an error message. In the case of a nonexistent terminal, the program terminates when the RETURN key is typed. Messages sent to an off-line terminal, however, are broadcast normally, but never received. In this case, typing the RETURN key does not terminate the program.

TALK prints the following instructions:

```
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE  
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
```

To send a message, type the RETURN key after typing each line. Characters are not transmitted until the RETURN key is typed, at which point the entire line is broadcast. Since each line of the message is broadcast immediately, no prompting characters are printed on the sending terminal between the lines of the message.

The ALT or ESCAPE key, used to terminate the program, can be typed after the last RETURN key or instead of it. This key prints an ESC character (echoed as \$) on the sending terminal. When typed instead of the RETURN key, the ALT or ESCAPE key also prints an ESC character on the receiving terminal. When this happens, the system does not perform a carriage return operation on the receiving terminal.

Shown below is a typical sending terminal dialog between keyboard 12 (sender) and keyboard 9 (receiver).

```
RUN $TALK
TALK V7.0 RSTS V7.0 TIME SHARING

TO WHICH KEYBOARD (KB)? 9
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
GEORGE

PLEASE MOUNT MY DECTAPE WHEN YOU GET A CHANCE.

                                THANKS...

                                SHELDON

$
READY
```

The receiving terminal's printout automatically identifies the sending device by enclosing it in asterisks. No ESC character is printed. The actual message is printed next to the identification, as follows:

```
** KB12 ** GEORGE
** KB12 **
** KB12 **      PLEASE MOUNT MY DECTAPE WHEN YOU GET A CHANCE.
** KB12 **
** KB12 **                                THANKS...
** KB12 **
** KB12 **                                SHELDON
$
```

If at this point, keyboard 9 returns a message, it is done as follows:

```
RUN $TALK
TALK V7.0 RSTS V7.0 TIME SHARING

TO WHICH KEYBOARD (KB)? 12
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
OKAY, SHELDON...

YOUR DECTAPE IS MOUNTED ON UNIT #2

SORRY FOR THE DELAY!

                                GEORGE$

READY
```

Notice that the ESC key was typed instead of the final RETURN key in the above example. This prints the ESC character on both the sending and receiving terminals (see below).

```
** KB9 ** OKAY, SHELDON...  
** KB9 **  
** KB9 ** YOUR DECTAPE IS MOUNTED ON UNIT # 2.  
** KB9 ** SORRY FOR THE DELAY!  
** KB9 **  
** KB9 **                                GEORGE$
```

Chapter 8

The BACKUP System Package

The RSTS/E BACKUP Package consists of programs that allow you to (1) create a disk or magnetic tape copy of files for off-line storage and (2) return the files to the system when necessary.

As you read this chapter keep in mind that:

- 'BACKUP', printed in all upper case letters, refers to the entire package; 'Backup', with initial capital only, is the name of a particular BACKUP mode.
- The BACKUP program does not preserve disk files larger than 65,535 blocks. If you try to backup a disk that contains such a file, BACKUP notifies you of the error, identifies the large file, and then continues the transfer operation. Use the SAVE/RESTORE Package, described in Chapter 9, or PIP.SAV to make an off-line copy of a large disk file.

8.1 BACKUP and System Management

To perform the operations in the BACKUP system package, BACKUP provides you with four operational modes:

- Backup, to create magnetic tape or disk copies of system files from system disks.
- Restore, to rebuild Backup-created files onto a RSTS disk.
- Loadindex, to copy the primary index file from the last volume of the Backup Set to a RSTS/E formatted disk.
- List, to print directory information contained in a Backup index file.

Because each installation is unique, the frequency with which the system manager should backup the system varies. Frequency of backup depends on how the system is used. For example, the system manager may backup the files daily on a system that contains large frequently updated files, but may only perform a backup weekly on a system that is less volatile.

By regularly backing up the system, the manager protects users from losing hours of work if the system crashes, if disk errors occur, or if the users themselves inadvertently destroy information. However, BACKUP does little to preserve files if the backup medium is error prone. The manager should use disks or tapes of reliable quality for BACKUP operations. In addition backup copies must be stored to protect them from unauthorized users and from a catastrophe that might destroy the system itself.

8.2 How BACKUP Works

Important operations within the BACKUP package can be described under these headings:

- dialogue with the user
- selection of files
- entering of accounts (optional - Restore only)
- transfer of files to/from backup medium
- file comparison and deletion (optional)
- building the listing file

The sections that follow outline the functions that BACKUP performs within each of these categories.

8.2.1 Dialogue

The dialogue begins when you invoke BACKUP. The resulting BACKUP question asks you which of the four operational modes you want to perform and allows you to create an indirect command file. It also asks where to create the work-file into which it writes directory and error information. BACKUP creates the work-file on the disk you specify, then asks additional questions to find out which files and accounts to transfer. When the dialogue ends, the program writes a summary of your commands into the listing file.

8.2.2 File Selection

Next, BACKUP searches for all the files and accounts you specified during the dialogue phase. In Backup mode, the program searches for accounts in the Master File Directory (MFD) of the source disk. After finding an account, it searches for files in the User File Directory (UFD) for that account. Backup then sorts the chosen accounts into ascending numerical order. Backup does not sort files within accounts, but selects them in the order in which it finds them in the UFD.

In Restore mode, the program looks up files in one of three index files, which are directories of all the tape or disk volumes of the Backup Set. Backup writes a primary index file and a secondary index file as the last two files on the Backup Set. The third copy remains in your account and is called the auxiliary index file. Because BACKUP assigns a temporary extension of .TMP to this file, to retain it you must copy it (with PIP) to a work-file after the Backup operation is completed or rename the auxiliary index file (with an extension other than .TMP) during the Backup dialogue.

As it makes the selections, BACKUP writes information about each selected file and account into the work-file. Therefore, the work-file can become quite large as the selection phase proceeds. The size of the work-file depends on the number and size of files selected for transfer. The system manager can specify during the dialogue that the work-file be placed on a private disk to ensure that enough space is available for it.

To estimate the size of the work-file for Backup and Restore, you should consider the following guidelines. The work-file has a fixed overhead of two to four blocks. Each file you specify requires one-eighth of a block. Each exception also requires one-eighth of a block. (Exceptions are described in Section 8.3.1.) Errors and file attribute blocks require an additional one-eighth of a block each. Therefore, for a system of 512 accounts with an average of 16 files each, no exceptions, and 16 files with attributes, you can expect the work-file to contain at least 1094 disk blocks. DIGITAL recommends that the system manager add approximately 10% as error space when backing up a large system. In this example, then, the total is 1204 blocks.

8.2.3 Entering Accounts (Optional – Restore only)

The Backup Set may contain accounts that do not exist on the disk (destination disk) you are restoring. If you want to reproduce these accounts on the destination disk, Restore asks you a dialogue question which allows you to perform this operation. When you have completed the dialogue, Restore first creates the selected accounts which do not exist on the destination disk, then transfers the files as described in Section 8.2.4. Accounts recreated on the destination disk retain the same password, quota, and cluster size but not the same account names. Furthermore, the ENTER ACCOUNTS <YES> question does not allow you to selectively create accounts. That is, Restore either creates all the selected accounts on the destination disk or creates none at all. If you want all accounts reproduced, accept the default; otherwise, type NO. Finally, Restore allows only users that are logged into a privileged account to enter accounts on restored disks. It will not ask the ENTER ACCOUNTS <YES> question when operating under a nonprivileged account.

8.2.4 File Transfer

When the selection process is complete, BACKUP begins to transfer the selected files and accounts to the disk or magnetic tape. On a system that has multiple disks in the public structure, Backup transfers all files stored under one account from the entire public structure before proceeding to the next

account. Restore transfers files in the order in which it finds them on the Backup Set. When BACKUP needs a new volume or encounters an error, it prints a prompt on the job's terminal or (if the job is DETACHED) on the operator services console. BACKUP also logs all errors in the listing file.

8.2.5 File Comparison and Deletion (Optional)

After it copies the files and accounts, BACKUP compares the original files with their duplicates on the Backup Set (if you requested a comparison). It records, in the listing file and on the job's console terminal, any differences between the two files.

A Backup operation then deletes files if so requested. Backup deletes only the files you named for deletion and does not delete any files that generated errors during transfer or comparison. Furthermore, Backup does not delete files from accounts [1,2], [1,1], or [0,1].

8.2.6 Building the Listing File

Finally, Backup and Restore generate the remaining listing file text. BACKUP lists each file that it selected for transfer, whether or not it transferred that file. (For example, BACKUP can select a zero-length file for transfer, but it does not transfer the file's data.) The generation of the listing file can be time consuming because BACKUP copies ASCII data from the workfile to the listing file (which can be output on a terminal or written to disk). The listing file can grow quite large for a large system backup or restore. If the listing file consumes all free disk space, you can abort the BACKUP run. (The ABORT command is described in Section 8.3.4.) To abort the run while BACKUP is generating the listing file does not harm the Backup Set or the restored files in any way. It merely leaves you with an incomplete record of the Backup or Restore.

8.3 Running BACKUP

To invoke BACKUP, type RUN \$BACKUP. This command runs BACKUP, which prints the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question. To run in one of the four BACKUP modes from an indirect command file, type one of the following in response to the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question, substituting the name of the indirect command file for filnam.ext.

```
BAC @filnam.ext
RES @filnam.ext
LOA @filnam.ext
LIS @filnam.ext
```

The defaults for indirect command file name are BACKUP.CMD, RESTOR.CMD, LOADIN.CMD, and LIST.CMD. You should note that, to use the assignable account specifier, two commercial 'at' signs are necessary, as in:

```
BAC @MYBACK.CMD
```

To create an indirect command file, append the /SAVE (or /SA) option in response to the BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T] question. The following example illustrates that procedure and BACKUP's response:

```
BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? BAC/SAVE
INDIRECT FILE NAME<SY:[ 1,213]BACKUP.CMD> ?
```

In specifying a file other than the default, you can designate a disk or DECTape as the output device. After you answer the question, BACKUP continues to prompt normally, writes your responses into the indirect command file, and carries out the commands.

8.3.1 File Specification

Several questions in the dialogue require you to respond with a file specification. The file specification lists criteria that files must meet for BACKUP to process them. The criteria can include filename, account, and dates of creation and last access. In addition, you can specify one or more files as exceptions from a process.

The file specification has the following form:

filename/keyword:comparison:date:time/exception (filename . . .)

The filename is a standard RSTS/E filename specification and can contain the wildcard characters * and ?. The keyword is CREATION or ACCESS. The comparison is BEFORE or AFTER. The date is a date string in DD-MMM-YY format. The YY element defaults to the current year, if none is specified. The meanings of keyword and comparison depend on the "milestone" date supplied later in the specification string. The keyword tells BACKUP whether the date represents file CREATION or ACCESS. The comparison tells BACKUP to process files created or accessed BEFORE or AFTER the date. Table 8-1 summarizes the file specification string.

Separate creation and access date comparisons from each other and from the filename (even if it is null) by a slash (/). A file specification applied to the default filename might be as follows:

```
/CR:BEF:01-SEP-78
```

The preceding specification designates any files named by the default that were created before 1-SEP-78 (that is, 31-AUG-78 or earlier).

Table 8-1: BACKUP File Specification

Element	Full name and Meaning
filename	[proj,prog] filename.ext Process the file(s) specified. Standard wild cards are allowed.
keyword	CR[EATION] Interpret date as a milestone creation date. AC[CESS] Interpret date as a milestone last access date.
comparison	BEF[ORE] Process only the file(s) created or accessed before the milestone date. AFT[ER] Process only the file(s) created or accessed after the milestone date.
date	DD-MMM-YY. Use this date (depending on keyword and comparison) as the milestone before or after which the file(s) must have been created or accessed in order to be processed.
time	HH:MM. Use to refine date, with CR only. (Time is optional, and given in 24-hr. format; default is 00:01 of date.)
exception	EXC[EPT] Exclude the file(s) in the following specification from the process. An EXC phrase can include all the previous elements—but not another EXC phrase. Multiple filenames in an EXC phrase must be separated by commas, and the list of excepted filenames must be enclosed in parentheses.

A colon (:) separates the components of the specification. You can include only one CREATION comparison and one ACCESS comparison in each file specification.

Another file specification might be as follows:

```
*.BAC/AC:BEF:06-JUN-78/CR:BEF:25-APR-78
```

This specification designates all files with the extension BAC (on the default account) that were created before 25-APR-78 and last accessed before 06-JUN-78.

In addition to specifying a date, you can designate the time of day (in 24 hour format) in the CREATION comparison only. The following specification designates all files with name PLANTZ that have been created since 21:13 (9:13 P.M.) on 07-MAR-78.

```
PLANTZ,*/CR:AFT:07-MAR-78:21:13
```


You can exempt one or more files from a comparison or operation by including an EXCEPT phrase in the file specification. EXCEPT has two forms, as follows:

1. EXCEPT : file specification
2. EXCEPT : (list of file specifications)

Use the first form if you need to exclude only one file specification and if the excluded specification does not include a CREATION or ACCESS comparison. For example, the following command line designates all files on account [100,250] (except SOCIO.TMP) that were created after 01-JAN-78.

```
[100,250]*.*/CR:AFT:01-JAN-78/EXC:SOCIO.TMP
```

To exclude multiple file specifications or to include a CREATION or ACCESS comparison in an exception, use the second form. You must enclose the files and their comparisons (if any) in parentheses, as in the following example:

```
*.*/CR:BEF:29-FEB-78/EXC:(*.RNO/CR:AFT:01-FEB-78,MT?./AC:AFT: 31-DEC-77)
```

This specification designates all files on the default account that were created before 29-FEB-78. The exceptions are as follows:

1. All files with extension .RNO that were created after 01-FEB-78
2. Any file whose name is three characters long, begins with MT, and has been accessed since 31-DEC-77

A CREATION or ACCESS comparison applies to the files in an EXCEPT comparison as well. To exclude an EXCEPT comparison from CREATION and ACCESS comparisons, you must specify a CREATION or ACCESS date for the exception (and, therefore, enclose the exception in parentheses). A file specification can include only one EXCEPT comparison. The EXCEPT comparison must follow any CREATION or ACCESS date comparisons.

A list of file specifications often encompasses several lines. Therefore, BACKUP accepts a hyphen (-) as the last character before a line terminator to indicate that the next line is a continuation of the current line. In response to the hyphen and line terminator, BACKUP prompts with CONT> and a tab. The specification must include all normal punctuation in addition to the hyphen. BACKUP performs no syntax or semantic processing on a response until you have entered all continuation lines. The following example illustrates a continuation line.

```
FROM FILES<[1,213]*.*>?BACK.RNO,CHAP3.RNO,TEST.BAS,-  
CONT>TAPE.BAS,[2,213]PRINT.BAC
```

BACKUP ignores blanks as separators in specification lines, but considers them significant as terminators. For example, EXC EPT:BEE.BAS is illegal

because the space terminates the scan after the characters EXC. On the other hand, in EXCEPT : BEE.BAS, BACKUP ignores the blanks and processes the text as EXCEPT:BEE.BAS.

You can include a comment in a command line by denoting the beginning of the comment with an exclamation point. To avoid confusion, BACKUP accepts the ! [1,3] account specifier only in quotes ("!" or '! '). In addition, the hyphen denoting a continuation line must precede the comment, as in the following example.

```
FROM FILES<[1,213]*.*>? CHAP2.RND,BACK.CMD,TAPE.BAS-!THIS IS A COMMENT  
CONT>
```

8.3.2 Running BACKUP under BATCH

The BATCH Processing Program can facilitate a BACKUP operation by allowing you to perform a backup without any dialogue interaction. This is possible if you create and then execute a batch control file that contains not only the standard BATCH commands but also the three BACKUP items listed here:

1. The backup control file must contain, RUN \$BACKUP, the BACKUP run command.
2. The backup control file must include the responses to all the dialogue questions either as an explicit list within the file or as a list in the indirect command file which is specified in the first dialogue question response. The /SA[VE] option that creates the indirect command file is discussed in the introduction to Section 8.3.
3. The backup control file must hold the responses to all the mount dialogue questions.

Refer to the chapter on BATCH Processing in the *RSTS/E System User's Guide* for the procedures you must follow to create and run a batch control file.

In reference to item 3), note that you must know the number of devices necessary for a BACKUP operation before submitting the batch job. If you do not know this number, it is best if you include too many devices rather than too few. Also, you must allocate a different drive for each device you specify. In other words, you need to ensure that there are as many drives of the device type you are using as there are volumes in the Backup Set.

In addition, you must include a response for every mount question that might be posted, such as the MOUNT IT ANYWAY question which only appears if a BACKUP volume's expiration date has not passed or if a magnetic tape cannot be read. The rule is: anticipate all mount questions and include an appropriate response. As an added option for the privileged user, you can bypass all magnetic tape label checking by specifying the /SCRATCH switch in the DEVICE question. Otherwise, account for every possible mount prompt by including an appropriate response in the batch control file.

8.3.3 The BACKUP Dialogue

The BACKUP dialogue allows you to specify the files and accounts to back up, restore, compare to their original versions, and delete. This section describes the parts of the dialogue that differ for privileged and nonprivileged users. Tables 8-2 through 8-5 summarize BACKUP's questions and the user's responses. For a complete description of the dialogue, refer to the *RSTS/E System User's Guide*.

During the dialogue, Backup asks the same questions of privileged and nonprivileged users. Restore asks the privileged user the following additional question:

ENTER ACCOUNTS <YES>?

If you answer NO, Restore does not add any accounts to the destination disk.

If you accept the default, Restore creates, on the destination disk, all accounts that are on the Backup Set but not already on the destination disk. Accounts retain all characteristics (password, quota, clustersize) except account name when Restore creates them. If you specify an account in FROM FILE(S) <[CUR ACT] *.*>? and that account does not exist on the destination disk, Restore transfers the account only if you answer YES to ENTER ACCOUNTS.

Privileged and nonprivileged users achieve slightly different results by accepting the default to Restore's TO DISK <__SY:[*,*]> question. Because a nonprivileged user can create files only under the current account, the entire Backup Set is restored, protection codes permitting, to the account by accepting the default. For a privileged user, the default restores all files to the accounts from which they were backed up.

Table 8-2: Backup Dialogue Summary

Question, Response and Meaning
1. BAC[KUP], RES[TORE], LOA[DINDEX], or LIS[T]? BAC[KUP] (/SA[VE]) Backs up files. If you append /SA[VE], Backup creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME <_SY:[CUR ACT] BACKUP.CMD>? File spec Creates an indirect command file with the specified name. Valid output devices are disk and DECtape. Asked only if the /SAVE option is appended in question 1.

(continued on next page)

Table 8-2: Backup Dialogue Summary (Cont.)

Question, Response and Meaning	
2. WORK-FILE NAME <_SY:[CUR ACT]Bddmmm,Jnn>?	
File spec	Uses the specified file as the work-file. If the default is accepted, Backup uses either Bddmmm or Bymmdd as the file-name format. If the system date format, selected during system generation, is ALPHABETIC, the default name format is Bddmmm, where dd is the day of the month and mmm is the alphabetic month abbreviation. If it is NUMERIC, Bymmdd is the default name format. The letter y represents the last digit of the current year, mm is the number of the month, and dd is the day of the month. In the extension Jnn, nn represents the job number under which BACKUP is running. Any mounted, write-enabled disk can be substituted for SY:. The user can create the auxiliary index file by renaming the work-file in this response.
3. LISTING FILE <_KB:>?	
KBn: or LPn:	Writes the listing file to the keyboard or line printer unit specified.
File spec	Writes the listing file to the specified file on disk or DECTape. The default file name is SY: [CUR ACT] BACKUP.LST.
4. FROM DISK <_SY:>?	
Disk name	Backs up from the specified disk. This disk must remain mounted.
5. FROM FILE(S) <[CUR ACT] *,*>?	
BACKUP File spec(s)	Backs up the specified files.
6. TO DEVICE <_MT:>?	
MT:, MM:, MS:, or disk name	Backs up to the specified medium. (Backup requests the unit number later.)
7. BEGIN AT <[*,*] *,*>?	
BACKUP File spec	Starts the back up with the file specified. The default starts with the first file matching a file specification in question 5. Answer with a single file specification only. No EXCEPT modifiers are allowed.
8. DELETE FILE(S) <NONE>?	
BACKUP File spec(s)	Deletes the specified files after backing up and comparing them.
9. COMPARE FILE(S) <NONE>?	
BACKUP File spec(s)	Compares the specified files to the originals after backing them up.

Table 8-3: Restore Dialogue Summary

Question, Response and Meaning	
1. BACKUP], RESTORE], LOADINDEX], or LIST]? RESTORE] (/SAVE)]	Restores files. If you append /SAVE], Restore creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME<_SY:[CUR ACT]RESTOR.CMD>? File spec	Creates an indirect command file with the specified name. Valid output devices are disk and DECTape. Asked only if the /SAVE option is appended in question 1.
2. WORK-FILE NAME<_SY:[CUR ACT] Bddmmm.Jnn>? File spec	Uses the specified file as the work-file. If the default is accepted, Backup uses either Bddmmm or Bymmdd as the file name format. If the system date format, selected during system generation, is ALPHABETIC, the default name format is Bddmmm, where dd is the day of the month and mmm is the alphabetic month abbreviation. If it is NUMERIC, Bymmdd is the default name format. The letter y represents the last digit of the current year, mm is the number of the month, and dd is the day of the month. In the Jnn extension, nn represents the job number BACKUP is running under. Any mounted, write-enabled disk can be substituted for SY:.
3. LISTING FILE<_KB:>? KBn: or LPn:	Writes the listing file to the specified keyboard or line printer unit.
File spec	Writes the listing file to the specified file on disk or DECTape. The default file name is SY: [CUR ACT] RESTOR.LST.
4. INDEX FILE<PRIMARY>? File spec	Uses the specified file as the index file. You can specify an auxiliary index file or an index file created during a LOADINDEX operation. If you accept the default, Restore uses the primary index file, which is on the final volume of the Backup Set.
5. FROM DEVICE<_MT:>? MT:, MM:, MS:, or disk name	Restores backed up files from the specified medium. (Restore requests the unit number later.)
6. FROM FILE(S) <[CUR ACT]*.*>? BACKUP File spec(s)	Restores the specified files.

(continued on next page)

Table 8-3: Restore Dialogue Summary (Cont.)

Question, Response and Meaning	
7. TO DISK<_SY:[*,*]>?	
Disk name	Restores the specified files to the designated disk.
8. BEGIN AT<[*,*]*,*>?	
BACKUP File spec	Restores starting with the file specified here. The default starts with the first file matching the file specification in question 6. The answer must be a single file specification. No EXCEPT modifiers are allowed.
9. (Privileged only) ENTER ACCOUNTS<YES>?	
YES	Creates, on the destination disk, all accounts on the Backup Set that do not already exist on the destination disk.
NO	Does not create any accounts. The user cannot selectively create accounts.
10. SUPERSEDE<NONE>?	
BACKUP File spec(s)	Overwrites the specified files on the destination disk with the backup versions. If you accept the default, Restore supersedes no files.
11. COMPARE FILE(S)<NONE>?	
BACKUP File spec(s)	Compares the restored files to the backup versions after transfer.

Table 8-4: LOADINDEX Dialogue Summary

Question, Response and Meaning	
1. BACKUP], RES[TORE], LOA[DINDEX], or LIS[T]? LOA[DINDEX]?	Copies index file from the Backup Set to a RSTS/E formatted disk. If you attach /SA[VE] to the LOA[DINDEX] response, BACKUP creates an indirect command file and asks question 1a.
1a. INDIRECT FILE NAME <_SY:[CUR ACT]LOADIN.CMD>?	
File Spec	Creates an indirect command file with the specified name. Valid output devices are disk and DECtape. This question is asked only if the /SA[VE] switch is included in the response to question 1.

(continued on next page)

Table 8-4: LOADINDEX Dialogue Summary (Cont.)

Question, Response and Meaning	
2. WORK-FILE NAME<_SY:[CUR ACT] Bddmmm.Jnn>? File Spec	Uses the specified file as the work file. If the default is accepted, LOA[DINDEX] uses either Bddmmm or Bymdd as the file name format. The letters dd represent the day of the month, mmm the 3-character month abbreviation, y the last digit of the current year, and mm the number of the month. In the extension Jnn, nn represents the job number BACKUP is running under. Any mounted, write-enabled disk can be substituted for SY:.
3. LISTING FILE<_KB:>? KB: or LP:	Writes the listing file to the keyboard or the line printer. If you press the RETURN or LINE FEED key, BACKUP sends the listing file to the keyboard <KB:>.
File Spec	Writes the listing file to the specified disk or DECTape. If you type SY: then press the CARRIAGE RETURN key, the BACKUP program creates the default listing file [CUR ACT]LOADIN.LST.
4. FROM DEVICE<_MT:>? MT:, MM:, MS:, or Disk Name	Copies the index file from the specified backup set device.
5. TO FILE<_SY:[CUR ACT] [BINDdd.IND]? File Spec	Loads the Backup Set index file into the specified file. You can only copy the index to a disk device. If you do not specify a file name and thus decide to accept the default, the format for the default file name is BINDdd.IND, where dd is the numeric day of the month.

Table 8-5: LIST Dialogue Summary

Question, Response and Meaning	
1. BACKUP], RES[TORE], LOA[DINDEX], or LIS[T]? LIS[T]/SA[VE]	Invokes the LIST dialogue. When you attach the /SA[VE] switch to the LIST response, BACKUP creates an indirect command file after you respond to question 1a.
1a. INDIRECT FILE NAME<_SY:[CUR ACT][LIST .CMD]>? File Spec	Creates an indirect command file with the specified name. Question 1a only appears if the /SA[VE] switch is attached to the LIST response in question 1. Valid output devices are disk and DECTape.

(continued on next page)

Table 8-5: LIST Dialogue Summary (Cont.)

Question, Response and Meaning
<p>2. WORK-FILE NAME<_SY:[CUR ACT]Bddmmm.Jnn>?</p> <p>File Spec</p> <p>Uses the specified file as the work-file. If the default is accepted by pressing the RETURN or LINE FEED key, LIS(T) uses either Bddmmm or Bymmdd as the file name format. The choice depends on the date format of your system. The letters dd represent the day of the month, mmm the 3-character abbreviation of the month, y the last digit of the current year, and mm the number of the month. In the extension Jnn, nn represents the job number BACKUP is running under. Any mounted, write-enabled disk can be substituted for SY:.</p>
<p>3. LISTING FILE<_KB:>?</p> <p>KB: or LP:</p> <p>Writes the backup directory information to the specified device. If you press the RETURN or LINE FEED key, the backup directory is written to your keyboard. Type LP: then press RETURN if you need a line printer listing.</p> <p>File Spec</p> <p>Writes the backup directory information to the specified file. Type SY: then press the RETURN key if you want BACKUP to create the default listing file, [CUR ACT]LIST.LST.</p>
<p>4. INDEX FILE<PRIMARY>?</p> <p>File Spec</p> <p>Extracts the index file from the specified file and writes it to the listing file. If you accept the default by pressing the RETURN or LINE FEED key, BACKUP proceeds to request the index volume. It then takes the directory information from the index volume and writes the information to the listing file. BACKUP asks the next question, FROM DEVICE<MT:>, if you accept the default in question 4.</p>
<p>5. FROM DEVICE<_MT:>?</p> <p>File Spec</p> <p>Asked only if the default is accepted in the INDEX FILE<PRIMARY> question. The file specification response must identify the device on which the index resides. You can, however, accept the magnetic tape default <MT:> by pressing the RETURN or LINE FEED key. After you respond to this question, the BACKUP program begins a mounting dialogue which asks you to mount the Backup Set index volume and identify the target device.</p>

8.3.4 Interruption Commands

The BACKUP package includes interruption commands for the convenience of the user. These commands enable you to detach, terminate, suspend, and continue processing and to check the status of the run.

BACKUP indicates its readiness to accept interruption commands by printing an asterisk on the job's console terminal after the dialogue is complete. You can type an interruption command as a response to a request in the mount procedure or any time after the asterisk appears. Table 8-6 describes the interruption commands.

Table 8-6: Interruption Commands

Command and Meaning
ABORT, ABO Terminates processing immediately and returns to your default run-time system.
CONTINUE, CON Continues processing after a PAUSE command.
DETACH, DET Detaches job from KB:. BACKUP detaches if 4 conditions are met: (1) OPSER is running, (2) the job has permanent privilege, (3) the listing file is not on KB:, and (4) no messages are currently pending for this copy of BACKUP. BACKUP is given a receiver identification of BACKnn where nn is its job number. All interaction with BACKUP is done through OPSER. (See Chapter 5 for OPSER description.)
END Terminates after processing the current file.
LAST, LAS Reprints, on KB:, the last text printed. If the program is awaiting a response, reprints the question or prompt. If not, reprints the last message.
LEGAL, LEG Prints, on KB:, a list of the commands and responses that are legal now. **OTHER** in the list means that a request for information from you (backup set name, for instance) is pending. The response to such a request is any appropriate string. No quotes need be used unless the response might be construed as an interruption command. For example, you can legally name the Backup Set PAUSE, but you must type the name in quotes ("PAUSE" or 'PAUSE').

(continued on next page)

Table 8-6: Interruption Commands (Cont.)

Command and Meaning
NOTICE, NOT Writes, in the listing file, the specified text. The format of the notice command is NOTICE message where message indicates one line of text. No line terminators are allowed in the message.
OFFLINE, OFF Same function as abort.
PAUSE, PAU Suspends execution until you type CONTINUE. Any legal command typed during a PAUSE is executed immediately.
STATUS, STA Prints, on KB:, BACKUP status information. The information printed varies by phase and appears in the examples in Sections 8.5, 8.6, 8.7, and 8.8.
TERMINATE, TER Closes the current volume at the end of the current file or at the end of the current output volume, whichever comes first.

8.3.5 Mounting and Dismounting Volumes

After BACKUP selects all files and accounts for transfer, it requests backup set labelling information and the device unit number for the first backup volume. It also prints a volume identification summary.

The labelling information BACKUP requests is the name and expiration date of the Backup Set. BACKUP uses the name as an identifier for the Backup Set. It interprets the expiration date as the date after which it can automatically write over the data on the volume. The default expiration date is one year from the current date. If you mount a volume for backup before its expiration date, BACKUP asks for confirmation before writing on the volume. For magnetic tapes, BACKUP also requests density (800 or 1600 BPI) and, for 800 BPI tapes, parity.

When backing up the system onto magnetic tape, you have the option to bypass all tape label checking. Backup will mount the magnetic tape volume with no label checking when you specify the /SCRATCH switch in response to the DEVICE question. For example:

```
DEVICE? MM1: /SCRATCH
```

After answering the labeling questions, mount the volume and either write-enable it for a Backup operation or write-lock it for a Restore operation.* In response to the DEVICE question, type the full device specification (device mnemonic and unit number followed by a colon) of the volume that is mounted.

NOTE

The BACKUP interruption commands and the BACKUP requests RET(RY), SKI(P), and IGN(ORE) will be executed during the mount dialogue if any of your responses to any of the questions have as its first three characters the first three characters of a command or request. To avoid this situation, either

- a) ensure that the response does not have the same first three characters as the commands and requests, or,
- b) enclose the response in quotes.

The following example shows the mount procedure print-out. Press the LINE FEED key to indicate the default response.

```
PLEASE ENTER BACKUP SET NAME<BACK28> - (LF)
PLEASE ENTER EXPIRATION DATE<08-Jul-77> - (LF)
PLEASE ENTER DENSITY IN BPI<800> - 800
PLEASE ENTER THE PARITY<ODD> - (LF)
MOUNT   DEVICE:      _MT      :
        ID:          BACK28
        SEQ#:         1
        DENSITY:      800 BPI
        PARITY:       ODD
        IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
DEVICE? MM1:
THIS VOLUME HAS NO BACKUP LABEL!
MOUNT IT ANYWAY<NO>? Y
```

BACKUP attempts to read the mounted tape at all valid density and parity settings (unless you have specified /SCRATCH) to determine whether there is a backup label on the tape. If no information is on the tape (that is, it is a new tape), this process can take several minutes and will cause magnetic tape errors to be written to the system error log. DIGITAL therefore recommends that you initialize new tapes with PIP using the /ZE switch before use with BACKUP. The tape will always be written at the requested density and parity settings.

When BACKUP has finished using a volume, it prints a dismount message like the following:

```
DISMOUNT DEVICE:      _MM1:
        ID:          BACK28
        SEQ#:         1
        DENSITY:      800 BPI
        PARITY:       ODD
        PLEASE LABEL THIS VOLUME!
```

*The disks from which BACKUP restores must be kept write-enabled because bad block files on disk must be updated as required.

When the dismount message appears, physically dismount the volume (if necessary) and ready the next volume for processing. BACKUP requests name, expiration date, parity, and density for only the first volume in each Backup Set. For subsequent volumes, BACKUP requires only the device specification.

8.3.6 Writing the BACKUP Structure on Disks

Each backup disk must contain the special BACKUP structure before the package can copy files to that disk. The BACDSK program, which is part of the BACKUP package, writes this format on initialized RSTS/E disks and rewrites this format on BACKUP disks to update bad block information. BACDSK runs automatically as part of the Backup process for privileged users. The system manager (or privileged user) must, however, run the BACDSK program once to write the initial BACKUP format on disks that will be used by nonprivileged users. The following is an example of its use:

```
RUN $BACDSK
BACDSK V7.0 RSTS V7.0 TIME SHARING
DEVICE? DKO:
PROJ,PROG? 1,227
```

READY

In response to the DEVICE question, type the device name and unit number (followed by a colon) of the nonprivileged user's disk. Answer the PROJ,PROG question with the user's project-programmer number. BACDSK then writes the BACKUP format on the disk. The BACKUP structure remains on the disk through subsequent Backup and Restore operations. Note, however, that all Backup disks must be reinitialized with the RSTS/E file structure before they can return to normal RSTS/E file-structured use.

8.4 BACKUP Error Handling

Four types of errors are possible with BACKUP: dialogue command errors, interruption command errors, volume mount errors, and BACKUP processing errors.

Dialogue command errors occur during the Backup, Restore, Loadindex, or List dialogue. The BACKUP package diagnoses these errors immediately. Usually, you can respond to such an error by typing the correct dialogue answer.

Interruption command errors can occur when you type an interruption command. BACKUP responds immediately to these errors, too. Simply type a valid command to correct the error situation.

Volume mount errors can occur when the BACKUP package mounts tape and disk backup volumes. Some volume mount errors are user errors (for example, specifying an illegal density setting for tape); others involve hardware (for example, tape errors that prevent labelling).

BACKUP processing errors can occur any time during the BACKUP run except during the dialogue. Processing errors can be due to either the hardware on which the BACKUP is running or to logic errors.

The following four sections describe each type of error.

8.4.1 Dialogue Command Errors

When BACKUP encounters a dialogue command error (usually a syntax error), it prints this error message on your terminal: ?COMMAND ERROR. The error message is followed by either a BACKUP-specific error message or a RSTS/E error message generated during the syntax processing of the command. After printing the error message, BACKUP prints a question mark and repeats its prompt. If you respond by typing the question mark (?) and the RETURN key, BACKUP prints the command line up to the position of the error, then reprints the prompt. Table 8-7 describes the BACKUP dialogue error messages.

Table 8-7: BACKUP Dialogue Error Messages

Message and Meaning
BAD DIRECTORY FOR DEVICE The directory structure on the device you specified is corrupt. Try to place the file on a different device.
CAN'T FIND FILE OR ACCOUNT BACKUP cannot find the file or account you specified. Ensure that the file name and account were typed correctly, and that they exist.
DEVICE HUNG OR WRITE-LOCKED The device you specified is hung, write-locked, or has generated a read or write error. Ensure that the device is ready, write-enable the device (if necessary) or specify a different device.
DEVICE NOT AVAILABLE The device you specified is disabled or assigned to another user. Specify a different device.
DUPLICATE SWITCH The file specification contains multiple ACCESS or CREATION comparisons. BACKUP accepts only one of each comparison in a file specification.
ILLEGAL EXCEPT NESTING The file specification includes more than one EXCEPT comparison. BACKUP accepts only one EXCEPT in a file specification.
ILLEGAL FIELD The response contains a field that is not permitted. For example, you cannot specify a device name in the response to the FROM FILES question.

(continued on next page)

Table 8-7: BACKUP Dialogue Error Messages (Cont.)

Message and Meaning	
ILLEGAL FILENAME	The filename you specified contains invalid characters or is in incorrect format.
ILLEGAL KEYWORD	The response contains a misspelled or incorrectly abbreviated keyword, or has incorrect punctuation marks.
ILLEGAL OPERAND	The EXCEPT comparison in the file specification contains an unmatched parenthesis, or the day or year number in a date is out of range.
INCOMPLETE COMMAND FILE	You typed CTRL/Z or the indirect command file ended before the expected end-of-file. BACKUP returns to the BACKUP, RESTORE, LOAD-INDEX, or LIST question.
NO DEFAULT	The current dialogue question does not have a default answer. You must type an explicit response.
NOT A VALID DEVICE	The device you specified is not on this system.
NOT A VALID DEVICE (PROHIBITED)	The device you specified is illegal in response to this question. Refer to Table 8-2, 8-3, 8-4, or 8-5 to ascertain the valid devices for each question.
PROTECTION VIOLATION	The file or account you specified is protected.
TOO MANY FILES OPEN ON UNIT	Only one file at a time can be open on a magnetic tape unit. Specify the second file on another device.
TOO MUCH DATA	The response contains more data than BACKUP accepts for this question.
UNIT NUMBER NOT VALID	BACKUP does not accept a device unit number in the response to this question.

8.4.2 Interruption Command Errors

An invalid interruption command generates one of three error messages. Table 8-8 summarizes these messages and their meanings.

Table 8-8: Interruption Command Error Messages

Text and Meaning
CAN'T DETACH The DETACH command is invalid because one or more of the 4 conditions described in Table 8-6 is not true.
UNRECOGNIZED COMMAND You typed a string that is not an interruption command.
ILLEGAL COMMAND You typed an interruption command that is illegal (for example, typing CONT when no PAUSE is in effect). Type LEGAL for a list of interruption commands that are currently legal.

8.4.3 Volume Mount Errors

Several errors can occur during the volume mount process. BACKUP prints an error message, then repeats the current prompt. Table 8-9 summarizes these error messages.

Table 8-9: BACKUP Volume Mount Error Messages

Message and Meaning
?DENSITY/PARITY CANNOT BE SET ON THIS DRIVE Specified parity or density is illegal on this type of hardware.
?DISK MUST BE INITIALIZED A disk has been mounted, but lacks BACKUP file structure format. Any disk to be used by BACKUP must be initialized by DSKINT. The disk must not be logically mounted or opened by any user.
?INVALID BACKUP SET NAME The set name contains illegal characters.
?INVALID DATE Expiration date is not in DD-MMM-YY format or has passed.
?INVALID DENSITY SETTING Valid density settings are 800 and 1600 bpi.
?INVALID DEVICE OR NO UNIT NUMBER The input device specification refers to a device not configured on the current system; or the device unit number is not specified.

(continued on next page)

Table 8-9: BACKUP Volume Mount Error Messages (Cont.)

Message and Meaning
<p>?INVALID PARITY SETTING Valid parity settings are ODD and EVEN.</p>
<p>?INVALID SWITCH SPECIFICATION The switch does not contain (as its first three characters) SCR.</p>
<p>?ILLEGAL SWITCH USAGE The /SCRATCH switch was specified during a RESTORE, was specified for a RESTORE, was specified for a disk device, or was specified by a non-privileged user.</p>
<p>?MAGTAPE NOT WRITE LOCKED During a RESTORE operation the magnetic tape must be mounted write-locked.</p>
<p>?MAGTAPE SELECT ERROR The magnetic tape unit specified is not READY or is OFFLINE. Ready the unit or specify another unit.</p>
<p>?MAGTAPE WRITE LOCKED During a BACKUP operation the magnetic tape must be mounted write-enabled.</p>
<p>?THIS IS NOT THE RIGHT VOLUME 1. Some volume data does not match with what BACKUP expected. 2. The user is not privileged or is not the owner of the volume.</p>
<p>?YOU ARE NOT PRIVILEGED TO USE THIS VOLUME The volume's expiration date has not yet passed, and the user is nonprivileged (and not the volume's owner).</p>
<p>?UNABLE TO WRITE ON THIS TAPE BACKUP cannot write a label on the magnetic tape because of tape errors. Try another magnetic tape.</p>

8.4.4 BACKUP Processing Errors

BACKUP processing errors can occur as the package selects, transfers, compares and deletes files and generates the listing file. A processing error can be a hardware error, which indicates a hardware problem (for example, ?DEVICE HUNG OR WRITE-LOCKED), or a logic error, which indicates an attempt at an illogical or prohibited action (for example, ?PROTECTION VIOLATION). The BACKUP package contains error handling routines for common hardware and logic errors.

8.4.4.1 Selection Errors — If a logic error occurs during the selection process, BACKUP automatically skips over the file or account causing the error. BACKUP prints an error message, then proceeds to select the next file or account, according to the following rules:

1. If an error occurs while BACKUP is looking up a file, BACKUP skips that file on that account and looks for the next sequential file.
2. If an error occurs during the search for an account, BACKUP terminates the search on that input volume and continues the search for that account on the next input volume.
3. If an error occurs during the search for an account on the final input volume, BACKUP returns to the first input volume and searches for the next sequential account.
4. If an error occurs in the final account on the final input volume, the selection process ends and BACKUP begins to transfer the selected files and accounts.

BACKUP contains error handling routines that allow you to correct or circumvent certain problems that cause the ?DEVICE HUNG OR WRITE-LOCKED hardware error. When this error occurs, BACKUP prints the error message followed by one or more valid request commands (RETRY or SKIP) set in parentheses. After BACKUP issues an asterisk prompt, you can respond to it with one of the suggested commands. A RETRY response instructs BACKUP to attempt the failed operation again. For example, device hung errors may simply indicate a transient hardware problem, that a device has gone off line, or that a file or account contains bad blocks. To rectify a transient hardware problem may only require that you use the RETRY command. Problems of this type may not occur during a subsequent retry attempt. Also, when a device goes off line you can usually place the device back on line and respond to the asterisk prompt by typing RETRY. Finally, if bad blocks in a file or account cause a device hung error, you can attempt to bypass the bad block area by using the Skip command. As a result of your response BACKUP will end its search of the file or account containing the bad block, proceed to look for the the next file or account, and then attempt to resume normal processing. Using the RETRY and SKIP request commands may correct otherwise irretrievable situations and thereby save you hours of work.

When BACKUP discovers a disk file larger than 65,535 blocks, it displays the following message and then continues the operation.

LARGE FILE <filespec> - WILL NOT BE BACKED UP

The filespec field includes the device, account, filename, and file extension.

The Backup operation transfers placed files to the Backup Set but does not preserve placement. If BACKUP encounters any placed files during processing, it prints a warning message, transfers the file, and ignores the file's placement. The message that may appear during a Backup run has the following form:

```
PLACED FILE <filespec> SELECTED - PLACEMENT WILL BE LOST UPON  
RESTORATION!
```

During a Restore, the message takes the following form:

```
PLACED FILE <filespec> SELECTED - PLACEMENT HAS BEEN LOST!
```

The filespec field includes the device type, account number, file name, and file extension. Both placed file messages are sent to the keyboard as well as to the listing file.

If a bad block error (?USER DATA ERROR ON DEVICE) occurs during the selection phase of a Backup operation, Backup prints an error message, stops searching for the current file or account, and looks for the next file or account as for a logic error.

During a RESTORE operation, a bad block error at index load time usually means that the index file is corrupt. If you specified that the primary index file was to be used, the following message is issued:

```
PRIMARY INDEX LOAD FAILED! STARTING SECONDARY LOAD
```

BACKUP will attempt to load the secondary index file (which may require that the user mount a different volume).

If an error occurs during the secondary index load, the message:

```
SECONDARY INDEX LOAD FAILED. RUN ABORTED!!!
```

is issued and BACKUP aborts. You must rerun BACKUP using the auxiliary index file.

If you specified that the auxiliary index file was to be used and a load error occurs, the message:

```
AUXILIARY INDEX LOAD FAILED - filename.ext DUE TO <text> ERROR.  
RUN ABORTED!!!
```

is issued and BACKUP aborts. The <text> is the RSTS/E error that caused the load failure.

A bad block error (?USER DATA ERROR ON DEVICE) that occurs in the Backup or Restore work-file is fatal. BACKUP prints ?UNEXPECTED ERROR — USER DATA ERROR ON DEVICE and aborts the run.

8.4.4.2 Transfer, Deletion, and Listing Errors — Logic errors during the transfer, deletion, and listing processes are usually the result of failure to open a file that BACKUP must transfer or delete. For example, the logic error ?PROTECTION VIOLATION means that you are prohibited by the protection code from deleting or transferring a certain file. Another logic error, SUPERSEDE FAILURE, indicates a failure to restore a file which already exists on the destination disk. This error can occur even if you did not request a supersede operation. The BACKUP package automatically skips over the file or account causing a logic error during transfer, deletion, or listing processes.

BACKUP error handling routines allow you to RETRY or SKIP over a ?DEVICE HUNG OR WRITE-LOCKED error that occurs during the transfer, delete, or listing process. RETRY and SKIP work in the same way here as they do during the selection process.

In addition to device hung errors, BACKUP can be interrupted by errors such as, '?CAN'T FIND FILE OR ACCOUNT' or '?NO ROOM FOR USER ON DEVICE'. If one of these errors occurs while BACKUP is processing a listing file, type the IGNORE request command in response to the asterisk prompt BACKUP provides you. This procedure forces BACKUP to disable future I/O to the listing file and to continue with the current operation.

The SEQUENCING PROBLEM ON MAGTAPE error can occur during a Restore from magnetic tape. This error means that Restore read a magnetic tape which didn't have the expected record number. When the sequencing error occurs, Restore usually prints several bad block error messages (as it tries and fails to space to the correct magnetic tape record), then recovers automatically. You cannot restore the file in which the sequencing error occurred, but you can restore the remainder of the files on the tape.

If BACKUP encounters a bad block during a transfer operation that involves a RSTS/E (not a Backup) disk, BACKUP prints an error message. It copies the rest of the current file although the file is corrupt. If the bad block is in the work-file, the BACKUP run is aborted.

The BACKUP package specially structures each backup disk when you mount it for a Backup operation. BACKUP creates a bad block file on the disk and allocates to this file all bad blocks it finds during formatting. The BACKUP package does not permit any disk that has an excessive number of bad blocks to be used as a backup disk. When a Restore operation discovers a bad block on a backup disk, it allocates the bad block to the bad block file. This procedure prevents future use of the block for backup data. (The system manager should be aware that BACKUP bad block files are not equivalent to the RSTS/E file, BADB.SYS. The system manager must reinitialize a backup disk before using it as a RSTS/E file-structured disk. Section 7.5 describes disk initialization procedures.)

8.4.4.3 Informational Messages — During the transfer process, BACKUP often prints informational messages. These messages usually do not indicate errors, but inform you about files that are open (and subject to change) and files that have changed in length since selection. BACKUP transfers these

files, but the copies may not be accurate. Other messages list files that BACKUP cannot transfer: those deleted since selection and those whose length is zero.

The BACKUP program displays a timed reminder after you type the PAUSE command and when it is waiting for a user response. Two minutes after you type the PAUSE command, which suspends the execution of a BACKUP operation, BACKUP prints the message, (IN PAUSE). The reminder appears only once. Similarly, BACKUP reminds you, after a two minute interval, that it is waiting for a user response. The message, (FURTHER RESPONSE NECESSARY), will not appear during any of the initial dialogue questions. Like the PAUSE reminder, this message is not repeated.

8.5 Backing Up System Files – Example

In the Backup mode you can create magnetic tape or disk copies of system files. The destination medium, to which you transfer the files, is called the Backup Set. The example that appears in Section 8.5.1 demonstrates the procedures to follow when you need to create a magnetic tape Backup Set from RK06 disk files. Responses to the dialogue questions are explained following the Backup example. Also, the listing file that results from the Backup operation is shown in Section 8.5.2. A description of the characteristics of the Backup listing file follows that example.

8.5.1 Terminal Printout – Backup

To duplicate the example that follows, log onto your system, invoke the BACKUP program, then answer the dialogue questions as follows:

```
RUN $BACKUP
BACKUP  V07.0 RSTS V07.0 TIME SHARING

a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? BAC
b  WORK FILE NAME<_SY:[ 1,110]B08SEP.J16> ? BACKUP.WKF/MO:256
c  LISTING FILE<_KB:> ? BACKUP.LST
d  FROM DISK<_SY:> ? DM1:
e  FROM FILES<[ 1,110]*.*> ? [* ,110]*.*/EXC:[1,110]*.BAK
f  TO DEVICE<_MT:> ?
g  BEGIN AT[* ,*]*.*> ?
h  DELETE FILES<NONE> ? [2,110]*.DAT
i  COMPARE FILES<NONE> ? [2,110]*.DAT
*
j  PLEASE ENTER BACKUP SET NAME<BACKUP> -
   PLEASE ENTER EXPIRATION DATE<08-Sep-79> -
   PLEASE ENTER DENSITY IN BPI<800> -
   PLEASE ENTER THE PARITY<ODD> -
MOUNT  DEVICE:      _MT      :
        ID:         BACKUP
        SEQ#:       1
        DENSITY:    800 BPI
        PARITY:     ODD
IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
```

```

k  DEVICE? MM0:
l  EXPIRATION DATE HAS NOT YET ARRIVED!
    ID:          BACKUP
    SEQ#:        1
m   DENSITY:     800 BPI
    PARITY:      ODD
    EXPIRATION DATE: 08-Sep-79
n  MOUNT IT ANYWAY<NO>?Y

*
DATA UNRELIABLE - FILE OPENED BY ANOTHER USER IN FILE
o  _DM1 :[1,110]ADDR.DAT
   (ON TRANSFER)
p  *STATUS

q  PHASE      : TRANSFER
   VOLUME #   : 1
   ACCOUNTS   : 1
   FILES      : 2
   BLOCKS     : 2010
   ERRORS     : 1

   CURRENT VOLUME : _MM0      :
r  CURRENT ACCOUNT : _DM1     :[1,110]
   CURRENT FILE   : _DM1     :[1,110]RT11 .RTS<60>

   ELAPSED TIME  : 82 SECONDS
s  CPU TIME      : 1.8 SECONDS
   KCTS          : 284

*
t  NO MORE ROOM ON VOLUME - TERMINATING.
*
DISMOUNT DEVICE:      _MM0      :
    ID:              BACKUP
    SEQ#:            1
    DENSITY:         800 BPI
    PARITY:          ODD
    EXPIRATION DATE: 08-Sep-79
    PLEASE LABEL THIS VOLUME!

MOUNT   DEVICE:      _MM      :
    ID:              BACKUP
u   SEQ#:            2
    DENSITY:         800 BPI
    PARITY:          ODD
    IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
v  DEVICE? MM1:/SCR
*
w  DISMOUNT DEVICE:      _MM1      :
    ID:              BACKUP
    SEQ#:            2(INDEX)
    DENSITY:         800 BPI
    PARITY:          ODD
    EXPIRATION DATE: 08-Sep-79
    PLEASE LABEL THIS VOLUME!

*
Ready

```

The letters that prefix the explanations that follow are keyed to identifying letters located within the previous Backup example. Follow the lettered steps through these sequence of steps:

- a. Select the Backup mode by typing BAC, then press the RETURN key. To acknowledge which operation you want to select, BACKUP must receive at least the first three letters of the mode. In the initial BACKUP prompt, the square brackets ([]) printed with each mode illustrate this requirement.
- b. Create the work file BACKUP.WKF. The /MO:256 switch, attached to the work-file name, enables data caching for the BACKUP.WRF file. For an explanation of data caching modes refer to the *RSTS/E Programming Manual*.
- c. Specify BACKUP.LST as the file location from which BACKUP prints the listing file. You usually print the listing file after completing a BACKUP operation to check the integrity of the run. Section 8.5.2 reproduces and explains the information contained in the listing file for this Backup example.
- d. Type DM1: to instruct BACKUP to copy files from an RK06. If the files you want to back up are on a system disk (SY:), you can accept the default.
- e. Specify the backup of all files in programmer account 110 except those in account [1,110] that have a .BAK extension.
- f. Transfer all files to a magtape Backup Set by accepting the MT: default. Thus, all the files specified in the FROM DISK<__SY:> question will be transferred to magnetic tape.
- g. Begin the transfer operation with the first file on the input device (RK06) by pressing the RETURN key.
- h. Delete all files in account [2,110] that have .DAT extensions. The wildcard designator, represented by the asterisk symbol, tells BACKUP that you want any filename in account [2,110] with a .DAT extension deleted.
- i. Compare the specified input files to the same files that have been transferred to the magnetic tape Backup Set. Note that BACKUP compares the [2,110]*.DAT files before it deletes them from the input volume. BACKUP then prints an asterisk that indicates the beginning of the selection process and Backup's readiness to accept interruption commands. After printing the asterisk, BACKUP selects files for transfer according to your specifications. When the selection process ends, Backup asks you additional questions at j.
- j. Accept the defaults to the next four questions. Backup always asks questions regarding the Backup Set and Expiration Date; however, the Density and Parity questions are printed only when the Backup Set is magnetic tape. After you respond to the last question, BACKUP prints a summary of the tape label information.

- k. Mount a magnetic tape to which Backup can copy files; type the mnemonic name and unit number of the device on which the tape is mounted, then press the RETURN key. Subsequently, Backup searches the magnetic tape for a valid label and finds one.
- l. Notice that as a result of the magnetic tape label inspection Backup discovered the expiration date had not arrived. The message indicates that the date you last placed data on the tape has not expired and thus the tape may still contain valuable information.
- m. Notice that Backup prints a summary of the label information it found. In this summary, Backup includes the expiration date.
- n. Override the default and use the magnetic tape despite Backup's warning. The data on the magnetic tape is no longer needed.
- o. Notice the error message. As it encounters errors in the transfer process, Backup prints messages at the job's console terminal. This error message warns you of open files which may have changed between selection and transfer. Any errors that occur during a BACKUP run will also appear in the listing file.
- p. Type the STATUS interruption command in response to the asterisk prompt. Backup responds by printing the current status of the operation.
- q. Note the information that the status report contains. You,
 - 1. Are in the Backup transfer PHASE.
 - 2. Continue to work on the first volume of the Backup Set.
 - 3. Transferred 1 account.
 - 4. Transferred 1 file.
 - 5. Copied 2010 blocks.
 - 6. Encountered 1 error.
- r. Notice that the summary identifies the Backup Set medium, the account, and file that is currently being transferred.
- s. Observe that BACKUP prints the elapsed time, the CPU time, and the number of kilo-core ticks used so far in the current phase. After printing the status report, Backup displays the asterisk prompt to indicate that it is ready to continue with the selection process.
- t. Note that Backup now requires the use of a second volume, since the first Backup Set volume is full. After printing another asterisk prompt, Backup displays label information for the magnetic tape medium you need to dismount.
- u. Notice that Backup assumes the same label characteristics for the second volume as it assumed for the first. Immediately after the label summary, Backup asks you to mount the second magnetic tape volume.

- v. Specify the mnemonic name and unit number of the device on which the tape is mounted. Attach the /SCR switch so that Backup will avoid all label checking. Backup resumes transfer processing, indicated by the appearance of the asterisk prompt.
- w. Notice that Backup issues a summary of the label information on the second volume of the Backup Set after the Backup operation is complete. You return to the control of your default run-time system.

8.5.2 Listing File - Backup

The Backup listing file summarizes the Backup dialogue, provides statistics on each of the Backup phases, and lists the operations performed on each selected file. The following is the listing file from the Backup example in the preceding section.

```

                                Backup Run Listing

Backup from '_DM1' to '_MT' :
Run started on 08-Sep-78 at 11:33 AM
Work-File is _SY :[1,110]BACKUP.WKF/MD: 256
a Transfer : [* ,110]?????.???/EXC:([1,110]?????.BAK)
Begin at : [* ,*]*.*
Delete : [2,110]?????.DAT
Compare : [2,110]?????.DAT

PHASE : LIST COMPLETE
ERRORS : 0
b ELAPSED TIME : 2 SECONDS
CPU TIME : .6 SECONDS
KCTS : 87

c NO FILES FOR ACCOUNT [3,110] ON DISK _DM1 :

PHASE : LOOKUP/SELECT COMPLETE
VOLUME # : 1
ACCOUNTS : 2
d FILES : 11
BLOCKS : 11589
ERRORS : 0

ELAPSED TIME : 11 SECONDS
CPU TIME : 1.3 SECONDS
KCTS : 208

PHASE : MOUNT / DISMOUNT COMPLETE
ERRORS : 0
e ELAPSED TIME : 82 SECONDS
CPU TIME : 1.4 SECONDS
KCTS : 224

```



```

ZERO LENGTH FILE - _DM1 :[1,110]OPEN ,TMP<60> - ONLY
DIRECTORY ENTRY TRANSFERRED.
f DATA UNRELIABLE - FILE OPENED BY ANOTHER USER IN FILE
_DM1 :[1,110]ADDR ,DAT
(ON TRANSFER)
g NO MORE ROOM ON VOLUME - TERMINATING.

PHASE : TRANSFER COMPLETE
VOLUME # : 1
ACCOUNTS : 2
FILES : 10
BLOCKS : 9493
ERRORS : 1

h CURRENT VOLUME : _MM0 :
CURRENT ACCOUNT : _DM1 :[2,110]
CURRENT FILE : _DM1 :[2,110]MASTER,DAT<60>

ELAPSED TIME : 290 SECONDS
CPU TIME : 11.9 SECONDS
KCTS : 1916

PHASE : COMPARE COMPLETE
VOLUME # : 1
ACCOUNTS : 1
FILES : 1
BLOCKS : 6904
ERRORS : 0

i CURRENT VOLUME : _MM0 :
CURRENT FILE : _DM1 :[1,1]

ELAPSED TIME : 368 SECONDS
CPU TIME : 27.4 SECONDS
KCTS : 4377

PHASE : MOUNT / DISMOUNT COMPLETE
ERRORS : 0

j ELAPSED TIME : 84 SECONDS
CPU TIME : 1.3 SECONDS
KCTS : 240

PHASE : TRANSFER COMPLETE
VOLUME # : 2
ACCOUNTS : 1
FILES : 1
BLOCKS : 2096
ERRORS : 0

k CURRENT VOLUME : _MM1 :

ELAPSED TIME : 48 SECONDS
CPU TIME : 2.2 SECONDS
KCTS : 348

PHASE : COMPARE COMPLETE
VOLUME # : 2
ACCOUNTS : 1
FILES : 1
BLOCKS : 2096
ERRORS : 0

l CURRENT VOLUME : _MM1 :
CURRENT FILE : _DM1 :[1,1]

```

ELAPSED TIME : 75 SECONDS
CPU TIME : 8.5 SECONDS
KCTS : 1356

PHASE : INDEX DUMP COMPLETE
ERRORS : 0

m CURRENT VOLUME : _MM1 :

ELAPSED TIME : 3 SECONDS
CPU TIME : .7 SECONDS
KCTS : 108

PHASE : MOUNT / DISMOUNT COMPLETE
ERRORS : 0

n ELAPSED TIME : 13 SECONDS
CPU TIME : .3 SECONDS
KCTS : 48

PHASE : DELETION COMPLETE
VOLUME # : 1
ACCOUNTS : 1
FILES : 1
o BLOCKS : 9000
ERRORS : 0

ELAPSED TIME : 1 SECONDS
CPU TIME : .5 SECONDS
KCTS : 60

p Backup Set Name : BACKUP Backup Device : _MM :
Volume Sequence # : 1 Owner : [1,110]
Creation Date : 08-Sep-78 Expiration Date : 08-Sep-79
Density : 800 BPI Parity : ODD

q Account : [1,110] Quota : 0
Clustersize : 4

	Name	Ext	Size	Prot	Creation Date	Time	Access Date	Clu	RTS	TSCD
r	SAVE	.DOC	10	60	10-Jul-78	05:01 PM	10-Jul-78	8	RSX	*T
s	OPEN	.TMP	0	60	08-Sep-78	10:46 AM	08-Sep-78	4	BAS4F	
	ADDR	.DAT	2000	60	08-Sep-78	10:46 AM	08-Sep-78	4	BAS4F	T

Errors :
DATA UNRELIABLE - FILE OPENED BY ANOTHER USER on FILE
Total of 1 error encountered on FILE

RT11	.RTS	20	60	31-Aug-78	04:39 PM	31-Aug-78	4	RSTS	T
SAVE	.RND	9	60	10-Jul-78	04:59 PM	10-Jul-78	8	TECO	T
UTILITY	.BAS	47	60	23-Nov-77	10:44 AM	23-Nov-77	8	BASIC	T
DIRECT	.BAS	56	60	21-Jul-78	11:24 AM	21-Jul-78	8	BAS4F	T

* Attributes associated with this file

Account total of 2142 blocks in 7 files on account [1,110]
Total of 1 error encountered on ACCOUNT

Account : [2,110] Quota : 0
Clustersize : 4

Name	Ext	Size	Prot	Creation		Access		Clu	RTS	TSCD
				Date	Time	Date				
BACKUP.BAS		121	42	23-Aug-78	09:56 AM	23-Aug-78		8	BASIC	T
BACFRM.BAS		158	42	15-Aug-78	04:49 PM	15-Aug-78		8	BASIC	T
BACKTO.BAS		168	42	15-Aug-78	04:36 PM	15-Aug-78		8	BASIC	T
t MASTER.DAT		6904	60	08-Sep-78	10:55 AM	08-Sep-78		4	BAS4F	T C

(of 9000)

u ***** File continued on next Volume *****

Account total of 7351 blocks in 4 files on account [2,110]
 ***** Account continued on next Volume *****

v Volume total of 9493 blocks in 11 files in 2 accounts on volume MM :
 Total of 1 error encountered on VOLUME

w Backup Set Name : BACKUP Backup Device : _MM :
 Volume Sequence # : 2(Index) Owner : [1,110]
 Creation Date : 08-Sep-78 Expiration Date : 08-Sep-79
 Density : 800 BPI Parity : ODD

Account : [2,110] Quota : 0
 Clustersize : 4

Name	Ext	Size	Prot	Creation		Access		Clu	RTS	TSCD
				Date	Time	Date				
MASTER.DAT		2096	60	08-Sep-78	10:55 AM	08-Sep-78		4	BAS4F	T CD

(of 9000)

Account total of 2096 blocks in 1 files on account [2,110]

Volume total of 2096 blocks in 1 files in 1 accounts on volume _MM :

x RUN total of 11589 blocks in 11 files on 2 accounts on 2 Volumes
 RUN total of 1 errors
 PHASE : LIST COMPLETE
 VOLUME # : 2
 ACCOUNTS : 2
 FILES : 11
 y BLOCKS : 11589
 ERRORS : 0
 ELAPSED TIME : 2 SECONDS
 CPU TIME : 1.7 SECONDS
 KCTS : 252

The Backup listing file:

- Summarizes the Backup dialogue, indicating which operations are to be performed on each selected file. Backup writes this summary into the listing file immediately after executing the dialogue.
- Displays accounting information about the listing process that recorded the dialogue summary. After the dialogue summary is written, the selection begins.

- c. Indicates that Backup found no files for account [3,110] and therefore has not accepted that account for transfer.
- d. Summarizes the selection phase which lists the total number of files, accounts, and disk blocks selected for transfer. The total number of errors, zero for this Backup, that occurred during the selection phase is also listed. The final part of the selection summary lists accounting information.
- e. Records the magnetic tape mounting process and its statistics. When the magnetic tape is mounted, the transfer begins.
- f. Lists messages from the transfer phase. The ZERO LENGTH FILE message informs you that [1,110]OPEN.TMP is a zero-length file which Backup does not transfer. However, Backup will transfer the directories for the file. One DATA UNRELIABLE error occurred for the file [1,110]ADDR.DAT. The error was printed at the job's terminal during the run, as noted in Section 8.5.1, and is listed in the summary at h.
- g. Prints the NO MORE ROOM ON VOLUME message, which appeared during the BACKUP operation and indicates volume #1 of the Backup Set is full.
- h. Summarizes the transfer phase.
- i. Summarizes the compare phase.
- j. Displays information concerning the dismount process for volume #1. Since there are two volumes in the Backup Set, the listing file also summarizes in the next four steps the transfer, compare, the index dump, the mount/dismount process, and the deletion process for the second volume.
- k. Summarizes the transfer phase for the second volume of the Backup Set.
- l. Summarizes the compare phase.
- m. Displays statistics for the index dump phase.
- n. Prints mount/dismount for volume #2.
- o. Summarizes the deletion phase. All of the summaries, h through o, have time statistics.
- p. Displays the label information for volume #1, then reports directory information for each selected file and account for that volume.
- q. Lists the account number, quota, and the cluster size of [1,110], the selected account. For each selected file in the account, the listing file displays the file name, size in decimal blocks, protection code, creation date and time, last access date, the file cluster size, and the associated run-time system. The final column, labelled TSCD, contains a T if the file was transferred, an S if it was superseded (possible only in a Restore operation), a C if it was compared, and D if it was deleted (possible only in a Backup operation).
- r. Transfers file SAVE.DOC and notifies you of the transfer by placing a T in the TSCD column. The asterisk before the T means that there are attributes associated with that file.

- s. Does not process OPEN.TMP, a zero-length file, and indicates this by leaving the TSCD column blank.
- t. Transfers and compares MASTER.DAT. Note that the MASTER.DAT file is continued onto the second volume of the Backup Set.
- u. Notifies you that the MASTER.DAT file is a split file and that it will be continued on the next volume.
- v. Provides a summary of blocks, files, accounts, and errors for that volume after listing directory information for the first volume of the Backup Set.
- w. Gives you directory information for each selected file and account for volume #2. The directory information for volume #2 is in the same format as it is for volume #1.
- x. Prints the total statistics for the Backup run after listing each volume.
- y. Ends with a summary of the statistics for the final listing phase.

8.6 Restoring Files – Example

Restore recreates a RSTS/E formatted disk from a magnetic tape or disk Backup Set. The Restore example in Section 8.6.1 illustrates the procedures to follow when restoring an RK06 disk from a magnetic tape Backup Set. The listing file that resulted from the Restore run in Section 8.6.1 is reproduced and described in Section 8.6.2.

8.6.1 Terminal Printout – Restore

To duplicate the following example, log onto your system, invoke the BACKUP program, and respond to the resulting dialogue questions. The following is a terminal printout of a Restore operation:

```

RUN $BACKUP
BACKUP  V7.0 RSTS V7.0 TIME SHARING

a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[IT] ? RES
b  WORK FILE NAME <_SY:[ 1,110]B08SEP.J16> ? RESTOR.WKF
c  LISTING FILE <_KB:> ? RESTOR.LST
d  INDEX FILE<PRIMARY> ?
e  FROM DEVICE<_MT:> ?
f  FROM FILES<[ 1,110]*,*> ? [2,110]BACKUP.BAS,BACKTO.BAS
g  TO DISK<_SY:[*,*]> ? DM1:
h  BEGIN AT<[*,*]*,*> ?
i  ENTER ACCOUNTS<YES> ? NO
j  SUPERSEDE FILES<NONE> ?

```

```

k  COMPARE FILE<NONE> ?
l  *
m  PLEASE ENTER BACKUP SET NAME<RESTOR> - BACKUP
   PLEASE ENTER DENSITY IN BPI<800> -
   PLEASE ENTER THE PARITY<ODD> -
   MOUNT      DEVICE:      _MT      :
                   ID:      BACKUP
                   SEQ#:    INDEX
                   DENSITY:  800 BPI
                   PARITY:   ODD
n      PLEASE MOUNT VOLUME WRITE LOCKED!
o  DEVICE? MM1:
p  *STATUS

   PHASE      : SELECT
   VOLUME #   : 2
   ERRORS     : 0

   CURRENT VOLUME : _MM      :
   CURRENT ACCOUNT : _DM1    :[2,110]

   ELAPSED TIME  : 1 SECONDS
   CPU TIME      : .4 SECONDS
   KCTS         : 64

*
q  DISMOUNT DEVICE:      _MM1      :
   MOUNT      DEVICE:      _MM      :
                   ID:      BACKUP
                   SEQ#:    1
                   DENSITY:  800 BPI
                   PARITY:   ODD
r      PLEASE MOUNT VOLUME WRITE LOCKED!
s  DEVICE? MM0:
   *
t  DISMOUNT DEVICE:      _MM0      :
   *
   Ready

```

The letters that prefix the explanations that follow are keyed to identifying letters located within the previous Restore example. Follow the lettered steps through these sequence of steps:

- a. Type RES in response to the initial BACKUP prompt, then press the RETURN key. Your specification invokes the Restore program which then issues a set of dialogue questions.
- b. Specify a work-file name with a non-.TMP extension such as RESTORE.WKF. BACKUP deletes any work file created during a Restore run that has a temporary extension of .TMP.
- c. Create the RESTOR.LST file which will contain the listing file generated from this Restore operation. Restore prints the listing file on your keyboard (KB:) if you accept the default.
- d. Press the LINE FEED key to notify Restore that you want the primary index file, located on the last volume of the Backup Set, transferred to the destination disk.
- e. Accept the MT: default since the files you are restoring reside on magnetic tape media.

- f. Specify the files that you want Restore to recreate on the output disk. In this case, Restore will return the two files, [2,110]BACKUP.BAS and BACKTO.BAS, to the RK06 disk.
- g. Identify the disk to which you plan to copy the files contained on the magnetic tape Backup Set. Type DM1: which is the mnemonic name and unit number of the RK06 disk.
- h. Instruct Restore to begin processing the first file it encounters on the Backup Set by accepting the default.
- i. Type NO to ensure that the files that exist on the Backup Set but do not exist on the output disk are not reproduced on the output disk.
- j. Press the LINE FEED key to indicate you want no files superseded. That is, you do not want any files on the Backup Set to overwrite the same files on the output disk.
- k. Accept the default; you do not want any files compared.
- l. Notice that Restore prints an asterisk prompt after you complete the dialogue. It also indicates its readiness to accept interruption commands. Ignore the prompt at this point since you have no reason to interrupt processing.
- m. Type BACKUP to specify the correct name of the Backup Set. After you press the LINE FEED key in response to the next two questions, Restore prints a summary of the label information contained on the Backup Set.
- n. Note that the message instructs you to mount the last volume of the Backup Set, the one containing the primary index file. Mount it without the write-enable ring.
- o. Specify the mnemonic name and unit number of the device on which the last volume of the Backup Set is mounted. Press the RETURN key and Restore reissues the asterisk prompt.
- p. Respond to the asterisk prompt by typing STATUS which summarizes the operational status of the Restore run. The summary indicates that Restore,
 1. Is in the select PHASE.
 2. Is processing the last volume of the Backup Set.
 3. Has encountered no errors.
 4. Is being performed from a magnetic tape Backup Set.
 5. Is processing account [2,110].
 6. Has consumed 1 second of elapsed time, .4 seconds of CPU time, and 64 kilo-core ticks.

After printing the summary, Restore returns to normal processing, as is indicated by the asterisk prompt.
- q. Note the dismount summary that Restore prints when Restore finishes processing the Backup Set volume that contains the primary index file.

- r. Notice that Restore requests that you mount the remaining volume in the Backup Set.
- s. Type the mnemonic name and unit number of the device on which you have mounted the second volume of the Backup Set. After you press the RETURN key, Restore selects from the mounted volume the specified files and transfers them to the output disk.
- t. Note that after Restore prints the dismount message for the second volume, it returns you to the control of your Run-Time System.

8.6.2 Listing File - Restore

The listing file summarizes the phases of the Restore operation and provides directory information on the restored files. The following is the listing file for this Restore run:

```

                                Backup Run Listing

Restore from '_MT   :' to '_DM   :[*,*]
Run started on 08-Sep-78 at 11:57 AM
Work-File is _SY      :[1,110]RESTOR.WKF/M0: 256
a Transfer   : [2,110]BACKUP.BAS,[2,110]BACKTO.BAS
Begin at    : [*,*]*.*
Supersede   : <none>
Compare     : <none>

PHASE       : LIST COMPLETE
ERRORS      : 0
b ELAPSED TIME : 1 SECONDS
  CPU TIME    : .5 SECONDS
  KCTS        : 72

PHASE       : MOUNT / DISMOUNT COMPLETE
ERRORS      : 0
c ELAPSED TIME : 61 SECONDS
  CPU TIME    : .9 SECONDS
  KCTS        : 160

PHASE       : INDEX LOAD COMPLETE
ERRORS      : 0

CURRENT VOLUME : _MM1   :
d ELAPSED TIME : 1 SECONDS
  CPU TIME    : .7 SECONDS
  KCTS        : 107

```



```

e  PHASE      : SELECT COMPLETE
    VOLUME #  : 2
    ERRORS    : 0

    ELAPSED TIME : 2 SECONDS
    CPU TIME     : .8 SECONDS
    KCTS        : 128

    PHASE      : MOUNT / DISMOUNT COMPLETE
    ERRORS     : 0

f  ELAPSED TIME : 24 SECONDS
    CPU TIME     : .8 SECONDS
    KCTS        : 144

    PHASE      : TRANSFER COMPLETE
    VOLUME #    : 1
    ACCOUNTS    : 1
    FILES      : 2
    BLOCKS     : 289
    ERRORS     : 0

g  CURRENT VOLUME : _MMO :
    CURRENT FILE  : _MMO :

    ELAPSED TIME : 50 SECONDS
    CPU TIME     : 1.5 SECONDS
    KCTS        : 254

    PHASE      : MOUNT / DISMOUNT COMPLETE
    ERRORS     : 0

h  ELAPSED TIME : 15 SECONDS
    CPU TIME     : .2 SECONDS
    KCTS        : 32

    Backup Set Name : BACKUP           Backup Device : _MM      :
i  Volume Sequence # : 1               Owner : [1,110]
    Creation Date   : 08-Sep-78        Expiration Date : 08-Sep-79
    Density : 800 BPI                 Parity : ODD

    Account : [2,110]                 Quota : 0
    Clustersize : 4

j  Name Ext Size Prot Date Time Access Date Clu RTS TSCD
    BACKUP.BAS 121 42 23-Aug-78 09:56 AM 23-Aug-78 8 BASIC T
    BACKTO.BAS 168 42 15-Aug-78 04:36 PM 15-Aug-78 8 BASIC T

k  Account total of 289 blocks in 2 files on account [2,110]
l  ***** Account continued on next Volume *****

m  Volume total of 289 blocks in 2 files in 1 accounts on volume _MM :

    Backup Set Name : BACKUP           Backup Device : _MM      :
n  Volume Sequence # : 2(Index)        Owner : [1,110]
    Creation Date   : 08-Sep-78        Expiration Date : 08-Sep-79
    Density : 800 BPI                 Parity : ODD

o  <NO TRANSFER SELECTED>

```

```

p Volume total of 0 blocks in 0 files in 0 accounts on volume _MM :

q RUN total of 289 blocks in 2 files on 0 accounts on 2 Volumes
  RUN total of 0 errors

  PHASE : LIST COMPLETE
  VOLUME # : 1
  ERRORS : 0

r ELAPSED TIME : 3 SECONDS
  CPU TIME : 1.1 SECONDS
  KCTS : 162

```

The Restore listing file:

- a. Summarizes the Restore dialogue which indicates that you are restoring two files in account [2,110], BACKUP.BAS and BACKTO.BAS. In addition, the summary shows that Restore will start processing at the beginning of the file, will not supersede any files, and not compare any files.
- b. Supplies accounting statistics on the writing of the dialogue summary into the listing file.
- c. Summarizes the mount/dismount phase, during which you mounted the backup index volume.
- d. Describes the index load phase.
- e. Summarizes the selection phase statistics.
- f. Indicates that Restore finished selecting data from the backup index volume and requested that it be dismounted. The summary also describes the various times required to complete the dismount operation and mount the other volume in the Backup Set.
- g. Describes the transfer phase which shows that two files selected for transfer were on volume #1.
- h. Displays the mount/dismount summary which is caused by the dismount request which occurs at the end of the transfer.
- i. Lists the label information for volume #1 after displaying each operational phase.
- j. Prints the directory information for each file and account for that volume. This section, like that in the Backup listing, includes the account, quota, and cluster size for each affected account and lists each restored file by name. The block size, protection code, date and time of creation, date of last access, cluster size, and associated run-time system are listed for each file. The TSCD column contains a T for each restored file since you specified that they only be transferred, not superseded (S), compared (C), or deleted (D).

- k. Indicates that a total of 289 blocks in 2 files were transferred for account [2,110].
- l. States in this message that account [2,110] is continued on the second volume.
- m. Displays the block, file, and account summary for the entire volume.
- n. Lists label information for volume #2.
- o. Notifies you that no files or accounts were transferred. The account message, displayed at k, does not appear for volume #2 because no accounts were transferred from volume #2.
- p. Always displays the volume summary information for each volume of the Backup Set even when the totals are zero.
- q. Reports the total statistics for the run.
- r. Supplies statistics that describe the listing phase that generated the listing file.

8.7 Loading the Index File – Example

Loadindex allows you to copy a primary index file from a Backup Set to a RSTS/E formatted disk. To perform the transfer, you must invoke the BACKUP program, type Loadindex in response to the initial BACKUP prompt, and answer the resulting dialogue questions. During the dialogue, you specify the name and location of the Backup Set and the file into which the program is to copy the index. BACKUP then prompts you to mount the Backup Set index file, copies the index when the device is ready, and requests the dismount of the index file volume. Finally, BACKUP prints a listing file that summarizes the Loadindex operation.

You can use the duplicate index file to facilitate other BACKUP operations. For example, it can be advantageous to place a copy of the Backup Set index file on a separate disk. In particular, when you perform a series of restores, you must have access to the index file located on the last volume of the Backup Set. If you have not created an index file with Loadindex, created an auxiliary index file by copying the work file using PIP.SAV, or specified a non-.TMP extension for the work file during the Backup operation, you must repeatedly load the last backup volume for each Restore operation. You can use Loadindex to create an easily accessible index and thus alleviate this inconvenience.

A discussion follows both the Loadindex example in Section 8.7.1 and the Loadindex listing file example in Section 8.7.2.

8.7.1 Terminal Printout - Loadindex

Log onto the system under account [1,110] and invoke BACKUP. The following represents a line printer listing from a Loadindex operation:

```
RUN $BACKUP
BACKUP  V7.0 RSTS V7.0 TIME SHARING

a  BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LOA
b  WORK FILE NAME<_SY:[ 1,110]B08SEP.TMP> ? LOADIN.WKF/MO:256
c  LISTING FILE<_KB:> ? LOADIN.LST
d  FROM DEVICE<_MT:> ?
e  TO FILE<_SY:[ 1,110]BIND08.IND> ? BACIND.IND
f  *
g  PLEASE ENTER BACKUP SET NAME<LOADIN> - BACKUP
h  PLEASE ENTER DENSITY IN BPI<800> -
i  PLEASE ENTER THE PARITY <ODD> -
    MOUNT      DEVICE:      _MT      :
                ID:          BACKUP
j                SEQ#:       INDEX
                DENSITY:     800 BPI
                PARITY:      ODD
k                PLEASE MOUNT VOLUME WRITE LOCKED!
l  DEVICE? MM1:
m  *
n  DISMOUNT DEVICE: _MM1:
o  *
p  Ready
```

Follow this sequence of steps (keyed to the identifying letters) through the example:

- a. Invoke the Loadindex dialogue by typing LOA in response to the initial BACKUP question. A LOADINDEX response, rather than LOA, produces the same results. BACKUP acknowledges only the first three letters of the option.
- b. Type LOADIN.WKF/MO:256 to locate the work file on the public disk structure. To give the specified file name a non-.TMP extension allows you to use, at some later date, the created work file as an auxiliary index file. However, if you accept the default or specify a work file with a .TMP extension, BACKUP deletes the file at the end of the Loadindex run. The MODE 256% switch, attached to your response, enables random data caching which means it facilitates data transfers within the LOADIN.WKF work file. For an explanation of the MODE 256% switch, consult the *RSTS/E Programming Manual*
- c. Specify LOADIN.LST as the file into which BACKUP prints the listing file. The listing file prints on the job's terminal if you accept the default.
- d. Accept the MT: default by pressing the RETURN key, since the index resides on magtape.
- e. Instruct BACKUP to copy the backup index to the file, BACIND.IND.

- f. Ignore the asterisk prompt which indicates that BACKUP is ready to accept interruption commands.
- g. Respond by typing the Backup Set name, BACKUP.
- h. Accept the default density of 800 BPI, since that is the density of the magnetic tape.
- i. Press the RETURN key to accept the ODD parity default setting for the magnetic tape.
- j. Check the magnetic tape label information summary.
- k. Ensure that the magnetic tape is mounted write-locked and is at its load point.
- l. Type MM1: which is the mnemonic name and unit number of device on which the magnetic tape is mounted.
- m. Type a BACKUP interrupt command if you want to temporarily suspend processing at this point. Since the operation is running smoothly, ignore the asterisk prompt. Before BACKUP issues the dismount message at n, it copies the index file to the specified area.
- n. Dismount the Backup index volume by typing the mnemonic name and unit number (followed by a colon) of the device on which the magnetic tape is mounted.
- o. Notice that the Ready message indicates that the Loadindex operation has terminated and you are now under the control of your run-time system.

8.7.2 Listing File – LoadIndex

The Loadindex listing file summarizes the Loadindex dialogue, provides statistics on each phase, and lists the operations performed on each selected file. The following is the listing file from the Loadindex example presented in Section 8.7.1.

```

Backup Run Listing

Loadindex from '_MT      :
Run started on 08-Sep-78 at 12:03 PM

a
Work-File is _SY      :[1,110]LOADIN.WKF/MO: 256
To file      : _SY      :[1,110]BACIND.IND

PHASE      : LIST COMPLETE
ERRORS     : 0

b
ELAPSED TIME : 1 SECONDS
CPU TIME      : .5 SECONDS
KCTS          : 72

```

```

        PHASE      : MOUNT / DISMOUNT COMPLETE
        ERRORS      : 0

c  ELAPSED TIME : 60 SECONDS
    CPU TIME    : .9 SECONDS
    KCTS        : 160

        PHASE      : INDEX LOAD COMPLETE
        ERRORS      : 0

d  CURRENT VOLUME : _MM1      :

    ELAPSED TIME : 1 SECONDS
    CPU TIME     : .3 SECONDS
    KCTS         : 46

        PHASE      : MOUNT / DISMOUNT COMPLETE
        ERRORS      : 0

e  ELAPSED TIME : 12 SECONDS
    CPU TIME    : .2 SECONDS
    KCTS        : 32

```

The Loadindex listing file:

- a. Summarizes the Loadindex dialogue described in Section 8.7.1. The summary shows the date and time of the Loadindex operation, the work-file name, the name of the output file, and that the index resides on magnetic tape.
- b. Provides time and error statistics on the writing of the dialogue summary into the listing file.
- c. Lists the statistical details concerning the mount phase of the Backup Set index volume.
- d. Describes what volume contains the index, the times required to complete the load, and the errors encountered during the index load phase.
- e. Statistically summarizes the dismount of the Backup Set volume.

8.8 Listing the Index File – Example

List initiates a set of dialogue questions like the other BACKUP modes. After you respond to the questions, BACKUP generates a copy of the Backup Set index file, an index created by Loadindex, or, if specified during the Backup operation, an auxiliary index file. These files contain a directory listing of the Backup Set as well as a log of any errors encountered during the Backup operation. Before it can copy the index file, BACKUP asks for the index name and location, then asks where you want the index listed. As options, you can list either the index file on your keyboard (___KB:) or place it in another file. Like the other modes, a summary of the dialogue is written to the listing file.

The example in Section 8.8.1 describes the procedures you follow to generate a list of an auxiliary index file.

8.8.1 Terminal Printout - List

You log onto your system, invoke BACKUP, and then respond to the dialogue questions as follows:

```
RUN $BACKUP
BACKUP V7.0 RSTS V7.0 TIME SHARING

a  BACKUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LIS
b  WORK FILE NAME<_SY:[ 1,110]B0BSEP.TMP> ? LIST.WKF/MO:256
c  LISTING FILE<_KB:> ? LIST.LST
d  INDEX FILE<PRIMARY> ? BACKUP.WKF
e  *
f  Ready
```

Follow this sequence of steps (keyed to the identifying letters) through the example.

- a. Type LIS in response to the initial Backup question, then press the RETURN key. This procedure invokes the List option. Since BACKUP recognizes only the first three characters of each option, you can type either LIS or LIST to invoke the List dialogue.
- b. Specify LIST.WRK as the work-file and attach the MODE 256% switch to optimize file transfer. Refer to the *RSTS/E Programming Manual* for a description of MODE 256%.
- c. Instruct BACKUP to copy the listing file to LIST.LST. If you accept the default, BACKUP prints the index file on your keyboard (_KB:).
- d. Type BACK.WRK to indicate which index file you want BACKUP to copy. In this case, you specify the BACK.WRK file which you created as an auxiliary index file during the Backup operation in Section 8.5.1. Because you included a non-.TMP extension (.WRK) when you specified the work-file name, BACKUP created BACK.WRK as a permanent auxiliary index file. After you complete the dialogue, BACKUP copies the BACK.WRK file to the listing file, LIST.LST. If, however, you accept the default, BACKUP prints the FROM DEVICE<MT:> question. In response, specify the device type on which the primary index volume is mounted. Since BACK.WRK is not the primary index file, you accepted the default and the question did not appear.
- e. Respond to the asterisk (*) prompt by pressing the RETURN key. The asterisk appears after you complete the dialogue; it also indicates that BACKUP is ready to accept interruption commands. After your response, BACKUP copies BACK.WRK to the LIST.LST listing file. The Ready message appears when the List operation ends.

8.8.2 Listing File - List

The List listing file summarizes the List dialogue, provides statistics on the each phase of the operation, and prints a copy of the Backup index file. The example that follows is the listing file for the List operation described in Section 8.8.1.

Backup Run Listing

List

a Run started on 08-Sep-78 at 12:08 PM

Work-File is _SY :[1,110]LIST .WKF/MO: 256

PHASE : LIST COMPLETE
ERRORS : 0

b ELAPSED TIME : 1 SECONDS
CPU TIME : .5 SECONDS
KCTS : 72

PHASE: : INDEX LOAD COMPLETE
ERRORS : 0

c ELAPSED TIME : 2 SECONDS
CPU TIME : .4 SECONDS
KCTS : 55

d Backup Set Name : BACKUP Backup Device : _MM :
Volume Sequence # : 1 Owner : [1,110]
Creation Date : 08-Sep-78 Expiration Date : 08-Sep-79
Density : 800 BPI Parity : ODD

Account : [1,110]
Clustersize : 4

Quota : 0

Name	Ext	Size	Prot	Creation Date	Time	Access Date	Clu	RTS	TSCD
SAVE	.DOC	10	60	10-Jul-78	05:01 PM	10-Jul-78	8	RSX	*T
OPEN	.TMP	0	60	08-Sep-78	10:46 AM	08-Sep-78	4	BAS4F	T
ADDR	.DAT	2000	60	08-Sep-78	10:46 AM	08-Sep-78	4	BAS4F	T

Errors :

e DATA UNRELIABLE - FILE OPENED BY ANOTHER USER on FILE
Total of 1 error encountered on FILE

Name	Ext	Size	Prot	Creation Date	Time	Access Date	Clu	RTS	TSCD
RT11	.RTS	20C	60	31-Aug-78	04:39 PM	31-Aug-78	4	RSTS	T
SAVE	.RNO	9	60	10-Jul-78	04:59 PM	10-Jul-78	8	TECO	T
UTILITY	.BAS	47	60	23-Nov-77	10:44 AM	23-Nov-77	8	BASIC	T
DIRECT	.BAS	56	60	21-Jul-78	11:24 AM	21-Jul-78	8	BAS4F	T

* Attributes associated with this file

Account total of 2142 blocks in 7 files on account [1,110]
Total of 1 error encountered on ACCOUNT

Account : [2,110]
Clustersize : 4

Quota : 0

Name	Ext	Size	Prot	Creation Date	Time	Access Date	Clu	RTS	TSCD
BACKUP	.BAS	121	42	23-Aug-78	09:56 AM	23-Aug-78	8	BASIC	T
BACFRM	.BAS	158	42	15-Aug-78	04:49 PM	15-Aug-78	8	BASIC	T
BACKTO	.BAS	168	42	15-Aug-78	04:36 PM	15-Aug-78	8	BASIC	T
f MASTER	.DAT	6904	60	08-Sep-78	10:55 AM	08-Sep-78	4	BAS4F	T C

(of 9000)

***** File continued on next Volume *****

Account total of 7351 blocks in 4 files on account [2,110]
***** Account continued on next Volume *****

g Volume total of 9493 blocks in 11 files in 2 accounts on volume MM :
 Total of 1 error encountered on VOLUME

h Backup Set Name : BACKUP Backup Device : _MM :
 Volume Sequence # : 2(Index) Owner : [1,110]
 Creation Date : 08-Sep-78 Expiration Date : 08-Sep-79
 Density : 800 BPI Parity : ODD

Account : [2,110] Quota : 0
 Clustersize : 4

	Name	Ext	Size	Prot	Creation Date	Time	Access Date	Clu	RTS	TSCD
i	MASTER.DAT		2096	60	08-Sep-78	10:55 AM	08-Sep-78	4	BAS4F	T CD
	(of 9000)									

Account total of 2096 blocks in 1 files on account [2,110]

j Volume total of 2096 blocks in 1 files in 1 accounts on volume MM :

k RUN total of 11589 blocks in 11 files on 2 accounts on 2 Volumes
 RUN total of 1 errors

PHASE : LIST COMPLETE
 VOLUME # : 2
 1 ACCOUNTS : 2
 FILES : 11
 BLOCKS : 11589
 ERRORS : 0

m ELAPSED TIME : 2 SECONDS
 CPU TIME : 1.7 SECONDS
 KCTS : 252

The List listing file:

- Summarizes the List dialogue which includes the name of the operation, the run date and time, and the work-file name.
- Provides statistics for the listing phase on the writing of the List dialogue summary into the listing file. Notice that Backup rounds off elapsed time to whole seconds and CPU time to tenths of seconds. It is possible, therefore, for the elapsed time to read zero while there is CPU time indicated.
- Summarizes the index load phase which prints the elapsed time, CPU time, and the kilo-core ticks needed to load the index into the listing file.
- Prints the Backup directory information extracted from the index file after it completes the List dialogue summary and reports on the various phases of the List operation. The directory report begins by listing the label information for the first volume of the Backup Set. It then displays a list of the files for each account on the directory. At the end of each account report, BACKUP prints the total number of blocks used by the files listed and the number of errors encountered on the account. In addition to supplying the file name, BACKUP includes in the account report the file block size,

protection code, date and time of creation, date of last access, the cluster size, and the associated run-time system. The TSCD column contains a T for each restored file, an S for each superseded file, a C for a compared file, and a D for each deleted file. An asterisk beside a letter in the TSCD column indicates that the file has attributes associated with it. Zero-length files, such as OPEN.TMP in account [1,110], are not processed and thus no letter is assigned to them in the TSCD column. Note that any errors in the dialogue sequence appear in the index.

- e. Prints the DATA UNRELIABLE message immediately after the file that committed the error. After enumerating the error messages, BACKUP prints the total number of errors detected in the file.
- f. Summarizes the files contained in account [2,110].
- g. Prints at the end of the account report the total number of blocks, files, and accounts on the entire volume.
- h. Prints label information found on the second volume of the backup set.
- i. Prints the account summary for volume #2. If there is more than one volume in the Backup Set, as there is in this example, BACKUP prints account and file information in the same format for the other volumes in the set.
- j. Summarizes the number of blocks, files, and the account that was transferred for the second volume of the set. Since there is only one account report, the volume and account totals are equal.
- k. Displays a summary of the blocks consumed by all the files on both volumes of the backup set. This completes the account and file statistics.
- l. Informs you that there are 2 volumes in the backup set, 2 accounts in the directory, and 11 files in the 2 accounts which require 11,589 blocks of disk space. Also, an item in the list phase report indicates that there were no errors encountered during the printing of the list phase summary.
- m. Prints the times required to complete the list phase.

Chapter 9

SAVE/RESTORE System Program

The RSTS/E SAVE/RESTORE system program is a disk backup and copy utility that provides you with four operational functions. SAVE, RESTORE, and IMAGE perform the backup, recreate, and the copy functions respectively while IDENTIFY extracts label information from a SAVE/RESTORE volume or RSTS/E disk. This chapter describes these four functions, how they are invoked and used.

The following section headings and numbers give you a preliminary understanding of this chapter's organization. If you are unfamiliar with SAVE/RESTORE, read all sections in order. Otherwise, use the section headings to find the specific information you need.

Section Headings	Section Number
When to use SAVE/RESTORE	9.1
Definitions of SAVE/RESTORE Terms	9.2
Running SAVE/RESTORE	9.3
SAVE/RESTORE Options	9.4
SAVE/RESTORE Dialogue	9.5
SAVE/RESTORE and Booting	9.6
Operator Interface during Processing	9.7
SAVE/RESTORE Summary Report.	9.8
SAVE/RESTORE Error Handling	9.9

9.1 When to use SAVE/RESTORE

SAVE/RESTORE and BACKUP perform similar system functions but differ from each other in some objectives. SAVE/RESTORE was developed to provide a non-selective, fast volume backup, and copy capability that requires few operator responses during the operational dialogue. Unlike BACKUP, SAVE/RESTORE processes entire volumes only, and does not allow selective

file transfers, or file deletes. SAVE/RESTORE does not require operator intervention when it encounters bad blocks. For these reasons, use the SAVE/RESTORE package when you need to:

1. Create a fast, reliable copy of an entire RSTS/E disk,
2. Backup files that are larger than 65,535 blocks,
- * 3. Make a fast copy to a disk that contains bad blocks which may not be reflected in the BADB.SYS file, and
4. Make a recovery medium (bootable) for a system disk in case of catastrophic errors.

9.2 Definitions of SAVE/RESTORE Terms

The SAVE/RESTORE terms defined here are used throughout this chapter.

SAVE Format The format of the output written by a SAVE operation and read by a RESTORE operation.

SAVE Set The set of magnetic tapes or disks created by a SAVE operation. A SAVE Set must be composed entirely of disks or tapes, not a combination of the two device types. However, different drive types may be mixed within one set, e.g., two RK05's and one RK06.

SAVE Volume One of the magnetic tapes or disks of a SAVE Set.

SAVE Set Name One to six alphanumeric characters used to identify a SAVE Set. By default, the SAVE Set Name is the same as the Pack ID from which it was created. You can specify another name for the SAVE Set rather than accept the system default.

LIKE Disks Like disks are units of the same device size (i.e., same number of data blocks). For example, SAVE/RESTORE considers the following devices alike:

- a. An RP06 is like another RP06
- b. An RM02 is like an RM03
- c. An RP04 is like an RP05
- d. An RK05 (RK05J) is like one unit of an RK05F

As a further clarification, if you copy an RM02 to a SAVE Set, that set can later be RESTORED to either an RM03 or another RM02.

9.3 Running SAVE/RESTORE

You can invoke the SAVE/RESTORE program off line while you are running the system initialization code (INIT.SYS). To do this, type SA[VRES] on the console terminal in response to the INIT OPTION: prompt. After you press the RETURN key, SAVE/RESTORE code requests the current date and time then responds with its own SAV/RES FUNCTION: prompt. The following example illustrates the proper response to the INIT OPTION: prompt and the appearance of the SAVE/RESTORE program prompt.

```
Option: SAVRES
DD-MMM-YY? 12-JAN-79
HH:MM? 12:15
SAV/RES Function:
```

NOTE

You must run SAVE/RESTORE off line in order to make a copy of your system disk or to restore the system disk.

You can also use SAVE/RESTORE on line, during time sharing. When you do so, the RSTS/E disk(s) on which SAVE/RESTORE operates must not be logically mounted. This ensures the integrity of the data on the disk(s). To invoke SAVE/RESTORE on line, you must type RUN \$SAVRES, then press the RETURN key. After you issue these commands, RSTS/E prints the SAVE/RESTORE program prompt, SAV/RES FUNCTION:. The example that follows represents the steps you take to initiate SAVE/RESTORE on-line.

```
RUN $SAVRES (RET)
SAV/RES Function:
```

Once the SAV/RES FUNCTION: prompt appears, either as a result of invoking SAVE/RESTORE off line or on line, you are under the control of SAVE/RESTORE and are ready for backup processing.

Next, select from Table 9-1 one of the following four operational functions: SA[VE], RE[STORE], IM[AGE], or ID[ENTIFY]. Unless you need to exit SAVE/RESTORE (by typing LINE FEED, or CTRL/Z) or need further help (by typing HE[LP] or RETURN key), type one of the four operational functions on the console terminal in response to the SAV/RES FUNCTION: prompt. The following example illustrates this procedure:

```
SAV/RES Function: SAVE
```

Press the RETURN key when you finish entering the selected function. This procedure invokes the SAVE/RESTORE dialogue. To begin the dialogue sequence, refer to Sections 9.5.5 through 9.5.8. Each section describes one of the operational functions.

Table 9-1: SAVE/RESTORE Functions

Function	Description
SA[VE]	Creates a copy of a RSTS/E file-structured disk. The SA[VE] function backs up to disk or tape.
RE[STORE]	Recreates a RSTS/E file-structured disk from a SAVE Set.
IM[AGE]	Copies a RSTS/E file-structured disk to a LIKE disk.
ID[ENTIFY]	Prints label information and other volume characteristics of a SAVE volume or a RSTS/E file-structured disk.
HE[LP]	Prints a table of SAVE/RESTORE functions.
Ⓕ	Press the LINE FEED key if you want to exit the SAVE/RESTORE program environment. If you invoke SAVE/RESTORE with a RUN command, pressing LINE FEED returns you to your default run-time system. If you are running SAVE/RESTORE off-line with INIT.SYS, pressing LINE FEED returns you to the "Option:" prompt issued by the INIT program.
Ⓗ	Press the RETURN key and the "Type HELP for help" message appears on the your terminal.
Ⓒ	Type CTRL/C to produce the same result as pressing the LINE FEED key.
Ⓐ	Type CTRL/Z to produce the same result as pressing the LINE FEED key.
other	Produces the same result as pressing the RETURN key.

Before proceeding to the appropriate SAVE/RESTORE dialogue, note that the function response in the previous example represents only one of three different ways you can answer the SAV/RES FUNCTION: prompt. The response to the prompt determines the type of dialogue to be carried out as well as the operation to be performed. A description and an example of each type of response appears in Table 9-2.

During the run, SAVE/RESTORE informs you of its operational status by printing messages on your terminal. Some of the messages are intended for your information; others notify you of error conditions that might require your attention. This latter group consists of errors that can occur during the dialogue, mount, transfer, or verification phases of SAVE/RESTORE.

When the program encounters an error during a dialogue or mount operation, it instructs you to rectify the problem, then asks you to re-enter the appropriate command. These errors never affect the integrity of the data being transferred.

On the other hand, errors that occur while SAVE/RESTORE is transferring or verifying data can jeopardize this data. In this case, SAVE/RESTORE resolves the problem, aborts the run, or requests that you continue or abort the program. In error situations of this type, the program gives you as much

information as it can to help protect your data. That is, SAVE/RESTORE lets you know what it is doing by issuing informational messages as well as error messages to your terminal.

Table 9-2: Response Types for the "SAV/RES Function:" Prompt

Response Type	Example and Description
Function (by itself)	SAV/RES Function:SAVE. If only a function is given, the full dialogue follows. See Section 9.5.5 through 9.5.8.
Function plus Option(s)	SAV/RES Function: SAVE/EXP:10-JUL-79 If the function is followed by options, dialogue questions will follow only for the devices to be used and the unspecified options. See Section 9.4.
Full Function Command Line	SAV/RES Function:IMAGE DM1:=DM0:MYPK/VER/NOERR If a complete command line is given, no dialogue questions will be asked. See Section 9.5.9.

Finally, SAVE/RESTORE prints (optional) summary information about SAVE, RESTORE, or IMAGE operations. This report describes the operation performed, device information, run start date and time statistics in addition to run total statistics. The number of errors encountered during the operation are listed as part of the run total statistics. The program prints this report at the end of a run.

9.4 SAVE/RESTORE Options

You can include SAVE/RESTORE options (switches) in a function response or in a single line command. The options that appear in Table 9-3 are of two types. The first two pairs of options correspond to SAVE/RESTORE dialogue questions and have option defaults. If neither of a pair of options is attached to a function response, an appropriate question will appear in the dialogue. However, if one of the pairs is specified, the corresponding question will not appear in the dialogue. If neither of the pair is specified in a full function command line, the default is assumed.

The last two options in Table 9-3 have no corresponding dialogue questions but do have default settings. Since no dialogue questions exist for the /STATS and /ERROR switches, they can only be initiated if specified with a function response or a full function command line. If you do not specify the /NOERROR switch, SAVE/RESTORE assumes the /ERROR setting. This default instructs SAVE/RESTORE to continue processing even if any of the errors described in Table 9-11 occur. You must explicitly specify the /NOERROR switch if you want SAVE/RESTORE to terminate as a result of any of these errors. The /STATS default setting forces SAVE/RESTORE to automatically print a summary report at the end of each run. Consequently, if you want to suppress the summary report, you must attach the /NOSTATS option to your function response. Refer to Table 9-3 for the definitions of these SAVE/RESTORE options.

Table 9-3: SAVE/RESTORE Options

Option	Default and Description
EX[PIRATION][:date] NOEX[PIRATION]	EXPIRATION This option is legal only for a SAVE operation. The date argument that you insert is the date after which you may overwrite the destination volume. You are warned by SAVE/RESTORE of any attempt to overwrite the volume(s) any time prior to the specified date. If you specify /EX[PIRATION] with no date argument, SAVE/RESTORE establishes the expiration date as one year after the current date. If you specify /NOEX[PIRATION], SAVE/RESTORE uses the current date as the expiration date.
VE[RIFY] NOVE[RIFY]	NOVERIFY This option is legal only for a SAVE, RESTORE, or IMAGE operation. If you specify the VE[RIFY] option, SAVE/RESTORE compares the volume(s) to ensure that the transfer was accurate.
ST[ATS] NOST[ATS]	STATS SAVE/RESTORE prints a summary report at the end of a run, unless you specify the /NOST[ATS] switch with either a function response or a single line command. The summary report is described in Section 9.8.
ER[ROR] NOER[ROR]	ERROR If you omit this option and a non-fatal error occurs during a SAVE/RESTORE run, an error message appears and processing continues. If you specify NOER[ROR] and any run-time error occurs (refer to Table 9-11), the program terminates after printing an error message.

9.5 SAVE/RESTORE Dialogue

The SAVE/RESTORE dialogue begins after you invoke the SAVE/RESTORE program and enter a function response to the SAV/RES FUNCTION: prompt. If you respond with SAVE, RESTORE, IMAGE, or IDENTIFY, the program initiates a set of dialogue questions. All questions in the dialogue have a short form or a long form. The short form appears automatically; the long form prints only if you press the RETURN key in response to the short form question. The long form provides information that may help you with an answer. Tables 9-5 through 9-8 contain the text of each of the long form questions.

Some of the dialogue questions have default values; these are printed in angle brackets next to the question. You may select the default value by pressing the LINE FEED key or by typing the proper default response. If the LINE FEED key is pressed in response to a question with no default value, the question is repeated.

There are two valid responses, CTRL/Z and CTRL/C, in the SAVE/RESTORE package that can alter the existing operating status. If you type CTRL/Z at any point in the dialogue, SAVE/RESTORE returns you to the previous question. If you want to abort the run at any time, type CTRL/C. After the abort, you will return to the OPTION: prompt if you are running off line under the control of the INIT.SYS program or to your default run-time system prompt if you invoked SAVE/RESTORE on line with a RUN command. SAVE/RESTORE automatically returns to the SAV/RES FUNCTION: prompt after the execution of a specified function. This occurs when you are either operating on line or off line. You may exit from the INIT environment version of SAVE/RESTORE to the OPTION: prompt by typing LINE FEED or CTRL/Z to the SAV/RES FUNCTION: prompt.

9.5.1 SAVE Volumes

You can perform SAVE/RESTORE operations on two classes of volumes: disks and tapes. The SAVE Set, a set of tapes or disks created from a SAVE operation, cannot include both disks and tapes. However, the SAVE Set may include one or more volumes of the same class. In other words, a tape volume SAVE Set could consist of two MM devices and one MT device while one RP04 and two RK05 disks constitute a legal disk SAVE Set. SAVE/RESTORE does not allow any disk/tape combination SAVE Sets.

A legal SAVE disk device is any disk supported by RSTS/E. Legal disk devices must meet the following requirements:

1. Must be formatted; that is, contain sector and track information.
2. All disks used by the on line version of SAVE/RESTORE must have a valid RSTS/E file structure; that is, the disk device must be formatted and DSKINTed prior to its use.
3. The first 16 blocks plus the Device Cluster Size blocks on a disk must not contain any bad blocks.
4. A RSTS/E output disk can contain no more than 161 bad clusters (the RSTS/E limit).

Magnetic tape is the only valid SAVE/RESTORE tape medium; DECtape or cassette are unacceptable. The statements here define what constitutes a legal tape in the SAVE/RESTORE package.

1. The magnetic tape must be long enough to contain the SAVE Set label and header information, a bootstrap, and INIT.SYS which is on the first volume only.
2. A bad block will be considered to be the end of the tape volume during the SAVE operation.
3. The default density is the lowest density allowed by the drive on which the first volume is mounted. Subsequent volumes cannot be written at any other density. Therefore, ensure that the density used on the first volume is compatible with all other tape drives to be used.

9.5.2 Device Specifications

Device specifications are required as responses to several dialogue questions. These specifications must be in the following general format.

`<device>[:[<id>]][/<option(s)>]`

where `<device>` is a two character device type (for disk either "Dx" or "Rx" is acceptable) followed by a single digit unit number in the range 0 to 7. The `<id>` represents either a SAVE Set name or a disk pack identification and the `<options>` are selected from Table 9-4.

Table 9-4: SAVE/RESTORE Device Specification Options

Option	Description
SCR[ATCH]	The SCR[ATCH] option is legal only on an output volume device specification. If you use the SCR[ATCH] option, SAVE/RESTORE by-passes most volume label checking. SAVE/RESTORE always checks the label to see if an output volume contains a SAVE Set written at the current density and ensures that it is not a volume from the SAVE Set currently being written. It also checks destination disks for bad block information.
DEN[SITY]:[800] [1600]	The DEN[SITY] option is legal only for magnetic tape. With this option you can specify the density at which the tape is to be written or read. If SAVE/RESTORE cannot read the tape at the specified density, it will try to read the tape at the other legal density setting.

9.5.3 Checking the Input Volume

SAVE/RESTORE checks the input volume prior to any actual data transfer and warns you of any problem that it foresees in saving, restoring, or copying the volume. All of the error messages that are printed at this point, concerning the input volume, are warning messages only; you can recover from them by following appropriate procedures. The error messages that result from a problem with the input volume are listed here and described separately below.

- %%% Input disk has only nn% free clusters.
Mount it anyway <NO>?
- %%% This is not the correct volume.
- %%% This volume has no label.
- %%% Input disk should be "CLEANed".

There must be at least as many good pack clusters on an output RSTS/E disk as there are clusters to be transferred from the original input disk. If SAVE/RESTORE determines that the input disk has very few clusters that are not allocated, it warns you that you may encounter problems copying or restoring the disk.

%%% Input disk has only nn% free clusters.
Mount it anyway <NO>?

Type NO or press the LINE FEED key to return to the previous device prompt. If you type YES, you are instructing SAVE/RESTORE to proceed with the SAVE or IMAGE operation. In the case of an IMAGE copy, the program also checks the output volume to see if the transfer can be made (see Section 9.5.4). No further checking takes place in the case of the SAVE operation. At this point, you may want to free up space on the input disk by deleting unnecessary files. This process will increase your chances of eventually completing a successful RESTORE.

When you specify a Pack ID or a SAVE Set name with the input device specification, SAVE/RESTORE compares the specified name with the name already written on the volume. If the names do not match, the following warning message is printed and the input device prompt is asked again:

```
%%% This is not the correct volume.
```

A dismount request follows the above warning message. It lists useful information from the volume label but does so only if it is a SAVE Set or a RSTS/E file-structured disk.

If the input volume is neither a RSTS/E disk nor a SAVE Set, the following warning message is displayed and the input device prompt is asked again:

```
%%% This volume has no label
```

If you remove a disk from a drive without logically dismounting it and subsequently attempt to remount it, SAVE/RESTORE will print the message that follows:

```
%%% Input disk should be "CLEANed".
```

To perform the cleaning operation, simply invoke the ONLCLN program or the REFRESH/CLEAN option of INIT. For information on ONLCLN refer to Section 7.6 of this manual. The REFRESH option of INIT is described in the *RSTS/E System Generation Manual*.

9.5.4 Checking the Output Volume

SAVE/RESTORE checks the output volume before SAVEing, RESTOREing, or IMAGE copying a RSTS/E disk. This procedure ensures that the SAVE/RESTORE operations will proceed to completion with the least chance of error. The messages that SAVE/RESTORE issues, as a result of checking the output volume, pertain to the expiration date, the label, and the available data space on the destination device. Some of the messages are printed only for your information; the rest warn you of serious output device problems. The paragraphs that follow exhibit the messages, describe why they appear, and explain how to respond to them.

When the expiration date of an output volume, labelled as a SAVE Set, has passed, SAVE/RESTORE prints this series of messages:

```
*** The volume on dev: is SAVE Set <xxxxxxx>
Density                : nnn
Creation date          : day-of-week, dd-mmm-yy
Expiration date        : day-of-week, dd-mmm-yy
```

and proceeds to the next dialogue question. If you decide not to use the expired date volume, type CTRL/Z to return to the TO DEVICE output dialogue question.

If the output volume is labelled as a SAVE Set and its expiration date has not passed, the following series of messages are printed:

```
*** The volume on dev: is SAVE Set <xxxxxx>
Density          : nnn
Creation date    : day-of-week, dd-mmm-yy
Expiration date: day-of-week, dd-mmm-yy
%%% Its expiration date has not passed
Mount it anyway <NO>?
```

If you type NO or press the LINE FEED key in response to the mount question, SAVE/RESTORE returns to the previous device prompt. SAVE/RESTORE will proceed to the next question if you type YES. If you press the RETURN key, the following message appears:

```
*** This is your last chance to prevent the
*** loss of any data on the output volume.
Mount it anyway <NO>?
```

SAVE/RESTORE notifies you with the following message that the output volume you are using is labelled as a RSTS disk:

```
*** The volume on dev: is a RSTS disk
*** Pack ID is <xxxxxx>
*** Pack will be reinitialized
Mount it anyway <NO>?
```

If you type NO or press the LINE FEED key, you direct SAVE/RESTORE to return to the previous output prompt. A YES response, at this point, indicates that you wish to use the present volume.

If you RESTORE or IMAGE copy a disk, SAVE/RESTORE checks to see whether the total number of clusters on the output disk minus the known bad blocks is greater than or equal to the number of allocated clusters on the original source disk. If SAVE/RESTORE determines that there is not enough space and the transfer cannot be made, it prints:

```
%%% Too many bad blocks on output disk.
```

and then returns to the previous device prompt. If, however, the program determines that there are very few free blocks, it prints the following warning:

```
%%% Only nn% of the output disk clusters are available for relocation.
Mount it anyway <NO>?
```

You must decide at this point whether the number of free clusters is large enough to make possible a successful SAVE/RESTORE operation. If you decide it is not, type NO or press LINE FEED to return to the output prompt. Type YES to direct SAVE/RESTORE to proceed with the operation if you think there are enough free clusters available. Whenever this warning appears, DIGITAL recommends that, when it is possible, you use another disk

for output. Follow this procedure since there exists a higher than usual possibility that the operation will not succeed. Note that DIGITAL does not guarantee that the operation will succeed even if this message does not appear. SAVE/RESTORE normally transfers each allocated pack cluster from the original source disk to the same cluster on the output RSTS/E disk. If the output cluster is found to be bad, SAVE/RESTORE will try to relocate the data. If the cluster size of the item being moved is larger than the pack cluster size, there must be enough contiguous free pack clusters on the disk to accommodate the entire entity cluster. If there is not SAVE/RESTORE will abort the operation.

If the specified output is not a RSTS/E or a SAVE Set disk it prints the new SAVE Set name and proceeds without further notice. The program does not check the output volume label if you specified the /SCR[ATCH] option, except to determine whether it is a volume of the SAVE Set that is currently being created. It always attempts to recover bad block information from a RSTS file-structured and SAVE disk.

9.5.5 Saving a RSTS/E Disk using the SAVE Dialogue

The SAVE function allows you to backup an entire RSTS/E formatted disk to either tape or disk. The output volumes created by this operation are collectively called the SAVE Set and are written by SAVE/RESTORE in the SAVE Format. You can recreate the original disk only by using the RESTORE operation. You cannot use other RSTS/E software, such as BACKUP or PIP.SAV to restore a SAVE Set. Note that you must use SAVE if the output medium is different from the input medium.

Once you have determined that the disk you are to SAVE and the output SAVE Set device(s) are valid media, type SAVE; this response to the SAV/RES FUNCTION: prompt initiates the SAVE dialogue questions described in Table 9-5.

NOTE

When SAVE/RESTORE creates a SAVE Set, it includes on the output volume an extra set of directory blocks in case some directory blocks are found to be bad during a RESTORE operation. These additional blocks are reflected in the total block count for a SAVE Summary Report. Since the extra blocks are needed only when bad blocks are encountered during a RESTORE operation, it is unnecessary for SAVE/RESTORE to transfer these blocks to the restored volume. Thus, you will notice a discrepancy between the number of blocks transferred for a SAVE operation and the number transferred during a RESTORE. In addition, because the extra directory blocks are not transferred to the restored volume, under certain circumstances you may restore your data to one less volume than that which existed in the SAVE Set.

The Pack ID of the input disk is the default SAVE Set Name of the output volumes. After you answer the FROM RSTS DISK question, SAVE/RESTORE informs you of the output SAVE Set Name it intends to use. The general format of this message is:

```
*** Pack ID/default SAVE Set Name is <xxxxxx>
```

If you do not wish the output volume(s) to have this SAVE Set name, specify a different Pack ID in the <devspec> response to the TO DEVICE question. After you select the ID and answer the question, SAVE/RESTORE prints:

```
*** SAVE Set Name is <xxxxxx>
```

SAVE/RESTORE will warn you if the output volume contains a SAVE Set that has an expiration date which has not passed. You may choose to proceed or re-specify the output volume (Section 9.5.4).

Table 9-5: SAVE Dialogue Questions

Question	Response and Description
From RSTS Disk?	<p><devspec> Data is transferred from the specified disk. If you include a Pack ID in the <devspec> response, SAVE/RESTORE checks to see whether it matches the one on the disk. If it does not, the program prints a warning message (Section 9.5.3) and prints the FROM RSTS DISK prompt again. The disk must be physically but not logically mounted and should be write-locked.</p> <p>"Type 2 characters to specify the input device type, followed by a single digit (0-7) to specify the unit number. Include a Pack ID if you want SAVRES to ensure it matches the one on the volume. This drive should be write-locked."</p>
To Device?	<p><devspec> All data is transferred from the previously specified input disk to this tape or disk. If you do not specify a SAVE Set Name in the <devspec>, SAVE/RESTORE gives the SAVE Set the same name as the input disk.</p> <p>"Type 2 characters to specify the output device type, followed by a single digit (0-7) to specify the unit number. Include a Save Set Name if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled."</p>
Expiration Date <DD-MMM-YY>?	<p><DD-MMM-YY> This is the dialogue form of the /EXPIRATION switch (Section 9.4). If you do not specify a date, SAVE/RESTORE creates an expiration date for the SAVE Set that is one year from the date that it was created.</p> <p>Type the desired expiration date or press the LINE FEED key to accept the date printed.</p>
Verify (Yes or No) <NO>?	<p>Y[ES] or N[O] This is the dialogue form of the /VERIFY switch (Section 9.4). If you respond to this question by typing YES, the output is then compared to the input at the end of each volume of the SAVE Set.</p> <p>"Type 'YES' if you want SAVRES to compare the input and output volumes at the end of each output volume." Type 'NO' or press the LINE FEED key if you do not want SAVRES to perform this verification.</p>
Proceed (Yes or No)?	<p>Y[ES] or N[O] This question allows you to double-check your dialogue responses, to abort the operation if errors have been made, or proceed with the SAVE operation.</p> <p>"Type 'YES' to proceed with the operation." Type 'NO' to abort and return to the SAV/RES Function: prompt."</p>

The SAVE example that follows illustrates the backup of an RM03 disk to tape. The SAVE procedures are:

- Physically mount the RM03 source disk on drive DR1:, write-locked.
- Physically mount the destination tape on drive MM1:, write-enabled.
- Type RUN \$SAVRES then press the RETURN key.

a. RUN \$SAVRES (RET)

b. SAV/RES Function: SA

c. From RSTS disk? DR1:

*** Pack ID/default SAVE Set Name is JOEM

d. To Device? MM1:SAVSET

*** SAVE Set Name is SAVSET

e. Expiration Date <23-Jan-80>? 1-JUN-79

f. Verify (Yes or No) <NO>?

g. Proceed (Yes or No)? Y

*** Initializing first SAVE volume

*** Begin SAVE from DR1: to MM1: at 08:07 PM

h. Dismount Device: MM1:
 Set Name: SAVSET
 Seq #: 1
 Density: 1600
 Creation date: Tuesday, 23-Jan-79
 Expiration date: Friday, 01-Jun-79

 Please label this volume!

Mount volume # 2 of SAVE Set SAVSET

i. Device ? MM2

j. Proceed (Yes or No)? Y

*** Begin SAVE from DR1: to MM2: at 08:14 PM

 Dismount Device: MM2:
 Set Name: SAVSET
 Seq #: 2
 Density: 1600
 Creation date: Tuesday, 23-Jan-79
 Expiration date: Friday, 01-Jun-79

 Please label this volume!

--- SAVE operation completed at 08:22 PM

k. Summary Report

SAVE of DR1:JOEM to SAVE Set SAVSET

 Input Device: DR1:
 Pack ID: JOEM
 Pack Clustersize: 8
 Creation date: Friday, 15-Dec-78

Output Device: Mastape
Set Name: SAVSET
of volumes: 2
Density: 1600
Creation date: Tuesday, 23-Jan-79
Expiration date: Friday, 01-Jun-79

SAVE started on Tuesday, 23-Jan-79, at 08:07 PM

l. Run Statistics

Transfer Totals:

Total of 55192 blocks transferred

Error Totals:

Total of 0 new bad blocks encountered on source.

Timing Totals:

Total elapsed time: 0 hrs., 14 mins., 41 secs.
Total wait time: 0 hrs., 0 mins., 19 secs.
Total process time: 0 hrs., 14 mins., 22 secs.

m. SAV/RES Function: ^Z

Ready

The following list of comments represents an explanation of the preceding SAVE operation. The letters preceding each comment in the explanation are keyed to ones in the example.

- a. Type RUN \$SAVRES, then press the RETURN key to invoke the SAVE/RESTORE program.
- b. After you press RETURN, SAVE/RESTORE prints its initial program prompt. Because you need to perform a backup operation of an RM03, you select the SAVE function of SAVE/RESTORE by typing SA and the RETURN key. You do not need to enter more than the function's first two characters.
- c. SAVE/RESTORE asks you for the device name and unit number of the disk from which you are extracting data. You respond with DR, the device mnemonic of the RM03 disk, and indicate that it is mounted on drive 1. As an option, you could have included a Pack ID with the device specification. If you had, SAVE/RESTORE would have compared your Pack ID response to the ID currently on the disk. By not specifying a Pack ID you indicate to SAVE/RESTORE that you want to accept the current ID. As soon as you terminate your response to this question, the program informs you that the Pack ID/default SAVE Set Name is JOEM.
- d. You specify the medium to which SAVE copies the input data. Since you want the SAVE Set to reside on tape, you type MM1:. Rather than allow SAVE to give the SAVE Set the same name as the input disk, you provide the SAVE Set its own. If you give the input and output media separate names, you will be better able to differentiate their identity (with IDENTIFY) at a future date. As soon as you terminate your response, SAVE stores the the name you specified on the output medium, then prints the SAVE SET NAME IS SAVSET message.

- e. SAVE creates a default expiration date that is equal to the current date plus one year. This represents the date after which you can write over the data. Rather than accept the default by pressing LINE FEED, you specify your own date and press the RETURN key.
- f. A YES or Y response instructs SAVE to perform a verify pass after it completes the data transfer process. In the verify pass SAVE compares the input data with the data it transferred to the output medium. You specify the Y response here. Alternatively, if you know that you need to verify the transferred data, you can attach the /VERIFY switch to the SAVE/RESTORE function and thus suppress the VERIFY (YES or NO) <NO> question.
- g. You have made no errors in the dialogue sequence; therefore, you can type Y to inform SAVE that it can proceed with the SAVE operation. Since no default exists for this question, you must enter either a YES or a NO response. Once you terminate your answer, SAVE notifies you that it is initializing the first SAVE volume. Initialization occurs only to the first volume of the SAVE Set. When that operation is complete, SAVE prints a message indicating that the transfer process has begun, which devices are involved, and the time the operation began.
- h. The dismount report summarizes the label information found on the initial SAVE Set volume. If SAVE/RESTORE requires another volume, it asks you to mount the next volume in the SAVE Set after the first volume is exhausted.
- i. You respond to the DEVICE question with the device name and unit number of the second SAVE Set volume, then press the RETURN key.
- j. Again, you have the option to proceed or to abort the operation. Since there is no reason to abort, you type Y and the RETURN key to proceed. SAVE then notifies you, as it did for the first volume, that the SAVE operation has begun, that the data is being transferred, and that 08:14 was the time the SAVE operation began. As soon as SAVE finishes the transfer, it issues the dismount message for the second volume of the SAVE Set, informs you that the SAVE operation is complete, and the time it finished.
- k. The Summary Report appears by default and can only be suppressed by attaching the /NOSTATS switch to a SAVE/RESTORE function. The report includes the Pack ID and SAVE Set names, label information for both the input and output volumes, and the date of the SAVE operation.
- l. The Run Statistics report prints the number of blocks transferred, the number of bad blocks encountered, and elapsed times. Note that the blocks transferred includes an extra set of directory blocks stored at the end of the SAVE Set. The additional set of directory blocks are included to ensure that SAVE/RESTORE has a valid directory to read when it attempts to restore from the SAVE Set.
- m. When the SAVE operation is complete, SAVE/RESTORE returns to its initial SAV/RES FUNCTION: prompt. If you want to return to your default run-time system, type CTRL/Z.

9.5.6 Restoring a RSTS/E Disk using the RESTORE Dialogue

To recreate a RSTS/E file-structured disk from a SAVE Set you must perform the RESTORE operation. Because SAVE/RESTORE writes the SAVE Set in the SAVE format, no other RSTS/E software can perform this rebuild process. Before transferring any data, SAVE/RESTORE tries to extract bad block information from the destination disk and if any bad blocks are found, incorporates them into the new BADB.SYS file.

The SAVE/RESTORE program may encounter bad blocks during a RESTORE run. If it does, it scans the SATT.SYS file to determine if there is a place where the block corresponding to a bad block can be relocated. If a large enough area exists on the destination disk, the program simply moves the data to the new location and modifies the directory information accordingly.

The relocation of blocks may require that RESTORE change a file's characteristics. This situation might occur, for example, if the program encounters a bad cluster while transferring a contiguous file. In order to relocate the cluster that falls on the bad block, SAVE/RESTORE must make the file noncontiguous. Changes of this type can cause you problems, but RESTORE will preserve the data and inform you of these adjustments.

The operation terminates if there is no way RESTORE can relocate the data cluster. This may occur if:

- No unallocated pack cluster remains on the disk.
- The file cluster size is larger than the pack cluster size and there is not an equivalent number of contiguous pack clusters left on the disk.

Table 9-6 lists the dialogue questions that appear if you choose the RESTORE function of SAVE/RESTORE. An example of a RESTORE procedure follows the description of the RESTORE dialogue questions.

Normally, SAVE/RESTORE uses the name of the input SAVE Set as the Pack ID of the output volume. After you answer the FROM DEVICE question, SAVE/RESTORE prints a message that informs you of the name it plans to use. The general format of the message is:

```
*** SAVE Set name/Default Pack ID is xxxxxx
```

After you successfully respond to the TO RSTS DISK question, SAVE/RESTORE prints the message:

```
*** Pack ID is xxxxxx
```

The example that follows illustrates the restoration of an RM03 disk from the tape SAVE Set created in the previous example.

- Mount the two SAVE Set volumes on drive MM1: and MM2:, write-locked.
- Mount the destination disk on DR1:, write-enabled.
- Invoke SAVE/RESTORE by typing RUN \$SAVRES.

Table 9-6: RESTORE Dialogue Questions

Question	Response and Description
From Device?	<p><devspec> Data is transferred from the specified device. If you include a SAVE Set name in the <devspec> (see Section 9.5.2 for a discussion of device specifications), SAVE/RESTORE checks to see if it matches the one on the device. If it does not, the program prints a warning message (Section 9.5.3) and prints the FROM DEVICE question again. The device must be physically mounted and should be write-locked.</p> <p>"Type 2 characters to specify the input device type, followed by a single digit (0-7) to specify the unit number. Include a SAVE Set Name if you want SAVRES to ensure it matches the one on the volume. This drive should be write-locked".</p>
To RSTS Dx: Disk?	<p><devspec> Data from the previously specified SAVE Set is restored to this disk. If you specify a Pack ID in the <devspec>, it becomes the Pack ID of the RESTORED disk. Otherwise, SAVE/RESTORE uses the SAVE Set Name (which may also be the Pack ID of the original source disk). Physically mount and write-enable the disk.</p> <p>"Type 2 characters to specify the output device type, followed by a single digit (0-7) to specify the unit number. Include a Pack ID if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled".</p>
Verify (Yes or No) <NO>?	<p>Y[ES] or N[O] This is the dialogue form of the /VER[IFY] switch. If you respond to this question with "YES", SAVE/RESTORE compares the output with the input at the end of each input volume to ensure that the volumes are equivalent.</p> <p>"Type `Yes` if you want SAVRES to compare the input and output volumes at the end of each input volume. Type `No` or press the LINE FEED key if you do not want SAVRES to perform this verification."</p>
Proceed (Yes or No)?	<p>Y[ES] or N[O] This question allows you to double-check your dialogue responses, to abort the operation if errors have been made, or proceed with the RESTORE operation.</p> <p>"Type `Yes` to proceed with the operation. Type `No` to abort and return to the SAV/RES Function: prompt."</p>

```

a. RUN $SAVRES

b. SAV/RES Function: RE

c. From device? MM1:
   *** SAVE Set Name/default Pack ID is SAVSET

d. To RSTS DR: Disk? DR1:TSTPAK
   *** The volume on DR1: is a RSTS disk
   *** Pack ID is MYPAK

   *** Pack will be reinitialized
   Mount it anyway <NO>? Y

   *** Pack ID is TSTPAK

e. Verify (Yes or No) <NO>?

f. Proceed (Yes or No)? Y
   *** Begin RESTORE from MM1: to DR1: at 08:28 PM

       Dismount Device:  MM1:
           Set Name:     SAVSET
           Seq #:        1
           Density:      1600
           Creation date: Tuesday, 23-Jan-79
           Expiration date: Friday, 01-Jun-79

       Mount volume # 2 of SAVE Set SAVSET
       Device ? MM2:

       Proceed (Yes or No)? Y
       *** Begin RESTORE from MM2: to DR1: at 08:32 PM

           Dismount Device:  MM2:
               Set Name:     SAVSET
               Seq #:        2
               Density:      1600
               Creation date: Tuesday, 23-Jan-79
               Expiration date: Friday, 01-Jun-79

           Dismount Device:  DR1:
               Pack ID:      TSTPAK
           Pack Clustersize: 8
               Creation date: Friday, 15-Dec-78

           Please label this volume!

       --- RESTORE operation completed at 08:37 PM

g.                Summary Report

   RESTORE of SAVE Set SAVSET to DR1:TSTPAK

       Input Device:  Mastape
           Set Name:   SAVSET
       # of volumes:  2
           Density:    1600
       Creation date:  Tuesday, 23-Jan-79
       Expiration date: Friday, 01-Jun-79

```

Output Device: DR1:
Pack ID: TSTPAK
Pack Clustersize: 8
Creation date: Friday, 15-Dec-78

RESTORE started on Tuesday, 23-Jan-79, at 08:28 PM

h. Run Statistics

Transfer Totals:

Total of 54184 blocks transferred

Error Totals:

Total of 0 new bad blocks encountered on destination.

0 files structurally altered.

Timing Totals:

Total elapsed time: 0 hrs., 9 mins., 36 secs.
Total wait time: 0 hrs., 0 mins., 16 secs.
Total process time: 0 hrs., 9 mins., 20 secs.

i. SAV/RES Function: ^Z

Ready

The following list of comments represent an explanation of the preceding RESTORE operation. The letters preceding each comment are keyed to ones in the example.

- a. Since you are under the control of your default run-time system, you type RUN \$SAVRES, then press the RETURN key. This procedure invokes the SAVE/RESTORE program.
- b. You initiate the RESTORE operation by typing RE and the RETURN key in response to the initial SAVE/RESTORE prompt. Because you need to recreate an RM03 from a tape SAVE Set, you selected the RESTORE function.
- c. The RESTORE dialogue begins when it prints the FROM DEVICE question. In response to it, you specify the device name and the unit number of the SAVE Set medium and press the RETURN key. Since the input medium is tape, mounted on drive number 1, you enter MM1:. If you had included a SAVE Set Name in the device specification, RESTORE would have checked it against the name already on the set. If they had not matched, RESTORE would have issued a warning message informing you of the discrepancy. After you terminate your response with the RETURN key, RESTORE extracts the SAVE Set Name from the input medium and prints it on your terminal.
- d. Since in this RESTORE operation you are recreating an RM03 disk, you respond to the TO RSTS DR: DISK question by typing DR1:TSTPAK. You indicate by your response that the device mnemonic of the RM03 is DR, that it is mounted on drive number 1, and that you wish it to have a Pack ID of TSTPAK. After pressing the RETURN key, RESTORE checks

the Pack ID of the output disk. Subsequently, it prints a message informing you that it is a RSTS disk, has a Pack ID of MYPAK, and that RESTORE will reinitialize it. RESTORE issues a mount question which gives you the option to mount the disk or not. Because the data on the disk is no longer valuable you respond to the mount question by typing Y and the RETURN key. RESTORE then initializes the disk with the new Pack ID.

- e. The comparison of the input data to the data already transferred to the destination disk is not essential during this particular run; therefore, you press the LINE FEED key to accept the NO verify default. When you request that RESTORE verify the transferred data, you are essentially asking RESTORE to perform two operations rather than one. That is, you make the program perform two passes on the data, once for the transfer and once for the comparison. If you do not have enough time to allow the double pass, having the option not to verify may be important.
- f. No mistakes were made during the dialogue phase; therefore, you instruct RESTORE to proceed with the recreate by typing Y. After you press the RETURN key to mark the end of entered data, RESTORE begins the transfer. When RESTORE starts to transfer data, it prints a message to this affect on your terminal. After RESTORE exhausts the first volume, it prints input medium information, asks you to mount the next volume, and asks if you want to continue the operation. Again, since there are no errors in the dialogue, you instruct RESTORE to proceed. RESTORE moves the remaining data to the second volume, issues the input and output dis-mount messages, and informs you that the operation is completed.
- g. The Summary Report prints the SAVE Set Name and the Pack ID, label information for both the input and output medium, in addition to the date and time of the RESTORE operation.
- h. The Run Statistics Report prints the number of blocks transferred, the number of bad blocks encountered, and the times required to perform various operations. Note that the number of blocks transferred does not include the extra set of directory blocks stored at the end of the SAVE Set.
- i. RESTORE returns to the SAV/RES FUNCTION: prompt when an operation has ended. If you type a CTRL/Z to this prompt, you return to your default run-time system control.

9.5.7 Copying a RSTS/E Disk using the IMAGE Dialogue

The IMAGE function allows you to make an equivalent copy of a RSTS/E file-structured disk; however, only disk to LIKE disk transfers are legal when using the IMAGE operation. LIKE disks are units that have exactly the same device size. The SAVE/RESTORE program handles bad block recovery during an IMAGE operation in the same way as for RESTORE.

SAVE/RESTORE asks the dialogue questions that appear in Table 9-7 if you choose the IMAGE function. An example of the IMAGE operation follows the Table 9-7 description.

Table 9-7: IMAGE Dialogue Questions

Question	Response and Description
From RSTS Disk?	<p><devspec> SAVE/RESTORE transfers all data from the disk specified in the <devspec> response. If you include a Pack ID in the <devspec>, SAVE/RESTORE checks to see if it matches the one actually on the disk. If it does not, the program prints a warning message (section 9.5.3) and prints the FROM RSTS DISK question again. The device must be physically mounted and for safety reasons should be write-locked.</p> <p>"Type 2 characters to specify the input device type, followed by a single digit (0-7) to specify the unit number. Include a Pack ID if you want SAVRES to ensure it matches the one on the volume. This drive should be write-locked."</p>
To RSTS Dx: Disk?	<p><devspec> SAVE/RESTORE transfers all information from the previously specified source disk to this disk. If you include a Pack ID in the <devspec>, it becomes the Pack ID of the output disk. Otherwise, SAVE/RESTORE uses the Pack ID of the source disk. The disk must be mounted and write-enabled.</p> <p>"Type 2 characters to specify the output device type, followed by a single digit (0-7) to specify the unit number. Include a Pack ID if you wish to override the default. This drive should have a SCRATCH volume mounted, and must be write-enabled."</p> <p>Note: The device mnemonic you specify in response to the FROM RSTS DISK question will appear in place of the Dx in the TO RSTS Dx: DISK? question. In some cases, you may specify, in your <devspec> response to the TO RSTS Dx: DISK question, a device mnemonic other than the one that appeared for Dx. See the discussion of LIKE disks in Section 9.2</p>
Verify (Yes or No) <NO>?	<p>Y[es] or N[o] This is the dialogue form of the /VER[IFY] switch. If you respond to this question with "YES", SAVE/RESTORE compares the output with the input at the end of each output volume to ensure that the volumes are equivalent. If you respond to this question with "NO", SAVE/RESTORE moves on to the next question without performing a verify pass.</p> <p>"Type `Yes` if you want SAVRES to compare the input and output volumes at the end of the IMAGE copy operation. Type `No` or press the LINE FEED key if you do not want SAVE/RESTORE to perform the verification"</p>
Proceed (Yes or No)?	<p>Y[ES] or N[O] This question allows you either to double-check your dialogue responses and abort the operation if you have made any errors or to proceed with the IMAGE operation.</p> <p>"Type `Yes` to proceed with the operation. Type `No` to abort and return to the SAV/RES Function: prompt."</p>

SAVE/RESTORE will, if you do not intervene, use the input Pack ID as the Pack ID of the output disk. After you answer the FROM RSTS DISK? question, the program informs you of the Pack ID it intends to use. The general format of this message is:

```
*** Input Pack ID/default Output Pack ID is XXXXXX
```

If you do not want the output disk to have the same name as the input volume, specify a different Pack ID in the <devspec> response to the TO RSTS DX: DISK question. After you select the Pack ID and answer the question, SAVE/RESTORE prints the following statement:

```
*** Output Pack ID is XXXXXX
```

The procedure below describes an IMAGE operation that copies an RP04 to an RP04 and verifies the transfer in the same pass.

The IMAGE procedures are:

- Mount the source disk on drive DB0:, write-locked.
- Mount the destination disk on drive DB1:, write-enabled.
- Type RUN \$SAVRES, then press the RETURN key.

a. RUN \$SAVRES

b. SAV/RES Function: IMAGE/NOERROR

c. From RSTS disk? DB0:

```
*** Input Pack ID/default Output Pack ID is SOURCE
```

d. To RSTS DB: Disk? DB1:DESTIN

```
*** The volume on DB1: is a RSTS disk
```

```
*** Pack ID is OUTPUT
```

```
*** Pack will be reinitialized
```

```
Mount it anyway <NO>? YES
```

```
*** Output Pack ID is DESTIN
```

e. Verify (Yes or No) <NO>? YES

f. Proceed (Yes or No)? YES

```
*** Begin IMAGE copy from DB0: to DB1: at 11:02 AM
```

```
*** Begin VERIFY Pass from DB0: to DB1: at 11:04 AM
```

```
*** 0 differences found
```

```
Dismount Device: DB1:
```

```
Pack ID: DESTIN
```

```
Pack Clustersize: 8
```

```
Creation date: Monday, 22-Jan-79
```

```
Please label this volume!
```

```
--- IMAGE copy operation completed at 11:07 AM
```


g. Summary Report

IMAGE copy of DB0:SOURCE to DB1:DESTIN

Input Device: DB0:
Pack ID: SOURCE
Pack Clustersize: 8
Creation date: Monday, 22-Jan-79

Output Device: DB1:
Pack ID: DESTIN
Pack Clustersize: 8
Creation date: Monday, 22-Jan-79

IMAGE copy started on Monday, 22-Jan-79, at 11:02 AM

h. Run Statistics

Transfer Totals:

Total of 24400 blocks transferred

Error Totals:

Total of 0 bad compares.

Total of 0 new bad blocks encountered on source.

Total of 0 new bad blocks encountered on destination.

0 files structurally altered.

Timing Totals:

Total elapsed time: 0 hrs., 5 mins., 22 secs.

Total wait time: 0 hrs., 0 mins., 0 secs.

Total Process time: 0 hrs., 5 mins., 22 secs.

i. SAV/RES Function: ^Z

Ready

The following list of comments represents an explanation of the preceding IMAGE operation. The letters preceding each comment are keyed to ones in the example.

- a. When you are under the control of your default run-time system, you type RUN \$SAVRES, then press the RETURN key to invoke the SAVE/RESTORE program.
- b. SAVE/RESTORE prints the SAV/RES FUNCTION: prompt to determine which function you wish to perform. Since you need to create a copy of an RP04, you select the IMAGE function by typing IM/NOERROR and the RETURN key in response to this prompt. Because the data on the disk is valuable, you attach the /NOERROR switch; this option instructs SAVE/RESTORE to terminate the operation whenever the program encounters a run-time error. Before the program aborts, it prints an error message indicating the source of the problem.

- c. In reply to the FROM RSTS DISK question, you specify the disk from which you are to extract the data. You enter the device name and unit number of the input disk and terminate the response with the RETURN key. After this response, IMAGE extracts the Pack ID from the input disk and prints it on your terminal.
- d. IMAGE requests the device name and unit number of the output disk. You type DB1:DESTIN to indicate that the RP04 is on drive 1 and you want DESTIN to be the new Pack ID. As soon as you type the RETURN key, SAVE/RESTORE informs you that the output medium is a RSTS disk, its Pack ID is OUTPUT, and it will be initialized. SAVE/RESTORE initializes the disk when you type Y and the RETURN key in response to this mount question. A message that SAVE/RESTORE prints informs you that the disk was initialized to the specified Pack ID.
- e. Since you are IMAGE copying valuable data, you type YES to instruct IMAGE to determine if the input data equals the data that was transferred to the output disk. When IMAGE finds that the data are not equivalent, it informs you of these differences.
- f. You are satisfied that you made no errors in the dialogue sequence and feel that the operation should proceed. After you press the RETURN key, IMAGE informs you when it begins to transfer data and when it begins the verify pass. After completing the data compare, IMAGE prints the number of differences found between the two devices. It then issues the output device dismount message. Immediately after IMAGE prints the dismount request, it notifies you that the copy operation has successfully ended.
- g. IMAGE prints the Summary Report unless you suppress it by using the /NOSTATS switch. The report includes the Pack ID and label information of each disk, in addition to the date and time that the IMAGE operation was run.
- h. The Run Statistics Report provides a total figure for the number of blocks transferred, the errors encountered, and the times required to perform the IMAGE operation.
- i. SAVE/RESTORE returns you to the SAV/RES FUNCTION: prompt when the operation ends. You type CTRL/Z to return to your default run-time system.

9.5.8 IDENTIFY Dialogue

If you want to extract label information from a SAVE Set volume or a RSTS/E file-structured disk, use the IDENTIFY function. SAVE/RESTORE asks only one question before the IDENTIFY operation begins. Table 9-8 describes this dialogue question. Examples of the IDENTIFY operation appear after the table.

Table 9-8: IDENTIFY Dialogue Question

Question	Response and Description
From Device?	<p><devspec> SAVE/RESTORE prints the device label characteristics of the SAVE Set or RSTS/E file-structured disk for the device specified in the dialogue response. If you include a SAVE Set name or Pack ID, SAVE/RESTORE checks to see if it matches the one actually on the device. If it does not, the program prints a warning message (see section 9.5.3) and then prints the FROM DEVICE question again. You must mount the device before responding to the FROM DEVICE question and write-lock it for protection.</p> <p>"Type 2 characters to specify the input device, followed by a single digit (0-7) to specify the unit number. Include a SAVE Set name or Pack ID if you want SAVE/RESTORE to ensure it matches the one on the volume. Mount this drive, write-locked, before answering the question."</p>

The following IDENTIFY procedure lists the label characteristics of both a SAVE Set and a RSTS/E disk.

The IDENTIFY procedures are:

- Mount a SAVE Set volume on MM1:, write-locked.
- Mount the RSTS/E disk on drive DR1:, write-locked.
- Type RUN \$SAVRES, then press the RETURN key.

a. RUN \$SAVRES

b. SAV/RES Function: ID MM1:

```

c.      Device:      MM1:
        Set Name:    SAVSET
        Seq #:       1
        Density:     1600
        Creation date: Tuesday, 23-Jan-79
        Expiration date: Wednesday, 23-Jan-80
  
```

d. SAV/RES Function: ID DR1:

```

        Device:      DR1:
        Pack ID:      TSTPAK
        Pack Clustersize: 8
        Creation date: Friday, 15-Dec-78
  
```

e. SAV/RES Function: ^Z

Ready

The following list of comments represents an explanation of the preceding IDENTIFY operation. The letters preceding each comment are keyed to ones in the example.

- a. To invoke SAVE/RESTORE you type RUN \$SAVRES, then press the RETURN key.

- b. The SAVE/RESTORE program prints the SAV/RES FUNCTION: prompt. You respond to it with the first two letters of the IDENTIFY function and with the device name and unit number of the medium you wish to identify. Include a Pack ID if you want SAVE/RESTORE to compare the ID you specify with the one on the input volume.
- c. IDENTIFY prints label information for the tape SAVE Set, then returns to the its initial prompt.
- d. SAVE/RESTORE returns to the SAV/RES FUNCTION: prompt. In response, you instruct SAVE/RESTORE to identify the disk pack mounted on DR1:. SAVE/RESTORE prints the information as soon as you terminate your response.
- e. You type CTRL/Z to return to your default run-time system.

9.5.9 Full Function Command Line

Use the full function command line format if you do not want to answer each dialogue question separately. If you specify a full command in response to the SAV/RES FUNCTION: prompt, SAVE/RESTORE issues only the PROCEED (Yes or No) question (except during an IDENTIFY question). The general format of the full function command line is:

SAV/RES Function:<function><outdevspec>=<indevspec>[/<options>]

Proceed (Yes or No)?

where <function> can be either SA[VE], RE[STORE], IM[AGE], or ID[ENTIFY] and <options> can be any of those listed as valid with the associated function as described in Section 9.4. The device specifications are presented in Section 9.5.2.

SAVE/RESTORE does the same standard volume label checking during the full function command line operation as it does during the longer dialogue procedure. If a discrepancy does occur after responding with a full command, SAVE/RESTORE will print the error message and reprint the SAV/RES FUNCTION: prompt. At that point, you must type the full function command line again.

9.6 SAVE/RESTORE and Booting

SAVE/RESTORE always attempts to create bootable media. The output of a RESTORE or IMAGE copy operation will be bootable if the original disk contained an INIT.SYS file in [0,1] (even if the original disk was not bootable). The first output volume produced by a SAVE operation is always bootable since SAVE/RESTORE copies the INIT.SYS file from the system disk and hooks it on this volume. Note that only the first volume in a SAVE Set is bootable.

If a SAVE volume is booted, a portion of INIT.SYS is loaded allowing you to perform the following INIT options:

1. SAVE/RESTORE
2. DSKINT
3. BOOT
4. HARDWR option specifying the LIST suboption

The types of SAVE/RESTORE operations that you can perform depend on the type of device from which you boot. You can invoke all four SAVE/RESTORE options if you boot from a RSTS/E disk. However, you can perform only an IDENTIFY, IMAGE, or RESTORE if you boot from a RSTS/E magnetic tape or SAVE volume. Note that if you boot a SAVE volume and subsequently perform a RESTORE operation, SAVE/RESTORE prints a message at the completion of the run:

```
*** Please boot from the system disk
```

```
Boot device?
```

In response to this prompt, you must specify the device name and unit number of the bootable device. The SAVE/RESTORE operations that you can perform while booting various devices are summarized in Table 9-9.

Table 9-9: Booting RSTS/E and SAVE Volumes

Device Booted From	SAVE/RESTORE Operations you can perform	SAVE/RESTORE Operations you can perform afterwards
RSTS/E Magnetic Tape	IDENTIFY, IMAGE, RESTORE	IDENTIFY, IMAGE, RESTORE
SAVE Volume (Disk or Tape)	IDENTIFY, IMAGE	IDENTIFY, IMAGE, RESTORE
	RESTORE	None - You receive a message which states you must boot from the system disk.
RSTS/E Disk	SAVE, RESTORE, IMAGE, IDENTIFY	SAVE, RESTORE, IMAGE, IDENTIFY

NOTE

If you create a magnetic tape SAVE Set with a density of 1600 BPI, you cannot use the console switches to boot the tape if the drive you have the tape mounted on has a TM02 formatter. However, you can boot the 1600 BPI tape on a drive with a TM02 formatter if you specify the BOOT option to the INIT Option: prompt. That is, you can boot the device with the software boot but not a hardware boot.

9.7 Operator Interface During Processing

SAVE/RESTORE requires operator intervention at certain points during processing. The operator must mount and dismount devices, reset hung devices, and abort operations when necessary. The following three sections describe these procedures.

9.7.1 Mounting and Dismounting Volumes

A SAVE Set can contain more than a single volume. Since you may be processing multi-volume SAVE Sets, the SAVE/RESTORE program contains procedures which allow you to mount the additional volumes in a set. When SAVE/RESTORE requires an additional volume to complete a SAVE or RESTORE operation, it prompts you with the following message:

Mount volume #nn of SAVE Set <setnam>
Device?

.
.
.

Proceed (Yes or No)?

where nn represents the volume sequence number and <setnam> indicates the SAVE Set Name and unit number of the drive on which the next volume has been readied. If you mount a device of the same type, you can respond to this prompt with only a unit number. After you press the RETURN key to terminate your device specification, SAVE/RESTORE performs volume label checking which is described in Sections 9.5.3 and 9.5.4. At this point, if the additional volume you mount violates any SAVE/RESTORE rule, the program prints the error message immediately after the DEVICE question.

SAVE/RESTORE prints a dismount message when:

1. SAVE/RESTORE exhausts a SAVE Set volume during a SAVE or RESTORE operation,
2. SAVE/RESTORE completes the data transfer to the output volume in an IMAGE copy operation, or
3. you mount an incorrect volume.

A dismount message generated for a SAVE Set differs from a dismount message printed for a RSTS/E disk. The SAVE Set format is illustrated under format 1 and the RSTS/E disk illustrated under format 2.

Format 1.

```
Dismount Device:  <device>
Set Name:         <setnam>
Seq #:           nn
Density:          [ 800]
                  [1600]
Creation Date:    <(day-of-week), dd-mmm-yy>
Expiration Date:  <(day-of-week), dd-mmm-yy>
```

Format 2.

```
Dismount Device:  <device>
Pack ID:          <packid>
Pack Clustersize: mm
Creation Date:    <(day of week), dd-mmm-yy>
```

The <device> field represents a two character device mnemonic, followed by a single digit unit number. The SAVE Set Name replaces <setnam>, and the disk pack identification replaces <packid> during a valid SAVE/RESTORE operation. Density figures are printed only for magnetic tape SAVE Sets. Finally, if SAVE/RESTORE has just written to the volume, it prints the PLEASE LABEL THIS VOLUME message immediately after the volume's dismount message.

9.7.2 Re-accessing Devices

During an operation, you may find that SAVE/RESTORE does not have access to a previously accessible device. If a device does become hung or is write-protected, the program notifies you in the following warning message:

```
%%% Device hung or write-locked
Retry (Yes or No)?
```

Correct the condition causing the problem if possible, then type YES to attempt to gain access to the device. Type NO to abort the attempt.

9.7.3 Aborting SAVE/RESTORE

To abort a SAVE/RESTORE operation at any time type CTRL/C. As a result you return to the control of your default run-time system if you invoked the program on-line. If you are running under the control of the INIT.SYS program and you abort, you return to the INIT.SYS OPTION: prompt. An abort during a SAVE, RESTORE, or IMAGE operation leaves the output volume unusable. On the other hand, you do not jeopardize any IDENTIFY volumes if you choose to abort this operation.

9.8 SAVE/RESTORE Summary Report

SAVE/RESTORE prints a summary report at the end of SAVE, RESTORE, and IMAGE operations. This report identifies the operation that was run, lists

characteristics of the input and output devices, and generates a list of data transfer, error and timing statistics. In addition, it documents the time and date of the operation. You receive the summary report automatically unless you suppress its output. This you can do if you specify the /NOSTATS option with a function response.

9.8.1 Summary Report Format

The format, in which SAVE/RESTORE prints the summary report, varies with the operation. At times the Input Device characteristics (see Figure 9-1) identify a SAVE Set, such as in the case of a RESTORE operation. In a SAVE operation, the input device characteristics represent those of a RSTS/E disk. The Output Device characteristics conform to the same pattern. Thus, depending on the operation, the SAVE Set format may be either represented under the Input or Output Device category. The RSTS/E disk format may be also printed under the Input or Output Device categories.

The general format of the SAVE/RESTORE Summary Report is illustrated in Figure 9-1.

Figure 9-1: Summary Report

```

SAVE
RESTORE      of      disk<packid>          disk<packid>
IMAGE        SAVE Set <setnam>          to    SAVE Set<setnam>

      Input Device:    <device>
      Pack ID:        <packid>
      Pack Clustersize: nn                      *RSTS/E disk format
      Creation date:   <day of week, dd-mmm-yy>

      [mastape]
      Output Device:   [disk]
      Set Name:        <setnam>                  *SAVE Set format
      # of volumes:    nn
      Density:         [800]
                      [1600]
      Creation date:   <day of week, dd-mmm-yy>
      Expiration date: <day of week, dd-mmm-yy>

      Run Statistics

      Transfer Totals:

      Total of nnnnnnnn blocks transferred

      Error Totals:

      Total of nnn bad compares.

      Total of nnn new bad blocks encountered on source.

      Total of nnn new bad blocks on destination disk.

      Total of nnn files structurally altered.

      Timing Totals:

      Total elapsed time:  nn hrs.,  nn mins.,  nn secs.
      Total wait    time:  nn hrs.,  nn mins.,  nn secs.
      Total process time:  nn hrs.,  nn mins.,  nn secs.

```


9.8.2 Summary Report Example

The following is an example of a Summary Report generated from a SAVE operation:

```

                          Summary Report

SAVE of DM1:INPUT to Save Set MYCOPY

      Input Device:  DM1:
        Pack ID:    INPUT
Pack Clustersize:  4
  Creation date:    Monday, 22-Jan-79

      Output Device:  Magtape
        Set Name:     MYCOPY
        # of volumes: 2
        Density:      1600
  Creation date:    Monday, 22-Jan-79
Expiration date:    Tuesday, 22-Jan-80

SAVE started on Monday, 22-Jan-79, at 11:49 AM

                          Run Statistics

Transfer Totals:

      Total of 23176 blocks transferred

Error Totals:

      Total of      0 new bad blocks encountered on source.

Timing Totals:

      Total elapsed time:  0 hrs.,  7 mins., 17 secs.
      Total wait   time:  0 hrs.,  0 mins., 19 secs.
      Total process time:  0 hrs.,  6 mins., 58 secs.

SAV/RES Function: ^Z

Ready
```

9.8.3 Summary Report Run Statistics

The run statistics that appear at the end of SAVE, RESTORE, or IMAGE operations are listed and described in Table 9-10. In addition, since each SAVE/RESTORE operation prints its own set of run statistics, Table 9-10 identifies the operation(s) in which each run statistic is printed. The operations are identified according to the following convention.

1. The letter (S) represents a SAVE operation.
2. The letter (R) represents a RESTORE operation.
3. The letter (I) represents an IMAGE operation

Table 9-10: Summary Report Run Totals

Type of statistic	Description	Operation
Transfer Statistics:		
Total of blocks transferred	This number refers to the total number of blocks transferred (directory blocks, file blocks). In a SAVE operation an extra set of directory blocks is stored at the end of a SAVE Set. These extra blocks are reflected in the figure for the total number of blocks transferred.	[SRI]
Error Statistics:		
Total of bad compares	The total number of errors which occurred during the VERIFY phase.	[I]
Total of new bad block errors encountered on source	The number of previously unreported bad blocks encountered on the source disk.	[SRI]
Total of bad block errors on destination disk	The number of previously unreported bad blocks encountered on the destination disk.	[RI]
Total of files structurally altered	The number of files which were structurally altered to enable relocation on the destination disk. Structurally altered means contiguous files were made non-contiguous or placed files were moved.	[RI]
Time statistics:		
Total elapsed time (TE)	The total amount of real clock time which elapsed during the run.	[SRI]
Total wait time (TW)	The amount of real clock time which elapsed while waiting for operator responses and magnetic tape rewinds.	[SRI]
Total process time (TP)	The amount of real clock time which was used for processing (TE-TW=TP).	[SRI]

9.9 SAVE/RESTORE Error Handling

SAVE/RESTORE issues dialogue, mount, transfer, and verification error messages. When you encounter an error message during a dialogue or in a mount phase, SAVE/RESTORE asks you to correct the condition, where appropriate, then instructs you to re-enter the command. Errors that you encounter during either of these phases never affect the integrity of the data being transferred. However, when SAVE/RESTORE reports an error during a transfer or verification phase, it has detected a problem that might affect the integrity of the data being transferred. Depending on the nature of the error, a single file or perhaps an entire volume might be affected. Generally, you will encounter less critical errors during a dialogue or mount phase than in a transfer or verification operation.

SAVE/RESTORE does not report an error when it finds a bad block on the output device unless it is unable to re-allocate that block in a way that is transparent to you. For instance, the one-block file SHIFT.LES will be moved without notice if a free cluster can be found. On the other hand, the program will issue an error message if it must move the placed file, MOVEME.NOT.

The /NOERROR switch directs SAVE/RESTORE to abort the run if it encounters any condition that warrants an error message. If you do not specify the /NOERROR switch, the program takes whatever action it deems necessary. SAVE/RESTORE always informs you of any corrective action taken so that you have the option of aborting the operation, if desired.

The most important problem that SAVE/RESTORE encounters during a RESTORE or IMAGE copy operation is finding a bad block on a RSTS/E file-structured disk when it wants to write to that block. The program attempts to rectify the problem by performing the following actions in the order listed:

- Mark a contiguous file as noncontiguous and relocate the bad cluster but not the entire file.
- Mark the file as non-placed if the bad cluster occurs at the beginning of a placed file.
- Find enough contiguous free pack clusters on the output disk to hold the number of clusters that must be moved. In other words, if a file's cluster size is 16, find 16 free contiguous blocks for the cluster. Proceed if the reallocation worked; abort if it has not worked.

SAVE/RESTORE always informs you of any changes it makes to a file that are necessitated by this bad block processing. If there is no way to relocate a cluster, the program aborts the operation. Unless you specify the /NOERROR, SAVE/RESTORE will never terminate an operation when it encounters a bad block that results in a non-fatal error (see Section 9.4).

9.9.1 General SAVE/RESTORE Error Messages

The errors that follow may occur during SAVE/RESTORE operations.

%%% Bad block in first n blocks of output volume

The current volume is unusable. Sufficient space must be available at the beginning of a SAVE Set to store certain critical information necessary for a SAVE/RESTORE operation.

Mount a different volume and try again.

??? Bad block in MFD

SAVE/RESTORE aborts the operation when it finds a bad block in the Master File Directory.

%%% Bad block in SATT.SYS

There is a bad block in the input SATT.SYS file. SAVE/RESTORE will not use an input disk which has a corrupt SATT.SYS file.

??? Bad block on mastape - can't continue

SAVE/RESTORE requires that a magnetic tape SAVE volume be long enough to contain a bootstrap, label information, and INIT.SYS. Whenever the program encounters a bad block before it transfers all required information, it aborts the current operation.

%%% Bad data in [MFD]

[<[P,PN] FILE.EXT>]

[UFD [P,PN]]

A cluster contained in the described account or file could not be relocated due to a fatal read error on the output volume.

??? Bad directory for device

SAVE/RESTORE has discovered a bad link or bad block in the input or output directory structure.

%%% Bad INIT.SYS on system disk

SAVE/RESTORE encountered a bad block in INIT.SYS while transferring it from the system disk to a SAVE volume.

%%% Can't mix device types in SAVE Set

A SAVE Set cannot be composed of both tapes and disks. Only all disk or all tape SAVE Sets are legal.

%%% Can't read bad block file

SAVE/RESTORE could not read the bad block file on a RSTS/E disk or a SAVE volume.

??? Can't read SAVE volume SATT

A non-recoverable I/O error occurred while attempting to read the Storage Allocation table on the SAVE volume.

%%% Can't write boot

The bootstrap could not be written to the first block of the output disk.

%%% Can't write INIT.SYS

SAVE/RESTORE encountered a fatal bad block while transferring INIT.SYS to SAVE volume.

%%% Device hung or write-locked

Retry (Yes or No)?

The output device is off line or is write-locked.

Correct the condition and try again.

Type Yes to continue or type No to abort.

%%% Device mnemonic must be specified

Enter the device name along with the unit number.

%%% Disk must be DSKINTed

When you are running SAVE/RESTORE on line the output disk must be RSTS/E file-structured or a SAVE volume. Use the DSKINT option of INIT.SYS to reinitialize the disk.

%%% Duplicate switches

You specified the same switch more than once.

??? Error in BADB.SYS allocation

SAVE/RESTORE was unable to rebuild the bad block file (BADB.SYS) upon completion of the current operation.

??? Fatal output bad block

A bad block was encountered at a critical position preventing SAVE/RESTORE from completing the current operation.

%%% Illegal command

Your response to a prompt was in an unacceptable format.

%%% Illegal command:

no output device specified

The full function line command you specified did not include an output device specification.

%%% Illegal date

The date you specified was either in an unacceptable format or was included in a function other than SAVE.

%%% Illegal density

The density you specified was in an unacceptable format, was included where none was acceptable, or specified an illegal density value.

%%% Illegal option

When booting from a SAVE volume, the only legal options are RESTORE, IMAGE, and IDENTIFY.

%%% Illegal switch

You used a switch in an unacceptable manner.

%%% Illegal switch combination

You specified conflicting switches.

%%% Input disk should be 'CLEANed'

The disk you mounted is "dirty". The SATT.SYS file may be corrupt. SAVE/RESTORE requires a valid SATT.SYS; therefore, it is recommended that you clean the disk using the REFRESH option of INIT or the ONLCLN program.

%%% Its expiration date has not passed

Mount it anyway (Yes or No)?

A SAVE Set on an output volume contains an expiration data that has not expired. SAVE/RESTORE prints the SAVE Set Name and expiration date before issuing this message.

Confirm whether you should use this volume or mount a different volume if appropriate.

%%% Mastape select error

Retry (Yes or No)?

When access to a magnetic tape drive was attempted, the selected unit was found to be off line.

%%% No default-type <cr> for more information

You typed a LINE FEED in response to a question which has no default.

??? No INIT.SYS on system disk

There is no INIT.SYS in account [0,1] on the system disk.

%%% No previous question - type <cr> for more information

You attempted to return to the previous question but it does not exist.

%%% Non-existent or hung device

You specified a disk or magnetic tape that does not exist on your system or is not on line.

Recheck your configuration or the device type and retry.

??? Non-recoverable I/O error

SAVE/RESTORE encountered a fatal error condition while trying to recover from a DEVICE HUNG OR WRITE LOCK error. The current operation is aborted and you are returned to the SAV/RES FUNCTION: prompt.

%%% Not a valid device

You specified an unacceptable device type in response to the question. This may occur if you specify a device other than magnetic tape or disk, if you specify a tape rather than a disk (or vice versa), or if you specify a wrong disk type on an IMAGE.

??? Operation aborted at user request

The user specified the /NOERROR switch and one of the errors described in Table 9-11 occurred or the user typed No to a RETRY (Yes or No) prompt.

%%% Same device specified

You designated the same device for both input and output.

Specify the correct output device.

%%% Tape won't respond to density XXXX

SAVE/RESTORE attempted to set the density of the specified magnetic tape to XXXX but the hardware rejected the request.

Specify a density setting that is legal for that device, or use a different device.

%%% Tape won't respond to any density

The input volume cannot be read at any legal density setting.

%%% This is not the correct volume

The SAVE Set or Pack Id you specified does not match the one on the mounted device. A DISMOUNT DEVICE: message follows the error message and identifies the volume if it is a SAVE Set or a RSTS/E file-structured disk.

Mount the correct volume.

%%% This volume has no label

SAVE/RESTORE did not find a valid SAVE Set label or Pack ID on the specified input volume.

%%% Too many bad blocks on input disk

The number of bad clusters on the input disk exceeds the RSTS/E limit.

%%% Too many bad blocks on output disk

The number of allocated clusters on the original source disk exceeds the number of available clusters on the output disk or the number of bad clusters on the output disk exceeds the RSTS/E limit (161 bad clusters).

Use a different disk pack.

%%% Unrecognized switch

The switch you specified is not a valid SAVE/RESTORE option.

??? Volume is full

There was not enough room for the relocation of data clusters due to the blocks encountered on the output volume.

%%% Wrong size disk

The output volume of an IMAGE or RESTORE must be the same size as the original source. (See discussion of LIKE disks in Section 9.2)

%%% You will be writing to the booted device

If you write to this device, you will need to reboot when you are finished with SAVE/RESTORE.

9.9.2 Transfer Errors--Fatal and Non-Fatal

Transfer errors that you may encounter in a SAVE/RESTORE operation can be either fatal or non-fatal. Fatal transfer errors cause an immediate termination of the current operation while non-fatal errors, if the /NOERROR switch is not in effect, allow the operation to continue despite the problem. On the other hand, if you specify the /NOERROR switch and one of the errors in Table 9-11 does occur, SAVE/RESTORE terminates the operation immediately. In other words, the non-fatal errors listed and described in Table 9-11 become fatal if the /NOERROR is attached to the function response.

The only fatal transfer error, i.e., regardless of whether the /NOERROR was specified, ???VOLUME IS FULL occurs if SAVE/RESTORE cannot relocate a file due to a bad block on the output device. The relocation did not take place because SAVE/RESTORE could not find any additional clusters available or did not find as many free clusters as it needed. This results from too many bad blocks on the output volume.

Table 9-11 contains a description of all non-fatal transfer errors (if /NOERROR is not specified).

Table 9-11: SAVE/RESTORE Non-Fatal Transfer Errors

%%% Bad block on input, PCN=nnnnnnn
A bad block was encountered on an allocated cluster on the input volume. A copied version of the data in that block might be corrupt.

%%% Bad block on input, output PCN's affected:
A bad block was encountered on the input SAVE volume which will affect certain pack clusters on the output disk. A copied version of the data in that block might be corrupt.

%%% Bad block on input in MFD [PCN=nnnnnnn]
in UFD [P,PN], [PCN=nnnnnnn]
in file [P,PN]FILE.EXT,[PCN=nnnnnnn]
A bad block was encountered on an allocated cluster on the input volume. A copied version of the data in that block might be corrupt.

%%% Bad Compare, PCN=nnnnnnn
The data on the input volume did not compare with the corresponding data on the output volume.

%%% Bad compare on input in MFD [PCN=nnnnnnn]
in UFD [P,PN], [PCN=nnnnnnn]
in file [P,PN]FILE.EXT,[PCN=nnnnnnn]
The data on the input volume did not compare with the corresponding data on the output volume.

%%% File XXXXXX has been changed to non-contiguous
SAVE/RESTORE found a bad block in file XXXXXX. In order to relocate the block, the program made the file non-contiguous.

Consider the consequences of this change and proceed accordingly.

%%% Placed file XXXXXX has been moved from aaaaaa to bbbbbb
A bad block was encountered in file xxxxxx. In order to relocate it, the first block of the file had to be moved.

Consider the consequences of this change and proceed accordingly.

Chapter 10

Using the Automated Patching Facility

The system manager uses the programs in the RSTS/E Automated Patching Facility to install patches. When generating a RSTS/E system, the system manager installs mandatory patches with the programs, as described in the *RSTS/E System Generation Manual*. As patches are published in the *RSTS/E Software Dispatch*, the system manager uses the programs to install required patches. When the system manager needs to create patching kits for several installations (or for customer-developed software), the programs can be used to create command files for the kits.

The most common use for the facility is installing patches published in the *RSTS/E Software Dispatch* and distributed on the RSTS/E autopatching kit. The program ONLPAT.SAV is used to install binary patches; the program CPATCH.BAS is used to patch ASCII source files. Instructions for using ONLPAT are in Section 10.1 and 10.2; for CPATCH in Section 10.3.

For system managers who need to create their own patching kits, information on building patching command files is contained in Sections 10.3 and 10.4.

10.1 Patching RSTS/E Binary Code – ONLPAT

The program ONLPAT.SAV is used to install patches to binary code. ONLPAT can run in keyboard mode to install individual patches to programs directly from a terminal, or can run in command-file mode, to install patches from a user-created command file.

10.1.1 Using ONLPAT in Keyboard Mode

When ONLPAT is used in keyboard mode, procedures are similar to those used for the PATCH option of the initialization code (INIT.SYS). There is one important exception: no patch is installed until the system manager types uparrow C (↑C) at the end of the last line in the patch. ONLPAT allows the system manager to review the entire patch before it attempts to install it.

The procedure for using ONLPAT in keyboard mode is:

1. Run ONLPAT from any logged-in terminal by typing:

```
RUN $ONLPAT
```

The program responds with the question:

```
COMMAND FILE NAME?
```

2. Type RETURN to use only the terminal as a log device, or type the name of a file (or device) to cause a log file of the patch to be created (as well as go to the terminal). If a file extension is not specified, the default is .LOG. The file can be any file in the public structure. Type an equal sign (=) after the file name, then press the RETURN key.
3. Patch the file, following instructions in the *RSTS/E Software Dispatch*. Remember that the patch will not be installed until it is ended by typing ^C (uparrow C or on some terminals ^C).

NOTE

Do not type CTRL/C to terminate a patch: CTRL/C terminates ONLPAT without making any patches.

The following ONLPAT message indicates that the patch has been installed:

```
PATCH COMPLETE  
1 PATCH INSTALLED
```

4. ONLPAT then returns the question:

```
COMMAND FILE NAME?
```

Install another patch, or terminate the program by typing CTRL/C.

10.1.2 Using ONLPAT in Command-File Mode

When ONLPAT is used in command-file mode, a previously-created command file containing the patches is executed by the program. This is the method used by DIGITAL when generating a new RSTS/E system using the Automated Patching Facility kit.

The procedure for using ONLPAT in command-file mode is:

1. Ensure that the command file medium is mounted.
2. Run ONLPAT from any logged-in terminal, by typing:

```
RUN $ONLPAT
```

The program responds with the question:

```
COMMAND FILE NAME?
```

3. Respond with

```
logfile=command file
```

where logfile is the name of a file for recording the patches and command file is the name of the previously-created command file.

The default extension for logfile is .LOG; for command file, the default extension is .CMD.

If logfile is not specified, no logfile is produced.

4. ONLPAT executes the command file. If so specified in the command file, ONLPAT may pause, to allow the system manager to enter the name of a file to patch.
5. When the processing has completed, ONLPAT prints a summary of patches installed (and skipped, if any), then returns to command level.

10.1.3 Patching a Running Monitor with ONLPAT

Patches made using ONLPAT are to the file which is the installed monitor Save Image Library. That is, patches are made to disk and only become effective after the system has been shutdown and restarted.

However, patches to monitor overlay code could cause problems. Therefore, ONLPAT detects when the file being patched is the monitor SIL, and does not install the patch when:

1. The patch involves more than one overlay block. ONLPAT prints the message:

```
PATCH TO MODULE OVR TOO LONG FOR INSTALLED SIL
```

2. The patch involves both OVR and some other module of the monitor. ONLPAT prints the message:

```
PATCH TO INSTALL SIL SPANS OVR AND OTHER MODULE(S)
```

The only way to install a patch of this type is to shut down the system and either use the PATCH option to the initialization code, or install another monitor SIL (such as the SYSGEN SIL) and patch the target SIL while it is not the running monitor.

10.2 Building ONLPAT Command Files

The procedures for creating ONLPAT command files is:

1. Run ONLPAT from any logged in terminal by typing:

```
RUN $ONLPAT
```

The program responds with the question:

```
COMMAND FILE NAME?
```

2. Respond with

Patchfile = KB:

where patchfile is the name of the patch file you are creating and KB: indicates you are creating the patch file from the keyboard.

3. Press the RETURN key and ONLPAT asks the following question:

FILE TO PATCH?

Type the name of the file you want to patch, press the RETURN key, then enter the text of the patch.

After you create the command file, you may find it necessary to modify it. To do this, apply the following rules.

1. Individual command lines in a patch always end with a carriage return.
2. When it executes the command file, ONLPAT forces all lower case characters to upper case, and reduces all tabs and spaces to a single space.
3. Blanks are significant. Use them exactly as shown for individual command lines.
4. The first lines of an individual patch can be comment lines. A comment line begins with an exclamation mark (!) and can contain any other ASCII data.

5. The next line in a patch is optional, but if present, must be entered as

OPTION: PATCH

6. The next line is required; it specifies the file to be patched. The format is:

FILE TO PATCH? *filenam.ext*

where *filenam.ext* is the file to be patched. Omit the space and *filenam.ext* to cause ONLPAT, when executing, to prompt for a filename.

The filename used here also is used for any subsequent patches in the command file that do not specify the *filenam.ext*.

7. Subsequent command lines in a patch are identical to the formats used in the Software Dispatch. The command line

BASE OFFSET OLD NEW

is optional, but when used must appear with spacing exactly as shown in the Software Dispatch articles. Also, do not use any parenthetical comments.

8. Notation must be exact for these constructs:

↑Z is uparrow, then Z (not CTRL/Z)

↑C is uparrow, then C (not CTRL/C)

<lf> is <, then l, then f, then > (not LINE FEED key)

10.3 Patching ASCII Source Code – CPATCH

The CPATCH program is used to install patches to the source code of the BASIC-PLUS library programs. CPATCH can be used to patch individual programs, as described in this section, or can be used to build patching command files, as described in Section 10.4.

The procedures to patch a BASIC-PLUS source file are:

1. Run the program from a privileged account by typing:

```
RUN $CPATCH
```

After printing a header line, CPATCH prints the prompt:

```
FILE TO PATCH-
```

2. Respond with:

```
outfilespec=infilespec
```

where outfilespec is any specification desired and infilespec is the specification for the unpatched source file. The default extension for both files is .BAS. If no output file is specified, it is assumed to be the same as the input file, and is given a .BAK extension.

3. CPATCH next prints a prompt (#) indicating that it is ready to patch the source program. Type the RETURN key.
4. CPATCH then prints an asterisk prompt (*) indicating that source editing can begin. Edit the source program, using the instructions in the *RSTS/E Software Dispatch*.
5. When the edit is completed, type an EXIT in response to the asterisk prompt. CPATCH indicates that it has installed the patch by printing:

```
PATCH FROM file specification COMPLETE.
```

Then CPATCH returns to the # prompt. If there are other patches to the current file, do them now, or type CTRL/Z.

6. If CTRL/Z was typed, CPATCH returns to the prompt:

```
FILE TO PATCH-
```

If there are no more files to patch, type CTRL/Z to exit from CPATCH. If there are other files to patch, resume at step 2.

7. Now, follow *RSTS/E Software Dispatch* instructions for including the patched source program in the system library.

10.4 Building CPATCH Command Files

This section describes the steps to create new patch files to use with CPATCH. These patch files are used by CPATCH to perform the actual edit of the source programs.

10.4.1 Building the Patching Command File

The first step in creating a new patch is to create a CPATCH command file by running CPATCH stand alone. The keyboard is used for command input and a file is used as the log. The log file produced will be the new CPATCH command file.

10.4.1.1 File Naming Convention — Patch files will be much easier to manage if a consistent file naming convention is adopted. The convention used in the Release Notes for all patches to the BASIC-PLUS System Library is presented here as an example.

All CPATCH command files correspond to an article in the *RSTS/E Software Dispatch*. The patch file names are based directly on the article sequence number. They are of the form:

PA`nnmm.kkk`

where `nn` is the component number, `mm` is the subcomponent number, and `kkk` is the number of the article corresponding to that patch. There will always be an article corresponding to all command files. However, there are articles which will have no corresponding patches (notes, restrictions, etc.).

A typical example might be the program QUMRUN. Assume the first article in the Software Dispatch concerning QUMRUN describes a patch. The article's sequence number is 14.6.1. The number 14 is the component number (Spooler and Operator Services Package), 6 is the subcomponent number and 1 indicates it is the first article published about QUMRUN. The corresponding patch file name would be PA1406.001.

10.4.1.2 Creating the CPATCH Command File — The CPATCH command file is created by running CPATCH. The file for which the patch file is being created is specified at the FILE TO PATCH prompt, and the keyboard is specified as the command file with the CPATCH command file being created as the log file. The patch is then manually applied as it would be using EDIT. Note, however, that CPATCH is a line editor. That is, the CPATCH text buffer contains only a single line of text at any one time. When the patch is completed, the log file will contain a CPATCH command file which will perform the same patch when applied using CPATCH. As an example, the commands to create the CPATCH command file to perform the first patch to QUMRUN.

The first step is to start the program by typing:

```
RUN $CPATCH
```

After a header line CPATCH will print the prompt:

```
FILE TO PATCH-
```

The response to this prompt would be:

```
out:QUMRUN.BAS=in:QUMRUN.BAS
```

The string in: would actually be typed as the device/PPN which contains the unpatched source file QUMRUN.BAS. Out: can be any device/PPN. It is recommended that a different device/PPN from the input be selected. The patched program can then be tested before replacing the original source.

CPATCH now issues the # prompt. The correct response for the example patch is:

```
PA1406.001=
```

This command specifies that PA1406.001 should be used as the log file. No command file name is provided. This causes the default command file (KB:) to be used.

CPATCH now prints the * prompt. You should now type the EDIT commands to perform the patch on the file. The commands are entered as they would be for EDIT run with the /PATCH switch. (Note the restrictions on the commands described below.)

When the patch is complete, use the EX command to return to the # prompt. A CTRL/Z will return the FILE TO PATCH prompt, and another CTRL/Z will exit the program. The file PA1406.001 will now be the CPATCH command file required to perform the patch.

The CPATCH program provides two switches to verify Patching operation accuracy. To ensure that CPATCH applies a patch correctly you must specify the /CS switch while creating a patch file and the /CS:n switch when you are applying the patch.

To create a patch file, invoke CPATCH and respond to the FILE TO PATCH prompt. After you enter the name of the file to patch and press the RETURN key, CPATCH prints a pound sign prompt (#). Since you are creating a file to store the patches, enter its file specification followed by an equal sign, attach the /CS switch, and press the RETURN key. CPATCH then prints an asterisk prompt to indicate its readiness to accept the patch.

After you enter the patch and terminate your input by typing EX, CPATCH calculates a number, called a checksum, and prints it on your terminal. This is the number that CPATCH will use, while applying the patch, to verify that it was correctly incorporated. CPATCH marks the end of the operation with a PATCH COMPLETE message and returns to the pound sign prompt. At this point, you can enter the name of a new patch file or type CTRL/Z to return to the FILE TO PATCH prompt. If you also type CTRL/Z to the FILE TO PATCH prompt, you terminate CPATCH and return control to your default

run-time system. The following terminal output illustrates the use of the /CS switch and the creation of a checksum number.

```
RUN $CPATCH
CPATCH  X7.0 RSTS V7.0 TIME SHARING
File to Patch -  PATCHA.BAS=PATCHA.BAS
#PA2325.001=/CS
*G/-10/-1D
*EX
Checksum = 53437
Patch from _KB:[1,247]CPATCH.CMD complete.
#CTRL/Z
File to Patch -  CTRL/Z

Ready
```

After creating the patch file, you can use CPATCH again to apply the patch. As before, invoke CPATCH, specify the file to patch and press the RETURN key to receive the pound sign prompt. In response, attach the /CS:n switch, replacing n with the checksum number that CPATCH calculated previously (53437). Press the RETURN key, enter the patch, and exit by typing EX. If you enter the patch correctly, CPATCH will calculate the same checksum number, apply the patch, and print the PATCH COMPLETE message. Otherwise, it will print a message such as the following.

```
?Actual checksum of 53437 does not match 53537
%Patch from _KB:[1,247]CPATCH.CMD skipped.
```

When these messages appear, you must retrace your steps and try again. An example of a successful attempt to apply a patch is illustrated as follows:

```
RUN $CPATCH
CPATCH  X7.0 RSTS V7.0 TIME SHARING
File to Patch -  PATCHA.BAS=PATCHA.BAS
#/CS:53437
*G/-10/-1D
*EX
Patch from -SY:[1,247]PA2325.001 complete.
#CTRL/Z
File to Patch -  CTRL/Z

Ready
```

The /CS:n switch, n representing a number from 0 to 65535, instructs CPATCH to recalculate the checksum as a result of your new input and compare it to the original calculation. In this case, you entered the patch accurately; CPATCH was able to calculate the same checksum and then apply the patch.

When the keyboard is the command file, type all commands exactly as they would be typed to EDIT. To prevent ambiguities when a file is used for command input, CPATCH requires all "invisible" characters, (ESCAPE, LINE FEED, etc.) to be translated to a visible form. This translation is always done on log file output to permit log files to be used for command input with no modification. Because the output is used for patch verification, it must be typed exactly. An error causes a rejection of the patch.

The translation is listed below:

Control Characters	Command File Format
ASCII 9 (horizontal tab)	<TAB>
ASCII 12 (form feed)	<FF>
ASCII 27 (escape)	<ESC>
ASCII 10, 13, 0 (line feed sequence)	<LF>
ASCII 13,10 (carriage return sequence)	<CR>
ASCII 0 (isolated null character)	<null>
ASCII 13 (isolated carriage return character)	<13>
ASCII 10 (isolated line feed character)	<10>

NOTE

CPATCH contains a line editor (AUTOED) for performing the patches. The editor uses the same command syntax as EDIT on RSTS/E, however, the command set is smaller and some of the commands are restricted in the arguments permitted. The commands and their restrictions are listed below:

Commands	Restrictions
I, G, H, V, and EX	Identical to EDIT.
A, and K	No negative arguments are permitted.
J, D, and C	As in EDIT, but the command will not affect any text beyond the current line. For example, "-400J" will only back the pointer to the beginning of the current line.
L	Only the arguments 0, 1, or no argument (default to 1) are permitted.

In addition to the above restrictions, no special character arguments such as @, or / are supported.

You create a colon-modified command when you prefix the G, H, V, or L command with the colon (:) character. When you precede one of these commands with a colon and

1. a search fail error occurs while executing the colon-modified command, or
2. a mismatch occurs with text in a log file,

a warning rather than a fatal error message is printed and the patch is skipped. This feature is useful when you are searching or verifying the patch level of a program to which the patch is to be applied. If the program is not already at the proper level, the patch will be skipped and no fatal error message generated. Finally, if a numeric argument is used with a colon-modified command, the colon must be positioned between the argument and the command, for example, O:L.

10.4.2 Building the PBUILD Command File

After building a command file of patches, the next step is to build another command file that will contain the instructions to PBUILD for performing the patches.

10.4.2.1 Using Comments — Comments may appear anywhere in a PBUILD command file. Any line of text with ! as its first character is treated as a comment. If ! is used as an account specifier it must be in a column other than the first. One or more blanks before the ! is sufficient.

10.4.2.2 Using Indirect Command Files — Any command line beginning with an @ is used as a file which will contain the next commands. The position in the current command file is saved, and all subsequent input comes from the file specified until an end-of-file is reached. Command input then resumes from the saved position in the command file. Indirect command files may be nested up to 15 levels deep.

The following rules must be observed when using indirect files. First, the default extension for indirect command files is .CMD. Therefore any other extension must be explicitly stated. Second, if either the device or the PPN is absent from an indirect file name, the device and/or PPN is obtained from the command file name at the previous level of indirect. In order to avoid ambiguity in device specification, it is highly recommended that all command files reside in the same account and that they do not specify a device/PPN. Rather, the device/PPN for all indirect command files should be specified at PBUILD's # prompt, setting the default value for all subsequent indirect command files.

10.4.2.3 Using Underscore (_) as a Quote Character — You can quote any character or command with the underscore (_) character. For example, if it is necessary to have the text "\$FORCE" not interpreted as a \$FORCE command, prefix the command with the underscore character as follows: _\$FORCE. In doing so, you alter the way in which the system responds to data following the underscore.

10.4.2.4 Command File Statements — There are four types of PBUILD command file statements:

\$FORCE	Forces command lines to the keyboard, until the next \$PATCH or \$END statement.
\$PATCH	Identifies the file to be patched.
patchfile	Contains the specification for a patching file.
\$END	Indicates the end of patching for one file.

NOTE

Comment lines can appear anywhere in a PBUILD command file.

To patch a file, the PBUILD command lines must be in this order:

```
$PATCH statement
patchfile statement
.
.
.
or
patchfile statement
$END statement

$FORCE statement
text to be forced
$END statement
```

The \$FORCE Statement – The \$FORCE statement is not required for patching BASIC-PLUS programs. Its purpose is to allow you to batch commands into the file (to run programs, delete files, etc.). For example, ONLPAT is run in this manner to patch binary files.

If used, \$FORCE and its related command lines must be placed outside the bounds of a \$PATCH/\$END set, either before a \$PATCH statement or after an \$END statement.

The command lines following a \$FORCE statement use logical names to allow device/PPN assignments (those made during the PBUILD dialogue). The substitutions are:

~SYSTEM:	Is the value for the device/PPN specified as the library device.
~SYSDSK:	Is the value for the device/PPN specified as the library or the system device.
~INPUT:	Is the value for the device/PPN specified as the unpatched source file.
~PATSAV:	Is the value for the device/PPN specified as the patched source file. If sources are not to be saved, O: defaults to the L: value.
^ Z	CHR\$(26%)(CTRL/Z)
^ C	CHR\$(3%)(CTRL/C)

The \$FORCE statement is in effect until a \$PATCH statement or end-of-file is encountered. Any command line beginning with an @ is treated as an indirect command file.

The \$PATCH Statement – The \$PATCH statement is the first statement in a sequence of statements to perform a patch. The format of the statement is:

```
$PATCH filename
```

The filename must be present. The default value for the extension is .BAS and the default device/PPN is that which was specified to the opening question of the dialogue. If an extension or a device/PPN is specified with filename, that which is specified in the filename will override the default. For example, suppose the input device entered at the beginning of the dialogue

was MM:[120,51]. The list below shows various possible values for filename in the \$PATCH statement, and the resulting filename after filling in the defaults:

\$PATCH filename	name actually used
TEST	MM:[120,51] TEST.BAS
TEST.	MM:[120,51] TEST.
SY:TEST	SY:[120,51] TEST.BAS
SY:[1,2] TEST.FIL	SY:[1,2] TEST.FIL

The statement consists of a single filename which is the name of a command file to be used by CPATCH to perform the actual source text editing. Any number of Patch file statements is permitted. There will be one Patch file statement for each patch to the program.

The default extension for the Patch files is .CMD. The default device/PPN is that of the current PBUILD command file which is open. Again, to avoid confusion it is recommended that all command files, including indirect files and patch files, be on the same account, and that no device/PPN be specified within any command files.

\$END Statement – The \$END statement indicates the end of a patch. It also selects a number of options for the operations which override those entered in the dialogue. The format of the end statement is:

```
$END [/NC] [newname]
```

When the \$END is encountered, PBUILD normally performs the OLD and COMPILE operations. However, the operation can be changed by providing either (but not both) of two optional arguments. The first is the /NC switch. When this switch is provided, no compile operation is performed. The patched source is saved on the account specified as the library in the opening dialogue.

The newname option allows the name of the file to be changed after it is compiled. The default name for the compiled program is L:file.BAC. L: is replaced by the library device and PPN specified in the opening dialogue and file is the filename specified in the \$PATCH statement. The protection code is always 124. If any of these values are to be changed, their new value is included as the newname argument. Only those values which will be different from the default must be included. For example, if the file's name is to be ~SYSTEM:file.BAC<232>, only the protection code is different from the default, therefore the argument would be <232>.

If it is desired to have a file compiled to the system device instead of the library device, the newname option is used. PBUILD will substitute the system device/PPN for the string ~SYSDSK:. For example, if the system device specified in the opening dialogue is SY0:[1,2], the new name string ~SYSDSK:<232> would result in the file name SY0:[1,2]file.BAC<232>.

10.4.2.5 Sample PBUILD Command File — The CPATCH file created in Section 10.4.1.2 must be referenced by a PBUILD command file. A new PBUILD file can be created with the required PBUILD commands or the commands can be inserted in an existing file. In either case, the file is edited using an editor.

In the example, the CPATCH file for the first patch to QUMRUN was created. Now the commands to reference this file are inserted in the PBUILD file (either new or existing). The commands required are:

```
$PATCH QUMRUN
PA1406.001
$END
```

Since this is the first patch to QUMRUN, all three of the commands must be added. If there were already existing patches to QUMRUN, the reference to the new patch must be added before the \$END statement of any previous patches to QUMRUN. For example, if the second article concerning QUMRUN is also a patch article, the CPATCH command file for that patch would be PA1406.002. A patch already exists for this program (PA1406.001), therefore PA1406.002 should be inserted immediately before the \$END statement in the sequence of commands which apply the previous patch. The resulting command sequence would be:

```
$PATCH QUMRUN
PA1406.001
PA1406.002
$END
```

These commands would cause PA1406.001 to be applied to QUMRUN. The file PA1406.002 would be applied to the patched file resulting from first applying PA1406.001.

10.5 Running the PBUILD Program

Once the patching command file and the PBUILD command file have been built, run the PBUILD program to install the patches.

To begin, type RUN \$PBUILD. The program begins a dialogue with the prompt:

```
READ FILES TO PATCH FROM <SY:[1,2]>:
```

PBUILD is requesting the device and account which contains the unpatched versions of the text files. The device/PPN in angle brackets is the default

value. If either the device or the PPN is incorrect, you should enter the correct values. In this case, respond in the following format:

```
logfile=<device>:(P,Pn)[/DETACH]
```

where

1. logfile is the name of the file in which the entire patching procedure will be recorded,
2. <device> represents the input device name and unit number,
3. (p,pn) is the project programmer number, and
4. the /DETACH option instructs PBUILD to detach from the keyboard while performing the patching operation. If you specify both a log file and the /DETACH option, some patching operations will be faster since they will not be limited by the speed of the terminal.

For example, if the files are on the public structure under account [120,5], the correct response would be [120,5]. If the files are on magnetic tape drive MM0: on account [1,2], the correct response would be MM0:. If both the account and the device are wrong, both would have to be entered, such as MM0:[120,5]. If both are correct, then just a carriage return is entered.

After the input device has been specified, PBUILD asks

```
COMPILE PATCHED PROGRAMS<YES>:
```

You may request the resulting patched files be compiled by accepting the Y(ES) default. Press the RETURN key to accept the default and thereby initiate subsequent patch questions. Type NO and the next three patch questions are not asked.

If the files are to be compiled, PBUILD asks where to store the compiled programs with the following questions.

```
LIBRARY DEVICE <SY:[1,2]>:  
SYSTEM DEVICE <SY0:[1,2]>:
```

The library device is the device and account which will be used for the programs which will normally go to the library account. The system device is the particular device which contains the RSTS/E system. The compiled programs normally go to the library account.

The next question asked is:

```
SAVE PATCHED SOURCES<NO>:
```

If the patched sources are to be saved, you should answer Y(ES) to this question. If any other answer is given, the next question is skipped and the patched sources are deleted after compiling.

If the patched sources are to be retained, the program asks

```
WRITE PATCHED SOURCES TO <SY:[200,200]>:
```

The device and account for the patched sources is entered. DIGITAL recommends that the patched sources not be placed in account [1,2].

When PBUILD issues the pound sign (#) prompt, respond to it in the following format:

```
logfile=commandfile[-]
```

CPATCH uses the logfile to write the log of the actual text file edits as they take place. If logfile is absent, the file name KB:PBUILD.CMD is used. If a file is specified, the default extension is .LOG. The commandfile field is the name of the file containing the commands to PBUILD. The default extension is .CMD. In both the log and command files, the default device is SY: and the default PPN is the current user account. As an option, you can place a hyphen after the commandfile specification to guarantee that PBUILD will prompt you for additional command files. Since the prompting of the additional control file question is also dependent on other factors, PBUILD may ask for additional control files if you do not attach a hyphen to the commandfile specification. After you terminate the logfile=commandfile[-] specification with a LINE FEED, PBUILD chains to the BUILD program. The ensuing dialogue sequence is now under the control of the BUILD program and thus BUILD will be referenced hereafter rather than PBUILD.

Once a correct command file name has been entered, BUILD prints a message to indicate it is copying the specified commandfile to the public disk structure. The variable [nnn,nnn] represents the command file account number and the nn in the PBLDnn.TMP specification is the job number. The format of the message is:

```
*** COPYING FILE dev:[nnn,nnn]commandfile TO SY:PBLDnn.TMP ***
```

BUILD then preprocesses the command file. It is usually at this point, during the preprocessing of the first command file, that BUILD prints a run-time system question. In effect, BUILD is asking you for the name of the run-time system that you want the executable programs to run under. The question has the following format:

```
Run-Time System<xxxxxx>?
```

BUILD prints the name of the system default run-time system in the default field, <xxxxxx>. Press the LINE FEED key to accept the default or respond with the name of another run-time system. If you decide not to accept the default, note that the run-time system you specify must be a keyboard monitor and have a default executable extension of .BAC or .TSK. If BUILD does

not accept your response, it notifies you of an error by printing one of the following messages:

1. XXXXXX IS NOT A KEYBOARD MONITOR
2. RUN-TIME SYSTEM MUST HAVE A DEFAULT EXECUTABLE EXTENSION OF .BAC OR .TSK
3. XXXXXX IS NOT INSTALLED
ATTEMPT TO INSTALL XXXXXX?

After BUILD prints either of the first two messages, it reprompts the RUN-TIME SYSTEM question. Enter a new run-time system name that conforms to the run-time system restrictions indicated in the error message. If BUILD issues the XXXXXX IS NOT INSTALLED message, it is indicating that it has found the specified run-time system in account [0,1], determined that it has a .RTS extension, but has also discovered that the run-time system is not installed. Immediately after printing this message, BUILD asks, in the following question, if you want it to install the run-time system.

ATTEMPT TO INSTALL XXXXXX<No>?

The XXXXXX field represents the run-time system BUILD will attempt to install. Type NO if you want BUILD to return to the RUN-TIME SYSTEM prompt. Type YES and one of the following situations occurs:

1. BUILD installs the run-time system successfully and prints the message, XXXXXX INSTALLED. BUILD asks the CUSP COMPILER question, described below, if you selected RSX or BP2COM as your run-time system.
2. BUILD installs the run-time system, determines whether it is a keyboard monitor and has the correct extension, then prints one of the following two error messages:
 - a. XXXXXX IS NOT A KEYBOARD MONITOR
 - b. RUN-TIME SYSTEM MUST HAVE A DEFAULT EXECUTABLE EXTENSION OF .BAC OR .TSK.

After printing either of these messages, BUILD informs you that it is removing the previously installed run-time system by printing the XXXXXX WILL BE REMOVED message. The program immediately returns to the RUN-TIME SYSTEM question.

3. BUILD may detect errors other than those already described. When it does, it prints the message:

RUN-TIME SYSTEM OPERATION FAILED<text>

The <text> field can be any valid RSTS/E error message. After printing the error, BUILD returns to the RUN-TIME SYSTEM prompt.

If you selected either the BP2COM or RSX Run-Time System, BUILD asks the following CUSP COMPILER question:

USE THE CUSP COMPILER 'CSPCOM'<YES>?

Type YES if your run-time system is RSX; type NO if it is BP2COM.

After the preprocessing of the first command file, BUILD may prompt you for the name of another command file by printing the following question:

ADDITIONAL CONTROL FILE IS <Process file>?

Type another command file specification and terminate the response by pressing the LINE FEED key. Like the previous command file procedure, BUILD prints the COPYING FILE message and then preprocesses the file. BUILD will continue to print the ADDITIONAL CONTROL FILE IS question until the default is selected. Press the LINE FEED key to accept the default. This response instructs BUILD to begin executing the command files. If, however, no hyphen was attached to the original commandfile specification, PBUILD may not ask for additional control files. Therefore, if you need to process more than one command file, attach a hyphen to the logfile=commandfile[-] specification to ensure that BUILD does ask the ADDITIONAL CONTROL FILE question. Note that PBUILD prints the COPYING FILE message, preprocesses the command files, but does not execute any command file until all files are specified.

During the preprocessing phase, BUILD may ask additional dialogue questions. These questions cannot be enumerated or anticipated here since they are initiated by the particular command file that BUILD is processing. Since there are many control files that might be processed, each with its own unique set of questions, you must refer to documentation regarding that command file in order to answer these questions. For example, if you are patching the Spooling Package, the question LOCATE SPOOLING PACKAGE ON <SY:[1,2]> might appear. This question asks where the Spooling Package resides.

PBUILD/BUILD performs all patch operations by printing the required commands on the pseudo keyboard and echoes them to the user's terminal. Every operation causes an output to the terminal to provide an accurate log of all operations. The steps that occur in a patch operation are summarized below. Note that the commands described are those which are printed on the pseudo keyboard and thus require no user input.

1. RUN \$CPATCH

This runs the CPATCH program.

2. outfile=infile

This string is typed to the prompt FILE TO PATCH issued by CPATCH. Infile will be the program to be patched, with the input device /PPN being that entered at the beginning of the dialogue. Outfile can be one of two possible values. If the patched sources are going to be saved, and the device is a file-structured disk, outfile will be the same as infile but with the device/PPN selected in answer to the WRITE PATCHED SOURCES TO question. If the sources will not be saved, or if they will be saved on a device other than disk, the device/PPN will be the library account.

3. logfile=commandfile

This is typed to the CPATCH # prompt. Commandfile is a command file for CPATCH which will perform the patch to the text file. The filename is obtained from the command file currently being executed by PBUILD. If the name as specified within the command file is missing either a device or PPN specification, the device and/or PPN which contained the command file executing is used. Logfile is the logfile entered by the user in response to the PBUILD # prompt. This step is repeated for each patch to the particular source program.

4. ↑Z (CTRL/Z)

This is forced to the keyboard after all patches have been applied. It causes CPATCH to go to the FILE TO PATCH prompt.

5. ↑Z

A second CTRL/Z causes the program to return to the RSTS/E keyboard monitor.

6. OLD outfile

This is typed only if the compile option was selected in the dialogue. It causes the source file to be compiled.

7. COMPILE comfile

This is typed if the compile option was selected. It causes the compiled version of the program to be saved as the file named comfile. Comfile will normally have the same file name as outfile but with the device/PPN being the library device entered in the dialogue. If a new name was provided in the PBUILD command file, that name will appear as comfile.

8. RUN [1,Z] PIP.SAV
*<Permfile>=<outfile>
^Z

If the patched sources are to be saved, and the save device is not a disk, then these commands are printed on the pseudo keyboard. Permfile is the same file name as outfile but with the device/PPN input to the WRITE PATCHED SOURCES TO question.

```
9. RUN [1,2] PIP.SAV
   *<outfile>/DE:NOWARN
   ^Z
```

This command is used to delete the output file which was created in the library account. It will only appear if the previous copy operation was performed or if the current patched sources were not to be saved.

```
10. RUN [1,2] PIP.SAV
    *<backfile>/DE:NOWARN
    ^Z
```

If outfile and infile are the same file name, CPATCH will create a backup file backfile. If that occurs, this command is typed to delete it.

In a normal patch operation only some of the CPATCH commands will actually appear. However, CPATCH does perform some combination of the above commands for each patch in the PBUILD command file.

10.6 Error Messages

This section describes the errors that may occur when running the patching programs. The error messages are divided into two sections, those originating in the program PBUILD, and those originating in CPATCH. In addition, those from CPATCH are classified as those which can occur when CPATCH is run by PBUILD and those which can occur when running CPATCH stand alone.

Errors can occur at three different times in PBUILD. The most likely is invalid user input while entering information required to run the program. Once the data is correctly entered, PBUILD begins to run from a command file. An error can be caused by a command that was incorrectly entered in the command file, or by incorrect specification of the source file input account in the dialogue. Finally, CPATCH, which is run by PBUILD, could attempt to apply an invalid CPATCH command file as a patch to a program, which will also cause an error.

In the first case above, the incorrect input is displayed with a text error message, and the prompt requesting the input is reissued. In the last case, however, the program will continue to run. Errors occurring in PBUILD while it is running from the command file are always fatal. PBUILD will print the error message and exit to the default run-time system command level.

CPATCH error messages will appear at the user's terminal. If they occur while running under PBUILD, PBUILD will continue to run regardless. If they occur running CPATCH stand alone, they may cause CPATCH to terminate execution.

When running CPATCH under PBUILD using patch files supplied by DIGITAL, the first error message in Table 10-2 is the only one which should ever occur.

Table 10-1: PBUILD Error Messages

Message and Meaning
<p>MUST BE PRIVILEGED TO RUN PBUILD An attempt was made to run PBUILD from a non-privileged account.</p>
<p>?FILE NAME ERROR Usually occurs while PBUILD is detached. Indicates that a bad filename was found in the command file currently executing.</p>
<p>?BAD FILE NAME: <string> <string> is not a valid file name. Can occur during the dialogue or while PBUILD is detached.</p>
<p>?<string> NOT PERMITTED <string> is a valid RSTS/E file string, but contained a feature which is not permitted for the operation, such as wild card characters, or a filename when only a device /PPN was expected.</p>
<p>?FILE NOT FOUND: <string> <string> is a valid filename, but names a nonexistent file.</p>
<p>?<string> ERR-<num> <string> was an invalid file string for a reason other than the previous 3 error messages. The <num> variable represents the error number produced by the BASIC-PLUS Run-Time System.</p>
<p>?COMMAND FILE MUST BE ON DISK A command file name was entered with a device specifier other than a file-structured disk. Command files must be copied from the device to disk before running PBUILD.</p>
<p>?CAN'T COPY <file1> TO <file2> <BASIC-PLUS error message> Will occur after the message !COPYING TEMP FILE <file1> TO <file2> while PBUILD is detached. PBUILD was attempting to copy the source file file1 from the current account to the permanent account. The copy failed for the reason given in the BASIC-PLUS error message. The BASIC-PLUS errors which are most likely are ?NO ROOM FOR USER ON DEVICE and ?DEVICE NOT AVAILABLE. Others are various magnetic tape errors or hardware I/O errors.</p>
<p>FATAL ERROR NO. <err> ON LINE <er1> LEAVING PROGRAM The program was terminated by an unexpected error. The <err> variable represents the BASIC-PLUS error number. The error may indicate a system error or a hardware failure.</p>

Table 10-2: CPATCH Error Messages

Message and Meaning
<p>?MISMATCH</p> <p>Occurs when editing a version of the file which was not correct. For example, attempting to patch a file which was already patched. When this occurs, the patch is ignored, and the file is restored to the state before attempting to apply the patch. The log file (or KB: if that is the log) will show the cause of the mismatch by printing the following:</p> <pre> =cmd=> string =log=> string! ?mismatch? </pre> <p>String is the contents of the command file which is the correct program response to the previous command in the command file. String! is the actual program response produced. It can be a verify on a line which has been modified, or a CPATCH error message, such as ?SRCH FAIL when a string which was expected has been deleted.</p> <p>This is a non fatal error when running under PBUILD. PBUILD will continue to run as normal but the source file used for any commands to follow will be the original, unmodified source.</p>
<p>?SRCH FAIL</p> <p>The string searched for with the G command was not found.</p>
<p>?EOF</p> <p>If the error occurs in response to the H command, it indicates that the string was not found, and the source input file has been read to the end of file. If it occurs while running from a command file, it indicates that an end of file occurred in the command file before an EX command was encountered.</p>
<p>?BAD ARG FOR <command></p> <p>The argument provided with command is invalid for that command. For example, the L command only permits 0 and 1 as valid arguments.</p>
<p>?BAD COM: <command></p> <p>Command is not a valid command string.</p>
<p>?CAN'T BE THE SAME <log>=<command></p> <p>Unless the log file and the command file are both the keyboard, they must be different files. If they are not, this message is printed:</p> <pre> FATAL ERROR NO. <err> ON LINE <er1> LEAVING PROGRAM </pre> <p>The same as in PBUILD. However, err=10 is possible in normal operation. That error number indicates a protection violation, which can occur since CPATCH does not have to be run in a privileged account.</p>

(continued on next page)

Table 10-2: CPATCH Error Messages (Cont.)

Message and Meaning
<p>?FATAL ERROR NO.<err>ON LINE <er1> IN AUTOED</p> <p>The same as above, however, the error occurred in the program AUTOED which is chained to by CPATCH. This error will not terminate CPATCH, however. It has the same effect as a ?MISMATCH error. But it may still indicate the need for a SPR.</p> <p>PLEASE RUN CPATCH</p> <p>An attempt was made to run AUTOED as a stand alone program.</p> <p><string> NOT PERMITTED</p> <p>?BAD FILE NAME: <string></p> <p>?,<string> ERR=<num> FILE NOT FOUND</p> <p>?CAN'T COPY <file1> TO <file2></p> <p>These all have the same meaning as in PBUILD: refer to Table 10-1.</p>

Chapter 11

Device Testing Package (DEVTST)

The Device Testing Package (DEVTST) exercises hardware on the RSTS/E V7.0 system during normal time-sharing operations. DEVTST is used to stimulate hardware errors on the system.

11.1 Introduction to DEVTST

DEVTST is a package of programs that provides the customer, System Manager or DIGITAL Field Service personnel with tests that verify the reliability of RSTS/E supported hardware. Execution of any DEVTST program is limited to users with privileged accounts. Operational data, conventions, complete device testing information and error messages are explained in this chapter. After execution of individual DEVTST exercisers, users should check the System Error Log for additional errors; for more information on the System Error Log refer to Section 6.1 of this manual.

The DEVTST exercisers are tests that can be initiated whenever hardware trouble is suspected. These device exercisers generate excess activity in an attempt to make the hardware fail during operation.

The DEVTST package should be used in conjunction with the RSTS/E Error package as described in Chapter 6 of this manual. The Error Logging facility provides the means for determining detailed information on hardware errors. In addition, the DEVTST programs return error messages which may be useful in determining the failure, however, these messages are not as specific as the information provided by the System Error Log.

The following conventions are used throughout the DEVTST package dialogue:

- < > Indicates default answer
- () Shows all possible replies
- [] Gives an optional reply; for example ASC [II], means enter either ASC, ASCII or ASCII.
- <ALT MODE> or <ESC> Either key moves the program back one question. If you type this key as a response to the first question, DEVTST will stop execution and return to the RSTS/E command level.
- CTRL/Z** Typing CTRL/Z causes the program to stop execution and return to RSTS/E command level. CTRL/Z is detected only when the system checks for new input.
- CTRL/C** Typing CTRL/C causes the program to stop execution immediately and return to a command level.
- RET** Use the RETURN key to terminate the input line.
- Detach <NO> All of the device exercisers have the ability to detach; this allows you to run an exerciser while keeping your terminal free for other operations.
- 1 Negative one, when used as a value for number of iterations, pages or lines, tells the program to run indefinitely, or until you enter CTRL/C.

11.2 DEVTST Operating Procedures

To run a DEVTST program contained within the system library account [1,2], type:

```
RUN $program name
```

for example:

```
RUN $KBEXER
```

To run a DEVTST program from another privileged account, type:

```
RUN [account number]program name
```

for example:

```
RUN [1,215]KBEXER
```

Where program name is one of the tests in Table 11-1. Table 11-1 lists the program names and the hardware they test:

Table 11-1: DEVTST Programs

Program Name	Hardware Tested
CPEXER CPUTST DSKEXR DSKSEK DTEXER DXEXER KBEXER LPEXER MTEXER PPEXER PREXER	CPU CPU FILE STRUCTURED DISK NON-FILE STRUCTURED DISK DECTAPE DISKETTE TERMINAL LINE PRINTER MAGNETIC TAPE PAPER TAPE PUNCH PAPER TAPE READER

A complete list of RSTS/E supported hardware, including designators and the associated devices, appears in Table 11-2.

Table 11-2: RSTS/E Device Designator

Device Designator	Device
DF:, DS:, DK:, DL:, DM:, DP:, DR:, DB:, or SY:	RSTS/E Public disk structure as a whole.
SY0:	System disk (the unit which was bootstrapped).
DF0:	RF11 disk (all platters).
DS0: to DS7:	RS03/RS04 fixed head disk units 0 through 7.
DK0: to DK7:	RK05 disk cartridge units 0 through 7.
DL0: to DL3:	RL01/RL02 disk cartridge units 0 through 3.
DM0: to DM7:	RK06/RK07 disk cartridge units 0 through 7.
DP0: to DP7:	RP02/RP03 disk pack units 0 through 7.
DR0: to DR7:	RM02/RM03 disk units 0 through 7.
DB0: to DB7:	RP04/RP05/RP06 disk pack units 0 through 7.
PR:	High speed paper tape reader.
PP:	High speed paper tape punch.
CR:	CR11 punched or CM11 mark sense card reader.
CD:	CD11 punched card reader.
MT0: to MT7:	TE10/TU10/TS03 magnetic tape units 0 through 7.
MS0: to MS7:	TS04 magnetic tape units 0 through 7.
MM0: to MM7:	TE16/TU16/TU45/TU77 magnetic tape units 0 through 7.
LP0: to LP7:	Line printer units 0 through 7.
DT0: to DT7:	TU56 DECTape units 0 through 7.
DD0: to DD7:	TU58 DECTape II units 0 through 7.

11.3 DEVTST Dialogue Errors

The following error messages may occur in DEVTST programs:

Message and Meaning

%Warnings: (Device Name) is a logical device: (Device Name) will be used.

DEVTST determines that the device name given is a logical name for another device. DEVTST performs a logical device translation and the translated device name is used.

?Illegal device: (Device Name) specified.

The device name given by the user is either incorrect or is a logical device name that cannot be translated (for example, incorrect characters or the wrong type of device). Retry the operation with the correct device name.

?Open failure on (Device Name):

(Followed by an error message from Appendix B).

This error message can mean one of five things:

1. The device specified does not exist.
2. No volume was mounted on that device.
3. Another user has ASSIGNED the device.
4. The volume on the device has not been formatted.
5. The device is not on-line or the device is not write-enabled.

To resolve: retry the operation using the correct option as determined by the error message received.

?Illegal number of iterations selected.

Retry the operation using the correct number of iterations.

11.4 CPEXER

The central processor exerciser is designed to put a heavy load on the Central Processing Unit (CPU). CPEXER runs compute bound for short bursts and then sleeps for five seconds so that average processor loading does not appreciably degrade the performance of the system. CPEXER was specifically designed to test the Floating Point Processor FPP option, but also serves to verify general CPU integrity and the PDP11/40 Extended Instruction Set EIS operation. There is also a test of the PEEK function for kernal addresses 0 through 22000.

The tests performed by CPEXER are described in the next section. In most of the tests, results are compared to known correct values. Two heavily compute-bound tests are included to verify consistent results of duplicate calculations. Finally, tests are included that verify FPP divide by zero trap, integer divide by zero trap, integer conversion error and EIS operation.

11.4.1 CPEXER Tests

There are thirteen CPEXER tests included in the CPEXER program:

SIN(X)	Uses SIN(X) extended function.
SIN(X)	Uses polynomial approximation to SINE function.
LOG(X)	Uses LOG(X) extended function.
EXP(X)	Uses EXP(X) extended function.
SQR(X)	Uses SQR(X) extended function.
SQR(X)	Uses Newton-Raphson method to determine the square root.
LOG(EXP(X))	Compute-bound test.
ATN(TAN(X))	Compute-bound test.
A=1.0/0.0	Verify FPP divide by zero trap.
A%=1%/0%	Verify integer divide by zero trap.
A%=60000	Verify integer conversion error.
INT(40.6621*100+0.5)/100= 40.66	Verify EIS operation.
PEEK(X)	Verify operation of PEEK function.

11.4.2 CPEXER Dialogue

The following is an example of the CPEXER dialogue:

```
RUN [1,215] CPEXER
CPEXER V7.0 RSTS V7.0 Time Sharing

How many minutes to run <15>? 1

Detach <No>? N

11.9 seconds of CPU time used in 1 pass.

CPEXER finished at 03:36 PM

Ready
```

NOTE

The CPEXER program will build correctly only if the version of BASIC-PLUS or BASIC-PLUS-2 includes TRIG and LOG. These mathematical functions must be included during System Generation of BASIC-PLUS or the installation of BASIC-PLUS-2.

11.4.3 CPEXER Errors

This program reports four types of errors:

1. Instructions failed to execute.
2. Errors failed to trap.
3. Computation errors.
4. Computation consistency checks failed.

Check the System Error Log for reported errors. These errors may indicate hardware problems.

When type 1 errors (instruction errors) occur, CPEXER prints a two line message: the first line gives the test that was being performed and the exact error message and the second line indicates that the test is continuing. For example:

```
?CPEXER PEEK failed at memory location m.  
-(Error Message)  
  
Continuing.....
```

Where:

(Error Message) is the RSTS/E error message from Appendix B.

m is the memory location at which the PEEK function failed.

Examples of type 2 errors (trap failure) are:

```
?CPEXER Floating point divide by 0 didn't trap  
  
Continuing.....  
  
?CPEXER Integer divide by 0 didn't trap  
  
Continuing.....  
  
?CPEXER Forced integer conversion didn't trap  
  
Continuing.....  
  
?CPEXER (INT(40,6621*100=0.5))/100< >40.66  
  
Continuing.....
```

Type 3 errors (computation errors) give the type of test (SIN,PLYNOM, LOG, EXP, SQR or NEWTON SQR), error in the computed value, and the location of the possible error. For example:

```
?CPEXER - SIN test failed for the following cases:  
?CPEXER - Computed value of SIN( 40 ) was .643 , should be .64  
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error  
Continuing ...  
  
?CPEXER - PLYNOM SIN test failed for the following cases:  
?CPEXER - Computed value of PLYNOM SIN ( 40 ) was .643 , should be .64  
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error  
Continuing...  
  
?CPEXER - LOG test failed for the following cases:  
?CPEXER - Computed value of LOG ( 140 ) was 4.942 , should be 4.939  
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error  
Continuing...  
  
?CPEXER - EXP test failed for the following cases:  
?CPEXER - Computed value of EXP ( 4 ) was 54.598 , should be 54.599  
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error  
Continuing ...
```

```

?CPEXER - SQR test failed for the following cases:
?CPEXER - Computed value of SQR ( 500 ) was 22.361 , should be 22.362
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error
Continuing...

?CPEXER - NEWTON SQR test failed for the following cases:
?CPEXER - Computed value of NEWTON SQR ( 500 ) was 22.361, should be 22.362
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error
Continuing...

```

Type 4 errors (consistency failure) in the CPEXER are illustrated by the examples shown below:

```

?CPEXER - EXP/LOG Consistency check failed for the following cases:
?CPEXER - EXP(LOG(...( 1 ))))))) <> EXP(LOG(...( 1 )))))))
?CPEXER - EXP(LOG(...( 4 ))))))) <> EXP(LOG(...( 4 )))))))
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error
Continuing...

?CPEXER - ATN/TAN Consistency check failed for the following cases :
?CPEXER - ATN(TAN(...(-1.5 ))))))) <> ATN(TAN(...(-1.5 )))))))
?CPEXER - CPU,EIS,FIS,FPP,MUL,DIV or SHIFT Error
Continuing...

```

11.5 CPUTST

The CPUTST dialogue first asks how many minutes you want the test to run. The program then checks for expiration of this request time at eight strategic points during each run sequence. CPUTST then clears and/or sets three 2-dimensional arrays to specified values. Matrix A is then transposed and the results placed in Matrix B. The transposition is checked by the statement $A(J,I) = B(I,J)$ while I and J are being varied. Any error is then reported; if none are encountered, the program returns to the matrix clear instruction. The program repeats this process until the run time has expired, at which time it prints the CPU time in seconds of expended job time.

11.5.1 CPUTST Dialogue

The following is an example of CPUTST dialogue:

```

RUN [1,215]CPUTST
CPUTST V7.0 RSTS V7.0 Time Sharing

How many minutes to run <15>? 1

Detach <No>? NO

6.9 seconds of CPU time used.

CPUTST Finished at 10:59 AM

Ready

```

11.5.2 CPUTST Errors

This device exerciser reports errors as shown in the following examples:

```
?CPUTST-Error in matrix transposition routine.  
?(Error message) at line mm in CPUTST V7.0.
```

Where:

(Error Message) is the RSTS/E message in Appendix C of the *RSTS/E System User's Guide* for the error that occurred.

mm is the BASIC-PLUS line number in CPUTST where the error occurred.

Check the System Error Log for reported errors. These errors may indicate hardware problems.

If your hardware does not include a floating point processor, submit a Software Performance Report (SPR) to DIGITAL whenever errors appear.

11.6 DSKEXR

This disk exerciser tests the normal operation of all file-structured disks. It is possible to run several copies of DSKEXR in order to test several drives simultaneously or to put a heavier load on any single drive. The disk must be physically and logically mounted. DSKEXR begins by asking questions to determine the disk drive to be tested and the number of test iterations to be performed. After this dialogue, the exerciser opens and extends a temporary file to a predetermined size which is dependent upon disk size and the number of free blocks. A pattern buffer is then loaded with one of four patterns (all 1's and 0's or a pattern of alternating 1's and 0's) and the file is written. Each block is then read and compared for each of the patterns. Upon completion of all iterations for each drive specified, a status report is printed.

NOTE

This write-operation does not use current disk files to read and write. The temporary file will always be deleted after use, regardless of how the DSKEXR program is terminated. Be sure to turn off data caching before you run this test.

11.6.1 DSKEXR Dialogue

The following is an example of DSKEXR dialogue:

```
RUN[1,215] DSKEXR  
DSKEXR V7.0 RSTS V7.0 Time Sharing  
Disk drive to test? DB1:  
  
Number of test iterations <8>? 1  
  
Detach <No>? NO  
  
No errors detected on _DB1:  
  
Ready
```

11.6.2 DSKEXR Errors

Errors detected by DSKEXR are displayed as shown below:

? (Error message) error at line mm in DSKEXR V7.0.

?nn errors detected on DBn.

Where:

(Error message) is the RSTS/E message in Appendix C for the error that occurred.

mm is the line number where the error occurred in the DSKEXR program.

nn is the number of errors detected by DSKEXR.

When errors are detected:

1. Insure that the volume is physically mounted and write-enabled.
2. Insure that the volume is logically mounted. This may be verified by the use of the \$SYSTAT program (see the *RSTS/E System User's Guide*). If the disk is not mounted, use the MOUNT command of the \$UTILTY program (Chapter 7 of this manual).

CAUTION

- a. Packs mounted as unlocked public disks will be available to other system users for creation and storage of their files. Subsequent removal of a public disk may result in the elimination of these user files from the system. For this reason, DIGITAL recommends the use of a private pack if a non-system disk is to be tested.
 - b. Never mount a potentially bad disk on another drive until you are sure that the drive and pack are free of physical damage.
3. Persistent errors may be due to bad blocks on the pack. Any potentially bad blocks will be flagged in the System Error Log following unsuccessful access. To prevent further access to the option of INIT. (Refer to the *RSTS/E System Generation Manual*).

11.7 DSKSEK

This disk exerciser tests the normal operation of non-file structured disks by randomly reading blocks from the disk. It is possible to run several copies of DSKSEK in order to test several drives simultaneously or to put a heavier load on any single drive. If the disk being tested is a mounted RSTS/E file-structured disk, DSKSEK will not read the known bad blocks. Should an error occur on a read operation, it will be reported by DSKSEK and logged into the System Error Log.

NOTE

This is a read only operation. Turn off data caching before you run this test.

11.7.1 DSKSEK Dialogue

The following is an example of DSKSEK dialogue:

```
RUN [1,215] DSKSEK
DSKSEK V7.0 RSTS V7.0 Time Sharing

Disk drive to test? DB0:

Number of test iterations <30>? 1

Detach <No>?

No bad block detections in 1000 sets from _DB0:

Ready
```

11.7.2 DSKSEK Errors

Errors detected by DSKSEK are displayed as follows:

?(Error message) error at line mm in DSKSEK V7.0.

Where:

(Error Message) is the RSTS/E message in Appendix B for the error that occurred.

mm is the BASIC-PLUS line number where the error occurred in the DSKSEK program.

When errors are detected:

1. Check to see if the volume on the testing device is physically mounted.
2. If it is, mount another scratch volume on the drive and retry the operation.

CAUTION

Never mount a potentially bad disk on another drive until you are sure that the drive and pack are free of physical damage.

3. If the test continues to fail, check the System Error Log to see if bad blocks are recorded.
4. If bad blocks are recorded, add them to the bad block list on the disk using INIT's REFRESH/BADS suboption. (See the *RSTS/E System Generation Manual*.) This can only be done if the disk has been initialized (DSKINTed) as a RSTS/E disk.
5. If the problems continue get a complete listing of the System Error Log (using the ERRDIS program).

11.8 DTEXER

The DECTape exerciser tests the normal operation of the TCII DECTape control and TU56 DECTape drives. DTEXER begins by opening a file on the drive being tested and fills the file with floating point numbers. The program writes 420 blocks (of a possible 578 blocks) on the tape. Numbers written on the tape are read and checked, keeping a count of incorrect values. If at the end of the test the error count is not 0, DTEXER prints the error count.

CAUTION

Any information previously contained on the DECTape used for this test will be destroyed.

11.8.1 DTEXER Dialogue

The following is an example of DTEXER dialogue:

```
RUN [1,10] DTEXER
DTEXER V7.0 RSTS V7.0 Time Sharing

Warning: This exerciser will destroy data on the tested DECTape.

DEctape drive to test? DT1:

Number of iterations <2>? 1

Detach <No>? N
No errors detected on _DT1:

DTEXER Finished at 04:13 PM

Ready
```

11.8.2 DTEXER Errors

This device exerciser shows the number of errors found during operation as shown in this example:

```
?mm errors on DTn during iteration nn.
```

Where:

mm is the number of errors detected by the DTEXER program.

DTn is the name of device being tested.

nn is the number of iteration during which the error occurred.

When errors are detected:

1. Check to see if the DECTape on the testing device is physically mounted and write-enabled.
2. If it is, mount another scratch DECTape on the drive and retry the operation.
3. Insure that the tape path and heads are free of oxide.

11.9 DXEXER

This exerciser tests the normal operation of the diskette controller and the specified diskette drives. DXEXER starts by opening a file 420 blocks long consisting of floating point numbers. These are written on the drive being tested, then read back and verified. The program then displays the number of words written and read, and the number of errors detected.

CAUTION

Any information previously contained on the diskette used for this test will be destroyed.

11.9.1 DXEXER Dialogue

The following is an example of DXEXER dialogue:

```
RUN [1,215] DXEXER
DXEXER V7.0 RSTS V07.0 Time Sharing

Warning: This exerciser will destroy data on the tested floppy
disk.

Floppy drive to test? DX0:
Number of iterations <20>? 2
Detach <No>? NO
Floppy disk DXEXER I/O data

Device      Reads      Words      Writes      Words
 _DX0:      300        76.8 K    300        76.8 K

DXEXER Finished at 10:53 AM

Ready
```

11.9.2 DXEXER Errors

DXEXER shows the number of errors detected during operation as shown below:

?mm errors on DXn during iteration nn.

Where:

mm is the number of errors detected by the DXEXER program.

DXn is the name of device being tested.

nn is the number of the iteration during which the error occurred.

When errors are detected:

1. Check to see if the floppy disk on the testing device is physically mounted and the door closed.
2. If it is, mount another scratch floppy disk on the drive and retry the operation.

11.10 KBEXER

The keyboard exerciser (KBEXER) is used to test local and remote terminals connected to the RSTS/E system. The four terminal tests are as follows:

1. The Space Test verifies that the terminal carriage will return reliably from any position.
2. The ASCII Pattern Test verifies that the terminal will print the standard ASCII character set in all print positions.
3. The Worst Case Test forces the head on ASR33 terminals to rotate a half revolution in every print position.
4. The Repeat Test repeats everything typed, one line at a time.

NOTE

The tests can be terminated at any time by typing **CTRL/C**.
HELP can be obtained by typing **HELP** in answer to any dialogue question.

11.10.1 KBEXER Dialogue

The following is an example of KBEXER dialogue:

```
RUN [1,215] KBEXER  
KBEXER V7.0 RSTS V7.0 Time Sharing  
  
Keyboard to test <_KB68:>  
  
Test (SPACE], ASC[II], WOR[ST], REP[EAT], HEL[P])? ASC  
  
Number of test iterations <8>? 1  
  
Detach <No>? N  
  
*** ROTATING ASCII CHARACTERS TEST ***  
  
##$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"$%&  
"#$%&'()*+,-./0123456789:;<=? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"$%&  
! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"$%&  
! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"  
_ ! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"  
^ _ ! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"  
^ _ _ ! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]^_ !"  
Z[\]^_ ! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]  
YZ[\]^_ ! "$%&'()*+,-./0123456789:;<=>? @ABCDEFGHIJKLMN OPQRSTUVWXYZ[\]  
,  
,  
KBEXER Finished at 10:33 AM  
  
Ready
```

11.10.2 KBEXER Errors

Errors detected by KBEXER appear as errors in the keyboard test patterns. Visually check the patterns for errors.

11.11 LPEXER

This exerciser prints a rotating pattern of ASCII characters on a specified number of pages and is used to test any line printer.

11.11.1 LPEXER Dialogue

The following is an example of LPEXER dialogue:

```
RUN [1,215] LPEXER
LPEXER V7.0 RSTS V7.0 Time Sharing

Line Printer to test? LPO:

Number of Pages to output <4>? 1

Detach <No>? NO

LPEXER finished at 11:38 AM

Ready
```

11.11.2 LPEXER Errors

Problems found on the line printer appear as errors in the print-out of ASCII characters. Visually check the line printer listing and if an error message is printed by LPEXER, check to make sure that the line printer selected is on-line and not out of paper.

11.12 MTEXER

The magnetic tape exerciser checks normal operation of the TM11 or TS11 magnetic tape control or RH11/TM02/TM03 controller and the specified number of 7-track or 9-track TE10, TU10, TS03, TS04, TU16, TE16, TU45 or TU77 drives. MTEXER allows you to select the drive to be tested, the length of tape to be written and the number of iterations to be performed. For each iteration, MTEXER initializes the tape, opens a file and writes data until the specified length of tape has been used. MTEXER then rewinds the tape, opens the file for input, reads and verifies the data. If errors are detected, MTEXER prints a count of the number of bytes found to be incorrect; MTEXER then proceeds to the next iteration.

The data pattern used is a worst case NRZI pattern for 9-track drives; it is not worst case for 7-track recording. The pattern is loaded into a 512 byte buffer and, successively greater numbers of records are written to the magtape until the requested length of tape has been written. Furthermore, the pattern base varies with X so that the contents of the pattern buffer also varies on successive iterations.

CAUTION

This exerciser will overwrite data contained on the magnetic tape used for the test.

11.12.1 MTEXER Dialogue

The following is an example of MTEXER dialogue:

```
RUN [1,10]MTEXER
MTEXER V7.0 RSTS V7.0 Time Sharing

Warning: This exerciser will destroy data on the tested magtape.

Magtape drive to test? MT2:

Density (800 or 1600) <800>

Number of feet to test <100>?

Number of iterations <2>? 1

Detach <No>? N

Magtape MTEXER I/O Data
Device      Reads      Words      Writes      Words
_MT2:       387        99,072 K    387        99,072 K

No errors detected on _MT2:

Ready
```

11.12.2 MTEXER Errors

Magtape errors are reported in these formats:

```
?Magtape select error at line nn in MTEXER V7.0-07.
?MTEXER-Selection error occurred.
```

Where:

nn is the line number in MTEXER at which the error occurred.

When errors are detected:

1. Check to see if the volume on the testing device is physically mounted, write-enabled and assigned to you.
2. If it is, mount another scratch volume on the drive and retry the operation.
3. Insure that the tape path and heads are free of oxide.

11.13 PPEXER

This program tests the normal operation of the paper tape punch by punching random patterns on the paper tape for input into the PREXER.

11.13.1 PPEXER Dialogue

The following is an example of PPEXER dialogue:

```
RUN [1,215] PPEXER
PPEXER V7.0 RSTS V7.0 Time Sharing
```

```
This test will punch a binary count pattern tape.
The tape will be equivalent to MAINDEC-00-D2G4 entitled
Special Binary Count Pattern Tape.
```

```
Detach <No>?
```

```
The generated paper tape may be used to test the paper tape
reader.
```

```
PPEXER finished at 11:23 AM
```

```
Ready
```

11.13.2 PPEXER Errors

Problems in the paper tape punch will result in an incorrectly generated paper tape which is detected by PREXER when the tape is read. If PPEXER prints an error message, check to see that the paper tape punch is on-line and not out of paper tape.

11.14 PREXER

This program tests the normal operation of the paper tape reader by reading the paper tape punched by the PPEXER program.

11.14.1 PREXER Dialogue

The following is an example of PREXER dialogue:

```
RUN [1,215] PREXER
PREXER V7.0 RSTS V7.0 Time Sharing
```

```
Load MAINDEC-00-D2G4-PT or the binary count pattern
tape generated by the punch exerciser into the reader
with the leader under the read station.
```

```
Detach <No>? NO
No errors detected on paper tape.
```

```
PREXER finished at 11:29 AM
```

```
Ready
```

11.14.2 PREXER Errors

PREXER errors are reported in this format:

```
?nn errors detected on paper tape
```

Where:

nn is the number of errors detected by the PREXER program.

NOTE

If the paper tape was loaded backwards and the reader is operating correctly, the following error message is printed: ?15360 errors detected on paper tape.

Any other errors detected by the PREXER may indicate a hardware problem.

Appendix A

Hardware Bootstrap Procedures

Bootstrapping a device involves using the Central Processor Unit (CPU) console switches to access and initiate a hardware loader. The hardware loader contains machine instructions for reading a special record from the device. The record, called a bootstrap record, is transferred into memory and executes a specially designed software program. For the bootstrap operation to succeed, the device accessed must be on line and ready; the medium accessed must contain a proper bootstrap record; and the console terminal must be on line.

The PDP-11 console switches and their use are described in the Processor Handbook. The bootstrap procedure that you should use depends upon the type of hardware bootstrap device on the system. Table A-1 summarizes the addresses needed to bootstrap each device. The detailed procedures to bootstrap a device are presented according to the types of hardware bootstrap devices available.

Table A-1: Summary of Hardware Bootstrap Addresses

Device to Bootstrap	Bootstrap Type								
	BM873-YA	BM873-YB*	MR11-DB	BM792-YB	M9301-YA	M9301-YB	M9301-YC	M9301-YF	M9312
RF11 Disk	773000	773136	773100	777462					
RS03/RS04 Disk					DS	DS	100	DS	DS
RK05 disk cartridge Unit 0	773010	773030	773110	777406	DK	DK	30	DK	DK
RP02 or RP03 disk pack unit 0	773100	773350	773154	776716	DP	DP	40	DP	DP
RP04, RP05, or RP06 disk pack unit 0		773320				DB	70	DB	DB
RK05 disk (unit n)		773032			DKn	DKn	30+n	DKn	DKn
RL01/RL02 disk unit n									DLn
RK06 disk (unit n)								DMn	DMn
RK07 disk (unit n)									DMn
RP02 or RP03 disk pack (unit n)		773352			DPn	DPn	40+n	DPn	DPn
RP04, RP05, RP06, RM02, RM03, or RP06 disk (unit n)		773322				DBn	70+n	DBn	DBn
TU10/TE10 and TS03 magnetic tape	773050	773110	773136	**	MT	MT	10	MT	MTn
TS04 magnetic tape									
TU16/TE16/TU45 /TU77 magnetic tape		773150				MM	60	MM	MMn
TU56 DECtape	773030	773070	773120	777344	DT	DT	20	DT	DTn
*To bootstrap a nonzero disk unit, set the address in the Switch Register, press the LOAD ADRS switch, set the unit number in the Switch Register, and press the START switch. **To bootstrap a TU10/TE10 or TS03 magnetic tape, use the loading routine described in Section A.5.									

A.1 BM873-YA Procedure

If the BM873-YA Restart/Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

773000 for RF11 disk
 773010 for RK05 Disk Cartridge
 773100 for RP02 or RP03 disk pack
 773050 for TU10/TE10/TS03 Magnetic Tape

Press the CPU LOAD ADRS switch.

Press the CPU START switch.

A.2 BM873-YB Procedure

If the BM873-YB Restart/Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

- 773136 for RF11 disk
- 773030 for RK05 disk cartridge
- 773350 for RP02 or RP03 disk pack
- 773320 for RP04, RP05, RP06, RM02, or RM03 disk pack
- 773032 for RK05 disk unit specified in the Switch Register
- 773322 for RP04, RP05, RP06, RM02 or RM03 disk unit specified in the Switch Register
- 773352 for RP02 or RP03 disk unit specified in the Switch Register
- 773110 for TU10/TE10/TS03 magnetic tape
- 773150 for TU16/TE16/TU45/TU77 magnetic tape

Press the CPU LOAD ADRS switch.

If necessary, set the CPU Switch Register to the unit number of the disk drive being bootstrapped.

Press the CPU START switch.

A.3 H324 Pushbutton Panel Procedure

If the system includes the H324 Pushbutton Panel option for the BM873 Restart/Loader, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

While holding down the UNLOCK switch, press the appropriate switch to bootstrap the desired device. The DIGITAL field service engineer who installed the computer should have informed you of the device that each switch bootstraps.

Press the CPU START switch.

A.4 MR11-DB Procedure

If the MR11-DB Bulk Storage Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

773100 for RF11 disk
773110 for RK05 disk cartridge
773154 for RP03 disk pack
773136 for TU10/TE10/TS03 magnetic tape

Press the CPU LOAD ADRS switch.

Press the CPU START switch.

A.5 BM792-YB Procedure

If the BM792-YB Hardware Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to 773100.

Press the CPU LOAD ADRS switch.

Set the CPU Switch Register to one of the following values.

777462 for RF11 disk
777406 for RK05 disk cartridge
776716 for RP03 disk pack

Press the CPU START switch.

To bootstrap a TM11/TU10 TE10/TS03 magnetic tape when the system does not have a bootstrap that handles it, you must manually enter a load routine into memory using the CPU console Switch Register and the DEP switch.

To load the routine, perform the following steps.

Move the CPU Console ENABLE/HALT Switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to 010000.

Press the CPU LOAD ADRS switch.

Load the following contents into memory using the Switch Register and DEP switch.

Address	Contents
010000	012700
010002	172524
010004	005310
010006	012740
010010	060011
010012	105710
010014	100376
010016	005710
010020	100767
010022	012710
010024	060003
010026	105710
010030	100376
010032	005710
010034	100777
010036	005007

Set the Console Switch Register to 010000.

Press the CPU LOAD ADRS switch.

Press the CPU START switch.

If the system reads the tape but halts at address 010034, the magnetic tape generated a parity error. Try another drive and make it unit 0. If the system appears to take no action and halts, verify the accuracy of the routine by using the CPU Console EXAM switch. Use the Switch Register and the DEP switch to correct any erroneous contents. Rewind the tape to its load point before executing the routine again. If no recovery is successful, you should have a DIGITAL field service representative check the device. If the hardware is working properly, you should use a new tape reel.

A.6 M9301-YA and M9301-YB Procedure

If the M9301-YA or M9301-YB Bootstrap Terminator is on the system, perform the following steps.

While holding down the CTRL switch, press the BOOT switch on the front of the computer.

The bootstrap prints the contents (in octal) of registers R0, R4, and the Stack Pointer, and the Program Counter at the console terminal. It then prints a prompting \$ character.

In response to the \$ character, type the 2-character device code and the unit number of the device to be bootstrapped. Terminate your response by pressing the RETURN key. Table A-2 describes the device codes.

Table A-2: Device Codes for M9301-YA and M9301-YB Bootstraps

Device	Code
TU10/TE10/TS03 Magnetic Tape	MT
*TU16/TE16/TU45/TU77 Magnetic Tape	MM
RK05 Disk Cartridge	DK
RP02/RP03 Disk Pack	DP
*RP04/RP05/RP06/RM02/RM03 Disk Pack	DB
*RS03/RS04 Fixed Head Disk	DS
*M9301-YB Bootstrap Terminator only.	

A.7 M9301-YC Procedure

If the M9301-YC bootstrap is on the system, perform the following steps.

Move the CPU ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the start address of 17765000 in the console switches.

Press the CPU LOAD ADDR switch.

Set the device unit number in switches 0 through 2.

Set the device code for the device to be bootstrapped in switches 3 through 6. Table A-3 lists the device codes.

Table A-3: Device Codes for M9301-YC Bootstrap

Device	Code
TU10/TE10/TS03 Magnetic Tape	1
RK05 Disk Cartridge	3
RP02/RP03 Disk Pack	4
TU16/TE16/TU45/TU77 Magnetic Tape	6
RP04/RP05/RP06/RM02/RM03 Disk Pack	7
RS03/RS04 Fixed Head Disk	10

Ensure that switches 7 through 21 are off (down).

Press the CPU START switch.

NOTE

Before the M9301-YC bootstrap actually bootstraps the system, it performs CPU tests, instruction and addressing tests, and memory and cache tests. If a hardware failure is detected, the diagnostic program halts. The lights contain the ROM address of the halt. If this occurs, call the DIGITAL field service engineer.

It may, however, be possible to continue with the bootstrap operation if the lights contain the address 1773764, which indicates a cache failure. To continue in this case, press CONT. This is the ONLY case in which it is possible to continue bootstrapping after the diagnostic detects an error.

A.8 M9301-YF Procedure

If the M9301-YF Bootstrap Terminator is on the system, perform the following steps.

While holding down the CTRL switch, press the BOOT switch on the front of the computer.

The bootstrap prints the contents (in octal) of registers R0, R4, the Stack Pointer, and R5 at the console terminal. It then prints a prompting \$ character.

In response to the \$ character, type the 2-character device code and the unit number of the device to be bootstrapped. Terminate your response by pressing the RETURN key. Table A-4 lists the device codes.

Table A-4: Device Codes for M9301-YF Bootstrap

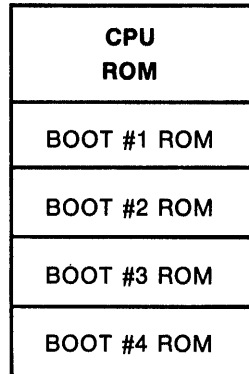
Device	Code
TU10/TE10/TS03 Magnetic Tape	MT
TU16/TE16/TU45/TU77 Magnetic Tape	MM
RK05 Disk Cartridge	DK
RK06/RK07 Disk Cartridge	DM
RP02/RP03 Disk Pack	DP
RP04/RP05/RP06/RM02/RM03 Disk Pack	DB
RS03/RS04 Fixed Head Disk	DS

A.9 M9312 Procedure

The M9312 Bootstrap Module has five sockets that accommodate one CPU ROM and up to four peripheral boot ROMs. Each CPU and boot device has a unique ROM developed for it. The CPU ROM plugs into a specific socket location within the module. The peripheral ROMs, however, may be placed in

any of the four peripheral socket locations as long as the sockets are filled in sequential order with no vacancies between loaded ROM sockets. Since a particular peripheral ROM has a different boot start address in each of the four sockets, it is important to know the locations of the ROMs in the bootstrap module. The ROM configuration is a diagram of where specific boot ROMs are located. This information must be known before the bootstrap procedure can begin. The following sections explain how to determine the ROM configuration if it is not known and how to bootstrap a device when the ROM configuration is known. Figure A-1 is a diagram of the ROM arrangement for the M9312 Bootstrap Module.

Figure A-1: M9312 ROM Arrangement



If the M9312 Bootstrap/Terminator is on the system, perform one of the following procedures:

- If you have the M9312 console emulator on the system, follow the procedures in Section A.9.1 (PDP-11/60's and PDP-11/70's usually do not have console emulators). If that procedure does not work, perform the next step to verify that you have a console emulator.
- If you have a CPU equipped with switch registers and *do not know* what devices the M9312 will boot, perform the procedures in Section A.9.2.
- If you have a CPU equipped with switch registers, *know* what device ROMs are on the M9312 Bootstrap Module, but do not have a console emulator, follow the procedures in Section A.9.3.
- If you do not have the M9312 Bootstrap/Terminator Module on your system and need to boot the RL01, RL02, or the RK07 disk device, perform the manual load routine described in Section A.10.

A.9.1 Using the M9312 Console Emulator

The M9312 console emulator can be initiated in two ways depending on the type of CPU (push button or switch register) on the system.

1. *Push button* boot using the M9312 console emulator.

While holding down the CTRL button, press the BOOT button on the CPU.

The bootstrap prints the contents (in octal) of registers R0, R4, the stack pointer, and R5 at the console terminal. It then prints a prompting @ character.

In response to the @ character, type the 2-character device code and the unit number of the device to be bootstrapped. Terminate your response by pressing the return key. If the unit number is not entered, it is assumed to be zero. Table A-7 contains the device codes.

Successful completion of these push button bootstrap procedures produce header information on the console terminal, similar to the following example:

```
Enabling only console, disks, and tapes
RSTS V7.0 (MTO)
Option:
```

Begin the system generation process in Chapter 3—TAILORING THE SYSGEN SYSTEM.

2. *Switch register boot* using the M9312 console emulator.

- Move the CPU console Enable/Halt switch to its Halt position then back to its Enable position.
- Set the CPU switch register to 765020.
- Press the start switch. The bootstrap prints the contents (in octal) of registers R0, R4, the stack pointer, and R5 at the console terminal. It then prints a prompting @ character.

In response to the @ character type the 2-character device code and the unit number of the device to be bootstrapped. Terminate your response by pressing the RETURN key.* If the unit number is not entered, it is assumed to be zero. Table A-7 contains the device codes.

Successful completion of the switch register bootstrap procedure produces header information on the console terminal, similar to the example given for the push button bootstrap. After the message appears, begin the system generation process in Chapter 3—Tailoring the Sysgen System.

A.9.2 Determining the M9312 Bootstrap ROM Configuration

If you *do not know the M9312 ROM configuration*, it can be determined by using Tables A-5 and A-7 and performing the instructions that follow Table A-5.

*The @ sign returns at this point if the correct boot ROM was not installed, or if a nonexistent device code is entered. If the contents of the registers R0, R4, Stack Pointer and R5 appear before the @ sign, this indicates that at least one boot ROM socket is empty and can accommodate another boot ROM.

Table A-5: ROM Locations

ROM Address	ROM Location
765774	Diagnostic ROM
773000	ROM 1
773200	ROM 2
773400	ROM 3
773600	ROM 4

Table A-6: M9312 Bootstrap ROM Configuration

ROM Location	Bootable Device	Device Mnemonic
Diagnostic ROM		
ROM 1		
ROM 2		
ROM 3		
ROM 4		

Repeat the following steps for each of the ROM addresses shown in Table A-5. After you have completed this process, you will know the M9312 bootstrap ROM configuration of your system. With this information, you can boot any device having a boot ROM in the M9312 Bootstrap Module.

- Select a six character ROM address from Table A-5.
- Set the CPU switches to the selected number.
- Press the LOAD ADRS switch to set the ROM address.
- Press the EXAM switch. The contents of the ROM address just loaded will appear in the CPU data register.
- Match the octal number represented in the CPU data register with the octal data number shown in Table A-7.
- Locate in Table A-7 the device mnemonic and the device name associated with the matching number. The ROM you have just investigated can boot the device(s) you located in Table A-7. Write the retrieved device information from Table A-7 with the appropriate ROM location in Table A-6. For example, if you selected a six character ROM address for ROM 1 from Table A-5, place the device mnemonic and name you located in Table A-7 with ROM 1 in Table A-6. Documenting the ROM configuration in Table A-6 will obviate performing the above procedure again.

Table A-7: Device Name and Mnemonic

Device to Be Booted	Mnemonic	Unit	Octal Data
RL01	DL	0-3	042114
RK06/RK07	DM	0-7	042115
RX01	DX	0-1	042130
RX02	DY	0-1	042131
RS03/RS04	DS	0-7	042123
RP02/RP03	DP	0-7	042120
RP04/RP05/RP06	DB	0-7	
RM02/RM03	DB*		
RK03/RK05	DK	0-7	042113
TU55/TU56	DT	0-7	
TU16/TE16/ TU45/TU77	MM	0-7	046515
TU10/TE10/TS03	MT	0-7	046524
Console Emul. CPU ROM	A0	N/A	040460
Dagnostic ROM PDP-11/60/70	B0	N/A	041060
			177776** xxx777***
*Type DB rather than DR if you are booting the RM02 or RM03 disk. **This is a continuation ROM of a multiple ROM boot. ***Bad ROM or NO ROM present.			

After you have determined the ROM configuration, perform the procedures in Section A.9.1 if you find you have a console emulator; otherwise, perform the procedures in Section A.9.3. The following section explains how to bootstrap with the M9312 module using a switch register controlled CPU.

A.9.3 Bootstrapping with the M9312

The following procedure describes how a device is bootstrapped when the M9312 Bootstrap ROM configuration is known. The data you need to bootstrap a device appears in Table A-8.

- Locate in Table A-8 the name of the device to be booted.
- Determine from the ROM configuration (Table A-6) the location (ROM 1/2/3/4) of the device boot ROM.
- Select the three character Switch Register Code (SWR) from Table A-8 for the particular ROM location.

- Move the CPU console Enable/Halt switch to its Halt position then back to its Enable position.
- Set the CPU switch register to 765744 (or 17765744 for PDP-11/70).
- Press the CPU LOAD ADRS switch.
- Set the Switch Register Code (SWR) selected from Table A-8 in CPU switches 0-8 as shown in Table A-9.
- Set the unit number of the boot device in CPU switches 9-11 (see Table A-9).
- Press the CPU start switch.

Successful completion of this process boots the device. A message, similar to the following example, appears on the console terminal. At this point, you may begin the system generation process in Chapter 3—Tailoring the Sysgen System.

Enabling only console, disks, and tapes.

RSTS V7.0 (MTO)

Option:

Table A-8: Switch Register Codes (SWR)

Device to Be Booted	Switch Register Code (SWR)			
	ROM 1	ROM 2	ROM 3	ROM 4
RL01/RL02	012	212	412	612
RK06/RK07	012	212	412	612
RX01	012	212	412	612
RX02	012	212	412	612
RS03/04	012	212	412	612
RP02/RP03	012	212	412	612
RP04/RP05/RP06	056	256	456	656
RM02/RM03	056	256	456	656
RK03/05/05J	012	212	412	612
TU56	042	242	442	642
TU16/TE10/TS06	012	212	412	612
TU10/TE10/TS03	012	212	412	612

Table A-9: Console Switch Register Settings

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Octal Unit Number							Switch Register Code (SWR) from Table A-8								

A.10 Load Program Bootstrap for the RL01/RL02 and RK07

To bootstrap an RL01, RL02 or RK07 device when the system does not have the M9312 Bootstrap Module, you must manually enter a bootstrap load routine using the CPU console. The following procedure describes the manual load process using the PDP-11/34. A similar procedure is followed using the CPU LOAD ADRS, DEP, and EXAM switches for the PDP-11/70 computer.

- While holding down the CTRL button, press the HLT/SS button on the CPU.
- Set address 1000 in the console register by using the numbered console buttons.
- Press the LAD(load address) button.
- Press the CLR button to clear the register.
- Load the *contents*, that appears below, into memory using the numbered buttons and the DEP button. Press the CLR button after you have placed the contents in the CPU register and pressed the DEP button.

	RL01/RL02 Bootstrap Routine	RK07 Bootstrap Routine
Address	Contents	Contents
1000:	12700	12700
1002:	174400	177440
1004:	12760	12760
1006:	13	0*
1010:	4	10
1012:	12701	12701
1014:	4**	2003***
1016:	10110	10110
1020:	105710	105710
1022:	100376	100376
1024:	12760	12760
1026:	177601	177400
1030:	4	2
1032:	5721	62701
1034:	10110	16
1036:	105710	10110
1040:	100376	105710
1042:	12760	100376
1044:	177400	5710
1046:	6	100001
1050:	5060	0
1052:	4	5007
1054:	62701	
1056:	6	
1060:	10110	
1062:	105710	
1064:	100376	
1066:	5710	
1070:	100001	
1072:	0	
1074:	5007	

*Select drive number 0-7 for the RK07.

**Load 4 for unit 0, 404 for unit 1, 1004 for unit 2, and 1404 for unit 3.

***Load 3 for RK06.

After the contents are loaded, perform the following steps.

- Place address 1000 in the console register *again* by using the numbered buttons.
- Press the CPU LAD(load address) button to load the address.
- While holding the CTRL button down, press the START button on the CPU to boot the device.

You can verify the accuracy of the load routine by using the CPU EXAM button. Place address 1000 in the register and press the LAD button. Press the EXAM button to reveal the contents of address 1000. It should be 12700 for both the RL01, RL02, and RK07 routines. Continue to press the EXAM button to check the contents of the remaining addresses in the load routine. Use the LAD and the DEP buttons to correct any erroneous contents.

Appendix B

System Generation Error Messages

B.1 Initialization Code Error Messages

The RSTS/E system initialization code routines make many checks to ensure the consistency of system structures. Initialization routines compare existing structures with their definitions and with references in other parts of the system. The checks must be successful; if not, the routines detect a consistency error, which indicates that the system is corrupted.

The initialization routines may also encounter errors while attempting to execute initialization options. The error text for many of these errors is preceded by the message `FATAL RSTS/E SYSTEM INITIALIZATION ERROR`. The initialization code prints a descriptive error message and returns to the `OPTION:` prompt. If such an error occurs, follow the error recovery procedures in Table B-1. If these procedures are unsuccessful, consult your DIGITAL Software Support Specialist.

For many of the errors, no recovery procedures exist. If such an error occurs, you should submit a Software Performance Report (SPR) as noted in the Recovery Procedure column for that error message.

Table B-1: Initialization Code Error Messages

Message and Recovery Procedure
ACCOUNT [1,2] MISSING FROM OUTPUT DISK Reinitialize the disk as a system disk.
BAD [0,1] DIRECTORY Rebootstrap the system disk and retry the operation. If the retry fails, use DSKINT to reinitialize the disk.
BADB.SYS NOT FOUND - RESULTS UNPREDICTABLE Rebootstrap the system disk and retry the operation. If the retry fails, use DSKINT to reinitialize the disk.
BUFF.SYS NOT FOUND OR TOO SMALL - DECTAPE DISABLED Warning message only. Shut down the system and use REFRESH to create the file at the required size.
CANNOT REFRESH THIS DISK Reinitialize the disk.
CANNOT START WITH THIS SWAP MAX AND MEMORY TABLE Use DEFAULT to change the swap max and/or memory allocation table.
CLUSTER 1 (REQUIRED FOR MFD) CONTAINS A BAD BLOCK If the pack cluster size for the disk is greater than the device cluster size, use DSKINT to reinitialize the disk with a lower pack cluster size. If the error recurs, use a different disk.
CLUSTER ALLOCATED TO [P,Pn] filnam.ext IS NOT ON A PACK CLUSTER BOUNDARY CLEAN asks whether to delete the file, then confirms your response.
[P,Pn] CLUSTER MAP IN UFD DISAGREES WITH MFD CLEAN asks whether to zero the account and confirms your response.
DECTAPE INPUT ERROR Ensure that the DECTape drive is clean and properly aligned. If the error recurs, try another drive.
DEFAULT RTS HAS INVALID SIZE PARAMETERS Regenerate.
DEFAULT RTS NOT FOUND OR INVALID Ensure that a valid run-time system is on the system disk. Regenerate if necessary.
DEVICE xx: DOES NOT INTERRUPT - DEVICE DISABLED Use HARDWR to set the correct vector address for xx: or call the DIGITAL field service engineer.

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
DEVICE ERROR WHEN TRYING TO BOOTSTRAP DEVICE Ensure that the device is on-line and ready. If the error recurs, try a different device.
DEVICE HUNG OR WRITE-LOCKED Ensure that the device is on-line and ready. If the error recurs, try a different device.
DIRECTORY ENTRY FOR [P,Pn] filnam.ext CONTAINS PACK CLUSTER NUMBER WHICH IS TOO BIG CLEAN asks whether to delete the file and confirms your response.
DIRECTORY ERROR DETECTED IN RDB Reinitialize the disk.
DIRECTORY ERROR - FILE NOT FOUND Reinitialize the disk.
[0,1] DIRECTORY NOT FOUND Reinitialize the disk.
DISK BLOCK 0 (REQUIRED FOR BOOTSTRAPS) IS BAD Reinitialize the disk. If the error recurs, use a different disk.
DISK ERROR DURING DSKINT BUILD PHASE Reinitialize the disk. If the error recurs, use a different disk.
DISK HUNG OR DATA ERROR DURING OVERLAY Ensure that the device is on line and ready. If the error recurs, try a different device.
DISK STRUCTURE IS IRREVOCABLY CORRUPT Reinitialize the disk.
DOUBLY ALLOCATED BLOCK FOUND AT DEVICE CLUSTER nn. THE BLOCK IS ALLOCATED TO [P,Pn] filnam.ext AND [P,Pn] filnam.ext CLEAN lets you choose the file to delete, then confirms your response.
DRIVE NOT READY: Ensure that the disk drive is on-line and READY. Press the LINE FEED key to retry the operation. Type any other character to abort the operation.
DSKINT ERROR - ATTEMPT TO FORMAT UNFORMATTABLE DISK Submit an SPR.
DSKINT OR SYSGEN ERROR - DSKINT ALLOCATION FAILURE Submit an SPR.
EOF READING INIT.SYS Submit an SPR.

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>ERR FILE INVALID Copy the file ERR.ERR from the distribution medium to the system disk and use the DEFAULT option to install ERR.ERR as the system default.</p> <p>ERR FILE NOT FOUND Copy the file ERR.ERR from the distribution medium to the system disk and use the DEFAULT option to install ERR.ERR as the system default.</p> <p>ERROR DETECTED WHILE READING ERROR RECORD SECTOR #n Warning message only. Error occurred during reading of factory error records. Disregard the error message.</p> <p>ERROR - KB INTERFACES OUT OF SEQUENCE Regenerate. If the problem recurs, submit an SPR.</p> <p>ERROR - NO SETUP FOR DISK xx: Regenerate. If the problem recurs, submit an SPR.</p> <p>ERROR - UNKNOWN DISK xx: Regenerate. If the problem recurs, submit an SPR.</p> <p>EXCESSIVE BAD CLUSTERS If you need to use the current disk, use DSKINT to increase the pack cluster size. If the error recurs, use a different disk.</p> <p>FATAL DISK ERROR DURING CONTROL RESET Retry the DSKINT. If the error recurs, call the DIGITAL field service engineer to repair the disk drive.</p> <p>FATAL ERROR - NO SMALL BUFFERS LEFT Regenerate the system and configure more small buffers</p> <p>FATAL I/O DURING OVERLAY Ensure that the device is on line and ready. If the error recurs, try a different device.</p> <p>xxx FILE NOT FOUND WHEN REMOVING IT Warning message only. xxx is SIL, RTS, or ERR. The initialization code did not find a file when trying to make it deletable.</p> <p>[P,PN] filnam.ext HAS A BAD BLOCK CLEAN asks whether to delete the file and confirms your response.</p> <p>[P,PN] filnam.ext HAS FIRST DCN OUT OF RANGE CLEAN asks whether to zero the account and confirms your response.</p> <p>[P,PN] filnam.ext HAS INVALID ACCOUNTING ENTRY LINK CLEAN asks whether to delete the account and confirms your response.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p><code>filnam.ext</code> HAS ILLEGAL FORMAT AND CANNOT BE COPIED Submit an SPR if DIGITAL supplied the file on the distribution medium. Regenerate if the system generation process created the file.</p> <p>FORMATTING FAILURE: Press the LINE FEED key to retry. Type anything else to abort the DSKINT and return to the OPTION: prompt. Ensure that the disk drive is write enabled. This error may indicate a bad disk pack or a bad disk drive.</p> <p>[P,PN] HAS A BAD BLOCK CLEAN asks whether to delete the account and confirms your response.</p> <p>[P,PN] HAS HOLES IN CLUSTER MAP CLEAN asks whether to zero the account and confirms your response.</p> <p>[P,PN] HAS INCONSISTENT CLUSTER MAPS CLEAN asks whether to zero the account and confirms your response.</p> <p>[P,PN] HAS INVALID CLUSTER SIZE CLEAN asks whether to delete the account and confirms your response.</p> <p>ILLEGAL CONTROLLER FOR TU58 Submit an SPR.</p> <p>ILLEGAL INTERFACE INDEX <code>nn</code> Regenerate. If the error recurs, submit an SPR.</p> <p>INIT BUG - BAD DISK INDEX Submit an SPR.</p> <p>INIT BUG - CAN'T ALLOCATE FILE 2ND TIME Submit an SPR.</p> <p>INIT BUG - CAN'T CHECK PARITY Submit an SPR.</p> <p>INIT BUG - CAN'T FIND [0,1] AGAIN Submit an SPR.</p> <p>INIT BUG - CAN'T FIND FILE TO DELETE Submit an SPR.</p> <p>INIT BUG - CLEAN FAILED TO FIND ALLOCATION ERROR Submit an SPR.</p> <p>INIT BUG - DIDN'T GET SAME CLUSTER 2ND TIME Submit an SPR.</p> <p>INIT BUG - DIDN'T GET SAME FILE CLUSTER 2ND TIME Submit an SPR.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
INIT BUG - DON'T KNOW HOW TO START Submit an SPR.
INIT BUG - FAILED TO CREATE FILE Submit an SPR.
INIT BUG - FAILED TO FIND FILE AGAIN Submit an SPR.
INIT BUG - FAILED TO FIND FILE JUST CREATED Submit an SPR.
INIT BUG - FILE EXISTS WHEN TRYING TO CREATE Submit an SPR.
INIT BUG - xxx - file type FILE NOT FOUND 2ND TIME Submit an SPR.
INIT BUG - xxx - FILE NOT FOUND WHEN REMOVING IT Submit an SPR.
INIT BUG - FILE SIZE COMPUTED WRONG Ensure that the system device you are using contains a valid system. If so, submit an SPR.
INIT BUG - FLOAT TABLE ERROR Submit an SPR.
INIT BUG - INSTALL NOT SYNCHRONIZED AT xxxxxx Submit an SPR.
INIT BUG - NO SATT.SYS AT WIPEOUT TIME Submit an SPR.
INIT BUG - NO UFD FOUND IN CREATE Submit an SPR.
INIT BUG - RNB TO WRONG DISK Submit an SPR.
INIT BUG - SATT.SYS NON-EXISTENT AT TIME OF WOMP Submit an SPR.
INIT BUG - SET NEW TABLE TOO SMALL Submit an SPR.
INIT BUG - UNKNOWN BAE DEVICE Submit an SPR.
INIT BUG - START LINKED TOO LOW Submit an SPR.

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>INIT LINK BUG - DEFBUF NOT ALIGNED Submit an SPR.</p> <p>INIT LINK BUG - DSTBL TOO LOW Submit an SPR.</p> <p>INIT LINK BUG - FIBUF NOT ALIGNED Submit an SPR.</p> <p>INIT LINK BUG - INSBUF NOT ALIGNED Submit an SPR.</p> <p>INIT LINK BUG - SATBUF NOT ALIGNED Submit an SPR.</p> <p>INIT.SYS NOT FOUND - RESULTS ARE UNPREDICTABLE Rebootstrap the system disk and retry the operation. If the retry fails, reinitialize the disk.</p> <p>INPUT FILE MISSING FROM SYSTEM DEVICE Retry the procedure that generated the error. If the error recurs, submit an SPR.</p> <p>INSTALLED MONITOR DOES NOT SUPPORT THIS SYSTEM DISK Regenerate a monitor that supports the system disk or use the COPY option to transfer the monitor to a disk it supports.</p> <p>INSTALLED SIL INVALID The installed SIL is invalid as a monitor SIL. Use the INSTALL option to install a valid SIL.</p> <p>INVALID ACCOUNT NUMBER [P,PN] CLEAN deletes the account after requesting your confirming response.</p> <p>INVALID DATA IN ERROR RECORD SECTOR #n Warning message only. The error occurred during reading of factory error records. Disregard the error message.</p> <p>INVALID RETRIEVAL ENTRIES FOR FILE [P,PN] filnam.ext - FIXED BY CLEAN Warning message. CLEAN truncates the file.</p> <p>I/O TO UNOPENED FILE Submit an SPR.</p> <p>[P,PN] IS NOT A VALID ACCOUNT NUMBER CLEAN asks whether to delete the account and confirms your response.</p> <p>MFD NAME ENTRY CONTAINS A BAD LINK. CLEAN WILL DELETE ALL [1,1] FILES AND ALL ACCOUNTS BEYOND [P,PN] filnam.ext CLEAN confirms your response.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
MONITOR SIL CHANGED - REINSTALL IT Use the INSTALL option to install the monitor SIL.
MONITOR SIL NOT FOUND Use the INSTALL option again to install the SIL. If the error recurs, submit an SPR.
MT ERROR OR INVALID FORMAT MM ON TAPE Ensure that the magtape drive heads are clean and correctly aligned. If the error recurs, try using a different drive.
NO LIBRARY ACCOUNT ON THIS DISK Reinitialize the disk and create a library account.
NO ROOM FOR DIRECTORY ON OUTPUT DISK Delete files or reinitialize the disk to create sufficient contiguous space on the disk for the file. Or use a different disk.
NO ROOM FOR FILE ON OUTPUT DISK Delete files, reinitialize the disk to create sufficient contiguous space on the disk for the new file, use a different disk.
NO ROOM FOR 2 JOBS IN SWAP.SYS Use REFRESH to increase the size of SWAP.SYS so that it can accommodate two jobs of size SWAP MAX.
NOT ENOUGH ROOM IN LOW MEMORY FOR MONITOR Generate a smaller monitor.
NOT ENOUGH ROOM IN MEMORY FOR RUN TIME SYSTEM Generate a smaller monitor or Run-Time System.
ONE OR MORE SYSTEM FILES MISSING Reinitialize the disk.
OPTION ATTEMPTED DURING BOOTSTRAP PHASE Rebootstrap the system and retry the operation. If the retry fails, submit an SPR.
OPTION NOT AVAILABLE The option you requested is not available until you bootstrap the system disk.
OUTPUT DISK IS DIRTY - CANNOT PROCEED Use REFRESH to clean the disk or use DSKINT to reinitialize the disk.
OUTPUT DISK IS NOT A SYSTEM DISK Reinitialize the disk as a system disk.
OVERLAY HANDLER IN ERROR Submit an SPR.

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>OVR NOT IN SIL Regenerate.</p>
<p>PACK CLUSTER SIZE IS NOT 1, 2, 4, 8 OR 16 Reinitialize the disk.</p>
<p>PARITY CSR FOUND, BUT NOT RELATING TO MEMORY Call the DIGITAL field service engineer.</p>
<p>PARITY CSR CONTROLS MORE THAN 31K Call the DIGITAL field service engineer.</p>
<p>PERMANENTLY MAPPED REGION OF MONITOR IS TOO LARGE Regenerate to create a smaller monitor. Configure fewer small buffers or less resident code.</p>
<p>PRIORITY OF xx: INTERRUPT (PRn) IS TOO HIGH - DEVICE DISABLED Call the DIGITAL field service engineer to install the device correctly.</p>
<p>QUESTION ATTEMPTED DURING AUTO-RESTART If the system disk needs cleaning, start the system by typing START or LINE FEED, thereby cleaning the disk. If any other problem occurs, submit an SPR.</p>
<p>REQUESTED DISK DRIVE NOT FOUND Request a disk that is on the system.</p>
<p>REQUIRED MODULE xxxxxx HAS INVALID LENGTH Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx HAS INVALID TRANSFER ADDRESS Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx HAS WRONG LOAD ADDRESS Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx IS IMPROPERLY LINKED Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx IS MISSING A REQUIRED SYMBOL Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx NOT FOUND IN SIL Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx OUT OF SEQUENCE IN SIL Regenerate. If the error recurs, submit an SPR.</p>
<p>REQUIRED MODULE xxxxxx TOO LARGE Regenerate. If the error recurs, submit an SPR.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>REQUIRED MODULE xxxxxx TOO SMALL Regenerate. If the error recurs, submit an SPR.</p> <p>REQUIRED SYMBOL xxxxxx MISSING FROM MONITOR Regenerate. If the error recurs, submit an SPR.</p> <p>RSTS/E REQUIRES AT LEAST 32K WORDS OF MEMORY Your configuration is too small for the system.</p> <p>RSTS/E REQUIRES EIS! Correct the hardware configuration to include EIS.</p> <p>RSTS/E REQUIRES A CLOCK! Correct the hardware configuration to include a KW11P or KW11L clock.</p> <p>RSTS/E REQUIRES MEMORY MANAGEMENT HARDWARE! Correct the hardware configuration to include a memory management unit.</p> <p>SATT.SYS IS OVER 16 BLOCKS LONG If the file structure on the disk is sound, submit an SPR. Otherwise, reinitialize the disk.</p> <p>SATT.SYS NON-EXISTENT AT TIME OF READ If the file structure on the disk is sound, submit an SPR. Otherwise, reinitialize the disk.</p> <p>SATT.SYS NOT FOUND - RESULTS UNPREDICTABLE Rebootstrap the system and retry the operation. If the error recurs, reinitialize the disk.</p> <p>filnam.SIL NOT FOUND - PLEASE INSTALL A SIL The SIL that you previously installed is not on the disk. Use the INSTALL option to install a monitor SIL.</p> <p>SWAP.SYS NOT CONTIGUOUS OR TOO SMALL Use the REFRESH option to create a contiguous SWAP.SYS of the correct size.</p> <p>SWAP.SYS NOT PRESENT IN [0,1] Use the REFRESH option to create SWAP.SYS in [0,1]</p> <p>SYMBOL xxxxxx NOT FOUND IN MONITOR SYMBOL TABLE Regenerate. If the problem recurs, submit an SPR.</p> <p>SYSTEM DISK DISABLED IN MONITOR Use the SET option to enable the system disk device.</p> <p>TOO MANY BAE DEVICES Submit SPR.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>TOO MANY BLOCKS IN DEVICE CONFIGURATION TABLE Regenerate.</p> <p>TOO MANY xx: CONTROLLERS GENERATED Ensure that any edits to CONFIG.MAC are correct. If there are no editing errors, regenerate the system.</p> <p>TOO MANY RH OR UNIBUS DEVICES Submit an SPR.</p> <p>TOO MANY SUBLINES FOR nn: Ensure that any edits to CONFIG.MAC ARE correct. If there are no editing errors, regenerate the system.</p> <p>TOO MANY UNIBUS DEVICES ON THIS 11/70 First, disable unnecessary devices with HARDWR or SET. If the error condition persists, use DEFAULT to lock all memory addresses above 124K. (The computer runs in 18-bit addressing mode instead of 22-bit addressing mode.) If the error condition is present after you lock memory, or if the computer has 124K words of memory or fewer, submit an SPR.</p> <p>TOO MUCH MODIFICATION TO MONITOR REQUIRED Regenerate to produce a monitor that is more closely configured to your system hardware.</p> <p>TU58 DEVTBL PACKET MISSING Submit an SPR.</p> <p>UFD HAS SIZE TOO LARGE FOR FILE [P,PN] filnam.ext - FIXED BY CLEAN Warning message only. CLEAN corrects the condition.</p> <p>UFD HAS SIZE TOO SMALL FOR FILE [P,PN] filnam.ext - FIXED BY CLEAN Warning message only. Clean corrects the condition.</p> <p>[0,1] UFD NOT FOUND Reinitialize the disk.</p> <p>UNABLE TO CREATE REQUESTED FILE(S) Delete some files and retry the procedure, or request smaller files. Also, try using the FILE suboption of REFRESH to create files one at a time.</p> <p>UNEXPECTED OVERLAY TRAP Submit an SPR.</p> <p>UNEXPECTED TRAP THROUGH THE VECTOR AT nnn. TRAP OCCURRED FROM PC = nnnnnn. Submit an SPR.</p>

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
UNKNOWN DEVICE BOOTED Submit an SPR.
UNKNOWN RH CONTROLLER PRESENT Call the DIGITAL field service engineer to correct the hardware configuration.
UNRECOVERABLE DISK ERROR ON xxn: Ensure that the disk is mounted and write-enabled (if necessary). If the disk is properly mounted, this error indicates a fatal disk error. Reformat and reinitialize the disk, or use a different disk.
USER-DEFINED ADDRESS FOR DEVICE xxn: NOT FOUND - DEVICE DISABLED Use HARDWR to set the correct CSR address for xx:, or call the DIGITAL field service engineer.
USING KW11L CLOCK Warning message only. The installed SIL is configured for a KW11P clock, but the KW11P clock is not present. RSTS/E uses the KW11L clock instead.
USING KW11P CLOCK AT LINE FREQUENCY Warning message only. The installed SIL is configured for a KW11L clock, but the KW11L clock is not present. RSTS/E uses the KW11P clock instead.
VECTOR FOR DEVICE xx: (nnnn) ALREADY IN USE - DEVICE DISABLED Use HARDWR to set the correct vector address for xx:, or call the DIGITAL field service engineer.
WARNING - BAD BLOCK DOUBLY ALLOCATED TO BADB.SYS Warning message only.
WARNING - DCN IN BADB.SYS TOO BIG Warning message only. The file structure on the disk being cleaned appears invalid.
WARNING ** DBn: IS DUAL PORTED , PROCEED WITH CAUTION Warning message only.
WARNING - DCN IN BADB.SYS NOT ON PACK CLUSTER BOUNDARY Warning message only.
WARNING - xxx FILE NOT FOUND - PLEASE SET DEFAULTS The SIL you are installing specifies a default run-time system or error message file that is not present. Use the DEFAULT option to specify a valid file.

(continued on next page)

Table B-1: Initialization Code Error Messages (Cont.)

Message and Recovery Procedure
<p>WARNING - FILE filnam.ext NOT FOUND WHEN REMOVING OLD SIL Warning message only.</p> <p>WARNING - LINK IN BADB.SYS IS BAD. BAD BLOCKS MAY BE LOST Warning message only.</p> <p>WARNING - MAIN MEMORY (cache) DISABLED AT STARTUP. RSTS/E WILL NOT USE (cache). SYSTEM MAY RUN SLOWLY. Warning message only. (cache) is CACHE, CACHE GROUP 0, or CACHE GROUP 1.</p> <p>WARNING - MAIN MEMORY (cache) IS FAILING REPEATEDLY. RSTS/E WILL NOT USE (cache). SYSTEM MAY RUN SLOWLY. Warning message only. (cache) is CACHE, CACHE GROUP 0, or CACHE GROUP 1.</p> <p>WARNING - ODT IN THE SIL IS IN AN ILLEGAL FORMAT Warning message only.</p> <p>WARNING - SUBLINES for xx: ARE OUT OF SEQUENCE Warning message only, but the system may crash. Regenerate.</p>

B.2 RT11 Run-Time System Error Messages

The RT11 Run-Time System can generate the errors described in Tables B-2 and B-3.

Table B-2: Recoverable RT11 Run-Time System Errors

Message and Meaning
<p>?ADDR? You specified an illegal address with an E, D, or B command.</p> <p>?BAD LOAD? RT11 encountered an error when reading the program into memory.</p> <p>?BAD PPN? You typed an illegal PPN in response to a LIB or PPN command.</p> <p>?BAD START ADDRESS? The program start address was odd or out of bounds.</p>

(continued on next page)

Table B-2: Recoverable RT11 Run-Time System Errors (Cont.)

Message and Meaning	
?FIL NOT FND?	The specified file cannot be found.
?FILE?	You specified no filename or an illegal filename with the R, RUN, or GET command.
?ILL CMD?	You typed an illegal command. The command includes an unrecognized command, a syntax error in a command string, a command line longer than 510 characters, or an attempt to change to an illegal size (2<size<swap max for this job).
?ILL DEV?	The command string includes an illegal device name.
?NO RESTART?	You typed the RESTART command, but the program is not restartable.
?OVR CORE?	The program is too large to fit into memory.

Table B-3: RT11 Run-Time System Fatal Execution Errors

Message and Meaning	
?M-BPT TRAP	The program issued a BPT instruction, but the job has an illegal vector location.
?M-FP TRAP	A floating point trap occurred.
?M-HALT	The program halted.
?M-ILL EMT	The program issued an invalid monitor call.
?M-IOT TRAP	The program issued an IOT instruction, but the job has an illegal vector location.
?M-OVLY ERR	The RT-11 Run-Time System encountered an error while reading a program overlay. This message may indicate a hardware error.
?PROGRAM LOST - SORRY	An unrecoverable error occurred. The run-time system resets user core image.

(continued on next page)

Table B-3: RT11 Run-Time System Fatal Execution Errors (Cont.)

Message and Meaning
?M-TOO MANY OPEN CHANNELS The job attempted to open more than 15 channels or attempted to open two DECtape files at once.
?M-TRAP TO 4 The program trapped to the vector at location 4.
?M-TRAP TO 10 The program trapped to the vector at location 10.
?M-TRAP TRAP The job issued the TRAP instruction, but the job has an illegal vector location.

B.3 Batch Error Messages

During the execution of the batch streams associated with CREATE.SAV and SYSBAT.SAV, two types of errors may occur. The RT11 Run-Time System returns most errors. The batch program translates these errors to their BASIC-PLUS equivalents and prints the BASIC-PLUS error message text. You can find a discussion of BASIC-PLUS error messages in the *BASIC-PLUS Language Manual*. The batch program itself may generate several other errors. Table B-4 summarizes these errors.

Table B-4: BATCH Error Messages

Message and Meaning
CREATE NOT RUN FROM DISTRIBUTION MEDIUM The CREATE program can be run only from the distribution medium.
DEVICE NOT MOUNTED The batch stream attempted to dismount a device that was not mounted.
ERROR CHAINING TO \$SYSGEN.SAV The CREATE program could not find the \$SYSGEN.SAV file.
ERROR DURING PSEUDO KEYBOARD INPUT An error occurred during input to the pseudo keyboard.
ERROR DURING PSEUDO KEYBOARD OUTPUT An error occurred during output to the pseudo keyboard.
ERROR OPENING OR READING \$LOGIN.SAV The CREATE program could not find the \$LOGIN.SAV file on the distribution medium, or an error occurred while reading the file.

(continued on next page)

Table B-4: BATCH Error Messages (Cont.)

Message and Meaning
<p>ERROR OPENING OR READING SYSGEN.CTL FILE The SYSBAT program could not find the SYSGEN.CTL file, or an error occurred while reading the file.</p>
<p>INVALID CARD A line contained a command in an incorrect format.</p>
<p>INVALID SWITCH A switch used in the command field or in the specification field is undefined or in an incorrect format.</p>
<p>JOB FAILED TO LOG IN The job cannot log in because the LOGIN program or the requested account was missing.</p>
<p>LOGINS HAVE NOT BEEN ENABLED Logins must be enabled before the batch stream can be processed.</p>
<p>MOUNT ERROR The volume to be mounted was not correct (pack IDs do not match) or the device was already in use.</p>
<p>TOO MANY MOUNTED DEVICES The job requested the mounting of more than twelve devices.</p>
<p>TWO MAGTAPE UNITS HAVE THE SAME UNIT NUMBERS TU16/TU45 TU77/TE16 Examine unit switches on magnetic tape drives and ensure that no two have the same unit number.</p>

Appendix C

Auxiliary System Program Files

Certain auxiliary files are built during the system library build procedures. The program which builds each file is stored in the library along with the file. If, for any reason, the file is damaged or destroyed, the file can be created by running the related program as described in this appendix.

C.1 Character Generation File - CHARS.QUE

The line printer spooling program SPOOL requires the character generation file CHARS.QUE. The file is a virtual core array and is stored on the system disk during system generation by commands in the SPLER.CTL file. To create the CHARS.QUE file, first ensure that the old copy is deleted from the system library. The CHARS program terminates with an error if a file named CHARS.QUE exists in the system library directory. Next, run the CHARS program by typing the following command.

```
RUN $CHARS
CHARS      V7.0 RSTS V7.0 TIME SHARING
READY
```

After terminating, CHARS returns control to BASIC-PLUS as indicated by the READY message.

C.2 Batch Command Decoding File - BATCH.DCD

The BATCH system program requires the command decoding file BATCH.DCD. The file is a virtual array and is created during system build by commands in the SPLER.CTL file. To create the BATCH.DCD file, first ensure that the old copy is deleted from the system library. Run the BATDCD program while logged into the system under a privileged account. The program terminates with an error if a file named BATCH.DCD exists in the

system library directory. The following sample dialogue shows the proper procedure.

```
RUN $BATDCD
BATDCD  V7.0 RSTS V7.0 TIME SHARING

READY
```

After terminating, BATDCD returns control to BASIC-PLUS as indicated by the READY message.

C.3 Backup Prompt File - BACKUP.PRM

The BACKUP system package requires the file BACKUP.PRM to store prompting text for commands and special parameters for creating the work file. The file is created in the system library during system generation by commands in the BACKUP.CTL file. To create the BACKUP.PRM file, run the BACPRM.BAC program while logged into the system under a privileged account. The following sample dialogue shows the procedure.

```
RUN $BACPRM
BACPRM  V7.0 RSTS V7.0 TIME SHARING

READY
```

After terminating, BACPRM returns control to BASIC-PLUS as indicated by the READY message.

C.4 Error Package Data File - ERRBLD

The system error package requires the ERRDAT.FIL file to store the data used in the error package for processing errors. The file is created in the Error Package Library during system library build by commands in the BUILD.CTL file. To create the ERRDAT.FIL file, run ERRBLD.BAC program while logged into the system under a privileged account. The following sample dialogue shows the procedure.

```
RUN $ERRBLD
ERRBLD  V7.0 RSTS V7.0 TIME SHARING

READY
```

After terminating, ERRBLD returns control to BASIC-PLUS as indicated by the READY message.

Appendix D

Number Conversion

Many applications require a number based on bit values in a PDP-11 word. The following list shows the octal and decimal values for each bit in the PDP-11 word.

Bit	Octal Number	Decimal Value	Value
0		1	1
1		2	2
2		4	4
3		10	8
4		20	16
5		40	32
6		100	64
7		200	128
8		400	256
9		1000	512
10		2000	1024
11		4000	2048
12		10000	4096
13		20000	8192
14		40000	16384
15		100000	32768(32767+1)



Index

Where more than one page number appears for an entry, the defining entry is indicated with bold face type.

A

- ABORT command
 - SPOOL program, 5-27, 5-36
- Account
 - associated with system logical name, 7-38
 - automatic creation, 4-4
 - changing password of, 7-12
 - changing quota of, 7-12
 - creating individual with REACT, 4-2
 - deleting, 4-3
 - entering for restore mode, 8-3
 - removing all files from, 7-12
 - of Spooling Package Library, 5-6
- Account, privileged
 - assignment of, 1-8
 - DEVTST program, 11-1
 - job running under, 1-6
 - preventing unauthorized access, 1-8
 - programming error, 2-5
 - running DEVTST under, 11-2
- Accounts
 - on a private disk, 4-2
 - accounting information on, 4-5
 - on the system disk, 4-2
- ACCT.SYS file, 4-2, 4-4
- Action requests
 - BATCH, 5-42
 - OPSER, 5-10
 - table of, 5-11
- ADD command
 - errors, 7-27
 - UTILITY program, 3-10, **7-25**
 - UTILITY program options, 7-26 to 7-28
- ADD ERROR command
 - UTILITY program, 7-35
 - UTILITY program errors, 7-36
- ADD LIBRARY command
 - UTILITY program, **7-29**, 7-30
 - UTILITY program options, 7-30
- ADD LOGICAL command
 - UTILITY program, 7-12, **7-39**
- ADD OVERLAY command
 - UTILITY program, 7-35
 - UTILITY program errors, 7-36
- ADD SWAPFILE command
 - errors, 7-33
 - UTILITY program, 7-13, **7-32**
- Address
 - specifying load for run-time system, 7-26
- Address space
 - access to locations in, 6-19
 - change byte locations, 6-15
 - change word locations, 6-15
 - examine byte locations, 6-15
 - examine word locations, 6-15
- ALIGN option
 - SPOOL program, 5-38
- ANALYS, 6-1, **6-13**
 - crash dump data, 6-14
 - error code report, 6-14
 - error logging printouts, 6-14
 - header line, 6-13
 - invoking, 6-13
 - output of, 6-14
 - query line, 6-13
 - running after system crash, 3-12
 - use of, 6-13
- ANALYS.CMD file, 3-12
- ANSWER command
 - BATCH program, 5-42
 - OPSER program, 5-10
- ASCII files
 - patching source, 10-1
- ASCII format
 - printing with ODT, 6-23
- ASCII source code
 - patching, 10-5
- ASSIGN option
 - SPOOL program, 5-25
- ATTACH command
 - INIT program, 3-7
- AUTOED
 - CPATCH editor, 10-9
- Automated patching facility, 10-1, 10-2
- Automatic restart
 - procedures, 2-2
 - RSTS/E system, 2-2
 - system initialized in, 2-7

Autopatching kit, 10-1
Auxiliary run-time system, 7-24
 adding, 7-25
 command file to add, 3-10
 creation of, 7-25
 establishing, 3-10
 removing, 7-25

B

BAC file
 error during loading, 6-12
 improper size, 6-12
 size of a, 6-13
BACDSK program, 8-18
BACKUP, 8-1
 ACCESS comparison, 8-6
 as an OPSER controlled program, 5-46
 attached to a terminal, 5-46
 bad block during transfer, 8-25
 bad block error, 8-24
 bad block file creation, 8-25
 batch control file, 8-8
 blank in specification line, 8-7
 building listing file, 8-4
 bypass tape label checking, 8-16
 comparison with SAVE/RESTORE, 9-1
 CONT prompt, 8-7
 creating indirect command file, 8-5
 CREATION comparison, 8-6
 dialogue command errors, 8-18, 8-19
 dialogue example, 8-26, 8-27
 disk bad block information, 8-18
 disk structure, 8-18
 dismount message, 8-17
 error handling, 8-18
 error handling routines, 8-22, 8-23, 8-25
 error processing, 8-22
 errors possible with, 8-18
 exempt files from, 8-7
 file comparison, 8-4
 file deletion, 8-4
 file selection, 8-2
 file selection after error, 8-23
 file specification, 8-5, 8-6t
 file transfer, 8-3
 files larger than 65535 blocks, 8-23
 format on disk, 8-18
 from an indirect command file, 8-4
 indirect command files, 8-2
 informational messages, 8-25
 interruption command errors, 8-18, 8-21
 interruption command prompt, 8-15
 interruption commands, 8-15
 labelling information, 8-16

BACKUP, (cont.)

 line continuation, 8-7
 listing file example, 8-30 to 8-33
 logic errors, 8-25
 magnetic tape mount procedure, 8-17
 modes, 8-1
 multiple disks in public structure, 8-3
 operations within, 8-2
 placed files, 8-24
 processing errors, 8-19
 prompt file, C-2
 running, 8-4
 running detached, 5-5, 5-46
 running under BATCH, 8-8
 selection errors, 8-23
 sort files in UFD, 8-2
 status report, 8-29
 submitting BATCH job, 8-8
 and system management, 8-1
 volume mount errors, 8-18, 8-21
 wildcard characters, 8-5
 work file, 8-2
BACKUP command
 interruption, 8-15
 through OPSER, 5-47t
 SAVE option, 8-5
 SCRATCH switch, 8-8
 STATUS interruption, 8-29
 table of interruption, 8-15, 8-16
BACKUP command line
 include comment in, 8-8
BACKUP dialogue, 8-2
 for nonprivileged users, 8-9
 for privileged users, 8-9
 summary of, 8-9, 8-10
Backup index volume
 dismounting, 8-43
BACKUP keyword
 ACCESS, 8-5
 AFTER, 8-5
 BEFORE, 8-5
 CREATION, 8-5
 EXCEPT, 8-7
Backup mode
 dialogue example, 8-26, 8-27
Backup set
 auxiliary index file, 8-3
 create on magnetic tape, 8-26
 expiration date, 8-16
 identifier for, 8-16
 primary index file, 8-3
 restoring from disk to tape, 8-35
 restoring from tape to disk, 8-35
 secondary index file, 8-3
 supersede files, 8-37

- Backup volume, 8-16
 - dismounting, 8-16
 - mounting, 8-16
- BACKUP.PRM backup prompt file, C-2
 - creating a, C-2
- Bad block file
 - BACKUP creation, 8-25
- Bad blocks
 - on BACKUP disk, 8-18
 - during BACKUP, 8-25
 - content of report, 6-7
 - criteria for, 6-7
 - delete files that contain, 7-57
 - ERRDIS report, 6-7
 - list of potential, 6-4
 - during SAVE/RESTORE, 9-16
 - on SAVE/RESTORE output, 9-33
- BADB.SYS file
 - zero, 7-53
- BASIC-PLUS
 - edit mode, 1-3
 - immediate mode statement, 1-3
 - patch source files, 10-5
 - patches to library programs, 10-5
 - in run mode, 1-4
 - run-time system, 7-24
- BASIC-PLUS commands
 - cause INIT to execute, 3-3
- BASIC-PLUS, Run-Time System, 1-3
 - operation of, 1-3
- BATCH, 5-4, 5-5, 5-38
 - answering action requests, 5-42
 - command decoding file, C-1
 - communicate with, 5-41
 - communication paths, 5-4
 - compiled version of, 5-39
 - default conditions for, 5-40
 - default receiver identification, 5-40
 - device type designators, 5-43t
 - dismount a volume, 5-43
 - error checking, 5-40
 - error messages, B-15, B-16
 - interrupt commands, 5-41
 - invoking, 5-39
 - logical device name, 5-39
 - modules, 5-38
 - mount requests, 5-42
 - processing command file, 5-42
 - pseudo keyboard, 5-39
 - requests for operator action, 5-42
 - running BACKUP under, 8-8
 - in Spooling Package Library, 5-39
 - start up options, 5-40
 - start up procedures, 5-42
 - system program, C-1
- BATCH command
 - ANSWER, 5-42
 - CONTINUE, 5-43
- BATCH MUST BE COMPILED message, 5-39
- Batch processor, 5-38
- BATCH.DCD decoding file, 5-4, C-1
 - creating a, C-1
- Baud rate
 - changing, 7-53
 - on keyboard, 7-53
- Binary code
 - install patches to, 10-1
 - patching, 10-1
- Block address
 - ODT, 6-25
- BM792-YB
 - bootstrap procedure, A-4
- BM873-YA
 - bootstrap procedure, A-2
- BM873-YB
 - bootstrap procedure, A-3
- Boot start address, A-8
- Bootable media
 - create with SAVE/RESTORE, 9-26
- Bootstrap
 - record, A-1
 - RSTS/E after system halt, 2-2
 - RSTS/E into memory, 2-1
 - write a dummy block, 7-53
- Bootstrap addresses
 - summary of hardware, A-2
- Bootstrap procedure, 2-2, A-1
 - BM792-YB, A-4
 - BM873-YA, A-2
 - BM873-YB, A-3
 - H324 pushbutton panel, A-3
 - M9301-YA, A-5
 - M9301-YB, A-5
 - M9301-YC, A-6
 - M9301-YF, A-7
 - M9312, A-7
 - MR11-DB, A-4
 - RK07, A-13
 - RL01, A-13
 - RL02, A-13
 - RSTS/E, 2-1
- Bucket size
 - cache, 7-22
 - RMS, 7-23
- Buffers, free
 - display program status, 7-48
- Buffers, small, 7-13
- BUILD, 10-16
 - CUSP COMPILER question, 10-17
 - install run-time system, 10-16

BUILD, (cont.)
 preprocessing phase, 10-17
Burst pages, 5-26
BYE command
 INIT program, 3-6

C

Cache
 clusters in, 7-17
 data block in, 7-16
 size of, 7-17
 space for, 7-17
 updated by the monitor, 7-17
Cache cluster
 eligible for replacement, 7-18
 last block in, 7-18
 size of a, 7-17, 7-20
 specify size of, 7-20
Cache clusters
 specify total number of, 7-21
Caching
 amount to be performed, 7-20
 checks by monitor, 7-20f
 data, 7-16. *See also data caching*
 default settings, 7-21
 directory, 7-16
 efficiency of, 7-22
 enable for BACKUP work file, 8-28
 guidelines, 7-22, 7-23
 MODE checks, 7-22
 override UFD entry, 7-22
 random mode, 7-17, 7-18
 sequential mode, 7-17, 7-18
 specify for specific file, 7-21
 types of, 7-16
 UFD entry marked for, 7-18
Catastrophic error, 2-2, 2-5
 automatic recovery from, 2-6
 causes of, 2-5
 handling of, 2-6
CCL command, 7-36
 abbreviations in, 3-11
 to add a, 7-37
 defined at start of time sharing, 7-37
 defining with UTILTY program, 7-36
 definition, 3-10, 7-37
 DISMOUNT, 7-36
 error checking, 7-37
 listing, 7-38
 MOUNT, 7-36
 PLEASE as, 5-45
 programs run by, 7-37
 redefine a, 3-11

CCL command, (cont.)
 to remove a, 7-38
 UTILTY program, 7-8, 7-36, 7-37
CCL.CMD file, 3-10
Central Processing Unit. *See CPU*
CHAIN statement
 immediate mode, 1-4
CHANGE command
 UTILTY program, 7-12
CHANGE LOGICAL command
 UTILTY program, 7-12, 7-39
CHARS.QUE character generation file, 5-4,
 5-36, C-1
 creating, C-1
Checksum, 6-12, 10-7
Checksum error, 6-12
 causes of, 6-12
Checksum number
 creation of a, 10-8
CLEAN command
 UTILTY program, 7-11
Cluster size
 cache and pack relationship, 7-22
 cache and RMS bucket size, 7-22
 UFD, 4-7
Command file
 BACKUP, 8-2
 building for ONLPAT, 10-3
 CPATCH format, 10-9
 creating for CPATCH, 10-5
 modifying for patches, 10-4
 for patching kits, 10-1
 PBUILD, 10-10
Command file mode
 ONLPAT in, 10-2
 ONLPAT procedures, 10-2
Command level
 system, 1-3
 terminal at, 1-3
Commands
 forced to terminal, 3-7
 INIT program, 2-7, 3-2
 system, 1-3
Compiled file
 as privileged, 1-7
 protection bit, 1-6
Concise Command Language. *See CCL*
Console emulator
 M9312, A-8
Console switch
 register settings, A-12t
Console terminal
 jobs logged in at, 7-13
 used to start system jobs, 3-7

- CONTINUE command
 - BATCH program, 5-43
- Control characters
 - CPATCH, 10-9
- Control file
 - contents of BACKUP, 8-8
 - INIT program, 3-3
 - order of operations, 3-6
 - QUEMAN start up, 5-20
 - system crash, 3-12
 - system start up, 3-11
- CPATCH, 10-1, 10-5
 - AUTOED line editor, 10-9
 - checksum number, 10-8
 - command file, 10-6
 - command file format, 10-9
 - control characters, 10-9
 - creating command files, 10-5, 10-6
 - CS switch, 10-8
 - CTRL/Z to, 10-5
 - EDIT command, 10-7
 - error messages, 10-19, 10-21t
 - EXIT command, 10-5
 - exit from, 10-5, 10-7
 - FILE TO PATCH prompt, 10-6, 10-18
 - header line, 10-6
 - PATCH COMPLETE message, 10-7
 - patch file names, 10-6
 - PATCH switch, 10-7
 - prompt, 10-5, 10-7
 - recalculate checksum, 10-8
 - running stand alone, 10-6
 - source text editing, 10-12
 - text buffer, 10-6
 - verify patching operation, 10-7
- CPEXER
 - computation errors, 11-6
 - consistency failure, 11-7
 - DEVTST program, 11-4
 - dialogue, 11-5
 - error types, 11-5
 - instruction errors, 11-6
 - test floating point processor, 11-4
 - test PEEK function, 11-4
 - tests included in, 11-5
 - trap failure, 11-6
 - verify CPU integrity, 11-4
- CPU
 - access to, 7-15
 - console switches, A-1
 - determine the state of, 2-4
 - exerciser, 11-4
 - place heavy load on, 11-4
- CPU ROM, A-7

- CPU time, 4-5
- CPUTST
 - DEVTST program, 11-7
 - dialogue, 11-7
 - errors, 11-8
- Crash control file
 - order of operations, 3-6
- Crash file
 - information from a, 6-1
- Crash, system, 2-2, 2-5
 - causes of, 2-5
 - command file for recovery, 3-8
 - handling of, 2-6
 - random, 2-5
 - recovery, 3-12
- CRASH.CTL file, 2-7, 3-12
- CRASH.SYS file, 6-1, 6-13
- CTRL/C
 - forced to INIT, 3-7
 - to QUEMAN program, 5-17
 - in SAVE/RESTORE dialogue, 9-7
 - to terminate a patch, 10-2
 - to terminate UTILTY program, 7-8
 - typed at terminal, 1-3
- CTRL/Z
 - to CPATCH program, 10-5
 - in SAVE/RESTORE dialogue, 9-7
 - to terminate UTILTY program, 7-8

D

- Data
 - verify transferred, 9-15
- Data access, 1-1
 - cached, 7-17
- Data block
 - in cache, 7-16
 - retrieval pointer, 7-58
- Data caching, 7-16, 7-17. *See also caching*
 - control, 7-16
 - disabling, 7-20
 - enable on the system, 7-19
 - MODE values, 7-17
 - monitor directives, 7-17
 - read operations, 7-16
 - on RMS files, 7-23
 - support for, 7-16
 - turn off for DSKEXR program, 11-8
 - write operations, 7-16
 - XBUF allocation, 7-17
- DATE command
 - UTILTY program, 7-16
- DECtape exerciser, 11-11

- DELETE function
 - dialogue, 4-4
 - REACT program, 4-3
 - responses to, 4-4
- Delimiter
 - keyboard, 7-15
- Density
 - printed for save tape set, 9-29
- Description block
 - run-time system, 7-25
- DET command
 - prompt, 5-17
 - QUEMAN program, 5-17
- DETACH command
 - INIT program, 3-7
 - UTILITY program, 7-16
 - UTILITY switches, 7-16
- DETACH option
 - PBUILD program, 10-14
- Detached job
 - in hibernate state, 7-42
- Device
 - bootstrapping a, A-1
 - exercisers, 11-1
- Device designators, 11-3
 - table of BATCH, 5-43
- Device I/O
 - jobs waiting for, 1-4
- Device name
 - mnemonics, A-11
- Device testing program
 - DEVTST, 11-1
- Device, file-structured
 - fixed head disk as, 1-3
- Device, logical
 - BATCH, 5-39
 - SPOOL, 5-23
- Devices, busy
 - display program status, 7-48
- Devices, printing
 - adjustable top of form, 5-27
- DEVTST
 - dialogue conventions, 11-2
 - error messages, 11-4
 - exercisers, 11-1
 - operating procedures, 11-2
 - package of programs, 11-1
 - privileged account, 11-1
 - running in system library account, 11-2
 - running under privileged account, 11-2
 - table of programs, 11-3
- DEVTST program, 11-1
 - CPEXER, 11-4
 - CPUTST, 11-7
- DEVTST program, (cont.)
 - DSKEXR, 11-8
 - DSKSEK, 11-9
 - DTEXER, 11-11
 - DXEXER, 11-12
 - KBEXER, 11-13
 - LPEXER, 11-14
 - MTEXER, 11-14
 - PPEXER, 11-15
 - PREXER, 11-16
- DLENGTH option
 - effect on line printer, 5-27
 - effect on terminals, 5-28
 - SPOOL program, 5-26
- Directory
 - delete invalid, 7-57
 - fragmentation, 1-2
 - minimize search overhead, 1-2
 - reducing accesses, 7-58
 - of SAVE/RESTORE, 9-15
- Directory caching, 7-16, 7-17
 - enable on the system, 7-19
 - optimize, 7-23
- Directory structure
 - optimizing on disk, 7-58
 - prevent a damaged, 7-60
- DISABLE CACHE command
 - UTILITY program, 7-21
- DISABLE KB command
 - UTILITY program, 7-16
- Disk
 - access files on locked, 1-5
 - add to system, 7-9
 - BACKUP format on, 8-18
 - changing logical name, 7-39
 - clean with ONLCLN program, 7-11, 7-56
 - copies of system files, 8-1
 - copy to one that contains bad blocks, 9-2
 - copying with IMAGE, 9-9, 9-10, 9-20
 - create copy of, 9-2
 - create files on, 1-2
 - directory entries scattered, 7-58
 - dismount under SHUTUP, 3-19
 - DSKINT operations, 7-53
 - exerciser, 11-8, 11-9
 - file structure on, 1-2
 - fixed head, 1-3
 - initialize nonfile structured, 7-53
 - logically mount with UTILITY program, 7-9
 - optimization, 1-2
 - optimizing directory structure, 7-58
 - organization, 1-1
 - preparing for use on drive, 7-8
 - private, 1-2

- Disk, (cont.)
 - procedure to mount, 7-9
 - public, 1-1
 - rebuild backed up files on, 8-1
 - reinitialize during time sharing, 7-53
 - remove from system, 7-9
 - reorder in public structure, 7-60
 - repair corrupt file structures, 7-56
 - restoring, 9-9, 9-10, 9-16
 - for save set, 9-2
 - saving, 9-9
 - SYSTAT status report, 7-10
 - system, 1-1
- Disk cartridge
 - logically mounted, 7-8
- Disk error
 - during loading of BAC file, 6-12
- Disk file
 - deletion of privileged, 1-7
 - preserving, 8-1
- Disk management
 - principles of, 7-8
 - UTILITY program, 7-8
- Disk mounting
 - UTILITY options, 7-11
- Disk pack
 - logically mounted, 7-8
 - remove a private, 7-10
 - replace a private, 7-10
- Disk storage
 - quota of, 7-12
- Disk structure
 - display program status, 7-48
 - public, 1-1
- Disk, file structured
 - pack label, 6-27
- Disk, fixed head
 - as file-structured device, 1-3
 - for swapping storage, 1-3
- Disk, private
 - accounts on a, 4-2
- Disk, system
 - accounts on the, 4-2
 - contents of, 1-1
 - MFD on, 1-2
- Diskette exerciser, 11-12
- DISMOUNT command
 - CCL, 7-36
 - UTILITY program, 7-10
- Dismount, logical
 - device, 7-10
- Display program
 - busy device statistics, 7-48
- Display program, (cont.)
 - commands, 7-44 to 7-46
 - detached, 7-43
 - disk structure, 7-48
 - free buffer status, 7-48
 - header line, 7-47
 - job status, 7-47
 - memory status, 7-47
 - memory status abbreviations, 7-48
 - message receiver status, 7-48
 - options, 7-43
 - resident library statistics, 7-48
 - run-time system statistics, 7-48
 - screen layout, 7-44
 - terminate the, 7-43
- DSKEXR
 - DEVTST program, 11-8
 - dialogue, 11-8
 - errors detected by, 11-9
 - test file structured disks, 11-8
 - turn off data caching, 11-8
- DSKINT, 7-53
 - error text, 7-55
 - header line, 7-54
 - long form questions, 7-54
 - operations on disk, 7-53
 - PROCEED prompt, 7-55
 - processing, 7-55
 - prompts, 7-54
 - recommended procedures, 7-56
 - running, 7-54
 - sample dialogue, 7-54
 - short form questions, 7-54
 - terminating, 7-54
 - unexpected errors, 7-55
 - write access to disk, 7-54
- DSKSEK
 - DEVTST program, 11-9
 - dialogue, 11-10
 - errors detected by, 11-10
 - test non-file structured disks, 11-9
- DTEXER
 - DEVTST program, 11-11
 - dialogue, 11-11
 - errors, 11-11
 - test DECTape, 11-11
- Dump
 - annotated version of memory, 6-14
 - crash data, 6-14
- DXEXER
 - DEVTST program, 11-12
 - dialogue, 11-12
 - errors detected by, 11-12

E

- EDIT command
 - CPATCH, 10-7
- Edit mode
 - BASIC-PLUS, 1-3
 - terminal returned to, 1-4
- ENABLE CACHE command
 - UTILITY program, 7-17, 7-19
 - UTILITY program options, 7-20, 7-21
 - UTILITY switches, 7-19
- END command
 - INIT program, 3-6
 - OPSER program, 5-46
- END statement
 - PBUILD program, 10-12
- ENTER function
 - dialogue, 4-2
 - REACT program, 4-2
 - responses to, 4-3
- ERRBLD, 6-5
 - header information, 6-5
 - running, 6-5
- ERRBLD.FIL file
 - creating a, C-2
- ERRCPY, 3-6, 6-2, 6-3
 - activating, 6-2
 - active during time sharing, 6-2
 - error message from SHUTUP, 6-3
 - messages queued to, 6-3
 - minimize size of, 6-1
 - number of messages queued to, 6-1
 - processing error messages, 6-3
 - shutdown procedures, 3-18
 - shutdown under SHUTUP, 3-18
 - total errors received by, 6-7
 - use of, 6-2
- ERRCRS.FIL file, 6-1
- ERRDAT.FIL file, 6-5
 - processing errors, C-2
- ERRDET, 6-7
 - content of report, 6-7
 - criteria for bad blocks, 6-7
- ERRDIS, 6-4
 - bad block report, 6-7
 - categories of errors, 6-7
 - detailed report, 6-4
 - dialogue explanations, 6-5, 6-6
 - disk error detailed description, 6-9, 6-10
 - error code mnemonic, 6-7
 - error sequence number, 6-7
 - error title line, 6-7
 - full report, 6-7
 - functions of, 6-4
 - help report, 6-6

ERRDIS, (cont.)

- modules, 6-4
- nondisk peripheral device format, 6-10
- nonperipheral error format, 6-11
- optional modes of, 6-5
- potentially bad blocks, 6-4
- running, 6-5
- running for first time, 6-5
- summary report, 6-4, 6-7
- user description data, 6-8
- zero error file, 6-4

ERRINT, 3-6, 6-1

- change the maximum size of, 6-2
- control file, 6-2
- example of dialogue, 6-3
- invoking, 6-2
- location of program, 6-2

ERRLOG.FIL file, 6-1, 6-2

- control information, 6-4
- data blocks, 6-3
- header, 6-3
- header contents, 6-4

Error checking

- CCL command, 7-37

Error code

- system crash, 6-15

Error detection

- RSTS/E monitor, 6-1

Error file

- adding, 7-35
- description of, 6-3
- ERRDAT.FIL, 6-5
- initialization of, 6-2
- removing, 7-35
- validation of, 6-2
- zero, 6-4

Error log

- report, 6-14
- for start up, 6-11

Error logging, 2-5, 6-3

- ANALYS printouts, 6-14
- creation of file, 6-1
- hardware errors, 6-1
- initial conditions for system, 3-6
- initialization, 6-2
- software errors, 6-1
- use of programs, 6-2

Error messages

- BATCH, B-15, B-16
- compilation of, 6-1
- CPATCH, 10-19
- DEVTST programs, 11-4
- extraction of, 6-1
- formatting of, 6-1

Error messages, (cont.)

- initialization code, B-1, B-2 to B-13
- PBUILD, 10-19
- processing of, 6-1
- processing with ERRCPY, 6-3
- PROGRAM LOST SORRY, 6-11
- retention of, 6-1
- RT11 Run-Time System, B-13 to B-15
- SAVE/RESTORE, 9-33 to 9-37
- SPOOL program, 5-37
- SPOOL syntax, 5-29
- system generation, B-1
- Error package, system
 - programs in, 6-1
- Error procedure
 - ODT, 6-27
- Error trap, 2-5
 - infinite loop of, 2-7
- Error, catastrophic, 2-2, 2-5
 - automatic recovery from, 2-6
 - causes of, 2-5
 - handling of, 2-6
- Errors
 - BACKUP dialogue, 8-19
 - BACKUP interruption command, 8-21
 - BACKUP volume mount, 8-21
 - categories of reported, 6-7
 - checksum, 6-12
 - detailed report of, 6-4
 - DEVTST dialogue, 11-4
 - disk, 6-7, 6-9, 6-10
 - displaying, 6-4
 - DSKINT program, 7-55
 - encountered in INIT, 3-2
 - missed, 6-8
 - nondisk peripheral device, 6-7
 - nonperipheral, 6-7, 6-11
 - ONLCLN program, 7-56
 - possible with BACKUP, 8-18
 - SAVE/RESTORE program, 9-4, 9-32
 - shutdown, 6-8
 - summary report of, 6-4
 - total received by ERRCPY, 6-7
 - unrecoverable, 6-12
- Errors, line printer
 - recovery from, 5-35
- Errors, programming, 2-5
 - privileged account, 2-5
- EXIT command
 - CPATCH program, 10-5
 - TTYSET program, 7-52
 - UTILITY program, 7-8
- Extension
 - change run-time system default, 7-28

Extension, (cont.)

- patch file defaults, 10-12
- .RTS, 7-24

F

File

- access on locked disk, 1-5
- access to, 1-2
- accounting information for, 5-36
- associate with run-time system, 7-29
- auxiliary system program, C-1
- BACKUP comparison, 8-4
- BACKUP deletion, 8-4
- backup larger than 65535 blocks, 8-1, 9-2
- BACKUP of placed, 8-24
- BACKUP transfer, 8-3
- caching, 7-23
- caching in random mode, 7-18
- caching in sequential mode, 7-18
- compiled protection bit, 1-6
- create on disk, 1-2
- directory entries, 7-58
- disk structure, 1-1, 1-2
- high access, 7-23
- indirect command, 3-8
- locked blocks on update, 1-5
- logical structure of, 1-1
- for off-line storage, 8-1
- PBUILD command lines to patch, 10-11
- privileged, 1-7
- read from sequentially cached, 7-18
- remove from UFD, 4-3
- removing from account, 7-12
- restoring from BACKUP, 8-35
- run request with no extension, 7-28
- specify caching parameters for, 7-21
- store blocks from, 7-16
- swapping, 1-4, 7-32
- terminal line speed characteristics, 7-49
- File data
 - minimize access overhead, 1-2
- File header, 5-36
- File specification
 - BACKUP, 8-5
- File, contiguous
 - bad cluster while transferring, 9-16
 - caching, 7-23
- File, large
 - display contents with ODT, 6-25
- File, privileged
 - process through QUE, 1-7
- Files, system
 - copies on tape and disk, 8-1

- Fixed head disk, 1-3
 - as file-structured device, 1-3
 - for swapping storage, 1-3
- FLAG command
 - UTILITY program, 7-21
 - UTILITY program options, 7-21, 7-22
- FORCE command
 - INIT program, 3-3, 3-7
 - UTILITY program, 7-14
- FORCE statement
 - PBUILD program, 10-11
- FORM command
 - SPOOL program, 5-37, 5-38
- Form length
 - line printer, 5-27
- FORM option
 - SPOOL program, 5-25
- Forms alignment, 5-26, 5-37
 - changing, 5-37
 - procedures, 5-26
 - during time sharing, 5-26
- Forms control, 5-26
- FREE block count
 - SYSTAT program, 7-41

G

- GRIPE, 7-62
 - exit from, 7-63
 - invoking, 7-63
 - output device, 7-63
- GRIPE command
 - LIST, 7-63
 - RESET, 7-63
- GRIPE.TXT file
 - clear the contents of, 7-63
 - examine the contents of, 7-63

H

- H324 pushbutton panel
 - bootstrap procedure, A-3
- HANGUP command
 - UTILITY program, 7-16
- Hardware
 - bootstrap procedures, A-1
 - exercise on system, 11-1
 - malfunctions, 2-5
 - summary of bootstrap addresses, A-2
 - verify reliability of, 11-1
- Hardware errors
 - information on, 11-1
 - logging of, 6-1
 - stimulate on system, 11-1

- Hardware loader, A-1
- Header page, 5-36
- Heading burst pages, 5-26
- HELP command
 - UTILITY program, 7-8
- Help report
 - ERRDIS program, 6-6
- Hibernation, 7-42
- HUNG TTY count
 - SYSTAT program, 7-41

I

- IDENTIFY
 - dialogue, 9-24, 9-25t
 - example of, 9-25
 - function of SAVE/RESTORE, 9-24
 - label information for save set, 9-26
 - procedures, 9-25
- IMAGE
 - dialogue, 9-21t
 - dismount request, 9-24
 - example of, 9-22, 9-23
 - extracting pack id, 9-24
 - function of SAVE/RESTORE, 9-20
 - run statistics, 9-24, 9-31
 - selecting, 9-23
 - summary report, 9-24
- Immediate mode
 - BASIC-PLUS, 1-3
 - CHAIN statement, 1-4
 - shutting down OPSER, 3-18
- Index file, 8-3
 - auxiliary, 8-3
 - copy a primary, 8-41
 - listing the, 8-44
 - loading the, 8-41
 - primary, 8-3
 - print directory information, 8-1
 - rebuild primary, 8-1
 - secondary, 8-3
- Indexed file, RMS
 - caching, 7-23
- Indirect command file, 3-8
 - to add auxiliary run-time systems, 3-10
 - BACKUP from, 8-4
 - for crash recovery, 3-8
 - creating BACKUP, 8-5
 - INIT program, 3-8
 - REORDR from, 7-60
 - to set terminal characteristics, 3-9
 - for system start up, 3-8
- INI command
 - QUEMAN program, 5-16

INIT, 2-7, 3-1

- BYE command in command file, 3-8
- control file, 3-1, 3-3
- control file commands, 3-4t
- CTRL/C forced to, 3-7
- END command in command file, 3-8
- error encountered, 3-2
- execute BASIC-PLUS commands, 3-3
- execution of, 3-1
- indirect command file, 3-8
- order of operations, 3-5
- program commands, 3-2
- run during time sharing, 3-3
- running attached, 3-2
- running detached, 3-2
- to start OPSER program, 5-13
- terminate without error, 3-6
- terminating, 3-2

INIT command, 2-7

- ATTACH, 3-7
- BYE, 3-6
- DETACH, 3-7
- END, 3-6
- FORCE, 3-3, 3-7
- LOGIN, 3-3
- LOGINS, 3-3
- MOUNT, 3-2
- processing FORCE, 3-7
- SEND, 3-3
- sequence of FORCE, 3-7

INIT.SYS, 2-1

Initialization code

- bootstrapped into memory, 2-1
- error messages, B-1, B-2 to B-13
- routines, B-1
- RSTS/E, 2-1
- START option, 3-1

Initialization options, 2-3

- requesting a, 2-3
- RSTS/E, 2-1
- summary of, 2-3, 2-4

INTERRUPT command

- OPSER program, 5-10, 5-12, 5-17
- response to, 5-12

Interrupt commands

- BACKUP, 8-15
- BATCH, 5-41
- SPOOL, 5-30

J

Job

- accounting information for, 5-36
- compute bound, 7-15

Job, (cont.)

- controlling, 7-13, 7-14
- currently running, 1-4
- decrease priority of, 7-15
- default run-time system, 7-24
- display program status, 7-47
- executing a read request, 7-16
- logged out, 1-6
- maximum size assigned to, 7-15
- priority of, 5-33
- run burst, 5-33
- running a privileged program, 1-6, 1-7
- running of a, 7-14
- running under privileged account, 1-6
- setting maximum size, 7-14
- setting priority, 7-14
- setting run burst, 7-14
- stalled, 7-41
- storage of, 1-1
- temporarily privileged, 1-7
- terminate a user, 7-14
- waiting for device I/O, 1-4
- waiting for keyboard input, 1-4

Job area

- size of, 1-3
- user, 1-3

Job header, 5-36

JOB MAX

- set at system start up, 7-13

Job, on line

- OPSER checks, 5-5
- OPSER table, 5-1

Job, privileged

- capabilities of, 1-5

Job, system

- console terminal to start, 3-7

K

KB command

- TTYSET program, 7-49, 7-51

KBEXER

- ASCII pattern test, 11-13
- DEVTST program, 11-13
- dialogue, 11-13
- errors detected by, 11-13
- repeat test, 11-13
- space test, 11-13
- worst case test, 11-13

KCT, 4-5

Keyboard

- baud rates on, 7-53
- controlling, 7-16

- Keyboard, (cont.)
 - default SPOOL values, 5-28
 - delimiters, 7-15
 - errors in KBEXER patterns, 11-13
 - exerciser, 11-13
 - malfunctioning, 7-41
 - monitor, 1-3
 - pseudo, 5-4
 - spooling, 5-28
 - start up on LA180, 5-34
 - start up on LA36, 5-33
- Keyboard input
 - jobs waiting for, 1-4
- Keyboard mode
 - ONLPAT procedures, 10-2
 - using ONLPAT in, 10-1
- KILL command
 - UTILITY program, 7-14
- Kilo Core Tick. *See* KCT

L

- LA180
 - form length, 5-27
 - start up keyboard on, 5-34
- LA36
 - start up keyboard on, 5-33
- Library device, 10-14
- Library programs
 - patches to BASIC-PLUS, 10-5
- Line printer
 - default SPOOL values, 5-28
 - DFLENGTH option effect, 5-27
 - form length, 5-27
 - output, 5-36
 - recovery from errors, 5-35
 - requests for output, 5-4
 - SPOOL spooling program, C-1
 - spooling, 5-22, 5-28
 - start up with all defaults, 5-31
 - start up with narrow width, 5-32
 - test with LPEXER, 11-14
- Line speeds
 - restricting, 7-50
- LIST CACHE command
 - UTILITY program, 7-19
- LIST CCL command
 - UTILITY program, 7-38
- LIST command
 - GRIPE program, 7-63
 - OPSER program, 5-11
- LIST JOBS command
 - OPSER program, 5-11, 5-45

- LIST LOGICAL command
 - UTILITY program, 7-40
- List mode
 - dialogue summary, 8-13, 8-14
 - invoke dialogue for, 8-45
 - listing file example, 8-46, 8-47
 - listing file for, 8-45
- LIST OPERATORS command
 - OPSER program, 5-11
- Listing file
 - generation of, 8-4
 - print BACKUP, 8-28
- LOAD command
 - UTILITY program, 7-28
 - UTILITY program options, 7-28, 7-29
- LOAD LIBRARY command
 - UTILITY program, 7-31
 - UTILITY program options, 7-31
- Loadindex, 8-41
 - dialogue, 8-41, 8-42
 - dialogue summary, 8-12, 8-13
 - example of, 8-42
 - listing file example, 8-43, 8-44
 - summary of, 8-41
- LOGFILE command
 - OPSER, 5-9
- Logged in jobs
 - controlling, 7-13
- Logical names, 7-38
 - account number associated, 7-38
 - adding new, 7-39
 - changing for disk, 7-39
 - job related, 7-38
 - listing system, 7-40
 - removing, 7-39
 - system defined, 7-38
- LOGIN command
 - INIT program, 3-3
- LOGIN priority, 7-15
- Logins
 - adjust the number of allowed, 7-13
 - command to enable, 3-6
 - decrease the number of, 7-34
 - disable, 7-13
 - limit under SHUTUP, 3-15
 - restricted by swapping space, 7-34
- LOGINS command
 - INIT program, 3-3
 - UTILITY program, 7-13, 7-34
- LPEXER
 - DEVTST program, 11-14
 - dialogue, 11-14
 - errors, 11-14

M

M9301-YA
 bootstrap procedure, A-5
 device codes for, A-6
M9301-YB
 bootstrap procedure, A-5
 device codes for, A-6
M9301-YC
 bootstrap procedure, A-6
 device codes for, A-6
M9301-YF
 bootstrap procedure, A-7
 device codes for, A-7
M9312
 bootstrap procedure, A-7
 bootstrap ROM configuration, A-8 to A-11
 bootstrapping with, A-11
 using console emulator, A-8
Magnetic tape
 backing up system onto, 8-16
 backup label on, 8-17
 BACKUP mount procedure, 8-17
 copies of system files, 8-1
 create backup set on, 8-26
 MTEXER tests, 11-14
 parity error during bootstrap, A-5
 for save set, 9-2, 9-27
MAKSIL program, 7-29
Master File Directory. *See* MFD
Memory
 adding resident library, 7-30
 available range of, 7-27
 avoiding fragmentation, 7-26
 bootstrapping RSTS/E into, 2-1
 display program status, 7-47
 dump of, 6-14
 initialization code bootstapped into, 2-1
 removing resident library, 7-30
Memory dump
 annotated version of, 6-14
MESSAGE command
 OPSER program, 5-9
Message receiver
 display program status, 7-48
 OPSER, 5-1
 QUEMAN declares itself, 5-16
MFD, 1-2
 storage of, 7-16
 on system disk, 1-2
Missed errors, 6-8
MODE values
 data caching, 7-17
MONEY, 4-5
 invoking, 4-5

MONEY, (cont.)

 maximum times for statistics, 4-8
 program options, 4-6t
 program output, 4-7t
 running during time sharing, 4-5
MONEY option
 RESET, 4-5
Monitor
 caching checks, 7-20f
 direct requests of, 7-24
 patches to overlay code, 10-3
 patching with ONLPAT, 10-3
 update cache, 7-17
Monitor directives
 data caching, 7-17
Monitor overlay code
 storage of, 7-16
MOUNT command
 CCL, 7-36
 INIT program, 3-2
 UTILITY program, 7-10
 UTILITY switches, 7-11, 7-12
Mount, logical
 disk cartridge, 7-8
 disk pack, 7-8
 with UTILITY program, 7-9
MR11-DB
 bootstrap procedure, A-4
MTXER
 DEVTST program, 11-14
 dialogue, 11-15
 errors, 11-15

N

NAME command
 UTILITY program, 7-29
NAME option
 SPOOL program, 5-25
NEXT command
 error in, 5-19
 QUEMAN program, 5-19
NO LOGINS command
 UTILITY program, 7-13
NO LOGINS message, 7-13
NORMAL option
 changing in SPOOL, 5-25
 SPOOL program, 5-25
Number conversion, D-1

O

Octal Debugging Tool. *See* ODT
ODT, 6-15
 address space access, 6-19

ODT, (cont.)

- block address, 6-25
- cautions on use, 6-15
- change address space, 6-15
- changing a location, 6-19
- characters and symbols, 6-16 to 6-18
- delimiterless character mode, 6-15
- determine content of word location, 6-20
- display large file content, 6-25
- error procedures, 6-27
- examine address space, 6-15
- file prompt responses, 6-18
- interpretive address quantities, 6-26
- octal values printed by, 6-16
- open absolute location, 6-21
- open location as a byte, 6-19
- open location as a word, 6-19
- open PC relative location, 6-21
- open preceding location, 6-20
- open relative branch offset location, 6-22
- printing ASCII format, 6-23
- printing location contents, 6-23
- printing Radix-50 format, 6-23
- reading pack identification, 6-27
- relocation registers, 6-24
- return to interrupted sequence, 6-22
- running, 6-18
- setting breakpoints with, 7-26
- specify relative address, 6-24
- submode, 6-15
- terminating, 6-18

ONLCLN, 7-11, 7-56, 9-9

- cleaning operation on disk, 7-56
- dialogue, 7-57
- help message, 7-57
- invoking, 7-57
- system errors, 7-56
- use of, 7-56

ONLPAT, 10-1

- in command file mode, 10-1, 10-2
- command file mode procedures, 10-2
- creating command files, 10-3
- CTRL/C termination, 10-2
- in keyboard mode, 10-1
- keyboard mode procedures, 10-1, 10-2
- logfile, 10-3
- messages, 10-2
- patching a running monitor, 10-3
- processing, 10-3

Operator communication program, 5-43

Operator services, 3-9, 5-1

- BACKUP program, 5-1
- BATCH program, 5-1
- OPSER program, 5-1, 5-5

Operator services, (cont.)

- overview of, 5-1
- QUEMAN program, 5-1
- SPOOL program, 5-1
- terminate without shutting down, 5-46
- terminating, 5-5, 5-46

Operator Services Console, 5-2

- INTERRUPT responses, 5-12
- output formats, 5-10
- send text to, 5-43

Operator table

- OPSER, 5-6

OPSER

- action request contents, 5-11t
- action requests, 5-10
- cause INIT to start, 5-13
- checks for on line jobs, 5-5
- command from PLEASE, 5-45
- command from SHUTUP, 5-46
- commands to, 5-2
- communicate with SHUTUP, 3-14
- compiled version of, 5-6
- controlled program, 5-1
- controlling BACKUP, 5-46
- initial operating conditions for, 5-13
- on line job list, 5-11, 5-12t
- log file, 5-2, 5-7
- manual shutdown procedure, 5-46
- message contents, 5-11t
- message formats, 5-9
- message levels, 5-9
- message receiver identification, 5-1
- modules, 5-5
- operator commands, 5-8
- operator communicate with, 5-44
- operator table, 5-6
- processing work files, 5-7
- restrictions of, 5-1
- running with SHUTUP, 3-13
- send command to QUEMAN, 5-18
- shutdown in SHUTUP, 3-17
- shutdown levels, 5-46
- shutdown procedures, 3-17
- shutting down in immediate mode, 3-18
- shutting down in logical end mode, 3-18
- in Spooling Package Library, 5-6
- start after system crash, 5-7, 5-14
- start up actions, 5-13
- start up procedure, 5-2, 5-12
- starting, 5-6
- during system shutdown, 5-5
- table of BACKUP commands, 5-47
- table of on line jobs, 5-1
- table of PLEASE commands, 5-45

OPSER, (cont.)
 terminating with SHUTUP, 5-5
 valid operators, 5-11, 5-45
 work file, 5-2, 5-6
 OPSER command, 5-8, 5-44
 abbreviation of, 5-8
 ANSWER, 5-10
 END, 5-46
 INTERRUPT, 5-10, 5-12, 5-17
 LIST, 5-11
 LIST JOBS, 5-11, 5-45
 LIST OPERATORS, 5-11
 LOGFILE, 5-9
 MESSAGE, 5-9
 through PLEASE, 5-44
 RETYPE, 5-10
 OPSER MUST BE COMPILED message, 5-6
 OPSER.LOG file, 5-7
 OSC. *See Operator Services Console*
 Output buffer
 text placed in, 7-13
 Overlay file
 adding, 7-35
 removing, 7-35

P

Pack identification
 extracting with SAVE/RESTORE, 9-24
 SAVE/RESTORE specification, 9-9
 Pack label
 DSKINT to determine, 6-27
 file structured disk, 6-27
 ODT to determine, 6-27
 PAGE EJECT option
 SPOOL program, 5-26
 Paper tape punch
 test operation of, 11-15
 Paper tape reader
 test operation of, 11-16
 Password
 changing for account, 7-12
 Patch file
 creating a, 10-7
 default extensions, 10-12
 naming convention, 10-6
 statements, 10-12
 PATCH statement
 PBUILD program, 10-11
 PATCH switch
 CPATCH program, 10-7
 Patches, 10-1
 to ASCII source files, 10-1
 to BASIC-PLUS library programs, 10-5

Patches, (cont.)
 to BASIC-PLUS source files, 10-5
 install to binary code, 10-1
 install to programs, 10-1
 install with PBUILD, 10-13
 mandatory, 10-1
 modifying command file, 10-4
 to monitor overlay code, 10-3
 published in Software Dispatch, 10-1
 release notes for, 10-6
 terminate a, 10-2
 verify on CPATCH, 10-7
 Patching facility
 automated, 10-1
 Patching kits, 10-1
 command files for, 10-1
 PBUILD, 10-10
 building command file, 10-10
 chaining to BUILD program, 10-16
 command file statements, 10-10
 command lines to patch a file, 10-11
 comments in command file, 10-10
 error messages, 10-19, 10-20t
 errors occurring in, 10-19
 to install patches, 10-13
 instructions for use, 10-10
 no compile operation, 10-12
 patching procedure, 10-14, 10-15
 running, 10-13
 sample command file, 10-13
 PBUILD option
 DETACH, 10-14
 PBUILD statement
 END, 10-12
 FORCE, 10-11
 PATCH, 10-11
 PDP-11 word
 decimal values, D-1
 octal values, D-1
 PEEK function
 test with CPEXER, 11-4
 use of privileged, 1-5
 Physical names, 7-38
 PHYSICAL option
 SPOOL program, 5-25
 Placed files
 BACKUP of, 8-24
 PLEASE, 5-2, 5-43
 as a CCL command, 5-45
 commands to OPSER, 5-2, 5-45
 identification line, 5-44
 message broadcast, 5-44
 OPSER commands through, 5-44
 restrict the use of, 5-43

- PLEASE, (cont.)
 - running, 5-44
 - terminating, 5-44
- PLEASE command
 - shutdown level, 5-46
- PPEXER
 - DEVTST program, 11-15
 - dialogue, 11-16
 - errors, 11-16
- PREXER
 - DEVTST program, 11-16
 - dialogue, 11-16
 - errors, 11-16
- Priority
 - basis of, 7-14
 - decrease for job, 7-15
 - LOGIN, 7-15
 - raise for SPOOL, 5-33
- Priority byte, 7-14
 - format of, 7-14
 - system definable portion, 7-15
- PRIORITY command
 - UTILITY program, 7-14
- Private disk, 1-2
- Privilege, 1-5
 - for a user job, 1-5
 - guidelines for, 1-8
 - permanent, 1-6, 1-7
 - regaining temporary, 1-8
 - temporary, 1-7
- Privilege, disk file
 - deletion of, 1-7
- Privileged account
 - assignment of, 1-8
 - job running under, 1-6
 - preventing unauthorized access to, 1-8
 - programming error, 2-5
- Privileged bit
 - in protection code, 1-6
 - setting the, 1-6
- Privileged file
 - compiled file as, 1-7
- Privileged program, 1-6
 - deletion of, 1-7
 - job running a, 1-6
- Privileged users
 - designate others as, 1-8
- Program Counter
 - ODT to open relative location, 6-21
- PROGRAM LOST SORRY error message, 6-11
 - types of, 6-12
- Program, privileged, 1-6
 - deletion of, 1-7
 - job running a, 1-7

- Program, system
 - privileged aspects of, 1-5
- Programming errors, 2-5
- Programs
 - designate as privileged, 1-5
 - install patches to, 10-1
 - privileged aspects of system, 1-5
- Protection bit
 - in compiled file, 1-6
- Protection code
 - privileged bit in, 1-6
- Pseudo keyboard, 5-4
 - BATCH, 5-39
 - using a, 5-4
- Public disk structure, 1-1
- Public structure
 - BACKUP multiple disks in, 8-3
 - reorder disks in the, 7-60

Q

- QUE, 5-2
 - process privileged file, 1-7
- QUEMAN, 5-2, 5-14
 - commands to start, 5-20
 - consistency checks, 5-16, 5-20
 - CTRL/C to, 5-17
 - detaching message, 5-17
 - entry in AFTER queue, 5-21
 - error in NEXT command, 5-19
 - initial conditions for spooling queues, 5-16
 - integrity checks, 5-16
 - interrupt commands, 5-17, 5-18t
 - message receiver declaration, 5-16
 - modules, 5-14
 - options, 5-16
 - requirements for running, 5-15
 - run detached, 5-16, 5-20
 - running after SHUTUP, 5-20
 - running after system crash, 5-20
 - in Spooling Package Library, 5-14
 - start up commands, 5-16
 - start up control file, 5-20
 - start up options, 5-17
 - start up procedure, 5-20
 - starting, 5-14
 - status printout, 5-19
 - table of on line spooling programs, 5-2
- QUEMAN command
 - DET, 5-17
 - INI, 5-16
 - NEXT, 5-19
 - received through OPSER, 5-18
 - STATUS, 5-19

QUEMAN MUST BE COMPILED message,
5-15
Queue management, 5-14
QUEUE.SYS file, 5-2
clear all entries from, 5-16
consistency checking on, 5-21
flag value, 5-16
initial conditions, 5-16
number of queue requests, 5-20
Quota
changing for account, 7-12
QUOTA command
UTILITY program, 7-12

R

Radix-50 format
printing with ODT, 6-23
Random data caching, 7-17
example of, 7-18
REACT, 4-1
creating user accounts, 4-1
deleting user accounts, 4-1
function specifications, 4-1
functions of, 4-1, 4-2
invoking, 4-1
REACT function
DELETE, 4-3
DELETE dialogue, 4-4
ENTER, 4-2
ENTER dialogue, 4-2
responses to DELETE, 4-4
responses to ENTER, 4-3
STANDARD, 4-4
Read request
job execution, 7-16
Receiver identification, 5-25
Registers, relocation
ODT, 6-24
Remote line
characteristics, 7-49
controlling, 7-16
disconnect, 7-16
ring characteristics, 7-49
setting characteristics, 7-49, 7-52
REMOVE ERROR command
UTILITY program, 7-35
REMOVE LOGICAL command
UTILITY program, 7-39
REMOVE OVERLAY command
UTILITY program, 7-35
REMOVE SWAPFILE command
UTILITY program, 7-35
REORDR, 7-58

REORDR, (cont.)
account designators, 7-60
dialogue, 7-59
directory restructuring functions, 7-58
error messages, 7-61
fatal errors, 7-62
indentification lines, 7-59
from indirect command file, 7-60
invoking, 7-59
precautions on use, 7-60
processing, 7-61
sample dialogue, 7-62
sample messages, 7-62
sort methods, 7-59
specification questions, 7-59
REQUE command
SPOOL program, 5-35, 5-36
RESET command
GRIPE program, 7-63
RESET option
MONEY program, 4-5
Resident library, 7-29
add a, 7-29
control, 7-29
display program status, 7-48
loading a, 7-31
one user access to, 7-30
permanently resident, 7-30
read only, 7-30
read/write, 7-30
removal under SHUTUP, 3-19
removed from memory, 7-30
RMS, 7-23
unloading a, 7-31
XBUF allocation, 7-23
RESTART command
SPOOL program, 5-35
RESTORE, 9-16
dialogue, 9-19
dialogue questions, 9-17t
example of, 9-18, 9-19
initiate the, 9-19
mounting volumes, 9-20
relocation of blocks, 9-16
restoring a RSTS/E disk, 9-16
run statistics, 9-31
run statistics report, 9-20
SAVE/RESTORE function, 9-16
summary report, 9-20
verify transferred data, 9-20
Restore
bad block error, 8-24
dialogue summary, 8-11, 8-12
listing file example, 8-38 to 8-40

- Restore mode, 8-3
 - entering accounts, 8-3
 - example of, 8-35, 8-36
- RESUME command
 - UTILITY program, 7-14
- Retrieval pointers
 - information for data blocks, 7-58
- RETYPE command
 - OPSER, 5-10
- Ring characteristics, 7-49
- RING command
 - TTYSET program, 7-52
- RK07
 - bootstrap routines, A-13
- RL01
 - bootstrap routines, A-13
- RL02
 - bootstrap routines, A-13
- RMS
 - cache and bucket cluster size, 7-22
 - caching files, 7-23
 - indexed file bucket sizes, 7-23
 - resident library, 7-23
- ROM
 - bootstrap configuration for M9312, A-9
 - configuration, A-8
 - CPU, A-7
 - locations, A-10t
 - peripheral boot, A-7
- RSTS/E
 - bootstrapping after system halt, 2-2
 - bootstrapping procedures, 2-1
 - device designators, 11-3
 - initialization code, 2-1
 - initialization options, 2-1
 - management of, 1-4
 - run-time systems, 7-24
 - shutdown procedures, 3-12
 - starting up, 2-1
 - structures of, 1-1
- RSTS/E monitor
 - error detection, 6-1
- RSTS/E system
 - configuring a, 1-4
 - halting the, 2-4
 - manage disks on, 7-8
- RT11 Run-Time System, B-13
 - fatal errors, B-14, B-15
 - recoverable errors, B-14
- RTS. *See Run-time system*
- RTS.CMD file, 3-10
- Run burst, 7-15
 - decrease the, 5-33
 - efficient system operation, 7-15
- Run burst, (cont.)
 - raise on SPOOL, 5-33
 - unit, 7-15
- RUN command
 - no extension specified, 7-28
- Run-time system
 - ADD command errors, 7-27
 - adding auxiliary, 7-25
 - associate a file with, 7-29
 - auxiliary, 7-24
 - BASIC-PLUS, 1-3, 7-24
 - change default extension, 7-28
 - change name of, 7-29
 - command file to add auxiliary, 3-10
 - commands, 7-25
 - control, 7-24
 - creation of auxiliary, 7-25
 - creation of default, 7-25
 - description block, 7-25
 - display program status, 7-48
 - establishing auxiliary, 3-10
 - install with BUILD, 10-16
 - job default, 7-24
 - language interface, 7-24
 - loading a, 7-26, 7-28
 - permanently resident, 7-28
 - placement of, 7-27, 7-28
 - removal under SHUTUP, 3-19
 - remove from system, 7-28
 - removing auxiliary, 7-25
 - RTS extension, 7-24
 - RT11, B-13
 - RT11 error messages, B-13
 - selection of default, 7-25
 - specifying load address, 7-26
 - storage of, 7-24
 - system default, 7-24
 - temporarily resident, 7-28
 - unloading a, 7-28
 - UTILITY commands, 7-25

S

- SAT. *See Storage allocation table*
- SATT.SYS file
 - create a, 7-53
- SAV/RES FUNCTION
 - answering prompt, 9-4
 - prompt, 9-3
 - responses to, 9-5
- SAVE
 - abort the operation, 9-15
 - default expiration date, 9-15
 - dialogue questions, 9-11, 9-12t
 - example of, 9-13, 9-14

SAVE, (cont.)

- function of SAVE/RESTORE, 9-14
- procedures, 9-13
- run statistics, 9-15, 9-31
- summary report, 9-11, 9-15
- summary report example, 9-31
- verify pass, 9-15

Save disk devices

- requirements of, 9-7

Save format, 9-2

- Save Image Library, 7-29, 10-3
- analyzing, 6-14

SAVE option

- BACKUP program, 8-5

Save set, 9-2, 9-7

- default name, 9-2, 9-11
- dismount message for, 9-28
- extract label information from, 9-24
- identify a, 9-2
- magnetic tape, 9-27
- magnetic tape density, 9-29
- processing multi-volume, 9-28
- recreate disk from a, 9-16
- restore a, 9-11
- set of directory blocks, 9-11
- volume, 9-15

Save set name, 9-2

- in device specification, 9-19
- SAVE/RESTORE specification, 9-9

Save volume, 9-2, 9-7

- booting, 9-27t
- initializing the, 9-15

SAVE/RESTORE, 9-1

- aborting, 9-29
- for backup processing, 9-3
- bad block on output device, 9-33
- bad block processing, 9-33
- changes to a file, 9-33
- checking input volume, 9-8
- checking output volume, 9-9
- comparison with BACKUP program, 9-1
- create bootable media, 9-26
- CTRL/C in dialogue, 9-7, 9-29
- CTRL/Z in dialogue, 9-7
- definition of terms, 9-2
- device specification options, 9-8
- dialogue, 9-3, 9-6
- dialogue defaults, 9-5, 9-6
- dismount message, 9-28
- dismount report, 9-15
- dismounting volumes, 9-28
- encounter bad blocks, 9-16
- error during dialogue, 9-32

SAVE/RESTORE, (cont.)

- error during transfer phase, 9-32
 - error during verification phase, 9-32
 - error handling, 9-32
 - error messages, 9-33 to 9-37
 - errors, 9-4
 - fatal transfer errors, 9-37
 - full function command line, 9-26
 - function response, 9-4
 - functions, 9-1, 9-4t
 - IDENTIFY function, 9-24
 - image copy a disk, 9-10
 - IMAGE function, 9-20
 - invoking, 9-3
 - like disks, 9-2
 - long form dialogue, 9-6
 - mounting volumes, 9-28
 - non-fatal transfer errors, 9-38
 - number of free clusters, 9-11
 - operator interface during processing, 9-28
 - options, 9-5, 9-6t
 - output cluster, 9-11
 - output volume expiration date, 9-9
 - pack id specification, 9-9
 - PROCEED prompt, 9-26
 - prompt, 9-3
 - reaccessing devices, 9-29
 - restore a disk, 9-10
 - RESTORE function, 9-16
 - run statistics, 9-31
 - running off line, 9-3
 - running on line, 9-3
 - save format, 9-2, 9-11
 - SAVE function of, 9-14
 - save set, 9-2
 - save set name, 9-2, 9-9
 - save volume, 9-2
 - short form dialogue, 9-6
 - single command line, 9-5
 - summary information, 9-5
 - summary report, 9-29, 9-30f
 - summary report format, 9-30
 - summary report totals, 9-32
 - tape medium, 9-7
 - transfer errors, 9-37
 - using, 9-1
 - valid directory, 9-15
 - volume label checking, 9-26
- ## SAVE/RESTORE switch
- ERROR, 9-5
 - NOERROR, 9-5, 9-33, 9-37
 - NOSTATS, 9-5, 9-15
 - STATS, 9-5
 - VERIFY, 9-15

- SCRATCH switch
 - BACKUP program, 8-8
- SEND command
 - INIT program, 3-3
 - UTILITY program, 7-13
- Sequential data caching, 7-17, 7-18
 - example of, 7-18
- SET LOGINS command
 - UTILITY program, 7-13
- Shutdown procedure
 - ERRCPY, 3-18
 - manual for OPSER, 5-46
 - OPSER, 3-17
 - RSTS/E system, 3-12
 - specify waiting period, 3-14
- SHUTUP, 2-4, 3-13, 5-46
 - attached jobs, 3-16
 - command to OPSER, 5-46
 - communicate with OPSER, 3-14
 - detached jobs, 3-16
 - disk dismount phase, 3-19
 - ERRCPY shutdown phase, 3-18
 - error message to ERRCPY, 6-3
 - final job killing phase, 3-19
 - final shutdown phase, 3-19
 - initial job killing phase, 3-16
 - LOGIN limit, 3-15
 - notes on operation of, 3-23
 - operational phases, 3-13
 - OPSER shutdown phase, 3-17
 - with OPSER, 3-13
 - without OPSER, 3-13
 - removing swapping files, 7-35
 - resident library removal phase, 3-19
 - run from console terminal, 3-13
 - run-time system removal phase, 3-19
 - running QUEMAN after, 5-20
 - sample operation runs of, 3-20 to 3-22
 - set up dialogue phase, 3-14
 - specify shutdown wait period, 3-14
 - swap file removal phase, 3-19
 - terminating OPSER, 5-5
 - unexpected errors, 3-19
 - warning message phase, 3-15
- SIL. *See Save Image Library*
- Small buffers, 7-13
 - not used for caching, 7-20
 - system efficiency, 7-13
- Software Dispatch
 - component numbers, 10-6
 - patches published in, 10-1
 - subcomponent numbers, 10-6
- Software errors
 - logging of, 6-1
- Software malfunction, 2-6
- Software Performance Report, B-1
- SPEED command
 - TTYSET program, 7-49
- SPOOL, 5-4, 5-22
 - changing NORMAL form, 5-25
 - compiled version of, 5-22
 - condition the operation of, 5-23
 - default form, 5-25
 - default keyboard values, 5-28
 - default line printer values, 5-28
 - error handling, 5-35
 - error messages, 5-37
 - error text in user output, 5-37t
 - file header, 5-36
 - forms control characters, 5-26
 - initial conditions, 5-30
 - interrupt commands, 5-30
 - invoking, 5-22
 - job header, 5-36
 - line printer errors, 5-35
 - line printer spooling program, C-1
 - logical device name, 5-23
 - modules, 5-22
 - queued requests to, 5-4
 - raise priority, 5-33
 - raise run burst, 5-33
 - restart queued request, 5-35
 - start up error processing, 5-29
 - start up examples, 5-31
 - start up options, 5-23, 5-24t
 - start up switches, 5-23
 - syntax error messages, 5-29
 - terminating queued request, 5-36
- SPOOL command
 - ABORT, 5-27, 5-36
 - FORM, 5-37, 5-38
 - REQUE, 5-35, 5-36
 - RESTART, 5-35
- SPOOL MUST BE COMPILED message, 5-22
- SPOOL option
 - ALIGN, 5-38
 - ASSIGN, 5-25
 - DFLENGTH, 5-26
 - FORM, 5-25
 - NAME, 5-25
 - NORMAL, 5-25
 - PAGE EJECT, 5-26
 - PHYSICAL, 5-25
- SPOOL.CMD file, 3-9
- Spooling, 3-9, 5-1
 - line printer, 5-28
 - terminating, 5-46

- Spooling jobs
 - determining number of, 5-19
 - limit of, 5-21
 - processing request, 5-19
- Spooling Package Library, 5-4, 5-5, 5-22
 - account of, 5-6
 - BATCH in, 5-39
 - OPSER in, 5-6
 - QUEMAN in, 5-14
- Spooling programs, 5-1
 - commands to, 5-12
 - controlling, 5-12
 - line printer, 5-22
 - messages to, 5-12
 - monitoring, 5-12
 - sleep state, 5-4
 - SPOOL, 5-22
 - table of in QUEMAN, 5-2
 - user requests of, 5-2
- Spooling queues
 - initial conditions for, 5-16
- SPR. *See Software Performance Report*
- STANDARD function
 - REACT program, 4-4
- START option
 - initialization code, 3-1
- Start up
 - BATCH options, 5-40
 - BATCH procedures, 5-42
 - controlling system, 3-1
 - error log for, 6-11
 - OPSER procedures, 5-2, 5-12
 - QUEMAN procedures, 5-20
 - set JOB MAX, 7-13
 - SPOOL examples, 5-31
 - SPOOL options, 5-23
 - system conditions at, 3-1
- Start up control file
 - operations of, 3-3
 - order of operations, 3-6
- START.CTL file, 3-11
 - operations of, 3-3
 - replace the, 3-3
 - TTYSET commands in, 7-49
- Statement
 - CHAIN, 1-4
- STATUS command
 - BACKUP program, 8-29
 - QUEMAN program, 5-19
- Storage allocation table
 - build new, 7-57
- SUSPEND command
 - UTILITY program, 7-14
- Swap file, 1-4, 7-32
 - adding, 7-32
 - creation of on line, 7-32
 - removal under SHUTUP, 3-19, 7-35
 - remove a, 7-32, 7-34
- Swapping, 1-4
- Swapping slot, 7-32
 - activate the added, 7-34
 - remove a, 7-34
- Swapping space
 - restricting logins, 7-34
- Swapping storage
 - fixed head disk for, 1-3
- Swapping violation, 7-27
- Switch register codes, A-12t
- SYS functions
 - programming with, 2-5
 - use of privileged, 1-5
- SYSTAT, 7-40
 - disk status report, 7-10
 - FREE block count, 7-41
 - free buffer status report, 7-13
 - HUNG TTY count, 7-41
 - use of, 7-40, 7-41
- System
 - access to, 1-1
 - add disk to, 7-9
 - changing date, 7-16
 - changing time, 7-16
 - command level, 1-3
 - default run-time system, 7-24
 - enable data caching on, 7-19
 - enable directory caching on, 7-19
 - exercise hardware on, 11-1
 - frequency of backup, 8-2
 - initialized in restart mode, 2-7
 - operational concepts, 1-3
 - operational control of, 7-12
 - remove disk from, 7-9
 - remove run-time system from, 7-28
 - run burst for efficient operation, 7-15
 - set JOB MAX, 7-13
 - stimulate hardware errors on, 11-1
 - unlimited access on, 1-5
- System commands, 1-3
- System crash, 2-2, 2-5
 - analyzing, 6-13
 - causes of, 2-5
 - command file for recovery, 3-8
 - control file, 3-12
 - documenting, 6-13
 - handling of, 2-6
 - OPSER restarted, 5-7
 - OPSER starts after, 5-14

- System crash, (cont.)
 - randomly occurring, 2-5
 - running ANALYS after, 3-12
 - running QUEMAN after, 5-20
 - table of error codes, 6-15
- System disk, 1-1
 - contents of, 1-1
 - create bootable, 9-2
 - MFD on, 1-2
- System efficiency
 - small buffers, 7-13
- System Error Log, 11-1
- System error package, C-2
 - programs in, 6-1
- System file
 - backing up, 8-26
 - control, 7-31
 - CRASH.CTL, 2-7
 - disk copies, 8-1
 - magnetic tape copies, 8-1
 - UTILITY commands for, 7-32
- System generation
 - error messages, B-1
- SYSTEM GOING DOWN message, 3-16
- System halt
 - bootstrapping RSTS/E after, 2-2
- System integrity
 - maintaining, 1-8
- System library
 - build procedures, C-1
- System logical names, 7-38
- System management, 1-4
 - and BACKUP, 8-1
- System program
 - ANALYS, 6-1, 6-13
 - auxiliary files, C-1
 - BACDSK, 8-18
 - BACKUP, 5-1, 8-1
 - BATCH, 5-1, 5-4, 5-5, 5-38
 - BUILD, 10-16
 - CPATCH, 10-1
 - DEVTST, 11-1
 - DSKINT, 7-53
 - ERRBLD, 6-5
 - ERRCPY, 6-1
 - ERRDET, 6-7
 - ERRINT, 6-1
 - GRIPE, 7-62
 - INIT, 2-7
 - MAKSIL, 7-29
 - MONEY, 4-5
 - ODT, 6-15
 - ONLCLN, 7-11, 7-56, 9-9
 - ONLPAT, 10-1

- System program, (cont.)
 - OPSER, 5-1
 - PBUILD, 10-10
 - PLEASE, 5-2, 5-43
 - QUE, 5-2
 - QUEMAN, 5-1, 5-2, 5-14
 - REACT, 4-1
 - REORDR, 7-58
 - SAVE/RESTORE, 9-1
 - SHUTUP, 2-4, 3-13
 - SPOOL, 5-1, 5-4, 5-22
 - SYSTAT, 7-40
 - TALK, 7-64
 - TTYSET, 3-3, 7-49
 - UTILITY, 7-1
 - VT50PY, 7-40
 - VT5DPY, 7-40
- System programs
 - privileged aspects of, 1-5
- System shutdown
 - OPSER during, 5-5
 - performing, 3-12
- System start up
 - command file for, 3-8
 - conditions at, 3-1
 - control file, 3-11
 - controlling, 3-1
 - crash recovery, 3-12
 - sample of control file, 3-11
 - TTYSET program at, 3-9
- System status
 - dynamic display of, 7-42
 - monitoring, 7-40
- System structures
 - consistency of, B-1
- System, RSTS/E
 - automatic restart, 2-2
 - conditions for shutting down, 3-13
 - halting the, 2-4

T

- TALK, 7-64
 - broadcasting a message, 7-64
 - instructions, 7-64
 - invoking, 7-64
 - send a message, 7-64
 - terminating, 7-65
- Terminal
 - automatic characteristic settings, 7-52
 - BACKUP attached to, 5-46
 - broadcast messages to, 7-13
 - change characteristics of, 7-52
 - characteristics, 7-49

Terminal, (cont.)

- command file to set characteristics, 3-9
- commands forced to, 3-7
- communicating with, 7-64
- CTRL/C typed at, 1-3
- DFLENGTH option effect, 5-28
- hardware top of form, 5-29
- line speed characteristics file, 7-49
- remove a single interface, 7-16
- restrict line speeds of, 7-50
- returned to edit mode, 1-4
- sending a message to, 7-64
- setting characteristics, 3-9, 7-52
- at system command level, 1-3
- text placed in output buffer, 7-13
- top of form capability, 5-26

Terminal, console

- jobs logged in at, 7-13
- running SHUTUP from, 3-13
- used to start system jobs, 3-7

TIME command

- UTILITY program, 7-16

Time sharing

- starting, 2-3

TTY.CMD file, 3-9

TTYSET, 7-49

- commands in START.CTL, 7-49
- header line, 7-51
- prompt, 7-51
- run by INIT, 3-3
- setting ring characteristics, 7-49
- at system start up, 3-9

TTYSET command, 7-49

- EXIT, 7-52
- KB, 7-49, 7-51
- RING, 7-52
- SPEED, 7-49

TTYSET.SPD file, 7-49

- creating a, 7-50
- format of lines in, 7-50

U

UFD, 1-2, 7-58

- BACKUP file selection, 8-2
- cluster size, 4-7
- contents of, 1-2
- forward by access date, 7-58
- forward by creation date, 7-58
- mark file for caching, 7-18, 7-21
- override marked for caching, 7-22
- remove all files from, 4-3
- reordering methods, 7-58
- reverse by access date, 7-58

UFD, (cont.)

- reverse by creation date, 7-58
- storage of, 7-16

UNLOAD command

- UTILITY program, 7-28

- UTILITY program options, 7-29

UNLOAD LIBRARY command

- UTILITY program, 7-31

UNLOCK command

- UTILITY program, 7-11

User comments

- processing, 7-62

User data

- access to, 1-1

User File Directory. *See* UFD

User job

- area, 1-3
- privilege for, 1-5
- storage of, 1-1

Users, privileged

- designating others, 1-8

UTILITY, 7-1

- CCL command, 7-8
- change a priority, 7-14
- changing account quota, 7-12
- changing password, 7-12
- commands of, 7-1
- control of system files, 7-32
- control system operation, 7-12
- defining CCL commands, 7-36
- disk management, 7-8
- disk mounting options, 7-11
- header line, 7-8
- LIBRARY keyword, 7-30
- logically mount a disk, 7-9
- prompt, 7-8
- removing all files from accounts, 7-12
- resident library, 7-30
- run-time system commands, 7-25
- running, 7-8
- send messages to terminals, 7-13
- setting job priority, 7-14
- setting job run burst, 7-14
- setting job size, 7-14
- specify run burst, 7-15
- table of commands, 7-1 to 7-7
- terminating, 7-8

UTILITY command

- ADD, 3-10, 7-25
- ADD ERROR, 7-35
- ADD LIBRARY, 7-29, 7-30
- ADD LIBRARY/1USER, 7-30
- ADD LIBRARY/NOLOGERR, 7-30
- ADD LIBRARY/RW, 7-30

UTILITY command, (cont.)

ADD LIBRARY/STAY, 7-30
ADD LOGICAL, 7-12, 7-39
ADD OVERLAY, 7-35
ADD SWAPFILE, 7-13, 7-32
ADD/1USER, 7-26
ADD/ADDR, 7-26
ADD/EMT, 7-26
ADD/EXT, 7-28
ADD/KBM, 7-26
ADD/LOGERR, 7-26
ADD/MAX, 7-28
ADD/MIN, 7-28
ADD/POSITION, 7-27
ADD/RW, 7-26
ADD/STAY, 7-28
CCL, 7-36, 7-37
CHANGE, 7-12
CHANGE LOGICAL, 7-12, 7-39
CLEAN, 7-11
DATE, 7-16
DETACH, 7-16
DETACH/NOCLOSE, 7-16
DISABLE CACHE, 7-21
DISABLE KB, 7-16
DISMOUNT, 7-10
ENABLE CACHE, 7-17, 7-19
ENABLE CACHE/ALL, 7-19
ENABLE CACHE/BUFF, 7-20
ENABLE CACHE/CL, 7-20
ENABLE CACHE/DATA, 7-21
ENABLE CACHE/DIR, 7-21
ENABLE CACHE/FILE, 7-20
ENABLE CACHE/LIMIT, 7-21
ENABLE CACHE/NOFILE, 7-20
EXIT, 7-8
FLAG, 7-21
FLAG/CACHE, 7-21
FLAG/NOCACHE, 7-21
FLAG/RAN, 7-22
FLAG/SEQ, 7-21
FORCE, 7-14
HANGUP, 7-16
HELP, 7-8
KILL, 7-14
LIST CACHE, 7-19
LIST CCL, 7-38
LIST LOGICAL, 7-40
LOAD, 7-28, 7-29
LOAD LIBRARY, 7-31
LOAD LIBRARY/ADDR, 7-31
LOAD LIBRARY/STAY, 7-31
LOAD/ADDR, 7-28
LOAD/STAY, 7-29

UTILITY command, (cont.)

LOGINS, 7-13, 7-34
MOUNT, 7-10
MOUNT/LOGICAL, 7-11
MOUNT/NOLOGICAL, 7-11
MOUNT/PRIVATE, 7-12
MOUNT/ROONLY, 7-12
NAME, 7-29
NO LOGINS, 7-13
PRIORITY, 7-14
QUOTA, 7-12
REMOVE, 7-28
REMOVE ERROR, 7-35
REMOVE LOGICAL, 7-39
REMOVE OVERLAY, 7-35
REMOVE SWAPFILE, 7-35
RESUME, 7-14
SEND, 7-13
SET LOGINS, 7-13
SUSPEND, 7-14
TIME, 7-16
UNLOAD, 7-28, 7-29
UNLOAD LIBRARY, 7-31
UNLOCK, 7-11
ZERO, 4-3, 7-12

V

Volumes

dismounting SAVE/RESTORE, 9-28
mounting SAVE/RESTORE, 9-28
VT50PY, 7-40, 7-42, 7-43. *See also display program*
display system status, 7-42
INTERVAL question, 7-42
running, 7-42
terminating, 7-42
VT5DPY, 7-40, 7-42, 7-43. *See also display program*
display system status, 7-42
running, 7-42
terminating, 7-42

W

Wildcard

BACKUP, 8-5
Window turning
reducing, 7-23
Work file
BACKUP, 8-2
bad block error, 8-24
enable caching on, 8-28

Work file, (cont.)

- estimate size of, 8-3
- fixed overhead for, 8-3
- on private disk, 8-3
- size of, 8-3

X

XBUF

- caching allocation, 7-17, 7-23
- resident library allocation, 7-23

XON/XOFF

- synchronization protocol, 5-34

Z

ZERO command

- UTILITY program, 7-12

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____

or
Country

--- Do Not Tear - Fold Here and Tape ---

digital



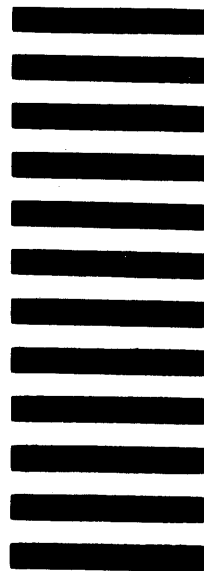
No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Commercial Engineering Publications MK1-2/H3
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK N.H. 03054



--- Do Not Tear - Fold Here and Tape ---

Cut Along Dotted Line