

**September 1979**

This manual describes the functions and usage of the RSTS/E Version 7.0 operating system for the non-privileged user.

**RSTS/E  
System User's Guide**

**Order No. AA-5133B-TC**

**Including: AD-5133B-T1**

**SUPERSESSION/UPDATE INFORMATION:** This manual contains information concerning 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**

First Printing: December 1976  
Revised: December 1977  
Revised: May 1979  
Updated: September 1979

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 © 1976, 1977, 1979 Digital Equipment Corporation

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's 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>xv</i>

## Part I      System Fundamentals and Resources

### Chapter 1 The System Fundamentals

1.1	Becoming a RSTS/E User: Project-Programmer Number and Password . . . . .	1-1
1.2	Logging In: The HELLO Command . . . . .	1-2
1.3	Logging Out: The BYE Command . . . . .	1-4
1.4	The Directory and Files . . . . .	1-5
1.4.1	Filenames and Extensions . . . . .	1-6
1.4.2	Protection Codes . . . . .	1-7
1.4.3	Account and File Statistics . . . . .	1-9

### Chapter 2 The System Resources

2.1	Introduction to the RSTS/E Devices . . . . .	2-1
2.2	The Public Disk Structure . . . . .	2-3
2.3	Private Disks . . . . .	2-4
2.4	Assignable Devices . . . . .	2-5
2.5	A List of RSTS/E Devices . . . . .	2-6
2.6	Device Names: Physical and Logical . . . . .	2-9
2.6.1	Physical Device Names . . . . .	2-9
2.6.2	Logical Device Names . . . . .	2-10

## Part II      Adapting System Resources

### Chapter 3 Functions of the Resource Commands

### Chapter 4 Controlling Devices and Accounts

4.1	Reserving a Device: The ASSIGN Command . . . . .	4-1
4.2	Releasing a Device: The DEASSIGN Command . . . . .	4-3
4.3	Transferring a Device: The REASSIGN Command . . . . .	4-3
4.4	Assigning and Using Logical Names: The ASSIGN and DEASSIGN Commands . . . . .	4-4
4.4.1	Associating Multiple Logical Names with One Device . . . . .	4-4
4.4.2	Associating a Valid Physical Name with a Device . . . . .	4-5
4.4.3	Reserving and Releasing a Logically Named Device . . . . .	4-5
4.4.4	Associating a Logical Name with a Device and Account . . . . .	4-6

	Page
4.5 System-Wide Logical Names . . . . .	4-7
4.6 Disk Access by Pack Identification Label or Logical Name . . . . .	4-8
4.6.1 Disk Access by Pack Identification Label . . . . .	4-8
4.6.1.1 Logically Mounting a Disk by System Command: MOUNT . . . . .	4-8
4.6.2 Disk Access by Logical Name: The ASSIGN Command . . . . .	4-9
4.7 Logical Assignment of a User Account . . . . .	4-9
4.8 Terminal Echo Settings . . . . .	4-10
4.8.1 Disabling the Terminal Echo: The TAPE Command . . . . .	4-10
4.8.2 Enabling the Terminal Echo: The KEY Command . . . . .	4-11
4.9 Input and Output Control Characters . . . . .	4-11
4.9.1 CTRL/C . . . . .	4-12
4.9.2 CTRL/O . . . . .	4-12
4.9.3 CTRL/R . . . . .	4-12
4.9.4 CTRL/S and CTRL/Q . . . . .	4-13
4.9.5 CTRL/Z . . . . .	4-13
4.9.6 RETURN Key . . . . .	4-13
4.9.7 ESCAPE or ALTMODE Key . . . . .	4-13
4.9.8 CTRL/T . . . . .	4-13

## **Chapter 5 Changing Defaults**

5.1 Changing the Default Protection Code: The ASSIGN <> Command . . . . .	5-1
5.2 Changing the Magnetic Tape Labeling Default . . . . .	5-1
5.3 Changing the Default Run-Time System for a Job . . . . .	5-2
5.3.1 SWITCH Program Error Messages . . . . .	5-4

## **Part III The BASIC-PLUS System Commands**

### **Chapter 6 Functions of the BASIC-PLUS System Commands**

6.1 Some Definitions for the New BASIC-PLUS User . . . . .	6-1
6.1.1 Source and Compiled Programs . . . . .	6-1
6.1.2 The Program Currently in Memory . . . . .	6-2
6.2 A Guide to the BASIC-PLUS System Commands . . . . .	6-2

### **Chapter 7 Creating and Running a BASIC-PLUS Program**

7.1 Writing the Program . . . . .	7-1
7.1.1 The NEW Command . . . . .	7-1
7.1.1.1 Creating a NONAME File . . . . .	7-2
7.1.2 Input of the New Program . . . . .	7-3

	Page
7.2 Saving the Program: The SAVE Command . . . . .	7-3
7.2.1 Saving the Current Program under a Different Name . . . . .	7-4
7.2.2 Using SAVE to Specify a Storage Device . . . . .	7-4
7.2.3 Using SAVE to Obtain Line Printer and Paper Tape Output . . . . .	7-4
7.3 Calling an Existing Program: The OLD Command. . . . .	7-5
7.4 Running and Compiling Programs: The RUN and COMPILE Commands . . . . .	7-6
7.4.1 Running a Program from the Public Structure: The RUN Command . . . . .	7-6
7.4.2 Running a Program from Private or Specific Devices: The RUN dev: Command . . . . .	7-7
7.4.3 The COMPILE Command . . . . .	7-8
7.4.3.1 The Purpose of COMPILE. . . . .	7-8
7.4.3.2 Using COMPILE . . . . .	7-9
7.4.3.3 Compiling a Program on a Specific Disk Pack . . . . .	7-10
7.5 Renaming the Current Program: The RENAME Command . . . . .	7-10
7.6 Replacing a Saved Program: The REPLACE Command . . . . .	7-11
7.7 Changing a Program's File Specification: The NAME AS Statement . . . . .	7-11
7.7.1 Using NAME AS to Rename a Program. . . . .	7-11
7.7.2 Using NAME AS to Change a Protection Code . . . . .	7-12
7.8 Finding a Program's Current and Maximum Length: The LENGTH Command . . . . .	7-12
7.9 Listing Device Directories: The CATALOG Command . . . . .	7-13
7.10 Using Scaled Arithmetic: The SCALE Command . . . . .	7-14

## **Chapter 8 Editing and Modifying a Program**

8.1 Editing and Modifying the Program. . . . .	8-1
8.1.1 Printing the Program: The LIST Command . . . . .	8-1
8.1.2 Deleting Lines: The DELETE Command . . . . .	8-3
8.1.2.1 Cautionary Notes about DELETE . . . . .	8-3
8.1.3 Simple Erasures: The RUBOUT Key and CTRL/U . . . . .	8-3
8.1.3.1 Erasing One Character at a Time: RUBOUT . . . . .	8-3
8.1.3.2 Erasing One Line at a Time: CTRL/U . . . . .	8-4
8.1.4 Removing a Program from a Storage Device: The UNSAVE Command and the KILL Statement . . . . .	8-5
8.1.4.1 The UNSAVE Command . . . . .	8-5
8.1.4.2 The KILL Statement . . . . .	8-6
8.1.5 Merging Programs: The APPEND Command . . . . .	8-6
8.1.6 Transferring Control between Programs: The CHAIN Statement . . . . .	8-7
8.2 Formatting the Program . . . . .	8-9
8.2.1 Changing Statement Format Rules: The EXTEND/NO EXTEND Commands . . . . .	8-9
8.2.1.1 Description of EXTEND Format. . . . .	8-9
8.2.1.2 Issuing the EXTEND/NO EXTEND Commands . . . . .	8-10

	Page
8.2.2 NO EXTEND Statement Formatting . . . . .	8-11
8.2.2.1 Multiple Statements on a Single Line . . . . .	8-11
8.2.2.2 A Single Statement on Multiple Lines . . . . .	8-11
8.2.2.3 Spaces and Tabs for Readability . . . . .	8-12

## **Chapter 9 Debugging a Program**

9.1 Executing Statements in Immediate Mode . . . . .	9-1
9.1.1 The Limitations of Immediate Mode . . . . .	9-2
9.2 Using Immediate Mode with STOP, CONT, CCONT, and GOTO . . . . .	9-2
9.2.1 The STOP Statement . . . . .	9-2
9.2.2 The CONT Command . . . . .	9-3
9.2.3 The CCONT Command . . . . .	9-3
9.2.4 The GOTO Statement . . . . .	9-3
9.3 Debugging with CTRL/C, PRINT LINE, and CTRL/O . . . . .	9-3
9.3.1 Halting and Checking Execution with CTRL/C and PRINT LINE . . . . .	9-3
9.3.2 Suppressing Output with CTRL/O . . . . .	9-4
9.3.3 Suppressing Output with CTRL/S and CTRL/Q . . . . .	9-4
9.4 An Example of Program Debugging . . . . .	9-4

## **Part IV System Library Programs**

### **Chapter 10 Introduction to Part IV**

### **Chapter 11 File Information and Standards**

11.1 device: The Device Designator . . . . .	11-1
11.1.1 Logical Device Names . . . . .	11-3
11.2 [proj,prog] The Project-Programmer or Account Number . . . . .	11-3
11.2.1 Special Account Characters. . . . .	11-4
11.3 filename.extension The Filename and Extension . . . . .	11-4
11.3.1 Wildcard Specifications . . . . .	11-6
11.3.1.1 The * Wildcard . . . . .	11-6
11.3.1.2 The ? Wildcard . . . . .	11-6
11.3.1.3 The * and ? Wildcards Combined . . . . .	11-6
11.4 <prot> Protection Code . . . . .	11-7
11.5 /option File Specification Option . . . . .	11-8
11.5.1 /FILESIZE Option . . . . .	11-8
11.5.2 /CLUSTERSIZE Option . . . . .	11-9
11.5.3 /POSITION Option . . . . .	11-10
11.5.4 /MODE and /RONLY Options . . . . .	11-11
11.6 File Attributes. . . . .	11-11

## **Chapter 12 Program Information and Characteristics**

12.1	The Concise Command Language (CCL) . . . . .	12-1
12.1.1	Cautionary Notes on Typing CCL Commands . . . . .	12-1
12.1.1.1	Embedded Spaces in CCL Commands . . . . .	12-1
12.1.1.2	Mistyping a BASIC-PLUS Command as a CCL Command . . . . .	12-1
12.2	Obtaining Help Files for System Programs . . . . .	12-2
12.3	Version Identification . . . . .	12-2
12.4	Indirect Command Files . . . . .	12-2
12.5	Obtaining Help: The HELP Program . . . . .	12-3

## **Chapter 13 Job Control Programs**

13.1	Entering the System: The LOGIN Program . . . . .	13-1
13.1.1	Running LOGIN from a Logged Out Terminal. . . . .	13-1
13.1.2	Running LOGIN at a Logged In Terminal. . . . .	13-4
13.1.3	Running Other Programs from a Logged Out Terminal. . . . .	13-7
13.2	Leaving the System: The LOGOUT Program . . . . .	13-7

## **Chapter 14 System Communication Programs**

14.1	Printing a System Status Report: The SYSTAT Program . . . . .	14-1
14.1.1	Contents of the Status Report . . . . .	14-4
14.1.2	SYSTAT as a CCL Command . . . . .	14-9.1
14.2	Obtaining a Disk Quota Report: The QUOLST Program . . . . .	14-10
14.3	Obtaining Account Data: The MONEY Program . . . . .	14-10
14.4	Sending a Message to the System Manager: The GRIPE Program . . . . .	14-11
14.5	Declaring a Terminal In Use: The INUSE Program . . . . .	14-12

## **Chapter 15 File Utility Programs: Listing, Editing and Reading Files**

15.1	Listing Directory of Files: The DIRECT Program . . . . .	15-1
15.1.1	DIRECTORY as a CCL Command . . . . .	15-6
15.2	Comparing Files: The FILCOM Program . . . . .	15-7
15.2.1	FILCOM Prompts . . . . .	15-8
15.2.2	FILCOM Single Command Line . . . . .	15-10
15.2.3	Wildcards . . . . .	15-13
15.2.4	FILCOM Examples . . . . .	15-14
15.2.5	FILCOM Error Messages . . . . .	15-17
15.3	Generating a Cross-Reference Listing - The BPCREF Program . . . . .	15-18
15.3.1	Running and Terminating BPCREF. . . . .	15-18
15.3.2	Output Listing Contents . . . . .	15-19
15.3.3	Error Messages . . . . .	15-21

## Chapter 16 Device Utility Programs: PIP, COPY, and FIT

16.1	Device Transfer: The PIP Program . . . . .	16-1
16.1.1	PIP Command Line Specifications . . . . .	16-2
16.1.2	Wildcard Specifications . . . . .	16-3
16.1.3	Indirect Command Files in PIP . . . . .	16-5
16.2	PIP Switches . . . . .	16-6
16.2.1	PIP Information Switch (/HE) . . . . .	16-9
16.2.2	File Transfer Operations . . . . .	16-10
16.2.2.1	Access Switch (/AC) . . . . .	16-12
16.2.2.2	Append and Extend Switches (/AP and /EX) . . . . .	16-13
16.2.2.3	ASCII Switch (/AS) . . . . .	16-13
16.2.2.4	Block Switch (/BL) . . . . .	16-13
16.2.2.5	Block Size Switch (/BSIZE:n) . . . . .	16-14
16.2.2.6	Clustersize Switch (/CL:n) . . . . .	16-15
16.2.2.7	Go or Ignore Switches (/GO or /IG) . . . . .	16-15
16.2.2.8	Mode Switch (/MO:n) . . . . .	16-15
16.2.2.9	New File and Retain Switches (/NE, /RET) . . . . .	16-15
16.2.2.10	No Attributes Switch (/NOA) . . . . .	16-16
16.2.2.11	Protect Switch (/PR) . . . . .	16-16
16.2.2.12	Run-Time System Name Switch (/RTS:name) . . . . .	16-16
16.2.2.13	Update Switch (/UP) . . . . .	16-16
16.2.2.14	Multi-Volume ANSI Magnetic Tape File Transfer . .	16-17
16.2.3	File Operations Involving Attributes . . . . .	16-19
16.2.3.1	RMS Switch (/RMS:options) . . . . .	16-20
16.2.4	Date Related Switches (/DLA, /CRE, /AF, /BE, /ON, /SIN, /TO, and /UN) . . . . .	16-21
16.2.5	File Operations Switches . . . . .	16-22
16.2.5.1	Halt Switch (/HA) . . . . .	16-23
16.2.5.2	Inspect Switch (/IN:option) . . . . .	16-23
16.2.5.3	Log, No Log, or Watch Switches (/LO, /NOL, or /WA) .	16-23
16.2.5.4	No Rewind Switches (/NO or /RW:NO) . . . . .	16-23
16.2.5.5	Version or Identification Switch (/VE or /ID) . . . . .	16-23
16.2.6	File Deletion Switches (/DE, /ER, /WO, and /WIPE) . . . . .	16-24
16.2.7	File Rename Switch (/RE:option) . . . . .	16-24
16.2.8	Directory Listing Switches (/BR, /F, /DI, /LI and /S) . . . . .	16-25
16.2.9	Zeroing Directories Switch (/ZE) . . . . .	16-26
16.2.10	Privileged Only Operations (/LOCK and /PRIOR) . . . . .	16-28
16.3	PIP Error Messages . . . . .	16-28
16.4	Copying Between Devices: The COPY Program . . . . .	16-28
16.5	File Transfer Between Devices: The FIT Program . . . . .	16-31
16.5.1	Transferring Files with FIT . . . . .	16-32
16.5.2	Maintaining RT-11 Format . . . . .	16-34
16.5.2.1	File Deletion on an RT-11 Format Device . . . . .	16-35
16.5.2.2	RT-11 Device Switches . . . . .	16-35
16.5.3	DOS Disk Directory Listings . . . . .	16-37

## **Chapter 17 Storing Files Off-Line: The Backup Program**

17.1 Preserving Files: The Backup Program . . . . .	17-1
17.1.1 BACKUP Mode . . . . .	17-1
17.1.2 RESTORE Mode . . . . .	17-2
17.1.3 LOADINDEX Mode . . . . .	17-3
17.1.4 LIST Mode . . . . .	17-4
17.2 Running Backup . . . . .	17-4
17.2.1 File Specifications in Dialogue Answers . . . . .	17-4
17.2.2 Typographical Considerations . . . . .	17-6
17.2.3 Backup Dialogue Rules . . . . .	17-7
17.2.4 Running Backup under BATCH . . . . .	17-7
17.3 The Dialogue . . . . .	17-8
17.3.1 BACKUP Dialogue . . . . .	17-9
17.3.2 RESTORE Dialogue . . . . .	17-12
17.3.3 LOADINDEX Dialogue . . . . .	17-16
17.3.4 LIST Dialogue . . . . .	17-17
17.4 Interruption Commands . . . . .	17-19
17.5 Running Backup from an Indirect Command File . . . . .	17-20
17.6 Dialogue Examples . . . . .	17-20
17.7 Mounting and Dismounting Volumes . . . . .	17-23
17.8 Backup Error Handling . . . . .	17-25
17.8.1 Dialogue Command Errors . . . . .	17-25
17.8.2 Interruption Command Errors . . . . .	17-27
17.8.3 Volume Mount Errors . . . . .	17-27
17.8.4 Backup Processing Errors . . . . .	17-29
17.8.4.1 Selection Errors . . . . .	17-29
17.8.4.2 Transfer, Deletion, and Listing Errors . . . . .	17-30
17.8.4.3 Informational Messages . . . . .	17-31

## **Chapter 18 Diskette Transfer: The FLINT Program**

18.1 Running FLINT: The Initiation Commands . . . . .	18-2
18.2 Listing the Directory of an IBM Diskette . . . . .	18-2
18.2.1 The Form of the Directory . . . . .	18-3
18.3 Transferring IBM Diskette Data to RSTS/E . . . . .	18-3
18.3.1 Specifying the Known Diskettes of a Data Set . . . . .	18-6
18.3.2 Format of the RSTS/E Disk File . . . . .	18-6
18.4 Transferring RSTS/E Files to IBM Diskettes . . . . .	18-7
18.5 Initializing and Erasing a Diskette . . . . .	18-8
18.6 Dialogue Examples: /DIRECTORY, /TORSTS, /TOIBM,/ZERO, and /ERASE . . . . .	18-9
18.7 FLINT Error Messages . . . . .	18-10
18.8 FLINT as a CCL Command . . . . .	18-12

## Chapter 19 Device Control Programs

19.1	Setting Terminal Characteristics: The TTYSET Program . . . . .	19-1
19.1.1	ESCAPE, ALTMODE, and PREFIX Characters . . . . .	19-5
19.1.2	Lower- and Upper-Case Characters . . . . .	19-5
19.1.3	Generalized Fill Characters . . . . .	19-9
19.1.4	XON/XOFF Remote Reader Control . . . . .	19-9
19.1.5	Output Parity Bit . . . . .	19-11
19.1.6	Private Delimiters . . . . .	19-11
19.1.6.1	Limitations of Private Delimiters . . . . .	19-12
19.1.7	SET as a CCL Command . . . . .	19-12
19.2	Mounting and Dismounting Magnetic Tapes and Private Disks: The UMOUNT Program . . . . .	19-12
19.2.1	Logically Mounting a Disk Pack or Cartridge . . . . .	19-12
19.2.2	MOUNT Options: Disk Pack or Cartridge . . . . .	19-13
19.2.3	Mounting a Magnetic Tape . . . . .	19-15
19.2.4	Logically Dismounting Disk Packs, Cartridges, and Magnetic Tapes . . . . .	19-16

## Chapter 20 Using System Spooling Services: The QUE Program

20.1	Running QUE at a Terminal . . . . .	20-2
20.2	Using the Q Command . . . . .	20-3
20.3	Using the L and S Commands . . . . .	20-8
20.4	Using the K Command . . . . .	20-10
20.5	Using the M Command . . . . .	20-10
20.6	Chaining to QUE from a User Program . . . . .	20-11
20.7	Error Messages and Codes . . . . .	20-12
20.8	Running QUE by CCL Commands . . . . .	20-14
20.9	Running QUE at a Logged Out Terminal . . . . .	20-15

## Chapter 21 The Batch Processing Program: BATCH

21.1	Control Statements . . . . .	21-1
21.1.1	Command Field . . . . .	21-2
21.1.2	Specification Fields . . . . .	21-3
21.1.3	Comments . . . . .	21-3
21.1.4	Syntactical Rules . . . . .	21-3
21.1.5	Syntax Example . . . . .	21-4
21.2	File Specifications . . . . .	21-5
21.2.1	Filename Specification . . . . .	21-5
21.2.2	File Extension Specification . . . . .	21-5
21.2.3	File Specification Defaults . . . . .	21-6
21.2.4	Switch Specification . . . . .	21-7

	Page
21.3 BATCH Commands . . . . .	21-7
21.3.1 \$JOB . . . . .	21-8
21.3.2 \$EOJ . . . . .	21-9
21.3.3 \$BASIC . . . . .	21-10
21.3.4 Utility BATCH Commands. . . . .	21-12
21.3.4.1 \$DELETE . . . . .	21-12
21.3.4.2 \$COPY . . . . .	21-13
21.3.4.3 \$PRINT . . . . .	21-13
21.3.4.4 \$DIRECTORY . . . . .	21-13
21.3.4.5 \$CREATE . . . . .	21-14
21.3.5 \$RUN . . . . .	21-14
21.3.6 \$DATA . . . . .	21-15
21.3.7 \$EOD . . . . .	21-15
21.3.8 \$MESSAGE . . . . .	21-15
21.3.9 \$MOUNT . . . . .	21-16
21.3.10 \$DISMOUNT . . . . .	21-17
21.3.11 \$SORT . . . . .	21-18
21.3.12 \$FORTRAN . . . . .	21-19
21.4 BATCH Operating Procedures . . . . .	21-20
21.4.1 Requesting a Batch Job Run . . . . .	21-20
21.4.2 Batch Processing. . . . .	21-21
21.4.3 Error Procedures. . . . .	21-22

## **Chapter 22 Formatting a Post-Mortem Dump: PMDUMP**

22.1 When to Use PMDUMP . . . . .	22-1
22.2 How to Invoke PMDUMP . . . . .	22-2
22.3 Contents of the Post-Mortem Dump . . . . .	22-2

## **Part V RSTS/E Peripheral Devices**

### **Chapter 23 RSTS/E Peripheral Devices**

23.1 ASR-33 Teletype . . . . .	23-2
23.1.1 Control Knob . . . . .	23-3
23.1.2 Keyboard . . . . .	23-3
23.1.3 Printer . . . . .	23-3
23.1.4 Low-Speed Paper Tape Reader . . . . .	23-4
23.1.5 Low-Speed Paper Tape Punch . . . . .	23-4
23.2 High-Speed Paper Tape Reader and Punch Units . . . . .	23-5
23.2.1 High-Speed Reader Unit . . . . .	23-5
23.2.2 High-Speed Punch Unit . . . . .	23-6
23.3 CR11 Card Reader. . . . .	23-6

	Page
23.4 LP11 Line Printer . . . . .	23-7
23.4.1 Line Printer Character Set . . . . .	23-8
23.4.2 Line Printer Operation . . . . .	23-9
23.5 TC11/TU56 DECTape Control and Transport . . . . .	23-11
23.6 TM11/TU10 and TJU16 Magnetic Tape Control and Transport . . . . .	23-13
23.6.1 Magnetic Tape Control Panel. . . . .	23-14
23.6.2 Magnetic Tape Operating Procedures . . . . .	23-16
23.7 VT05 Aplhanumeric Display Terminal . . . . .	23-19
23.7.1 Controls and Operating Procedures . . . . .	23-21
23.8 2741 Communications Terminals . . . . .	23-23
23.8.1 The ATTN Key . . . . .	23-24
23.8.2 The RETURN Key . . . . .	23-24
23.8.3 The BKSP Key . . . . .	23-24
23.8.4 Bracket Characters. . . . .	23-25
23.8.5 Changing Codes . . . . .	23-25
23.9 LA36 DECwriter II Operator Controls . . . . .	23-31
23.10 RX11 Diskette. . . . .	23-31

## **Appendix A BASIC-PLUS Language Summary**

A.1 Summary of Variable Types . . . . .	A-1
A.2 Summary of Operators . . . . .	A-2
A.3 Summary of Functions . . . . .	A-2
A.4 Summary of BASIC-PLUS Statements . . . . .	A-6

## **Appendix B BASIC-PLUS Command Summary**

## **Appendix C Error Messages**

C.1 User Recoverable . . . . .	C-4
C.2 Non-Recoverable. . . . .	C-11
C.3 Software Performance Report Guidelines . . . . .	C-16

## **Appendix D BASIC-PLUS Character Set**

D.1 BASIC-PLUS Character Set . . . . .	D-1
D.2 ASCII Character Codes. . . . .	D-2
D.3 RADIX-50 Character Set. . . . .	D-3

## **Index**

## **Figures**

2-1 A Configuration of Devices . . . . .	2-3
22-1 Post-Mortem Dump . . . . .	22-4
23-1 ASR-33 Teletype Console . . . . .	23-2
23-2 Teletype Keyboard. . . . .	23-3

	Page
23-3 High-Speed Paper Tape Reader/Punch . . . . .	23-5
23-4 CR11 Punched Card Reader . . . . .	23-6
23-5 LP11 Line Printer System (80-column model) . . . . .	23-8
23-6 Line Printer Control Panel . . . . .	23-9
23-7 TU56 DECTape Transport . . . . .	23-11
23-8 Magnetic Tape System . . . . .	23-14
23-9 Control Panels . . . . .	23-15
23-10 Magnetic Tape Transport Threading Diagram . . . . .	23-18
23-11 VT05 Keyboard . . . . .	23-20
23-12 VT05 Alphanumeric Display Terminal . . . . .	23-21
23-13 Correspondence Code Keyboard . . . . .	23-27
23-14 EBCD Keyboard . . . . .	23-28
23-15 BCD Keyboard . . . . .	23-29
23-16 CALL/360 BASIC . . . . .	23-30

## Tables

1-1 Protection Codes . . . . .	1-8
3-1 A Guide to the Resource Commands . . . . .	3-2
4-1 Assignable Device Specifications . . . . .	4-2
6-1 A Guide to the BASIC-PLUS System Commands . . . . .	6-2
10-1 Overview of Programs and File Information . . . . .	10-2
10-2 CCL Commands that Run System Programs . . . . .	10-3
11-1 File Specification Elements and System Defaults . . . . .	11-1
11-2 RSTS/E Device Designators . . . . .	11-2
11-3 RSTS/E Filename Extensions . . . . .	11-5
11-4 File Protection Codes . . . . .	11-7
11-5 File Attributes . . . . .	11-14
13-1 LOGOUT CONFIRM: Responses . . . . .	13-8
14-1 SYSTAT Options . . . . .	14-2
14-2 SYSTAT Abbreviations . . . . .	14-7
14-3 QUOLST Column Headings . . . . .	14-10
14-4 The MONEY Report . . . . .	14-11
15-1 DIRECT Options . . . . .	15-2
15-2 DIRECT Program Error Messages . . . . .	15-6
15-3 FILCOM Switches . . . . .	15-11
15-4 BPCREF Command Formats . . . . .	15-18
15-5 BPCREF Command Switches . . . . .	15-19
15-6 BPCREF Error Messages . . . . .	15-21
16-1 PIP Switches . . . . .	16-6
16-2 Directory Listing Options . . . . .	16-23
17-1 BACKUP Dialogue Summary . . . . .	17-9
17-2 RESTORE Dialogue Summary . . . . .	17-12
17-3 LOADINDEX Dialogue Summary . . . . .	17-16
17-4 LIST Dialogue Summary . . . . .	17-17
17-5 Interruption Commands . . . . .	17-19
17-6 Backup Dialogue Error Messages . . . . .	17-26
17-7 Interruption Command Error Messages . . . . .	17-27
17-8 Volume Mount Error Messages . . . . .	17-28
17-9 Backup Selection Error Messages . . . . .	17-29
18-1 FLINT Error Messages . . . . .	18-11

	Page
19-1 RSTS/E TTYSET Commands . . . . .	19-2
19-2 Default Single Characteristic Settings. . . . .	19-6
19-3 TTYSET Error Messages. . . . .	19-8
19-4 Generalized Fill Characters. . . . .	19-10
19-5 The UMOUNT Options . . . . .	19-17
19-6 UMOUNT ERROR <text> Messages . . . . .	19-18
20-1 QUE Program Commands . . . . .	20-2
20-2 QUE Job Output Options . . . . .	20-5
20-3 Q Command Options . . . . .	20-6
20-4 QUE Error Messages and Codes . . . . .	20-12
21-1 BATCH Special Characters . . . . .	21-4
21-2 BATCH Commands - Related Default File Extensions . . . . .	21-6
21-3 File Specification Defaults . . . . .	21-6
21-4 Summary of BATCH Error Messages . . . . .	21-23
23-1 A Guide to the RSTS/E Peripheral Devices . . . . .	23-2
23-2 Card Reader Controls . . . . .	23-7
23-3 Line Printer Controls . . . . .	23-10
23-4 DECtape Controls . . . . .	23-12
23-5 Magnetic Tape Transport Controls . . . . .	23-15
23-6 Magnetic Tape Transport Indicators . . . . .	23-16
23-7 FILL Characters Required for VT05. . . . .	23-21
23-8 VT05 Controls and Switches . . . . .	23-22
23-9 2741 Transmission Code Identifiers . . . . .	23-25
23-10 DECwriter II Operator Controls. . . . .	23-31
C-1 Severity Standard in Error Messages . . . . .	C-2
C-2 Special Abbreviations for Error Descriptions. . . . .	C-3
C-3 Non-Trappable Errors in Recoverable Class . . . . .	C-4

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

835ALL

# Preface

## Using This Document

This document describes the RSTS/E Version 7.0 system and its resources for the non-privileged user. That is, the features and system programs discussed in this manual are available to all RSTS/E users. The manual is divided into five parts, and organized as follows:

**PART I, SYSTEM FUNDAMENTALS AND RESOURCES**, introduces you to the RSTS/E system. Part I explains fundamentals such as entering and leaving the system, how data is organized and located, and what Input/Output devices are available.

**PART II, ADAPTING SYSTEM RESOURCES**, explains the use of system resource commands, which allocate and manipulate devices and adapt certain system characteristics to your individual needs.

**PART III, BASIC-PLUS SYSTEM COMMANDS**, explains the use of BASIC-PLUS system commands, which enable you to write, modify, debug, and manipulate BASIC-PLUS programs.

**PART IV, SYSTEM LIBRARY PROGRAMS**, describes and explains the use of the RSTS/E system library programs, which perform various functions of general utility. Among these functions are transferring data, saving data off-line, printing time-sharing statistics, setting terminal characteristics, and batch processing. The overview in Chapter 10 lists these system library programs by function.

**PART V, RSTS/E PERIPHERAL DEVICES**, describes some of the Input/Output devices available to you, and explains their hardware settings and operations.

Your approach to the User's Guide will depend upon your experience with computer systems and languages in general, and with RSTS/E and BASIC-PLUS in particular.

The beginning user, who may have little or no computer experience, should first read Parts I and II, in order to gain an overall understanding of the RSTS/E system and its resources. Anyone programming in the BASIC-PLUS language must use the BASIC-PLUS system commands in order to write and modify programs. The newcomer to BASIC-PLUS, therefore, may expect to use Part III concurrently with the *BASIC-PLUS Language Manual*, which describes the BASIC-PLUS language itself. Beginning users may also find the *Introduction to BASIC* helpful. Users, who write programs in other RSTS/E-supported languages (BASIC-PLUS-2, FORTRAN, etc.), should refer to the RSTS/E User's Guide for that particular language.

The intermediate user, who typically may have some BASIC-PLUS programming experience, and may be still learning about the RSTS/E system, would probably employ Parts III and IV largely as references.

The experienced user, who is familiar with both BASIC-PLUS and RSTS/E, would probably refer most often to Part IV, since some of the system programs it describes are new or modified since the previous version of RSTS/E.

Refer to the *RSTS/E System Manager's Guide* for information on privileged system resources. Refer to the *RSTS/E Documentation Directory* for information on other manuals in the RSTS/E set. Note that all manuals referenced in this text are the most recent version.

Throughout this manual symbols and other conventions are used to represent keyboard characters or aid in the presentation of information. The symbols and conventions used are as follows:

RET	The RET symbol represents a carriage return/line feed combination.
LF	The LF symbol represents a line feed character.
^	The circumflex preceding an upper-case character indicates a control character.
color	Color-highlighted information in examples is typed by the user.
type print	As these terms are used in the manual, the user types and the system prints.
upper case lower case	As used in examples of format, information that must be typed as shown is in upper case, information that is supplied by the user is in lower case.

# **Chapter 1**

## **The System Fundamentals**

### **1.1 Becoming a RSTS/E User: Project-Programmer Number and Password**

To become a user on the RSTS/E system, the system manager must assign you two codes: a unique project-programmer number and a password. These enable you to log in, or gain access to the RSTS/E system and its resources, which include I/O (Input and Output) devices, programs of various functions, information you have stored, the BASIC-PLUS language, and optional resources such as COBOL, BASIC-PLUS-2, FORTRAN-IV, etc. Together, the project-programmer number and the password constitute your identification to the RSTS/E system; the means by which it recognizes you as an authorized user of its time and resources. Each time that you attempt to log into the system, the system will ask for both these codes. If you correctly type your assigned number and password, RSTS/E logs you into the system; but if your response is incorrect, it denies you access.

A project-programmer number, as you would type it, looks like this:

100,101

When printed by the system or in the text of this manual, the project-programmer number is enclosed in square brackets or in parentheses. For example, [100,101] or (100,101) where 100 is the project number, possibly shared by a group of users with a common activity or interest; and 101 is the programmer number, held by only one user within the project group. Thus, each user at a RSTS/E site has a unique project-programmer number. These unique numbers are used to identify sets of data (files) according to the users who own them, and to protect these files from access by certain groups of users. (File protection is discussed in Section 1.4.2.) The project-programmer number is also called the account number, because it identifies each user's reserved area for data storage, or account. Note that the RSTS/E documentation that describes system utilities and optional software may contain references to UICs (User Identification Codes). A UIC is synonymous with a project-programmer number and has the same format and use.

The password is an alphanumeric code that the system manager assigns to you. When you type it during the login procedure, it is not printed on the paper or displayed on the screen. This secrecy prevents unauthorized persons from learning your password and thus gaining illegal access to the system or to your data.

#### NOTE

To preserve the security of the system, do not write down your password; memorize it as soon as it is assigned.

## 1.2 Logging In: The HELLO Command

Once you are assigned a project-programmer number and password, you can log in at a terminal. First, make sure that the terminal's LINE-OFF-LOCAL knob (or switch)\* is set to LINE. This setting establishes a line of communication from terminal to computer. When this communication exists, the terminal is said to be on-line.

Once the terminal is on-line, begin the login procedure by typing the command:

HELLO

then press the RETURN key. This command tells RSTS/E that you wish to use the system. RSTS/E responds to the HELLO command by printing a system identification line and then a number sign (#), which appears at the left margin of the paper or screen. This number sign is the system's prompt to you; it tells you that the system is waiting for you to type your project-programmer number and to press RETURN. Once you have performed these actions, the system responds with another prompt:

PASSWORD :

The system then waits for you to type your password and to press RETURN. Remember, the password is not printed at the terminal.

If the codes are acceptable to the system, you are logged in and the system prints the daily message at your terminal. This message, written by the system manager, contains information on operation schedules, any changes or additions to the system, etc. The system also assigns you a job number, which the system uses to distinguish you from other users currently on the system. Once you are logged into the system, the terminal becomes your console terminal -- the terminal that initiated your access to the system.

If the codes entered are incorrect, the system prints:

INVALID ENTRY - TRY AGAIN

and you can attempt once more to log in.

\*Some user terminals may have a different knob or switch designed to put them on-line. See the appropriate hardware terminal manual.

After five unsuccessful attempts to log in, the system prints the message ACCESS DENIED and ends the login procedure.

The entire process of entering the system is shown in the example that follows. Note that although the RETURN key is pressed to enter each line to the system, it does not echo (appear in print) on the terminal paper or screen. It does, however, have a visible effect, which is to perform a carriage return/line feed operation. This example, as all others in this manual, employs an LA36 (DECwriter II) terminal. The characters you type are printed in a contrasting color to differentiate them from characters printed by the system.

HELLO

RSTS V7.0 Timesharing Job 29 KB3 28-Oct-78 01:57 PM

\*2,201

Password:

17-OCT-78

TO ALL USERS--  
RSTS/E TIMESHARING HOURS WILL BE FROM  
9:30 AM TO 7:30 PM TODAY. FOUR NEW  
TERMINALS ARE AVAILABLE IN THE SYSTEM  
ROOM FOR GENERAL USE.

Ready

The READY prompt following the daily message is printed when you successfully log into the system.\* READY is preceded by a carriage return/line feed operation and followed by a carriage return and 2 line feeds. It indicates that you are at command level, a condition of operation in which you can issue any valid RSTS/E command. At command level, for example, you can type NEW to create a BASIC-PLUS program, or OLD to retrieve one already saved. Or, you can type any of the other BASIC-PLUS system commands, any of the resource commands, or any of the commands that run RSTS/E system library programs. (These types of commands are described in Parts II, III, and IV of this manual.)

Another way to log into the system is to enter the project-programmer number on the same line with the HELLO command, as shown in the following example. This action causes the system to print the PASSWORD: prompt only.

HELLO 2/210

Password:

Ready

Note that in this example, you type a slash (/) rather than a comma to separate the project and programmer numbers. The slash causes no daily message to be printed.

---

\*When you log into most RSTS/E systems, your job is placed under the control of the BASIC-PLUS Run-Time System, which prints a READY prompt. Some systems, however, may assign your job to a run-time system other than BASIC-PLUS, in which case, a different prompt may be printed. In this manual, the examples show the READY prompt. Note that the prompt may differ on your system.

### 1.3 Logging Out: The BYE Command

When you wish to end time sharing and leave the terminal, type the command:

BYE

and then press the RETURN key. Thus you tell RSTS/E that you wish to leave the system. RSTS/E responds to the BYE command by printing the prompt:

CONFIRM:

As the CONFIRM: prompt indicates, RSTS/E is waiting at this point for you to confirm your intention to leave the system. The simplest and most obvious reply is to say "yes"; this you do by typing Y, then pressing RETURN. In response, RSTS/E dismisses you from the system (i.e., logs you out) and prints a statistical report on your storage area and time-sharing session. This report tells you how much of your allotted space you have used, and confirms your identity and that of the system. It also informs you about time: how much has elapsed since you logged in, how much of that time was spent processing your input, and the approximate time of your logout.

On the other hand, after you type BYE, you may change your mind about leaving the system. Perhaps you have remembered that you intended to modify a BASIC-PLUS program, or proofread a passage of text; for whatever reason, you wish to stop the logout sequence, and remain at RSTS/E command level. Obviously, you need to say "No" to the CONFIRM: prompt. This you do by typing N, then pressing RETURN. RSTS/E responds by returning you to command level and printing the READY prompt.

Y and N are two of the five possible replies you may give to the CONFIRM: prompt. The other replies are: I, which allows individual examination and deletion of files; ?, which requests a printing of all possible CONFIRM: replies; and F, which results in a Fast logout. All five replies are described in the following list:

#### The CONFIRM:

Reply	Means:
N	No logout is performed, and the system replies READY.
I	Individual file descriptions are printed, each followed by a ?. To delete a file, type K and the RETURN key; to retain it, just type the RETURN key.
?	A complete listing of CONFIRM: replies is printed.
Y	Yes. Logout is performed if you have not exceeded your disk quota. If you have, RSTS/E will ask that you delete some file(s). To log out, you must be within quota.
F	Fast logout is performed (same effect as Y except that the report is not printed and 3 form feeds are generated).

## NOTE

If you are unfamiliar with RSTS/E files, mentioned in the descriptions of I and Y, read Section 1.4.

For a full description of the logout procedure, including an example of the I response, see Section 13.2.

In the following example, you log out by typing Y.

```
BYE
Confirm: Y
Saved all disk files; 204 blocks in use
Job 29 User 2,201 logged off KB3 at 28-Oct-78 01:58 PM
System RSTS V7.0 Timesharing
Run time was .3 seconds
Elapsed time was 26 seconds
Good afternoon
```

Before leaving the terminal, turn the LINE-OFF-LOCAL knob or switch to OFF. (The LOCAL setting leaves the terminal with power, but disconnected from the system; it then operates as a typewriter.)

## 1.4 The Directory and Files

Once you have written information into your account, you are said to have a directory: a list of the files created and stored, along with information about them--individually and as a group. Files, on the RSTS/E system, can be separate and distinct programs, bodies of data, or empty areas designated for future input.

Perhaps the best way to describe the directory is to look at an example. To print the directory at your terminal, type the command DIR in response to the system's READY prompt. Many users customarily request their directories immediately after logging in; thus, they see at a glance the current status of their work on the system, and are reminded of the names they have chosen for their files.

Consider the following example:

```
DIR
  Name .Ext  Size   Prot      Date    SY:[200,57]
AVERAG.BAS     2 < 60>  28-Oct-78
PERCNT.BAS     2 < 60>  28-Oct-78
TERM .DOC      11 < 60>  28-Oct-78
REPDLO.TXT     43 < 60>  28-Oct-78
PHONE .DIR      43 < 60>  28-Oct-78
EXPENS.BAS     29 < 60>  28-Oct-78
CHOICE.BAS      2 < 60>  28-Oct-78
SOURCE.DAT     20 < 16>  28-Oct-78
BACKUP.CMD      1 < 60>  28-Oct-78
XX .TST         1 < 60>  28-Oct-78
VISITS.LST     14 < 40>  28-Oct-78
```

Total of 168 blocks in 11 files in SY:[200,57]

Ready

The elements of the directory are described in the remaining sections of this chapter.

#### 1.4.1 Filenames and Extensions

Reading the directory from left to right, one notices first each file's filename and extension, which are separated by a period (.). These appear under the headings "Name" and ".Ext" in the example. The filename and extension may be compared to the two parts of a person's name. The filename, or "first name," is unique in that it distinguishes the individual, or file, from others having the same extension or "family name." Thus, one knows that the first two files in the directory, AVERAG and PERCNT, are separate and distinct members of a family or class of files identified by the extension .BAS. This extension in turn identifies the files' class as comprising only files written in the BASIC-PLUS language. Thus, AVERAG.BAS and PERCNT.BAS are the names of BASIC-PLUS programs. Moreover, considering their filenames, one might infer that the first file's function is to calculate an average, and the second file's function is to calculate a percentage.

It is often necessary to abbreviate the identifying words because RSTS/E allows a maximum of six alphanumeric characters in a filename. The maximum allowed for an extension is three alphanumeric characters.

When you create files, you specify the filenames, such as AVERAG and PERCNT. However, you need not provide their common extension, .BAS. Rather, you can allow the system to supply this extension automatically. That is, the RSTS/E system recognizes a BASIC-PLUS source program and supplies the default extension. There are other extensions whose meaning the system recognizes; one of these, .CMD, appears in the directory example. CMD tells the system that this file contains commands, and can automatically control a program--in this case, the program called BACKUP.

After the first two .BAS files, the directory contains some files whose extensions announce that they are not members of the "family" of BASIC-PLUS programs, but belong to other identifiable classes or types of files. TERM.DOC, for example, may be a document of some kind--perhaps a set of instructions on operating a terminal. PHONE.DIR is almost certainly a telephone directory--perhaps of people in your department. SOURCE.DAT has a commonly used extension, which indicates that this file contains data of some kind. The last file, VISITS.LST, may be a listing of visitors to the RSTS/E site. Thus, a filename and extension, which together are often called the file specification, can indicate the nature and function of the file to which they are applied.

Because you can eventually accumulate quite a number of files, of several types, in your directory, it is suggested that you devise a system of descriptive nomenclature, and use it consistently in order to prevent confusion. The names that appear in this sample directory are typical of those used at many RSTS/E sites.

Note the following rules for creating RSTS/E file specifications:

1. Only alphabetic and numeric characters (alphanumerics) are allowed in the filename and extension. Embedded spaces or tabs are not allowed.
2. A filename must contain from one to six characters.
3. Any extension specified must begin with a dot, and can contain one to three additional characters. A null or blank extension is permitted, in which case the dot is included in the file specification, but the extension itself is omitted.

#### **1.4.2 Protection Codes**

Each entry of the directory under the heading "Prot" contains a number in angle brackets < >. This number is called the protection code. Every file is assigned its own protection code, either by the system (as a default) or by the creator of the file. The function of the protection code is to prevent other users from renaming, deleting, changing, or even reading the file. As is evident in the following detail from the sample directory, files can have different protection codes. The differences determine degrees or levels of protection.

CHOICE.BAS	2	< 60>	28-Oct-78
SOURCE.DAT	20	< 16>	28-Oct-78
BACKUP.CMD	1	< 60>	28-Oct-78
XX .TST	1	< 60>	28-Oct-78
VISITS.LST	14	< 40>	28-Oct-78

Verbally, one can express a file's level of protection in terms of two variables: the types of users from whom the file is protected, and the actions--reading and writing--from which it is protected. For purposes of file protection, the system recognizes three classes of users, identified by their project-programmer numbers:

1. The owner (the creator of the file).
2. The owner's group, composed of all users having the same project number as the owner.
3. All other users not in the owner's group.

Since these two variables--read/write privileges and class of user--determine protection, files are protected by a number of combinations. One file, for example, might be write protected against the owner's project group; that is, other users with the owner's project number could read the file, but could not rename it, write anything into it, change any information presently there, or delete the file. Another file might be read protected as well as write protected against this same group, whose members would then be unable even to view the file's contents at their terminals. Still another file might be write protected only against users who do not share the owner's project number; this file could be read by any user at the RSTS/E site, but could be modified only by the owner and members of his project group.

These two combined criteria for file protection—read/write access and user class—are designated and recognized numerically by the system, in the form of the protection code. Table 1-1 lists and describes the numerical protections and their equivalents.

**Table 1-1: Protection Codes**

Code	Protection Provided
1	Read protected against owner.
2	Write protected against owner.
4	Read protected against all others in owner's project group.
8	Write protected against all others in owner's project group.
16	Read protected against all others who do not have owner's project number.
32	Write protected against all others who do not have owner's project number.
64	An executable program; if the file has protection <64>, the other codes (1 to 32 above) have different meanings (see Section 11.4).
128	An executable program with temporary privileges or a protected data file; overwritten with zeroes when deleted.

Typically, a file's protection code is a decimal number that is the sum of the desired combination of the values in Table 1-1. In the sample directory, for instance, the most common protection code is <60>, or the sum of the following codes as listed in the table:

32	=	Write protection against users without the owner's project number.
+16	=	Read protection against the group above.
+ 8	=	Write protection against the owner's project group.
+ 4	=	Read protection against the group above.
<hr/>		
=<60>		

Thus, the file AVERAG.BAS, whose protection code is <60>, can be neither read nor written into by anyone except its owner; that is, the user who created it. It may be of some interest to the reader that <60>, on this particular system, is the default—the protection that the system automatically assigns when you create a file without specifying a protection of your own choice. Thus, the system may be said to respect the individual user's privacy to a high degree, automatically protecting your files against all others, including members of your own project.

Note that the file VISITS.LST in the sample directory has a protection code of <40>, or the sum of the values 32 (write protection against non-members of the owner's project group) and 8 (write protection against the owner's project group). VISITS.LST, therefore, cannot be written into or modified by anyone except its owner, although any user at the site may read its information. For example, if this file contains a listing of visitors to the RSTS/E site and the creator of VISITS.LST has sole authority in determining who shall be considered a "visitor" (as opposed to a client, candidate for employment, or intruder), there would be good reason to protect the file from being capriciously edited or amended by other users. Thus, one appreciates some of the reasons for the variety of protection codes that may be assigned. The following are some typical protection codes and their meanings:

- |      |           |   |
|------|-----------|---|
| <60> | 32+16+8+4 | Read and write protection against everyone but owner.                         |
| <48> | 32+16     | Read and write protection against all who do not have owner's project number. |
| <40> | 32+8      | Write protection against all but owner.                                       |
| <42> | 32+8+2    | Write protection against all including owner.                                 |
| <0>  |           | No protection at all (any user may read and write).                           |

For executable programs, 64 is added to the above codes and a slightly different meaning is applied to them (see Section 11.4).

#### NOTE

A file which has a protection code of <128> is overwritten with zeroes when it is deleted. That is, when you execute an UNSAVE or KILL on a privileged program or protected file (code <128>), the system overwrites that file with zeroes.

#### 1.4.3 Account and File Statistics

The directory, in addition to listing the names of files and their protection codes, also lists information about the individual and collective status of files under your account. This information tells both you and the system who "owns" the files, when they were created, how many there are, and how much storage space they occupy.

Name	.Ext	Size	Prot	Date	SY:[200,57]
AVERAG.BAS		2	< 60>	28-Oct-78	
PERCNT.BAS		2	< 60>	28-Oct-78	
TERM .DOC		11	< 60>	28-Oct-78	
REPDL0.TXT		43	< 60>	28-Oct-78	
PHONE .DIR		43	< 60>	28-Oct-78	
EXPENS.BAS		29	< 60>	28-Oct-78	

File ownership is indicated by the item at the right of the directory, SY:[200,57], which shows that all the files listed are on the public structure (identified by SY:, see Section 2.2) and belong to the user who holds account (or project-programmer) number [200,57].

Within the individual file listings, the number appearing under the heading "Size" refers to the number of blocks that each file occupies on the disk. On the RSTS/E system, a block is equal to 512 bytes of 8 bits each. The file TERM.DOC for instance, occupies 11 blocks of disk space. PHONE.DIR occupies 43 blocks, EXPENS.BAS 29 blocks, and so on.

A summary line at the end or bottom of the directory tells you how many blocks are occupied by all the files together, how many files there are, and to what account number they belong. In this sample directory, the summary line looks like this:

```
Total of 169 blocks in 11 files in SY:[200,57]
```

Finally, note the date that appears at the end of each file listing. This date indicates when the file was created. In this example, all files were created on October 28, 1978.

## **Chapter 2**

# **The System Resources**

### **2.1 Introduction to the RSTS/E Devices**

Without a set of devices at your disposal, you would be unable to print directories, obtain access to files, write programs, or even log into the system. In short, you would be unable to accomplish any work with the computer, because you would have no way to input your data, or to cause it to be output once it has been processed. The RSTS/E devices, therefore, are the vital media by which you and computer write and read information, and communicate it to one another.

The terminal is a familiar example of a device. By typing at the terminal, you transmit information to the computer. If, for instance, this information consists of a project-programmer number and password, the computer acts on it by logging in a new job. A program can also make use of a terminal by causing it to print text and by accepting input data from its keyboard.

Thus, these media of communication between human being and computer, because of the kinds of data transfer they help to perform, are often called I/O (for Input and/or Output) devices. And, as that abbreviation suggests, some are used for input and output, some for input only, and some for output only. In the procedure just described, for example, the terminal functions as an input and output device: input, because you employ it to write information into the system; and output, because the system employs it to write out an appropriate response to you.

Another previously described procedure, asking for a directory, illustrates the I/O function of the disk as well as that of the terminal. You type DIR at your terminal. The system, recognizing this command, reacts to it by scanning the disk for your directory. Once the system finds the directory, it inputs it from the disk into memory. Finally, it outputs the directory to your terminal.

While the terminal and disk can both be used for input and for output, some devices can be used for only one of these functions. The paper tape reader and the card reader are input-only devices. The computer can use them to read in your input from punched tape or cards, but not to write out its own processed output. The line printer, however, is an output only device; the computer can use it to write out (print) its processed output, but not to read in your input.

Because the RSTS/E system affords you some degree of choice in I/O devices, you should be aware of the functional differences among them, and of the limitations and advantages of each. If you know the devices, you can easily decide which one best suits current informational needs. For example, you may wish to produce a printed copy of LOTS.TXT, a file that contains 100 blocks of text: clearly, a voluminous body of data. You can, of course, request that the system use the terminal for output. But if you do, you are in for a long and probably a frustrating wait while the terminal labors to print all 50,000 characters, a character at a time. You would be far better advised to choose the line printer for output, since this device is specifically designed for fast printing. Moreover, while the line printer is printing your file, you are able to employ your terminal for input of other information.

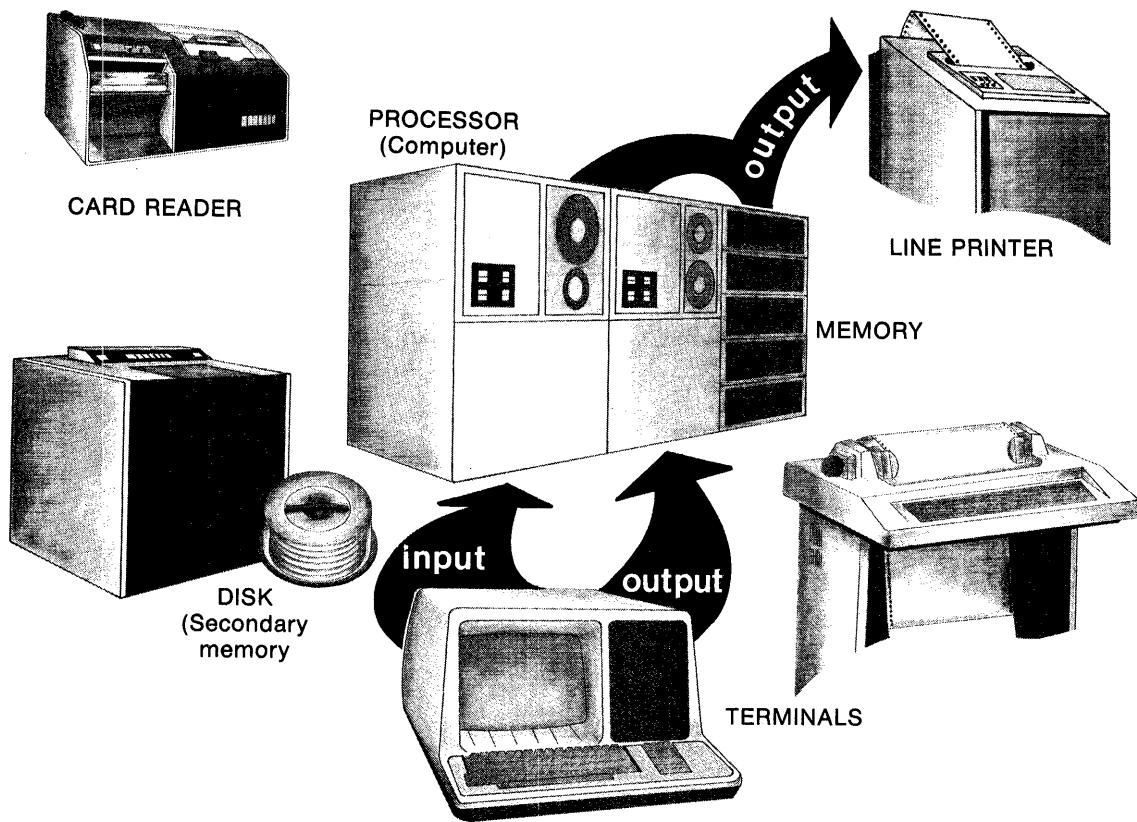
Experienced users of the RSTS/E time-sharing system often make such I/O choices, preferring one device over another because of its greater efficiency or convenience. At a typical site, for example, one might find a user who has written (input) the BASIC-PLUS program MYFIL.BAS onto the public disk structure, the set of disks that is shared by all the other current users. Now assume that this user wishes literally to take the program away and run it on another RSTS/E system three hundred miles away. Obviously, a person can not remove a disk from the public structure and carry it off; it contains the programs and data of many other users. The user can have the system copy (output) MYFIL.BAS onto a DECTape, a small I/O device consisting of a 260-foot magnetic tape wound on a plastic spool. This device, measuring about .75 by 3.75 inches in diameter, solves the problem of portability. Once this user arrives at the distant RSTS/E site, the computer can copy (output) MYFIL onto its own public structure, from which the user may then run the program.

Another user may need to air-mail a FORTRAN program IFILE.FOR to an overseas RSTS/E site; to save postal costs and prevent damage, the program is copied (output) from the public structure--where it currently resides--to another device, the paper tape punch. This device "writes" the program, in the form of coded perforations, on a paper tape, a long, accordion-folded strip of paper that, obviously, is lighter in weight and less fragile than a spooled tape. Once the paper tape arrives at its destination, programmers there may first input IFILE by using the paper tape reader (an input-only device), then have their system output the program to the public disk structure.

Thus, a number of specialized Input/Output operations--reading, writing, copying--are carried on by user and computer with the help of devices, each with its own unique capabilities and limitations.

Figure 2-1 illustrates these devices as they might be configured--put together and linked to the computer to form a system--at a typical RSTS/E site. Note that the machine labeled "computer" is also called the "processor," since it computes or processes the data. Sometimes, it is called the CPU (Central Processing Unit).

**Figure 2-1: A Configuration of Devices**



## 2.2 The Public Disk Structure

Users share computing time on the RSTS/E system, with each being allotted a "slice" of the processor's time. Users may share another system resource as well: the array of devices. In the foregoing section, it was noted that one set of devices (disks) is shared by a number of users. This shared set of devices is called the public disk structure, because it is always accessible to all users and because the system treats it as a unit. On some systems, it may include many separate disk packs. One of these, the system disk, contains the system code, language processors, and, possibly, the library of system programs, some of which are described in Part IV of this manual. The other disk packs (the

public disks) contain the directories and files of the users. (The system disk too may contain some user directories and files in addition to its system information.)

When you are logged into the system and working with a disk file, you are usually unconcerned about which disk in the public structure happens to contain that file. The particular disk is chosen by the system according to current time-sharing needs. Each of the disks contains a master list of users' accounts, and, for each account, a list of all files stored under that account. The system, by using these lists, is able to locate your file when you request it from your terminal.

This location process can be explained by returning to an example of system operation discussed in the preceding section: requesting a directory. When you type DIR, you ask the system to print your file directory at your terminal. Once the system recognizes this command, it responds by running the DIRECT program, which scans the master list of accounts on each disk. The master list discloses the location of the list of your files--your directory. Having found the directory, the DIRECT program prints it at your terminal.

Another frequent user action--listing a stored program at the terminal--further illustrates the operation of the public structure. In this case, you are logged in under account [100,105] and have been working on one file for some time--perhaps modifying a BASIC-PLUS program. You now complete your modifications, check the revised file, and save it. Before logging out, however, you wish to edit another BASIC-PLUS program: MYFILE.BAS, a file containing 30 or so lines: brief enough to be quickly and conveniently printed at the terminal. Before you can edit the program, you must take several steps: you must tell the system that you wish to access an "old" (previously stored) program, you must tell it the program's filename, and finally you must tell it to list the program at your terminal.

You begin by typing the command OLD; when the system prompts with OLD FILE NAME--, type MYFILE (the system assumes the extension .BAS). Once the system receives the filename, it locates the file, scanning each public disk's master list of accounts in an effort to locate your file directory on that disk. Thus, the system is guided through the public disk structure to file directory [100,105], and ultimately to the location of MYFILE.BAS. When the READY prompt appears at your terminal, type LIST, whereupon the system, understanding this as a request that it list the current program, MYFILE.BAS, prints (outputs) the file at your terminal. Now you can read the file and modify it as you wish.

### 2.3 Private Disks

A good deal of system file activity--such as creation, access, editing, and deletion--takes place on the public disk structure. Since it is the largest constantly available medium of file storage, it is generally the busiest. But not all the disks used on a system need be in the public structure. Some of them may be private disks, disk packs or cartridges that belong to a single user account or perhaps to a few user accounts, in the sense that these accounts alone are

on the disk(s). Only if a private disk already contains an account can files be created under that account on the disk. Without one of these private disk accounts, you can read or edit a private disk file, but only if its protection code permits.

For example, assume that a private disk mounted on drive DL3: belongs only to the members of a specific project group, to those users with project number 200. In such a case, the users who hold account numbers [200,30], [200,31], [200,32], [200,33], etc. may all create files on private disk DL3:. If you have account [210,33], however, you cannot create files on DL3:, although, protection codes permitting, you can read or edit files already created on that disk.

From your point of view, then, one important difference between a private disk and one on the public structure is that you can always create a file on the public disk, whereas you can do so on a private disk only if you have an account number there. On both types of disk, file protection codes govern your read and write access to existing files. Another difference is that a private disk can be mounted or dismounted while the system is running, whereas a public disk must always be mounted during time sharing. This difference, obviously, is of special concern to the owner(s) of the private disk and to the system manager, who, it should be noted, determines which disks on the system are public and which are private.

## 2.4 Assignable Devices

Disk, public and private, are generally not assignable. That is, you cannot, except in rare and special cases, request one for your exclusive, temporary use. Even a private disk is usually shared by a number of users. To satisfy the frequent need for input and output media devoted to your work alone, RSTS/E provides an assortment of assignable devices. DECTapes and magnetic tapes, for example, are assignable by you for your own input or output; card punches and paper tape punches are assignable for your output. Their assignability depends, of course, on their physical availability: obviously, if a DECTape or a card punch is being used, you cannot immediately assign it to yourself; should you try, the system will print an appropriate message at your terminal (?DEVICE NOT AVAILABLE, for example).

Within this class of assignable devices, there are degrees of accessibility. A DECTape, for example, with a single directory and no accounts, permits access to all the files it contains. A magnetic tape, on the other hand, which contains account numbers associated with its files, permits access to any files if you know their account numbers. Paper tape punches and card punches, being unit record devices, contain no files and therefore impose no such restrictions on access; nor do line printers, which are also unit record devices. If nobody else is using a punch or printer, and if the system manager has not restricted its availability, you are free to assign it. And anyone may use an unoccupied terminal--possibly even a person who lacks an account number and password and is therefore not a recognized user, since there are commands that do not require logging in.

## 2.5 A List of RSTS/E Devices

The following list contains brief descriptions of devices available on RSTS/E systems, including their general functions, advantages, and disadvantages. Following the full name of each device are two items: the abbreviated name, or the specification, by which it is known to the system, and its I/O function (input and output, input only, or output only).

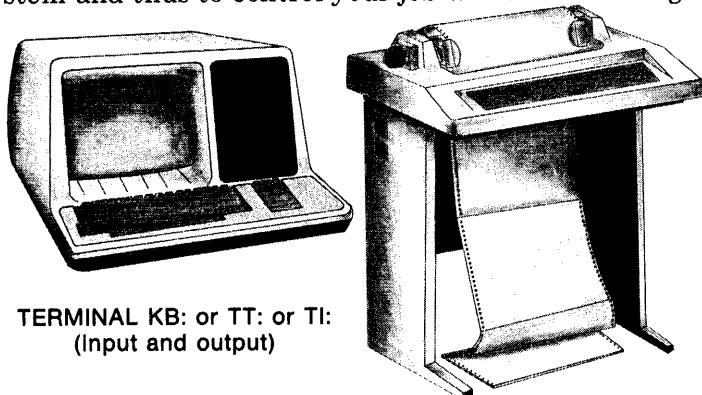
Devices are also classified as file structured or non-file structured; a file structured device can store files, while a non-file structured device cannot. However, a file structured device can be treated by the system as non-file structured. This capability allows you to bypass some of the limitations of file structured devices.

The devices are listed according to degree of legibility: those specifically intended for human reading appear first; next come those which, with some effort, are humanly decipherable; finally come those which only the computer can read.

The terminal is a non-file structured device. In addition to human readability, its most significant advantage is that it is interactive; that is, it allows you to communicate with the system and thus to control your job while it is running. Another obvious advantage of the terminal is its operational similarity to a common off-line, non-file structured, input-only device: the typewriter.

A disadvantage of the terminal is its inconvenience for output of large amounts of data. Com-

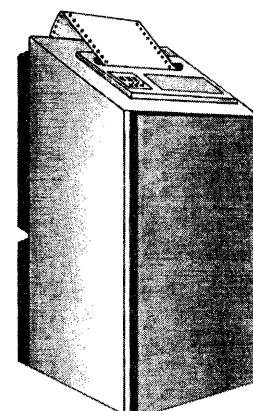
TERMINAL KB: or TT: or TI:  
(Input and output)



pared with a line printer, a hard-copy terminal prints quite slowly. And though a video terminal may print rapidly, it may not produce a paper copy that you can remove and read at your own pace. (Some video terminals are equipped with hard-copy devices, but these print slowly.)

The line printer is a non-file structured device. Like the terminal, it produces human-readable output, but at much greater speed. This speed is its most important advantage over the terminal; the line printer is the fastest available device for producing hard copies of information.

The line printer's relative disadvantages arise from its singular purpose: simply to output information in the form of a human-readable hard copy (a listing). Its output therefore, unlike that of tapes, cards, and disks, cannot be read by the computer: there is no way to recycle one of its file listings through the system. Therefore, if the printed file has been deleted from the machine-readable device (disk, DECtape, etc.) on which it resided, you have no way to edit it by computer. You must resort to manual editing:

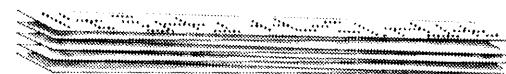


LINE PRINTER LP:  
(output)

handwriting, mark-up, cutting, pasting, typing, etc. And, if that deleted file was a program, you obviously cannot run it. Another disadvantage of the line printer is its inability to output more than one user's data concurrently, in the manner of the disk. Before assigning a busy line printer, you must wait for it to finish its current job.

Paper tape is a non-file structured device. It has three significant advantages over other machine-readable media: it is inexpensive, easily shipped or mailed, and can be deciphered by a person who knows its punched code.

Most of paper tape's disadvantages are inherent in its physical makeup. Holes punched in paper cannot be erased or satisfactorily repaired: thus, a paper tape must always contain the same data; changing or editing requires a new tape. Also, paper is a low-density storage medium: a good deal of it is required to hold information in any form--even as tiny perforations. And paper, of course, is easily torn or creased.



**PAPER TAPE PR: and PP:  
(input and output)**

In addition to its intrinsic disadvantages, paper tape, for full usability, requires two additional devices, one for input and one for output: namely, the reader and the punch. These are described below:

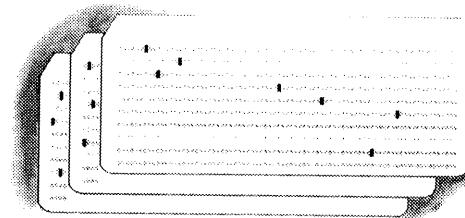
#### **PAPER TAPE READER PR: (input)**

The paper tape reader is a non-file structured device. Its advantages are that it is compact and performs automatic input that is faster than user input from a terminal. Its input speed, however, is much slower than that of DECtape, magnetic tape, and disk.

#### **PAPER TAPE PUNCH PP: (output)**

The paper tape punch is a non-file structured device. Its important advantages and disadvantages for output are the same as those of the paper tape reader for input.

Cards are non-file structured. Like paper tape, they are machine readable but humanly decipherable. Because they are inexpensive, easy to mail individually or in small quantities, and can be coded or annotated with a pencil, they are widely used to gather data for public and commercial operations: school registrations, consumer billings and surveys, and so on. Cards can be prepared off-line (by keypunch), thus conserving system time and resources.

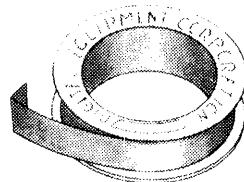


**CARDS CR: and CD:  
(input and output)**

The disadvantage of cards is their susceptibility to damage. Composed of card paper, they also share all the disadvantages of paper tape. For full usability, cards require two additional devices: the CARD READER (CR: and CD:) and CARD PUNCH.

DECtape is a file structured device. Its significant advantages over paper storage devices are its far greater I/O speed, its higher density of data storage,

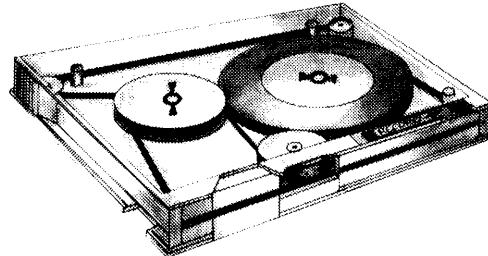
and its reusability. Unlike paper tape or cards, it can be erased, edited, and rewritten. And compared with the paper media, it involves less handling, because it requires only one additional device to fulfill its I/O capability: the DECTape drive, which performs both input and output. DECTape, because it has a directory structure, also allows you to change files in place, and therefore requires less manipulation by the system than does a magnetic tape. Because of its small size, DECTape is easily handled, carried, and stored. It is physically stronger and less sensitive to climate than cards, paper tape, and magnetic tape.



DECTape DT:  
(input and output)

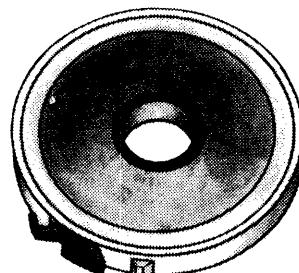
DECTape's small size presents one disadvantage: it cannot store as much data as a magnetic tape or disk. Also, its storage density is less than that of a disk. And, because DECTape is a magnetic storage device, it cannot be humanly decoded.

DECTape II is a non-file structured device. Except for its non-file structured property, it has the same advantages and disadvantages as the previously described DECTape. That is, I/O speed, high density of data storage, reusability, and in place file modification are all advantages of DECTape II. And, as with DECTape, DECTape II has the disadvantage of lower storage density compared to disk or magnetic tape.



DECTape II DD:  
(input and output)

Magnetic tape (also called magtape) is not a true file structured device. Unlike DECTape, it has no directory; it does, however, contain an individual account associated with each file on the tape. It shares with DECTape the following advantages over paper storage devices: greater speed, higher density, reusability, and less handling. Also, it requires only one other device for both input and output: the magnetic tape drive. The magnetic tape's size and shape make it convenient for storing system files off-line.

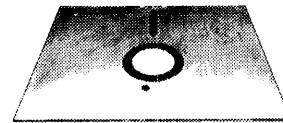


MAGNETIC TAPE:  
MT:, MS:, or MM:  
(input and output)

Magnetic tape shares two relative disadvantages with DECTape: its speed and density are not as great as those of the disk. And, as noted, it is somewhat more susceptible to damage than DECTape, and requires more system manipulation. Also, magnetic tape is a sequential medium: in order to reach a specific file on the tape, the system must first read all the files preceding it. And a specific magnetic tape file cannot be deleted without also deleting all the files that follow it.

The flexible diskette is a non-file structured device. Diskettes share with DECTape and magnetic tape the speed, density, reusability, and handling

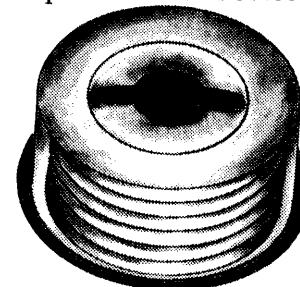
advantages that paper storage devices lack. The flexible diskette, also called a floppy disk, resembles a 45 RPM record. It consists of a thin, flexible surface on which data is recorded. This surface is encased in a plastic envelope with a slot for a read/write head and a drive spindle hole in the center. These physical characteristics make the diskette easy to store and ship.



FLEXIBLE DISKETTE DX:  
(input and output)

Because data can be stored only on one side of the diskette, its storage capacity is less than that of disk, DECTape, or magnetic tape. Also, because it is a magnetic device, it is not humanly decodable. A flexible diskette drive is required to read and write the diskette.

The disk is a file structured device. Of all devices, it is fastest, highest in density, largest, most reliable, and most durable. Like DECTape, flexible diskette, and magnetic tape, it is reusable. And it far surpasses those devices in the number of accounts and files it can hold. For input and output, the disk requires one additional device: the disk drive. The disk involves less human handling than any other device. For all these reasons, it is chosen for the RSTS/E public structure.



DISK SY:, DF:, DS:, DK:, DL:, DM:, DP:, DR:, or DB:  
(input and output)

## 2.6 Device Names: Physical and Logical

### 2.6.1 Physical Device Names

In the foregoing list of RSTS/E devices, each device's specification, the name by which it is known to the system, appears beside the device's full name. For DECTape, for example, the system's name is DT: or DD:. Such a specification, also called a physical name, is generally followed by a decimal unit number, which, in the case of a storage medium, is the number of the drive on which the medium is mounted. This number, therefore, serves to distinguish devices of the same kind according to their physical locations on the system.

For example, three users may be simultaneously creating files on three DECTapes, physically named DT0:, DT1:, and DT2:. These physical device names separate the three DECTapes—from the point of view of each user and of the system.

To elaborate on this example, one of these three users, the current "owner" of DT0:, is creating on that device a text file named NANCY.TXT. From the system's point of view, that file's more specific name, or file specification, is DT0:NANCY.TXT—the filename and extension, preceded by the physical name of the device on which the file resides. The standard colon (:) serves not only to identify the device name, but also to separate and to distinguish it from the filename.

A device name, in a file specification, performs the important function of identifying a file unequivocally. For example, assume that you are working on a file contained on DECtape whose specification is DT1:FIRST.BAS and that you have assigned the device DT1: (see Section 4.1). Furthermore, you already have on your account an existing file of the same name--not on DECtape but on the public disk structure, a resource you have not been using because you have been working on the DECtape instead. Now you close your DECtape file and return to the public structure, without first relinquishing your ownership of the DECtape DT1: via the DEASSIGN command (see Section 4.2). Thus, although you are sharing the public disk structure with a number of current users, you still own the DECtape DT1:, because you have not revoked your original device assignment command, ASSIGN DT1:.

Sometime later, you wish to reopen and edit the file DT1:FIRST.BAS--that is, the FIRST.BAS that resides on your private DECtape, not the FIRST.BAS that resides on the public structure. If you were to forget the physical device name DT1: in specifying the file, and simply type FIRST.BAS, you would summon not the DECtape but the disk file, and with it ample possibility for confusion. The system retrieves the disk file FIRST.BAS because the system always assumes the public disk structure as the default device.

Protect yourself against this sort of confusion by being careful when specifying duplicate filenames. The system, fortunately, cannot be so confused: it always remembers files not only by their names and extensions, but by their host devices as well. Thus, while you may have created, at various times, a number of files on devices other than public disks, the RSTS/E system would recognize these files by the following specifications: DT2:DATA3.REP, DT3:INDEX.001, MT1:ARTHUR.NAA (a magnetic tape file), and DK5:PRIVAT.GRP (a private disk file).

### **2.6.2 Logical Device Names**

Any RSTS/E device can have, in addition to its physical name, an assigned logical name. In other words, you may give to a device a name of your own choosing. This name, like a filename, can contain from one to six alphanumeric characters, without embedded nulls, tabs or spaces, and including the standard colon separator (:). By issuing a variation of the ASSIGN command, you may, for example, assign the logical name INDEX: to DECtape DT3:. This logical name will be recognized by the system until you DEASSIGN the logical name.

Logical names are used to describe the nature of the information residing on a device, and therefore may be easier for you to remember than the physical name with its unit number.

But there are more specific reasons for logical names. The fundamental advantage of a logical name is that, unlike a physical name, it does not depend upon where (on what drive) the medium is mounted. Therefore, you can choose a logical name without regard to what device drives may be available at some future time. For example, if you write a BASIC-PLUS program that, at several points, accesses a specific magnetic tape for statistics, you can refer

to this tape by the logical name STATS. The advantage of STATS over a physical name is this: if you were to specify a physical name such as MT1: for the magnetic tape, the success of your program, as written, would depend on the availability of tape drive #1 at the time the program runs. What if the drive were in use or inoperative at the time? If another drive, say #2, were available, you could use it, but would first have to edit your program accordingly, changing each occurrence of "MT1:" to "MT2:"- a tedious procedure. So by assigning the magnetic tape the logical name STATS before running the program, you ensure the system's recognition and access of that tape whether it is mounted on drive #1, #2, #3, etc.

A similar use of logical names is to enhance the efficiency of a batch job, an operation which does not require terminal interaction. Typically, such a job is "programmed" by one user for later execution by another. For instance, assume that you create a large batch job in the morning to be run by an operator at night, when there are fewer users on the system. The batch "program" or control file, like the BASIC-PLUS program in the preceding example, accesses a magnetic tape. If you refer to the tape by a logical name in the batch control file, and inform the operator of this name, the operator need not be concerned about the availability of a specific tape drive. Any drive will do, since the system recognizes the logical name once the operator assigns it.

Note that the system always recognizes and accepts a device's physical name, whether or not a logical name has been assigned to that device. Thus, you can specify a DECTape running on drive #2 and assigned the logical name STAR as DT2: or as STAR. Also, a physical device name preceded by an underscore causes the system to access that device, regardless of any prior assignment. Some logical names, at the system manager's discretion, can be made system-wide: known to, and usable by, all other users. One such system-wide logical name is SY:, which designates the public disk structure. Since SY: is a logical name for a set of devices, it need not have a unit number, and would probably be confusing if it did. SY0:, however, is always the logical name for the system disk.



## **Chapter 3**

### **Functions of the Resource Commands**

This part of the User's Guide contains a set of commands which apply and adapt system resources such as devices and accounts to the needs of the individual user. Among the functions performed by these commands are device assignment and deassignment, logical naming of devices and accounts, and changing protection of files. Thus, the resource commands enable you to make full use of available hardware and of services.\*

The resource adaptation commands have a format that resembles English grammar. Note that most of these commands are English verbs: ASSIGN, DEASSIGN, and REASSIGN. And as verbs, they often specify objects of their actions. These objects can be device specifications, protection codes, accounts, etc. Therefore, the command string:

ASSIGN DTO:

which consists of the command ASSIGN followed by the device specification DTO:, causes the system to reserve DECTape unit 0 for the user who has given the command.

Sometimes a resource command need not specify an object of its action, either because the object is understood by the system, or because an object is not needed to complete the command's meaning. DEASSIGN used alone, for instance, causes the system to release all devices from the user's control. And TAPE, which never needs an object, causes the system to disable the terminal echo feature.

---

\*Before attempting to issue commands, you should check to see if your job has been set in BASIC-PLUS EXTEND mode, in which spaces and tabs are significant (see Section 8.2.1).

Table 3-1 is an overview of the resource commands—a guide to their functions and their locations within Part II.

**Table 3-1: A Guide to the Resource Commands**

	Command & Format	Section
<b>Physical Devices</b>		
Reserving	ASSIGN dev:	4.1
Releasing	DEASSIGN DEASSIGN dev:	4.2
Transferring control	REASSIGN dev:job-number	4.3
Disabling terminal echo	TAPE	4.8.1
Enabling terminal echo	KEY	4.8.2
<b>Logical Names</b>		
Associating with device	ASSIGN dev:logical-name	4.4 4.4.1 4.4.2
	ASSIGN dev: [proj,prog] logical-name	4.4.4
Associating with account	ASSIGN [proj,prog]	4.7
Cancelling logical association	DEASSIGN logical-name DEASSIGN dev: logical-name DEASSIGN DEASSIGN [proj,prog]	4.4.3 4.7
System-wide logical names		4.5
Pack identification label		4.6.1
<b>Logically Named Devices</b>		
Reserving	ASSIGN logical-name:	4.4.3
Releasing	DEASSIGN logical-name:	4.4.3
Logically named disk pack or DECpack cartridge		4.6.2
Logically mounting disk pack or DECpack cartridge		4.6.1.1
<b>Changing Defaults</b>		
Changing protection default	ASSIGN <prot>	5.1
Changing magnetic tape labeling default	ASSIGN MTn:.label	5.2

## **Chapter 4**

### **Controlling Devices and Accounts**

#### **4.1 Reserving a Device: The ASSIGN Command**

The ASSIGN command reserves an I/O device for the use of one programmer (i.e., one job number).

To reserve a device for your exclusive use, type the command ASSIGN and an object, in this form:

ASSIGN dev:

The object dev: is a device specification. (For a list of possible specifications, see Table 4-1.) If the device is available for use, the system prints the prompt:

READY

Following the READY prompt, you can then perform I/O with the assigned device.

If the device is not available for use, the system prints the message:

?DEVICE NOT AVAILABLE

There are several reasons for a device's unavailability; it may be 1) opened or assigned by another user, 2) reserved for a specific operation (a terminal reserved for network communications), 3) restricted by the system manager for privileged use or hardware maintenance.

If the device is not configured on the system, the system prints the error message:

?NOT A VALID DEVICE

The following example illustrates successful assignment of line printer 0, but the assignment of the high-speed paper tape reader is unsuccessful because that device is unavailable.

ASSIGN LP:

READY

ASSIGN PR:

?DEVICE NOT AVAILABLE

If more than one job is logged into the system under a single account number, only the job (i.e., user) performing an ASSIGN (or DEASSIGN) is affected by that command. Devices reserved by a job remain in that job's control until the job releases the devices or logs off the system. Because devices are controlled by a job, the device assignments remain in effect even if you change accounts.

**Table 4-1: Assignable Device Specifications**

Specification	Device
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.
MM0: to MM7:	TE16/TU16/TU45/TU77 magnetic tape units 0 through 7.
MS0: to MS3:	TS11 magnetic tape units 0 through 3.
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.
KB:	Current user terminal.
KBn:	Terminal n in the system.
TTn:	Terminal n in the system (synonym for KBn:).
TI:	Current terminal (synonym for KB:, the terminal that initiated the job).
DX0: to DX7:	RX01/RX02 flexible diskette units 0 to 7.

#### NOTE

You can reference LPn:, DTn:, DXn:, KBn:, MMn:, MTn:, MSn:, and DDn: where n is between 0 and the maximum number of such units on the system. LP:, DT:, DX:, MM:, MT:, MS:, and DD: are each the same as specifying unit 0 of the related device.

If your system has only TS11 tape units, the designators MS: and MT: are synonymous. If your system has only TE16, TU16, TU45, or TU77 tape units, the designators MM: and MT: are synonymous. On systems which have the CD11 card reader, the designator CR: is synonymous with CD:.

## 4.2 Releasing a Device: The DEASSIGN Command

The DEASSIGN command releases an assigned device from your control back to the system's supply of available devices. Thus, DEASSIGN makes the device available for use by other jobs.

Issued with no device specification, DEASSIGN releases all of the job's assigned devices. For example:

```
DEASSIGN
```

releases all devices that you have assigned under the current job number. If you do not issue this command before logging out, the system itself performs a DEASSIGN during log out.

To release a specific device, type the DEASSIGN command followed by a device specification. For example:

```
DEASSIGN LP:
```

releases line printer unit 0.

## 4.3 Transferring a Device: The REASSIGN Command

The REASSIGN command transfers control of a device to another job. For example, if DECtape unit 1 is under control of the current job, the command:

```
REASSIGN DT1:8
```

transfers control of the DECtape to job number 8.

In performing this transfer between two jobs, the REASSIGN command also prevents a third job from gaining control of the device. In the foregoing example, job number 8 might be busy with an operation that prevents it from using DT1: immediately after reassignment. If job number 20, for instance, attempts to assign this DECtape before job number 8 is ready to use it, the attempt will be unsuccessful.

An attempt to reassign control of a device to a nonexistent job causes the system to print the %ILLEGAL NUMBER error. If the device is open or has an open file, the system generates the error ?ACCOUNT OR DEVICE IN USE. Before transferring control, you must close the device, or close any files open on the device.

Magnetic tape users should note that the labeling format, density, and parity characteristics assigned to a tape unit are preserved in the REASSIGN command's transfer.

## 4.4 Assigning and Using Logical Names: The ASSIGN and DEASSIGN Commands

The logical names that you assign for devices, discussed in Section 2.6.2, do not depend on the physical device specifications. Thus, if you write a program referencing physical devices, you can give these devices logical names of your own choosing in the program. Before running the program, issue the ASSIGN command to associate your chosen names with the devices. This action makes the program independent of the devices' physical locations on the system.

To associate a logical name with a physical device, type the following form of the ASSIGN command:

```
ASSIGN dev:[(proj,prog)] logical name
```

where dev: is the specification of the physical device and the optional (proj,prog) is a user account (see Section 4.4.4). The logical name can be from one to six alphanumeric characters. A job can have a maximum of four logical name assignments at a time if only device names are specified. However, if one or more of the logical names are associated with an account, the job is limited to three logical name assignments. If you attempt to make more than the legal number of logical assignments, the system prints the error message ?ACCOUNT OR DEVICE IN USE. A logical association is unique to the job and is preserved during CHAIN operations. And because the logical assignment is job-related, it is also preserved when you change accounts. The command does not reserve the device but merely associates the logical and physical names.

### 4.4.1 Associating Multiple Logical Names with One Device

If you make two logical name assignments for the same device, the system recognizes both logical names as belonging to that device. For example:

```
ASSIGN DT1:A  
READY  
  
ASSIGN DT1:B  
READY
```

As a result of these commands, the system associates both logical names A: and B: with DECtape unit 1.

If you associate two different devices with the same logical name, the system replaces the former logical assignment with the latter assignment. For example:

```
ASSIGN DT1:A  
READY  
  
ASSIGN DT2:A  
READY
```

After execution of these commands, the system associates logical name A: with DECTape unit 2.

#### **4.4.2 Associating a Valid Physical Name with a Device**

If you associate a device with a logical name that is also a valid physical device name, the system recognizes the logical, and not the physical, assignment. For example:

```
ASSIGN DTO:DT4
READY
```

The system subsequently associates the logical name DT4: with DECTape unit 0. The system makes this association only for the job that has assigned this logical name. When another job requests DT4:, the system attempts to access the physical device DT4:. Note that if you precede a device designator with an underscore, the system will access that device regardless of any logical name assignment.

#### **4.4.3 Reserving and Releasing a Logically Named Device**

To reserve a device for which a logical name exists, type the ASSIGN command, followed by the logical name and a colon. The command takes the following form:

```
ASSIGN logical name:
```

The system reserves the associated physical device if it is available. Note that the colon is required.

The following commands reserve DECTape unit 1 and associate a logical name with that device:

```
ASSIGN DT1:
READY
ASSIGN DT1:ABC
```

As a result of these commands, a BASIC-PLUS statement of the form OPEN "ABC:FILE.EXT" AS FILE 1 in a subsequently executed BASIC-PLUS program attempts to open FILE.EXT on DECTape unit 1. Also, the subsequent use of ABC: in any system command refers to DECTape unit 1. An attempt to refer to a device by an unassigned logical name generates the error ?NOT A VALID DEVICE.

To release control of the physical device if a logical name is still in effect, type the following form of the DEASSIGN command:

```
DEASSIGN logical name:
```

Note that the colon is required. This command releases control of the physical device associated with the logical name. And any device, of course, can be released by issuing DEASSIGN with a physical name. Neither of these forms of the DEASSIGN command, however, cancels the association between physical device and logical name. To cancel the association between the physical device name and the logical device name, use the following form of the DEASSIGN command:

```
DEASSIGN dev:[(proj,prog)] logical name
```

#### 4.4.4 Associating a Logical Name with a Device and Account

You can use the ASSIGN command to associate a device and a project-programmer number with a logical name. For example, you can associate the logical name LIB (Library Program) with account [2,10] on the RL01 disk cartridge DL2: with the following command:

```
ASSIGN DL2:[2,10]LIB
```

Following this command, you can run a program STAT that is under account [2,10] on DL2: with the command:

```
RUN LIB:STAT
```

This command causes the system to search for the compiled file STAT under account [2,10] on RL01 disk unit 2.

This capability is similar to the system manager (privileged) function described in Section 4.5; except that while the system manager's logical assignment is system-wide, your assignments are local and apply only to your job. That is, you can associate a logical name with a device and account, but other jobs cannot use the logical name to refer to the associated device and account. Other users can, however, make their own logical assignments and may use the same logical name. These local logical name assignments override system-wide logical names, but only for the job which executed the ASSIGN command.

#### NOTE

As described in Section 4.4, you are normally allowed to make a maximum of four logical assignments. However, if you include a project-programmer number in one of those assignments, you are allowed a maximum of three. Also, because of the manner in which the Monitor stores logical assignments, you cannot make an account assignment if four previous logical assignments were made. Your job would have to deassign two of the previous assignments before an account assignment could be made.

## 4.5 System-Wide Logical Names

At the system manager's discretion, a system-wide logical name may be associated with a device or a device and an account on the device. This name, once assigned, may be used by all jobs on the system. For example, a manager associates the logical name CUP: (for Commonly Used Programs) with account [3,4] on the RK05 disk cartridge DK3:. Subsequently, you can execute a program FOO that is under account [3,4] on DK3: by typing the RUN command:

```
RUN CUP:FOO
```

As a result of this command, the system searches for the file FOO.BAC under account [3,4] on RK05 disk unit 3.

The default account associated with a system-wide logical name may be overridden by specifying another account to the right of the logical name. For example, if your account is [200,210] and you wish to run the program OTHER from account [200,240] on disk pack SCRACH: type the following command:

```
RUN SCRACH:[200,240]OTHER
```

As a result, the system searches the disk pack SCRACH: for the file OTHER.BAC under account [200,240] rather than under any account associated with the logical name or under your own account [200,210]. Thus, the account appearing to the right of the logical name overrides the default account.

### NOTE

If the system manager has associated an account with the system-wide logical name, the placement of the device name and account is significant. If, in the preceding example, you were to type:

```
RUN [200,240]SCRACH:OTHER
```

The system would generate the ?ILLEGAL FILENAME error.

On the other hand, the manager may associate a system-wide logical name with a physical device name only, without specifying an account on that device. In this case, the default account is that of the job accessing the device. If, for example, the disk pack DP1: has been assigned the system-wide logical name SCRACH:, type the command:

```
RUN SCRACH:MYFILE
```

to cause the system to look only under your account on DP1: for the program MYFILE.

## 4.6 Disk Access by Pack Identification Label or Logical Name

This section illustrates how a disk may be accessed by one of three types of names: a physical name, a system-wide logical name (e.g., a pack identification label), or a user-assigned logical name.

### 4.6.1 Disk Access by Pack Identification Label

Each disk pack or DECpack cartridge on the system has written on it a pack identification label: for example, MYPACK. This label is not a physical name, because it is independent of which drive unit currently holds the pack. But, as this section explains, you can turn a pack identification label into a temporary system-wide logical name.

In order to access files on the disk, first logically mount the disk pack--that is, establish, on the system, the association between the pack and its identification label, MYPACK. Since this association is established within the Monitor's tables, the label becomes a system-wide logical name for the disk pack. Logical mounting is performed when you specify the pack identification label in the CCL command MOUNT (see Section 19.2.1).

After logically mounting a disk pack or DECpack cartridge, you can refer to it by its pack identification label. For example, to print a directory of the current account on MYPACK, which is logically mounted on drive unit 1, type the command:

```
CAT MYPACK:
```

The label, because it is a system-wide logical name, allows all current jobs to refer to a disk pack or DECpack without concern about its current drive unit number. The name MYPACK is of course temporary, because you can later logically dismount the disk pack or DECpack.

**4.6.1.1 Logically Mounting a Disk by System Command: MOUNT** — Privileged users can logically mount a disk pack during time sharing with the MOUNT command. This command's function is similar to that of the UMOUNT library program. MOUNT, however, is not a CCL command; thus, the UMOUNT program need not be present in the system library in order for the MOUNT command to work. But if the CCL command MOUNT is installed on the system, it takes precedence over the system command MOUNT. (See Section 12.1.) Non-Privileged users must use the CCL command MOUNT to logically mount a disk pack.

To logically mount a disk pack, type the system command MOUNT in the following format:

```
MOUNT dev:pack id[/RONLY]
```

In this format, dev: represents the device designator of the specified disk drive (DK1:, for example). The term pack id represents the pack identification label of the disk pack (MYPACK, for example). If desired, the file specification option /RONLY (Read ONLY) may be included (see Section 11.5.4).

#### **4.6.2 Disk Access by Logical Name: The ASSIGN Command**

The ASSIGN command, followed by 1) the pack identification label of a logically mounted disk pack or DECpack, and 2) any alphanumeric string from one to six characters long, logically associates the alphanumeric string with the pack. In other words, the string becomes a logical name for the pack. The following example illustrates the logical mounting of an RP disk pack and the use of ASSIGN to make the logical association:

```
MOUNT DP1:MYPACK  
READY  
  
ASSIGN MYPACK:ZOOMAR  
READY
```

Following this procedure, you can type the logical name ZOOMAR: to refer to the pack logically mounted on RP disk drive unit 1. Note that ZOOMAR: is a job-local logical name and applies only to one current job, whereas the label MYPACK is a system-wide logical name, and applies to all current jobs.

After the assignment of this job-local logical name, you can refer to a file FILE.EXT on the pack MYPACK (logically mounted on RP drive unit 1) in any of the following four ways:

by physical device name,	DP1:FILE.EXT
by pack identification label,	MYPACK:FILE.EXT
by logical device name,	ZOOMAR:FILE.EXT
or by system-wide logical name	SCRACH:FILE.EXT

The association between the pack identification label and the physical device remains in effect until the pack is logically dismounted by the DISMOUNT command (see Section 19.2.4). The association between the logical name and the physical device remains in effect until the DEASSIGN logical name command is issued, or until the job is logged off the system.

In searching for a specified device, the system follows this procedure: first, it determines if the device name is a job-local (user-assigned) logical name; if it is not, the system determines if it is a system-wide logical name; if not, the system determines if it is a pack identification label; finally, if the device name is none of these, the system assumes that it is a physical device designator.

#### **4.7 Logical Assignment of a User Account**

The ASSIGN command, followed by an account number, establishes a logical association between that account and the commercial at sign (@) character. For example, the command:

```
ASSIGN [100,101]
```

associates the @ character with the account [100,101]. The @ character, therefore, when used in subsequent commands and program statements, refers to account [100,101]. For example, the command DIR @ prints a directory of account [100,101]. Moreover, BASIC-PLUS statements such as:

```
OPEN "[100,101]FILE.EXT" AS FILE 1
```

can be shortened in the following manner:

```
OPEN "@FILE.EXT" AS FILE 1
```

If an account has not been logically assigned, an attempt to refer to it by the @ character generates the error ?ILLEGAL FILE NAME.

To cancel the association between the @ character and the account, type the DEASSIGN command in one of the following two forms:

```
DEASSIGN [100,101]
```

or

```
DEASSIGN @
```

The logical assignment remains in effect until you issue the DEASSIGN @ command or until you log off the system, or until you make another assignment. Because the logical assignment of an account is job-related, it remains in effect even when you change accounts.

## 4.8 Terminal Echo Settings

### 4.8.1 Disabling the Terminal Echo: The TAPE Command

The TAPE command, followed by a carriage return, disables the terminal echo feature while the low-speed reader (located on some terminals) is reading a paper tape into the system. This command stops the terminal from printing what is on the paper tape, and thus avoids meaningless output by the terminal. To use the command, type:

```
TAPE
```

followed by the RETURN key then insert the tape in the low-speed reader and set the reader's control switch to START.

Before giving the TAPE command, you must cause the system to expect the tape input. For example, the sequence:

```
NEW PROG
```

```
READY
```

```
TAPE
```

causes the system to await entry of a source program file from the terminal tape reader. Note that the system does not print READY after the TAPE command. The terminal echo feature is disabled so that the program is not listed on the terminal as it is read. This suppression of printing allows faster input than typing:

NEW PROG

and then making the system read the tape through the low-speed reader. A program listing can be obtained on a line printer or on the terminal at a later time, if necessary.

In tape mode, RUBOUT key characters are ignored. RETURN key characters (that normally have an appended line feed) and LINE FEED key characters (that normally have an appended carriage return) are transmitted. However, because the input tape always contains the proper second character, appended line feeds or carriage returns are removed from transmitted RETURN or LINE FEED key characters.

The TAPE command does not cause suppression of error messages.

#### **4.8.2 Enabling the Terminal Echo: The KEY Command**

Since no characters input from the terminal keyboard or reader are echoed following the TAPE command, the KEY command is provided to reenable the terminal echo feature. You are advised to type the LINE FEED key before issuing the KEY command in case the last line input was not terminated with a carriage return/line feed pair. The command is typed as:

KEY

and entered to the system with the LINE FEED or ESCAPE key.\* (Carriage return characters are not treated as delimiters when the terminal is in TAPE mode.) Note, however, that the KEY command is not echoed at the terminal, because echoing has been disabled. The READY prompt, on the other hand, is echoed and indicates that the terminal echo is once again in operation. Following successful entry of the KEY command, characters are again echo-printed at the terminal.

### **4.9 Input and Output Control Characters**

The control characters described in this section aid in performing input/output operations at the terminal. A character preceded by the word "control" (abbreviated as CTRL) is typed by holding down the CTRL key, typing the character (C, for example), and releasing both keys.

---

\*ESCAPE is shown as ALTMODE on some terminals.

#### **4.9.1 CTRL/C**

Typing a CTRL/C causes RSTS/E to print READY and return to command mode where commands can be given or editing done. CTRL/C stops whatever RSTS/E was doing at the time (execution or output) and returns control of the system to you.

Note that CTRL/C interrupts processing. For example, if CTRL/C is used after the REPLACE command is given and before the READY reply is received, the file is not replaced in its entirety and is not closed. Since the REPLACE is not completed, parts of the program will be lost. Similarly, if the OLD command is issued and a list of error messages is being printed, the CTRL/C key should not be used since the current program is only half compiled at that point.

If your private default run-time system is BASIC-PLUS, type CONT to cause a BASIC-PLUS program to continue execution. Some keyboard output, however, may be lost.

Note that, unless disabled by the system manager, the terminal BREAK key operates as a CTRL/C combination.

#### **4.9.2 CTRL/O**

The CTRL/O combination suppresses output to the terminal until the next time CTRL/O is typed. When a program produces a large amount of output, you may not wish to wait for the printing of the complete information. CTRL/O enables you to monitor the output while not stopping it completely. Typing CTRL/O while output is occurring does not stop the computer's output; the terminal, however, does not print it. The second time CTRL/O is typed, the output is again printed at the terminal. Printing, however, does not resume at the point of the first CTRL/O, but at the point of the second CTRL/O; thus, some output may be skipped in the printing.

Unlike CTRL/O, CTRL/C completely terminates program output. It is useful to think of CTRL/O as a switch, whose first setting creates a condition and whose second setting releases the condition.

#### **4.9.3 CTRL/R**

The CTRL/R combination causes all buffered terminal input to print. For example, if you type a line which contains characters deleted by RUBOUT and you type CTRL/R prior to typing a line terminator, the system retypes the line with deleted characters removed. Thus on hard copy terminals, CTRL/R allows you to examine a corrected line prior to its transmission by a line terminator.

Note that the use of CTRL/R can be enabled or disabled with the TTYSET program.

#### **4.9.4 CTRL/S and CTRL/Q**

These two control characters work together on Cathode Ray Tube display (CRT) terminals. CTRL/S temporarily suspends output to the display terminal. It is used to examine the lines currently displayed before they are replaced on the screen by additional lines. Output can be resumed at the next character by typing the CTRL/Q combination. The CTRL/S and /Q feature is usable only if the terminal has been initially defined with the STALL characteristic (see Section 19.1).

#### **4.9.5 CTRL/Z**

The CTRL/Z combination is used to mark the end of a data file. When data is input from a file, the CTRL/Z character marks the end of recorded data. The message ?END OF FILE ON DEVICE is printed by the system when a CTRL/Z is detected, unless an ON ERROR GOTO statement is used to enable a BASIC-PLUS routine to handle the error.

#### **4.9.6 RETURN Key**

The RETURN key, when typed, echoes as a carriage return/line feed operation on the terminal, as long as the terminal is not in tape mode. The RETURN key is normally used to terminate a line and enter that line to the system. In tape mode (following a TAPE command; see Section 4.8.1), all carriage returns are ignored.

#### **4.9.7 ESCAPE or ALTMODE Key**

The ESCAPE key, like the RETURN key, is used to terminate the current line and causes the line to be entered to the system. The ESCAPE key, however, echoes on the terminal as a \$ character and does not perform a carriage return/line feed. ESCAPE is used to enter the KEY command to the system (see Section 4.8.2).

On some terminals the ESCAPE key is replaced by the ALTMODE key, which performs the same functions.

#### **4.9.8 CTRL/T**

The CTRL/T combination allows you to generate a status report on the current job. When you type CTRL/T, a report in the following format is printed:

33        KB37        NONAME+BAS4F        ^C(OR)    Z(16)K+16K        0,5(+0,5)

where:

33        is the current job number.

KB37        is the keyboard number of this terminal.

NONAME        is the program or operation that is being performed by the current job.

+BAS4F	is the current run-time system.
^C(0R)	is the current state of the job; in this case, the job is in CTRL/C state (keyboard monitor wait on channel 0).
	The possible job states reported by CTRL/T include the following:
ss (ccx)	is returned if the job is in an I/O wait state; where ss is the type of wait, cc is the channel number of the pending operation, and x is R for a read operation and W for a write. This form of job state report is shown in the example.
FP (fff)	is returned if the job is in a file processor wait; where FP represents file processor and fff is one of the following functions:  CLS close. OPN open an existing file. CRE create a file. DLN delete a file. REN rename a file. DIR directory look up. UUO process a UUO directive. ERR look up error message. RST close (reset) channel(s). LOK look up a file. ASS assign a device. DEA deassign a device. DAL deassign all devices. CRT create a temporary file. CRB create a binary (executable) file. RUN open a binary (executable) file for execution. WIN window turn for disk files. EXT extend an open disk file. BYE logout a user. SND send a message. RCV receive a message. NET DECnet function. DSP Monitor function. ??? not decodable.
ss fnn	where ss is the job state and fnn is a swap slot number (if the job is swapped out); this information is returned in SYSTAT format as described in Section 14.1.1.
2	the current program's size (in words) is 2K.
(16)K	the maximum program size (in words) is 16K.
+16K	the size (in words) of the run-time system is 16K.
0.5	the job has used 0.5 seconds of CPU time.
(+0.5)	the amount of elapsed time since the last CTRL/T.

Note that if your system is part of a DECnet network, your system's local node name will appear between the job number and keyboard number.

To have effect, CTRL/T capability must be installed on the system and the TTYSET command SET CTRL/R (see Section 19.1) must be executed.

CTRL/T does not interrupt the job; processing takes place entirely in the terminal service.



## **Chapter 5**

### **Changing Defaults**

#### **5.1 Changing the Default Protection Code: The ASSIGN <> Command**

The ASSIGN command, followed by a protection code in angle brackets <>, changes the default protection code which the system assigns to files created by the current job during time sharing. Usually, this default is <60> when you log into the system. To change it, type ASSIGN followed by the new code enclosed in angle brackets.

The following command, for example, changes the default protection to <40> and assigns that value to all files subsequently created under the job's control.

`ASSIGN <40>`

The default protection remains in effect until you assign another default protection or until you log off the system. Because the default protection is job-related, its assignment remains in effect when you change accounts. When you type a DEASSIGN command with no arguments, the assigned default protection reverts to the system's default. Note that the system always assigns a minimum protection code of <64> when a program is compiled.

#### **5.2 Changing the Magnetic Tape Labelling Default**

The magnetic tape labeling default is set by the system manager, and is system-wide. Though it normally remains in effect during the time-sharing session, it can be changed for an individual job.

RSTS/E supports two types of magnetic tape file labels; DOS and ANSI. DOS labels were first used by the DOS/BATCH-11 operating system. ANSI labels, where used in RSTS/E documentation, refer to the RSTS/E implementation of American National Standard X3.27-1978 – magnetic tape labels and file structure for information interchange.

To change the default, type the ASSIGN command followed by 1) the tape's physical name, and 2) either .DOS or .ANSI. For example:

```
ASSIGN MTO:.DOS  
READY
```

As a result of this command, the system reserves unit 0 for the current job and treats files on unit 0 as having DOS labels. Similarly, to change the default to ANSI labeling, type ASSIGN, the physical device name, and the new default:

```
ASSIGN MTO:.ANSI  
READY
```

Note the importance of the dot (.) character in each example. If it is omitted, the system assigns the logical name DOS or ANSI to the magnetic tape unit. The labeling default remains in effect when the device is reassigned to another job and when you change accounts.

The system program PIP can initialize (i.e., zero) either ANSI or DOS labeled magnetic tape (see Section 16.2.9). A magnetic tape, after being initialized, conforms to the system defaults in format and labeling unless the system default is overridden by a private user default. To specify ANSI labeling, use the .ANSI option in the ASSIGN statement. To specify DOS labeling, use the .DOS option. The ASSIGN statement and option are unnecessary if the magnetic tape is labeled in the system default format.

#### NOTE

If the magnetic tape is labeled in other than the system default and either the .ANSI or the .DOS option is not used, an error message is returned. The error message:

```
?BAD DIRECTORY ON DEVICE
```

appears when a directory listing is attempted. Note that you can use the DIRECT system utility program (see Section 15.1) to obtain a directory listing of the tape provided that it is written with ANSI or DOS labels under RSTS/E.

The DEASSIGN command automatically makes the magnetic tape unit return to the system's labeling default.

### 5.3 Changing the Default Run-Time System for a Job

On systems with multiple run-time systems installed, a job can execute a program that uses any of the auxiliary run-time systems. When the program terminates, it returns to the job's default run-time system.

Each job on the system can have a private default run-time system different from the system default. In such a situation, a job using a different run-time system returns control to the private run-time system default rather than to the system default. If, for instance, the system has two installed run-time systems called BASIC and BAS4F, you can elect to use BAS4F as a private run-time system. The following command in immediate mode changes the run-time system to BAS4F for the execution time of the job (note that this command has effect only if it originates from a BASIC-PLUS Run-Time System):

```
A$=SYS(CHR$(9%)+MID(SYS(CHR$(6%)+CHR$(-10%)+"BAS4F"),7%,4%))
```

### NOTE

If a CHAIN operation (see the *BASIC-PLUS Language Manual*) is executed to a source program, an exit from that chained program can cause the initiating program to return to a run-time system that differs from the job's default. That is, if a source program is chained to, the current run-time system executes OLD and RUN operations (see Sections 7.3 and 7.4) on the source. When the source program terminates, the initiating program exits to the run-time system that performed the OLD and RUN. This run-time system may not be the job's default run-time system. However, if you then run a compiled program, termination of that program returns control to the job's default run-time system.

The command:

```
EXIT
```

returns you to your private default run-time system. Note that if you are currently under the BASIC-PLUS Run-Time System and it is also your default run-time system, an EXIT command clears the current program from memory and sets up a program name of NONAME.

The command:

```
RUN $SWITCH
```

initiates a dialogue which allows you to change the run-time system from the system default to the run-time system of your choice. The dialogue to switch to BASIC-PLUS-2, for example, is as follows:

```
RUN $SWITCH  
RUN-TIME SYSTEM TO SWITCH TO? BP2COM
```

The system now calls the BASIC-PLUS-2 Run-Time System to run your job. A carriage return typed in response to the above query causes the job to return to the system default run-time system.

The system manager can install the CCL command:

**SWITCH**

to invoke the SWITCH program. In this case, type:

**SW BPZCOM**

to change the current run-time system to BASIC-PLUS-2. Typing the CCL command alone causes the system to revert to the system default run-time system.

### **5.3.1 SWITCH Program Error Messages**

The following error messages are generated by the SWITCH program.

#### **Message and Meaning**

##### **?ILLEGAL RUN-TIME SYSTEM NAME**

The run-time system name is misspelled or incorrect. Repeat the request to change the run-time system.

##### **?NO KEYBOARD MONITOR**

There is no keyboard monitor in the run-time system requested. The run-time system is inaccessible.

##### **?NO RUN-TIME SYSTEM**

The run-time system requested is not installed on this system.

## **Chapter 6**

# **Functions of the BASIC-PLUS System Commands**

This part of the User's Guide describes the BASIC-PLUS system commands, special characters, and system features such as EXTEND/NO EXTEND format and immediate mode. Most of the commands described in this part do not belong to the BASIC-PLUS language itself, but are nonetheless needed by the BASIC-PLUS user. They enable you to write, run, edit, save, and debug--that is, test and correct--programs. They comprise, in short, the support system for the BASIC-PLUS language. It is assumed that you know something of the BASIC-PLUS language--enough, at least, to write some short programs. If you need to learn more of the language refer to the *BASIC-PLUS Language Manual*.

In addition to BASIC-PLUS, RSTS/E supports other optional software such as BASIC-PLUS-2, FORTRAN, and COBOL. The RSTS/E support system for a particular option is described in the RSTS/E User's Guide specific to that software.

### **6.1 Some Definitions for the New BASIC-PLUS User**

The following sections contain explanations of the terms used in Part III.

#### **6.1.1 Source and Compiled Programs**

Most of the commands and operations described in Part III are designed to affect and manipulate a source program. Briefly, a source program is any program which can be translated by BASIC-PLUS, is stored on disk in ASCII (or human readable) format, and is available to you for listing (printing at the terminal) and for modification (editing and debugging). Source programs are stored with .BAS file extensions, and are sometimes called BAS programs.

A source program is distinguished from a compiled program, which has already been translated by BASIC-PLUS, and has been stored in its translated form. The translated BASIC-PLUS code is called intermediate code. It is not human readable and must be executed by the BASIC-PLUS Run-Time System. This type of program is not available to you for listing and modification, but only for execution. For this reason, compiled programs are sometimes called run-only programs. Compiled programs are stored with .BAC file extensions, and are sometimes called BAC programs.

### 6.1.2 The Program Currently in Memory

Often in Part III, a BASIC-PLUS program is described as "currently in memory" or as "the current program." These phrases both refer to a program that you are currently writing, editing, listing, running, or debugging. This "current program" may be a new one that you created during the present time-sharing session, or it may be an "old" program that you retrieved from your storage area. Similarly, it may be a source program or a compiled program. The important point to remember is that it is the program with which you are now working.

Unless otherwise noted in an individual command description, the BASIC-PLUS system commands do not alter the current program.

## 6.2 A Guide to the BASIC-PLUS System Commands

Table 6-1 is an overview of the BASIC-PLUS system commands, features, and special characters, and to their locations within Part III.

Table 6-1: A Guide to the BASIC-PLUS System Commands

Effect on Program	Command/Feature	Section
Calling old program Calling NONAME	OLD	7.3
Compiling, description of	COMPILE	7.4.3.1
Creating and Naming	NEW NEW filename	7.1.1 7.1.1
Creating NONAME	NEW	7.1.1.1
Debugging halting execution	STOP CTRL/C PRINT LINE	9.2.1 9.3.1 9.3.1
continuing execution	CONT CCONT(privileged)	9.2.2 9.2.3
suppressing/continuing output	CTRL/O CTRL/S,CTRL/Q	9.3.2 9.3.3

(continued on next page)

**Table 6-1: A Guide to the BASIC-PLUS System Commands (Cont.)**

Effect on Program	Command/Feature	Section
Deleting entire contents specific lines segments  program from disk program from private dev:	DELETE  UNSAVE KILL	8.1.2  8.1.4.1 8.1.4.2
Device for storage specifying running from private removing from	SAVE dev: RUN dev: UNSAVE	7.2.2 7.4.2 8.1.4
Directory (catalog) printing at terminal	CATALOG	7.9
EXTEND/NO EXTEND descriptions changing formats	EXTEND/NO EXTEND	8.2.1
Formatting lines (see also EXTEND above) multiple statements, 1 line 1 statement, multiple lines spacing	: or \ Line Feed space/TAB	8.2.2.1 8.2.2.2 8.2.2.3
Immediate mode execution		9.1
Length of program finding current finding maximum	LENGTH	7.8
Line printer obtaining output	SAVE LP:	7.2.3
Listing at terminal whole program specific lines segments  without header	LIST  LISTNH	8.1.1  8.1.1
Listing at line printer	SAVE LP:	7.2.3
Merging programs	APPEND	8.1.5
Paper tape output	SAVE PP:	7.2.3
Protection code changing adding to compiled	NAME AS COMPILE <prot>	7.7.2 7.4.3.2
Renaming current program disk/DECtape file	SAVE filename RENAME NAME AS	7.2.1 7.5 7.7.1
Replacing saved program	REPLACE	7.6

(continued on next page)

**Table 6-1: A Guide to the BASIC-PLUS System Commands (Cont.)**

Effect on Program	Command/Feature	Section
Running current program old program omitting header from private dev:	RUN RUNNH RUN dev:	7.4.1 7.4.1 7.4.2
Saving compiled version compiled version, renamed	SAVE COMPILE	7.2 7.4.3.2
Scaled arithmetic	SCALE	7.10
Stopping execution (see Debugging)		
Transferring control	CHAIN	8.1.6
Writing new program	NEW	7.1.1

# **Chapter 7**

## **Creating and Running a BASIC-PLUS Program**

### **7.1 Writing the Program**

#### **7.1.1 The NEW Command**

The NEW command allows you to name and to create a new program. To issue this command, type:

NEW

and press the RETURN key. The system responds by printing the following request for the new program's name:

NEW FILE NAME--

You then type the new program's filename, without an extension. (The system does not require an extension at this point because the SAVE and COMPILE commands, described in Sections 7.2 and 7.4.3, automatically append one.)

Another way to issue the NEW command is to type NEW, then the new program's name, then the RETURN key. Thus, you avoid the NEW FILE NAME-- prompt. For example, the command:

NEW CASH01

is equivalent to the following sequence:

```
NEW  
NEW FILE NAME-- CASH01
```

When the NEW command is issued, it has the following effects in your memory area:

1. It deletes any program currently in memory.
2. It causes RSTS/E to remember the new program's name.

#### NOTE

A command of the following form is meaningless:

```
NEW DTO:STAT
```

This command is meaningless because new programs can only be input from the user terminal. To input programs from other devices, the OLD command must be used.

When you name a file in the NEW command, the system does not check for a file of the same name. This check occurs when you give the SAVE command.

Whenever you create a program (with the NEW command) or call an existing program (with the OLD command, see Section 7.3), the system creates the file TEMPnn.TMP in your area on the public structure; the nn represents the current job number. TEMPnn.TMP contains the ASCII text of the source program, and any revisions you make to that text. As its name indicates, this file is temporary and is deleted when you log off the system. TEMPnn.TMP is used by the BASIC-PLUS system, as a "scratch" file; you are normally unaware of this file, except in so far as it occupies space in your area, and reflects the size of the current program.

**7.1.1.1 Creating a NONAME File** — Instead of specifying a filename in response to the prompt NEW FILE NAME--, you can merely press the RETURN key. This action creates a file called NONAME. Later NONAME can be saved or compiled, and referenced as NONAME.BAS or NONAME.BAC. You can change this name at any time (see Sections 7.5 and 7.7.1). The following example shows the creation of the file NONAME (the RETURN key, although typed, does not echo):

```
NEW  
NEW FILE NAME-- READY  
READY
```

If you issue a SAVE at this point, the file NONAME.BAS is created.

#### NOTE

NONAME, although a legal filename, should not be used for a program that you wish to save. Any user logged into the same account can create a NONAME file. Only the latest version of the file is saved. Therefore, NONAME should be used only for programs which you intend to run once.

### 7.1.2 Input of the New Program

Once you create a new file in memory with the NEW command, you can type the program into the system. Or, if your terminal has a low-speed reader, you can use the reader to input a program from a pre-punched paper tape as described in Section 4.8.1.

If you type your program into the system, you are likely to use the RSTS/E editing features, described in Chapter 8. At this point, however, it is useful to describe two simple editing tools: RUBOUT and CTRL/U.

Should you make a typographical error, you can erase the incorrect characters by typing the RUBOUT key (on some terminals, the DELETE key) once for each character to be erased. As each character is erased in this manner on a hard-copy terminal, it is echoed between backslashes \\\. After erasure, type the correct characters on the same line.

On a video (CRT) terminal, typing the RUBOUT/DELETE key moves the cursor back one space, thus deleting the erroneous character. Neither the character nor any backslashes appear.

If you wish to delete the current line entirely, type the CTRL/U combination: that is, hold down the CONTROL key while typing U. As a result of this action, the system deletes the entire current line, and performs a carriage return/line feed. You can then retype the line.

## 7.2 Saving the Program: The SAVE Command

The primary function of the SAVE command is to store a current BASIC-PLUS source program on the public disk structure. To be saved, the source program must be currently in memory. And after the program has been saved, it remains in memory and therefore can be run, changed, or deleted.

To store the current program under your account on the public structure, type:

SAVE

then press the RETURN key. As a result, the program currently in memory is saved on the public structure under the current account, with its current filename and the extension .BAS. If a file of the same name exists, the system prints the error message:

?FILE EXISTS-RENAME/REPLACE

This message tells you that an identically named file exists in your area and warns you against unintentionally destroying that file. Should you wish to destroy it by replacing it with the current program, issue a REPLACE command (described in Section 7.6). But if you have no desire to destroy the identically named program, use the SAVE command to save the current program under a different filename, a procedure described in Section 7.2.1.

### **7.2.1 Saving the Current Program under a Different Name**

If for any reason the filename of the current program is not the one you desire, save the program under a different name by issuing the SAVE command in the following form:

```
SAVE NEWNAM
```

This command saves the program currently in memory on the public structure under the name NEWNAM.BAS. When writing the file to any storage device, the SAVE command appends the .BAS extension by default.

You can specify an extension other than the default .BAS by including a filename and extension in the SAVE command. The following example illustrates:

```
SAVE NEWNAM.EGG
```

As a result of this command, the current program is saved on the public structure as NEWNAM.E66.

The SAVE command may be used with a complete file specification of the form:

```
dev:[acct]filename.ext<prot>/option(s)
```

See Chapter 11 for a description of the complete file specification.

### **7.2.2 Using SAVE to Specify a Storage Device**

You may wish to save the current program, renamed or not, on a storage device other than the public structure. To store the program, under its current filename or a new filename, type a SAVE command in one of the following forms:

SAVE dev: which stores the program on the device specified as dev:, or

SAVE dev:NEWNAM which stores the program as NEWNAM.BAS on the device specified as dev:.

The following command, for example, saves, on DECtape unit 0, a copy of the program currently in memory. The program is saved under the filename ARCH.BAS.

```
SAVE DTO:ARCH
```

### **7.2.3 Using SAVE to Obtain Line Printer and Paper Tape Output**

To obtain a line printer listing of the current program, type:

```
SAVE LP:
```

To punch a paper tape of the current program on the high-speed punch, type:

SAVE PP:

Because both devices are output-only and non-file structured, you need not specify a filename.

#### NOTE

To punch a tape on the low-speed punch (on the ASR 33 terminal) you can issue a LISTNH command, and, before pressing RETURN, turn the punch on line. This procedure is not recommended, however, because the word READY is punched at the end of the tape.

Tapes punched on the low-speed punch should be read only through the low-speed reader.

### 7.3 Calling an Existing Program: The OLD Command

The OLD command allows you to retrieve the source file of a previously saved BASIC-PLUS program; OLD causes any program currently in memory to be overwritten. This command is used to retrieve source programs only (.BAS files by default), since compiled programs (.BAC files) can be run but not changed. Thus, you can edit any program called with the OLD command at the terminal.

BASIC-PLUS source programs can be given extensions other than the default .BAS.

To issue the OLD command, type:

OLD

and press the RETURN key. The system then prints the following request for the program's filename:

OLD FILE NAME--

You then type the filename of the saved program. Or, you can avoid the prompt OLD FILE NAME-- by typing the old filename on the same line as the command, as follows:

OLD TAXES

This command calls the saved file TAXES.BAS from the public structure. If the file is unavailable or protected against you, an appropriate message is printed.

If the file has an extension other than .BAS, you must specify that extension, as in the following example:

OLD TAXES.TL1

If you do not respond to the OLD FILE NAME-- prompt with a filename, but press RETURN instead, RSTS/E looks for the file NONAME (which you or the system may have created; see Section 7.1.1.1). The following example retrieves the file NONAME:

```
OLD  
OLD FILE NAME-- RET  
READY
```

As a result of this procedure, whatever was stored in the file NONAME.BAS on the public structure under your account is now in memory and available to you.

The OLD command may be used with a complete file specification of the form:

```
dev:[acct]filename.ext/option(s)
```

See Chapter 11 for a description of the complete file specification.

## 7.4 Running and Compiling Programs: The RUN and COMPILE Commands

### 7.4.1 Running a Program from the Public Structure: The RUN Command

The RUN command is used to execute any BASIC-PLUS source or compiled program. (Source programs are stored as you typed them; compiled programs are described in Section 7.4.3.)

In order to identify and run the program that is currently in memory, type:

```
RUN
```

This command not only runs (executes) the current program, but also prints, before execution, a program header consisting of the program's name and the current (system) date and time. If you do not require this information, type the RUNNH (No Header) command:

```
RUNNH
```

which executes the current program, but does not print the header material. The RUNNH command runs only the program currently in memory; any filename or characters specified with this command are ignored.

To execute an old program--one not currently in memory--type the RUN command and the old program's filename, in the following form:

```
RUN PROG35
```

This command causes RSTS/E to search the public structure for the file PROG35.BAC or PROG35.BAS, and then to load it, to compile it if necessary, and to run it. Any current program is overwritten. This procedure, of course, assumes that the file can be found.

#### NOTE

Because blanks are not significant in BASIC-PLUS NO EXTEND mode commands, the RUN filename command does not execute properly if the filename begins with NH. RUN NHTAX, for example, is the same as a RUNNH command. It will therefore run the program currently in memory. In the EXTEND mode of BASIC-PLUS, however, which does treat blanks as significant, RUN NHTAX causes BASIC-PLUS to execute the program NHTAX. (See Section 8.2.1 for a description of EXTEND mode.)

If the source file PROG35.BAS and the compiled file PROG35.BAC both exist, RSTS/E loads and executes PROG35.BAC--because it is already compiled and uses less time. Note that because the program loaded is the compiled version, it can only be run--not edited or modified. If, however, you wish to execute PROG35.BAS, you first retrieve it by issuing the OLD command, then execute it by issuing the RUN command; or, you can type the command RUN PROG35.BAS. After the program has been retrieved by OLD, it is currently in memory and may be edited or modified.

To continue with this example, assume that the compiled program PROG35.BAC does not exist. Only the source program PROG35.BAS exists. In this case, the command:

RUN PROG35

loads, compiles, and executes PROG35.BAS, the source program.

The RUN command may be used with a complete file specification of the form:

dev:[acct]filename.ext/option(s)

See Chapter 11 for a description of the complete file specification.

#### 7.4.2 Running Programs from Private or Specific Devices: The RUN dev: Command

To run a program stored on a device other than the public structure, or stored on a particular device in the public structure, type RUN, the device specification, and the program's filename. The form is as follows:

RUN dev:FILNAM

The following command, for example, runs the file TRANS.BAC or TRANS.BAS, which resides on DECpack unit 1:

```
RUN DK1:TRANS
```

To read a source program from the high-speed paper tape reader and run that program, type the command:

```
RUN PR:
```

Because the reader PR: is an input-only, non-file structured device, you need not specify a filename.

If, on the other hand, you specify a filename with a non-file structured device in the RUN command, that filename is used as the current program name when the program is read into memory. In the following sequence, for example, you issue a command to run a program from the high-speed reader, name the program SALE, and receive its output:

```
RUN PR:SALE
AVERAGE SALE FOR JULY: 77.26
READY
```

If, however, you do not specify a filename, the program in memory will have no name. It is important to remember, in this case, that a simple SAVE--or any other command without a filename--cannot be applied to the program in memory unless that file has a name (see the RENAME command, Section 7.5). In the following sequence, you run a program from the high speed reader, but--unlike the previous example--you do not name the program in the RUN command. Thus, when you attempt to save the program, you receive an error message:

```
RUN PR:
66 ELECTORAL VOTES STILL NEEDED TO NOMINATE
READY
SAVE
?ILLEGAL FILE NAME
READY
```

In order to save this program, you must give it a filename in the SAVE command or name it by another method--the RENAME command, for example (see Section 7.5).

### 7.4.3 The COMPILE Command

**7.4.3.1 The Purpose of COMPILE** — Normally, BASIC-PLUS accepts each line of a program as you enter it; if a line is syntactically correct, BASIC-PLUS then translates it into a form that is understood by the system. This translation is called compiling of the program.

When you edit the program, only the changed lines are compiled-- that is, translated. And when you give the SAVE command, only the source version of the program (i.e., text that is typed in response to the LIST command) is stored in the .BAS file created. Thus, when you give the OLD command, the BASIC-PLUS system, in response, must first read the text of the saved program, and then must compile it--translate it--line by line again, just as it did when you originally entered the program from the keyboard.

This process of compiling a saved program every time it is brought into memory may consume considerable system time. To avoid such repeated compilation, issue the COMPILE command. COMPILE permits you to save an image of your compiled program, rather than (or in addition to) the source text of the program. This compiled version, stored with the default extension .BAC, can be read from the disk with a minimum of system time.

#### NOTE

Compiled files have a minimum size requirement of 7 blocks. This size may be greater than necessary to actually store the compiled (or even the source) version of a short program. In such cases, be aware that you are trading disk space for execution speed.

Because of the transformation that occurs when a program is compiled, a .BAC file can only be executed; it cannot be edited or modified. Therefore, a compiled program cannot be retrieved by the OLD command, whose function is to make a program available for editing. A compiled program must be called by the RUN command, which merely executes it.

**7.4.3.2 Using COMPILE** — If the program's current filename (i.e., the name printed in the header) is ACT01, then the command:

COMPILE

saves the compiled program under filename ACT01.BAC on the public structure under your account. This command does not alter or replace ACT01.BAS.

If you desire another name for the compiled program, you may specify it in a command of the following form:

COMPILE SCENE2

The preceding command creates, on the public structure, a compiled program named SCENE2.BAC.

The COMPILE command:

1. outputs compiled programs only to the disk structure,
2. appends the extension .BAC (unless otherwise specified) to the current or specified filename for storage in your public disk area, and

3. stores compiled programs with a default file protection of <124>; this default consists of the compiled file protection code <64> plus the system-assigned default code <60>. (Specifying another protection is discussed in the following paragraph.) Note that the system default may be a code other than <60>; in either case, <64> is added when COMPILE is executed.

If you wish to add, to the compiled code <64>, a code other than the system default <60>, specify the additional code in a COMPILE command of the following form:

```
COMPILE <40>
```

This command creates, on the public structure, a file with the current filename, the extension .BAC, and the assigned protection <104>. This protection consists of the compiled code <64> and the user-specified code <40>.

At any time, a protection code can be altered by the BASIC-PLUS statement NAME AS (see Section 7.7), or by the renaming facility of the PIP system library program (see Chapter 16). However, only a privileged programmer can change a compiled <64> protection code.

The COMPILE command may be used with a complete file specification of the form:

```
dev:[acct]filename.ext<prot>/option(s)
```

See Chapter 11 for a description of the complete file specification.

**7.4.3.3 Compiling a Program on a Specific Disk Pack** — To compile the current program--under its current filename or a new filename--on a specific disk pack, type a COMPILE command in one of the following forms:

COMPILE dev:                   which compiles the program, under its current filename, on the disk specified as dev:, or

COMPILE dev:NEWNAM           which compiles the program as NEWNAM.BAC on the disk pack specified as dev:.

Note that in either case the specified device must be a disk pack.

The following command, for example, compiles the program currently in memory on disk pack DK2:. The program is compiled as TELL.BAC.

```
COMPILE DK2:TELL.BAC
```

## 7.5 Renaming the Current Program: The RENAME Command

The RENAME command allows you to change the name of the program currently in memory. To issue this command, type RENAME, the new name for the program, and the RETURN key in the following form:

```
RENAME NEWNAM
```

As a result of this command, the old name of the program currently in memory is discarded; the current program is now known by the name NEWNAM. If you now issue a SAVE command, NEWNAM.BAS is stored on the public structure.

## 7.6 Replacing a Saved Program: The REPLACE Command

The REPLACE command outputs the program currently in memory to the public structure or anywhere else, where it replaces a program with the same filename. REPLACE performs the same function as SAVE, but destroys any existing program of the same name. You can issue REPLACE alone or with a filename, in one of the following forms:

```
REPLACE  
or  
REPLACE FILNAM
```

REPLACE appends the extension .BAS to the filename already associated with the program or specified in the command. If you specify a filename, REPLACE assigns that name to the new--or replacement--version of the program. In such a case, the current name of the program in memory is ignored.

The REPLACE command may be used with a complete file specification of the form:

```
dev:[acct]filename.ext<prot>/option(s)
```

See Chapter 11 for a description of the complete file specification.

## 7.7 Changing a Program's File Specification: The NAME AS Statement

### 7.7.1 Using NAME AS to Rename a Program

NAME AS is a BASIC-PLUS statement that can be used in immediate mode to rename and/or assign a new protection code to a disk or DECTape file. Type this statement in the following format:

```
NAME [string] AS [string]
```

The specified file, the first [string] indicated, is renamed as the second [string] indicated.

If the file resides on a device other than the public disk structure, that device must be specified in the first string; you can also specify it in the second string. The NAME AS statement, however, does not copy a file from one device or account to another device or account. Therefore, if a device name is used in the second [string], it must be the same as that specified in the first [string]. NAME AS assumes no default filename extensions; if the file to be renamed has an extension, each of the two strings must include an extension.

The following two statements, for example, are equivalent:

```
NAME "DTO:OLD.BAS" AS "NEW.BAS"  
and  
NAME "DTO:OLD.BAS" AS "DTO:NEW.BAS"
```

Note, however, that since the NAME AS statement cannot perform device transfers, there is no need to mention DT0: twice; that is, the device of the new filename need not be specified. (For information on device transfers, see Chapter 16.)

To change or create only the extension of a disk file on the public structure, type a statement of the following form:

```
NAME "ARC" AS "ARC.01"
```

This statement changes only the extension of the disk file ARC (with no extension) to .01.

The following statement, however, is not advised because of the absence of an extension in the second string:

```
NAME "FILE1.BAS" AS "FILE2"
```

As a result of this statement, the renamed file will have a null extension: FILE2.

### 7.7.2 Using NAME AS to Change a Protection Code

You can change a program's file protection code by specifying the new code, within angle brackets <>, as part of the second string. If specified, this protection code becomes that of the renamed file. If you do not specify a protection code, the old protection code is retained.

If you wish to keep the program's present filename, and change only its protection code, type a NAME AS statement in the following form:

```
NAME "YORK.BAS" AS "YORK.BAS<40>"
```

This statement changes only the protection code of the program YORK.BAS stored on the public structure.

#### NOTE

Only privileged users can use NAME AS to assign a protection code higher than <63>.

## 7.8 Finding a Program's Current and Maximum Length: The LENGTH Command

The LENGTH command prints, at your terminal, a message containing the current program's length, and, in parentheses, the maximum length that the system allows for the program. To issue this command, type:

```
LENGTH
```

and press the RETURN key. For example:

```
LENGTH  
5 (16) K OF MEMORY USED
```

In this example, you are informed that the current program is 5K words long, and that its maximum is 16K words. The present length of the program is reported to the next highest 1K increment. ("K" is a common abbreviation for 1024.)

## 7.9 Listing Device Directories: The CATALOG Command

The CATALOG or CAT command allows you to list all the files in a device directory, under your account or that of another user, or in the system library. Note that the system manager has the option of restricting the use of CATALOG to a listing of your own directory.

To list your own public disk structure directory, type a simple CATALOG or CAT command, followed by a carriage return. In response, the system prints the directory at your terminal. For example:

```
CAT  
AVERAG.BAS    2     60   28-Oct-78 28-Oct-78 02:37 PM  
PERCNT.BAS    2     60   28-Oct-78 28-Oct-78 02:37 PM  
TERM .DOC     11    60   28-Oct-78 28-Oct-78 02:37 PM  
REPDLO.TXT    43    60   28-Oct-78 28-Oct-78 02:38 PM  
PHONE .DIR    43    60   28-Oct-78 28-Oct-78 02:39 PM  
EXPENS.BAS    29    60   28-Oct-78 28-Oct-78 02:39 PM  
CHOICE.BAS    2     60   28-Oct-78 28-Oct-78 02:40 PM  
SOURCE.DAT    20    16   28-Oct-78 28-Oct-78 02:41 PM  
BACKUP.CMD    1     60   28-Oct-78 28-Oct-78 02:42 PM  
XX .TST       1     60   28-Oct-78 28-Oct-78 02:42 PM  
VISITS.LST    14    40   28-Oct-78 28-Oct-78 02:45 PM  
TEMP24.TMP    0     60   28-Oct-78 28-Oct-78 03:01 PM
```

Ready

The directory printed with the CATALOG command lists filenames and extensions, the size of the file, protection code, date of creation, and the date and time of last access.

To obtain a catalog of files stored under another user's account number on the public structure, issue a command in this form:

```
CATALOG [100,102]
```

This command lists the public structure files owned by user account [100,102].

To obtain a listing of all files in the system library account, type:

```
CATALOG $
```

or

```
CAT $
```

In either command, \$ indicates account [1,2], the system library.

To obtain a catalog of files on a device other than the public structure, issue the command:

```
CATALOG dev:
```

where dev: is a device specification (a user account specification may be included in such a command). For example:

```
CAT DT1:
```

lists all files on DECTape unit 1 (no account numbers are associated with DECTape files).

To list all files stored under a specific account on a non-disk device, type a command in this form:

```
CATALOG MT1: [200,220]
```

This command lists all files stored under account [200,220] on magnetic tape unit 1.

## 7.10 Using Scaled Arithmetic: The SCALE Command

The SCALE command implements and controls the scaled arithmetic features of BASIC-PLUS. This feature is used to overcome accumulated round-off and truncation errors in fractional computations performed on systems using the floating-point format. The feature enables the system to maintain decimal accuracy of fractional computations to a given number of places determined by a scale factor. Scaled arithmetic is described in the *BASIC-PLUS Language Manual*.

Users on a system with the double-precision, four word floating-point format may issue the SCALE command to control the scale factor. An attempt to use the SCALE command on systems with the single-precision floating-point format produces the ?MISSING SPECIAL FEATURE error message.

### NOTE

The SCALE factor you specify is a parameter that is unique to your BASIC-PLUS program. If a non-BASIC-PLUS program is executed, the SCALE factor is not preserved. For example, if a system program (such as PIP) or another language compiler (such as COBOL or BASIC-PLUS-2) is executed, the SCALE factor you specify is not preserved. Also, if there are two or more BASIC-PLUS run-time systems or a system program running under another run-time system (such as RSX), the SCALE factor may not be preserved.

To specify the scale factor to be used, type the SCALE command with a decimal integer between 0 and 6. For example:

```
SCALE 2
```

```
READY
```

The SCALE 2 command sets the current scale factor to 2. Subsequently, all programs compiled for that job have a scale factor of 2. If an invalid scale factor is typed with the command, the system prints the ?SYNTAX ERROR message.

#### NOTE

SCALE is a command and cannot be executed as a BASIC-PLUS statement. Also, a program cannot refer to or modify the scale factor.

If you type a value of 0 with the SCALE command, it disables the scaled arithmetic feature. For example:

```
SCALE 0
```

```
READY
```

The SCALE 0 command sets the scale factor to 0. Programs compiled for that job subsequently treat all floating-point calculations in the standard fashion (decimal places are not shifted prior to calculation). When you log a job onto the system, the initial scale factor is 0.

To determine the current scale value, type the SCALE command without a value. For example:

```
SCALE
```

```
6
```

```
READY
```

The system prints the current value (6) and the READY message. If the program in memory has a value set other than the current value, the system prints two values. For example:

```
SCALE
```

```
6,2
```

```
READY
```

The first value (6) indicates the current value for the job, and the second value (2) is the one set for the program currently in memory. If the scaled arithmetic option is currently disabled, the system prints a zero as the first value.

When you load source statements into memory, whether by an OLD command, a RUN command to a .BAS file, or by entering statements after a NEW command, the system sets the scale factor for the program in memory to the current scale factor.

```
SCALE 4
READY
OLD TEST
READY

LISTNH
10      A$ = "   "
20      X = ,12345
30      PRINT USING A$,X
40      END

READY

RUNNH
0,12340

READY
```

If you execute a RUN command for the program in memory, the system performs floating-point calculations using the scale factor of the program currently in memory. Similarly, the system uses the scale factor of the program in memory when executing immediate mode statements.

After loading source statements into memory, you cannot change the scale factor for the program in memory. This action prevents you from possibly executing floating-point calculations for a program, parts of which the system assigned a different scale value.

An attempt to execute such a program or source statement causes the %SCALE FACTOR INTERLOCK warning message. For example:

```
SCALE
4
READY
OLD TEST
READY

SCALE 6
READY

RUNNH
%SCALE FACTOR INTERLOCK
0,12340

READY
```

#### NOTE

The %SCALE FACTOR INTERLOCK message may not appear, since the system manager may have suppressed its printing. On such systems, the message does not appear but the operation of the scale factor remains unchanged.

The program executes using the scale factor of the program in memory (4) rather than the current scale factor (6). To execute such a program with a changed scale value, type a SAVE or REPLACE command followed by an OLD or RUN command for the related file. For example:

```
REPLACE TEST  
READY  
OLD TEST  
READY  
RUNNH  
0.12345  
READY  
SCALE  
6  
READY
```

The system consequently executes the program with the changed scale factor 6.

The following example illustrates the effect of different scale values for the sample program TEST.BAS, listed earlier in this section.

```
SCALE 2  
READY  
OLD TEST  
READY  
RUNNH  
0.12000  
READY  
SCALE 4  
READY  
OLD TEST  
READY  
RUNNH  
0.12340  
READY
```

The system loads the source file using the current scale factor 2. When executed, the program generates results to 2 decimal places, and truncates the remaining places. If you change the current scale factor and load the same program, the system executes the program with changed scale factor and truncates the remaining places.

If you store the compiled file and later run it with a different scale value in effect, the following sequence takes place:

```
SCALE 6
READY
OLD TEST
READY
COMPILE TEST
READY
SCALE 4
READY
RUNNH TEST
0.12345
READY
SCALE
4,6
READY
```

The system loads the program TEST and executes it with the scale value (6) set when you compiled it. The system does not use the current value (4) but rather the value of the compiled program (6). This action occurs whether the program executes because of a RUN command or because of a CHAIN command.

If a compiled program is currently in memory, you cannot change its scale factor and run that program. Since it is impossible to alter the scale factor of a compiled program except by compiling the program again, the system generates the %SCALE FACTOR INTERLOCK warning message (see NOTE earlier in this section). For example:

```
SCALE
4,6
READY
RUNNH
%SCALE FACTOR INTERLOCK
0.12345
READY
```

The program executes with the scale factor of the compiled program (6).

# **Chapter 8**

## **Editing and Modifying a Program**

### **8.1 Editing and Modifying the Program**

#### **8.1.1 Printing the Program: The LIST Command**

The LIST command prints, at your terminal, a clean copy of all or part of the current program. The program must be in memory before it can be printed by the LIST command. If you wish to print a source program that is not in memory, issue an OLD command, then a LIST command as described in this section. LIST is especially useful during and after an editing session in which you change the current program.

To print a complete copy of the current program as it exists in memory, type:

LIST

When the system prints the whole program, it lists source lines in ascending line number sequence--regardless of the order in which you entered them.

To print a single line, type LIST followed by the line number, as follows:

LIST 100

This command prints a copy of line 100.

To print a specified section of the program, type LIST followed by the number of that section's first line, a hyphen, and the number of that section's last line:

LIST 100-200

This command lists all program lines from 100 to 200, inclusive. If 100 and 200 do not exist in the program, any lines within the range 100 to 200 are listed. This form of the command is especially useful with a video terminal, because it prevents long programs from scrolling off the screen.

One or more single lines or program sections may be specified in a single LIST command by separating the individual elements with commas. For example, the command:

```
LIST 25,34,60-85,95,200-220
```

prints lines 25, 34, and 95 along with the program lines between (and including) 60 and 85 and between (and including) 200 and 220. Lines or program sections need not be indicated in sequential order in the LIST command; they are printed out in the requested order.

The system, in response to a LIST command, prints a program header containing the program name and the current system date and time. If you do not desire this information (and during a normal editing session, you may not), issue the LISTNH command to avoid printing of the header.

In executing any form of the LIST or LISTNH command, BASIC-PLUS prints the ? character at the left of a line it considers to be in error; for example:

```
LISTNH
10      LET A,B=25
?20      PPRINT A+B
        :
        :
READY
```

The following is a summary of the forms of LIST and LISTNH:

LIST Command	Meaning
LIST	List the entire program as it currently exists.
LISTNH	Same as LIST, but omit program header.
LIST n	List line n.
LISTNH n	List line n without the program header.
LIST m,n,o	List lines m, n, and o.
LISTNH m,n,o	List lines m, n, and o without the program header.
LIST n1-n2	List lines n1 to n2, inclusive.
LISTNH n1-n2	List lines n1 to n2, inclusive, without the program header.

Extensive examples of program listings appear in the *BASIC-PLUS Language Manual*.

#### NOTE

LIST sends output to your terminal only. If a line printer is available, the command SAVE LP: is the fastest way to obtain a printed copy of the program (see Section 7.2.3).

### **8.1.2 Deleting Lines: The DELETE Command**

The DELETE command deletes one or more specified lines from the current program; typing a simple DELETE, without line numbers, deletes all lines from the program.

To delete one line from the program, type DELETE followed by that line's number in the following form:

```
DELETE 100
```

This command deletes line 100. For convenience, however, you can type only the number of the line to be deleted and the RETURN key. For example:

```
15
```

is equivalent to DELETE 15 and deletes line 15.

To delete a section of the program, type a command of the following form:

```
DELETE 100-200
```

This command deletes all lines between and including lines 100 and 200. If 100 and/or 200 do not exist in the program, any lines within the range 100 to 200 are deleted.

If you wish to delete several lines or groups of lines, specify them in one DELETE command, separating the elements with commas as follows:

```
DELETE 100-200,255,300-400,470,1000-1100,475
```

This command deletes all lines from 100 to 200, line 255, all lines from 300 to 400, lines 470 and 475, and all lines from 1000 to 1100. Note that lines or sections need not be listed in sequential order in the DELETE command.

**8.1.2.1 Cautionary Notes about DELETE** — Remember that typing DELETE without a line number deletes all lines from the program.

Before deleting any line, check for other lines containing references to the line you intend to delete--GOTO statements, for example. Obviously, such lines must be replaced or modified to reflect deletions of the lines that they reference. When the program is run, a reference to a missing line number will generate the error message ?STATEMENT NOT FOUND and halt the program's execution.

### **8.1.3 Simple Erasures: The RUBOUT Key and CTRL/U**

**8.1.3.1 Erasing One Character at a Time: RUBOUT** — If you type the RUBOUT key (the DELETE key on some terminals) prior to entering the line with a RETURN key, it causes the last character typed to be erased. If typed in tape mode (see Section 4.8.1), the RUBOUT key is ignored.

If typed at a display (CRT) terminal, the RUBOUT key moves the cursor back one space and deletes the last character typed. No characters are echoed.

If typed at a hard-copy terminal, the RUBOUT key causes the erased character or characters (if RUBOUT is repeatedly typed) to be echoed between backslashes. The following example contains a typographical error (LEF for LET):

```
10      LEF X=X*X
```

To correct this error, type the RUBOUT key 7 times (to move back to and remove the F) and type the remainder of the line correctly. For example:

```
10      LEF X=X*X\X*X=X F\T X=X*X
```

To the system the line appears as:

```
10      LET X=X*X
```

If RUBOUT is used extensively on a hard-copy terminal line, it may become difficult to determine the actual content of the line. To print the line prior to its transmission to the system, type a CTRL/R (see Section 4.9.3).

In cases where the mistake is toward the beginning of a line, it may be easier to simply retype the entire line, as shown in the following example:

```
10      LEF X=X*X  
?ILLEGAL VERB AT LINE 10  
READY  
10      LET X=X*X
```

Once the second line (the corrected one) is entered to the system, the first line number 10 is deleted.

RUBOUT may be typed repeatedly on a display terminal, just as it may on a hard-copy terminal. The only differences in its effect are that characters are not echoed but are in fact erased, no backslashes appear, and the cursor is moved back one space for each character erased.

**8.1.3.2 Erasing One Line at a Time: CTRL/U** — The CTRL/U combination deletes the current input line. This combination is typed by holding down the CTRL key, typing U, and releasing both keys. CTRL/U is particularly useful when you have typed a long line which is incorrect. Rather than type RUBOUT repeatedly, type CTRL/U to delete the entire line. This feature may be used when typing either commands or statements. CTRL/U deletes the entire physical line, as in the following example:

```
10      PPRINT "ALPHABET" ^U  
LISTNH  
READY
```

In typing line 10, you made a mistake (PPRINT for PRINT) and deleted the line with CTRL/U. Note that the line does not appear in the program listing caused by the LISTNH command.

In the next example, you have typed the LINE FEED key to continue line 20 onto a second line. Thus, a physical line has been terminated.

```
20      LET M = LF
278.12^U
?SYNTAX ERROR AT LINE 20
READY
```

The logical statement at line 20, however, is continued onto the following line and its deletion with a CTRL/U causes the statement at line 20 to be entered as:

```
LISTNH
?20 LET M =
READY
```

which is syntactically incorrect. Had the last line been terminated with the RETURN key it would be entered to the system as:

```
20      LET M = LF
278.12 RET
READY
```

which would be accepted as equivalent to:

```
20      LET M = 278.12
```

#### **8.1.4 Removing a Program from a Storage Device: The UNSAVE Command and the KILL Statement**

**8.1.4.1 The UNSAVE Command** — The UNSAVE command removes a .BAS or .BAC file from a storage device. To remove a file from the public disk structure (the default device), type a command in this form:

```
UNSAVE VOR45.BAC
```

This command removes the file VOR45.BAC from the public structure.

Unless a filename extension is specified in the UNSAVE command, the extension BAS is assumed. Thus, if you issue the command:

```
UNSAVE VOR45
```

The system attempts to remove the file VOR45.BAS.

To remove a file from a storage device other than the public disk structure, issue an UNSAVE command in this form:

```
UNSAVE dev:filename.extension
```

where dev: is the device designation. For example, the command:

```
UNSAVE DT1:FLIX
```

removes the file FLIX.BAS from DECtape unit 1, if the file can be found.

To remove a file with an extension other than .BAS, specify the extension in a command of the following form:

```
UNSAVE FILE.001
```

The null extension cannot be used.

**8.1.4.2 The KILL Statement** — The KILL statement, placed within a program, deletes a specified file from your directory. This statement takes the following form:

```
<line number> KILL <string>
```

The string can be a full file specification of the form:

```
"dev:[acct]filename.ext"
```

If the specified file does not exist, the system prints the error message ?CAN'T FIND FILE OR ACCOUNT. If the specified device does not exist, the system prints the error message ?NOT A VALID DEVICE.

In order to delete a file with the KILL statement, you must have write access to the file.

The KILL statement may also be used in immediate mode. Note that in the file specification no defaults are applied for filename or extension. The [string] may be either a string variable or a literal string enclosed in quotes.

### **8.1.5 Merging Programs: The APPEND Command**

The APPEND command merges the contents of a previously saved BASIC-PLUS program into a BASIC-PLUS program currently in memory. To issue this command, type:

```
APPEND
```

The system responds by printing the request:

```
OLD FILE NAME--
```

You then type the name of the saved program to be appended. This program is read into memory, and, depending upon the ordering of its source statement line numbers and those of the current program, the saved program's contents are merged into or appended to the current program. If both programs contain an identical line number, the line in memory is replaced by the appended program line.

You can issue an APPEND without receiving the system's prompt by typing a command of this form:

APPEND RUTINE

This command merges the contents of the saved program RUTINE.BAS from the public structure into the program currently in memory. If the specified program is not available on the public structure or is protected against you, an appropriate message is printed.

If you do not specify a filename in response to the prompt OLD FILE NAME--, BASIC-PLUS looks for the file NONAME.BAS, which could have been created by yourself or the system. (See Section 7.1.1.1 for a description of NONAME.BAS.) In the following example, you do not specify a filename in response to the prompt, but merely press the RETURN key after receiving the prompt:

```
APPEND  
OLD FILE NAME-- [RET]
```

READY

As a result of this procedure, the contents of the program NONAME.BAS on the public structure are merged into or appended to the current program. Thus, the contents of NONAME.BAS becomes available as part of the current program.

In order to retain, under your account, the merged program that results from an APPEND, issue a SAVE or a REPLACE command. If you use SAVE, each of the component programs still exists separately under your account.

The APPEND command can retrieve only BASIC-PLUS source programs (.BAS files), because compiled programs (.BAC files) can be run but not changed. Any file called with APPEND is available to you at the terminal.

The APPEND command may be used with a complete file specification of the form:

```
dev:[acct]filename.ext<prot>/option(s)
```

See Chapter 11 for a description of the complete file specification.

### 8.1.6 Transferring Control Between Programs: The CHAIN Statement

CHAIN is a BASIC-PLUS statement that transfers control from one program to another. CHAIN may be used in immediate mode or as part of a program.

#### NOTE

If a CHAIN operation (see the *BASIC-PLUS Language Manual*) is executed to a source program, an exit from that chained program can cause the initiating program to return to a run-time system that differs from the job's default. That is, if a source program is chained to, the current run-time system executes OLD and RUN operations (see Sections 7.3 and 7.4) on

the source. When the source program terminates, the initiating program exits to the run-time system that performed the OLD and RUN. This run-time system may not be the job's default run-time system. However, if you then run a compiled program, termination of that program returns control to the job's default run-time system.

If your program is too large to be loaded into memory and run in one operation, segment it into two or more separate programs. In doing so, assign each of these programs a unique name. To transfer control from one of these programs to another, issue the CHAIN statement.

In immediate mode, the form of the CHAIN statement is:

```
CHAIN <string> [LINE] <line number>
```

in which <string> is the file specification of the next program segment, LINE is an optional part of the statement (included for compatibility with BASIC-PLUS-2) that can appear when a line number is specified, and <line number> is the line number in that program at which execution is to begin. If no line number is specified, execution begins with the lowest numbered line.

The following example is a CHAIN statement given in immediate mode:

```
CHAIN "PHASE2.BAC" 20
```

This statement executes the segmented program PHASE2.BAC, beginning with line 20. A device specification, project-programmer number, extension, and filename can be included in the file specification string. Note that if you specify a filename with no extension, the Monitor searches all run-time systems for a compiled file of that name. Thus, it is possible for you to specify:

```
CHAIN "FILE"
```

expecting to access FILE.BAS but instead accessing FILE.TSK. To ensure access to the desired file, always include the filename and extension in the CHAIN statement.

Communication between various program elements can be achieved by means of a user-created file or core common (see the discussion of system function calls in the *RSTS/E Programming Manual*).

When the CHAIN statement is executed, all currently open files are closed, all string and numeric variables are deassigned, the new program is loaded, and execution continues. CHAIN loads the designated program into memory and overwrites the current program. The CHAIN statement is similar to the RUN command, but has the additional capability of specifying the number of the line at which execution is to start.

Since each CHAIN statement closes all files, and causes a program load, be careful to minimize the number of CHAIN statements that must be executed in the run of a program.

## NOTE

It is recommended that you explicitly close all open file channels by placing CLOSE statements in the program. The reason for this precaution is that the implicit closing feature of CHAIN will not allow the last write to be performed for virtual core and sequential files.

The <string> in the CHAIN command may be a file specification of the form:

```
"dev:[acct]filename.ext"
```

See Chapter 11 for a description of the complete file specification.

## 8.2 Formatting the Program

### 8.2.1 Changing Statement Format Rules: The EXTEND/NO EXTEND Commands

**8.2.1.1 Description of EXTEND Format** — BASIC-PLUS offers two formats, EXTEND and NO EXTEND. EXTEND, when chosen by you or when installed as a system default, allows expansion of statements and commands beyond the NO EXTEND format.

The following examples illustrate the differences between NO EXTEND and EXTEND format. The first example is a line from the BASIC-PLUS program OCTDEC.BAS, which appears in its entirety in Section 9.4. The second example shows this same line, 300, written in EXTEND mode. Points of difference in format are labeled in both examples and are described in the paragraphs that follow.

```
300 0% = -1% \ D% = 0% ————— (a) ————— (c)
\ FOR Z% = L% TO 1% STEP -1%
\ 0% = 0% + 1%
\ V% = ASCII (RIGHT(S$,Z%))
\ IF V% < 48% OR V% > 55% THEN
    PRINT 'INVALID INPUT'
    GOTO200
! INITIALIZE THE DIGIT POINTER AND ACCUMULATOR,
! FROM THE LAST DIGIT TO THE FIRST,
! UPDATE THE DIGIT POINTER,
! GET THE ASCII VALUE OF THE DIGIT,
! AND SEE IF IT'S VALID.
(b) (d) (a) (c)
```

```
300 DIGITPOS% = -1% \ ACCUMULATOR = 0,      ! INITIALIZE VARIABLES
\ FOR Z% = LENGTH% TO 1% STEP -1%          ! FOR EACH DIGIT
\ DIGITPOS% = DIGITPOS% + 1%                ! INCREMENT DIGIT POINTER
\ VALUE% = ASCII (RIGHT (NUMBER$,Z%))        ! GET THE ASCII VALUE
\ IF VALUE% < 48% OR VALUE% > 55% THEN     ! IF THE DIGIT IS INVALID
    PRINT "INVALID INPUT"                   ! THEN PRINT ERROR MESSAGE
    GOTO 200                                ! TRY FOR ANOTHER NUMBER.
(b) (d)
```

- a. In EXTEND format, variable/function names can have up to 29 alphanumeric (and dot) characters in addition to the leading alphanumeric character, any FN prefix and/or \$ or % suffix. If you mimic existing BASIC-PLUS keywords and built-in function names (for example, IF, GOTO, LEN, etc.), an %ILLEGAL SYMBOL error is returned.

NO EXTEND format allows only 1 alphabetic character or 1 alphabetic and 1 numeric character for variable/function names.

- b. In EXTEND format, spaces and tabs are significant; thus, the statement GOTO200 and the command ASSIGNDT1: are illegal, but GOTO 200 and ASSIGN DT1: are legal.

NO EXTEND format ignores spaces and tabs; thus, GOTO200, GO TO 200, ASSIGNDT1:, and ASSIGN DT1: are all legal.

- c. In EXTEND format, a BASIC-PLUS program line ending with:

&[<space/tab>sequence]<RET>

or

[LF]

signals continuation of the logical line on the next physical line.

NO EXTEND format uses a [LF] to signal continuation of the logical line on the next physical line.

- d. In EXTEND format, a BASIC-PLUS program line ending with:

!<random text>&[<space/tab>sequence]<RET>

signals a comment on the same physical line, and continuation of the logical line on the next physical line.

NO EXTEND format requires you to finish typing the logical line, then place the comment on the next physical line.

**8.2.1.2 Issuing the EXTEND/NO EXTEND Commands** — Issued in immediate mode, the EXTEND or NO EXTEND command changes both your current format and your default format to EXTEND or NO EXTEND, respectively. The default format is the one applied when you issue the NEW or OLD command.

To issue one of the format commands, type:

EXTEND

or

NO EXTEND

and follow either command by pressing the RETURN key.

Issued as a program statement, EXTEND or NO EXTEND changes only the current format to EXTEND or NO EXTEND respectively. The statement has the form:

<line number> EXTEND

or

<line number> NO EXTEND

where line number is the number of the current line.

### 8.2.2 NO EXTEND Statement Formatting

This section describes formatting features used in NO EXTEND format.

**8.2.2.1 Multiple Statements on a Single Line** — More than one statement can be written on a single line as long as each statement (except the last) is terminated with a colon or a backslash. (The backslash is preferred.) Thus only the first statement on a line can (and must) have a line number. For example:

10 INPUT A,B,C

is a single statement line, while:

20 LET X=X+1.\ PRINT X,Y,Z\ IF Y=2, THEN GOTO 10

is a multiple-statement line containing three statements: a LET, a PRINT, and an IF-GOTO statement.

Any statement can be anywhere in a multiple-statement line except as noted in the discussion of the individual statements (see the *BASIC-PLUS Language Manual*).

**8.2.2.2 A Single Statement on Multiple Lines** — A single statement can be continued on successive lines of the program. To indicate that a statement is to be continued, terminate the line with the LINE FEED key instead of the RETURN key. The LINE FEED performs a carriage return/line feed operation on the terminal, and the line to be continued does not contain a line number. For example:

10 LET W7=(W-X4\*3,)\*(Z-A/\  
(A-B)-17,)

where the first line was terminated with the LINE FEED key is equivalent to:

10 LET W7=(W-X4\*3,)\*(Z-A/(A-B)-17,)

Note that the LINE FEED key does not cause a printed character to appear on the page.

The length of a multiple-line statement is limited to 255 continuation lines.

Where the LINE FEED key is used, it must occur between the elements of a BASIC statement. That is, a BASIC verb or the designation of a subscripted array element, for example, cannot be broken with a LINE FEED.

```
10 IF A1=0.LF
THEN GOTO 100
```

is acceptable where a LINE FEED follows 0., but:

```
10 IF ALF
1=0 THEN GOTO 100
?ILLEGAL CONDITIONAL CLAUSE
```

is not acceptable, nor is:

```
10 IF A1=0. THEN GOTO 1LF
00
?MODIFIER ERROR AT LINE 10
```

and each illegal form generates an error message. A number of multi-word elements are processed as one word and cannot be broken by a LINE FEED. For example, AS FILE, FOR INPUT AS FILE, FOR OUTPUT AS FILE, GO TO, INPUT LINE, and ON ERROR GO TO are each treated by the system as one word.

**8.2.2.3 Spaces and Tabs for Readability** — Spaces should be used freely throughout the program to make statements easier to read. For example:

```
10 LET B = D*2, + 1,
```

instead of:

```
10LETB=D*2.+1.
```

or:

```
10 L ETB = D * 2, + 1.
```

The above statements are identical in effect.

Tabs, like spaces, are used to make a program easy to read. An example follows:

```
20 IF A>B THEN
    IF B>C THEN
        PRINT "A>B>C"
    ELSE    IF C>A THEN
        PRINT "C>A>B"
    ELSE    PRINT "A>C>B"
ELSE      IF A>C THEN
        PRINT "B>A>C"
    ELSE    IF B>C THEN
        PRINT "B>C>A"
    ELSE    PRINT "C>B>A"
```

## **Chapter 9**

### **Debugging a Program**

The phase of program development during which you test the program is called the debugging phase. BASIC-PLUS provides you with ways to debug programs without having to run them over and over. These methods, described in this chapter, are immediate mode, the STOP statement, and the commands PRINT LINE, CONT, and CCONT (a privileged command).

#### **9.1 Executing Statements In Immediate Mode**

The immediate mode of BASIC-PLUS is so called because it enables you to enter a statement and to cause its immediate execution, without having to include that statement in a complete program. Immediate mode thus treats a BASIC-PLUS statement in much the same way as a command. Some of the operations included in Chapters 7 and 8 as system commands--NAME AS and CHAIN, for instance--are actually BASIC-PLUS statements issued in immediate mode.

Immediate mode is especially useful for two general purposes: performing simple calculations which occur too infrequently to be programmed, and debugging programs--the subject of this chapter.

To execute a BASIC-PLUS statement in immediate mode, type that statement without a line number. BASIC-PLUS distinguishes between a programmed and an immediate statement solely by the presence or absence of a line number: numbered statements are stored; non-numbered statements are executed immediately on being entered. Thus, the statement:

10 PRINT "STORE IT AWAY"

produces no effect at your terminal upon entry. But a similar statement, entered without a line number, causes immediate output, as shown in the following example:

```
PRINT "DO IT NOW"
DO IT NOW

READY
```

The system prints the READY prompt to indicate its readiness for more input.

### 9.1.1 The Limitations of Immediate Mode

Before discussing the role of immediate mode in program debugging, it is useful to consider some of its limitations. Some BASIC-PLUS statements must belong to a program and interact with its other statements in order to produce results. The following statements would be logically stranded in immediate mode, and would make no sense:

```
DEF
DEF*
FNEND
DIM
DATA
FOR
NEXT
```

(Note, however, that FOR as a modifier will work in immediate mode.)

When you give any of these statements in immediate mode, the system prints the message ?ILLEGAL IN IMMEDIATE MODE.

Immediate mode does not permit multiple statements on a single line. The following example shows an attempt to enter two statements on one line, and causes an error message:

```
A=1.\ PRINT A
?ILLEGAL IN IMMEDIATE MODE

READY
```

## 9.2 Using Immediate Mode with STOP, CONT, CCONT, and GOTO

### 9.2.1 The STOP Statement

Rather than repeatedly executing and altering a program, you can facilitate debugging by strategically placing STOP statements throughout the program. Upon execution, each STOP statement causes a program halt, and a message of the form:

```
STOP AT LINE 50
```

is printed. In this case, the STOP statement was located at line 50. After the STOP AT LINE message is printed, you can give statements in immediate mode to examine and/or change data values. You can also add, delete, or modify lines. In addition, you can resume execution at another location by issuing a GOTO statement. When you are ready to resume execution, type CONT.

### **9.2.2 The CONT Command**

After you have performed debugging operations during a program halt caused by a STOP, continue program execution by issuing the CONT command as follows:

```
CONT
```

Execution then continues from the next statement after the STOP.

Note, however, that a syntax error or an illegal statement in immediate mode can prevent the CONT command from continuing program execution. When execution cannot be continued, the system prints the message:

```
?CAN'T CONTINUE  
READY
```

### **9.2.3 The CCONT Command**

The CCONT command, available only to privileged users, performs the same actions as the CONT command but also detaches the job. Its special use lies in continuing a lengthy job that needs no further terminal interaction.

#### **NOTE**

Issued by a non-privileged user, CCONT produces the message ?ILLEGAL SYS() USAGE. To continue, the non-privileged user must issue the CONT command.

### **9.2.4 The GOTO Statement**

You can resume execution at a particular line by issuing the GOTO statement in immediate mode. Note that any GOTO which causes transfer of control into or out of a FOR loop, function, or subroutine may cause unexpected results.

## **9.3 Debugging with CTRL/C, PRINT LINE, and CTRL/O**

### **9.3.1 Halting and Checking Execution with CTRL/C and PRINT LINE**

Another way to halt program execution is to type the CTRL/C combination (see Section 4.9.1). CTRL/C, however, gives you less control over where the program halts than does the STOP statement.

When the program is halted by CTRL/C, the integer variable LINE contains the line number of the statement being executed at the time of the halt. Therefore, if you type CTRL/C followed by PRINT LINE in immediate mode, the contents of LINE--i.e., the current line number--will be displayed at the terminal. The following example illustrates this procedure:

```
^C  
READY  
PRINT LINE  
300  
READY
```

As when the program is halted by a STOP statement, the CONT command, CCONT command, or GOTO statement may be used to continue execution.

If a multi-statement line is being executed, you have no way of knowing where in the line the program stopped. Some system programs are designed to trap CTRL/C to prevent their being interrupted within critical sequences.

### 9.3.2 Suppressing Output with CTRL/O

The CTRL/O combination suppresses output to the terminal without halting execution of the program; to continue output to the terminal, type another CTRL/O combination. Note that after a CTRL/O is issued, the program's output continues but is not printed at the terminal.

A program has no way of determining whether a CTRL/O has been typed at the terminal. There is, however, a system function call that overrides the ability of CTRL/O to suppress output.

### 9.3.3 Suppressing Output with CTRL/S and CTRL/Q

Also useful for temporarily suppressing output on display (CRT) terminals only is the CTRL/S combination (hold down the CONTROL key and type S). To continue output to the terminal, type the CTRL/Q combination. These two features are usable only if the STALL characteristic is set on the terminal (see Section 19.1).

## 9.4 An Example of Program Debugging

In the following example, assume that you have found that the program OCTDEC, written to convert octal numbers to decimal, returns incorrect results. (The octal number 23, for example, is converted to decimal number 56.) Employing STOP statements, CONT statements, and immediate mode, you debug the program. The step-by-step procedure appears after the listing of OCTDEC.

```

10      !THIS IS AN OCTAL TO DECIMAL CONVERSION PROGRAM.
100     ON ERROR GOTO 900
      \ PRINT
      \ PRINT "OCTAL TO DECIMAL CONVERTER"
      \ PRINT "CONVERTS NUMBERS BETWEEN 0 AND 177777 (OCTAL) TO"
      \ PRINT "THEIR DECIMAL EQUIVALENTS"
      ! PRINT OUT THE INSTRUCTIONS AND HEADER.

200     INPUT "OCTAL NUMBER";S$
      \ L% = LEN(S$)
      \ GOTO 600 IF L% = 0%
          ! INPUT A CHARACTER STRING,
          ! GET ITS LENGTH,
          ! AND CLOSE OUT IF ITS LENGTH IS 0.

250 STOP
300     0% = -1% \ D% = 0%
      \ FOR Z% = L% TO 1% STEP -1%
      \     0% = 0% + 1%
      \     V% = ASCII (RIGHT(S$,Z%))
      \     IF V% < 48% OR V% > 55% THEN PRINT 'INVALID INPUT'
      \ GOTO 200
          ! INITIALIZE THE DIGIT POINTER AND ACCUMULATOR.
          ! FROM THE LAST DIGIT TO THE FIRST,
          ! UPDATE THE DIGIT POINTER,
          ! GET THE ASCII VALUE OF THE DIGIT,
          ! AND SEE IF ITS VALID.

350 STOP
400     V% = V% AND 7%
      \ D% = D% + (V% * INT(B% ^ 0%))
      \ NEXT Z%
          ! CHANGE THE ASCII REPRESENTATION TO A NUMERIC
          ! REPRESENTATION,
          ! AND THE VALUE WILL ACCUMULATE IN D%.
          ! GO BACK AND DO NEXT DIGIT IF THERE IS ONE.

500     PRINT "DECIMAL VALUE IS ";NUM1$(D%)
      \ GOTO 200
          ! PRINT OUT THE RESULT AND TRY ANOTHER.

600     GOTO 32767
900     IF ERL = 400 THEN
          PRINT "NUMBER ";S%;" TOO BIG FOR CONVERSION"
          \ RESUME 200
910     PRINT ERL
          \ RESUME 32767
32767 END

```

1. You suspect that the bug is in line 300 or 400 and therefore insert a STOP statement before each of these lines, at lines 250 and 350.
2. You run the program and input the octal number 23. The program stops at line 250:

```

RUNNH
OCTAL TO DECIMAL CONVERTER
NUMBERS BETWEEN 0 AND 3641077
OCTAL NUMBER? 23
Stop at line 250

```

Ready

3. To check the value of variable L%, issue a PRINT statement in immediate mode:

```
PRINT L%
```

```
2
```

```
Ready
```

4. Finding the value of L% correct (23 being 2 digits), issue a CONT statement to continue execution. The program stops on encountering the next STOP statement:

```
CONT
```

```
Stop at line 350
```

```
Ready
```

5. To check the values of variables D, O%, V%, and Z%, issue another PRINT statement in immediate mode:

```
PRINT D,O%,V%,Z%
```

```
0
```

```
1
```

```
51
```

```
2
```

```
Ready
```

You recognize that the value of O%, printed as 1, should be 0 at this point in execution. (The program is dealing with the digit in position 0, and O% was initialized -1.) You find the source of the error in the third physical line at 300, where O% (the letter variable) was mistyped as 0% (zero).

You can now edit the program to correct the typographical error.

## **Chapter 10**

### **Introduction to Part IV**

This part of the User's Guide is largely devoted to programs in the RSTS/E library. These programs perform diverse services for the RSTS/E user: functions as simple as printing a "sign" at the terminal that tells others it is in use, and more complex functions such as comparing and editing files, transferring data, and running batch jobs. Table 10-1 provides you with a guide to the library programs and the functions they perform.

Chapter 11 sets forth an "anatomy" of the RSTS/E file specification -- its parts, their meanings, their system-assigned defaults and the alternatives to those defaults. Since nearly all the library programs deal with files -- the basic units of data in RSTS/E -- you should understand file specifications before reading about, and attempting to work with, most programs in the RSTS/E system library.

Chapter 12 introduces the system library programs as a group by discussing their shared characteristics and features: for example, how some of them can be run by CCL commands, and the ways in which they identify themselves and provide helpful information at your request.

Table 10-1 serves as a guide to the library program chapters. Table 10-2 lists the CCL (Concise Command Language) commands, which are brief ways to run some system programs. For more information on CCL commands, see Section 12.1.

In Table 10-2, the shortest forms of the CCL commands appear outside square brackets. Any number of characters within square brackets may also be typed, in sequence. For example, the SYSTAT program may be run by typing any one of the following commands: SY, SYS, SYST, SYSTA, SYSTAT.\*

---

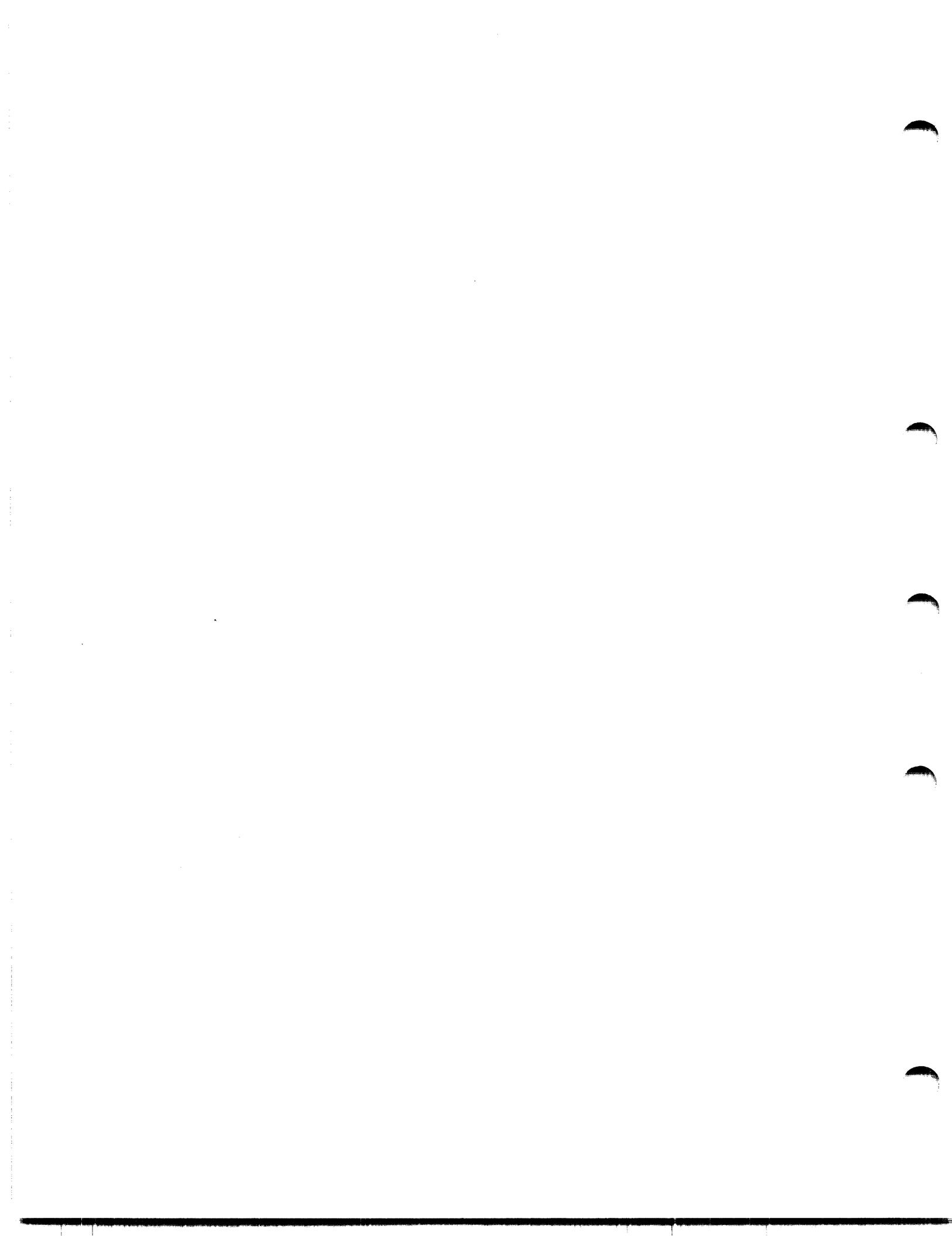
\*Note that the CCL abbreviations in the table are those supplied by DIGITAL. A system manager can install additional CCL commands, and can alter any existing CCL commands or their abbreviations.

**Table 10-1: Overview of Programs and File Information**

Function	Program	Location
Account reporting	MONEY	14.3
Batch processing	BATCH	Chapter 21
Comparing files	FILCOM	15.2
Device, copying	COPY	16.4
Device, requesting	QUE	Chapter 20
Device transfers	PIP	Chapter 16
Directory, listing	DIRECT	15.1
Diskette; IBM transfer	FLINT	Chapter 18
Disk; RT-11, DOS, and RSTS/E transfer	FIT	16.5
Disk, mounting private	UMOUNT	19.2
Dumping user memory	PMDUMP	Chapter 22
Flexible diskette transfer	FLINT	Chapter 18
Information on system programs	HELP	12.5
Logging in	LOGIN	13.1
Logging out	LOGOUT	13.2
Magnetic tape, mounting private	UMOUNT	19.2
Message to manager	GRIPE	14.4
Message: "terminal in use"	INUSE	14.5
Preserving data off-line	BACKUP	Chapter 17
Quota report	QUOLST	14.2
Status report on system	SYSTAT	14.1
Terminal settings	TTYSET	19.1
<b>Elements of the file specification:.</b>		
Element	Location	
device:	11.1	
account (proj,prog) number	11.2	
filename.extension	11.3	
protection code	11.4	
filespec options	11.5	

**Table 10-2: CCL Commands that Run System Programs**

Program	Command	Function of Command
DIRECT	DIR[ECTORY] DIR[ECTORY] filespec DIR[ECTORY]/option(s)	Lists standard directory of your account. Lists directory of specified file. Lists directory according to option(s) specified.
FLINT	FLI[NT]	Runs the FLINT dialogue.
HELP	HELP topic	Lists information on system programs and resources (use, command syntax, etc.).
UMOUNT	MOU[NT] device:pack id label/option(s)	Logically associates the specified disk pack with the specified identification label, and applies the specified option(s).
	DIS[MOUNT] device:pack id label	Logically dismounts the disk pack on the specified drive, and removes the specified pack identification label from system's table of logical names.
PIP	PIP PIP [command line]	Runs the PIP program. Executes [command line] as a standard PIP command.
TTYSET	SET [command line]	Executes [command line] as a standard TTYSET command.
QUE	QU[EUE] [command line]	Executes [command line] as a standard QUE command.
SYSTAT	SY[STAT] SY[STAT]/option(s)	Prints standard system status report. Prints system report according to option(s) specified.
SWITCH	SW[ITCH] SW[ITCH] run-time system	Invoke another run-time system.
SUBMIT	SU[BMIT] file spec/option	Causes QUE to submit a job to the Batch processor.



# **Chapter 11**

## **File Information and Standards**

A RSTS/E file specification contains some or all or the following information:

```
device:[proj,prog]filename.extension<prot> /option(s)
```

The elements of the file specification are discussed in the following sections. Table 11-1 presents the default supplied by the system when you do not specify an element of the file specification.

**Table 11-1: File Specification Elements and System Defaults**

Element	Meaning	Default
device:	device designator	public disk structure.
[proj,prog] or [account]	project-programmer or account number	your current account number.
filename .extension	title of file .type of file	defaults depend upon current program and procedure.
<prot>	protection code	on most systems, <60>.
/option	cluster size, position, data mode, and /or total size of file	defaults for options, of which there are 5, depend on the system.

### **11.1 device: The Device Designator**

If you do not specify a device designator, the system supplies the public structure by default. For non-file structured devices (paper tape, line printer, etc.), only the device designator need be specified; the system ignores any filename, extension, account number, or protection code that you specify.

A device designator, or device specification, consists of 2 alphabetic characters optionally followed by a decimal unit number, and always terminated by a colon (:). The alphabetic characters are generally an abbreviation of the device's generic name (DT for DECtape, DK for disk, MT for magnetic tape, etc.), and the unit number uniquely identifies the device itself or the drive on which it is currently mounted.

Table 11-2 lists and explains the RSTS/E device designators.

**Table 11-2: RSTS/E Device Designators**

Designator	Device
DF:,DS:,DK:,DL:,DM:,DP:,DR:,DB:, or SY:	RSTS/E public disk structure.
SY0:	System disk (the unit which was booted).
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.
MM0: to MM7:	TE16 /TU16 /TU45 /TU77 magnetic tape units 0 through 7.
MS0: to MS3:	TS11 magnetic tape units 0 through 3.
LP0: to LP7:	Line printer units 0 through 7.
DT0: to DT7:	TU56 DECtape units 0 through 7.
DD0: to DD7:	TU58 DECtape II unit 0 through 7.
KB:	Current user terminal.
KBn:	Terminal n in the system.
TTn:	Terminal n in the system (synonym for KBn:).
TI:	Current terminal (synonym for KB:, the terminal that initiated the job).
DX0: to DX7:	RX01 /RX02 flexible diskette units 0 to 7.

## NOTE

You can reference LPn:, DTn:, DXn:, KBn:, MMn:, MTn:, MSn:, and DDn: where n is between 0 and the maximum number of such units on the system. LP:, DT:, DX:, MM:, MT:, MS:, and DD: are each the same as specifying unit 0 of the related device.

If your system has only TS11 tape units, the designators MS: and MT: are synonymous. If your system has only TE16, TU16, TU45, or TU77 tape units, the designators MM: and MT: are synonymous. On systems which have the CD11 card reader, the designator CR: is synonymous with CD:.

If you attempt to specify a device or type of device not available on the system, the error message ?NOT A VALID DEVICE is printed.

### 11.1.1 Logical Device Names

A device may be identified either by a physical device name (i.e., a device designator) or by a logical device name; logical device names are associated with devices by the ASSIGN command (see Section 4.4) or by the system manager. A logical device name consists of 1 to 6 characters terminated by a colon. Examples of logical device names are DTA:, DTFAC:, MTMINE:, MYPACK:, DK5:. A project-programmer (account) number may be associated with a logical name.

## NOTE

When you specify a device name, physical or logical, the system seeks the device by the following procedure: 1) the system determines if the device name is specific to one job; 2) if it is not, the system determines if it is a system-wide logical name or disk pack identification label; 3) if the device name is neither job-local nor system-wide, the system determines if it is a physical device designator.

## 11.2 [proj,prog] The Project-Programmer or Account Number

If you do not specify a project-programmer code, the system assumes your number by default; that is, the owner of the file is assumed to be the current user. This code is meaningful only for disk and magnetic tape files; it has no significance for files on DECtape or on non-file structured devices. The [proj,prog] code consists of two decimal numbers between 0 and 254, separated by a comma, and enclosed within square brackets[ ] or parentheses( ). These numbers have the following meanings:

proj represents a project group consisting of users with a common task or interest.

prog represents an individual user in that project group.

RSTS/E uses project-programmer codes for two general purposes: 1) to identify files according to their owners, and 2) to apply the files' protection codes to individual users and to groups of users (see Section 11.4). Note that a

project-programmer number (PPN) is identical in use and format with a User Identification Code (UIC) described in some RSTS/E utility and optional software documentation.

### 11.2.1 Special Account Characters

In any location where a project-programmer code is valid, you can specify one of the special non-alphanumeric characters listed below. Each of these characters designates an account. The \$, for example, designates the system library account [1,2]. Thus, a file specification containing a \$ denotes a file stored in that account. The usual application of \$ is in calling a system library program -- the command RUN \$PIP, for example.

The special account characters and the accounts they designate are as follows:

Character	Account
\$ (ASCII 36)	[1,2], system library account.
! (ASCII 33)	[1,3], determined by manager.
% (ASCII 37)	[1,4], determined by manager.
& (ASCII 38)	[1,5], determined by manager.
# (ASCII 35)	[proj,0].
@ (ASCII 64)	assignable account.

The # character allows each project on the system to have its own library of files common to all members of that project.

Assignment and use of the @ character are explained in Section 4.7. If you specify the @ character without having previously assigned it to an account, the ?ILLEGAL FILE NAME error is printed. Note that the @ character can also be used in an indirect command file specification.

## 11.3 filename.extension The Filename and Extension

For file structured devices, each file is assigned a filename and extension.

The filename is a string of one to six alphanumeric characters, without embedded nulls, tabs, or spaces. It is the only element of a file specification without a delimiting mark of punctuation. The filename may also include or consist of the wildcard character ?. Or, it may consist of only the wildcard character \* (see Section 11.3.1).

The filename extension is a string of one to three alphanumeric characters without embedded nulls, tabs, or spaces and is preceded by a dot (.). Usually, the extension denotes the file's type: .BAS, for example, indicates a BASIC-PLUS source file; .CTL, a program control file; .TMP, a temporary file (see Table 11-3 for common RSTS/E extensions and their meanings). Note that a temporary file (i.e., an explicit .TMP extension) is subject to deletion when the system reboots, when the disk is cleaned, or a similar situation occurs. An extension may include, or consist of, the wildcard character ?. Or, it may consist of only the wildcard character \* (see Section 11.3.1).

**Table 11-3: RSTS/E Filename Extensions**

Extension	Meaning
.B2S	BASIC-PLUS-2 source file.
.BAC	BASIC-PLUS compiled output file.
.BAS	BASIC-PLUS source file.
.BAK	BAckUp file created by EDIT program.
.CBL	COBOL source file.
.CMD	indirect CoMmand file for running a system program.
.CRF	Cross ReFerence listing file.
.CTL	BATCH ConTroL file.
.DAT	DATA file.
.DIF	FILCOM output file.
.DIR	DIRectory file.
.DOC	RUNOFF output file.
.FOR	FORTRAN source file.
.HLP	system program HeLP text file.
.LIB	Resident LIBrary file.
.LOG	BATCH output LOG file.
.LST	LiSTing file created by a system program.
.MAC	MACro source file.
.MAP	MAP file.
.MLB	MACRO LiBray file used by the MACRO Assembler.
.OBJ	compiled COBOL, FORTRAN, or BASIC-PLUS 2 program.
.ODL	Overlay Description Language input file.
.OLB	Object module LiBray file used by the Task Builder.
.PMD	Post-Mortem Dump.
.RNO	RUNOFF source file.
.RTS	RSTS/E run-time system.
.SAV	compiled and linked FORTRAN, or MACRO program.
.SIL	Save Image Library.
.SRT	SORT-11 file.
.STB	Symbol TaBle file produced by the Task Builder.
.SYS	SYStem file, usually for internal use.
.TMP	TeMPorary file created by a system program (deleted when you log off or if the disk is CLEANed).
.TSK	Executable COBOL or BASIC-PLUS 2 program.
.TXT	ASCII TeXT file.

A null extension, in which only the dot is specified, or a blank extension, in which case the dot and filename extension field are both omitted from the file designation, are permitted.

### 11.3.1 Wildcard Specifications

Many of the RSTS/E system library programs, when requiring a file specification, allow you to specify the wildcard characters \* (asterisk) and ? (question mark). The \* character replaces an entire field, while the ? character replaces a character within a field.

**11.3.1.1 The \* Wildcard** — The \* (asterisk), replacing a filename, extension, or both (one \* for each), denotes all filenames, all extensions, or all filenames with any extensions. For example:

- FILE.\* denotes all files with filename FILE and any extension; for example, FILE.DAT, FILE.TXT, FILE.8, FILE.9B.
- \*.EXT denotes all files with EXT extensions; for example, DATA.EXT, MYFILE.EXT, FB10.EXT, DD.EXT.
- \*.\* denotes all files; for example, all the files listed as examples above.

**11.3.1.2 The ? Wildcard** — The ? (question mark), in any position of either the filename or extension, denotes any alphanumeric character appearing in that position. The following examples illustrate:

- FILE.EX? denotes all files with filename FILE and an extension consisting of EX, or of EX and any other alphanumeric character; for example, FILE.EX1, FILE.EX2, FILE.EXF, FILE.EXE, FILE.EX.
- FILE???.EXT denotes all files with extension .EXT and a filename consisting of FILE and any other two alphanumerics, including trailing blanks; for example, FILE01.EXT, FILE54.EXT, FILE3B.EXT, FILE3.EXT.
- FILE???.E?? denotes all files with a filename consisting of FILE and any other two alphanumerics and an extension consisting of E and any other two alphanumerics (trailing blanks are included); for example, FILE54.EXT, FILE01.ERA, FILEJQ.E91, FILE91.EJQ.
- FI?.EXT denotes all files with extension .EXT and a filename consisting of FI and any other alphanumeric (trailing blanks are included); for example, FI1.EXT, FIL.EXT, FI7.EXT, FIG.EXT.

**11.3.1.3 The \* and ? Wildcards Combined** — The \* and ? wildcards may be intermixed in a file specification. The following examples show such mixtures and their meanings:

- FILE???.\* denotes all files with any extension and a filename consisting of FILE and any two alphanumerics; for example, FILE60.DAT, FILE75.DAT, FILEZX.TXT, FILES5.B, FILEB6.AM.
- \*.EX? denotes all files with an extension consisting of EX and any other alphanumeric; for example, MYFILE.EXT, YRFILE.EXT, MCR.EXE.

## 11.4 <prot> Protection Code

The protection code is a string of one to three decimal digits enclosed by angle brackets <>. This string determines the file's degree of protection on two levels: the actions -- reading, writing, and deleting -- against which it is protected, and the user or class of users against whom it is protected. There are three such user classes, which the system recognizes by project-programmer numbers as follows:

1. The individual user (owner)  
who is recognized by his programmer number: [200,25].
2. The user's project group  
which is recognized by the user's project number: [200,25], [200,57], [200,70].
3. All other users on the system  
who are recognized by the existence of valid project-programmer numbers: [225,60], [250,35], [254,10].

Thus, two variables -- read /write privileges and class of user -- determine protection. Degrees of protection for data files are enforced by the following individual codes; typically, a file's total protection code is the sum of the desired combination of individual codes. These individual codes and their meanings are listed in Table 11-4.

**Table 11-4: File Protection Codes**

Code	Meaning
1	read protection against owner.
2	write protection against owner.
4	read protection against owner's project group.
8	write protection against owner's project group.
16	read protection against all others who do not have owner's project number.
32	write protection against all others who do not have owner's project number.
64	executable program: can be run only.  Individual codes added to the compiled protection <64> have meanings different from those of the data file protection codes above. These compiled codes follow:
1	execute protection against owner.
2	read and write protection against owner.
4	execute protection against owner's project group.
8	read and write protection against owner's project group.
16	execute protection against all others who do not have owner's project number.
32	read and write protection against all others who do not have owner's project number.
128	program with temporary privileges or file with protected data.

In accordance with the codes in Table 11-4, therefore, a data file with protection <60> -- the usual system default -- is protected against reading, writing, and deleting by all users except its owner: <60>=4+8+16+32.

Note that a file which has a protection code of <128> is overwritten with zeroes when it is deleted. That is, when you execute an UNSAVE or KILL on a protected file (code <128>), the system overwrites that file with zeroes.

#### NOTE

When the system overwrites a protected file with zeroes, all other file processing requests are deferred until the overwrite operation is completed. It is recommended that PIP be used to delete large protected files (see Section 16.2.6).

### 11.5 /option File Specification Option

A file specification option may be included as the final element of the specification string. Five options are possible: /FILESIZE, /CLUSTERSIZE, /POSITION, /MODE, and /RONLY. These options specify, respectively, the disk size -- in blocks -- to which the file is pre-extended, the minimum number of contiguous disk blocks forming a cluster, the specific block number on disk at which the file is to be placed, and the read /write mode in which the file's data is passed to the device driver. Note that /FILESIZE, /CLUSTERSIZE, and /POSITION are used for disk files.

If a file specification option is in a position other than the final one, is missing a colon, or is not a valid form, the system prints the error message ?ILLEGAL SWITCH USAGE. For example, either of the following specifications will produce the switch usage error:

ABC/SI:100 [1,2]

or

ABC/SI@

If an argument is missing, or contains an illegal character, the system generates the ?ILLEGAL NUMBER error.

#### 11.5.1 /FILESIZE Option

The /FILESIZE option allows the creation of a disk file of the specified size, in blocks, before any read /write operations are performed. Thus, /FILESIZE reserves space on the disk for data to be placed in the file. This option consists of 1) a slash (/), 2) any one of the unique abbreviations shown in the list that follows, 3) a colon (:), 4) an optional pound sign (#) for octal conversion of the argument n, 5) the argument n, a decimal number which indicates the number of blocks in the pre-extended file, and 6) an optional trailing decimal point (.) to ensure that n is interpreted as a decimal number.

The argument n specifies the length in disk blocks to which the file is pre-extended. That is, it reserves a specified portion of disk space for the file. The value of the /FILESIZE argument is dependent on the type of file. If the file is compiled or is to be executable, the argument n must be in the range of 0 to

65535. If the capability to create large files is present on your system and the file is not compiled or executable, the argument n can be in the range of 0 to  $2^{23}-1$  (assuming that the disk is large enough). Note that you can not run or compile a file that is larger than 65535 blocks. An attempt to pre-extend a compiled file beyond block 65535 returns a ?PROTECTION VIOLATION error.

The /FILESIZE option may be minimally abbreviated to /FI or to /SI. The following list shows its valid forms:

```
/FI:[#]n[.]  
/FIL:[#]n[.]  
/FILE:[#]n[.]  
  
:  
  
/FILESIZE:[#]n[.]  
  
or  
  
/SI:[#]n[.]  
  
:  
  
/SIZE:[#]n[.]
```

### 11.5.2 /CLUSTERSIZE Option

The /CLUSTERSIZE option establishes the minimum cluster size for a disk file; a cluster is a number of contiguous blocks taken together as a unit. The /CLUSTERSIZE option is especially useful for large files; specifying a large cluster size speeds up random access to the data and prevents such files from crowding or filling your directory, whose size is limited. RSTS/E permits cluster sizes of 1, 2, 4, 8, 16, 32, 64, 128, or 256 blocks.

The /CLUSTERSIZE option consists of 1) a slash (/), 2) CLUSTERSIZE or a minimum abbreviation of CL, 3) a colon (:), 4) an optional minus sign (-) to specify a negative cluster size (explained in the next paragraph), 5) an optional pound sign (#) for octal interpretation of the argument n, 6) the argument n specifying the cluster size in blocks, and 7) a decimal point (.) to ensure decimal interpretation of n. The following list shows the valid forms of the option:

```
/CL:[-][#]n[.]  
/CLU:[-][#]n[.]  
/CLUS:[-][#]n[.]  
  
:  
  
/CLUSTERSIZE:[-][#]n[.]
```

Specifying a negative cluster size avoids certain errors associated with disk devices. A negative cluster size causes the system to use the absolute value of the cluster size, if the device on which the file is created allows that value. If the ?ILLEGAL CLUSTER SIZE error is encountered because the absolute value is less than the device cluster size at which the file is to be created, the system uses the device cluster size rather than return the error. For example, assume you are accustomed to creating a file with a cluster size of 2 on an

RK05 disk cartridge, which is a system scratch disk. The scratch disk, however, temporarily changes to an RP06 disk pack, which has a device cluster size of 8. Your customary cluster size of 2, therefore, would be illegal on the RP06. But by specifying the negative cluster size of -2, you guarantee that the file creation will not fail because of the RP06 disk's cluster size restriction.

### 11.5.3 /POSITION Option

The /POSITION option allows you to specify the location of a file on disk. The option consists of 1) a slash (/), 2) POSITION or a minimum abbreviation of PO, 3) a colon (:), and 4) the argument n, a decimal number that specifies the desired placement of the file.

The argument n indicates a device cluster number on the disk. The device cluster numbers vary from disk to disk; refer to the *RSTS/E System Generation Manual* for the table of device sizes. If the value of n in /POSITION:n is 0, no placement is performed and the system determines the location of the file on the disk. If the value of n is a non-zero number, the system attempts to place the first block of the file at the specified device cluster number on the disk. If the file can not be placed at that location, the system places the first block of the file at the first free cluster number that is greater than the specification. When a file is placed, it is marked as such by the Monitor. Note that no error is returned if the system is unable to place the file at the specified location. To determine the actual location of the file on disk, use the /PO,/FU, or /S options of the DIRECT program (see Section 15.1).

For example:

```
DIR /PO
SY:[2,211]
RMSTST.B2S 18698
RMSTST.OBJ 18720
RMSTST.CMD 18721
RMSTST.ODL 18722
RMSTST.TSK 18701
RMSTST.MAP 18723
PROOF.BAS 18741
PFILE.DAT 18742
RDPLN1.RNO 18743
RDPLN1.DOC 18747
UBAK17.RNO 18751
UBAK17.DOC 18767
CHTST.BAS 6020
CHTST1.BAS 9255
CHTST1.B2S 16257
CHTST1.OBJ 16313
CHTST1.CMD 16433
CHTST1.ODL 16522
CHTST1.TSK 16562
```

The following list shows the valid forms of the /POSITION option:

```
/PO:n
/POS:n
/POSI:n
+
+
/
/POSITION:n
```

Note that the position of a file on disk may change if you backup and restore the file with the BACKUP program.

#### 11.5.4 /MODE and /RONLY Options

The /MODE option enables the passing of up to 15 (decimal) bits of information to the device driver at file open time. The meaning of these bits (if any) is device dependent, and determines the read /write mode for data transfer. For explanations of the bits, see the *RSTS/E Programming Manual*.

The /MODE option consists of 1) a slash ( /), 2) MODE or a minimum abbreviation of MO, 3) a colon (:), 4) an optional pound sign (#) for octal interpretation of the argument n, 5) the argument n specifying a mode setting between 0 and 32767 (decimal) inclusive, and 6) an optional decimal point (.) to ensure decimal interpretation of n. The following list shows the valid forms of the option:

```
/MO:[#]n[.]  
/MOD:[#]n[.]  
/MODE:[#]n[.]
```

The /RONLY option enables setting of the read only MODE value for a disk file. The /RONLY option consists of 1) a slash ( /), and 2) RONLY or a minimum abbreviation of RO. The following list shows the valid forms of the option:

```
/RO  
/RON  
/RONL  
/RONLY
```

### 11.6 File Attributes

File attributes are file description data required by RMS-11 software. RMS-11 is used by COBOL-11, BASIC-PLUS-2, and RPG II. The attributes of a given file are recorded in the User File Directory (UFD) entry for that file. Some of the more important file attributes are:

- file organization
- record type
- record size
- file size
- location of the next free byte within the file

You can list the attributes of one or more files with the DIRECT program options /SA, /AT, /OA, and /S (see Section 15.1) or with the PIP program directory options :OA, :SA, :S, and :AT (see Section 16.2.8). The options list

the file's attributes symbolically or octally on the line below the filename. The symbolic attribute fields are defined as follows:

- RF is record format and, if given, the maximum record length. The possible formats are UDF (undefined), FIX (fixed), VAR (variable), VFC (variable / fixed control), and STM (stream).
- FO is file organization; SEQ (sequential), REL (relative), and IDX (indexed).
- USED is the number of blocks currently in use followed by the number of bytes being used in the last block.
- RECSI is the size in bytes of the largest record in the file.
- CC is carriage control; IMP (implied) or FOR (FORTRAN).
- NOSPAN is printed if records are not allowed to span block boundaries.
- BK is printed if the file organization is relative or indexed and indicates the bucket size in blocks.
- EX if the default is not used, indicates the file extension quantity.
- HS if the default is not used, indicated the fixed header size.

For example:

```
DIR /SA
SY:[2,211]
RMSTST.B2S
RMSTST.OBJ
    RF:VAR=128    FO:SEQ    USED:7:22        RECSI:128    CC:IMP
PROOF.BAS
PFILE.DAT
    RF:FIX=56    FO:IDX    USED:8:0        RECSI:56     CC:IMP
RDPLN1.DOC
CHTST.BAS
CHTST1.BAS
CHTST1.B2S
CHTST1.OBJ
    RF:VAR=128    FO:SEQ    USED:2:396      RECSI:90     CC:IMP
CHTST1.CMD
CHTST1.ODL
CHTST1.TSK
    RF:FIX       FO:SEQ    USED:8:0        RECSI:512
FILE2.BAS
FILE.DIF
FILE1.BAS
FIL.DIF
INSRT.MAC
    RF:STM       FO:SEQ    USED:3:234      RECSI:63     CC:IMP
INSRT.OBJ
    RF:VAR       FO:SEQ    USED:1:144      RECSI:42     CC:IMP
INSRT.LST
    RF:VAR       FO:SEQ    USED:5:456      RECSI:94     CC:IMP
UBAK01.RNO
    RF:STM       FO:SEQ    USED:121:159    RECSI:121    CC:IMP
UBAK01.DOC
    RF:STM       FO:SEQ    USED:134:377    RECSI:80
RDPLN1.RNO
    RF:STM       FO:SEQ    USED:26:164     RECSI:79     CC:IMP
RDPLN2.RNO
    RF:STM       FO:SEQ    USED:24:475     RECSI:110    CC:IMP
RDPLN2.DOC
    RF:STM       FO:SEQ    USED:26:286     RECSI:80
```

The OA options lists the file's attributes as a series of 6-digit octal numbers below the filename. For example:

```
DIR /OA
SY:[2,211]
RMSTST.B2S
RMSTST.OBJ
 001002 000200 000000 000007 000000 000007 000026 000000 000200
RMSTST.CMD
RMSTST.ODL
RMSTST.TSK
 000001 001000 000000 000222 000000 000223
RMSTST.MAP
 000002 000167 000000 000211 000000 000211 000354
PROOF.BAS
PFILE.DAT
 001041 000070 000000 000007 000000 000010 000000 000001 000070
```

Normally, such a listing shows seven 6-digit octal numbers (called words) for each file that possesses attributes. RMS-created files show nine or ten words. The additional words contain information pertinent to RMS-11. Table 11-5 describes these words and their meanings.

For example, if a directory listing contains an entry as follows:

```
FILE.DAT      7 < 60> 15-Jun-78 15-Jun-78 02:10 PM 8 RSX 0
 001041 000070 000000 000010 000000 000007 000000 000001 000070 000010
```

The first word, 001041, indicates:

000001 – the file has fixed length records  
000040 – the file has indexed organization  
001000 – carriage control is implied

The second word, 000070, indicates that the number of bytes in each record is 70 (octal) (56 decimal).

The third and fourth words, 000000 and 000010, indicate that the file size is 10 (octal) blocks (8 decimal).

The fifth and sixth words, 000000 and 000007, indicate that the first 7 blocks in the file are in use.

The seventh word, 000000, indicates that no bytes are in use in the seventh block.

The eighth word, 000001, indicates that the bucket size is 1 block and the fixed header length is 2 bytes.

The ninth word, 000070, indicates that the record size is 70 (octal) bytes (56 decimal).

The tenth word, 000010, indicates that the file is allocated 10 new blocks (8 decimal) each time it is extended.

**Table 11-5: File Attributes**

Word	Meaning
1	Low order digit - Record Format 00000 - undefined 00001 - fixed length records 00002 - variable length records 00003 - variable with fixed control 00004 - stream format records Second digit - File organization 00000 - sequential organization 00020 - relative organization 00040 - indexed organization Third and fourth digit - Record attributes 000400 - FORTRAN carriage control 001000 - implied carriage control 004000 - no span Fifth and sixth digits - Not used
2	The size in bytes for fixed-length records or the size of the largest record for variable-length records.
3 and 4	A two-word integer that represents the file size. Word 3 is high order, word 4 is low order.
5 and 6	A two-word integer that represents the number of blocks that are currently in use for data. Word 5 is high order, word 6 is low order. Note that words 3 and 5 are non-zero only if the file is larger than 65535 blocks.
7	The number of bytes in use for the last block used. Note that the size of a file with n full blocks can be described as n blocks with 512 characters in the last block or as n+1 blocks with 0 characters in the last block.
8	Low byte      The bucket size in blocks in the range of 1 to 32. If this byte is 0, the bucket size is 1 block. High byte      The fixed header size in bytes in the range of 1 to 255. If this byte is 0, the header size is 2 bytes.
9	The maximum record size in bytes. If the file contains fixed-length records, this word contains the record length. If this word is 0, no maximum size is set for the file.
10	The number of blocks to which the file is extended by default. That is, the content of this word represents the number of blocks that are added to this file each time the file is extended.

File attribute data is processed by RMS-11 and all languages that use RMS-11. BASIC-PLUS and FORTRAN IV do not use RMS-11 and therefore do not correctly process files that have attributes. Thus, programs that are written in BASIC-PLUS or FORTRAN IV, except those that are especially programmed to handle file attributes, cannot be used with files produced by RMS-11. FORTRAN IV-PLUS does use file attributes but does not use RMS-11.

System programs that do handle file attributes correctly include the BACKUP package, the Spooler package, and PIP.

The system program PIP (see Chapter 16) is the only program generally used that preserves file attributes and run-time systems in disk-to-disk transfers within one system. When PIP copies a disk file to a non-disk device, it performs file format conversions, either automatically or as you specify through switch specifications. PIP can also perform file format conversions on a copying operation from a non-disk device to a disk file, and create the appropriate file attributes in the process.

To move files between systems using magnetic tape as the transportation medium, you have the following alternatives:

- to use the RSTS/E programs BACKUP or PIP for transfers between two RSTS/E systems.
- to use the RMS BACKUP package for transfers between a RSTS/E system and a system in the RSX-11 family, when the files involved are produced by RMS-11.

More information on RMS-11 and file attributes can be found in the *RMS-11 User's Guide*. More information on the program PIP can be found in Chapter 16 of this manual.



## **Chapter 12**

### **Program Information and Characteristics**

#### **12.1 The Concise Command Language (CCL)**

RSTS/E allows you to run some of the system library programs by typing a unique system command called a Concise Command Language (CCL) command. The number of programs that can be run by CCL commands is generally decided by the system manager, although Digital supplies CCL commands with some of the library programs. These standard CCL commands are listed in Chapter 10. Specific descriptions of Digital-supplied CCL commands are found in the chapters describing their associated programs.

The chief advantage of CCL commands is that they allow you to call a system program with one brief command (by typing PIP, for example, instead of RUN \$PIP), and to give that program a specific command to execute: QUE MYFILE.DAT, for example, calls the QUE program and also causes it to print MYFILE.DAT at the line printer.

##### **12.1.1 Cautionary Notes on Typing CCL Commands**

**12.1.1.1 Embedded Spaces In CCL Commands** — The CCL translator, unlike the BASIC-PLUS translator in NO EXTEND mode, interprets spaces embedded in strings as significant. Thus, an attempt to run the EDIT program with the typed command E DIT, will not succeed but will cause an error message.

**12.1.1.2 Mistyping a BASIC-PLUS Command as a CCL Command** — When you enter a line at command level, BASIC-PLUS first checks for a legal line number. If BASIC-PLUS finds a line number, it compiles the line as a program statement. But if it finds no line number, it sends the line to the CCL translator; if the CCL translator recognizes the line as a valid CCL command, it causes the appropriate program to be run, thus destroying the current contents of your memory area. If, on the other hand, the CCL translator does not recognize the line as a valid CCL command, it sends the line back to the BASIC-PLUS Run-Time system, which then attempts to execute the line as a BASIC-PLUS system command or immediate mode statement.

Thus, when you issue a BASIC-PLUS system command or immediate mode statement, be careful not to mistype that command or statement so that it mimics a valid CCL command: if it does, your memory area (i.e., the current program) will be destroyed and replaced by a system library program.

It is unlikely that a BASIC-PLUS statement or command would be mistyped as any of the Digital-supplied CCL commands. Some systems, however, may have CCL commands of their own that do resemble BASIC-PLUS statements or commands. Therefore, become familiar with the Concise Command Language as it exists on your system.

## 12.2 Obtaining Help Files for System Programs

Many of the library programs offer you assistance in the form of help files; these files contain information on running the program and using its commands and/or options, and may be printed at your terminal. Generally, the simple, one-command programs such as INUSE, GRIPE, MONEY, and QUOLST do not provide help files.

The usual method of requesting a help file depends upon the program, but generally you invoke the program and type the /HELP or /HE option. Refer to individual program descriptions in Part IV.

## 12.3 Version Identification

Most system library programs, on being called by the RUN \$ command or by a CCL command that merely calls the program, print a program header before the prompt. This header contains the program name and the RSTS/E version number, along with other information such as the system number and the current job number.

The following output, for example, is the QUE program's response to either the RUN \$QUE command or the CCL command QUE:

```
QUE V7.0 RSTS V7.0 TIMESHARING  
*
```

For other examples of program headers, see the individual descriptions in Part IV.

## 12.4 Indirect Command Files

Three system programs, BACKUP, FILCOM, and PIP, can be run by indirect command files. You create an indirect command file that contains commands which the program executes as it reads the file. Indirect command files are useful when a sequence of operations is to be performed repeatedly; once you have placed the commands for these operations in a file, you need not issue them one by one when you wish to perform the operations.

Methods of creating and running indirect command files vary according to program. BACKUP, which runs by an ordered dialogue, involves methods different from those of PIP, which does not. See the individual program descriptions in Part IV.

## 12.5 Obtaining Help: The HELP Program

The system library program HELP consists of files that contain descriptions of other system programs and system resource commands. HELP is an interactive program that prompts you for topics and subtopics. To obtain information on system programs or commands, you type the desired topic and optional subtopic(s) in response to the HELP prompts. Your response causes HELP to display the contents of the appropriate file.

To invoke the HELP program, type:

```
RUN $HELP
```

When successfully invoked, HELP prints an identifying header line and a prompt for the desired topic. For example:

```
RUN $HELP  
HELP V7.0 RSTS V7.0 Timesharing  
Topic?
```

In response to the Topic? prompt, type one of the following:

1. The name of the system program or resource command you wish information on. If you abbreviate the name, information on all topics that match that abbreviation are displayed. In addition to the program or command, you can specify one or more of the subtopics (separated by spaces) associated with that program or command.
2. The RETURN key, which causes HELP to print descriptive text on its use and a list of possible topics.
3. An asterisk wildcard (\*), which causes HELP to print information on all of the available topics.

You can also invoke the HELP program with a CCL command, as follows:

```
HELP  
  
Help can be obtained on a particular topic by typing:  
      HELP topic subtopic subsubtopic...  
A topic can have the following format:  
      1) an alphanumeric string (e.g., a command name, option, etc.)  
      2) same preceded by a "/" (=> interpreted as a switch)  
      3) the match-all symbol "*"  
Examples:  
      HELP DIRECTORY /S  
      HELP SET STALL  
Abbreviations result in all matches being displayed.  
Additional information is available on:  


|          |           |
|----------|-----------|
| /OUTPUT  | /PROMPT   |
| ASSIGN   | ATTACH    |
| DISMOUNT | DIRECTORY |
| HELP     | HELLO     |
| PIP      | QUE       |
| RSX      | RUN       |
| TECO     | TYPE      |
| BASIC    | BYE       |
| EXIT     | FILENAMES |
| KEYBOARD | LOGIN     |
| REASSIGN | RT11      |
| SET      | SWITCH    |
| VTEDIT   | SYSTAT    |


```

Topic?

When HELP is invoked with the CCL command, it displays descriptive text on its use and a list of possible topics as well as the Topic? prompt (i.e., as if you had typed a RETURN key in response to the Topic? prompt).

By default, HELP displays the requested information on your terminal. To override the default, you can specify the /OUTPUT: switch that causes HELP to print the requested information to a user file. The switch has one of the following formats:

```
HELP/OUTPUT:filespec topic [subtopic [...]]
```

or

```
HELP/OUTPUT:filespec * [...]
```

where filespec can be a complete RSTS/E file specification as described in Chapter 11. A filename is required; HELP defaults the extension to .LST. Note that the /OUTPUT: switch can only be used with the HELP CCL command. Consider the following example:

```
HELP/OUTPUT:INFO SYSTAT WHO
```

causes the HELP program to create a file in your account on the public structure named INFO.LST. This file contains a copy of the HELP file on the SYSTAT subtopic WHO.

If you attempt to specify a HELP topic for which no file exists, HELP prints:

```
SORRY, NO INFORMATION AVAILABLE ON topic
```

followed by information on valid topics (as if you had typed HELP and a RETURN key or a RETURN key in response to the Topic? prompt).

Many of the topics described by HELP are further explained in subtopics. For example:

```
Topic? SYS
```

```
SYSTAT
```

```
The SYSTAT program provides current system information. It may be run with the RUN command or the CCL command, "SYS[TAT]". The command string is of the form:
```

```
[output file] [/options]
```

```
Only logged-in users may specify an output file. If none is specified, the output will be to the keyboard. If no options are specified, the status of jobs, devices, disks, buffers, run-time systems, resident libraries and message receivers will be reported.
```

```
Examples:
```

```
SYSTAT
```

```
SYS/A
```

Additional information is available on:

/A	/B	/D	/F
/J	/Kn	/L	/M
/N	/P	/PROJ,PROG	/PROJ,*
/R	/S	/U	/O,O
/-			
JOB	WHO	WHERE	WHAT
SIZE	STATE	SWAPPING	RUN-TIME
RTS			

SYSTAT Subtopic?

In this example, the initial description of the SYSTAT program (abbreviated to SYS) makes note of additional information available on various options. To cause HELP to print the additional information, type the desired subtopic(s) in response to the HELP Subtopic? prompt. For example:

SYSTAT Subtopic? WHO

SYSTAT

WHO

The "Who" column in the job status report gives the account under which the job is running. This column will contain one of the following:

nn,mm	the job is running under account [nn,mm],
[OPR]	the job is running under a system operator account,
[SELF]	the job is running under your account,
**,**	the job is not logged in.

SYSTAT Subtopic? ^Z

Topic? ^Z

Ready

The HELP program continues to prompt for subtopics until you type CTRL/Z, which causes HELP to return to the Topic? prompt; another CTRL/Z terminates the program.

A topic and optional subtopics can be specified with the HELP CCL command. In this case, HELP displays the requested information and exits. For example:

HELP SYS WHO

SYSTAT

WHO

The "Who" column in the job status report gives the account under which the job is running. This column will contain one of the following:

nn,mm	the job is running under account [nn,mm],
[OPR]	the job is running under a system operator account,
[SELF]	the job is running under your account,
**,**	the job is not logged in.

Ready

However, if you wish to use the CCL command to display information but not exit upon completion, use the /PROMPT switch as follows:

```
HELP/PR[PROMPT] topic
```

This switch (as with /OUTPUT) can only be used with the CCL command. Also, when /PROMPT is used with no topic, it causes HELP to inhibit the display of HELP use and topic text and print only the Topic? prompt. For example:

```
HELP/PR EXIT  
EXIT  
Exit to the job default run-time system  
Topic?
```

# **Chapter 13**

## **Job Control Programs**

### **13.1 Entering the System: The LOGIN Program**

The LOGIN system program runs from either a logged in or a logged out terminal. It activates a job at a terminal, attaches a detached job\* to a terminal, or runs designated system programs from a logged out terminal.

#### **13.1.1 Running LOGIN from a Logged Out Terminal**

As described in Section 1.2, the LOGIN system program runs when either HELLO, LOG, LOGIN, ATTACH, ATT or I is typed at a terminal connected to the RSTS/E system. This section describes more fully the actions which occur when LOGIN runs at a logged out terminal.

When a terminal is connected to the RSTS/E system by a dial-up connection, the automatic answering signal causes the system Monitor to insert an I in the input buffer for that terminal. This action simulates an I being typed at a terminal directly connected to the system. The description of the resultant actions in this section applies to the cases of a terminal directly connected to the system and of a terminal connected by a dial-up line.

When you enter typed characters to the system from a terminal directly connected to but not logged into the system, the RSTS/E Monitor runs the LOGIN system program which checks the characters for a valid command. If only the RETURN key is typed, the system takes no action. If you enter either SYS or SET commands, it causes LOGIN to chain to the SYSTAT or TTYSET system programs, respectively. The system manager can alter the LOGIN program to run other programs in the same manner.

If you type HELLO, LOGIN, LOG, ATTACH, ATT, or I, LOGIN prints the system identification line as in the following sample printout:

```
RSTS V7.0 Timesharing Job 12 KB16 21-Sep-78 08:37 AM  
#120,80
```

---

\*A job becomes detached because the connection of a dial-up line is broken or a privileged job executed the SYS system function to detach the job from the terminal.

The line contains the system name and version number, the local installation name, the job number activated, the keyboard number of the terminal, and the current system date and time. The pound sign (#) character printed by LOGIN on the following line requests you to type your account number.

Type the project and programmer numbers separated by either a comma or a slash and terminated with the RETURN key. (Typing the slash as a separator inhibits printing of any system notice messages.) You can specify the project and programmer numbers on the same line as the HELLO, LOGIN, LOG or I commands as shown in the following sample dialogue:

```
HELLO 120,80  
Password:
```

When an account number is included in the command, LOGIN does not print the # character but immediately prompts you to enter the password. LOGIN disables echo printing at the terminal when the password is typed.

If either the account does not exist or the password does not match, LOGIN prints the INVALID ENTRY - TRY AGAIN message and the # prompting character. You can try the sequence to a maximum of five times. LOGIN allows 30 seconds in which to type an entry. After the fifth invalid entry, LOGIN prints the ACCESS DENIED message and frees the job for other usage.

A valid entry causes LOGIN to check for any other jobs which may be running on the system under the same account number. If other jobs are running and none are detached, LOGIN reports how many such jobs by printing a message similar to the following sample and prints the system notice messages:

```
1 other user is logged in under this account
```

If any jobs are running detached under the current account, LOGIN instead reports the number of each such job and requests you to type the number of the job to be attached to the terminal. The following sample printout shows the procedure:

```
Job 1G is detached under this account  
Job number to attach to? 1G  
Attaching to Job 1G
```

To attach a job to the terminal, type its number in response to the query. LOGIN prints the ATTACHING TO message and attempts to attach the specified job to the current terminal. When the job is attached, the current job is freed for other usage and the attached job runs at the terminal.

To continue running the current job, type the RETURN key in response to the query. LOGIN subsequently prints the message concerning other jobs running under the same account and prints the system notice messages, if any:

```
Job 1G is detached under this account  
Job number to attach to? RET  
2 other users are logged in under this account
```

System notices convey information which the system manager places in the file NOTICE.TXT in the system library. If the file does not exist, LOGIN proceeds. LOGIN exits to the system default run-time system, which clears the LOGIN program from memory, and prints a prompt (usually READY).

The complete sequence to log a job into the system when other jobs are running detached is shown below:

```
HELLO 1/210
Password:
Job 16 is detached under this account
Job number to attach to? RET
2 other users are logged in under this account

Ready
```

The complete sequence to attach another job to the terminal when logging into the system is shown in the following sample printout:

```
HELLO

RSTS V7.0 Timesharing Job 11 KB2 02-Nov-78 11:31 AM
#1/210
Password:
Job 16 is detached under this account
Job number to attach to? 16
Attaching to job 16

Ready
```

To attach a job to a terminal when the job number is known, type the ATTACH or ATT command as follows:

```
ATTACH

RSTS V7.0 Timesharing Job 13 KB2 02-Nov-78 12:01 PM
Job number to attach to? 34
Attaching to job 34

Ready
```

LOGIN runs and prints the system identification line and, on the next line, the JOB NUMBER TO ATTACH TO query. You must type the number of the detached job. If the job is not detached or does not exist, LOGIN prints an appropriate message followed by ACCESS DENIED. You must type another command to log into the system.

If the job is detached, LOGIN prompts you for the password of the account under which the detached job is running. After you enter the password, LOGIN prints the ATTACHING TO JOB x message and attempts to attach the specified job to the terminal. An incorrect password causes LOGIN to print the FAILURE TO ATTACH TO JOB x message and to terminate as shown in the following sample printout:

```
ATTACH

RSTS V7.0 Timesharing Job 13 KB2 02-Nov-78 12:03 PM
Job number to attach to? 21
Password:
Attaching to job 21
Failure to attach to job 21
```

You must try again. If the system successfully attaches the job to the terminal, the terminal becomes the console terminal of the job. Further terminal output is under programmed rather than system control.

To omit the printing of the identification and the query lines, type the job number on the same line as the ATTACH or ATT command as follows:

```
ATT 27  
Password:  
Attaching to Job 27  
  
Ready
```

The READY message indicates that the attached job is at the system monitor level.

### 13.1.2 Running LOGIN at a Logged In Terminal

If you type the HELLO or the ATTACH command at a terminal already logged into the system, the LOGIN system program is loaded into your job area and is started. Note that to use the ATTACH command at a logged in terminal, ATTACH must be installed as a CCL command on the system. The previous contents of your area are destroyed. LOGIN prints the system identification line with one additional item inserted. Between the job number and the keyboard number printed on the line, LOGIN inserts the project-programmer numbers of the account under which the current job is running. Typing the LOGIN, LOG, ATT or I command at a terminal already logged into the system causes the system monitor to print the ?WHAT? error message and the READY message, unless these commands have been installed as valid CCL commands.

LOGIN determines if any other jobs are running under the same account and prints the message informing you of the number of those jobs. The following sample dialogue shows the procedure:

```
Ready  
  
HELLO  
  
RSTS V7.0 Timesharing Job 15 [1,210] KB3 02-Nov-78 02:53 PM  
1 other user is logged in under this account  
  
Ready
```

If any such jobs are running detached, LOGIN also prints the message informing you of the number of each such job and, on the following line, prints the ATTACH TO query as follows:

```
Ready  
  
HELLO  
  
RSTS V7.0 Timesharing Job 27 [1,210] KB2 02-Nov-78 01:02 PM  
Job 15 is detached under this account  
Job number to attach to? 40  
No job by that number - try again  
Job number to attach to?
```

To attach one of the jobs to the terminal, type one of the job numbers reported in the message. LOGIN determines whether the job is nonexistent or whether it is already attached to another terminal. In either case, the program prints an appropriate error message saying try again and subsequently reprints the ATTACH TO query.

To continue running the current job, type the RETURN key in response to the ATTACH TO query. As a result, LOGIN prints the information message telling how many other jobs are running under the same account and prints the READY message. The system clears the LOGIN program out of memory:

```
Job number to attach to? RET
2 other users are logged in under this account
Ready
```

When you respond to the ATTACH TO query by typing one of the job numbers reported in the message, the program proceeds as shown in the following sample printout:

```
HELLO

RSTS V07.0 Timesharing Job 36 [1,210] KB3 02-Nov-78 02:55 PM
Job 15 is detached under this account
Job number to attach to? 15
Attaching to Job 15

Ready
```

The READY message indicates that the new job is at system command level.

To attach to a job known to be running detached under the same account, type the job number on the same line as the ATTACH command (ATTACH must be installed as a CCL command). The LOGIN program determines if the specified job exists and is detached. If not, it prints an appropriate error message and the ATTACH TO query. You can type another job number or the RETURN key. If the job exists and is detached, LOGIN compares the account numbers under which both the current job and the detached job are running. If the accounts are different, the program prompts you for the password of the account under which the detached job is running. The following sample printout shows the procedure:

```
ATTACH 51
No job by that number - try again
Job number to attach to? 15
Password:
Attaching to Job 15
Failure to attach to Job 15

Ready
```

After you enter the password, the program prints the ATTACHING TO message and attempts to attach the detached job to the terminal. If the password is not valid, the program prints the FAILURE TO ATTACH and the READY messages and exits to the Monitor, which clears the LOGIN program from your job area.

When the account numbers of the two jobs are the same, LOGIN omits the PASSWORD prompt message and attaches the job as shown below:

```
ATTACH 24
Attaching to Job 24

Ready
```

The READY message indicates that the job is at the system monitor level.

To change accounts without logging off the system, type the HELLO command followed by the account number. For example:

```
Ready
HELLO
RSTS V7.0 Timesharing Job 20 [2,227] KB2 02-Nov-78 05:16 PM
Ready
HELLO 1/210
Password:
Ready
HELLO
RSTS V7.0 Timesharing Job 20 [1,210] KB2 02-Nov-78 05:17 PM
Ready
```

To have the system print the system notice message, replace the / character in the account number with a comma.

#### NOTE

When you attach to a detached job, the detached job becomes the current job. Any devices that were assigned to the detached job as well as the devices assigned to the attaching job are now assigned to the current job. Also, when you change accounts without logging off the system, all devices assigned to the job remain assigned. Similarly, all logical device names, default protection codes, and project-programmer numbers that were assigned to the job, remain assigned. When you log off the system, all assignments are deassigned.

### **13.1.3 Running Other Programs from a Logged Out Terminal**

Certain commands typed at a logged out terminal cause LOGIN to chain to another program in the system library. The system manager can modify the LOGIN program to recognize other commands and to chain to a program stored in the system library.

For example, it is convenient to determine job status without logging a job into the system. The following printout shows the procedure:

```
SYS/4
4      [1,210] KB25      NONAME     16K     SL      0.1   BAS4F
Bye
```

LOGIN runs and recognizes the SYS command of the SYSTAT system program. LOGIN writes the option given in the command in the core common area and chains to the SYSTAT program at line 32000. SYSTAT reads the option from the core common area and prints the appropriate report. It then exits to the Monitor, which prints the BYE message and clears the contents of memory.

## **13.2 Leaving the System: The LOGOUT Program**

LOGOUT is called when you have completed all processing and are ready to leave the terminal. The LOGOUT program is started when the BYE command is typed at a user terminal logged into the RSTS/E system. LOGOUT checks your current disk quota to ensure that you do not log out of the system with more than the acceptable amount of disk storage being used for your files. If your disk files are within the acceptable disk quota size, LOGOUT logically disconnects the terminal from the system, removes the current job number from the list of active jobs and prints some information on the duration of the current job.

In response to the BYE command, LOGOUT prints:

CONFIRM:

You can type any of the responses shown in Table 13-1.

In individual deletion mode, LOGOUT prints the name, size, protection code, and creation date of each file stored under your current account number on the system disk. This information is followed by a ? after which the system awaits a response from you which can be:

<b>File Deletion Mode Response</b>	<b>Meaning</b>
RETURN key	Save the file just listed.
K	Delete (kill) the file just listed.

**Table 13-1: LOGOUT CONFIRM: Responses**

CONFIRM: Response	Meaning
Y	The system performs the checks described above. If successful, the LOGOUT messages are printed. If not successful, an error message is printed and you must delete some files.
N CTRL/C	These responses indicate that you do not want to log out of the system. The LOGOUT procedure is terminated without logging you off the system and the system prints the READY message.
?	Causes LOGOUT to print an explanation of the acceptable responses to CONFIRM:.
RETURN key	Causes LOGOUT to print a message instructing you to type ? to obtain a description of logout procedures.
I	Causes LOGOUT to enter individual file deletion mode.
F	Causes a fast logout procedure if your disk storage space is within acceptable limits.
Other	Same as RETURN key.

An example of a LOGOUT sequence is shown below:

```

BYE
Confirm: Y
Disk quota of 400 exceeded by 52 blocks
Some file(s) must be deleted before logging out
Type '?' for help
Confirm: I
DATA      .001      200      60      03-Nov-76 ?
DATA      .002      150      60      03-Nov-76 ?
DATA      .003      100      60      03-Nov-76 ? K
Confirm: Y
Saved all disk files; 352 blocks in use, 48 free
Job 24 User 100,101 logged off KB3 at 03-Nov-76 03:09 PM
System RSTS V7.0 Timesharing
Run time was 1.4 seconds
Elapsed time was 3 minutes, 59 seconds
Good afternoon

```

You can omit the CONFIRM: message by typing the BYE command and the response to the CONFIRM: message. For example, to perform a fast logout, type:

```
BYE F
```

The LOGOUT program runs and performs the fast logout procedure by printing a series of LINE FEED characters instead of printing the final accounting information. If your job exceeds the acceptable limit for disk storage, LOGOUT prints the QUOTA EXCEEDED message and the CONFIRM: message to allow you to delete some files before logging out.

# **Chapter 14**

## **System Communication Programs**

### **14.1 Printing a System Status Report: The SYSTAT Program**

The SYSTAT program provides current system information in the areas of job, device, disk, buffer status, run-time systems, resident libraries, and message receivers. You can call SYSTAT while logged into the system or from a terminal which is on-line but not logged into the system.

To start SYSTAT while logged into the system, type:

```
RUN $SYSTAT
```

If you are not logged into the system, type:

```
SYSTAT
```

which can be abbreviated to SYS.

If you are already logged in, the system responds by printing:

```
OUTPUT STATUS TO?
```

at which point you can indicate any RSTS/E device or a filename specification for the status report output. Possible replies when logged into the system are described below:

<b>SYSTAT Output Response</b>	<b>Meaning</b>
LP:	send status report to the line printer if only one line printer is on the system or to line printer unit 0 if multiple line printers are on the system.
LPn:	send status report to line printer unit n if that printer is not currently in use.

SYSTAT Output Response	Meaning
KB:	send status report to your terminal. (The RETURN key is equivalent to responding KB:.)
KBn:	send status report to terminal n in the system if that terminal is on-line and not currently in use.
PP:	send status report to the high-speed paper tape punch.
dev:filename.ext	send the status report to the file specified. The default device is the system device. No extension is appended unless specified.
?	send status report to a file on the public structure, and print the filename. The file is named according to the current date and time of day; its extension is .RPT. (The filename is explained in the following paragraph.)

An example of an output file created by the ? response to the OUTPUT TO query is F04N59.RPT. The filename has 4 parts. The first is a letter from A to L, and denotes the month according to its alphabetical position; here, F, the sixth letter of the alphabet, denotes June. The second part is two digits from 01 to 31 denoting the day of the month; here, the fourth. The third part is a letter from A to X denoting the hour from 00 to 23; here, N denotes the 14th hour (2:00 p.m.). The fourth part is two numbers denoting the minute of the hour.

If SYSTAT is run while you are not logged into the system, the report is always sent to the user terminal requesting the report.

Following the device or filename specification, you can specify one of the options in Table 14-1 to obtain a partial system status report. The option specifications are preceded by a slash if you are logged into the system. The options can be typed following the SYS command if you are not logged into the system.

**Table 14-1: SYSTAT Options**

SYSTAT Option Specification	Meaning
/A	Report only status of attached jobs.
/B	Report only busy device status.
/C	Report memory allocation on system (privileged).
/D	Report only disk status.
/F	Report only free buffer status.
/Kn	Report only job status of terminal n in the system.
/L	Report only resident library status.
/M	Report only message receiver status.

(continued on next page)

**Table 14-1: SYSTAT Options**

SYSTAT Option Specification	Meaning
/N	Report only status of non-privileged accounts.
/n	Report status of job n only.
/n,m	Report status of account [n,m] only.
/n,*	Report status of jobs with project number n only.
null	Report complete system status to include job, run-time system and resident library status, busy device, disk structure, free buffer status, and message receiver statistics.
/P	Report only status of privileged accounts.
/R	Report only run-time system status.
/S	Report only job status.
/U	Report only status of unattached (i.e., detached) jobs.
/O,0	Report only status of jobs not logged into the system.
/O	Report all open files (privileged).
/O:dev	Report all files open on a specific device (privileged).
/W	Report all open files and the jobs accessing them (privileged).
/W:dev	Report all open files and the jobs accessing them for a specific device (privileged).

A minus sign (-) may be included with any option, in any position following the slash, to cause printing of an account number instead of [OPR] and [SELF].

The options S,A,B,D,F,N,M,L,O,C,W,R,P and U can be specified as separate options or in any combination. If multiple options are specified, only one slash is required. The options /O, /C, and /W can only be specified by a privileged user; the options /O and /W can only be specified on a system with large file capability (i.e., a system that supports files greater than 65535 blocks).

The following examples are performed on a terminal logged into the system:

RUN \$SYSTAT OUTPUT STATUS TO? STAT	creates complete system status report in the file STAT under the current account in the public structure.
RUN \$SYSTAT OUTPUT STATUS TO? LP: /3	causes output of a status report for job 3 to the line printer.
RUN \$SYSTAT OUTPUT STATUS TO? /D	causes output of disk status report to the user terminal.
RUN \$SYSTAT OUTPUT STATUS TO? /SF	causes output of job and free buffer status to the user terminal.

The following examples are performed on a terminal not logged into the system:

- SYS causes output of complete system status report to the terminal.
- SYS/D causes output of the disk status report to the terminal.
- SYS/BF causes output of the busy device and free buffer status reports to the terminal.
- SYS/5 causes output of a status report for job 5 to the terminal.
- SYS A/B causes the ILLEGAL OPTION error since SYSTAT cannot create a file for a logged out job.
- SYS /AP causes output of a report of all attached, privileged jobs.

#### 14.1.1 Contents of the Status Report

A complete system status report is shown below:

```
RUN $SYSTAT
SYSTAT V7.0 RSTS V7.0

Output Status to? RET

RSTS V7.0 status at 28-Aug-78, 10:49 AM Up: 10:11:04

Job Who Where What Size State Run-Time RTS
 1 [OPR] Det ERRCPY 5K SR D56 12:17.4 BAS4F
 2 [OPR] Det OPSRUN 16K SL D62 5:43.1 BAS4F
 3 [OPR] Det QUMRUN 16K SL D58 32:26.8 BAS4F
 4 [OPR] Det SPLIDL 16K RN 4.1 BAS4F
 5 [OPR] Det SPLIDL 16K SL D61 0.4 BAS4F
 6 [OPR] Det BATIDL 13K SL D49 0.1 BAS4F
 7 [OPR] Det BATIDL 13K SL D48 0.0 BAS4F
 8 [OPR] Det BATIDL 13K SL D60 0.0 BAS4F
 9 [OPR] Det NCU 14K SR D50 7.0 BAS4F
10 [OPR] Det NPKDVR 8K SR D63 14.4 BAS4F
11 [OPR] Det DYN_ZR 2K SL D57 48:34.9 RSX
12 [OPR] Det V52DPY 16K SL D52 5:01:48.9 BAS4F
13 1,203 KB39 SPEED 8K KB 3:58.6 BASIC2
14 1,224 KB44 QUE 14K KB A02 6:15.3 BAS4F
15 1,253 KB38 ERRDET 9K RN A06 8:16.9 BAS4F
16 1,251 KB32 ALARM 3K SL D51 4:38.8 BAS4F
17 1,225 KB30 VTEDIT 15K KB A14 9:58.6 TECO
18 1,243 KB29 VTEDIT 15K KB 14.7 TECO
19 [SELF] KB68 SYSTAT 9K RN Lock 2.4 BAS4F
20 1,250 KB33 VTEDIT 15K KB A15 4:02.4 TECO
21 1,241 KB45 QUE 14K ^C A01 18.6 BAS4F
22 1,200 KB22 NONAME 2K ^C A11 18.5 BAS4F
23 1,201 KB43 ...TKB 14K RN 5:38.7 RSX
24 1,202 KB19 VTEDIT 16K KB A03 5:16.1 TECO
25 1,246 KB18 NONAME 2K ^C A12 14.3 BAS4F
26 1,218 KB21 VTEDIT 19K KB 1:08.1 TECO
27 1,243 KB40 LOGOUT 5K RN A08 2:32.5 BAS4F
28 1,244 KB41 VTEDIT 15K RN A03 1:29.4 TECO
29 1,222 KB26 VTEDIT 15K KB A13 3:11.3 TECO
30 1,242 KB31 SPEED 8K KB A05 14.9 BASIC2
31 1,234 KB72 SPEED 8K RN A03 53.9 BASIC2
```

**Busy Devices:**

Device	Job	Why
KB70	12	INIT
XMO	NSP	AS+INIT

**Disk Structure:**

Disk	Open	Free	Cluster	Errors	Name	Comments
DS0	7	0	8	1	SWAPO	Pri, DLW
DS1	1		1	0		Pri, NFS
DB0	4	5472	8	0	VO6C	Pri, DLW
DB1	47	20184	8	0	ARK	Pub, DLW
Small	Large	Jobs	Hung	TTY'S		Errors
40	1	31/63		0		158

**Run-Time Systems:**

Name	Ext	Size	Users	Comments
BAS4F	BAC	1G(16)K	19	Perm, Addr:41, KBM, CSZ
TECO	TEC	7(24)K	7	TEMP, Addr:107
RT11	SAV	4(28)K	0	Non-Res, KBM, CSZ, EMT:255
RSX	TSK	3(28)K	2	TEMP, Addr:253, KBM
RMS11	TSK	4(28)K	0	Non-Res
BP2COM	TSK	4(28)K	0	Non-Res, KBM
BASIC2	TSK	1G(16)K	3	TEMP, Addr:214

**Resident Libraries:**

Name	Prot	Acct	Size	Users	Comments
RMSRES	<42>	[0,1]	23K	0	Non-Res:Addr:66

**Message Receivers:**

Name	Job	Miss	Max	Senders
ERRLOG	1	0	40	Priv
OPSER	2	0	30	Local
QUEMAN	3	0	60	Local
LPOSPL	4	0	5	Priv
LP1SPL	5	0	5	Priv
BAOSPL	6	0	5	Priv
BA1SPL	7	0	5	Priv
BA2SPL	8	0	5	Priv
NCU	9	0	5	Priv, Network
NWPK10	10	0	5	Network

The job status information includes a list of all active jobs by job number, the account number under which each job runs, the keyboard involved (the keyboard number is followed by an asterisk if the job logged on by means of a dial-up line), the program name and size, the job state, the total amount of central processor run time exhausted, and the run-time system. If a job is running with temporary privilege, a plus sign (+) is appended to the job number; if a job has temporarily dropped its temporary privilege, a minus sign (-) is appended to the job number. In the WHO column, SYSTAT substitutes OPR for the project-programmer number to denote an operator account. An operator job has a project number of 1 and a programmer number between 1 and 200. In the WHERE column, DET appears in place of the keyboard number for jobs which run detached from a keyboard. Also, the abbreviation PxJy can appear for a job running on a pseudo keyboard. The value Px identifies pseudo keyboard unit x; and the value Jy denotes job number y, under which the controlling job is running. The SIZE column shows the current size of the job. The STATE column contains an abbreviation (see Table 14-2) telling the current condition of the job. The RUN-TIME column gives

hours, minutes, seconds, and tenths of seconds of central processor time the job has consumed. The RTS column gives the name of the run-time system under which the job is running.

The busy device status information reports devices which are assigned or opened by a specific user. Items reported are the device specification, the job owning that device, and the condition of the device (in the WHY column). Assigned disk units are reported in the disk status information.

The disk status information describes each disk in use on the system. Items reported are: disk name (device specification), number of open files, number of free 512-byte blocks, pack cluster size, disk hardware error count, the pack identification or system logical name (if any) assigned for the device, and comments on its status. Disks assigned to jobs are reported as non-file structured and private and can include the job number and the designation "Dirty" (if appropriate). See Table 14-2 for abbreviations in the COMMENTS column.

The open files report lists the device, account, filename and extension, and protection code of each file opened on the system or specified device. The report also lists the number of jobs which have the file opened (or opened in read regardless mode), the cluster size of the file, and comments on the status of the file. Refer to Table 14-2 for an explanation of comments. Note that if the /W switch is used (see Table 14-1), the report also lists the jobs that are accessing the file.

The buffer status provides the following information: 1) the number of small (16-word) and large (256-word) buffers not currently in use, 2) the number of jobs currently running and the maximum number allowed to run (two numbers separated by a slash), 3) the total number of errors logged on the system, and 4) a count of the number of times a hung terminal was found. A hung terminal is one that fails to respond to character transmission within a given time period.

The run-time system information gives the name of each run-time system, the default extension for (executable) files created by that run-time system, the size of the run-time system in K words, the number of user jobs currently executing under its control, and comments regarding its status. See Table 14-2 for abbreviations in the COMMENTS column.

The resident library information gives the name of the resident library, its protection code and account number, its size, the number of users accessing that library, and comments concerning its status. See Table 14-2 for abbreviations in the COMMENTS column.

The message receiver report gives the receiving job's name and number, the number of messages queued for the job, and the declared maximum number of messages the job can have queued. It also tells whether local and network senders are allowed, whether the job should handle only one link, and whether local senders must be privileged.

The abbreviations used in the SYSTAT report are defined in Table 14-2.

**Table 14-2: SYSTAT Abbreviations**

Abbreviation	Meaning
<b>job status (states)</b>	
DET	job is detached from all terminals.
**,**	job is not logged into the system.
OPR	job runs under a system operator account.
SELF	job runs under your account.
RN	job is running or waiting to run.
RS	job is waiting for residency.
BF	job is waiting for buffers (no space is available for I/O buffers).
SL	job is sleeping (SLEEP statement).
SR	job is sleeping and is a message receiver.
FP	job is waiting for file processing action by the system (opening or closing a file, file search).
TT	job is waiting to perform output to a terminal.
HB	job is detached and waiting to perform I/O to or from a terminal.
KB	job is waiting for input from a terminal.
^C	job is in CTRL/C state, awaiting input to the run-time system.
CR	job is waiting for card reader input.
MT, MM, or MS	job is waiting for magnetic tape I/O.
LP	job is waiting to perform line printer output.
DT, or DD	job is waiting for DECtape I/O.
PP	job is waiting to perform output on the high-speed paper tape punch.
PR	job is waiting for input from the high-speed paper tape reader.
DK,DM,DB,DS, DP,DL,DF,DR	job is waiting to perform disk I/O.
DX	job is waiting for flexible diskette I/O.
RJ	job is waiting for RJ2780 I/O.
??	job's state cannot be determined.
<b>The following status descriptions may appear after one or more of the other job state abbreviations:</b>	
Lck	job is locked in memory for the current operation.
Nsw	job has requested that it not be swapped from memory and cannot be swapped unless it requests additional memory.
Swi	job is currently being swapped into memory.
Swo	job is currently being swapped out of memory.
Xnn	job is swapped out and occupies slot nn in swapping file X; file is denoted by A,B,C, or D to represent files 0 through 3 of the swapping structure.

(continued on next page)

**Table 14-2: SYSTAT Abbreviations**

Abbreviation	Meaning
<b>busy device status</b>	
AS	device is explicitly assigned to a job.
INIT	device is open on a channel.
DOS	magnetic tape is assigned with DOS labeling format.
ANSI	magnetic tape is assigned with ANSI standard labeling format.
<b>disk status</b>	
PUB	cartridge or pack is public.
PRI	cartridge or pack is private.
NFS	disk is open as non-file structured device.
R-O	disk unit is read-only (write-locked).
DLW	date of last write (modify), rather than date of last access, is stored in file accounting entries.
Lck	disk is in a locked state.
NFF	new files on this disk put at beginning of directory.
Job n	private disk is assigned to job n (reported with PRI and NFS).
Dirty	disk needs cleaning (reported with PRI and NFS).
<b>run-time system and resident library status</b>	
Non-Res	run-time system or library is non-resident.
Loading	run-time system or library is being loaded into memory.
Temp	run-time system or library will be removed from memory when not being used.
Perm	run-time system or library will stay in memory when not being used.
Addr:xxx	denotes the run-time system's or library's starting address.
KBM	run-time system can serve as keyboard monitor.
IUS	run-time system or library can serve only 1 user.
R/W	run-time system or library allows read/write access.
NER	errors occurring within run-time system or library will not be sent to system error log.
Rem	run-time system or library will be removed from memory as soon as all its jobs switch to another run-time system or library.
CSZ	proper job image size (in K words) to run a program can be computed as K-size=(filesize+3)/4.
EMT:yyy	denotes the EMT code for special EMT prefix.
<b>message receiver status</b>	
Local	local senders are allowed for this receiver ID.
Priv	local senders must be privileged to send to this receiver ID.
Oneshot	user has indicated that job should handle only one network link.
Network	network senders are allowed for this receiver ID.

Note that when a privileged user runs SYSTAT, the listing contains additional information as follows:

1. The SIZE column includes a number preceded by a slash that indicates the size to which the job can expand.
2. A column headed Pri/RB prints a set of numbers that indicate the job's priority and run burst.
3. If the system supports large files and the /O option is specified:

Abbreviation	Meaning
<b>open files</b>	
Pla	file is placed.
Upd	file is open in UPDATE mode.
Tent	Tentative file.
MDL	file is marked for deletion.
Ctg	file is contiguous.
NoK	file can not be renamed or deleted.
UFD	file is a UFD-type entry.
None	no files are open on the disk.
Null List	only swap files or run-time systems are open on the disk.
<b>jobs accessing open files</b>	
Rd	user has read access to the file.
Wr	user has write access to the file.
Ca	file is open for user data caching.
Sq	file is open for sequential user data caching.
RR	file is open read regardless.
Tent	tentative file.
Up	file is open for UPDATE.
SpUp	file is open in special UPDATE mode.

4. If /C is specified, SYSTAT prints a table showing the allocation of memory on the current system. Because the table shows starting and ending addresses for currently installed run-time systems and resident libraries, the table is useful to privileged users who need to determine available addresses for the installation of new run-time systems or resident libraries.

The table lists the following items:

- a. The system memory allocation as set by INIT.SYS (see the *RSTS/E System Generation Manual*). These entries include:

Monitor (the resident portion)  
XBUF (extended buffer pool, if present)  
Locked out memory (if any)  
Non-existent memory (if any exists before the end of physical memory)

- b. Resident libraries.
- c. Run-Time Systems that were added at specific addresses or were loaded with the STAY attribute.

The table has the following form:

```
systat/c

Memory allocation table:
  Start   End   Length Permanent    TEMPORARY
    0K -   46K ( 47K) MONITOR
    47K -   62K ( 16K) BAS2F RTS
    63K -   68K (   6K) ** XBUF **
    69K -  100K (  32K) (User)
   101K -  123K (  23K) (User)      RMSRES LIB
  124K - *** END ***
```

where:

Start and End are the starting and ending addresses of that portion of memory.

Length the length of that memory segment.

Permanent those items allocated by INIT.SYS, and run-time systems and resident libraries that are permanently loaded (including the system default run-time system). Memory that is not permanently allocated in one of these ways is labeled USER.

Temporary run-time systems and resident libraries that were added at specific addresses but not permanently loaded.

Note that the table lists only those items assigned to specific addresses; it does not reflect the dynamic state of memory.

#### 14.1.2 SYSTAT as a CCL Command

SYSTAT as a CCL command works similarly to SYSTAT typed at a logged out terminal. The CCL command, however, can contain a file specification. The following commands show the proper procedure.

```
SYS B
READY
```

SYSTAT creates file B and writes to it a complete system status report. The file resides under your current account in the public structure.

```
SYS /B
```

This page intentionally left blank.

SYSTAT prints a busy device status report at the terminal. To create a status report in a file, type the file specification with an option as follows.

```
SYS B/B  
READY
```

SYSTAT creates a busy device status report in file B.

## 14.2 Obtaining a Disk Quota Report: The QUOLST Program

The QUOLST system program allows you to determine what portion of your disk quota is currently occupied and the number of free blocks remaining on the system disk. QUOLST is called as follows:

```
RUN $QUOLST
```

Output from QUOLST includes your account number and information printed under the following headings:

**Table 14-3: QUOLST Column Headings**

Column Heading	Meaning
STR	STRUCTure, device being reported.
USED	number of used 256-word blocks under your account. If the system supports large files, the number of blocks can appear as $>= 65535$ .
FREE	number of free blocks remaining in your account disk quota.
SYSTEM	number of free blocks remaining to the system on the structure indicated.

Output from QUOLST appears as follows:

```
RUN $QUOLST  
QUOLST V7.0 RSTS V7.0 Timesharing  
User: [200,57]  
Str Used Free System  
SY: 60 4940 5452  
Ready
```

In this example, you are logged into the system under account [200,57] and have used 60 blocks on the public disk structure with a quota of 5000 blocks ( $5000 - 60 = 4940$  free blocks). There are 5452 free blocks on the public disk structure.

## 14.3 Obtaining Account Data: The MONEY Program

MONEY is the RSTS/E system accounting program which allows you to obtain printed data on your account status. The program is called as follows (only when you are logged into the system):

```
RUN $MONEY
```

In the following example, you are logged into the system under account [100,100], and run the MONEY program:

```
RUN $MONEY
MONEY    V7.0 RSTS V7.0 Timesharing
System Accounting Program

Acct      Password  CPU-Time KCT'S Connect Device Disk   Quota   UFD
100,100          26.8     2436    23       2      31G    5000    16

Ready
```

The headings and information contained in the MONEY report are described in Table 14-4.

**Table 14-4: The MONEY Report**

Heading	Information
ACCT	Your account number.
PASSWORD	Not printed for non-privileged users.
CPU-TIME	The total number of seconds of central processor time used by the account.
KCT's	The total number of "kilo-core ticks" used by the account. This is a measure of total central processor usage, along with central processor time and memory usage. Whenever a program uses one-tenth of a second of CPU time, this value is incremented by the size of the program in K words.
CONNECT	The total number of minutes the account is logged into the system.
DEVICE	The total number of minutes devices are assigned by this account.
DISK	The number of blocks used on the public structure. If the account is using more than 65535 blocks, the entry is shown as > = 65535.
QUOTA	The disk quota in blocks.
UFD	The cluster size of your file directory.

These values do not reflect the current time-sharing session, because the accounting information is updated when the job is logged off.

## 14.4 Sending a Message to the System Manager: The GRIPE Program

The GRIPE system program allows you to communicate comments to the system manager. Comments, which you type while running GRIPE, are written to a common file where they are retained for inspection by the system manager.

Run GRIPE by typing the following command:

```
RUN $GRIPE
```

GRIPE prints an identification header followed by a query line to indicate its readiness to accept comments:

```
GRIPE V7.0 RSTS/E V7.0 TIMESHARING  
YES? (Press the ESCAPE key to end)
```

Following the prompt, type the text of your comment which is entered into the common file. Terminate the text of your comment by typing CTRL/Z or the ESCAPE or ALTMODE key. (Typing the ESCAPE or ALTMODE key is echoed at the terminal by a dollar sign (\$) being printed.) No carriage return-line feed operation is performed. The program indicates its acceptance of the text and its termination by printing the following lines:

```
THANK YOU
```

```
READY
```

## 14.5 Declaring a Terminal In Use: The INUSE Program

The INUSE system program prints or displays the words IN USE in block letters on the terminal to warn others not to use it. This message is followed by your job number and account number.

INUSE is called as follows:

```
RUN $INUSE
```

The printout from this program is shown below.

```
RUN $INUSE
```

```
IIIIII NN NN UU UU SSSSSS EEEEEEEEEE  
IIIIII NN NN UU UU SSSSSS EEEEEEEEEE  
II NNNN NN UU UU SS SS EE  
II NNNN NN UU UU SS SS EE  
II NNNN NN UU UU SS EE  
II NNNN NN UU UU SS EEEE  
II NN NN NN UU UU SSSSSS EEEEEEE  
II NN NN NN UU UU SSSSSS EEEEEEE  
II NN NNNN UU UU SS EE  
II NN NNNN UU UU SS EE  
II NN NNNN UU UU SS SS EE  
II NN NNNN UU UU SS SS EE  
IIIIII NN NN UUUUUU SSSSSS EEEEEEEEEE  
IIIIII NN NN UUUUUU SSSSSS EEEEEEEEEE
```

```
BY JOB 24 USER [2,201]
```

To regain control of the terminal, type the CTRL/Z or CTRL/C combination, or any valid command.

# **Chapter 15**

## **File Utility Programs: Listing, Editing, and Reading Files**

### **15.1 Listing Directory of Files: The DIRECT Program**

The DIRECT program lists file-related information from a disk directory. The benefits of DIRECT are increased speed and more options compared with other methods of listing directories. DIRECT opens your directory as a file and reads information by immediately accessing the blocks in the directory. This action is usually faster than the conventional method of passing the request for such information to internal system functions. Since the program follows pointers through a disk directory, incorrect information may be printed if the pointers are changed during program execution. For example, if a file is opened on the current account by another user after the directory operation is begun, then the information printed may be incorrect.

You run DIRECT by typing the RUN \$DIRECT command or by using the CCL command (explained later in this section). When DIRECT runs as a result of the RUN command, it prints a header line and the # character, which acts as a prompt. You can then type a command to DIRECT. If you run DIRECT by the CCL command, include the command to DIRECT in the CCL command.

The general format of the command is as follows:

```
output=input?option(s), input?option(s),...
```

Output is optional and can be a device specification or a disk file specification. If output is not given, the = character is optional and DIRECT prints output at your terminal. If an extension does not appear with an output filename, DIRECT appends .DIR unless you force a null extension by specifying the . character with the filename. Input can be any number of full disk file specifications. The full disk file specification on input can include a device, filename, extension and project-programmer number. If a device is not given, DIRECT uses the public structure and denotes it by the SY: specification.

The filename, extension, and project-programmer fields can contain \* and ? characters to denote wildcard specifications. If no file specification is given or if an \* character is given as the file specification, DIRECT processes all files in the directory. DIRECT applies the default interpretations shown for the following specifications for a given directory.

**User types:**      **Program interprets as:**      **Meaning:**

null	*.*	All files
*	*.*	All files
.*	.*	All files with null extensions
.	*	All files with null extensions
.EXT	*.EXT	All files with extension EXT
FILE	FILE.*	All files with filename FILE

With a file specification, you can specify one or more options. If no options appear, DIRECT proceeds as if you had specified /DI. Table 15-1 lists and describes the options.

**Table 15-1: DIRECT Options**

Type	Format	Meaning
Individual	/NA /EX /SZ /SI /AL /OP  /RT /PR /LA /DA /TI /CL	List filenames only. List filenames and extensions of each file. List filename, extension, size (in blocks) of each file; prints C if contiguous, P if file is protected (non-deletable), and L if file is placed. The /SI option cannot immediately follow the DIRECT CCL command (DIR/SI); /SZ is allowed. List filename, extension, and number of blocks allocated to file. List filename, extension, open status, and access count. Note that the designation U is appended to the access count if the file is opened in UPDATE mode; W is appended if the file allows write access.  If the system supports large files, the access count is printed as two numbers separated by a slash. The first number indicates the number of times the file is currently open in any mode but read regardless; the second number indicates the number of times the file is currently open in read regardless mode. Print the name of the run-time system that created the file. List filename, extension and file protection code. List filename, extension and date of last access for each file. List filename, extension and date of creation for the file. List filename, extension, date and time of day when file was created. List filename, extension, and file cluster size.

(continued on next page)

**Table 15-1: DIRECT Options (Cont.)**

Type	Format	Meaning
Aggregate	/SU	List only summary data to include number of designated files and total number of blocks occupied by designated files.
	/PO	List filename, extension, and the position of the file on disk (i.e., the device cluster number of the file's first block). Refer to Section 11.5.3.
	/SA /AT	List filename, extension, symbolic file attributes (see Section 11.6), and caching status. Caching is indicated by one of the following entries: CACHE:ON:RAN; file will be automatically cached randomly when open. CACHE:ON:SEQ; file will be automatically cached sequentially when open. CACHE:OFF:SEQ; file will not be automatically cached, but if cached, it will be cached sequentially.
	/OA	List filename, extension, and a series of octal numbers that represent file attributes. See Section 11.6 for an explanation of file attributes.
	/BR /F	List filenames and extensions with a brief summary message.
	/FU	List headings, filenames, extensions, size, protection code, date of last access, date of creation, time of creation, cluster size, associated run-time system, file position, open status, and summary.
	/DI:S /LI:S /S	List all relevant data to include headings, filenames, extensions, size, protection code, date of last access, date of creation, time of creation, cluster size, associated run-time system, file position, symbolic attributes, open status (if possible to print on the same line), files marked for deletion (see /MD), and summary. (Called slow directory.)
	/DI /LI null	List most important data to include heading, filename, extension, size, protection code, date of last access and summary.
	/MD	List files marked for deletion and flag them by printing an * after the file name.
	/N	Print only the entries which do not match the file specification given.
	/HD	Print heading at top of columns on the listing.
General	/W	List data across the width of a line rather than one item per line. Useful with large directory listings and individual options. When used alone, /W prints filenames and extensions across the width of a line with a summary message .
	/HE	Print the file DIRECT.HLP which describes the DIRECT program.
	/BK	List the directory for the specified device in reverse order. As a result, the files at the end of the directory appear at the beginning of the listing. If /BK is used to list files in the public structure and multiple disks are in the public structure, the listing reflects reverse order for each disk.

Consider the following example:

```
RUN $DIRECT
DIRECT V7.0 RSTS V7.0 Timesharing
*/DI:S
Name .Ext Size Prot Access Date Time Clu RTS Pos Op/rr SY:[2,211]
RMSTST.B2S 2 <60> 16-May-78 11:58 AM 8 BF2COM 22642 0/0
RMSTST.OBJ 7 <60> 16-May-78 11:58 AM 8 BF2COM 22643 0/0
RF:VAR=128 FU:SEQ USED:7:22 RECSI:128 CC:IMP
PROOF.BAS 2 <60> 13-Jun-78 03:00 FM 8 BAS4F 22644 0/0
PFILE.DAT 7 <60> 15-Jun-78 02:10 FM 8 BASIC2 22645 0/0
RF:FIX=56 FO:IDX USED:8:0 RECSI:56 CC:IMP
RFLN1.DOC 32 <60> 20-Jun-78 12:19 FM 8 BAS4F 22646 0/0
CHST1.BAS 1 <60> 24-Aug-78 02:04 FM 8 BAS4F 22650 0/0
CHST1.BAS 1 <60> 24-Aug-78 02:11 FM 8 BAS4F 22651 0/0
CHST1.B2S 1 <60> 24-Aug-78 02:14 FM 8 BFYCOM 22652 0/0
CHST1.OBJ 2 <60> 24-Aug-78 02:14 FM 8 BFYCOM 22653 0/0
RF:VAR=128 FO:SEQ USED:2:396 RECSI:90 CC:IMP
CHST1.CMD 1 <60> 24-Aug-78 02:16 FM 8 BFYCOM 22654 0/0
CHST1.ONL 1 <60> 24-Aug-78 02:16 FM 8 BFYCOM 22655 0/0
CHST1.TSK 7C <124> 24-Aug-78 02:17 FM 8 BASICY 22641 0/0
RF:FIX FO:SEQ USED:8:0 RECSI:51:2
FILE2.BAS 1 <60> 01-Sep-78 11:41 AM 8 BAS4F 22656 0/0
FILE.DIF 1 <60> 01-Sep-78 11:46 AM 8 BAS4F 22657 0/0
FILE1.BAS 1 <60> 01-Sep-78 11:53 AM 8 BAS4F 22658 0/0
FILE.DIF 1 <60> 01-Sep-78 11:55 AM 8 BAS4F 22659 0/0
INSLT.MAC 3 <60> 08-Sep-78 03:20 FM 8 RSX 22660 0/0
RF:STM FO:SEQ USED:3:234 RECSI:63 CC:IMP
INSLT.OBJ 1 <60> 13-Sep-78 02:21 FM 8 RSX 22661 0/0
RF:VAR FO:SEQ USED:1:144 RECSI:42 CC:IMP
INSLT.LST 5 <60> 13-Sep-78 02:21 FM 8 RSX 22662 0/0
RF:VAR FO:SEQ USED:5:456 RECSI:94 CC:IMP
```

```
FTNOTE.DOC 38 <60> 19-Jan-79 19-Jan-79 11:56 AM 8 ...RSX 1160 0/0
RF:STM FO:SEQ USED:38:487 RECSI:80
SORT1.BAS 1 <60> 22-Jan-79 22-Jan-79 03:30 PM 8 BASIC 1986 0/0
```

Total of 937 blocks in 97 files in SY:[2,211]

\*~Z

Ready

In this example, you run the DIRECT program and select the /DI:S option. This option causes the program to list all of the relevant information on the files in your account. This information is printed under the following headings:

Name.Ext	the filename and extension.
Size	the size (in blocks) of each file.
Prot	the protection code.
Access	the date of last access.
Date	the date of creation.
Time	the time of creation.
Clu	the file cluster size.
RTS	the file's run-time system.
Pos	the position of the file on disk (the device cluster number of the file's first block).
Op/rr	the number of times the file is currently open (Op) in any mode except read regardless; and the number of times the file is currently open in read regardless mode (rr).

Note that those files which have attributes contain a description of the attributes on the line following the file. Refer to Section 11.6 for the meaning of the listed attributes.

To list a directory at the line printer, specify the device designator and options in the command. For example:

```
#LPO:=*,BA?/F/W  
#
```

The command lists filenames and extensions of all files with BA as the first 2 characters of the extension. DIRECT formats the data across the width of the line printer paper.

To list a directory of a specific file, type the filename and extension (with options) in the command.

To list directories of several accounts and place them in a disk file, specify the project-programmer field and the disk file specification in the command. For example:

```
#PROJ=[120,*]/DI  
#
```

DIRECT creates the file PROJ.DIR in the current account and writes, to the file, directory listings of all accounts with project number 120.

An error encountered in a command causes DIRECT to print a message followed by the # character. You must retype the entire command correctly. The messages are described in Table 15-2.

**Table 15-2: DIRECT Program Error Messages**

Message and Meaning
<b>?DEVICE NOT DIRECTORY STRUCTURED</b> Device specified does not use a directory for file access.
<b>?DIRECTORY OF dev:[n,m] IS EMPTY</b> DIRECT finds the account [n,m] on device dev: contains no entries.
<b>?DISK PACK IS NOT ON-LINE</b> The pack or cartridge referred to is either not mounted or is off-line.
<b>?ILLEGAL FILE NAME &lt;filename&gt;</b> The file specified by <filename> contains a logical device name which you have not reserved by the ASSIGN command.
<b>?ILLEGAL INPUT FILE SPEC &lt;file spec&gt;</b> The file specification indicated by <file spec> generates the error ?ILLEGAL FILE NAME.
<b>?ILLEGAL SWITCH text</b> File specification contains an undefined option indicated by text.
<b>?INVALID DEVICE SPECIFICATION</b> The device specification is invalid or the device referred to does not exist on the system.
<b>?NO DIRECTORY FOR [n,m] ON dev:</b> DIRECT cannot find an account for user account [n,m] on the device dev: or else DIRECT encounters a protection violation.
<b>?NO HELP AVAILABLE</b> The file DIRECT.HLP is not on the system library account.
<b>?NO SUCH FILE AS &lt;file spec&gt; ON [n,m]</b> DIRECT cannot find the requested file indicated by <file spec> in the account [n,m].
<b>?OUTPUT FILE MUST BE IN THE USER'S AREA</b> DIRECT does not create an output disk file in another account if you are not privileged.
<b>?TOO MANY FILES FOR INVERTED DIRECTORY LISTING</b> DIRECT limits use of the /BK option to accounts with less than 200 files.

### **15.1.1 DIRECTORY as a CCL Command**

The following commands show some useful methods of requesting listings with the standard CCL command DIRECTORY or its abbreviation DIR. To list the filenames and extensions of all files in the current account, type the following command:

**DIR /W**

DIRECT prints the listing at your terminal and lists the information across the width of the page. To list at another device the filenames, extensions, sizes, protection codes and creation dates for certain files in the current account, type the following command:

```
DIR LP1:=*,BAS  
READY
```

DIRECT prints the listing of all BASIC-PLUS source files on line printer unit 1. The READY message indicates DIRECT has completed printing. To obtain a reverse listing, type the following command:

```
DIR /DI/BK
```

DIRECT prints, at the current terminal, directory and summary information in reverse chronological order. To print the file DIRECT.HLP on the line printer, type the following command:

```
DIR LP1:=/HE  
READY
```

## 15.2 Comparing Files: The FILCOM Program

The FILCOM (File Compare) system program allows you to compare two ASCII files, line by line, and identify differences. When the program runs, you specify the files you wish to compare and, optionally, one or more switches that direct that comparison.

To invoke the FILCOM program, type:

```
RUN $FILCOM
```

If the invocation is successful, FILCOM prints a two line identification header as follows:

```
FILCOM V7.0 RSTS V7.0 TIMESHARING  
FILE COMPARISON PROGRAM 30-Aug-78 11:59PM
```

Following the identification header, FILCOM prints a prompt:

```
OUTPUT TO <KB:> ?
```

In answer to the prompt, you can type one of three responses.

1. the device designator of the peripheral device to which FILCOM writes the comparison data.

If you only specify a device or type a RETURN key to accept the user terminal as a default, FILCOM prints an additional series of prompts. These prompts request the files you wish compared and allow you to set

the parameters of the comparison. These prompts are described in Section 15.2.1. Note that you can also specify a switch, or switches, as described in Table 15-3 in answer to this prompt. These switches provide an alternative method for setting comparison parameters.

2. a single command line which contains all of the information that FILCOM needs to make the file comparison.

The command line takes the form:

```
output=input1,input2/sw
```

If you specify the full command line, FILCOM does not issue any other prompt and begins the file comparison. Use of the command line allows you to specify wildcards in the file specifications and to specify one or more optional switches. Section 15.2.2 describes the command line specification and Table 15-3 describes the switches.

3. a command line that contains an indirect command filename. The indirect command file is an ASCII text file that you create with PIP or an editor. The file contains all of the information that FILCOM needs to make the file comparison.

The command line takes the form:

```
OUTPUT TO <KB:> ? @filename
```

where filename represents the indirect command file. The indirect command file can contain a single line command as described in the previous response 2, or it can contain multiple line answers to the FILCOM prompts as described in response 1. If the command file contains answers to the FILCOM prompts, each prompt must be answered in sequence as described in Section 15.2.1. Note that you cannot nest another indirect command file within the original response. If FILCOM encounters an at sign (@) in the command file, it processes it as an assignable account specification.

Note that after a file comparison is complete, FILCOM returns to the initial prompt for additional input. To exit from FILCOM, type CTRL/Z in response to the OUTPUT TO prompt.

### 15.2.1 FILCOM Prompts

If you answer the initial FILCOM prompt with a device designator or a RETURN key, the FILCOM program prints a series of additional prompts. These prompts request the following information:

1. the files you wish to compare.
2. the number of successive lines in those files that constitute a match.
3. whether FILCOM will consider BASIC-PLUS continuation lines as part of a numbered line.
4. whether FILCOM will compare blank lines.

Note that you can specify one or more of the switches described in Table 15-3 in response to FILCOM's initial prompt. You can use a switch specification to override the appearance of prompts and to provide additional instructions to the FILCOM program.

Following the initial prompt, FILCOM prints requests for the filenames of the ASCII files to be compared. For example:

```
OUTPUT TO <KB:> ? RET
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
```

In this command series, you specify that the file SORT1.BAS is to be compared to SORT2.BAS and the result of that comparison is to be printed at your terminal. When you specify the files to be compared, include the file extension. Note that you can use the wildcard specification \* and ? to designate the input files. The use of wildcards is described in Section 15.2.3.

The prompt which follows the input file requests asks you the number of lines in the file that will constitute a match. As FILCOM compares the lines in a file, it prints the differences until it comes to a series of lines that are identical. The number of lines that constitute a series is determined by your answer to this prompt. For example:

```
OUTPUT TO <KB:> ? SORT.DIF
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? RET
```

In this command series, you specify the creation of a file named SORT.DIF which contains the result of a comparison between the files SORT1.BAS and SORT2.BAS. FILCOM examines the files and records the differences until it finds three lines to match. That is, a RETURN key response accepts the default of three lines.

Following the HOW MANY TO MATCH prompt, FILCOM asks if you wish it to consider BASIC-PLUS continuation lines as part of a numbered program line. For example:

```
OUTPUT TO <KB:> ? RET
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? RET
BASIC+ LINES <NO> ? Y
```

In this command series, you specify that FILCOM is to consider continuation lines as part of a numbered program line when it compares SORT1.BAS and SORT2.BAS. Note that the default for this prompt, <NO>, causes FILCOM to compare only single lines. Consider the following program line:

```
100 FOR J=10, TO 15,    &
\   PRINT 3.14*j/2,      &
\   NEXT J
```

If you answer the BASIC+ LINES prompt with Y (for YES), FILCOM compares all three multi-line statements (FOR, PRINT, and NEXT) as a single line. If you accept the NO default, FILCOM only compares the first line (FOR J=10. TO 15. &) in SORT1 to a single line in SORT2.

The last prompt in the series asks if FILCOM is to compare blank lines. For example:

```
OUTPUT TO <KB:> ? SORT.DIF
INPUT FILE #1? SORT1.BAS
INPUT FILE #2? SORT2.BAS
HOW MANY TO MATCH <3> ? 5
BASIC+ LINES <NO> ? Y
BLANK LINES <NO> ? RET
```

In this command series, you specify that FILCOM is to compare SORT1.BAS and SORT2.BAS and print the differences in SORT.DIF. You instruct FILCOM to match five successive lines, consider BASIC-PLUS continuation lines, and to ignore the presence of blank lines. That is, if FILCOM encounters two blank lines in SORT1 and five blank lines in SORT2, it is not considered a difference.

Section 15.2.4 contains examples of FILCOM comparisons.

### 15.2.2 FILCOM Single Command Line

In response to its initial prompt, FILCOM accepts a single command line that specifies all files and comparison parameters.

This command line has the following format:

```
OUTPUT TO <KB:> ? output=input1,input2/sw
```

where:

output is one of the following;

1. a complete RSTS/E file specification containing device, project-programmer number, filename, and extension designators.
2. a RETURN key which specifies the default user terminal. If you specify a RETURN key, FILCOM prints the prompts as described in Section 15.2.1.
3. a wildcard specification as described in Section 15.2.3.
4. no specification. If you do not specify a file or equal sign, but rather an input specification only, output is printed on the user terminal.

= required only if a file specification is given for output. Note that an equal sign with no output specification causes FILCOM to output the result of its file comparison to a file on the public structure. FILCOM creates this file in the current account with a filename taken from the first input file and a .DIF extension.

- input1 the two files that FILCOM is to compare separated by commas. You  
 input2 can describe these files with full file specifications, filenames and  
 extensions, or wildcards as described in Section 15.2.3.  
 /sw one or more optional switches as described in Table 15-3.

**Table 15-3: FILCOM Switches**

Switch	Meaning
/MA[TCH]:[:n]	specifies the number of lines in the file that will constitute a match (see Section 15.2.1). If you do not specify the number of lines (n), the default is three.
/BL[ANK] /NOBL[ANK]	specifies whether FILCOM will consider blank lines in its comparison (see Section 15.2.1). The default is not to consider blank lines (/NOBL).
/BA[SIC] /NOBA[SIC]	specifies whether FILCOM will consider BASIC-PLUS continuation lines as part of a numbered program line (see Section 15.2.1). The default is not to consider continuation lines (/NOBA).
/PA[TCH]	specifies that FILCOM is to create the output file as a patch file. You can use an APPEND command (see Section 8.1.5) to include the patch file in a succeeding FILCOM operation to cause the resulting output to be equivalent to the second input file.
/AP[PEND]	causes FILCOM to open the output file for APPEND. If this switch is not specified, FILCOM deletes any existing output file of the same name prior to writing differences. With the /AP switch, you can cause FILCOM to write differences into an existing output file.
/SU[MMARY]	causes FILCOM to list the total number of differences found in the files without listing individual lines.
/LI[LIMIT]:[:n]	specifies the maximum number of lines that are allowed to differ before FILCOM stops examining the file. If you include the /LIMIT switch but do not specify a limit in n, the default is 60 differing lines.
/CO[MPARE]:[:m:n]	converts tabs to the appropriate number of spaces and causes FILCOM to compare characters starting in column position m and continuing for n characters. If you include the /COMPARE switch but do not specify m or n, FILCOM defaults m to column position 1 and n to 72 characters.

The use of these switches is not restricted to single line commands. That is, you can specify any of these switches in answer to the OUTPUT TO prompt, regardless of whether you specify the full FILCOM command line in that prompt.

If you do not specify any switches, FILCOM assumes that three lines constitute a match, blank lines will not be compared, and BASIC-PLUS continuation lines will not be considered part of a numbered program line (i.e., /MA:3/NOBL/NOBA). However, if you specify the /PATCH switch, FILCOM does compare blank lines and does consider BASIC-PLUS continuation lines. That is, a /PA specification overrides the normal defaults and sets /BA and /BL.

If you specify a switch in illegal format or a switch that is not one of the legal set, the following error message is printed:

```
ILLEGAL COMMAND LINE /x
```

where /x is the illegal switch.

If you specify the /LIMIT switch, FILCOM ceases to compare the files when a set number of lines are found to be different during a single compare. You can specify the number in the switch or accept the default of 60 lines. Once the limit is exceeded, FILCOM stops comparing the files and prints the following message:

```
LIMIT x REACHED ON LAST COMPARE, SKIPPING REST OF FILE
```

where x is the /LI specification or the default 60. Note that no limit is set unless you specify the /LI switch.

The /COMPARE switch is useful for comparisons of lines of text where the characters beyond a certain column need not be considered. For example, a program that contains comments beginning in column position 80 can be compared and the /CO switch used to ignore the comments. That is, you specify /CO:1:79 to cause FILCOM to convert all tabs in the lines to the appropriate number of spaces (in order to properly count character positions) and compare each line from column 1 to column 79. FILCOM ignores the content of the lines from column 80 to the end of the line.

Consider the following examples.

```
OUTPUT TO <KB:> ? SORT.DIF=SORT1.BAS,SORT2.BAS
```

This command series causes FILCOM to compare SORT1 and SORT2 and report the differences in the file SORT.DIF. FILCOM sets the /MA:3, /NOBA, and /NOBL switches by default.

```
OUTPUT TO <KB:> ? SORT1.BAS,SORT2.BAS/LI:10
```

This command series causes FILCOM to compare SORT1 and SORT2 until the number of differing lines in a single compare exceed 10 or the comparison is complete. FILCOM creates a file named SORT1.DIF in the current account to store the result of the comparison. Note that the /MA:3, /NOBA, and /NOBL switches are set by default.

```
OUTPUT TO <KB:> ? SORT1.BAS,SORT2.BAS/BA/BL/SU
```

This command series causes FILCOM to compare SORT1 and SORT2 and print a summary of the differences on the user terminal. FILCOM is instructed to compare blank lines (/BL) and to consider BASIC-PLUS continuation lines (/BA).

### 15.2.3 Wildcards

You can specify FILCOM input files as full RSTS/E file specifications or as wildcard characters; FILCOM output files can contain a wildcard as the filename or extension but not both. For output, you must specify a filename or extension or default both fields as described in Section 15.2.2. The use of wildcard characters \* and ? are documented in Section 11.3.1. The asterisk (\*) wildcard replaces an entire specification field (i.e., filename, extension, etc.). The question mark (?) wildcard replaces a character within a field. When wildcards are specified in a FILCOM command line, FILCOM bases its search for files on the first input file specification.

FILCOM uses wildcard specifications in the following manner:

If the specification is:	FILCOM
*.EXT,*.XYZ	compares all files with the extension .EXT and all files with the same filenames and extension .XYZ.
ABC.*,DEF.*	compares all files named ABC with any extension and all files named DEF and the same extensions.
[5,30]*.*,[5,31]*.*	compares all files in account [5,30] and all files with the same filenames and extensions in account [5,31].
[5,30]*.*,[5,31]*.EXT	compares all files in account [5,30] and all files with the same filenames and an .EXT extension in account [5,31].
AFILE?.*,BFILE?.*	compares all files whose filename consists of AFILE and any character with any extension and all files whose filename consists of BFILE and the same character and the same extension.
*.DIF=*.ABC,*.DEF	compares all files with the extension .ABC and all files of the same filename with the extension .DEF. As FILCOM compares each set of files, it creates a series of output files with the extension .DIF. The output filenames duplicate the filenames of each .ABC file compared.

With wildcards, you can cause FILCOM to make multiple file compares based on a single command line specification. When such a method is used, FILCOM prints the differences found in each set of files in the standard manner (see Section 15.2.4). In addition, FILCOM prints a summary at the completion of the comparisons. The summary has the form:

```
x DIFFERENCES FOUND IN y FILES OF z TOTAL FILES COMPARED
```

where x is the total number of differences in all files, y is the number of files that contain differences, and z is the total number of files compared based on the wildcard specification.

Wildcard filenames for the first input file can only be specified if the file resides on disk. If the first input file is not a disk file, FILCOM prints the message:

```
WILDCARD FILENAME ILLEGAL ON DEVICE x
```

(where x is the specified input device) and returns to the OUTPUT TO prompt. Note that the second input file does not have to reside on disk.

If a wildcard specification would cause FILCOM to create a filename for output that is syntactically incorrect, the program prints the following message:

```
ILLEGAL FILENAME x CREATED FROM y BY z
```

where x is the illegal filename that would have been created, y is the wildcard specification you made for the first input file, and z is the file specification for the second input file. For example:

```
OUTPUT TO <KB:> ? *,DIF=*,EXT,???OLD,EXT
```

would result in an output filename of A OLD.DIF if A.EXT existed as the first input file. Because spaces are not allowed in a filename, the error message is returned.

#### 15.2.4 FILCOM Examples

Assume that the following two files are in your account on the system disk.

TEST1	TEST2
10 REM A	10 REM A
20 REM B	20 REM B
30 REM C	30 REM C
40 REM D	70 REM G
50 REM E	80 REM H
60 REM F	90 REM I
70 REM G	100 REM J
80 REM H	110 REM L
90 REM I	120 REM 2
100 REM J	130 REM 3
110 REM K	140 REM N
120 REM L	150 REM O
130 REM M	160 REM P
140 REM N	170 REM Q
150 REM O	180 REM R
160 REM P	190 REM S
170 REM Q	200 REM T
180 REM R	210 REM U
190 REM S	220 REM V
200 REM T	222 REM 4
210 REM U	224 REM 5
220 REM V	230 REM W
230 REM W	240 REM X
240 REM X	250 REM Y
250 REM Y	260 REM Z
260 REM Z	

To compare these files, you can run FILCOM as follows:

```
RUN $FILCOM
FILCOM V7.0  RSTS V7.0  TIMESHARING
FILE COMPARISON PROGRAM  01-Sep-78 11:14 AM
OUTPUT TO <KB:> ? TEST1.BAS,TEST2.BAS
```

This command line causes output to your terminal and defaults the switches /MA:3, /NOBL, and /NOBA. The output appears as follows:

```
COMPARING: 1) [2,211]TEST1.BAS TO 2) [2,211]TEST2.BAS
*****
1) [2,211]TEST1.BAS
40 REM D
50 REM E
60 REM F
70 REM G
*****
2) [2,211]TEST2.BAS
70 REM G
*****
1) [2,211]TEST1.BAS
110 REM K
120 REM L
130 REM M
140 REM N
*****
2) [2,211]TEST2.BAS
110 REM I
120 REM Z
130 REM 3
140 REM N
*****
1) [2,211]TEST1.BAS
230 REM W
*****
2) [2,211]TEST2.BAS
222 REM 4
224 REM 5
230 REM W
```

?3 Differences Found

OUTPUT TO <KB:> ? ^Z

FILCOM prints the lines from both files that do not compare followed by the first line that does compare. The first line in each group is the first line that differs and the last line in each group is the first line that is the same. In this example, the first group consists of lines 40 through 70 in TEST1.BAS. FILCOM printed line 40 because it had no match in TEST2.BAS. FILCOM continued to print TEST1 until it found three matching lines (as directed by the /MA:3 default), the first line of which was line 70.

As another example, consider the following two programs.

5 EXTEND	5 EXTEND
10 REM A	10 REM A
20 REM B	20 REM B
30 REM C &	30 REM C
\ PRINT "LINE 30"	40 REM D
40 REM D	50 REM E &
50 REM E &	\ PRINT "LINE 50"
\ PRINT "LINE 50"	70 REM F
60 REM F	80 REM G &
70 REM G	\ PRINT "LINE 80"
90 END	90 END

The following series of commands cause FILCOM to create a file named FIL.DIF.

```
RUN $FILCOM
FILCOM V7.0 RSTS V7.0 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-78 11:54 AM
OUTPUT TO <KB:>? FIL.DIF=FILE1.BAS,FILE2.BAS/BA/MA:2/SU
OUTPUT TO <KB:>? ^Z

READY
```

The file FIL.DIF contains a comparison of FILE1.BAS and FILE2.BAS. The specified switches instruct FILCOM to consider BASIC-PLUS continuation lines as part of a numbered program line and to print differences when more than two lines fail to match. The /SU switch cause FILCOM to print a summary of the differences in FIL.DIF. After completion of the compare, FIL.DIF contains the following information:

```
FILCOM V7.0 RSTS V7.0 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-78 11:54 AM

COMPARING: 1) [2,211]FILE1.BAS TO 2) [2,211]FILE2.BAS

?2 DIFFERENCES FOUND.
```

The following series of commands cause FILCOM to output its comparison of FILE1.BAS and FILE2.BAS to the user terminal. However, the number of lines that is needed to match is four and a full description of the differences is requested.

```
RUN $FILCOM
FILCOM V7.0 RSTS V7.0 TIMESHARING
FILE COMPARISON PROGRAM 01-SEP-78 11:54 AM
OUTPUT TO <KB:>? FILE1.BAS,FILE2.BAS/BA/MA:4

COMPARING: 1) [2,211]FILE1.BAS TO 2) [2,211]FILE2.BAS

*****
1) [2,211]FILE1.BAS
30      REM C  &
\      PRINT "LINE 30"
40      REM D
50      REM E  &
\      PRINT "LINE 50"
60      REM F
70      REM G
80      END
*****
2) [2,211]FILE2.BAS
30      REM C
40      REM D
50      REM E  &
\      PRINT "LINE 50"
70      REM F
80      REM G  &
\      PRINT "LINE 80"
90      END

?1 DIFFERENCE FOUND.

OUTPUT TO <KB:>? ^Z
```

### **15.2.5 FILCOM Error Messages**

FILCOM can return the following errors and informational messages.

#### **Message and Meaning**

##### **?A NULL LENGTH FILE**

The file examined by FILCOM (the first or second input file) is empty.

##### **?LIMIT n REACHED ON LAST COMPARE, SKIPPING REST OF FILE**

The number of differing lines in a single compare (n) as specified in the /LIMIT switch is exceeded. FILCOM continues to the next set of files or prints the OUTPUT TO prompt.

##### **?LINE TOO LONG AT LINE n**

FILCOM attempted to input a line at program line number n that is greater than 255 characters. The program continues to compare the rest of the lines in the file.

##### **?WILDCARD FILENAME ILLEGAL ON DEVICE x**

You typed a wildcard specification for the first input file where that device was not a disk device.

##### **?WILDCARD '\*.\*' ILLEGAL FOR OUTPUT FILENAME**

FILCOM requires either a filename or extension in the output file specification. A wildcard can be used for one of these fields (but not both) or both fields can be defaulted.

##### **?OUTPUT FILENAME 'file.ext' WOULD OVERWRITE AN INPUT FILE**

###### **- SKIPPING COMPARE**

During a check of the input and output file specifications, FILCOM discovered an output specification that would cause the overwriting of an input file. The operation is aborted.

##### **?ILLEGAL COMMAND LINE x**

The FILCOM command line or file specification contains illegal syntax.

##### **?FILE HAS NON-ASCII STREAM ATTRIBUTE - CAN'T READ <filename>**

The FILCOM program can only compare ASCII file data. If the file <filename> contains file attribute data that is not ASCII stream, this error is returned. The program continues to the next set of files or prints the OUTPUT TO prompt.

##### **?INPUT FILE #n<filename> NOT FOUND**

FILCOM was unable to find the specified input file. The message indicates the first or second input file (#n) and the filename.

##### **?ILLEGAL FILENAME**

The filename specification (input or output) is syntactically incorrect.

##### **?ILLEGAL FILENAME x CREATED FROM y BY z**

Based on the wildcard input file specification (y and z), FILCOM would have created a filename (x) of illegal syntax. This error is printed and the program continues to the next set of files or prints the OUTPUT TO prompt.

##### **?MAXIMUM MEMORY EXCEEDED**

The files being compared are significantly different and FILCOM lacks the work space to compare the remainder of the file. If the /BASIC switch was specified, retry the compare with the /NOBASIC switch.

## **15.3 Generating a Cross-Reference Listing: The BPCREF Program**

The BPCREF (BASIC-PLUS cross reference) program examines a BASIC-PLUS compiled image and generates a listing of variables, line number references, and some statistical data. Optionally, the program can include a listing of the source file, local or global references, and can automatically queue the output to the line printer spooler. You can examine the BPCREF output to find errors and can use the listing to assist in updating or modifying your code.

The BASIC-PLUS cross reference program consists of two compiled modules - BPCREF and BPCR1. The system manager has the option of installing these modules in a non-standard library account (they must be in the same account); check with the system manager for the location of these modules. The first module is the one you run. It parses the command line and chains to the second module which produces the output. For simplicity, the documentation refers only to one program, BPCREF, which includes functionally the two modules.

BPCREF works only on BASIC-PLUS compiled images (that is, .BAC files). It does not work on BASIC-PLUS-2 object or task images or on source files. Note that the local and global reference switches (see Table 15-5) can be useful when converting the BASIC-PLUS program to BASIC-PLUS-2. BPCREF generates a fatal error when it tries to process a file in other than the BASIC-PLUS compiled image format.

### **15.3.1 Running and Terminating BPCREF**

If the program is installed in the system library account, BPCREF is run by the following command:

```
RUN $BPCREF
```

After printing a header line, the program displays a query:

```
BPCREF COMMAND?
```

Type a response in one of the formats shown in Table 15-4.

**Table 15-4: BPCREF Command Formats**

Format	Result
file1	Examines the file file1.BAC and generates an output file with the same name and the extension .CRF.
file1=file2	Examines the file file2.BAC and generates an output file named file1.CRF.
file1,file2	Examines file1.BAC and includes in the output file file1.CRF the source listing of file2.BAS.
file1=file2,file3	Examines file2.BAC and includes in the output file file1.CRF the source listing of file2.BAS. The source listing is in the file file3.BAS.

Switches can be included in the command to control the processing. The switches are listed in Table 15-5. You can abbreviate a switch to one or more characters. If an undefined switch appears in the command without the /Q switch, the program prints a warning message and ignores the switch. With the /Q switch in the command, BPCREF passes any undefined switches to the queueing program. A message is printed documenting this action.

Queuing can be done to any spooling unit. When you request queuing, BPCREF generates a request to queue the output file to the default line printer. By specifying the spooling unit, for example, LP1:=FILE1/Q, you can direct BPCREF output to a unit other than 0.

**Table 15-5: BPCREF Command Switches**

Switch Format	Format
/NOD[ELETE]	Tells the spooling program not to delete the output files after printing.
/NOH[EADER]	Tells BPCREF to omit header lines in the output file. The resulting .CREF file is more suitable as input to the FILCOM program.
/P[AGE]:nnn	Specifies nnn as the number of lines per page to print in the output listing. Sixty lines per page is the default.
/Q[UEUE]	Automatically queues the output file to the printer with deletion after printing unless the /NODELETE switch is specified.
/S[OURCE]	Includes the source file in the output file. The .BAS form of the compiled image must reside in the same account as the .BAC file. If this is not the case, use the format file1=file2, file3 to include the source file file3.BAS.
/W[IDTH]:nnn	Tells BPCREF to use nnn columns per line as the width of the variable cross reference table. The default is 132 columns. The minimum width is 72 columns.
/GL[OW]:nnn /GH[IGH]:nnn	Tells BPCREF to print the variables referenced both inside and outside program boundaries as represented by the line numbers nnn. BPCREF does not print variables that are only referenced inside or only referenced outside the boundaries.
/LL[OW]:nnn /LH[IGH]:nnn	Tells BPCREF to print the variables referenced within program boundaries as represented by the line numbers nnn. BPCREF does not print variables that are referenced outside the boundaries or are referenced both inside and outside the boundaries.
/NOC[REF]	Tells BPCREF to suppress a full cross reference listing where only a local or global listing is desired.

### 15.3.2 Output Listing Contents

BPCREF generates an output file containing the following information:

1. an ordered and cross-referenced list of line numbers and variables used in the compiled image and the line numbers on which they are referenced.
2. a list of line numbers and variables whose use is questionable.
3. a statistical summary.

Each page of the output file has a header line that contains the related program name, the current date and time of day, and a page number. (This header line can be suppressed with the /NOHEADER switch.)

The following codes (printed after line numbers) are used to show the kind of references to variables:

- @ Destructive reference. The reference to the variable on the line changed the value of the variable.
- # Definition. A DIM statement or function defines the variable.

The optional list of suspect line numbers and variables is introduced by the following line:

PLEASE CHECK THAT THE FOLLOWING VARIABLES HAVE BEEN REFERENCED PROPERLY:

BPCREF determines, by cross checking references to variables, whether a variable may be incorrectly referenced (never read, never written, never defined). This list points out possible typing errors in the source code. BPCREF, however, cannot logically analyze the program but simply tries to find simple typing and logic errors.

The last part of the output, the statistical data, includes the following information:

- the number of variables in the code.
- the number of references to line numbers and variables.
- the number of bytes needed to store variable definitions.
- the number of statements in the code.
- the number of internal code elements.
- the elapsed and processor time (in seconds) required to create the output.

BPCREF also prints the size (in K-words) of the program's data and code areas, and the amount of memory reserved, actually used, and free for program expansion.

If the local (/LL and /LH) or global (/GL and /GH) switches are present in the BPCREF command line, it causes the program to print a listing of local or global variable references. These switches allow you to partition a program into two regions and print a listing of variable references that indicate whether the reference is within one region (local) or common to both (global). Such a listing is useful if the program is to be segmented into chainable modules or is to be converted to BASIC-PLUS-2.

To request a listing of globally referenced variables, specify the /GL:nnn and /GH:nnn switches in the command line. These switches cause BPCREF to generate a listing of all variables that are referenced both inside and outside a specified program region. BPCREF omits from the listing any variable that is only referenced inside the region or only referenced outside the region. You define the program region with a lower boundary line number specification in

/GL:nnn and an upper boundary line number specification in /GH:nnn; where nnn is the line number. If no line number is specified, BPCREF assigns line number 0 to /GL and line number 32767 to /GH.

To request a listing of locally referenced variables, specify the /LL:nnn and /LH:nnn switches in the command line. These switches cause BPCREF to generate a listing of all variables that are referenced within a specified program region. BPCREF omits from the listing any variable that is referenced outside the region. You define the region with a lower boundary line number specification in /LL:nnn and an upper boundary line number specification in /LH:nnn; where nnn is the line number. If no line number is specified, BPCREF assigns line number 0 to /LL and line number 32767 to /LH.

You can combine the global and local switches to generate a listing of both types of references. Also, when more than one global or local switch of the same boundary type appear on the command line, BPCREF uses the last specification. For example:

```
BPCREF COMMAND? FILE.EXT/GL:30/GL:20
```

causes BPCREF to use line number 20 as the global reference lower boundary. Note that if only a local or global reference listing is desired, specify the /NOC switch to suppress the full reference listing.

### 15.3.3 Error Messages

BPCREF reports errors for incorrect responses to its query or for fatal or unexpected conditions discovered during processing. In most cases, BPCREF prints a message and returns to monitor command level. The messages are listed and described in Table 15-6.

Table 15-6: BPCREF Error Messages\*

Message and Meaning
?BPCREF Version FATAL ERROR mmm AT LINE nnn – text BPCREF or BPCRF1 encountered the condition described by RSTS/E error number mmm while executing an operation at line number nnn. BPCREF cannot continue. (Error numbers are listed in Appendix C.) This condition should be reported to the installation system manager. If the error cannot be traced to a hardware problem, the system manager should submit an SPR to DIGITAL.
?CAN'T OPEN filename – text BPCREF cannot access the related file because of the condition described by the RSTS/E error text.
?COMMAND ERROR – text While processing the response to the command query, BPCREF encountered the RSTS/E error described in the text. BPCREF prints a help message and reprints the query.

\*The first character of each message indicates the severity of the error. The severity standard in error messages is described in Appendix C.

(continued on next page)

**Table 15-6: BPCREF Error Messages (Cont.)**

Message and Meaning
<b>%CORRECT PRIME - nnn</b> The program has been modified to change the variable table size. To operate correctly, respecify the correct value given by nnn (which must be a prime number) and re-compile the program. This message should not ordinarily appear.
<b>?COULD NOT CHAIN TO filename - text</b> BPCREF must chain to the auxiliary module described by filename. The RSTS/E error is printed in text.
<b>%ILLEGAL /WIDTH, MINIMUM = 72</b> In response to the command query, you specified a value in the /WIDTH switch less than the allowed minimum. BPCREF sets the width to the minimum (72 columns) and continues processing.
<b>?filename IS NOT A COMPILED BASIC-PLUS PROGRAM</b> The file indicated by filename is not a valid BASIC-PLUS compiled image. BPCREF cannot process a source file, task image, or other file formats.
<b>?MAXIMUM MEMORY EXCEEDED</b> Because of the high number (or great length) of variable names, the BPCREF variable table has overflowed. Subdivide the program and process each module separately.
<b>?filename MUST BE ON DISK</b> The file indicated by filename is not on a disk device. BPCRF processes only disk files.
<b>?PLEASE RUN BPCREF</b> You attempted to run the BPCRF1 module. BPCRF1 prints this message and terminates.
<b>%REFERENCE TABLE FULL</b> An internal table is full. BPCREF continues processing, but loses some data. Have the system manager submit an SPR to DIGITAL with a machine-readable copy of the source form of the program that caused this error.
<b>%UNKNOWN COMPILED CODE nn AT LINE mm</b> BPCREF encountered a BASIC-PLUS operation code (nn) that it did not recognize. BPCREF continues processing, but it may lose some data or output incorrect data (for example, the program may miss a variable reference at this point).
<b>%VARIABLE TABLE FULL</b> BPCREF can process a maximum of 421 variables (plus line numbers used in GOTO and GOSUB statements)*. Decrease the number of variables in the program and try again.

\*The system manager can install a patch to increase the maximum number of variables BPCREF can process.

## **Chapter 16**

### **Device Utility Programs: PIP, COPY and FIT**

#### **16.1 Device Transfer: The PIP Program**

The PIP (Peripheral Interchange Program) system program is a file transfer and maintenance utility that can copy files from one RSTS/E device to another, concatenate, delete, and rename files, transfer protection codes and file attribute data, zero accounts, and list directories. PIP requires 12K words of memory and runs under the control of the RT11 Run-Time System.

To invoke PIP, you must be logged into the system and type the following command:

```
RUN $PIP  
*
```

In response, PIP prints an asterisk (\*) prompt to indicate its readiness to accept input.

If the current default run-time system is RT-11, you can alternatively type one of the following commands in response to the RT-11 prompt:

```
.R PIP
```

or

```
.RUN $PIP
```

In response to either of these commands, PIP prints its asterisk prompt.

In addition to the RUN commands, the RSTS/E CCL command PIP invokes the program. For example:

```
PIP  
*
```

Note that if you use the PIP CCL command, you can include PIP commands on the same line.

To return to the system level (or your current default run-time system), type CTRL/Z in response to the PIP prompt:

```
PIP  
*^Z  
READY
```

A CTRL/Z is equivalent to an end-of-file on your terminal and causes PIP to complete the current operation and exit in an orderly fashion. If you type CTRL/C, PIP exits but does not complete the current operation. If you type CTRL/C during a PIP operation, PIP halts the operation (some data may be lost) and reissues its prompt.

### 16.1.1 PIP Command Line Specifications

The PIP command line specifies the actions you wish performed and the files involved. PIP accepts a command line of up to 80 characters in the following general format:

```
*[output[/sw]=]input[/sw][,input[/sw],...]
```

where output and input are file specifications and /sw are one or more switches that specify the action to be taken on the files. You terminate a PIP command line with the RETURN key.

An output file specification has the form:

```
dev:[proj,prog]name.ext<prot>
```

where:

- |             |   |
|-------------|---|
| dev         | is a device specification. If none is specified, the default is DK:. Note that DK: defaults to the public disk structure (SY:) unless you have previously assigned DK: to another device.   |
| [proj,prog] | is a project-programmer number. If none is specified, the default is your current account.  |
| name        | is a filename specification. PIP allows you to specify only one output file specification. If none is specified, the default is an asterisk (*) wildcard character (see Section 16.1.2). Note that for output-only (non-file structured) devices such as KB:, LP:, and PP:, the system ignores the filename and extension specifications. |
| .ext        | is a file extension. If none is specified, the default is an asterisk (*) wildcard (see Section 16.1.2). If you specify only the dot, the extension is null.  |
| <prot>      | is a protection code. If none is specified, PIP assigns your current default protection code. Note that if the file extension indicates an executable file (.BAC, .TSK, .SAV), PIP adds the compiled (64) bit to the default protection code.   |

If you do not type any output specification, PIP assumes the following:

KB:\*,\*

An input file specification has the form:

dev:[proj,prog]name.ext<prot>

where:

- dev: is a device specification. If none is specified, the default is DK: as described for output. Note that if a preceding input file contains a device specification, that device becomes the default for the current command line until overridden by another specification.
- [proj,prog] is a project-programmer number as described for output.
- name is a filename specification. PIP allows you to specify up to 6 input file specifications separated by commas. If none is specified, the default is an asterisk (\*) wildcard (see Section 16.1.2). Note that an input file specification is required in a delete operation (see Section 16.2.6). For input-only (non-file structured) devices such as KB:, PR:, and CR:, the system ignores a filename and extension specification.
- .ext is a file extension as described for output.
- <prot> is a protection code. PIP ignores input file protection codes unless the file is the target of a rename operation (see Section 16.2.7).

If you do not type any input file specification, PIP returns the error:

?NO INPUT FILE

A PIP switch specification (see Section 16.2) always begins with a slash (/). When you specify more than one switch in the command line, you must separate them with slashes. When no switch is specified, PIP performs a block mode transfer as described in Section 16.2.2.4.

Note that both the input and output file specifications can contain the RSTS/E file switches; /FILESIZE, /CLUSTERSIZE, /MODE, /RONLY, and /POSITION. However, file switches must immediately follow the file specification; an error is returned if file and PIP switches are interspersed. Also, /CLUSTERSIZE is ignored on non-disk output file specifications. File specification switches and their format are described in Section 11.5.

### 16.1.2 Wildcard Specifications

PIP allows you to use the asterisk (\*) and question mark (?) wildcard characters in input and output file specifications. The use of wildcards can save you

typing time and permits you to designate multiple files in a single file specification and thereby circumvent the PIP restriction of six maximum input file specifications.

Wildcards can be specified in the following manner:

Specification	Result
FILE.*	All files with the name FILE and any extension.
*.EXT	All files with the extension .EXT.
*.*	All files.
FILE.EX?	All files with the name FILE and .EX as the first two characters of the extension. The third character is any character including blank.
FILE???.EXT	All files with 4- to 6-character names, the first four of which are FILE, and an EXT extension.
FILE???.E??	All files with 4- to 6-character names, the first four of which are FILE, and any extension whose first character is E.
AB?.EXT	All files with 2- to 3-character names, the first two of which are AB, and an .EXT extension.
FILE???.*	All files with 4- to 6-character names, the first four of which are FILE, and any extension.
*.EX?	All files of any name whose extension begins with .EX.

In the absence of an element or elements of the full input or output file specification (see Section 16.1.1), PIP applies the following defaults to the missing elements (note that delete operations do not permit default input specifications):

Specification	Default	Meaning
null	*.*	All files in the current account.
*	*.*	All files in the current account.
*.	*.	All files with null extensions in the current account.
.	*	All files with null extensions in the current account.
.EXT	*.EXT	All files with an .EXT extension in the current account.
FILE	FILE.*	All files named FILE with any extension in the current account.

Note that the use of a question mark wildcard character in the output specification can cause PIP to create a file whose filename contains embedded spaces. Such an occurrence causes PIP to print the error ?ILLEGAL FILENAME, ignore the file, and continue with the next specified file.

PIP allows you to specify an asterisk wildcard character in a project-programmer number specification. However, if wildcard accounts are specified for input files, the file must be resident on disk or magnetic tape.

### 16.1.3 Indirect Command Files in PIP

PIP allows you to specify an indirect command file in response to its prompt. An indirect command file is a user-created (with PIP or an editor) ASCII file that contains all of the information PIP needs to perform an operation. If a line of the file begins with a semicolon (;), PIP treats the line as a comment and skips to the next line in the file. If the file causes the execution of a /ZE or /IN switch operation (see Section 16.2), confirmation questions are printed at your terminal.

You specify an indirect command file to PIP in the following manner:

```
*@dev:[proj,proj]file.ext
```

where the at sign (@) is the first character following the prompt. The device defaults to the public structure if none is specified. The project-programmer number defaults to the current account. The filename must be specified. However, if the extension is not included, it defaults to .CMD.

An indirect command file, in addition to containing PIP commands and file specifications, can contain references to other indirect command files. This process is called nesting. The number of files you are allowed to nest is dependent on the amount of memory available to you. If memory is exceeded, PIP prints the error ?NO BUFFER SPACE FOR INDIRECT COMMAND FILE, halts the operation, and returns to the prompt level. Note that a file, which contains nested indirect commands, must be resident on a file structured disk.

Most errors during command file processing cause PIP to return to prompt level. On the following errors, however, PIP prints the error message and continues processing:

1. ?NO FILES MATCHING <spec>  
where <spec> is a file specification and the operation was /DE or /LI (see Section 16.2).
2. ?FILE OR ACCOUNT ALREADY EXISTS  
where the operation was /RE (see Section 16.2).

The following command shows the procedure you would use to execute an indirect command file:

```
*@ABC
```

PIP reads the file named ABC.CMD in the current account on the public structure. If you had specified KBn: following the at sign (@KB3:, for example), PIP would read the specified keyboard for commands. You terminate the keyboard command set with a CTRL/Z.

Because a logical account assignment (see Section 4.7) and the PIP indirect command are both designated with an at sign, the placement of the at sign in the PIP command line is important. When the command line contains an at

sign, PIP scans the line for validity as an indirect file command. If PIP encounters more than one file specification, wildcard file specifications, or a PIP switch (see Section 16.2), PIP assumes that the at sign is a logical account and processes the command line accordingly.

## 16.2 PIP Switches

You specify the operations that PIP is to perform with one or more switch specifications in the command line (see Section 16.1.1). A switch is always preceded by a slash (/) and, where more than one is specified, separated by slashes. You terminate the command line with the RETURN key.

Table 16-1 lists the PIP switches, their format, and the operations that they perform. The upper-case characters in each switch name indicate the minimally permissible abbreviation for that switch.

**Table 16-1: PIP Switches**

Switch	Format	Meaning
<b>Informational switches</b>		
HELP	/HE	Causes PIP to print a text message at your terminal that describes file specifications, switches, and options.
<b>File transfer switches</b>		
none		See the description of /BL.
ACcess	/AC	For disk input only, causes PIP to change the file's last access date to the current date. If you do not specify this switch, PIP preserves the last access date.
APPend EXTend	/AP /EX	For disk output only, causes PIP to append the input file to the output file (extend the output file). If no output file exists, PIP prints a warning message and creates the file.
ASciI	/AS	Formatted ASCII transfer, causes PIP to ignore nulls, parity bits, and rubout characters.
BLOCK	/BL	Causes PIP to perform a block-by-block transfer with no data translation.
Block SIZE	/BSIZE:n	For magnetic tape output only, causes PIP to use the physical blocksize specified in n (in bytes) on data written to the tape.
CLuster size	/CL:n	For disk only, sets the output file cluster size to n. To have effect, this switch must be adjacent to the output file specification. If this switch is omitted from a disk-to-disk transfer, the current cluster size is preserved.
IGnore GO	/IG /GO	Causes PIP to ignore ?USER DATA ON DEVICE errors.

(continued on next page)

**Table 16-1: PIP Switches (Cont.)**

Switch	Format	Meaning
MOde	/MO:n	Sets the mode in which the file is to be opened. To have effect, this switch must be adjacent to the file specification. For an explanation of modes, see the <i>RSTS/E Programming Manual</i> .
NEw file	/NE	For disk output only, creates a new file with the current date of creation and access.
RETain	/RET	For disk output only, retains creation and access date of input file.
NOAttributes	/NOA	Causes PIP to transfer the file without writing attributes to the output file.
PRotect	/PR /PR:NOWARN	Causes PIP to print an error message and ignore the operation if an output file already exists. The :NOWARN option suppresses the printing of the error.
Run-Time System	/RTS:name	Sets the output file's run-time system name to that specified in name. If no output file is specified, PIP renames the input file's run-time system. PIP's default action is described in Section 16.2.2.12.
UPdate	/UP	Causes PIP to open a pre-existing disk file and overwrite the existing data. If no file exists, PIP creates a new one.
<b>Files with attributes, translation switches and options</b>		
RMS	/RMS	For input disk files, translates RMS format to formatted ASCII or formatted binary.  For output disk files, translates formatted ASCII or formatted binary to RMS variable length records.  For non-disk transfers, translation depends on current format (see Section 16.2.3).
	/RMS:FA	If the /RMS switch is not specified for disk files, no translation is performed. The switch cannot be applied to both input and output files in the same command line.  For input or output files, specifies formatted ASCII in the translation.
	/RMS:FB	For input or output files, specifies formatted binary in the translation.
	/RMS:FTN	For input files, translates FORTRAN carriage control to formatted ASCII. For non-disk output files, translation is automatic.
	/RMS:IM	For input files only, performs the same as /RMS but no data translation (i.e., suppresses transfer of attributes and removes RMS variable length header information).
	/RMS:PRN	For input disk files, translates RMS print files to formatted ASCII.

(continued on next page)

**Table 16-1: PIP Switches (Cont.)**

Switch	Format	Meaning
<b>Date switches</b>		
AFter	/AF:dd-mmm-yy	Causes PIP to include only files that were created after, but not on, the given date.
BEfore	/BE:dd-mmm-yy	Causes PIP to include only files that were created before, but not on, the given date.
ON	/ON:dd-mmm-yy	Causes PIP to include only files that were created on the given date.
SINce	/SIN:dd-mmm-yy	Causes PIP to include only files that were created on or after the given date.
TOday	/TO	Causes PIP to include only files that were created on the current date.
UNtil	/UN:dd-mmm-yy	Causes PIP to include only files that were created on or before the given date.
Date of Last Access	/DLA	For disk only, causes PIP to use the date of last access in file operations.
CREation	/CRE	Causes PIP to use the date of creation in file operations.
<b>File operation switches - general</b>		
HAlt	/HA	Causes PIP to halt a magnetic tape wildcard search as soon as it detects a file that does not match.
INspect	/IN	Causes PIP to display (one-at-a-time) the filenames that match a wildcard specification. If used in transfer or /DE operations, as each filename is displayed, type Y (for yes) to transfer or delete; type any other character to omit. Type CTRL/Z to end the display and operation.
	/IN:S	Prints filename, extension, filesize, protection code, creation date, and last access date.
LOG WAtch	/LO /WA	Causes PIP to print a report on all actions taken during execution. The report is printed at your terminal.
NO Log	/NOL	Causes PIP to suppress the printing of a report on actions taken during execution.
NOrewind	/NO /RW:NO	Causes PIP to suppress the default rewind on magnetic tape prior to an input file search.
VErsion IDentify	/VE /ID	Causes PIP to print the program's current version number.

(continued on next page)

**Table 16-1: PIP Switches (Cont.)**

Switch	Format	Meaning
<b>File operation switches - deletion</b>		
DElete	/DE	For disk and DECtape only, causes PIP to delete the specified file.
	/DE:NO	Causes PIP to suppress the printing of an error message if the file to be deleted does not exist.
ERase WIPE Out	/ER /WIPE or /WO	For disk, causes PIP to overwrite the file with zeroes prior to deletion. These switches are synonymous and must be used in conjunction with the /DE switch.
<b>File operation switch - rename</b>		
REname	/RE:option	For disk and DECtape only, causes PIP to rename the input file to that of the output file.
<b>File operation switches - listing</b>		
BRief Fast	/BR /F	Causes PIP to print a brief directory listing.
DIRECTory LIst Slow	/DI /LI /S	Causes PIP to print a full directory listing. The listing switches accept option specifications that modify the directory listing (see Section 16.2.8).
<b>File operation switches - zeroing</b>		
ZERO	/ZE	Causes PIP to zero (initialize) the directory of an account or device or initialize the labels on a magnetic tape.
DENsity	/ZE/DEN:n	For magnetic tape only, sets the tape density prior to the zero operation.
PARity	/ZE/PAR:ODD /ZE/PAR:EVEN	For magnetic tape only, sets the parity of the tape prior to the zero operation.
<b>Privileged only switches</b>		
LOCK	/LOCK	Causes the system to lock PIP in memory for the duration of the current operation. Note that /LOCK is not abbreviated.
PRIORity	/PRIOR	Causes the system to run PIP at special priority for the duration of the current operation.

### 16.2.1 PIP Information Switch (/HE)

The /HE switch (/HELP) causes PIP to print a file that contains information on PIP to your terminal. This file describes the input and output file specifications required by PIP as well as the legal switches and options.

### **16.2.2 File Transfer Operations**

File transfer operations have the following command format:

```
*output/sw=input/sw
```

where output is a single file specification, and input can be one to six file specifications as described in Section 16.1.1. If you specify only one input file, that file is copied to the output file (the input file is unchanged). If you specify more than one input file, copies of the input files are concatenated and written to the output file (again, the input files are unchanged).

You can also specify one or more switches (described in the following sections) to modify the file transfer. If you do not specify a switch on disk-to-disk transfer, a block mode transfer (/BL, see Section 16.2.2.4) is performed by default. A block mode transfer causes PIP to copy the input file data, block-by-block, and any attributes on the input file into the output file. The transfer continues until the last block of the input file is copied.

If you do not specify a switch on disk to non-disk file transfers, a file transfer with format translation (see Section 16.2.3) is performed by default. That is, the transfer of RMS files with attributes to non-disk structured devices can cause PIP to translate the file to ASCII stream or formatted binary, depending on the attributes of the file. However, if the file contains fixed length records of 512 bytes (for example, a task image or library file), PIP does not translate the file but, rather, performs a block mode transfer. Do not use PIP to transfer a bootable magnetic tape between tapes of different densities. The bootstrap block on the tape works at only one density.

Because PIP can write magnetic tapes that conform to a subset of the ANSI (American National Standard Institute) standard X3.27-1978 – magnetic tape and file structure for information interchange, file transfer operations with tape can preserve file attribute and data format information.

To ensure the proper transfer of information, the following procedures are recommended.

#### **RSTS/E to RSTS/E tape transfer:**

1. Mount the output magnetic tape and use the ASSIGN command to specify ANSI label format. For example:

```
ASSIGN MTO:,ANSI
```

2. Use PIP to initialize the tape, write a volume ID, and set density and parity, if desired (see Section 16.2.9). For example:

```
PIP MTO:TAPIT/ZERO/DEN:800/PARITY:ODD
```

3. Use the /BL switch (see Section 16.2.2.4) to copy the files to the tape. For example:

```
PIP MTO:FORMS.DAT=FORMS.DAT/BL
```

The output file is created in U (undefined) format and all attribute information is retained in the file header label. Note that this type of transfer is only used between RSTS/E systems. Only PIP recognizes U format; it is not defined by ANSI standard X3.27-1978.

#### RSTS/E to non-RSTS/E transfer:

1. Mount, ASSIGN, and initialize the output tape as described in the previous steps 1 and 2.
2. To transfer a file without attributes that contains stream ASCII data to ANSI D record format, use the PIP /RMS:FA switch and option (see Section 16.2.3.1). For example:

```
PIP MMO:REPORT.DAT/RMS:FA=REPORT.DAT
```

To transfer FCS (RSX File Control Services) or RMS files with attributes, no switch is needed (see Section 16.2.3). However, only RMS sequential organization files can be copied. Thus, to transfer RMS relative and indexed organization files to ANSI labeled tape, you must first use the RMS conversion utility (RMSCNV) to convert them to sequential organization. Sequential files with attributes will be translated to their appropriate ANSI record format as described in Section 16.2.3.

#### Tape to disk transfer:

1. Mount the input tape and use the ASSIGN command to specify ANSI label format, or use the UOUNT program to mount and assign the tape (see Section 19.2). For example:

```
MOUNT MTO:/ANSI/NOVERIFY/DEN:800/PARITY:ODD
```

2. Use PIP to copy the tape file to disk. For example:

```
PIP FORM.DAT=MTO:FORM.DAT
```

PIP reconstructs the attributes of the file from tape and restores the file to its former format.

If the file that is being transferred to disk was originally transferred to tape with the PIP /RMS:FA switch and option (to convert from stream ASCII to ANSI D format), the output disk file will be an RMS variable length record file with implied carriage return.

Note that you can use the /CL switch (see Section 16.2.2.6) to ensure that the files are copied with the correct cluster size. Also, if the tape contains RMS relative or indexed organization files that were converted to sequential organization, you must use RMSCNV to convert them back to their original format. Note that if the file is contained within one volume (tape reel), you can use either RMSCNV to convert to relative organization format or RMSIFL to load an indexed file directly from tape. For information on these RMS utility programs, refer to the *RMS-11 User's Guide*.

## NOTE

Because PIP has the ability to transfer files that are larger than 65535 blocks, PIP is the only method whereby you can selectively backup large files. That is, the RSTS/E BACKUP system program (see Chapter 17) can not preserve files that are larger than 65535 blocks; the RSTS/E SAVE/RESTORE system program (see the *RSTS/E System Manager's Guide*) can preserve large files but only as an entire disk-to-disk or disk-to-tape copy. Thus, to preserve selected files from one disk to another disk or tape when those files are larger than 65535 blocks, use the PIP file transfer.

Input and output file specifications can also contain wildcard characters as described in Section 16.1.2. For example, the command line:

```
*DL1:*,*/CL:4=[2,2]*.MAC
```

causes PIP to copy all files with a .MAC extension in account [2,2] on the public structure to your account on RL01 unit 1. Protection codes are preserved and all output files are created with a cluster size of 4 (see Section 16.2.2.6). Because PIP substitutes wildcards for missing elements of the command line, the following command is equivalent to the previous one:

```
*DL1:/CL:4=[2,2].MAC
```

Note that asterisk wildcards are legal for output file accounts and for input file accounts if the input file is on disk or magnetic tape.

If you specify a non-file structured device in the input file specification, PIP ignores the filename and extension even if wildcards are present.

If you transfer data to the line printer and the printer goes off-line, PIP prints the message:

```
LINE PRINTER HUNG - PUT ONLINE TO CONTINUE OR TYPE CTRL/C
```

Following the message, PIP causes the terminal bell to signal every ten seconds until the printer is on-line or you type a CTRL/C. If you type a CTRL/C, PIP returns to prompt level.

**16.2.2.1 Access Switch (/AC)** — The /AC switch is applied only to disk resident input files. When you specify this switch, PIP changes the input file's date of last access to the current date prior to copying the file to the output file. If you do not specify this switch, PIP preserves the file's date of last access during the transfer. Note that even if the input disk is initialized for an access update on date of last write, the /AC switch causes the access date to change.

**16.2.2.2 Append and Extend Switches (/AP and /EX)** — The /AP and /EX switches are applied only to output files that are resident on disk. When you specify either of these switches, PIP transfers the data from the input files (which can be on tape or disk) and appends it to the end of the output file. This has the effect of extending the output file. For example:

```
*A.DAT=B.DAT/AP
```

causes A.DAT to contain its original data followed by the data that was transferred from B.DAT. Note that the use of these switches is not recommended if the output file has attributes and was created under control of the RSX File Control Services or RMS, because the end-of-file attribute is not updated.

If the specified output file does not exist, PIP returns the error ?CAN'T FIND FILE OR ACCOUNT and creates the file.

**16.2.2.3 ASCII Switch (/AS)** — The /AS switch is applied to output files that contain formatted ASCII data. When you specify this switch, PIP transfers the data from the input file, discards null and RUBOUT characters, and copies the data to the output file. This switch can be useful following an append transfer (see Section 16.2.2.2). That is, the /AP switch causes PIP to append input data to the last free block of the output file. If the last block contains null characters, these characters remain in the file following the transfer. You can use the /AS switch in a subsequent transfer to delete the null characters.

You should not use the /AS switch to transfer files that contain 8-bit binary data. The bytes in a virtual core array or Record I/O file can take any pattern of 8 bits, one of which can be CTRL/Z. Because PIP terminates the transfer when it encounters CTRL/Z, some data can be lost. To transfer such a file, use the /BL (Block) switch described in Section 16.2.2.4.

**16.2.2.4 Block Switch (/BL)** — The /BL switch causes PIP to perform a block-by-block image copy of the input file to the output file. No translation is performed on the data and all of the file's attributes are preserved. However, if an RMS sequential file is being transferred to a non-disk device without an /RMS or /BL switch in the command line, PIP automatically converts the RMS format to formatted ASCII or formatted binary. To override this conversion and preserve the RMS format, specify the /BL switch. The /RMS switch is described in Section 16.2.3.1. If you specify the /BL switch on a transfer to an ANSI labeled magnetic tape, PIP writes U format records and performs the block mode transfer. Note that you should use the /BL switch to correctly transfer RMS relative and indexed files between RSTS/E systems.

If you specify a single contiguous input disk file, the /BL switch, and a single output disk file, PIP attempts to create a contiguous disk file. If PIP fails to create the output file contiguously, the file is created non-contiguous and the % FILE CREATED NON-CONTIGUOUS message is printed.

When PIP performs a block mode transfer (either by default or by means of a /BL specification) from a non-disk device to a disk, it examines the output file extension. If the extension is executable (.BAC, .TSK, .SAV, etc.), PIP creates a compiled file (sets the compiled bit) and names the file as executable under its appropriate run-time system.

#### NOTE

PIP determines if a file is compiled by comparing it to a list of default executable extensions. PIP creates this list from the list of installed run-time systems that are current when PIP is invoked. If PIP detects conflicting extensions (for example, RSX and BASIC-PLUS-2 both use .TSK), PIP assigns the name of the first run-time system on its list whose extension matches the output file extension. See Section 16.2.2.12.

Note that a block mode transfer is actually an image mode transfer. Thus, a /BL specification allows transfers between DECtape (510 byte block) and disk or magnetic tape (512 byte block). When such a transfer takes place, /BL causes PIP to transfer each DECtape block to a disk block and fill the excess bytes in the last block with nulls. For a disk to DECtape transfer, the two extra bytes are placed in a new block; PIP does not pad excess bytes with nulls until the last output block is reached.

**16.2.2.5 Block Size Switch (/BSIZE:n)** — The /BSIZE switch allows you to specify a blocksize for output to magnetic tape. If this switch is not used, PIP transfers data to an output magnetic tape with a default blocksize of 512 bytes. The /BSIZE:n switch, where n is an even integer in the range of 18 to 4096 bytes, can be used to create a magnetic tape file whose blocksize is larger or smaller than the default of 512 bytes. If an illegal value for n is specified, PIP prints a BAD BLOCK SIZE error message.

The use of the /BSIZE switch is subject to the following restrictions:

1. The switch only applies to output on magnetic tape; PIP automatically handles input magnetic tape with blocksizes other than 512 bytes.
2. The switch can only be used for magnetic tape; its use on disk is ignored.
3. If the output tape is in ANSI format and is intended for interchange on another operating system, the blocksize specification must be between 18 and 2048 bytes.
4. If the output tape is intended for use on the RT-11 operating system, the blocksize must be 512 bytes.
5. The use of the /BSIZE switch can increase the amount of buffer space used by PIP. Thus, when reading or writing blocksizes larger than 512, invoke PIP with the CCL /SIZE:n switch or, if under the RT-11 Run-Time System, with the SIZE n immediate mode command.

**16.2.2.6 Clustersize Switch (/CL:n)** — The /CL:n switch specifies the cluster size in an output disk file specification. The cluster size you specify in n becomes the cluster size of the output file or files that PIP creates. The size you specify in n must be:

1. a power of 2,
2. greater than or equal to the pack cluster size of the disk involved in the transfer,
3. less than or equal to 256, or,
4. the negative of a power of 2 to specify either a cluster size of -n or the pack cluster size of the output device, whichever is greater

When this switch is present on the PIP command line, it must immediately follow the output filename (or the /MO switch if it is also present, see Section 16.2.2.8). If you do not include the /CL switch in the command line, PIP preserves the cluster size of the input file during the transfer. For non-disk devices, the switch is ignored. For a complete discussion of cluster size, refer to the *RSTS/E Programming Manual*.

**16.2.2.7 Go or Ignore Switches (/GO or /IG)** — You can use the /GO or /IG switches to cause PIP to ignore ?USER DATA ERROR ON DEVICE or magnetic tape record length errors. If this error occurs and one of these switches is present on the command line, PIP ignores the error and continues with the operation. With these switches, you can use PIP to salvage files from damaged or compromised media.

**16.2.2.8 Mode Switch (/MO:n)** — The /MO:n switch allows you to specify the mode in which files are to be opened. That is, RSTS/E allows you to specify modes in OPEN statements that set special characteristics for card readers, disks, flexible diskettes, line printers, magnetic tapes, and terminals. These mode values and their uses are described in the *RSTS/E Programming Manual*. Note that when you include the /MO switch in the command line, it must immediately follow the filename or, if specified, the /CL switch (see Section 16.2.2.6).

**16.2.2.9 New File and Retain Switches (/NE, /RET)** — The /NE switch is specified for disk output files and causes PIP to create a new file with the current date as both date of creation and date of last access. The /RET switch is also specified for disk output files, but unlike /NE, it causes PIP to retain the creation date and last access date of the input file. The /RET switch is the default.\*

---

\*The system manager can install an optional feature patch that causes the /NE switch to be the default.

**16.2.2.10 No Attributes Switch (/NOA)** — The /NOA switch causes PIP to suppress the transfer of disk file attributes. That is, if the input file contains attribute information that you do not want to copy to the output file, specify /NOA in the command line. The presence of the switch instructs PIP to refrain from writing the input file's attributes to the output file. Without this switch, attributes are automatically transferred.

**16.2.2.11 Protect Switch (/PR)** — For an output file, the /PR switch prevents PIP from making a file transfer if the specified output file already exists. This switch allows you to protect the output file from any possible deletion or overwrite. Note that the action of this switch is independent of the file's protection code, which can also prohibit writing to the file.

When you specify /PR and the output file does not exist, PIP creates the file and transfers the data. No error is returned. When you specify /PR and the output file does exist, PIP prints the ?NAME OR ACCOUNT ALREADY EXISTS error and cancels the data transfer. If you specify the NOWARN option (/PR:NOWARN) and the output file exists, PIP cancels the data transfer but does not print an error.

**16.2.2.12 Run-Time System Name Switch (/RTS:name)** — For a file structured disk transfer, the /RTS switch allows you to change the run-time system name of the output file. If there is no output file specification, PIP changes the name of the input file's run-time system to the name you specify in the switch.

If you do not include the /RTS switch in the command line, PIP transfers a run-time system name to the output file based on the following conditions:

1. if the input file is on a file structured disk, the input file's run-time system name is transferred to the output file, or
2. PIP searches for a run-time system whose runnable extension matches the output file extension. The first such run-time system encountered is used to assign a name to the output file. Note that if this condition is met, PIP also assigns a compiled protection code to the output file.
3. if neither condition 1 nor condition 2 is met, PIP assigns the run-time system name under which it is running to the output file.

**16.2.2.13 Update Switch (/UP)** — The /UP switch is used for output files to disk and causes PIP to attempt an update in existing files of the same name before it creates a new file. That is, when you include /UP in the command line, PIP searches the output device for files which match the output file specification. If it finds a matching file, PIP opens the file and replaces the file's data with that of the corresponding input file. If PIP is unable to locate a matching output file, it creates a new file.

If a matching output file is found that is larger than the corresponding input file, PIP performs the following operations:

1. PIP writes the input file's data to the output file without disturbing the output file's excess data. For example, a 34-block input file changes only the first 34 blocks of a 48-block output file.
2. if the /LO or /WA switches are also specified (see Section 16.2.5.3), PIP reports that the input file was COPIED INTO PREFIX of the output file.

If a matching output file is found that is shorter than the corresponding input file, PIP extends the output file to contain the excess data. For example, if the input file is 48 blocks and the output file is 34 blocks, the output file is extended to 48 blocks and its data is replaced by that of the input file.

**16.2.2.14 Multi-Volume ANSI Magnetic Tape File Transfer** — PIP allows you to transfer files from any input device to one or more ANSI labeled magnetic tapes, thus creating a volume set. You can also begin a file on one volume of the set and end it on another, assign identification labels to the volumes of the set, and transfer the content of the volume set to any output device. Note that DOS format does not allow files to be segmented across tape volumes, and thus does not allow multi-volume transfers.

To perform a multi-volume transfer, the output magnetic tapes must be initialized (with the /ZE switch, see Section 16.2.9). The first tape of the output set may contain some data, but each succeeding tape must be free of data. PIP allows you to specify volume identifications for each output tape, which allows you to create an identifiable volume set containing the transferred data.\*

When you initiate a multi-volume transfer, PIP prints a series of messages that notify you of the end of the output tape. PIP then asks you to specify the device name and unit number of the second output tape (the second volume). This process is repeated for each end-of-tape until all of the specified input is exhausted. Also, after PIP reaches the end of an output tape volume, it rewinds the tape and takes it off-line.

To ensure that files are retrieved in the correct order, PIP assigns section and sequence numbers to the files it transfers. These numbers are checked when files are transferred from the tape volumes. These numbers are internal and not user-specified.

PIP allows you to use file transfer switches (see Section 16.2.2) on multi-volume transfers. The directory switches (see Section 16.2.8) however, return directory listings on only a single volume. Thus, if a file is segmented across volumes, a directory of both volumes is required to show the true state of the file.

---

\*The system manager can install an optional feature patch that causes PIP to require a volume ID specification.

Consider the following example, which transfers the contents of a user account to three magnetic tape volumes:

```
RUN $PIP  
*MM0:*,*=20,25*,*/BL  
  
%END OF ANSI MAGTAPE OUTPUT VOLUME HAS  
%BEEN REACHED  
  
%PLEASE TYPE THE DEVICE NAME AND UNIT  
%NUMBER OF THE DRIVE WHERE THE NEXT  
%VOLUME MAY BE FOUND  
  
?MM1:TAPE2  
  
%END OF ANSI MAGTAPE OUTPUT VOLUME HAS  
%BEEN REACHED  
  
%PLEASE TYPE THE DEVICE NAME AND UNIT  
%NUMBER OF THE DRIVE WHERE THE NEXT  
%VOLUME MAY BE FOUND  
  
?MM0:TAPE3
```

In this example, you specify the block mode transfer of all files in account [20,25] on the system disk to the tape mounted on MM0:. When PIP reaches the end of the output tape, it prints a message that requests the next volume. In response to the prompt, type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the output volume ID, if specified, and that the tape contains ANSI labels and is initialized. If the ID is valid, the tape is properly initialized, and free of data, PIP continues with the transfer. This process is repeated each time the end of a tape is reached until all of the specified files on the input device are transferred.

When the transfer is complete, PIP prints its asterisk prompt.

Consider the following example, which transfers the contents of three tape volumes to a user account:

```
RUN $PIP  
*[20,25]=MM0:*,*/BL  
  
%END OF ANSI MAGTAPE INPUT VOLUME HAS  
%BEEN REACHED  
  
%PLEASE TYPE THE DEVICE NAME AND UNIT  
%NUMBER OF THE DRIVE WHERE THE NEXT  
%VOLUME MAY BE FOUND  
  
?MM1:TAPE2  
  
%END OF ANSI MAGTAPE INPUT VOLUME HAS  
%BEEN REACHED
```

%PLEASE TYPE THE DEVICE NAME AND UNIT  
%NUMBER OF THE DRIVE WHERE THE NEXT  
%VOLUME MAY BE FOUND

?MM0:TAPE3

\*

In this example, you specify the block mode transfer of all files on the tape mounted on MM0: to account [20,25] on the system disk. When PIP reaches the end of the input tape, it prints a message that requests the next volume. In response to the prompt, type the device name and unit number (MM1:) and the optional volume ID (TAPE2). PIP checks the input volume ID, if specified, and ensures that the file is in the correct sequence (i.e., PIP checks the section and sequence numbers it assigned to the files). PIP also checks that the input tape contains ANSI labels. If the volume ID is valid, the files correct, and the tape has ANSI labels, PIP continues with the transfer. This process is repeated each time the end of a tape is reached until all of the files on the input tape volumes are transferred.

When the transfer is complete, PIP prints its asterisk prompt.

Note that if you respond to the volume prompt (?) with a CTRL/Z, PIP interrupts the transfer and terminates.

### 16.2.3 File Operations Involving Attributes

A RSTS/E disk file may or may not have attributes, depending on the type of file. For example, all RMS files have attributes, while BASIC-PLUS files have no attributes. In addition, RMS files and non-RMS files are formatted differently. That is, RMS files can have fixed length or variable length record format and one of several carriage control formats: implied, embedded, FORTRAN, or print format. Non-RMS files can have stream (formatted ASCII) records, binary record format, BASIC-PLUS Record I/O, or virtual array files. Similarly, DOS labeled magnetic tape differs from ANSI labeled tape in file attributes and format. DOS labeled tapes have no attributes and can contain formatted ASCII or formatted binary files. ANSI labeled tapes have limited file attribute information that is similar to RMS sequential files.

As PIP transfers input disk file data to output disk files, it allows you to translate from one format to another. PIP provides this capability by means of the /RMS switch and its options as described in Section 16.2.3.1.

If you do not specify the /RMS switch on a transfer of disk files to DOS labeled magnetic tape, the following default transfers occur:

1. If the input RMS file has implied or RMS FORTRAN carriage control, or if it is a print format file, PIP creates an output file in stream (formatted ASCII) format.

2. In all other cases, PIP creates a file in formatted binary.

If you do not specify the /RMS switch on a transfer of disk files to ANSI labeled magnetic tape, the following default transfers occur:

1. If the input RMS file contains fixed length records, PIP creates an output magnetic tape file in ANSI F record format (fixed length format).
2. If the input RMS file contains variable length records, PIP creates an output magnetic tape in ANSI D record format (variable length format).
3. If the input file contains no attributes, PIP creates an output magnetic tape file in U (undefined) record format. Tapes with U format may not be transferable to non-RSTS/E systems.

If you do not specify the /RMS switch on an ANSI magnetic tape to disk transfer, the following default transfers occur:

1. If the input magnetic tape file is in ANSI F record format, PIP creates an output disk file with RMS fixed length records.
2. If the input magnetic tape file is in ANSI D record format, PIP creates an output disk file with RMS variable length records.
3. If the input magnetic tape contains ANSI labels and is not in F or D record format or the tape format does not contain a header label 2 (HDR2), PIP performs a block mode transfer (see Section 16.2.2).

The use of the PIP /RMS switch options to transfer files with attributes is subject to the following restrictions:

1. PIP cannot concatenate two or more RMS data files. An attempt to do so results in a file with invalid attributes. That is, all the data in the file is not accessible.
2. If a file contains variable length records or fixed length records with RMS FORTRAN carriage control, PIP can translate the file to stream (formatted ASCII) records.
3. If a file is an object module, PIP can translate the file to formatted binary.
4. To concatenate RMS files, use the RMS utility RMSCNV as described in the *RMS-11 User's Guide*.

**16.2.3.1 RMS Switch (/RMS:options)** — The /RMS switch is used only on input or output files that are resident on disk devices or ANSI labeled magnetic tape. The switch allows you to force the translation of the file's data format when PIP transfers the file.

If you include the switch in the command line for input disk files, PIP translates RMS format to formatted ASCII or formatted binary. If you include the switch in the command line for output disk files, PIP translates formatted ASCII or formatted binary to RMS format.

If you do not specify this switch, PIP transfers the file as described in Section 16.2.3. Note that you cannot apply this switch to both input and output files on the same command line.

When you specify /RMS, PIP determines whether to translate between RMS format and formatted ASCII or formatted binary. However, PIP allows you to include one of five options in the switch to specify a particular translation. The options available to you are as follows:

- :FA For input or output disk files, translate to or from formatted ASCII.
- :FB For input or output disk files, translate to or from formatted binary.
- :FTN For input disk files, translate RMS FORTRAN carriage control to formatted ASCII.
- :IM For input disk files, translate from RMS format to formatted ASCII or formatted binary (as with the switch with no option), but also remove record lengths from variable length records and padding bytes.
- :PRN For input disk files, translate RMS print file format to formatted ASCII.

Note that to translate a Task Builder .MAP file to formatted ASCII, you must use the /RMS:FA switch.

#### **16.2.4 Date Related Switches (/DLA, /CRE, /AF, /BE, /ON, /SIN, /TO, and /UN)**

The date switches allow you to specify the inclusion or exclusion of certain files in a PIP transfer based on the creation date or date of last access of those files. These switches are especially useful for modifying wildcard file specifications.

By default, file transfers are based on the creation date.\* However, you can specify a /DLA or /CRE switch in the command line to override the installed default. The /DLA switch causes PIP to use the date of last access; the /CRE switch causes PIP to use the date of creation.

---

\*The system manager can install an optional features patch that causes PIP to use the date of last access instead of the date of creation.

The Today switch, /TO, does not contain an argument and causes PIP to operate only on those files that were created or accessed on the current date.

The On switch, /ON:date, contains a single argument that specifies a date in the form:

/ON: dd-mmm-yy

where dd is the day of the month, mmm are the first three letters of the month, and yy are the last two digits of the year. When you include the /ON switch in the command line, PIP operates only on those files with a date that matches the switch specification.

The After, Before, Since, and Until switches, when used in combination, allow you to specify a range of dates. That is, only one switch is necessary in the PIP command line but two switches can be used to specify an upper boundary and a lower boundary. When you specify a range in the command line, PIP processes only those files that fall within the range and ignores those whose dates are outside the specification. Note that because of a conflict between the PIP switches /SIZE and /SINCE, the minimum abbreviation for the /SINCE switch is /SIN.

You specify a date in these switches in the following format:

: dd-mmm-yy

as described for the /ON switch. Note that you can omit the year from a date specification and accept the default of the current year.

Consider the following examples:

Switches	Lower Boundary	Upper Boundary
/SIN:10-APR-79 /UN:23-SEP-79	10-APR-79	23-SEP-79
/AF:10-APR-79 /UN:23-SEP-79	11-APR-79	23-SEP-79
/SIN:19-APR-79 /BE:06-JUL-79	19-APR-79	05-JUL-79
/AF:08-MAY-79 /BE:22-SEP-79	09-MAY-79	21-SEP-79

### 16.2.5 File Operation Switches

The switches described in the following sections (Section 16.2.5.1 through 16.2.5.5) can be applied to the four major operations performed by PIP; copy, delete, rename, and directory listing operations. You can use these switches; /HA, /IN, /LO, /NOL, and /WA, /NO and /RW:NO to speed PIP operations and as auxiliaries to those operations.

**16.2.5.1 Halt Switch (/HA)** — You can specify the /HA switch when the PIP operation is to conduct wildcard searches on magnetic tape devices. When /HA is included in the command line, it causes PIP to stop processing when it encounters a file on magnetic tape that does not match the wildcard specification. Without this switch, PIP continues its search until it reaches the end of the tape.

**16.2.5.2 Inspect Switch (/IN:option)** — You can specify the /IN switch when the PIP operation involves wildcard specifications. The switch can appear at any point in the command line and, when specified, causes PIP to print at your terminal the filename of each file that matches the wildcard specifications. The filenames print one at a time and PIP allows you to respond with one of the following:

- Y (for yes) Causes PIP to execute the operation on this file and search for the next matching file.
- CTRL/Z Causes PIP to exit without executing the operation.
- any other response Causes PIP to skip this file and search for the next matching file.

Note that when the /IN switch appears in the command line, it applies to every input file wildcard specification.

The /IN switch accepts an option (/IN:S) that causes PIP to print the filename, extension, filesize, protection code, creation date, and date of last access of each file that matches the wildcard specifications.

**16.2.5.3 Log, No Log, or Watch Switches (/LO, /NOL, or /WA)** — The /LO or /WA switches are synonymous and can be specified at any point in the command line.\* When you specify /LO or /WA, PIP prints on your terminal a report on all actions taken during execution.

The /NOL switch is used to suppress printing of the execution report and is used when logging is enabled as the system default.

**16.2.5.4 No Rewind Switches (/NO or /RW:NO)** — When PIP begins its search for an input file on magnetic tape, it first rewinds the tape. You can specify the /NO or /RW:NO switches at any point in the command line and cause PIP to suppress the automatic rewind on tape. When these switches are specified, PIP begins its file search at the current tape position.

**16.2.5.5 Version or Identification Switch (/VE or /ID)** — You can specify the /VE or /ID switch at any point in the command line and cause PIP to print its version number on the terminal.

---

\*The system manager can install an optional feature patch which causes logging to occur by default.

### **16.2.6 File Deletion Switches (/DE, /ER, /WO, and /WIPE)**

When you apply the /DE switch to a command line input file specification, it causes PIP to delete all files that match that specification. The switch can only be applied to a disk or DECtape input file specification. If you do not specify a filename or an extension, PIP prints the following error:

```
?FILENAME AND EXTENSION MUST BE SPECIFIED
```

Note that PIP allows you to use wildcard characters in the file specification. If this is the case, you may also include the /IN switch (see Section 16.2.5.2) in the command line.

The /DE switch applies to all input file specifications on the command line in which it appears. Therefore, you should not attempt to specify any other PIP operation in a command line that contains a /DE switch.

PIP allows you to modify the /DE switch with a NO option, /DE:NO. When NO is appended, it instructs PIP to suppress the printing of an error message if the file to be deleted does not exist.

If the file to be deleted has the privileged (<128>) bit set in its protection code, PIP writes zeroes on the file, renames it to a non-privileged code, and deletes it. When a file with the privileged bit is deleted from the system by means of UNSAVE or KILL commands, the File Processor performs the same actions as the PIP delete. Therefore, if you desire such a delete operation, you should use PIP (which is faster) rather than the File Processor, which is subject to greater system demand.

In conjunction with the /DE switch, PIP allows you to specify an /ER, /WO, or /WIPE switch. When you include one of these switches with /DE in the command line, PIP writes zeroes in the file prior to its deletion. As with the /DE switch, these switches apply to every input file specification on the command line.

### **16.2.7 File Rename Switch (/RE:option)**

When you apply the /RE switch to a disk or DECtape input file specification, PIP changes the input filename to that of the output file specification.

The /RE switch can be used with wildcard specification to cause multiple file renames. Also, PIP requires that you specify only those elements of the output file specification that will change. For example:

```
PIP *.C??=STAT?.B?S/RE
```

causes PIP to change the extension of every file whose filename begins with STAT and whose extension begins with B and ends with S. The extension is changed, in each case, such that the first character is C. Note that if no input file specification appears on the command line, PIP renames all files in the current account on the public structure.

When you specify the /RE switch, PIP does not transfer any data.

Consider the following:

```
* newname.ext<prot> =oldname.ext/RE
```

upon execution, the file oldname.ext is renamed to newname.ext and its protection code is changed to that specified in the command line. However, the data in newname is unchanged.

To change only the protection code, include the new protection code on the input file specification and omit the output file specification. For example:

```
*oldname.ext<new prot>/RE
```

If you attempt to rename a file that does not exist, PIP returns the error ?CAN'T FIND FILE OR ACCOUNT, skips that operation, and continues processing. If you attempt to rename a file to a name that already exists, PIP returns the error ?NAME OR ACCOUNT NOW EXISTS, skips that operation, and continues processing. Note that you can use the NOWARN option (/RE:NOWARN) to suppress the printing of the ?NAME OR ACCOUNT NOW EXISTS message.

#### **16.2.8 Directory Listing Switches (/BR, /F, /DI, /LI, and /S)**

The PIP program provides five switches and several options that allow you to print a directory listing. This listing is similar to that which you can obtain using the DIRECT system program (see Section 15.1).

The /DI and /LI switches cause PIP to print a full directory listing and also accept options that allow you to modify the listing. Options are specified in the following format:

```
/DI:option
```

or

```
/LI:option
```

The /BR and /F switches do not accept an option specification, and when included in the command line, cause PIP to print an abbreviated form of the directory listing. That is, only the filenames and extensions are printed. The /S switch causes PIP to print a full directory that is similar to the slow listing obtained from the DIRECT system program (see Section 15.1).

The /DI, /LI, /S, /F, and /BR switches can list the directory of any file structured device. When one of these switches appears in a command line file specification, which can contain wildcards, only the files that match that specification are printed. PIP prints the listing on your terminal by default but you can output the listing to another device or file with an output specification.

Table 16-2 lists the options that you can specify to the /DI or /LI switches.

**Table 16-2: Directory Listing Options**

Option	Result
:AL	Prints filename, extension, and the number of blocks allocated to the file.
:AT	Same as :SA.
:CL	Prints filename, extension, and file cluster size.
:DA	Prints filename, extension, and file creation date.
:DI	Prints filename, extension, size, protection code, creation date, and column headers.
:EX	Prints filename and extension.
:FU	Full listing, print all information except file attributes.
:HD	Includes column headers in the listing.
:LA	Prints filename, extension, and date of last access or date of last write (depending on disk initialization).
:NA	Prints filename.
:OA	Prints filename, extension, and octal representation of file attributes.
:PR	Prints filename, extension, and protection code.
:RT	Prints filename, extension, and associated run-time system.
:S	Slow listing, prints all information.
:SA	Prints filename, extension, symbolic file attributes (see Section 11.6), and caching status. Caching is indicated by one of the following entries: CACHE:ON:RAN; file will be automatically cached randomly when open. CACHE:ON:SEQ; file will be automatically cached sequentially when open. CACHE:OFF:SEQ; file will not be automatically cached, but if cached, it will be cached sequentially.
:SI	Prints filename, extension, and file size.
:SU	Prints a summary report.
:SZ	See :SI.
:TI	Prints filename, extension, and date and time of creation.
:W	Prints filename and extension across the length of the page or screen.
:WI	Prints filenames only across the length of the page or screen.

Note that the :OA, :SA, :AT, and :S options cause PIP to print file attributes in octal or symbolic representation. Section 11.6 describes the format and meaning of these attribute representations.

### **16.2.9 Zeroing Directories Switch (/ZE)**

To cause PIP to zero (i.e., initialize) a device or account directory, include the /ZE switch in the PIP command line. When you specify this switch, it causes

PIP to remove all of the files stored in a specified account on disk or, if you specify a magnetic tape or DECtape, to initialize the volume mounted on the specified drive.

The use of the /ZE switch is subject to the following restrictions:

1. No output file specification can be present on the command line and only one input file specification is allowed. The format is as follows:

```
*dev:[proj,proj]/ZE
```

Note that only a privileged user can specify a zeroing of another's account.

2. To initialize an ANSI labeled magnetic tape, specify the device and a filename, which represents the volume label.
3. If a device is not specified, PIP assumes the public disk structure and the current account.

When PIP detects the /ZE switch in its command line, it prints the following prompt:

```
REALLY ZERO dev:[proj,proj] ?
```

Two possible replies are available to you:

- Y confirms that you wish PIP to zero the device or account.  
any other aborts the operation and returns you to the PIP command level.  
response

The /ZE switch also permits you to specify a change in the density or parity of the magnetic tape prior to the zero operation. System-wide defaults for magnetic tape density, parity, and label are set by the system manager during system initialization. If you wish the system to accept a tape with non-default characteristics, you must ASSIGN the tape prior to setting those characteristics (with MOUNT or PIP). Note that when the tape is DEASSIGNED, the system reverts to the default characteristics set during system initialization. The specifications for density and parity are as follows:

```
/ZE/DEN:n
```

and

```
/ZE/PAR:parity
```

The /ZE/DEN:n switch combination causes PIP to set the density of the tape prior to initializing the tape. This operation results in a tape initialized at the desired density. The densities that you can specify in n are as follows:

- 200 BPI for 7-track tape.  
556 BPI for 7-track tape.  
800 BPI for 7-track or 9-track tape.  
1600 BPI for 9-track tape.

The /ZE/PAR:parity switch combination causes PIP to set the desired parity of the tape prior to initializing the tape. This operation results in a tape initialized at the desired parity. The parities that you can specify are as follows:

/ZE/PAR:ODD for odd parity.  
/ZE/PAR:EVEN for even parity.

Note that 1600 BPI tape can only support odd parity. Also, unless the tape is to be processed on a system that requires even parity, odd parity is recommended for all output tapes.

A zero operation, which contains a volume label specification, is recommended for all new tapes.

#### **16.2.10 Privileged Only Operations (/LOCK and /PRIOR)**

The /LOCK and /PRIOR switches can be specified in a PIP command line only if the user is privileged.

When a privileged user specifies /LOCK in the command line, it causes the system to lock PIP into memory for the duration of the current operation.

When a privileged user specifies /PRIOR in the command line, it causes the system to run PIP at a special priority for the duration of the current operation. If this switch is not specified, PIP runs at the current user's default priority. When /PRIOR appears in the command line, the system boosts the priority of PIP by 4.

### **16.3 PIP Error Messages**

Two types of errors are possible during a PIP operation; RSTS/E system errors (such as ?CAN'T FIND FILE OR ACCOUNT) and PIP program errors. Appendix C describes the RSTS/E errors and the recovery actions that you can take. Errors that are distinctive to PIP are as follows:

```
?WRITE FAILURE  
?CLOSE FAILURE  
?READ FAILURE  
?LOOKUP FAILURE  
?ENTER FAILURE  
?RENAME FAILURE  
?ZERO FAILURE
```

These errors indicate problems within PIP itself and the only recovery is to retry the operation (the error may be due to a transient condition). If the error continues, submit an SPR along with a listing of terminal output and a directory of the files that were operated on. Refer to Section C.3 for more information on SPR submittals.

### **16.4 Copying Between Devices: The COPY Program**

All of the information on a DECTape, magnetic tape or RK05 disk can be copied by using the COPY system program and the /FC (fast copy) switch.

Note that COPY will not compensate for the presence of bad blocks. COPY is called as follows:

```
RUN $COPY
```

COPY prints a pound sign (#) when it is ready to receive instructions. To print a help message, type /HE. To copy a device, respond in the format shown below.

```
*new device=old device/FC
```

for example:

```
#DT1:=DT2:/FC
```

In this example, all of the data, programs and directories are copied, block by block, from the DECTape on unit 2 to the DECTape on unit 1. The original information on unit 2 is, of course, still intact.

Notice that all of the information on a device is copied; individual files cannot be specified.

Only magnetic tapes, DECTapes, and RK05 disk packs should be copied. In addition, information can be copied only to different units of the same device. That is, COPY can be used to copy a DECTape to another DECTape or a magnetic tape to another magnetic tape, but not to copy, say, a DECTape to magnetic tape or vice versa. If an attempt is made to copy the information on one type of device onto another type of device, the error message ?MUST HAVE SAME TYPE DEVICES is given. Do not use COPY to transfer a bootable magnetic tape between tapes of different densities. The bootstrap block on the tape works in only one density.

If COPY encounters a bad block, it aborts the operation and prints the following error message:

```
?USER DATA ERROR ON DEVICE dev:
```

where dev: is the device containing the bad block.

Finally, if the same unit number of a device is specified twice, the error message ?MUST HAVE DIFFERENT UNIT NUMBERS is given.

When the information is successfully copied from one device to another, the program terminates and the system prints READY.

To verify that the information on a device unit has been copied properly, issue the /VE (verify) option to the COPY program in either of the following ways:

```
#DT1:=DT2:/FC/VE
```

or

```
#DT1:=DT2:/NC/VE
```

Since verification is performed block by block, it requires as much time as the copy operation. Therefore, it is important to understand the difference between the commands. The first command, which includes the /FC switch, copies information from DT2 to DT1 and then verifies that the information was copied correctly. The second command, which must include the /NC (no copy) switch, does not copy information; it only verifies that the information on both DECTapes is identical.

The two sample commands are the only allowable forms for verifying. If you type a command string omitting the /FC or /NC option, the error message ?/FC OR /NC MUST BE SPECIFIED is returned.

If you specify both the /FC and /NC options in the same command line, the error message ?CANNOT SPECIFY BOTH /FC AND /NC is returned.

As with the /FC option, individual files cannot be specified with the /VE option; all of the information on a device is verified.

As the verification is performed, the first message COPY prints is:

BEGINNING VERIFICATION PASS

If all of the information has been copied correctly from one device to another, the next message COPY prints is:

VERIFICATION COMPLETE 0 BAD BLOCKS

READY

If, however, the information has not been copied correctly, COPY prints the decimal number of blocks in which inconsistencies appear. For example:

```
#DT1:=DT2:/FC/VE
BEGINNING VERIFICATION PASS
THE FOLLOWING BLOCKS ARE BAD:
17
31
89
VERIFICATION COMPLETE 3 BAD BLOCKS
READY
```

The /BL switch, which specifies blocksize, speeds up copying or copies magnetic tapes written with non-standard record sizes. To speed up copying, specify a larger blocksize. For example:

```
#DK0:=DK1:/BL:2048/FC
```

causes the disk on drive DK1: to be copied to the disk on drive DK0: in 2048-byte blocks, rather than the default 512-byte blocks. If you specify an illegal blocksize, the error message ?ILLEGAL BLOCK SIZE is returned. If you specify a blocksize that is too large, the error message ?MAXIMUM MEMORY EXCEEDED is returned.

To specify other than standard density and parity settings when copying magnetic tape, issue the /DENSITY:d and /PARITY:p options. These may be minimally abbreviated to /DE:d and /PA:p. In these options:

d can be:	p can be
200	ODD
556	EVEN
800	
DUMP	
1600 (phase-encoded)	

If the user specifies a density or a parity that does not appear in this list, the error message ?ERROR IN SPECIFYING DENSITY (or PARITY) is returned.

The following example illustrates /DENSITY and /PARITY:

```
*MTO:/DENSITY:556/PARITY:EVEN = MT1:/DENSITY:DUMP/FC
```

COPY reads tape unit 1 at 800 bits per inch dump mode and writes unit 0 in even parity at 556 BPI.

Several error messages are returned for general command errors. If you misplace or mistype an option, the program prints ?ERROR IN SPECIFYING SWITCHES. If you specify an illegal or nonexistent device, the program prints ?ILLEGAL DEVICE. If you specify anything besides a device in the input or output specification, the program prints ?ILLEGAL INPUT (or OUTPUT) SPECIFICATION. Finally, if you type a command string that the program cannot decode, it prints ?SYNTAX ERROR IN COMMAND STRING.

## 16.5 File Transfer Between Devices: The FIT Program

The system program FIT (File Transfer) allows you to perform file transfers between devices. The transfers that can be performed by FIT are as follows:

1. Transfer files between disk and TU58 DECTape II or between disk and RX01 or RX02 flexible diskette. FIT also allows you to list the directory of a DECTape II or diskette.
2. Transfer files between RSTS/E disk and RT-11 disk or between RSTS/E disk and RT-11 DECTape. FIT also allows you to list the directory of an RT-11 disk.
3. Transfer files from a DOS disk to a RSTS/E disk. FIT also allows you to list the directory of a DOS disk.

TU58 DECTape II and flexible diskette files are maintained in the RT-11 directory structure format. Thus, DECTape II and diskettes are directly transferrable between RSTS/E and RT-11 systems.

To invoke the FIT program, you must be logged into the system and type the following command:

```
RUN $FIT
```

If FIT is successfully invoked, it prints an identifying message followed by a FIT> prompt. The prompt indicates that FIT is prepared to accept a command line (see Section 16.5.1). To cause FIT to print a help text file, type the /HE switch in response to the prompt. To exit from FIT, type CTRL/Z in response to the prompt, for example:

```
FIT>^Z
```

```
Ready
```

### 16.5.1 Transferring Files with FIT

To transfer a file with FIT, type a command line of the following form in response to the FIT prompt:

```
FIT>[output[/sw]=]input[/sw][/WATCH]
```

where output and input are file specifications and /sw is an optional switch that specifies the structure of the device on which the file resides (RSTS/E, RT-11, or DOS). If you specify the optional /WATCH switch, FIT logs all file transfers on the terminal. Note that you can abbreviate the /WATCH switch to /W.

The input and output file specifications have the following form:

```
dev:[proj,prog]name.ext<prot>
```

where:

dev: is a device specification. If none is specified, the default is the public structure. The output device must not be a DOS disk. The input device can be a RSTS/E, RT-11, or DOS disk, or TU56 DECTape, TU58 DECTape II, or flexible diskette with RT-11 file structure.

proj,prog is a project-programmer number. If none is specified, the default is your current account. FIT ignores the project-programmer number if the device has an RT-11 file structure. Wildcards can be used as described in Section 16.1.2.

name.ext is a filename and extension. If none is specified for output, FIT uses the input filename and extension. An input filename must be specified. Wildcards can be used as described in Section 16.1.2.

<prot> is a protection code. If none is specified on output, FIT assigns your current default protection code. FIT ignores any input file protection code specification.

The legal input devices for a FIT file transfer are as follows:

- disk (RSTS/E format)
- RX01 or RX02 flexible diskette (RT-11 format)
- TU58 DECTape II (RT-11 format)
- disk or TU56 DECTape (RT-11 format)
- disk (DOS format)

The legal output devices (except for DOS input) for a FIT file transfer are as follows:

- disk (RSTS/E format)
- RX01 or RX02 flexible diskette (RT-11 format)
- TU58 DECTape II (RT-11 format)
- disk or TU56 DECTape (RT-11 format)
- line printer, keyboard, etc.
- DECTape (RSTS/E format)

If the input device is a DOS disk, the legal output devices are as follows:

- disk or DECTape (RSTS/E format)
- line printer or keyboard

The switch specification /sw can be one of the following:

- /RT11      The file is on an RT-11 file structured device. When this switch is specified, FIT will not process files with .BAD extensions.
- /DOS      The file is on a DOS format disk (legal only for an input file specification).
- /RSTS      The file is on a RSTS/E disk or device. This switch is illegal with flexible diskette and TU58 DECTape II.

These switches can be abbreviated to the first two letters (for example, /RT, /DO, and /RS).

If you do not specify a switch on the input or output specification, FIT attempts to determine the file structure of the device. If the device does not contain a valid RSTS/E, RT-11, or DOS directory structure, FIT prints the following error:

?NOT A RECOGNIZED DIRECTORY STRUCTURE

If this error occurs on the output device, it indicates that the device has not been properly initialized. To initialize (or zero) an RT-11 structured device, use the /ZE switch as described in Section 16.5.2.2. To initialize any other device, use PIP as described in Section 16.2.9.

If the specified input file does not exist on the device, FIT prints the following error:

?CAN'T FIND FILE OR ACCOUNT

If you do not specify a switch on the output specification, FIT assumes a default file structure based on the type of device. That is, if the device is TU58 DECTape II or flexible diskette, FIT assumes RT-11 file structure; if the device is disk or TU56 DECTape, FIT assumes RSTS/E file structure.

If the output device has an RT-11 file structure, the following errors are possible:

#### ?DIRECTORY OVERFLOW

The output device has no space for directory entries. It may be possible for you to use the /SQ switch (see Section 16.5.2.2) to free more space for directory. If this is not successful, you must delete some of the files on the device before more can be added. Note that if you intend to have many small files on the device, you should use the /N switch to allocate more than the default number of directory segments (see Section 16.5.2.2) when the device is initialized.

#### ?DEVICE FULL

The output device has no free space for the file transfer. It may be possible for you to use the /SQ switch (see Section 16.5.2.2) to create enough contiguous free space for the additional files. If this is not successful, you will have to delete enough files from the output device to add the files from the input device.

These errors are generated prior to the transfer of any data to the RT-11 structured device. If an error is encountered during a transfer, a RSTS/E error is reported and the transfer is aborted. Because the directory entry for the RT-11 file is not finalized until the transfer is complete, an error during transfer causes the space reserved for the file to be freed.

If an error is encountered during the open or transfer phase of a RSTS/E file operation, the appropriate RSTS/E error (see Appendix C) is generated and the transfer is aborted.

### 16.5.2 Maintaining RT-11 Format

In addition to transferring files to and from an RT-11 directory structured device, FIT can perform the following operations:

1. You can use the /ZE switch (see Section 16.5.2.2) to zero or initialize a device as an RT-11 device. This operation creates an RT-11 directory structure on the device and must be performed prior to transferring files to and from the device.
2. You can use the /LI switch (see Section 16.5.2.2) to obtain a directory listing of the RT-11 device.
3. You can use the /DE switch (see Section 16.5.2.1) to delete a file on an RT-11 device.
4. You can use the /SQ switch (see Section 16.5.2.2) to compress (squeeze) the data and directory of the device and allow more files to be stored on that device. This operation is useful if the available space on the device is fragmented.

Note that the RT-11 operating system requires that files used to contain bad blocks have a .BAD extension. The following procedure must be used to transfer files between RSTS/E and RT-11 systems on bad block disks:

1. A disk, which contains bad blocks, must be initialized on an RT-11 system before FIT can process it. The bad blocks must be located and contained in files with .BAD extensions.
2. Whenever the bad block disk is referenced in the FIT command line, the /RT switch (see Section 16.5.2.2) must be included with the file specification.

Using this procedure, you cause FIT to ignore any files with a .BAD extension on transfer to and from RT-11 devices. FIT also ignores .BAD files on delete operations. During squeeze operations, FIT prints a warning message and aborts the operation if it detects a .BAD file.

#### NOTE

FIT does not prevent the zeroing of RT-11 disks with bad blocks. When such a disk is zeroed, information on bad blocks is lost. Thus an RT-11 bad block disk should not be zeroed with FIT; rather, you should delete all of the non .BAD files with a wildcard delete operation.

**16.5.2.1 File Deletion on an RT-11 Format Device** — To delete a file on an RT-11 directory structured device, enter the following command format in response to the FIT> prompt:

```
FIT>dev:filename.ext/DEL[/WATCH]
```

where dev: identifies the device and filename.ext specifies the file to be deleted. Note that wildcards can be used in the filename and extension specifications (see Section 16.1.2). The /WATCH switch is optional and, if specified, causes FIT to log all file deletions at the terminal. The /WATCH switch can be abbreviated to /W. If the specified device is not RT-11 directory structured or if the file does not exist, FIT prints an error message. If an error is encountered by FIT as it accesses the device, a RSTS/E error is reported.

**16.5.2.2 RT-11 Device Switches** — You can perform zeroing, compressing, and listing operations on RT-11 directories by entering the following command format in response to the FIT> prompt:

```
FIT>dev:/sw[/sw][/sw]
```

where dev: indicates the device on which the operation is to be performed and /sw specifies the operation. The switches you specify can be one or more of the following:

/ZE [:m]      Zero (initialize) the device and build an RT-11 directory structure. If :m is specified, m extra words of directory information for each file is added to the minimal RT-11 directory structure.

If m is not specified, FIT adds 13 extra words by default to the directory structure. These words are used to store the run-time system name and file attributes. This switch can be abbreviated to /Z.

Note that RT-11 systems currently ignore these extra directory entries. However, if future releases of RT-11 attach significance to these words, a specification of /ZE:0 may be required for RT-11 files that are read by RT-11 systems. FIT currently ignores extra directory information unless there are 13 extra words (this may change in future releases). Therefore, you should only specify values of 0 or 13 in the /ZE switch.

- /N:n      Used with the /ZE switch to create n directory segments on the initialized device. Each segment consists of two 512-byte blocks. The value of n must be in the range of 1 to 31 and determines the size of the directory on the device (and, therefore, the number of files that can be stored on the device). The number of files that can fit on a device is determined by the following formula:

$$n * \text{INT}(507/(m+7))$$

where n is the number of directory segments and m is the number of extra words for each directory entry (that is, /N:n and /ZE:m, respectively).

Note that you may have to compress the device with the /SQ switch before the maximum number of files is reached.

If you do not specify n, FIT applies defaults based on the device type. These defaults are indicated in the following table. The table also lists the maximum number of files allowed on a particular device corresponding to your specification in /ZE:n.

Device	Default number of Directory segments		Max. number of files	
	/ZE:0	/ZE:13	/ZE:0	/ZE:13
TU58 DECTape II	4	10	288	250
RX01/RX02 Diskette	4	10	288	250
RK05 disk	16	31	1152	775
RK06 disk	31	31	2232	775

- /LI or /DI      List the directory of the device. The information listed includes filenames, extensions, file size, protection codes, dates of creation, and position on the device. If the device has 13 extra words for each directory entry (/ZE:13, the default value), the run-time system name and file attributes are also printed. Those sections of the device which are free space are marked <UNUSED> in the listing. To list a specific directory, include a filename and extension between the device specification and the

- /LI or /DI switch. Note that the filename and extension specification can contain wildcards. The /LI switch can be abbreviated to /L.
- /RT11      Used with the /LI or /DI switches. /RT11 indicates that the device must have RT-11 directory structure. That is, if this switch is not present and the device is a RSTS/E or DOS disk, a directory of the disk is listed. If this switch is present and the device is not an RT-11 device, an error is returned. This switch can be abbreviated to /RT.
- /SQ      Compress (squeeze) the device. This switch allows you to compress the files that are currently on the output device and make room for files that are to be transferred. The space on RT-11 devices that is allocated to the directory or files can become fragmented. If this is the case, an attempt to add another file to the device can result in a ?DIRECTORY OVERFLOW or ?DEVICE FULL error. You can use the /SQ switch to compress the contents of the device and provide enough room for the additional files. Note that if a squeeze operation is interrupted, data may be lost on the device being squeezed. Thus, FIT ignores any CTRL/C typed at the keyboard during a squeeze operation.

Note that because of the manner in which RT-11 directory entries are made, you may need to compress the device to use the directory space even if files are added contiguously without deletions.

#### NOTE

RT-11 format DECtape (TU56) is much slower to access than RSTS/E format DECtape and thus should be used only when necessary to transfer between RSTS/E and RT-11 systems. For example, a FIT squeeze operation (/SQ) on an RT-11 DECtape can take up to an hour or longer.

#### 16.5.3 DOS Disk Directory Listings

To list the directory of a DOS disk, enter the following command format in response to the FIT prompt:

```
FIT> dev:[proj,prog]filename.ext[/DOS] /L[I]
```

where dev: is the device designator of the DOS disk, [proj,prog] is the account number (UIC) of the directory to be listed, and filename.ext indicates the file(s) to be listed. Note that the project-programmer and filename.ext fields can contain wildcards. The switch /DI can be used in place of /L or /LI. If the /DOS switch is omitted, FIT lists the directory of the device whether it DOS, RSTS/E, or RT-11 directory structured. If the /DOS switch is present, an error is returned if the device is not a DOS disk.

The listing generated by this operation includes the following information: account numbers (UICs), filenames, extensions, file length, protection codes, dates of creation, the first block numbers of the files, and whether or not the file is contiguous.

# **Chapter 17**

## **Storing Files Off-Line: The Backup Program**

### **17.1 Preserving Files: The Backup Program**

The Backup system program allows you to preserve files by creating off-line copies on disk or magnetic tape. Backup, in its RESTORE mode, can return those off-line copies to the on-line disk structure. Backup communicates with you by means of a dialogue as described in Section 17.3. That is, the program asks you a series of questions that you answer one at a time. Your answers determine the program's mode, the device to which files are copied, and where Backup prints its directory and command listings, (the record and results of its work). Note that Backup can also be run in a batch stream as described in Section 17.2.4.

#### **NOTE**

Backup can not be used to preserve files that are larger than 65535 blocks. To preserve such files, have the system manager run the SAVE/RESTORE program as described in the *RSTS/E System Manager's Guide* or use PIP as described in Chapter 16. Backup can be used to preserve a placed file; however, the file's placement is not preserved. Also, file attributes are preserved when attributed files are copied.

The group of disks or magnetic tapes on which the files are preserved is called the Backup Set. A single member of this set, whether a disk or tape, is called a volume.

#### **17.1.1 BACKUP Mode**

In BACKUP mode, the Backup program selects the files to be processed, records the accounts to be transferred, and sorts the accounts into ascending numerical order. Information about accounts, files, and errors is written into the work file, which the Backup program uses for communication between the

different operations and to control the entire operation. The work file is a directory of the Backup Set and, after completion of the BACKUP operation, the program writes two copies of the file on the Backup Set's last volume. The completed work file is called the Backup Set index file. At the end of a successful BACKUP operation, three copies of the index file exist; two copies exist on the Backup Set (the primary and secondary index file) and a third remains in your account (the work file). Subsequent Backup operations must use a copy of the index file, because it is the only record of the Backup Set's content.

Backup, in its LOADINDEX mode, copies the Backup index file from the Backup Set to a specified disk file. That is, LOADINDEX copies the index file from a non-readable Backup Set volume to a readable disk file. This file is in RSTS/E format and can be used in multiple RESTORE operations as described in Section 17.3.2. Backup also provides a LIST mode, which is used to print the directory information contained in the index file. The generated listing describes the contents of the Backup Set (a directory of files) and notes any errors that may have occurred during the BACKUP operation.

In transferring files, Backup copies them to the backup medium. After it transfers all files, it updates the listing file with an entry for each account and file it transfers and for each error it encounters. Also, in the summary line for each account, Backup reports the number of files and blocks it has transferred and the number of errors it has encountered.

When a Backup volume is filled, Backup requests a new volume by printing a prompt on the job's keyboard as described in Section 17.7. After the files have been transferred, Backup prints a completion message and returns you to the current default run-time system.

#### NOTE

To backup files to a disk, a disk in Backup format is required. This format differs from the standard RSTS/E format. To change a disk from RSTS/E format to Backup format, a non-privileged user must have the system manager run the BACDSK program as described in the *RSTS/E System Manager's Guide*.

#### 17.1.2 RESTORE Mode

In RESTORE mode, Backup performs a file transfer which is the opposite of that described in Section 17.1.1. That is, RESTORE transfers files from the Backup Set to a RSTS/E format disk. As with the BACKUP mode, RESTORE mode obtains information from you by means of a dialogue.

In the RESTORE dialogue, you specify the files to be transferred. You can specify the entire Backup Set or a subset of it. You can also specify the desired files and accounts in the dialogue but exclude files from the transfer if they already exist on the destination disk.

During the transfer process, the Backup program in RESTORE mode copies account and file information from the specified index file to a temporary disk file. Upon completion of the process, RESTORE records the number of accounts, files, and blocks transferred along with accounting statistics. As with BACKUP mode, RESTORE mode records this information in the listing file.

Once the selection is complete, RESTORE causes the Backup program to copy the files from the Backup Set to the destination disk. Files and accounts are processed in order of appearance on the Backup Set volume. For each account entered and transferred, and for each file transferred, RESTORE generates an entry in the listing file. Errors that can destroy data are also logged in the listing file and are printed on the job's keyboard.

After all files and accounts are transferred and all pertinent information logged, the Backup program prints a completion message and returns you to the current default run-time system.

#### **NOTE**

RESTORE mode does not supersede files that are open for UPDATE. If RESTORE encounters an open file during the transfer, the program generates an error message, cancels the transfer of that file, and continues to the next sequential file.

#### **17.1.3 LOADINDEX Mode**

In LOADINDEX mode, the Backup program copies the primary index file from the Backup Set to a RSTS/E format disk. As with BACKUP and RESTORE modes, LOADINDEX obtains information from you by means of a dialogue.

In the LOADINDEX dialogue, you specify the location of the Backup Set and the file into which the program is to copy the index. The specified destination for the index copy must be a disk file. After you specify the destination disk, the Backup program prompts you to mount the Backup Set index volume.

LOADINDEX mode causes the Backup program to copy the index from the Backup Set to a RSTS/E format disk file. In addition, LOADINDEX mode produces a listing file that contains a summary of the dialogue and a report on any errors that may have occurred.

The creation of a separate file, which contain the Backup Set index, can be useful for subsequent RESTORE operations. RESTORE mode requires the index in order to perform its operations. If you have not created an index file with LOADINDEX or, alternatively, specified a work file during the BACKUP operation, you must physically mount the last volume of the Backup Set. For example, if multiple restores are to be performed on magnetic tape (where the index is at the end of the tape), LOADINDEX can retrieve the index and place it in your account for immediate use. The last volume of the set contains the primary index and, if no other index is available, must be mounted for each RESTORE operation.

#### **17.1.4 LIST Mode**

In LIST mode, the Backup program lists the contents of a Backup Set index directory. The directory information can be obtained from the Backup Set primary index file or, if it exists, the work file generated during the BACKUP operation (see Section 17.1.1). The LIST mode allows you to obtain a directory of the Backup Set contents as well as a list of any errors that occurred during the BACKUP operation which created the set.

LIST, as with other Backup program modes, initiates a dialogue with you. The dialogue prompts you to specify the filename and location where the listing is to be written, and the index filename and location. If you specify an on-line index file (either the work file created during a BACKUP operation or a disk file created during a LOADINDEX operation), LIST causes the Backup program to extract the directory information and write it to the specified listing file. If you specify the Backup Set primary index, the Backup program prompts you to mount the index volume before it extracts the directory information. Note that a summary of the LIST dialogue is also written to the listing file.

### **17.2 Running Backup**

The system manager can optionally install the Backup program in a non-standard system account. Before attempting to access Backup, see the system manager for the location of the program on your system.

To run the Backup program when it is contained in the system library account [1,2], type:

```
RUN $BACKUP
```

If access to Backup is successful, it prints an identification header followed by a mode prompt:

```
BACKUP V7.0      RSTS/E V7.0  
BAC[KUP], RES[TORE], LOA[DINDEX], OR LIS[T] ?
```

In response to the prompt, type the name of the Backup program mode you wish to use. The name you type can be a minimum 3-letter abbreviation as indicated by the square brackets. After you type the mode name, the dialogue for that mode begins.

Note that you can also run Backup under control of the Batch Processor as described in Section 17.2.4.

#### **17.2.1 File Specifications in Dialogue Answers**

At several points in the dialogue, Backup requests that you provide file specifications. You must include certain file characteristics in these specifications before Backup can process them. These characteristics can include a filename, account, and dates of creation and last access. You can also use these characteristics to specify exceptions from a Backup process.

A Backup file specification takes the form:

filename/keyword:comparison:date

where;

- |             |   |
|-------------|---|
| filename    | is a standard RSTS/E filename and may, optionally, contain the wildcards * and ?.   |
| /keyword    | is a slash separator followed by CREATION or ACCESS, both of which can be abbreviated to the first two letters.               |
| :comparison | is a colon separator followed by BEFORE or AFTER, both of which can be abbreviated to the first two letters.                  |
| :date       | is a colon separator followed by the desired date in dd-mmm-yy format. If you omit the year, Backup assumes the current year. |

Consider the following example:

\*.TEC/AC:BEF:07-MAY-78

In this example, Backup processes all files in the current account with a .TEC extension that were accessed before May 7, 1978 (i.e., May 6, 1978 or earlier).

The CREATION keyword allows you to specify the time of day in the Backup file specification. For example:

PHONE,\*/CR:AFT:09-JUL-78:21:13

This specification designates all files with the name PHONE that have been created since 21:13 (9:13pm)on July 9, 1978.

A Backup file specification can contain only one creation field and one access field. A slash must separate the fields from each other and from the filename (even if the filename is null). For example:

/CR:BEF:01-APR-78:17:30:/AC:AFT:22-SEP-78

This specification designates all files in the current account that were created before 17:30 (5:30 pm) on April 1, 1978 and accessed after September 22, 1978.

To exclude one or more files from the Backup operation, specify an EXCEPT field in the file specification. EXCEPT can be abbreviated to a 3-letter minimum and must be separated from other fields by a slash. EXCEPT has two forms:

1. EXCEPT:file specification.
2. EXCEPT:(list of file specifications).

You specify the first form to exclude one file specification from the Backup process. Note that the single exclusion specification must not contain a creation or access field. For example:

[100,250]EDIT,\*/CR:AFT:19-AUG-78/EXC:EDIT.TMP

This specification designates all files named EDIT in account [100,250] that were created after August 19, 1978 with the exception of EDIT.TMP.

To exclude more than one file from the Backup process or to include a creation or access field in the specification, use the second form of EXCEPT. For example:

```
*.*/CR:BEF:12-JUL-78/EXC:(*,RNO/CR:AFT:01-JAN-78,UG?,*/AC:AFT:01-JAN-78)
```

This specification designates all files in the default account that were created before July 12, 1978 with the following exceptions:

All files with a .RNO extension created after January 1, 1978.

All files with a 3-character name that begins with UG accessed after January 1, 1978.

An EXCEPT field that does not contain its own creation or access field within parentheses is affected by those fields already present in the specification.

Note that a specification can contain only one EXCEPT field.

### **17.2.2 Typographical Considerations**

The following conventions are used in Backup file specifications.

1. Continuing a specification line.

A lengthy file specification list can require several lines. To continue a Backup line, type a hyphen as the last character on the line prior to the line terminator. After you type a hyphen and line terminator, Backup prints CONT> and awaits further input. Backup will not process any responses until you terminate the line without a hyphen.

2. Spacing in specification lines.

Backup treats spaces as field terminators. That is, the line:

```
EXC EPT:BEE,BAS
```

is illegal because Backup ends the line scan after EXC. The line:

```
EXCEPT : BEE,BAS
```

is legal.

3. Including comments in a specification line.

To place a comment in Backup line, type an exclamation point and then the comment. Note that Backup ignores any characters that follow an exclamation point. Therefore, to continue a line that contains a comment, type the hyphen before the exclamation point. Also, if you specify account [1,3] as !, enclose the account specifier in quotation marks ("!" or `!'). Backup strips these quotation marks from the line when it is processed.

### **17.2.3 Backup Dialogue Rules**

If there is a default answer to a Backup dialogue prompt, it appears in brackets before the question mark. For example:

```
LISTING FILE <KB:> ?
```

To accept the default (in this case, your keyboard), type the RETURN key. If you type the RETURN key in response to a prompt that has no default, Backup prints a ?NO DEFAULT message and reprints the prompt.

To obtain an explanation of any Backup dialogue prompt, type /HELP.

If Backup encounters a syntax error, it prints the appropriate error message, a question mark, then repeats the prompt. If you respond with a question mark and the RETURN key, Backup reprints the erroneous line up to the point of error and then repeats the dialogue prompt.

### **17.2.4 Running Backup under BATCH**

Chapter 21 describes the procedures for creating and running a batch control file. Creation of such a file and its execution under BATCH allows you to run a program, such as Backup, without interaction at the terminal. All of the procedures for creation of a batch control file apply for running a Backup job under the Batch Processor with the addition of the following considerations.

1. The batch control file must include the RUN \$BACKUP command as described in Section 17.2.
2. The batch control file must contain responses to all of the appropriate dialogue prompts. These responses can be explicit or can be included in an indirect command file as described in Section 17.5. Note that if you use an indirect command file, the batch control file must contain the indirect command file specification as the response to the first Backup prompt.
3. The batch control file must contain the responses to Backup's device mounting prompts.

You must know the quantity and type of the devices required for the Backup operation prior to submitting the batch job. The commands for mounting each device must be included in the batch control file and each specified device must be mounted on a different drive. Thus, to run Backup under batch control, there must be as many appropriate devices available on the system as there are volumes in the Backup Set.

Also, all Backup dialogue mount prompts must be accounted for in the batch control file. That is, you must anticipate the appearance of MOUNT IT ANYWAY? prompts. These prompts appear during the Backup operation if a mounted volume's expiration date has not passed or if a mounted magnetic tape can not be read. You must include the appropriate response for these and all other prompts in the batch control file.

Consider the following:

```
$JOB/NOLIMIT/NOCPU !BEGIN BATCH
$RUN $BACKUP !RUN BACKUP
$DATA !BEGIN DIALOGUE
BACKUP !BACKUP DIALOGUE
B05SEP .J33 !WORK FILE
LPO: !LISTING FILE
DL2: !FROM DISK
RET !FROM FILES
MT: !TO DEVICE
RET !BEGIN AT
RET !DELETE FILES
RET !COMPARE FILES
BACKUP
RET
800
ODD
MM1:
YES
MM2:
YES
$EOJ !END BATCH
```

Note that this example includes default answers to some of the BACKUP dialogue questions, as indicated by the RETURN key response. Also note that no comments are allowed with the mount dialogue responses.

### 17.3 The Dialogue

After Backup is successfully accessed, it begins the dialogue with the following prompt:

```
BAC[KUP], RES[STORE], LOA[LDINDEX] or LIS[T] ?
```

You can respond with BAC, RES, LOA, or LIS to select the desired Backup mode, or you can type /HELP to print Backup's help text.

If you wish to create an indirect command file containing your response to the dialogue, type your choice of mode followed by the /SAVE option (minimally, /SA). For example:

```
BAC/SA
```

If you type the /SAVE option, Backup responds as follows:

```
INDIRECT FILE NAME <SY:[cur act]mode.CMD> ?
```

where [cur act] is the current account number and mode is BACKUP, RESTOR, LOADIN, or LIST. The defaults, in angle brackets, specify the command file on the system disk under the current account. If you do not accept the default but specify an indirect command file, Backup prints the rest of the responses into that file. The file you specify can be on disk or DECtape. However, the specified device must be available to you.

The rest of the dialogue depends on the mode you have chosen. The following sections describe the dialogues for BACKUP (Section 17.3.1), RESTORE (Section 17.3.2), LOADINDEX (Section 17.3.3), and LIST (Section 17.3.4).

### 17.3.1 BACKUP Dialogue

Table 17-1 summarizes the BACKUP dialogue initiated when you type BAC.

**Table 17-1: BACKUP Dialogue Summary**

BAC[KUP],RES[TORE], LOA[DINDEX], OR LIS[T] ? Response: BAC[KUP] (/SA[VE]) Effect: Initiates the BACKUP dialogue.
INDIRECT FILE NAME<SY:[cur act]BACKUP.CMD> ? Response: file specification. Effect: Asked only if the /SAV option is appended to the BAC response. Creates an indirect command file with the specified name. Valid output devices are disk and DECtape.
WORK FILE NAME<SY:[cur act]Bddmmm.Jnn> ? Response: file specification. Effect: Use the specified file as the work file. If the default is accepted, dddmm is the current day and month and Jnn is the current job number. Any disk can be substituted for SY:. This file can be used as an auxiliary index file.
LISTING FILE<KB:> ? Response: KBn: or LPn:. Effect: Output the listing file to the specified device. Response: file specification. Effect: Write the listing file to the specified file on disk or DECtape. The default file specification is [cur act]BACKUP.LST.
FROM DISK<SY:> ? Response: disk name. Effect: Backup from the specified disk. The disk must remain mounted throughout the BACKUP process.
FROM FILES<[cur act]*.*> ? Response: file specification(s). Effect: Backup the specified file(s).
TO DEVICE<MT:> ? Response: MT: or disk name. Effect: Backup to the specified medium.
BEGIN AT<*,*]*.*> ? Response: file specification. Effect: Backup starting with the specified file. The default is to begin with the first file matching the specification given in the FROM FILES prompt. Answer with a single file specification. No EXCEPT fields are allowed.
DELETE FILES<NONE> ? Response: file specification(s). Effect: Delete the specified file(s) after they are backed up.
COMPARE FILES<NONE> ? Response: file specification(s). Effect: Compare the specified files(s) to the originals after they are backed up.

In the following descriptions of the BACKUP dialogue, the questions are followed by explanation.

WORK FILE NAME<SY:[cur act]Bddmmm.Jnn> ?

This question asks you to specify a filename, extension, and location for the Backup work file. The work file is a record of the accounts, files, and errors processed by BACKUP that the program uses to create its primary index file. If you accept the default, the work file is placed in the current account on the system disk. The default filename is Bddmmm.Jnn, where dd is the current day and mmm is a 3-letter abbreviation of the month. If your system is configured for a numeric date format, the default filename is Byymmdd.Jnn (where y is the last digit of the year, mm is the number of the month, and dd is the day). In either case, Jnn indicates the current job number. To preserve the index file, use PIP to copy the work file into an auxiliary file (see Chapter 16) after the BACKUP operation is complete but before logging off. Note that an error is returned if you attempt to rename the work file to the filename of an existing file.

To directly preserve the work file and thus obtain a permanent copy of the Backup Set index, type a file specification in response to the prompt. The file must be on disk and should have a non-.TMP extension. Note that if you are creating a large Backup Set, it is recommended that you specify a private disk that can accommodate the length of the work file. The following formula can be used to estimate work file length:

.15\*(acc+files+att)

where acc is the number of accounts, files is the number of files, and att is the number of files that have attributes.

In order to conserve space, you may wish to prevent the transfer of the work file to the Backup Set. To prevent the transfer, include the work file in an EXCEPT field in response to the FROM FILES prompt.

LISTING FILE<KB:> ?

This question asks you to specify a location for the listing file. The listing file (BACKUP.LST) contains a record of the files and blocks transferred and errors encountered during the operation. If you accept the default, BACKUP.LST is output to your keyboard. To print the file on another keyboard or non-file structured device, respond to the prompt with a device specification. If you respond with KB0: for example, the file has the following specification:

KB0:[cur act]BACKUP.LST

You can also type a file specification as the destination of the file. The default specification applied to the response is as follows:

SY:[cur act]BACKUP.LST

If you type a file specification, the listing file is opened for output and immediately accessed to verify the availability of the file and device.

FROM DISK<SY:> ?

This question asks you to specify the input device: the device from which the files are to be transferred. The default is the public structure. You can specify only one device, which must be a disk.

FROM FILES<[cur act]\*,\*> ?

This question asks you to specify the file(s) you wish to backup from the input disk. If you accept the default, all files in the current account are backed up. If you specify files, type them in the format described in Section 17.2.1. Multiple specifications must be separated by commas. The default for each specification is all files in the current account. BACKUP transfers files in accordance with the following rules.

1. BACKUP does not transfer any file that is greater than 65535 disk blocks in length. The program issues an error message (see Section 17.8.4.1) and proceeds to the next file.
2. BACKUP transfers placed files but does not preserve their placement on the disk. If BACKUP encounters a placed file, it prints a warning message (see Section 17.8.4.1), transfers the file, and ignores the file's placement.
3. BACKUP can not transfer a file to which you do not have read access.
4. BACKUP transfers files in order of their appearance in the directory, not in the order they are specified.
5. BACKUP transfers accounts in ascending numeric order, regardless of their order in the Master File Directory.
6. If the input device is SY: and the public structure contains two or more disk devices, BACKUP transfers files from the same account on all public disks before it processes the next account. Also, only accounts that exist in the system disk Master File Directory are transferred.

TO DEVICE<MT:> ?

This question asks you to specify the target device type to which BACKUP will transfer the specified files. The default is magnetic tape. You can specify disk, but only one specification can be made and no unit number can be specified.

BEGIN AT <\*,\*,\*,\*,\*> ?

This question asks you to specify the first file to be transferred. If you accept the default, BACKUP transfers all files beginning with the first file matching your response to the FROM FILES question. If you specify a file, only one specification can be made and any files and accounts that appear before the specified file on the input device are not transferred.

You can use this feature to save typing time. That is, you can make a general response to the FROM FILES question or accept the default (all files in the current account), rather than specifying multiple files. Note that you can not include an EXCEPT field in answer to the BEGIN AT question.

DELETE FILES<NONE> ?

This question asks you to specify any file(s) that you wish to delete from the input disk after a copy is transferred to the Backup Set. If you accept the default, BACKUP preserves all files on the input disk. If you wish BACKUP to delete any files from the input disk, include their specifications (separated by commas) in the format described in Section 17.2.1. There is no default specification for this question. Thus, all fields, including account numbers, must be specified.

#### NOTE

Even if a file is specified in the DELETE FILES question, it is not deleted unless BACKUP transfers a copy of it to the Backup Set. Also, BACKUP does not delete files to which you do not have write access nor does it delete a file which causes an error during transfer or verification.

COMPARE FILES<NONE> ?

This question asks you to specify any backed up files which are to be checked against their originals on the input disk. If you accept the default, no comparison is made. If you wish BACKUP to compare any file, include its specification in response to this question in the format described in Section 17.2.1. Note that any comparison occurs prior to a file deletion. Note also that if an error occurs during the compare operation, the file is not deleted.

#### 17.3.2 RESTORE Dialogue

Table 17-2 summarizes the BACKUP dialogue initiated when you type RES.

**Table 17-2: RESTORE Dialogue Summary**

BAC[KUP],RES[TORE],LOA[DINDEX] OR LIS[T] ? Response: RES[TORE](/SA[VE]) Effect: Initiates the RESTORE mode dialogue.
INDIRECT FILE NAME<SY:[cur act]RESTOR.CMD> ? Response: file specification. Effect: Asked only if the /SAVE option is appended to the RES response. Creates an indirect command file with the specified name. Valid output devices are disk and DECtape.
WORK FILE NAME<SY:[cur act]Rddmmm.Jnn> ? Response: file specification. Effect: Use the specified file as the work file. If the default is accepted, dddmm is the current day and month and Jnn is the current job number. Any mounted disk can be substituted for SY:.

(continued on next page)

**Table 17-2: RESTORE Dialogue Summary (Cont.)**

LISTING FILE<KB:> ?
Response: KBn: or LPn:.
Effect: Output the listing file to the specified device.
Response: file specification.
Effect: Write the listing file to the specified file on disk or DECTape. The default file specification is [cur act]RESTOR.LST.
INDEX FILE<PRIMARY> ?
Response: file specification.
Effect: Use the specified file as the index. If you accept the default, Restore uses the primary index file located in the last volume of the Backup Set. You can specify the index file created in a LOADINDEX operation.
FROM DEVICE<MT:> ?
Response: MT: or disk name.
Effect: Restore the backed up files on the specified medium.
FROM FILES<[cur act]*.*> ?
Response: file specification.
Effect: Restore the specified file(s).
TO DISK<SY:[*.*]> ?
Response: disk name.
Effect: Restore the specified files to the specified disk.
BEGIN AT<*,*.*> ?
Response: file specification.
Effect: Restore files beginning with the specified file. If you accept the default, Restore begins with the first file that matches the specification given in answer to the FROM FILES question. You can specify only one file and no EXCEPT field is allowed.
SUPERSEDE FILES<NONE> ?
Response: file specification.
Effect: On the destination disk, Restore overwrites the specified files with the backed up versions.
COMPARE FILES<NONE> ?
Response: file specification.
Effect: Compare the specified restored file(s) to their backed up versions.

In the following description of the RESTORE dialogue, the questions are followed by explanations.

```
WORK FILE NAME<SY:[cur act]Rddmmm.Jnn> ?
```

This question asks you to specify the name and location of the backup work file. If you accept the default, RESTORE uses the file Rddmmm.Jnn (where ddmmm is the current day and month) under the current account on the system disk. If your system is configured for a numeric date format, the default filename is Rymmdd.Jnn (where y is the last digit of the year, mm is the number of the month, and dd is the day). In either case, Jnn indicates the current job number. You can specify any disk file specification.

```
LISTING FILE <KB:> ?
```

This question asks you where you wish Backup to print the listing file RESTOR.LST. If you accept the default, the listing file is output to your keyboard. To request printing on another available keyboard or non-file structured device, specify the device designator. Thus, if you specify LP0, the following listing is created:

```
LP0:[cur act]RESTOR.LST
```

You can also specify any file to which you have write access for the listing file. In this case, the default file specification is as follows:

```
SY:[cur act]RESTOR.LST
```

```
INDEX FILE<PRIMARY> ?
```

This question asks you to specify the index file, which Backup, in RESTORE mode, uses as the source of information about accounts and files selected for restoration. From this information, Backup writes the work file and, when the restoration is complete, copies the information into the listing file.

If you accept the default, Backup uses the Backup Set primary index file for the restoration.

You can specify an index file created with LOADINDEX mode (see Section 17.3.3) in response to this question. If you specify an index file other than the default, Backup loads its work file from the information contained in the specified file. You can specify a file on disk, magnetic tape, or DECTape. For example:

```
DT1:MYRES.IND
```

```
FROM DEVICE<MT:> ?
```

This question asks you from what device type the backed up files will be restored. If you accept the default, Backup assumes that the device is magnetic tape. You can specify a disk designation in response to this question.

```
FROM FILES<[cur act]*,*> ?
```

This question asks you to specify the backed up files to be restored to the destination disk. If you accept the default, all backed up files under the current account are restored. Note that when you restore a file that was originally placed, the file's placement is lost. To enter file specifications in response to this question, use the format described in Section 17.2.1. The default account for each file specification is the one that last appeared in the response. See the description of this question in Section 17.3.1 for the rules governing BACKUP and RESTORE file transfer.

```
TO DISK<SY:[*,*]> ?
```

This question asks you to specify the destination disk and account for the restored files. If you accept the default, Backup restores and transfers all files to the system disk under the account from which they were backed up. The default also applies to any specification you enter. Note that you cannot restore a file to an account to which you do not have access, even if the file was originally backed up from that account. That is, only a privileged user can restore files to another's account.

```
BEGIN AT<[*,*]*,*> ?
```

This question asks you to specify the first file to be restored. If you accept the default, the first file matching your response in the FROM FILES question is the first file restored. If you specify a file, only one specification can be made and no EXCEPT fields can be included.

```
SUPERSEDE FILES<NONE> ?
```

This question asks you to specify a file or set of files that you wish Backup to overwrite. RESTORE, during the transfer of the files you specify, causes them to replace their originals on the destination disk. If you accept the default for this question, no files are overwritten. That is, if a file already exists on the destination disk, RESTORE does not transfer its copy from the backup set.

To specify a file or set of files, follow the format described in Section 17.2.1. Note that Backup matches your specification against file information in the Backup Set, not in the destination disk directory.

#### **CAUTION**

When you respond to SUPERSEDE, ensure that the response does not unintentionally replace a recently created disk file with an older version from the Backup Set.

```
COMPARE FILES<NONE> ?
```

This question asks you to specify any restored files on the destination disk that you wish to compare against their copies on the Backup Set. If you accept the default, no comparison is made. If you wish comparisons, specify the desired file or set of files in the format described in Section 17.2.1.

### 17.3.3 LOADINDEX Dialogue

Table 17-3 summarizes the Backup dialogue initiated when you typed LOA.

**Table 17-3: LOADINDEX Dialogue Summary**

BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? Response:LOA[DINDEX] (/SA[VE]) Effect: Initiates the LOADINDEX dialogue.
INDIRECT FILE NAME<SY:[cur act]LOADIN.CMD> ? Response: file specification. Effect: Asked only if the /SAVE option is appended to the LOA response. Creates an indirect command file with the specified name. Valid output devices are disk and Dectape.
WORK FILE NAME<SY:[cur act]Lddmmm.Jnn> ? Response: file specification. Effect: Use the specified file as the work file. If you accept the default, dddmm is the current day and month and Jnn is the current job number. Any mounted disk can be substituted for SY:.
LISTING FILE<KB:> ? Response: KBn: or LPn:. Effect: Output the listing file to the specified device. Response: file specification. Effect: Write the listing file to the specified file on disk or DECtape. The default file specification is [cur act]LOADIN.LST.
FROM DEVICE<MT:> ? Response: MT: or disk name. Effect: Copy the index from the backup volume on the specified medium.
TO FILE<SY:[cur act]BINDnn.IND> ? Response: file specification. Effect: Loads the backup set index into the specified disk file under the current account. If you accept the default, the filename is BINDnn.IND, where nn is the backup job number.

In the following description of the LOADINDEX dialogue, the questions are followed by explanations.

WORK FILE NAME<SY:[cur act]Lddmmm.Jnn> ?

This question asks you to specify the name and location of the backup work file. If you accept the default, LOADINDEX uses the file Lddmmm.Jnn (where dddmm is the current day and month) under the current account on the system disk. If your system is configured for a numeric date format, the default filename is Lymmmdd.Jnn (where y is the last digit of the year, mm is the number of the month, and dd is the day). In either case, Jnn indicates the current job number.

**LISTING FILE<KB:> ?**

This question asks you where you wish Backup to print the listing file LOADIN.LST. If you accept the default, the listing file is output to your keyboard. To request printing to another available keyboard or non-file structured device, specify the device designator. You can also specify any file specification to which you have write access as the listing file.

**FROM DEVICE<MT:> ?**

This question asks you from what device type the backup index will be copied. If you accept the default, LOADINDEX assumes that the source volume is contained on magnetic tape. You can specify a disk designation in response to this question. Note that following completion of the dialogue, Backup prints a series of prompts that request you to mount the index volume.

**TO FILE<SY:[cur act]BINDnn.IND> ?**

This question asks you to specify the filename and location to which Backup is to copy the index file. If you accept the default, BACKUP copies the primary index file into the file BINDnn.IND (where nn is the backup job number) on the system disk under the current account. If you specify a file, it must be a disk file.

#### **17.3.4 LIST Dialogue**

Table 17-4 summarizes the BACKUP dialogue initiated when you type LIS.

**Table 17-4: LIST Dialogue Summary**

BACK[KUP],RES[TORE],LOA[DINDEX] OR LIS[T] ? Response: LIS[T] (/SA[VE]) Effect: Initiates the LIST dialogue.
INDIRECT FILE NAME<SY:[cur act]LIST.CMD> ? Response: file specification. Effect: Asked only if the /SAVE option is appended to the LIS response. Creates an indirect command file with the specified name. Valid output devices are disk or DECTape.
WORK FILE NAME<SY:[cur act]Lddmmm.Jnn> ? Response: file specification. Effect: Use the specified file as the work file. If you accept the default, dddmm is the current day and month and Jnn is the current job number. Any mounted disk can be substituted for SY:.
LISTING FILE<KB:> ? Response: KBn: or LPn:. Effect: Output the backup directory information to the specified device. Response: file specification. Effect: Write the backup directory information to the specified file on disk or DECTape. The default file specification is [cur act]LIST.LST.

(continued on next page)

**Table 17-4: LIST Dialogue Summary (Cont.)**

INDEX FILE<PRIMARY> ?
Response: file specification. Effect: Specify the name and location of the file from which the backup directory is to be extracted. If you accept the default, Backup extracts the directory from the Backup Set index volume.
FROM DEVICE<MT:> ?
Response: device specification. Effect: Asked only if the default is accepted for the INDEX FILE question. Specify the device type on which the primary index file is located.

In the following description of the LIST dialogue, the questions are followed by explanations.

WORK FILE NAME<SY:[cur act]Lddmmm.Jnn> ?

This question asks you to specify the name and location of the backup work file. If you accept the default, LIST uses the file Lddmmm.Jnn (where ddmmm is the current day and month) under the current account on the system disk. If your system is configured for a numeric date format, the default filename is Lymmdd.Jnn (where y is the last digit of the year, mm is the number of the month, and dd is the day). In either case, Jnn indicates the current job number.

LISTING FILE<KB:> ?

This question asks you to specify where Backup is to print the directory information that LIST causes it to extract from the index file. If you accept the default, Backup extracts the directory information from the Backup Set index and outputs it to your keyboard. If you specify a filename (in the format described in Section 17.2.1), LIST causes Backup to copy the directory to the specified file. The default file specification is LIST.LST under the current account on the system disk.

INDEX FILE<PRIMARY> ?

This question asks you the index file from which Backup will extract the directory information. If you accept the default, Backup uses the Backup Set index volume as the source of the directory information. Following your default answer, LIST prints the FROM DEVICE question. Backup prints a prompt requesting that the index volume be mounted on the specified device type before it can extract the directory.

An alternate response to the INDEX FILE question is the file specification of the index file produced by means of LOADINDEX. If you respond with a file specification, Backup proceeds to extract the directory from that file and no further LIST questions are asked.

FROM DEVICE<MT:> ?

If your response to the INDEX FILE question was the default, Backup prints this question to ask the device type on which the index volume is mounted. You can accept the magnetic tape default or specify a device designation. After you respond to this question, Backup prints a mounting dialogue which requests you to mount the Backup Set index volume.

## 17.4 Interruption Commands

Table 17-5 describes the interruption commands provided with the Backup program. Backup prints an asterisk at your terminal to indicate its readiness to accept an interruption command. You can type a command in response to the asterisk or to a mount procedure request (described in Section 17.7). Note that not all Backup operations support the full set of interruption commands. If your command is unacceptable, Backup prints an ?ILLEGAL COMMAND message. Also, interruption commands should not be included in indirect command files.

**Table 17-5: Interruption Commands**

ABO[RT]	Terminates processing and immediately returns you to the current default run-time system.
CONTINUE	Continues processing after a PAUSE command.
OFFLINE	Same as ABORT.
END	Terminates after the current file operation is complete.
LAST	Reprints the last printed text on the keyboard. If the program is awaiting response, LAST reprints the question or prompt. If not, LAST reprints the last message.
LEGAL	Prints the current legal set of commands and responses. When the message **OTHER** appears in the list, it means that the program's request for information is pending. Your response to such a request is the appropriate string. Quotation marks must delimit the response if the response is the same as an interruption command. For example, you can name the Backup Set PAUSE, but you must enclose the name in quotation marks.
NOT[ICE]	Writes the specified text into the listing file. The format of the NOTICE command is as follows:  NOTICE<message> where message is one line of text. No line terminators are allowed in the message.
PAU[SE]	Suspends execution until you type CONTINUE. If you type a legal command during PAUSE state, it is immediately executed.
STA[TUS]	Prints BACKUP status information on the keyboard.
TERMINATE	Allowed only on BACKUP; closes the current volume at the end of the current file or end of the current output volume, whichever comes first, and continues with the next volume.

## 17.5 Running Backup from an Indirect Command File

You can run Backup from an indirect command file that you create by specifying the /SAVE option in the dialogue (see Section 17.3). When Backup runs from an indirect command file, it does not print dialogue questions. Rather, Backup reads the responses you include in the command file. Backup does, however, print an asterisk to indicate its readiness to accept an interruption command. Backup also prints a completion message after the file transfer operation.

To run Backup from an existing command file, respond to Backup's first dialogue question with the mode specification followed by a commercial at sign @ and the command filename. For example if you used the /SAVE option in a previous RESTORE operation and accepted the default filename, you can specify that command file in another RESTORE:

```
BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? RES@RESTOR.CMD
```

However, if you did not accept the default but specified a command filename (MYRES.CMD, for example), your response would be as follows:

```
BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? RES@MYRES.CMD
```

Note that if the assignable account specifier is used, you must type two commercial at signs. For example:

```
BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LOA@LOADIN.CMD
```

## 17.6 Dialogue Examples

The following examples illustrate the dialogue and output of the BACKUP, RESTORE, and LOADINDEX modes of the Backup program.

The first example is a backup of files in account [11,110] on device DM1:

```
RUN $BACKUP  
BACKUP V7.0 RSTS V7.0 Timesharing  
  
BAC[KUP],RES[TORE], LOA[DINDEX] OR LIS[T] ? BAC  
WORK FILE NAME<SY:[11,110]BOBSEP.J32> ? BACKUP.WKF  
LISTING FILE<KB:> ? BACKUP.LST  
FROM DISK<SY:> ? DM1:  
FROM FILES<[11,110]*.*> ? [*,110]*.*/*EXC:[11,110]*.BAK  
TO DEVICE<MT:> ?  
BEGIN AT <[*,*]*.*> ?  
DELETE FILES<NONE> ?[11,110]*.DAT  
COMPARE FILES<NONE> ? [1,110]*.DAT  
*  
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
DEVICE? MMO:
EXPIRATION DATE HAS NOT YET ARRIVED!
          ID: BACKUP
          SEQ#: 1
          DENSITY: 800 BPI
          PARITY: ODD
EXPIRATION DATE: 08-Sep-79
MOUNT IT ANYWAY<NO>?Y
```

```
*
```

DATA UNRELIABLE - FILE OPENED BY ANOTHER USER IN FILE  
DM1 : [11,110]ADDR.DAT  
(ON TRANSFER)  
\*STATUS

```
PHASE      : TRANSFER
VOLUME #   : 1
ACCOUNTS   : 1
FILES      : 2
BLOCKS     : 2010
ERRORS     : 1
```

```
CURRENT VOLUME :MMO
CURRENT ACCOUNT :DM1    :[11,110]
CURRENT FILE    :DM1    :[11,110]RT11 .RTS<60>
```

```
ELAPSED TIME : 82 SECONDS
CPU TIME     : 1.8 SECONDS
KCTS         : 284
```

```
*
NO MORE ROOM ON VOLUME - TERMINATING,
*
DISMOUNT DEVICE: MMO:
          ID: BACKUP
          SEQ#: 1
          DENSITY: 800 BPI
          PARITY: ODD
EXPIRATION DATE: 08-Sep-79
          PLEASE LABEL THIS VOLUME!
```

```
MOUNT DEVICE: MM:
          ID: BACKUP
          SEQ#: 2
          DENSITY: 800 BPI
          PARITY: ODD
          IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
DEVICE? MM1:
*
```

```
DISMOUNT DEVICE: MM1:
          ID: BACKUP
          SEQ#: 2[INDEX]
          DENSITY: 800 BPI
          PARITY: ODD
EXPIRATION DATE: 08-Sep-79
          PLEASE LABEL THIS VOLUME!
```

```
*
```

Ready

The second example shows a restoration of two files from the account that was backed up in the previous example.

```
RUN $BACKUP          RSTS V7.0 Timesharing

BACKUP V7.0          RSTS V7.0 Timesharing

BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? RES
WORK FILE NAME<SY:[11,110]ROBSEP.J32> ? RESTOR,WKF
LISTING FILE<KB:> ? RESTOR,LST
INDEX FILE<PRIMARY> ?
FROM DEVICE<MT:> ?
FROM FILES<[11,110]*.*> ? [11,110]BACKUP.BAS,BACKTO.BAS
TO DISK<SY:[*,*]> ? DM1:
BEGIN AT<[*,*]*,*> ?
SUPERSEDE FILES<NONE> ?

COMPARE FILE<NONE> ?
*
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:             800BPI
PARITY:               ODD
PLEASE MOUNT VOLUME WRITE LOCKED!
DEVICE? MM1:
*STATUS

PHASE   :SELECT
VOLUME# : 2
ERROR   : 0

CURRENT VOLUME :MM    :
CURRENT ACCOUNT :DM1   :[11,110]

ELAPSED TIME   :1 SECONDS
CPU TIME       ::.4 SECONDS
KCTS           :64

*
DISMOUNT DEVICE:      MM1:
MOUNT   DEVICE:      MM:
ID:                 BACKUP
SEQ#:                1
DENSITY:             800 BPI
PARITY:               ODD
PLEASE MOUNT VOLUME WRITE LOCKED!
DEVICE? MMO:
*
DISMOUNT DEVICE:      MMO:
*
Ready
```

The final example illustrates the output of a LOADINDEX operation.

```
RUN $BACKUP  
BACKUP V7.0          RSTS V7.0 Timesharing  
  
BAC[KUP], RES[TORE], LOA[DINDEX] OR LIS[T] ? LOA  
WORK FILE NAME<SY:[11,110]L08SEP,J32> ? LOADIN.WKF  
LISTING FILE<KB:> ? LOADIN.LST  
FROM DEVICE<MT:> ?  
  
TO FILE<SY:[11,110]BIND08,IND> ? BACIND.IND  
*  
PLEASE ENTER BACKUP SET NAME<LOADIN> - BACKUP  
PLEASE ENTER DENSITY IN BPI<800> -  
PLEASE ENTER THE PARITY <ODD> -  
MOUNT DEVICE:           MT:  
                      ID:      BACKUP  
                      SEQ#:    INDEX  
                      DENSITY: 800 BPI  
                      PARITY:   ODD  
                      PLEASE MOUNT VOLUME WRITE LOCKED!  
DEVICE? MM1:  
*  
DISMOUNT DEVICE:       MM1:  
*  
Ready
```

## 17.7 Mounting and Dismounting Volumes

After Backup selects the files and accounts you wish transferred, it prints requests on your terminal for the label and device unit number of the first backup volume. It also prints a volume identification summary.

In response to the Backup label request, type the name and expiration date of the Backup Set. Backup uses the name as an identifier for the Backup Set. Backup uses the expiration date as the date after which it can automatically write over the data on the volume. If you mount a volume that bears your account number for backup before its expiration date, Backup asks for confirmation before writing on that volume. If the volume does not bear your account number, Backup does not allow you to write on that volume. If the volume is on magnetic tape, Backup issues a request for tape density (800 or 1600 BPI) and, for 800 BPI tape, parity.

After you answer the label request, mount the volume write-enabled for a BACKUP operation. Magnetic tape volumes used in RESTORE, LOADINDEX, and LIST operations must be mounted write-locked. However, disk volumes must be mounted write-enabled for all operations to allow updating of the bad block file.

Backup requests the device unit number with a DEVICE ? prompt. In reply, type the device designator on which the volume is physically mounted. Ensure that the volume is mounted and the device is ready before you respond to the DEVICE ? prompt.

### CAUTION

The Backup interruption commands and Backup requests RET[RY], SKI[P], and IGN[ORE] are executed during the mount procedure if the first three characters of the Backup Set name are the same as the first three characters of a command or request. The Backup Set name should not contain characters that would cause such a match. If you must specify such a Backup Set name, enclose the name in quotation marks.

The following is an example of the mount procedure printout:

```
PLEASE ENTER BACKUP SET NAME<B27JUN> -
PLEASE ENTER EXPIRATION DATE<01-JAN-79> -
PLEASE ENTER DENSITY IN BPI<800> - 800
PLEASE ENTER THE PARITY <ODD> -
MOUNT    DEVICE:          MT:
           ID:          B27JUN
           SEQ:         1
           DENSITY:     800 BPI
           PARITY:      ODD
EXPIRATION DATE:        01-JAN-79
IDENTIFICATION WILL BE FINAL UPON SUCCESSFUL MOUNT
DEVICE? MM1:
THIS VOLUME HAS NO BACKUP LABEL!
MOUNT IT ANYWAY<NO> ? Y
```

In this example, the message:

```
MOUNT IT ANYWAY<NO> ?
```

is issued because the mounted volume had no Backup label. Similar messages appear when the volume is unreadable or if the expiration date has not expired. The message is an aid to ensure that you have mounted the correct tape.

Backup also prints a message if it must change tape density or parity in order to read the tape. The message has the following form:

```
DENSITY CHANGED TO {800 }BPI, PARITY CHANGED TO {ODD }<text>
{1600 } {EVEN}
```

where <text> describes the operation for which the change has effect. In BACKUP, the change is made to read the volume label and the transfer proceeds as specified. In RESTORE, LOADINDEX, and LIST modes, the change is made for transfers from the current volume.

When use of a volume is complete, Backup prints a dismount message. For example:

```
DISMOUNT DEVICE: MM1:  
ID: B27JUN  
SEQ: 1  
DENSITY: 800 BPI  
PARITY: ODD  
PLEASE LABEL THIS VOLUME!
```

When this message appears, physically dismount the volume and ready the next volume for processing. Backup requests the name, expiration date, parity, and density of only the first Backup Set volume. For subsequent volumes, specify only the device unit number.

## 17.8 Backup Error Handling

The Backup program can encounter four types of errors; dialogue command errors, interruption command errors, volume mount errors, and Backup processing errors.

Dialogue command errors, described in Section 17.8.1, occur during BACKUP, RESTORE, LOADINDEX, and LIST dialogues. The Backup program diagnoses these errors as they occur and you can usually respond by typing the correct dialogue answer.

Interruption command errors, described in Section 17.8.2, can occur when you type an interruption command. Backup diagnoses these errors as they occur and you can correct them by typing a valid command.

Volume mount errors, described in Section 17.8.3, occur when the Backup program mounts tape and disk Backup Set volumes. These errors can be user errors (specifying an illegal tape density) or hardware errors (tape errors that prevent labeling).

Backup program processing errors, described in Section 17.8.4, can occur at any time except during a dialogue. Processing errors can be caused by the hardware on which Backup is running or can be caused by logic errors.

### 17.8.1 Dialogue Command Errors

Backup prints an error message on your terminal when it encounters a dialogue command error. The message, ?COMMAND ERROR, is followed by an error message indicating the nature of the error. After printing the message, Backup prints a question mark and repeats the dialogue question. If you respond with a question mark and RETURN key, Backup prints the command line up to the point of error then reprints the question. Table 17-6 describes the Backup dialogue error messages. Appendix C describes the RSTS/E error messages.

**Table 17-6: Backup Dialogue Error Messages**

<b>Message and Meaning</b>
<b>?BAD DIRECTORY FOR DEVICE</b> The directory structure on the specified device is corrupt. Place the file on a different device.
<b>?CAN'T FIND FILE OR ACCOUNT</b> The specified file or account can not be found. Ensure that the specification is correct and that the file and account exist.
<b>?DEVICE HUNG OR WRITE-LOCKED</b> The specified device is write-locked or generated a read/write error. Ready or write/enable the device or specify a different device.
<b>?DEVICE NOT AVAILABLE</b> The specified device is disabled or assigned to another user. Specify a different device.
<b>?DUPLICATE SWITCH</b> The file specification contains multiple ACCESS or CREATION fields. Backup accepts only one of each field type in a file specification.
<b>?ILLEGAL EXCEPT NESTING</b> The file specification contains more than one EXCEPT field. Backup accepts only one EXCEPT field in a file specification.
<b>?ILLEGAL FIELD</b> Your response contains a field that is not permitted. For example, a device name in response to the FROM FILES question generates this error.
<b>?ILLEGALFILENAME</b> The specified filename contains invalid characters or is in incorrect format.
<b>?ILLEGAL KEYWORD</b> Your response contains a misspelled or incorrectly abbreviated keyword, or has incorrect punctuation.
<b>?ILLEGAL OPERAND</b> The EXCEPT field in the file specification contains unmatched parentheses or a date is out of range.
<b>?INCOMPLETE COMMAND FILE</b> You typed a CTRL/Z or the indirect command file ended before the last dialogue question was answered. Backup returns to the beginning of the dialogue.
<b>?NO DEFAULT</b> The current dialogue question does not have a default answer. Type an explicit response.

(continued on next page)

**Table 17-6: Backup Dialogue Error Messages (Cont.)**

Message and Meaning
<b>?NOT A VALID DEVICE</b> The specified device is not present on the system.
<b>?NOT A VALID DEVICE (PROHIBITED)</b> The specified device is illegal in response to the current question.
<b>?PROTECTION VIOLATION</b> The specified file or account is protected against you.
<b>?TOO MANY FILES OPEN ON UNIT</b> Only one file can be currently open on a magnetic tape unit. Specify the second file on another device.
<b>?TOO MUCH DATA</b> Your response contains more data than Backup accepts for the current question.
<b>?UNIT NUMBER NOT VALID</b> Backup does not accept a device unit in response to the current question.

### **17.8.2 Interruption Command Errors**

An invalid interruption command can generate one of two error messages. Table 17-7 summarizes these errors and their meanings.

**Table 17-7: Interruption Command Error Messages**

Message and Meaning
<b>?UNRECOGNIZED COMMAND</b> You typed a string that is not an interruption command.
<b>?ILLEGAL COMMAND</b> You typed an illegal interruption command (for example, CONT when no PAUSE is in effect). To obtain a list of legal interruption commands, type LEGAL.

### **17.8.3 Volume Mount Errors**

Table 17-8 summarizes the errors that can occur during the volume mount process. When such an error occurs, Backup prints the error message and repeats the current prompt.

**Table 17-8: Volume Mount Error Messages**

Message and Meaning
<b>?CAN'T WRITE ON THIS TAPE</b> Backup can not write a label on the magnetic tape because of tape errors. Use another tape.
<b>?DENSITY/PARITY CANNOT BE SET ON THIS DRIVE</b> The specified parity or density is illegal on the current hardware.
<b>?DISK MUST BE INITIALIZED</b> A disk is mounted but is not formatted in the Backup or RSTS/E file structure. Have the system manager initialize the disk (to establish the bad block file) with DSKINT. The disk must not be logically mounted or opened.
<b>?INVALID BACKUP SET NAME</b> The Backup Set name contains illegal characters.
<b>?INVALID DATE</b> The expiration date is in incorrect format or has already passed.
<b>?INVALID DENSITY SETTING</b> Density setting must be either 800 or 1600 BPI.
<b>?INVALID DEVICE OR NO UNIT NUMBER</b> The specified input device is not present on the system or does not contain a unit number specification.
<b>?INVALID PARITY SETTING</b> Parity setting must be either odd or even.
<b>?MAGTAPE NOT WRITE-LOCKED</b> RESTORE, LOADINDEX, and LIST operations require a mounted, write-locked magnetic tape.
<b>?MAGTAPE SELECT ERROR</b> The specified magnetic tape unit is not ready or is off-line. Ready the unit or specify another.
<b>?MAGTAPE WRITE-LOCKED</b> BACKUP operations require a mounted, write-enabled magnetic tape.
<b>?THIS IS NOT THE RIGHT VOLUME</b> The volume contains unexpected data or you are not privileged or the owner of the volume.
<b>?YOU ARE NOT PRIVILEGED TO USE THIS VOLUME</b> The volume's expiration date has not passed or you are not privileged or the owner of the volume.

#### 17.8.4 Backup Processing Errors

Backup processing errors can occur as the program selects, transfers, compares, and deletes files and generates a listing file. A processing error can be a hardware error, which indicates a hardware problem such as ?DEVICE HUNG OR WRITE-LOCKED, or it can be a logic error, which indicates that you attempted an illegal or illogical action such as ?PROTECTION VIOLATION. The Backup program contains error-handling routines for common hardware and logic errors.

**17.8.4.1 Selection Errors** — If a logic error occurs during the selection process, Backup automatically skips over the file or account that causes 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 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.

Table 17-9 describes three of the errors that can occur during the selection process.

**Table 17-9: Backup Selection Error Messages**

Message and Meaning
<b>%LARGE FILE &lt;filespec&gt; - WILL NOT BE BACKED UP</b> You specified a file that is larger than 65535 disk blocks in responds to the FROM FILES prompt. The question is ignored and Backup continues with the next file.
<b>%PLACED FILE &lt;filespec&gt; SELECTED - PLACEMENT HAS BEEN LOST</b> This message appears when you include a placed file in a RESTORE operation. The file is restored, but its placement is not preserved.
<b>%PLACED FILE &lt;filespec&gt; SELECTED - PLACEMENT WILL BE LOST UPON RESTORATION</b> This message appears when you include a placed file in a BACKUP operation. The file is backed up, but its placement is not preserved.

Backup contains error-handling routines for the ?DEVICE HUNG OR WRITE-LOCKED hardware error. These routines allow you to request that Backup retry the procedure that encountered the error. Because the DEVICE HUNG error occasionally indicates transient hardware problems, you can sometimes recover from the error without missing any files or accounts.

You can also request that Backup skip over the error (type SKIP in answer to the prompt that follows the error). If you request a skip over a DEVICE HUNG error, Backup ends the search for the current file or account and proceeds to look for the next file or account, as it does for logic errors.

If a bad block error (?USER DATA ERROR ON DEVICE) occurs during a BACKUP mode selection phase, Backup prints the error message, stops the search, and proceeds to the next file or account.

During a RESTORE operation, a bad block error at index load time can mean that the index is corrupt. RESTORE asks you to mount a different volume (if necessary) so that it can read from another index file. If an error occurs in the secondary index file, the operation is aborted. You must then rerun BACKUP and use the work file or the file created by means of LOADINDEX.

A bad block error that occurs in a work file is fatal. Backup prints the message ?UNEXPECTED ERROR - USER DATA ERROR ON DEVICE, and aborts the job.

**17.8.4.2 Transfer, Deletion, and Listing Errors** — Errors during a transfer, deletion, or list operation usually result from an open failure on the file that Backup is attempting to transfer or delete. For example, ?PROTECTION VIOLATION means that the file's protection code does not permit you to transfer or delete it. The error, ?SUPERSEDE FAILURE, indicates failure to find or replace the specified file. Backup automatically skips over files or accounts that cause logic errors in transfer, deletion, or listing operations.

Backup error-handling routines allow you to retry or skip over ?DEVICE HUNG errors in transfer, deletion, or list operations as described in Section 17.8.4.1

The ?SEQUENCING PROBLEM ON MAGTAPE error can occur during a RESTORE operation and indicates that RESTORE is unable to read the record numbers stored on the tape during the BACKUP operation. When the sequencing error occurs, RESTORE usually prints several bad block error messages (as it tries and fails to read records), then recovers automatically. You can not restore the file in which the error occurred, but you can restore the remainder of the files on the magnetic tape.

If Backup encounters a bad block during a transfer operation that involves a RSTS/E disk, Backup prints an error message, then copies the rest of the file. Note that the copied file can be corrupt. If a bad block is encountered in the work file, the operation aborts.

**17.8.4.3 Informational Messages** — During the transfer process, Backup often prints messages that inform you about open files, which can be subject to change, and files whose length has changed since selection. These are not errors and Backup transfers the files. However, the copied files can be inaccurate. Backup also prints an informational message when it encounters a placed file. The file is transferred, but you are warned that the file's placement is lost. Other informational messages list files that Backup can not transfer because they were deleted after selection or because their length is zero.

Note that RESTORE never supersedes a file if the BACKUP copy of that file is questionable (that is, errors were detected during the BACKUP operation). To restore such a file, delete the file of the same name on the destination disk.



## **Chapter 18**

### **Diskette Transfer: The FLINT Program**

The system program FLINT (FLexible diskette INTerchange) allows you to transfer information from IBM\* diskettes (flexible diskettes) to a standard disk that RSTS/E can read (for example, DK:, DP:, SY:). Conversely, FLINT also allows the transfer of information from RSTS/E-readable disks to the diskettes that an IBM system can read. Either of these two transfers may involve multiple diskettes, referred to as volumes in the dialogue and in its description here. FLINT can transfer IBM diskette information to a RSTS/E disk only. For information on the transfer of files between RSTS/E disks, diskettes, and RT-11 format diskettes, see Section 16.5.

#### **NOTE**

FLINT can only process single density diskettes. FLINT cannot process double density diskettes on an RX02 drive.

FLINT also allows you to obtain a directory of an IBM diskette; this directory contains information related to the unique data format of an IBM diskette.

To perform these transfers and to print the directory of an IBM diskette, FLINT communicates with you by dialogue. The program asks you a series of questions, which you answer one at a time. There are two different dialogues for the transfer operations, IBM-to-RSTS/E and RSTS/E-to-IBM. There are also dialogues for the printing of an IBM directory and zeroing and erasing a diskette.

---

\*IBM is a trademark of International Business Machines Inc.

## 18.1 Running FLINT: The Initiation Commands

To run the FLINT program, you type:

```
RUN $FLINT
```

FLINT identifies itself and prints a number sign (#) to show its readiness to accept one of five initiation commands. Each of these initiates a different dialogue, as follows:

**Command      initiates the dialogue for:**

/DIRECTORY	listing an IBM diskette directory.
/TORSTS	transferring from an IBM diskette to a RSTS/E disk.
/TOIBM	transferring from a RSTS/E disk to an IBM diskette.
/ZERO	initializing the directory of a diskette.
/ERASE	initializing the directory and erasing all data on a diskette.

Each command may be abbreviated, at the minimum, to its first three letters: /DIR, /TOR, /TOI, /ZER, and /ERA. You can also run FLINT by the CCL command FLI[NT] if the FLINT command was installed on your system at startup; see Section 18.8.

## 18.2 Listing the Directory of an IBM Diskette

Typing /DIR in response to FLINT's number sign (#) prompt will cause FLINT to print a dialogue of two questions. In the following description of the dialogue, FLINT's questions are followed by explanations.

**OUTPUT TO?**

This question asks you where you wish FLINT to print the directory of the diskette(s). If you wish it printed at your terminal, you give a null response (i.e., press RETURN). If, however, you wish to save the directory in a file, respond with an output file specification of the form:

**dev:[proj,prog]name.ext<prot>**

You must specify at least the device and filename. By default, [proj,prog] is the current account and <prot> is <60>. Note that wildcard characters (?) or (\*) cannot be used to designate project-programmer numbers.

**DIRECTORY OF?**

This question asks you to specify the diskette(s) for which the directory will be printed. A null response specifies the diskette DX0:. Other disk specifications must be typed in the form:

**DXn: or n**

where n = 0 to 7

Multiple diskette specifications must be separated by commas.

### **18.2.1 The Form of the Directory**

The following example is a listing of an IBM diskette directory obtained by FLINT:

```
DIRECTORY OF DX1:  
DSN BRL BOE EOE EOD BI MVI VSN  
DATA 080 01001 73026 01001  
TOTAL OF 1 DATA SET ON DX1:
```

The abbreviated headings on the second line are listed and defined here, reading from left to right:

1. DSN Data Set Name: the 1-8 character name you have given the data set (corresponds to filename in RSTS/E).
2. BRL Block/Record Length: in each 128-position sector, the number of positions containing data.
3. BOE Beginning Of Extent: the address of the first sector in the data set; output is in the form tt0ss, where tt is track number and ss is sector number.
4. EOE End Of Extent: the address of the last sector reserved for this data set (in the same format as BOE above).
5. EOD End Of Data: the address of the next unused sector within the data set extent.
6. BI Bypass Indicator: A blank in this field means the data set is intended for processing; a B means it is not.
7. MVI Multi-volume Indicator: A blank in this field means a data set is wholly contained on this diskette; a C means that a data set is Continued on another diskette; an L means that this diskette is the Last on which a continued data set resides.
8. VSN Volume Sequence Number: the sequence of volumes in a multi-volume data set. Blanks indicate that volume sequence checking is not to be performed.

### **18.3 Transferring IBM Diskette Data to RSTS/E**

Type /TOR in response to FLINT's number sign (#) prompt to cause FLINT to print the dialogue for transferring diskette data to a standard RSTS/E disk (DK0:, for example). In the following description of that dialogue, FLINT's questions are followed by explanations.

OUTPUT TO?

This question asks you to specify the RSTS/E file that is to receive data from the diskette(s). Respond with an output file specification of the form:

dev:[proj,prog]name.ext<prot>/[NH]

By default, dev: is the system disk, [proj,prog] is the current account, and <prot> is <60>. Note that the device must be a disk (DP1:, for example) and that wildcard characters (?) or (\*) cannot be used to designate project-programmer numbers. The optional switch /NH suppresses the printing of the 6-byte logical header (see Section 18.3.2).

TRANSLATE FROM EBCDIC TO ASCII <YES>?

This question asks you if you wish the diskette data to be translated from its current EBCDIC mode, unreadable to RSTS/E, into ASCII mode, readable to RSTS/E. As the default -- enclosed in angle brackets -- indicates, a null response (i.e., pressing RETURN) will cause the data to be transferred "in translation". If, however, you type NO, FLINT will perform an "image mode" (byte-for-byte) transferral.

INPUT FROM?

This question asks you to specify the diskette(s) from which data will be transferred to the output file specified in answer to the OUTPUT TO question. You can specify such an input diskette by typing a response of the form DXn: or simply n, where n is a device number from 0 to 7. Such a response will cause FLINT to transfer the first data set on the specified diskette. If, however, you give a null response (i.e., press RETURN), FLINT transfers the first data set from DX0:.

If you wish FLINT to transfer a specific data set, give a response of the form DXn:DSN where DSN is the data set name, composed of one to eight ASCII characters, including blanks (spaces and tabs). Since these blanks are recognized as part of the IBM data set name, care should be taken in their use. If FLINT cannot find the specified data set, it prints the message ?FILE NOT FOUND and repeats the INPUT FROM question.

If you do not specify a data set name, FLINT prints a message of the form:

DSN IS THE DATA SET BEING TRANSFERRED

where DSN is the name of the first data set on the first diskette specified.

FLINT examines the data set label--a header with statistical information--to determine if the data set resides on more than one diskette. If it does, and if you have specified only one diskette, FLINT prints the message/question:

DSN RESIDES ON MORE THAN ONE DISKETTE -  
DO YOU WISH TO CONTINUE <NO>?

If you respond by typing YES, FLINT will proceed to its next question. As the default [NO] indicates, however, a null response causes no transfer, and instead causes FLINT to print its # prompt.

FLINT next computes the blocking factor to be used in the transfer of the current data set, and prints a message expressing that factor as the ratio of IBM to RSTS/E records. This message has the form:

```
N XXX CHARACTER IBM RECORDS = 1 RSTS RECORD
```

FLINT, after printing this message, proceeds with the transfer, extracting data from each specified input device.

FLINT computes this blocking factor by determining how many of the 128 positions in each diskette sector contain data in the form of characters. This count is the IBM record size, and is used to divide the RSTS record size (512). The product of this division is printed as N in the above message. For example, if each IBM record contains 80 data characters, FLINT will find that 6 full IBM records can be placed in a 512-byte "RSTS record" ( $512/80 = 6.4$ ). Thus, "6 80 CHARACTER IBM RECORDS = 1 RSTS RECORD".

#### NOTE

The first logical record on the RSTS/E output file is reserved for a statistical header of the form described in Section 18.3.2. This header is 6 bytes long. Thus, if the logical record length of the dataset to be transferred is less than 6 bytes, FLINT will use as many logical records as are necessary to contain the header. If, for example, the data set's logical records are only 4 bytes long, FLINT will have to allocate 2 of those records to hold the header. However, if you specify the /NH switch in response to the OUTPUT TO? dialogue question, the statistical header is not printed and the 6 bytes are freed for transferred data.

If the entire data set is not contained on the input device(s) specified, FLINT prints the question:

```
NEXT INPUT DEVICE?
```

Respond with the number of the drive on which the next portion of the data set resides. FLINT continues to perform transfers and to print NEXT INPUT DEVICE requests until it determines that the current diskette contains the end of the specified data set.

Note that on multi-volume input FLINT continually checks data set names against the one you specified, and, if possible, also checks volume numbers for correct sequence. If, for example, data set IDAT is being transferred and the third input volume specified contains no IDAT, FLINT prints the message:

```
FILE NOT FOUND - DXn:IDAT
```

and repeats the message NEXT INPUT DEVICE.

Moreover, if the volumes contain sequence numbers, and the number on a particular diskette is not 1 greater than that on the previous diskette, FLINT prints the message:

```
VOLUME #[m] CANNOT FOLLOW VOLUME #[n]
```

and repeats the message NEXT INPUT DEVICE.

If no fatal errors occur, FLINT, on completing the IBM-to-RSTS/E transfer, prints the message:

```
EXCHANGE COMPLETED
```

at your terminal, and lists both the number of IBM records read and the number of RSTS records written.

### 18.3.1 Specifying the Known Diskettes of a Data Set

Before you initiate the transfer of a multiple-volume data set, you may know on which diskettes the data set resides. In that case, respond to the INPUT FROM question by specifying the diskette on which the data set begins, then the data set's name, and then the other diskette(s) onto which the data set is continued. For example:

```
DX1:DATASETA,DX2:,DX3:,4
```

Note that this example, in accordance with good data procedure, causes the diskettes containing DATASETA to be mounted in order on consecutively numbered drives. You can, on the other hand, mount and specify diskettes out of sequence, but there is a chance for confusion. Note also that the unsequenced order of specifications must match exactly the unsequenced order of mounting. If it does not, FLINT will print the message VOLUME #[m] CANNOT FOLLOW VOLUME #[n], which is described in Section 18.3.

### 18.3.2 Format of the RSTS/E Disk File

FLINT writes, on the RSTS/E output file, a header record. This header contains the following information (all fields are 2-byte integer fields in standard CVT%\$ format; see the *BASIC-PLUS Language Manual*):

Byte	Contents
1-2	Physical block number of the last logical block in the file.
3-4	Number of logical records in the last physical block.
5-6	Logical record length in bytes.
7-	Remainder of the logical record contains no data (the entire logical record is reserved for the header).

If the logical record length of the data set is less than 6 bytes, this header will occupy more than one RSTS/E logical record. Note that the /NH switch can be used to suppress this header (see Section 18.3).

## 18.4 Transferring RSTS/E Files to IBM Diskettes

Type /TOI in response to FLINT's number sign (#) prompt to cause FLINT to print the dialogue for transferring a RSTS/E file to IBM diskette(s). In the following description of that dialogue, FLINT's questions are followed by explanations.

OUTPUT TO?

This question asks you to specify the IBM data set to which FLINT will output the RSTS/E data. Respond with a data set name in the form:

DXn:DSN

DXn: specifies a diskette on which the data set is written; n is a device number from 0 to 7. DSN is the output data set name, composed of 1 to 8 characters, including blanks. If you omit the DSN, FLINT assumes the name RSTS.

Multiple volumes must be separated by commas; for example:

DX0:DATSETC,DX2:

TRANSLATE FROM ASCII TO EBCDIC <YES>?

This question asks you if you wish the RSTS input data to be translated from its current ASCII mode into EBCDIC mode. As the default <YES> indicates, a null response causes FLINT to transfer the data "in translation." If, however, you type NO, FLINT performs an "image mode" (byte-for-byte) transfer.

INPUT FROM?

This question asks you to specify the RSTS/E file from which data will be transferred to the data set specified in answer to the OUTPUT TO question. Specify an input filename in RSTS/E format:

dev:[proj,prog]name.ext<prot>

By default, dev: is the system disk, [proj,prog] is the current account, and <prot> is <60>. Note that the device must be a RSTS/E disk and that wild-card characters (?) or (\*) cannot be used to designate project-programmer numbers.

RECORD LENGTH <128>?

This question asks you to specify the number of characters to be included in each IBM output record. As the default indicates, a null response causes the output records to contain 128 characters each. (IBM records contain 1 to 128 characters and are fixed length.) To specify a shorter record length, respond by typing any number smaller than 128 and greater than 0.

On receiving this response, FLINT computes the blocking factor to be used in the transfer of RSTS/E data, and prints a message expressing that factor as the ratio of IBM to RSTS/E records. This message has the form:

```
1 RSTS RECORD = N IBM RECORDS
```

For an explanation of how FLINT computes the blocking factor, see "INPUT FROM?" in the IBM-to-RSTS/E dialogue (Section 18.3).

After computing the blocking factor, FLINT proceeds with the transfer. If FLINT determines that the RSTS/E file cannot fit on the number of diskettes specified in answer to the OUTPUT TO question, it prints the question:

```
VOLUME *(n + 1)?
```

This question asks you to specify an additional diskette for output; respond with DXn: or with n, where n in either case is a device number from 0 to 7. Or, give a null response, which is equivalent to DX0:. Until FLINT can complete the transfer, it will continue to ask the VOLUME # question and to accept one of these responses. All diskettes used in the transfer will be labeled with volume sequence numbers, and, where necessary, with continuation markers.

#### NOTE

No more than 99 diskettes can be used for one data set. If you attempt to use more, FLINT will print the message:

```
MAXIMUM NUMBER OF VOLUMES EXCEEDED
```

and will issue its # prompt.

If no fatal errors occur, FLINT, on completing the RSTS/E-to-IBM transfer, prints at your terminal a message of the form:

```
EXCHANGE COMPLETE  
data set name RESIDES ON x DISKETTES
```

where x is the number of diskettes on which the named data set resides.

## 18.5 Initializing and Erasing a Diskette

Type /ZERO in response to the FLINT number sign prompt (#) to initiate the dialogue for initializing (zeroing) the directory of a single density diskette. In the following description of that dialogue, the FLINT questions are followed by explanations.

```
ZERO WHICH DISKETTE?
```

This question asks you to specify the diskette drive of the disk containing the directory you wish to zero. The drive specification takes the following form:

DXn: or n

where n is the unit number in the range of 0 to 7.

REALLY ZERO DXn: <NO>?

This question is asked to ensure that you have specified the correct diskette, where DXn: indicates the directory to be initialized. If you answer yes (Y) to this question, FLINT zeroes the directory of the diskette. If you accept the default (NO), FLINT prints a message and reprompts with a number sign.

Type /ERASE in response to the FLINT number sign prompt to initiate the dialogue for erasing a single density diskette. In the following description of that dialogue, the FLINT questions are followed by explanations.

ERASE WHICH DISKETTE?

This question asks you to specify the diskette drive of the disk you wish to erase. The drive specification takes the following form:

DXn: or n

where n is the unit number in the range of 0 to 7.

REALLY ERASE DXn: <NO>?

This question is asked to ensure that you have specified the correct diskette, where DXn: indicates the diskette to be erased. If you answer yes (Y) to this question, FLINT initializes the directory of the diskette and overwrites all data with zeros. Note that this operation can take some time. If you accept the default (NO), FLINT prints a message and reprompts with a number sign.

## 18.6 Dialogue Examples: /DIRECTORY, /TORS, /TOIBM, /ZERO, and /ERASE

In the first example, the directory of a diskette is requested and data from that diskette is transferred to a RSTS/E file.

```
FLINT V7.0 RSTS V7.0 TIMESHARING
#/DIREC
OUTPUT TO? RET
DIRECTORY OF? DX0:

DIRECTORY OF DX0:
DSN     BRL    BOE    EOE    EOD    BI    MVI   VSN
XFILEXMA    128    01001   30011    30012
WUMPUSDB    080    30012   43019    43020
PHONE DB    128    43020   47003    47004

TOTAL OF 3 DATA SETS ON DX0:

#/TORS
OUTPUT TO? PHONE.DAT
TRANSLATE FROM EBCDIC TO ASCII <YES>? RET
INPUT FROM? DX0:PHONE DB
```

```
DX0:  
    4 128 CHARACTER IBM RECORDS = 1 RSTS RECORD  
  
EXCHANGE COMPLETED  
    88 IBM RECORDS READ  
    89 LOGICAL ( 23 PHYSICAL ) RSTS RECORDS WRITTEN
```

In the /DIRECTORY dialogue, the directory of DX0: is printed at the terminal.

In the /TORSTS dialogue, data set PHONE DB is transferred from diskette DX0: to the RSTS/E output file PHONE.DAT. By default, the data is translated into ASCII on being transferred. FLINT computes and prints the blocking factor (4 IBM records to 1 RSTS/E record).

In the next example, a RSTS/E file is transferred to an IBM diskette.

```
FLINT V7.0 RSTS V7.0 TIMESHARING  
#/TOIBM  
OUTPUT TO? DX1:MYDATA  
TRANSLATE FROM ASCII TO EBCDIC <YES>? RET  
INPUT FROM? MYDATA.DAT  
RECORD LENGTH <128>? RET  
1 RSTS RECORD = 4 IBM RECORDS  
  
EXCHANGE COMPLETED  
MYDATA RESIDES ON 1 DISKETTE
```

In this /TOIBM dialogue, the RSTS/E file MYDATA.DAT is transferred to the IBM data set MYDATA on diskette DX1:. By default, the data is translated to EBCDIC on being transferred. FLINT computes and prints the blocking factor (1 RSTS/E record to 4 IBM records).

In the next example, a single density diskette is initialized and erased by means of the /ZERO and /ERASE operations.

```
RUN $FLINT  
FLINT V7.0 RSTS V7.0 Timesharing  
# ZER  
ZERO which diskette? 0  
Really ZERO DX0: <No> ? Y  
# ERA  
ERASE which diskette? 1  
Really ERASE DX1: <No> ? Y  
# ERA  
ERASE which diskette? 1  
Really ERASE DX1: <No> ? N  
ERASE aborted
```

## 18.7 FLINT Error Messages

FLINT, on encountering an error, prints the appropriate error message. If the error is non-fatal, FLINT repeats the prompting question; if the error is fatal, FLINT aborts the transfer. Table 18-1 lists and describes the error messages

specific to FLINT. Error messages specific to RSTS/E are described in Appendix C.

**Table 18-1: FLINT Error Messages**

<b>Message and Meaning</b>
<b>?DATA SET NAME TOO LONG</b> The specified IBM data set name has more than eight characters, including spaces and tabs.
<b>?DATA SET NOT FOUND</b> FLINT cannot find the specified data set on the current diskette.
<b>?ILLEGAL DATA SET NAME</b> A data set name has been specified where none is permissible.
<b>?ILLEGAL EXTENSION</b> The specified extension contains unacceptable characters or violates the specification format.
<b>?ILLEGAL FILE NAME</b> The specified filename contains unacceptable characters or violates the specification format.
<b>?ILLEGAL PPN</b> The specified account number contains wildcards, which are not allowed.
<b>?LOGICAL DEVICE NAME NOT ASSIGNED</b> The specified logical device name has not been previously assigned.
<b>?OUTPUT DEVICE MUST BE DISK</b> In a /TORSTS transfer, an output device other than the only permissible type--a RSTS/E disk--has been specified.
<b>?VOLUME #[m] CANNOT FOLLOW VOLUME #[n]</b> In a multi-volume /TORSTS transfer, the specified input volume number [m] is not 1 greater than the previously specified volume number [n].
<b>The following errors are fatal and cause the transfer to be aborted:</b>
<b>?BOE = EOD - TRANSFER ABORTED</b> In the specified data set, the Beginning Of Extent address is the same as the End Of Data address; i.e., the data set has no length.
<b>?DATA SET LABEL MUST BE HDR1 - TRANSFER ABORTED</b> The label ID of a specified data set for transfer is not HDR1, which it must be according to IBM format.
<b>?TRACK 00 DOES NOT CONTAIN ERMAP FIELD - TRANSFER ABORTED</b> There is no ERMAP field in track 00 of the specified diskette; according to IBM format, this field must be present.

## **18.8 FLINT as a CCL Command**

You can run FLINT (if the CCL command has been installed) by first typing the CCL command FLI[NT], then typing one of the dialogue initiation commands--/DIR[ECTORY], /TOR[STS], /TOI[BM], /ZER[O], or /ERA[SE]--depending on which dialogue you wish to run. Note that all these commands, including the CCL, may be minimally abbreviated to their first three letters. Here are three examples of CCL command strings:

```
FLI/DIRECT  
FLIN/TOR  
FLINT/TOIB
```

/DIRECT initiates the dialogue for listing a diskette directory, /TOR the dialogue for IBM-to-RSTS/E transfer, and /TOIBM the dialogue for RSTS/E-to-IBM transfer.

# **Chapter 19**

## **Device Control Programs**

### **19.1 Setting Terminal Characteristics: The TTYSET Program**

The TTYSET system program establishes terminal characteristics for your terminal. You can run TTYSET before or after you are logged into the system. If the terminal is not logged into the system, only one command can be entered to the TTYSET program at a time in the following format:

```
SET xxxx
```

where xxxx represents one of the TTYSET commands shown in Table 19-1. Each command is entered with the RETURN or ESCAPE key. If you are logged into the system, run TTYSET as follows:

```
RUN $TTYSET
TTYSET V7.0 RSTS V7.0 TIMESHARING
TERMINAL CHARACTERISTICS PROGRAM
? XXXX
```

where xxxx is again one of the TTYSET commands shown in Table 19-1. When the command has been entered to the system with the RETURN or ESCAPE key, TTYSET again prints ? to accept additional commands. To return control to command level, type EXIT, CTRL/C, or CTRL/Z.

**Table 19-1: RSTS/E TTYSET Commands**

Command	Function
LIST	Lists the current characteristics of the terminal.
CTRL/R	Enables the CTRL/R retype facility (see Section 4.9.3).
NO CTRL/R	Disables the CTRL/R retype facility (see Section 4.9.3).
WIDTH n	Sets the width of the print line for this terminal to n, which can be between 1 and 254. As a result, the system automatically generates a carriage return/line feed sequence if n printing characters have been printed or echoed without a carriage return/line feed sequence and another printing character is to be transmitted.
TAB	Enables hardware tab control. System transmits <HT> characters without translation.
NO TAB	Disables hardware tab control. To move to the next tab stop, the system transmits the correct number of space characters instead of transmitting an <HT> character.
FORM	Enables hardware form feed and vertical tab control. System transmits form feed and vertical tab characters without translation.
NO FORM	Disables hardware form feed and vertical tab control. System transmits 4 line feed characters in place of a form feed or vertical tab character.
LC OUTPUT	System transmits the lower-case characters CHR\$(96) through CHR\$(126) inclusive to the terminal without modification.
NO LC OUTPUT	System translates any lower-case character to its upper-case equivalent before transmitting it to the terminal.
XON	Special terminal hardware allows the computer to interrupt transmission of characters from the terminal by sending the terminal an <XOFF> character CHR\$(19). Similarly, the computer instructs the terminal to resume transmission of characters by sending the terminal an <XON> character CHR\$(17). The terminal hardware must respond to <XOFF> and <XON> characters by ceasing and resuming transmission, respectively. <b>The VT100 requires XON.</b>
NO XON	Terminal does not have hardware required for XON feature.
RESUME ANY CTRL/C	Defines the XON/XOFF processing. RESUME ANY enables typeout and echo when any character is typed after XOFF. RESUME CTRL/C enables typeout and echo only when <XON> or CTRL/C are typed after <XOFF>. <b>The VT100 requires RESUME CTRL/C.</b>
LOCAL ECHO	Terminal, or its acoustic coupler, echo prints characters as they are generated locally. System does not echo characters received from such a terminal.
FULL DUPLEX	Characters generated are sent only to the computer. System therefore echoes each character received so that it will be printed locally and translates certain characters to perform the proper action. For example, a <CR> character received is echoed as a carriage return and line feed character sequence.

(continued on next page)

**Table 19-1: RSTS/E TTYSET Commands (Cont.)**

Command	Function
SCOPE	<p>Terminal uses a CRT display and has the following characteristics:</p> <ul style="list-style-type: none"> <li>a. Conforms to synchronization as described under the STALL command,</li> <li>b. System echoes a &lt;DEL&gt; character (RUBOUT) as backspace, space, and backspace sequence.</li> <li>c. System generates &lt;NUL&gt; characters as fill for timing the following operations: home, erase to end of screen, erase to end of line, direct cursor addressing, and line feed.</li> </ul>
NO SCOPE	<p>Terminal is not a CRT display device. The system echoes a &lt;DEL&gt; character (RUBOUT) by printing a \ character and the last character typed and removes the last character typed from the terminal input buffer. Subsequent &lt;DEL&gt; characters cause the next to last characters to be sequentially printed and removed from the terminal input buffer until a character other than &lt;DEL&gt; is received. As a result, the system echoes another \ character to delimit the erased characters and echoes the correct character.</p>
LC INPUT	<p>Terminal transmits the full ASC11 character set and system does the following:</p> <ul style="list-style-type: none"> <li>a. Recognizes only the ESC character CHR\$(27) as an escape character,</li> <li>b. Echoes and uses the CHR\$(125) and CHR\$(126) characters without translation, and</li> <li>c. Echoes and uses lower-case alphabetic characters without translation.</li> </ul>
NO LC INPUT	<p>System treats ESC, }, and ~ characters (CHR\$(27), CHR\$(125), and CHR\$(126) respectively) as escape characters and translates lower-case characters received to their upper-case equivalents.</p>
NO FILL	<p>System does not generate fill characters for any characters transmitted.</p>
FILL LA30S	<p>Sets the fill characteristics for a serial DECwriter (LA30S).</p>
FILL n	<p>Sets the fill factor to n for this terminal where n is between 0 and 6. As a result, the system generates a multiple of fill characters for each hardware control character it transmits. See Section 19.1.3 for a discussion of generalized fill characters.</p>
SPEED n	<p>Sets to n the rate at which the terminal's interface can accept or pass characters. The value for n can be any legal speed for the interface, unless restricted by the system manager in the system file TTYSET.SPD. This command can be used only on lines whose interfaces can be programmed to handle more than one speed (e.g. DH11).</p>
SPLIT SPEED i/o	<p>Sets to i the rate at which the terminal's interface passes input to the computer and set to o the rate at which the terminal's interface accepts output from the computer. The values i and o can be any legal speed for the interface, unless restricted by the system manager in the system library file TTYSET.SPD. This command can be used only on lines whose interfaces can be programmed to handle more than one speed (e.g., DH11). <b>Split speeds are not recommended for VT100.</b></p>

(continued on next page)

**Table 19-1: RSTS/E TTYSET Commands (Cont.)**

Command	Function
NO PARITY	System ignores the parity bit on characters it receives and treats the parity bit on characters it transmits to the terminal as if it were a data bit.
EVEN PARITY	System sends characters to the terminal with the parity bit properly set for even parity but ignores the parity bit on characters received.
ODD PARITY	System sends characters to the terminal with the parity bit properly set for odd parity but ignores the parity bit on characters received.
STALL	Terminal obeys the following synchronization standard: if the terminal sends an <XOFF> character (equivalent to the CTRL/S combination), the computer interrupts transmission until the terminal sends either an <XON> character (equivalent to the CTRL/Q combination) or a CTRL/C combination. If the system receives an <XON> character, it does not keep that <XON> character as data. If the system receives another character, it resumes transmission and keeps that character as data. No characters are lost. <b>The VT100 requires STALL.</b>
NO STALL	<XON> and <XOFF> characters sent by the terminal have no special meaning.
UP ARROW	System echoes a control and graphic character combination as the ^ or † character (CHR\$(94)) followed by the proper graphic. For example, CTRL/E prints a ^ E or †E.
NO UP ARROW	System echoes control and graphic character combination as is.
PRINT filename	Sends the specified file to the terminal, in binary mode. This command's special use is in initializing a terminal for which special software settings must be made: setting 8 spaces to a TAB, for example. In such a case, 8 escape sequences would be placed in the file to be specified with PRINT.
ESC SEQ	System treats an ESC character CHR\$(27) as an indication of the start of an incoming escape sequence. The character is not echoed, nor are the characters in the sequence. The processing of input escape sequences is described in the <i>RSTS/E Programming Manual</i> .
NO ESC SEQ	System treats an ESC character CHR\$(27) as a line terminating character and echoes it as a \$ character.
DELIMITER x	Makes the printable character x a private delimiter for use with GET statements. Character will not be echoed.
DELIMITER "x"	Makes the nonprintable character x (TAB, for example) a private delimiter for use with GET statements. This format also works for printable characters. Character will not be echoed.
DELIMITER	Disables the private delimiter. Private delimiter is also disabled by any macro (multiple characteristic) command.
ESC	System treats only ASCII 027 (decimal) code as ESCAPE (ESCAPE or ALTMODE key).
NO ESC	System treats ASCII 027, 125, 126 (decimal) as ESCAPE (ESCAPE or ALTMODE key).
EXIT	Terminates a program and return control to the Monitor.
/RING	(Privileged only) Specifies characteristics of disabled terminals.
HELP	Prints a listing of single and macro commands.

In addition to the commands described in Table 19-1, TTYSET allows you to set all the individual characteristics for a certain type device by executing one command called a macro command. Each macro command assigns a set of predefined characteristics, any of which can be altered in turn by proper use of an individual characteristic command. Table 19-2 describes the default values for each macro command.

#### NOTE

1. None of the macro commands in Table 19-2 performs an automatic PRINT.
2. If no speed is given, the terminal speed is not changed from the current setting.

The TTYSET program checks commands before and during execution and reports errors by printing the messages described in Table 19-3. If any errors involve the terminal speed file, immediately notify the system manager or responsible system programmer.

If you wish to define non-standard macros or override any of the terminal characteristics set by existing macros, you can create a file \$TTYSET.MCM and include the desired TTYSET macro. Each entry must be on a separate line in the file and have the form:

MACRO,"char;char;...;char"

where MACRO is the name of your command and each characteristic (char) is a valid TTYSET macro or command.

For example:

VT52X,"VT52;WIDTH 120"

is a \$TTYSET.MCM file entry that alters the standard VT52 macro to allow a 120-character line. Note that \$TTYSET.MCM is examined only if the specified macro has no match in the standard TTYSET.BAS file (i.e., the macro names that you define must be unique).

#### 19.1.1 ESCAPE, ALTMODE, and PREFIX Characters

The RSTS/E system translates certain ASCII characters in a special manner. Some terminals have the outmoded ASCII character keys ALTMODE (125 decimal) and PREFIX (126 decimal). More recently designed terminals incorporate the 1968 ASCII character set and include the following control characters:

```
ESCAPE = 27 (decimal)
} = 125 (decimal)
~ = 126 (decimal)
```

The RSTS/E system interprets CHR\$(27) as a line terminating character and automatically translates any CHR\$(125) and CHR\$(126) characters input from a terminal into 27 (decimal). The system thus treats 125 (decimal) and 126 (decimal) as line terminators.

**Table 19–2: Default Single Characteristics Settings**

Individual Characteristics	Macro Command						ASR35
	VT05	VT05B	VT50	VT50H	VT52	VT55	
Width n	72	72	80	80	80	80	72
Tab/Notab	Tab	Tab	Tab	Tab	Tab	Tab	Notab
Form/Noform	Noform	Noform	Noform	Noform	Noform	Noform	Form
LC output/ No LC Output	LC Output						
XON/XOFF	NOXON	NOXON	NOXON	NOXON	NOXON	XON	XON
Local echo Full Duplex	Full Duplex						
Scope/ No Scope	Scope	Scope	Scope	Scope	Scope	Scope	No Scope
LC Input/ No LC Input	No LC Input	No LC Input	No LC Input	No LC Input	LC Input	LC Input	No LC Input
Fill n Fill LA30s	0	3	0	0	0	0	1
Speed n Split Speed i/o 2741	-	-	-	-	-	-	110
No Parity Even Parity Odd Parity	No Parity						
Stall/ No Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall
Up Arrow/ No Up Arrow	Up Arrow						
ESC Seq/ No ESC Seq	No ESC Seq						
ESC/No ESC	ESC	ESC	ESC	ESC	ESC	ESC	No ESC
Delimiter 0	DEL						
Resume <b>CR</b> / Resume Any	-	-	-	-	-	-	-

(continued on next page)

**Table 19-2: Default Single Characteristics Settings (Cont.)**

Individual Characteristics	Macro Command						
	LA30	LA30S	LA36	LA120	LA180	KSR33	KSR35
Width n	80	80	132	132	132	72	72
Tab/Notab	Notab	Notab	Tab	Notab	Notab	Tab	Tab
Form/Noform	Noform	Noform	Form	Form	Noform	Form	Noform
LC Output/ No LC Output	LC Output						
XON/NOXON	NOXON						
Local Echo/ Full Duplex	Full Duplex						
Scope/ No Scope	No Scope						
LC Input/ No LC Input	No LC Input	No LC Input	LC Input	LC Input	No LC Input	LC Input	No LC Input
Fill n	0	Fill LA30S	0	0	0	1	2
Speed n	-	-	-	-	110	110	2741
Split Speed i/o	2741	-	-	-	-	-	-
No Parity/ Even Parity/ Odd Parity	No Parity						
Stall	Stall	Stall	Stall	Stall	Stall	Stall	Stall
Up Arrow/ No Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	Up Arrow	No Up Arrow
ESC Seq/ No ESC Seq	No ESC Seq						
ESC/No ESC	ESC	ESC	ESC	ESC	No ESC	No ESC	No ESC
Delimiter 0	DEL						
Resume <b>CTRL-Q</b> Resume Any	-	-	-	-	-	-	-

**Table 19-3: TTYSET Error Messages**

Message and Meaning
?ILLEGAL PRIVATE DELIMITER - <string> Delimiter was not specified or has been disabled.
?ILLEGAL FILE TO PRINT No filename specified in PRINT command.
?TERMINAL OPEN ERROR - <text> The RSTS/E error denoted by <text> was encountered while executing the command.
?<string> IS ILLEGAL The keyboard number denoted by <string> is not between 0 and 63 or is not a valid number.
?ILLEGAL FILL FACTOR Fill factor specified is not between 0 and 6.
?ILLEGAL WIDTH Width specified is not between 1 and 254.
?COMMAND <string> ILLEGAL Command indicated by <string> is undefined.
?ILLEGAL SPEED Speed given is not one defined for the device in the \$TTYSET.SPD file.
%WARNING - \$TTYSET.SPD ERROR Program warns you of possible corruption in the terminal speed file and denotes the exact error by printing the COMMAND ERROR message.
?COMMAND ERROR - <text> The RSTS/E error denoted by <text> was encountered when executing the command.
?ERROR - <text> The RSTS/E error denoted by <text> was encountered when executing the system function to change terminal status.
%WARNING - CANNOT OPEN \$TTYSET.SPD FILE Program could not access the terminal speed file and denotes the exact error by printing the COMMAND ERROR message.

If you have a terminal with the 1968 ASCII character set, you can make the system treat 125 (decimal) and 126 (decimal) characters as they are printed rather than as control characters. The TTYSET commands LC INPUT and NO LC INPUT;ESC alter internal parameters for a given terminal so that the system treats 125 (decimal) and 126 (decimal) as they are printed or translates them automatically to their upper case equivalents.

#### **19.1.2 Lower- and Upper-Case Characters**

Some terminals can send and print both lower-case and upper-case characters. Such terminals can therefore print the echo returned by the system to give you an accurate visual representation of the character transmitted.

Terminals such as the VT05 alphanumeric display can send either lower-case or upper-case characters but can print only upper-case characters. Consequently, the echo response of such a terminal to a lower-case character is the upper-case counterpart. The terminal gives you no visual indication that the character transmitted was a lower-case character.

Terminals such as the ASR-33 and KSR-33 type devices neither send nor receive lower-case characters. If such a terminal receives a lower-case character, it prints the corresponding upper-case character.

Normally, RSTS/E software translates all lower-case characters to their upper-case counterparts before processing them. To take advantage of different features of terminal hardware, you can include or omit lower-case translation depending upon whether the terminal produces lower-case output, lower-case input, or both. The LC OUTPUT and LC INPUT commands, respectively, omit translation of lower-case characters generated as output on and input from a terminal. NO LC OUTPUT causes lower- to upper-case translation on output to the terminal; NO LC INPUT causes lower- to upper-case translation on input from the terminal.

### 19.1.3 Generalized Fill Characters

The RSTS/E system automatically generates a variable number of <NUL> characters (CHR\$(0%)) as fill characters after outputting certain control characters. The generation of these meaningless <NUL> characters allows the terminal sufficient time to complete the physical action initiated by the control character, thus permitting the terminal to synchronize itself properly for printing the next meaningful character. The values in Table 19-4 interrelate the numbers of <NUL> characters generated for control characters at certain TTYSET characteristic settings.

#### NOTE

If the fill factor is 0, no fill characters are ever generated. The expressions in Table 19-4 do not apply for FILL 0.

The TTYSET command FILL n sets the fill factor to be used in Table 19-4; the command NO FILL or FILL 0 disables the automatic generation of fill characters.

In Table 19-4, the fill factor is used as an exponent of 2. When the fill factor is increased by 1, therefore, the number of fill characters generated is doubled. Most terminals require no fill characters at low baud rate settings. But at some baud rate (for example, 600 baud for VT05s), a terminal starts to require fill characters. As the baud rate increases, the number of fill characters required also increases. Because the common baud rates increase by doubling, increase the fill factor by 1 each time the baud rate is doubled. (For example, the appropriate fill factor for a VT05 at baud 600 is 1; at 1200, 2; at 2400, 3; etc.)

### 19.1.4 XON/XOFF Remote Reader Control

To operate a low speed paper tape reader connected to the RSTS/E system by either a data set (dial up) or a communications line, two requirements must be fulfilled as follows.

**Table 19-4: Generalized Fill Characters**

Control Character	Decimal Value	Scope	No Scope	Form	No Form	Tab	No Tab
CR	13	0	2741: POS/10+1	NOT 2741: 1 x 2 <sup>Fill-1</sup>	N/A	N/A	N/A
			Fill LA30s*				
LF	10	1 x 2 <sup>Fill-1</sup>	0	N/A	N/A	N/A	N/A
HT(tab)	9	N/A	N/A	N/A	N/A	1 x 2 <sup>Fill-1</sup>	Spaces are sent.
VT(Vert. Tab)	11	1 x 2 <sup>Fill-1</sup>	See FORM/NO FORM	5 x 2 <sup>Fill-1</sup>	Do 4 LFs	N/A	N/A
FF	12	N/A	N/A	9 x 2 <sup>Fill-1</sup>	Do 4 LFs	N/A	N/A
CTRL/N (direct cursor addressing) VT05	14	1 x 2 <sup>Fill-1</sup>	0	N/A	N/A	N/A	N/A

\*Function of type head's position at time of CR; this function is non-linear.

- a. The terminal must be equipped with the requisite hardware option for XON/XOFF remote reader control.
- b. The XON/XOFF feature must be enabled for the given terminal.

You can selectively enable and disable remote reader control for a remote terminal by the TTYSET commands XON and NO XON.

For low speed readers connected to the system by a local line interface, none of these requirements is necessary.

Terminals with the ability to transmit characters to the RSTS/E system at a high rate of input require the XON characteristic, which allows temporary suspension of input when the system's buffers are full.

#### **19.1.5 Output Parity Bit**

The RSTS/E software always ignores the parity bit on characters received from a terminal and omits the parity bit on characters it transmits. Because some terminals not supplied by DIGITAL can receive even or odd parity characters, the software must be conditioned to send parity characters. The TTYSET commands EVEN PARITY, ODD PARITY, or NO PARITY condition the software to send the correct parity. The software ignores parity bits on characters input to the system.

#### **19.1.6 Private Delimiters**

TTYSET enables you to establish, on your keyboard, a special delimiter for use in BASIC-PLUS GET statements. Creating such a private delimiter is especially useful on a data entry terminal with a specialized keyboard: you can designate a large or conveniently located key as the delimiter key. Note, however, that a character designated as a private delimiter will not echo at the terminal. This rule applies to nonprintable as well as printable characters; <TAB>, for instance, will have no visible effect.

To make a private delimiter of a printable character, type the SET DELIMITER command followed by the character, either with or without enclosing quotation marks. The following examples illustrate the format:

SET DELIMITER A

or

SET DELIMITER "A"

Either of these commands makes the character A into a private delimiter.

To make a private delimiter of a nonprintable character, type the SET DELIMITER command followed by the nonprintable character enclosed in quotation marks. The following example makes the <TAB> character into a private delimiter:

SET DELIMITER "

To disable the private delimiter, type the SET DELIMITER command with no argument:

SET DELIMITER

The private delimiter is also disabled by any macro (multiple characteristic) command: for example, SET VT05, SET LA36, etc.

**19.1.6.1 Limitations of Private Delimiters** — As previously mentioned, a character designated as a private delimiter will not echo at the terminal. Thus, making a private delimiter of a common formatting character such as <TAB> may cause confusion in editing a program.

The SET DELIMITER command causes the existing ASCII code for the specified character to be overridden. Therefore, if the character normally has an expanded or alternate function, that function will be eliminated by the command. RUBOUT, for example, will neither backspace (on a display terminal) nor print backslashes (on a hard-copy terminal).

Private delimiters work only in the GET statement--not in the INPUT or INPUT LINE statement.

### **19.1.7 SET as a CCL Command**

Under a standard RSTS/E system, you can execute TTYSET by the correct Concise Command Language (CCL) command. To execute a CCL command, issue the proper CCL command followed by the desired TTYSET command(s). Multiple TTYSET commands on the same line are separated by semicolons (;). For example:

```
SET LA36;WIDTH 80
```

For more information on CCL commands, see Section 12.1.

## **19.2 Mounting and Dismounting Magnetic Tapes and Private Disks: The UMOUNT Program**

A non-privileged or privileged user can execute the MOUNT or DISMOUNT commands of the UMOUNT system program to logically mount and dismount magnetic tapes, private disk packs, and cartridges. MOUNT and DISMOUNT are CCL commands and must be installed on the system by the system manager. Otherwise, the system prints the ?WHAT? error message. An attempt to run the UMOUNT program by any other means generates the message PLEASE USE THE 'MOUNT' OR 'DISMOUNT' COMMAND.

### **19.2.1 Logically Mounting a Disk Pack or Cartridge**

The following steps are taken to logically mount a private pack or cartridge on the system:

1. Ensure that the device is available by running the SYSTAT program, then load the pack or cartridge in the available drive.
2. Ensure that the drive is write enabled.

3. Type the MOUNT command followed by the device designator, the appropriate identification label, and any option(s) desired.

The following example illustrates the format of the MOUNT command:

```
MOUNT DK1:MYPACK
```

```
READY
```

As a result of this command, the UMOOUNT program runs and logically associates RK05 drive unit 1 with the identification label MYPACK. (Disk options are described in Section 19.2.2.) The READY message indicates that the disk is available for read and write access and in the UNLOCK state.

An attempt to write a file on the disk when the current account is non-existent generates the ?DISK PACK IS PRIVATE error. Ask the system manager or responsible system programmer to create the current account on the disk.

If any error occurs in the device designator or in the identification label, the program prints the following error message:

```
?DECODE ERROR IN MOUNT - <text>
```

The notation <text> represents the UMOOUNT or RSTS/E error encountered. On printing this message, the program terminates.

If any error occurs in mounting the disk on the system, the program prints the following error message:

```
?PROCESS ERROR IN MOUNT - <text>
```

The notation <text> represents the UMOOUNT or RSTS/E error encountered. Typical errors include having the device protected against writing (?DEVICE HUNG OR WRITE LOCKED), specifying an incorrect identification label (?PACK IDS DON'T MATCH), and trying to mount a disk which is already mounted (?DISK IS ALREADY MOUNTED.).

If the program encounters an error when attempting to unlock the disk to enable write access, it prints the following message:

```
?PROCESS ERROR IN UNLOCK - <text>
```

The notation <text> represents the UMOOUNT or RSTS/E error encountered. The pack, however, is mounted and available for read access only. Report any error of this kind to the system manager or responsible system programmer.

### **19.2.2 MOUNT Options: Disk Pack or Cartridge**

The options described in this section may be used with the MOUNT command when mounting a private disk. Options are summarized in Table 19-5.

If the disk to be mounted must be kept locked, append the /LOCK option to the MOUNT command, as in the following example:

```
MOUNT DR1:DATA1/LOCK  
READY
```

As a result of this command, UMOUNT runs and logically associates the identification label DATA1 with RM02 unit 1. The READY message indicates that the disk pack is in the LOCK state. Only privileged users have read and write access to the disk pack.

If the disk to be mounted is to be read only, append the /READ ONLY option to the MOUNT command, as in the following example:

```
MOUNT DK1:MYPACK/RONLY
```

As a result of this command, the disk may be referred to as DK1: and MYPACK, and can be read but not written.

If a logical name other than the pack identification label is desired for the disk, append the option /LOGICAL:, followed by a logical device name, to the MOUNT command. The following example illustrates:

```
MOUNT DK1:MYPACK/LOGICAL:PACK3
```

As a result of this command, the disk may be referred to either as DK1: or PACK3.

If only the physical device specification is to be allowed for the mounted disk, append the option /NOLOGICAL to the MOUNT command, as in the following example:

```
MOUNT DK1:MYPACK/NOLOGICAL
```

As a result of this command, the disk may be referred to only by its physical specification, DK1:. No logical names are allowed.

The MOUNT command also has a privileged option that allows a qualified user to mount a public disk as private. The /PRIVATE option is appended to the MOUNT command, as follows:

```
MOUNT DK1:PUB001/PRIVATE
```

As a result of this command, the disk is mounted as a private disk, and is not used as part of the public structure. The disk may be referenced as PUB001 and as DK1:. Note that the /PRIVATE option is privileged.

### 19.2.3 Mounting a Magnetic Tape

The MOUNT command can be used with a magnetic tape designator to assign a unit to the current job, to set default labeling format, and to set the tape's density and parity. The options described in this section are summarized in Table 19-5.

The following steps are taken to mount a magnetic tape:

1. Ensure that the device is available by running the SYSTAT program.
2. Physically mount the tape on the drive and ready the drive.
3. Type the MOUNT command.

To set the magnetic tape labeling format, append the option /DOS or /ANSI to the MOUNT command. (For a description of label formats, see Section 5.1.) If /ANSI is specified, the appropriate volume identification label should appear following the device designator. The following example shows the procedure:

```
MOUNT MT1:MYVOL/ANSI  
READY
```

As a result of this command, the identification label is checked and, if correct, tape unit 1 is assigned to the current job with ANSI labeling default. The /ANSI option sets the labeling default to ANSI standard format.

Under normal conditions, the system checks the tape identification label to ensure that it matches the /ANSI or /DOS specification. To override this check and cause the system to mount the tape based solely on the format specification, specify the /NOCHECK option prior to the format specificaiton. For example:

```
MOUNT MT1:/NOCHECK/ANSI
```

causes the system to assign tape unit 1 to the current job with ANSI labeling format, regardless of the tape's current labeling format. This option is useful when you wish to initialize a new tape (see Section 16.2.9).

To set the tape's density and parity, the options /DENSITY:nnnn and /PARITY:nnnn should be appended to the MOUNT command, for example:

```
MOUNT MT1:/DOS/DENSITY:800/PARITY:ODD
```

As a result of this command, tape unit 1 is assigned to the current job with DOS/BATCH labeling format, 800 BPI density, and ODD parity. If the density or parity specified is illegal on the current tape drive or is not that found on the mounted tape, an error message is printed. System-wide defaults for magnetic tape density, parity and label are set by the system manager during system initialization. If you wish the system to accept a tape with non-default

characteristics, you must ASSIGN the tape prior to setting those characteristics (with MOUNT or PIP). Note that when the tape is DEASSIGNED, the system reverts to the default characteristics set during system initialization.

If the magnetic tape is to be mounted read only, the /RONLY switch should be appended to the MOUNT command. For example:

```
MOUNT MT1:/DOS/ONLY
```

As a result of these commands, tape unit 1 is assigned to the current job with DOS/BATCH labelling format, the currently assigned default density and parity, and the read only option.

If the current job is privileged, the /JOB:n option can be used with the MOUNT command to reassign the unit to job number n. The following example shows the procedure:

```
MOUNT MT1:/JOB:5
```

```
READY
```

As a result of this command, tape unit 1 is reserved to job 5 with the currently assigned default labeling format. If job 5 is not active, an error message is printed. If an unassigned logical name is given for the magnetic tape unit in the MOUNT command, UMOUNT prints the error text ?ERROR IN MOUNT - ?ILLEGAL DEVICE and returns to command level.

If any error occurs while mounting a magnetic tape, UMOUNT prints:

```
?PROCESS ERROR IN MOUNT - <text>
```

The <text> will be the UMOUNT or RSTS/E error encountered.

When mounting ANSI labeled tapes, the volume ID should be specified. If neither the volume ID nor the /NOCHECK option is specified, UMOUNT prints the warning:

```
%VOLUME ID SHOULD BE SPECIFIED ON ANSI MAGTAPE MOUNT
```

and continues processing.

#### **19.2.4 Logically Dismounting Disk Packs, Cartridges, and Magnetic Tapes**

The following steps are taken to logically dismount a private pack or cartridge:

1. Determine the number of OPEN files on the device by running SYSTAT. If non-zero, wait until all files are closed before proceeding. If zero, proceed. Note that a device, which contains files currently in a queue or batch stream, must remain mounted until the queue or batch is executed (see Chapters 20 and 21).

- Type the DISMOUNT command followed by the device designator of the drive and the pack identification label. (The pack identification is only required if you are not privileged.)

The following example illustrates the format of the DISMOUNT command:

```
DISMOUNT DK1:PACKID
```

```
READY
```

As a result of this command, the UMOUNT program runs and logically dismounts the pack on RK05 drive 1; the pack identification is removed from the system's table of logical names. The READY message indicates that the drive is free for other usage and that you can safely remove the pack or cartridge from the drive.

If the program encounters an error when it attempts the dismount action, it prints a message in the following format:

```
?PROCESS ERROR IN DISMOUNT - <text>
```

The notation <text> represents the UMOUNT or RSTS/E error encountered. A typical error is attempting to dismount a disk which has open files (?ACCOUNT OR DEVICE IN USE).

To rewind a magnetic tape and take it off line, append the /UNLOAD option to the DISMOUNT command, as in the following example:

```
DISMOUNT MTO:/UNLOAD
```

As a result of this command, the tape on unit 0 is rewound and placed off line.

**Table 19-5: The UMOUNT Options**

Device	Option	Function
disk pack	/LOCK	Keep the mounted disk locked.
disk pack	/LOGICAL	Allow a logical name other than the pack ID for the mounted disk.
disk pack	/NOLOGICAL	Disallow logical name references to the mounted disk; allow only physical specification.
disk pack	/PRIVATE	Allow a public disk to be mounted as private (privileged only).
disk pack	/RONLY	Allow the mounted disk to be read only; prevent all write access.
magnetic tape	/ANSI	Set magnetic tape labeling default to ANSI format.
magnetic tape	/DOS	Set magnetic tape labeling default to DOS format.

(continued on next page)

**Table 19-5: The UMOUNT Options (Cont.)**

Device	Option	Function
magnetic tape	/NOCHECK	Override system check and set magnetic tape labeling default as specified (/ANSI or /DOS).
magnetic tape	/JOB:n	Reassign magnetic tape unit to job number n (current job must be privileged).
magnetic tape	/DENSITY:800	Set density to 800 BPI.
magnetic tape	/DENSITY:1600	Set density to 1600 BPI.
magnetic tape	/PARITY:ODD	Set to odd parity.
magnetic tape	/PARITY:EVEN	Set to even parity.
magnetic tape	/UNLOAD	(With DISMOUNT command) Rewind a magnetic tape and take it off line.

**Table 19-6: UMOUNT ERROR <text> Messages**

Message and Meaning
?ILLEGAL LOGICAL NAME - <name> <name> is illegal because it is not a system logical name.
?ILLEGAL OPERAND <operand> <operand> is not one of the legal values.
?1600 BPI ILLEGAL Parity has been specified for this volume; 1600 BPI density is therefore illegal.
?ILLEGAL PARITY SWITCH Density of 1600 BPI has been specified for this volume; parity specification is therefore illegal.
?DISK PACK IS PUBLIC UMOUNT is used for private disks only.
?DISK IS ALREADY MOUNTED Cannot logically mount a disk that is already logically mounted.
?DISK DOES NOT HAVE RSTS FILE STRUCTURE UMOUNT processes RSTS/E file structured disks only.
?PACK ID MUST BE SPECIFIED To dismount a disk, a non-privileged user must specify the disk's pack ID.

(continued on next page)

**Table 19-6: UMOUNT ERROR <text> Messages (Cont.)**

<b>Message and Meaning</b>
?DENSITY/PARITY SETTING CANNOT BE SET ON THIS DRIVE Cannot use a magnetic tape of the specified density and/or parity on this drive.
?INVALID ANSI MAGTAPE LABEL The value of the volume label identifier and label number is not VOL1.
?VOLUME ID'S DON'T MATCH When mounting an ANSI magnetic tape, volume ID of tape did not match with the ID specified (/NOCHECK was not specified).
?TAPE IS NOT WRITE LOCKED When magnetic tape was mounted, /RONLY was specified, but tape is physically write enabled.
?TAPE IS WRITE LOCKED The magnetic tape must be mounted write-enabled when /RONLY is not specified in mount command.
?ILLEGAL DEVICE Device specified is not disk or magnetic tape.
?ILLEGAL COMMAND Command unrecognizable or incomplete.
?NO DISK DEVICE UNIT NUMBER SPECIFIED Disk device unit number must be specified.
?SYNTAX ERROR Illegal field(s) in command string.



## **Chapter 20**

# **Using System Spooling Services: The QUE Program**

The QUE system program creates requests, or jobs, which are to be executed by spooling programs. QUE also executes auxiliary operations such as listing pending requests, killing pending requests, and modifying pending requests. The QUE program checks the syntax and validity of each request and passes it to another program for further processing if the request is valid. If any part of a request is invalid, the program rejects the entire request, prints an appropriate error message, and allows you to try again.

For the system to execute a request created by QUE, the system must contain three other programs: a queue manager program (QUEMAN), an operator services program (OPSER), and a spooling program. QUEMAN processes a request created by QUE and places it in the system queue file. OPSER communicates between the operator and the spooling programs. The particular spooling program executes requests on its related device when the QUEMAN program passes it a pending request. Valid spooling devices are as follows:

- LP: Line Printer
- BA: Batch
- RJ: RJ2780
- PP: Paper Tape Punch
- NR: Nine Thousand Remote Batch

The spooling programs supplied on standard RSTS/E systems are SPOOL (LP:), BATCH (BA:), and (optionally) RJ2780 (RJ:). A spooling program is not supplied for paper tape. Note that if the NR: spooling device is present on your system, the default file type is .NTR.

## 20.1 Running QUE at a Terminal

A privileged or non-privileged user runs QUE at a terminal logged into the RSTS/E system by typing the RUN \$QUE command as follows:

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

When QUE runs, it prints a header containing the program name, its version and revision numbers, and the system name and version and revision numbers. The program ensures that the system queue file exists and is accessible on the system library account. If no errors occur, QUE prints a number sign (#) prompt to signal its readiness for a command. QUE prints this number sign after processing each command.

If QUE runs and encounters an error when accessing the queue file, it prints the following message and reprompts:

```
?QUEUE NOT INITIALIZED
#
```

To run QUE without an error, have the system manager run the QUEMAN program and initialize the queue file.

You can terminate QUE by typing E or the CTRL/Z combination in response to the # sign character, as follows:

```
# E
READY
```

Control is returned to system command level, as indicated by the READY prompt. Commands to queue, list, kill, and modify jobs and the related error messages are explained in the following sections. A summary of the commands appears in Table 20-1.

**Table 20-1: QUE Program Commands**

Command	Format	Description
Que	Q device:jobname=file spec(s)	Allows you to give file specification(s) of a file to be processed. A comma is used to separate multiple file specifications. See Table 20-2 for options you can give with a file specification.
List	L device:=jobname(s)	Prints at the terminal a list of the currently pending requests for the spooling program. If device: is not given, LP0: is assumed. If device: is not given and jobname(s) is given, the = character is required. Valid devices are LP:, LP0:-LP7:, BA:, BA0:-BA7:, RJ:, PP:, and NR:. Specifying LP: or BA: indicates all units of that device. If no jobname appears, QUE prints all jobs.

(continued on next page)

**Table 20-1: QUE Program Commands (Cont.)**

Command	Format	Description
Short	S device:=jobname(s)	Prints at the terminal a short form list of the currently pending requests for the spooling program. If device: is not specified, LP0: is assumed. If device: is not specified and jobname(s) is specified, the = character is required. Valid devices are LP:, LP0: - LP7:, BA:, BA0: - BA7:, RJ:, PP:, and NR:. Specifying LP: or BA: indicates all units of that device. If no jobnames appear, QUE prints all jobs.
Kill	K device:=jobname(s)	Removes from the queue the job(s) indicated by device specification and jobname. If device: is not given, QUE assumes LP0:. If device: is not given and jobname(s) is given, the = character is required.
Modify	M device:jobname/options	Modifies the parameters of a job already in the queue. The following parameters can be modified by the M command: priority, forms name, after date or time, type of forms control, number of job copies, job status, and mode.
Flush	F dev: F BA:	Causes QUE to abort all current jobs in the queue. BA: specifies the BATCH Processor, dev: specifies a line printer unit (LP0: is the default). If you specify LP: with no unit number, all jobs in any device queue are aborted. The F command (FLUSH) is privileged and is available only to privileged users or to those included in the OPSER operator table.
Help	H	Causes an information message to be printed at the terminal.
Exit	E	Causes QUE to terminate and to return control to system command level.
CTRL/Z	You type the CTRL/Z combination	Causes QUE to terminate (same as E).

## 20.2 Using the Q Command

The Q command typed in response to the # sign creates a request for a spooling program. The Q command has the following general format:

Q device:jobname=file spec,file spec,...,file spec

where:

device: is the device designator LP:, LP0:-LP7:, BA:, BA0:-BA7:, or RJ: and the unit number of the device which the spooling program services. If a designator is not given, LP0: is assumed. The designators LP: and BA: cause the request to be queued for any available spooling program for that device type. If a device is specified, the = sign is required.

If a unit number is specified, the request is queued for that unit and is executed only if a spooling program is running on that unit. If a unit number is not specified (LP: or BA:), the request is for the first available device. The general batch processor BA: processes only those jobs queued for BA: and not jobs queued for BA0: through BA7:. BATCH processors BA0: through BA7: however, process requests queued for their respective units and requests queued for BA:.

**jobname** is a maximum of six alphanumeric characters to identify your request in the queue. If device: and jobname are not specified, the = sign is optional and the filename of the first file specified in the request is the job name. Several options described in Table 20-2 may accompany the jobname.

#### NOTE

For privileged users, or for non-privileged users included in the OPSER program's operator table (by the system manager), an account specified in the jobname parameter can be any valid account on the system. If not specified, the default is your current account. If specified, you assume the level of privilege of the account specified. Also, a file with the privileged protection code (<128>) can only be queued by a privileged user; for such a file, the /DE switch (see Table 20-3) is not allowed.

**file spec** is the file specification of the file to be processed from the queue and any combination of Q command options. Up to 11 files can be included in a request. File specifications are separated by a comma. (See Table 20-3 for the options.)

The jobname (on the left hand side of the = character) can have several output options which apply to the entire request. Each option must be preceded by a slash character (/). Options may be minimally abbreviated to their first two letters; for example, /MO, /AF, /PR, etc. Table 20-2 lists and describes the output options.

The /AFTER option, which can be specified with the jobname (and is briefly described in Table 20-2), causes the queue manager to initiate a requested job after a particular date and time. The /AFTER option arguments that specify the date and time take one or more of the following forms:

**calendar date** in the form :dd-mmm-yy, where dd is the date, mmm is the first 3 letters of the month, and yy is the year. For example, to initiate a job after midnight of September 22, 1979 (i.e., September 23):

/AFTER:22-SEP-79

If the year is not specified, the default is the current year.

numeric date	in the form :yy.mm.dd, where yy is the year, mm is the number of the month, and dd is the day. For example, to initiate a job after midnight of November 14, 1979:  /AF:79.11.14
relative date	in the form :+n D[AYS], where n is the number of days from the current date. For example, to initiate a job after midnight of September 5 (when the current date is September 3):  /AFTER:+2 D
military time	24-hour time in the form :hh:mm, where hh is the hour and mm is the minute. For example, to initiate a job on December 12 of the current year after 1:30 pm:  /AFTER:12-DEC:13:30
AM/PM time	in the form :hh:mm:am/pm, where hh is the hour, mm is the minute, and am or pm is used to designate before or after noon. For example, to initiate a job after 2:30 p.m. on October 15, 1979:  /AF:79.10.15:02:30:PM
relative time	in the form :+n H[OURS], where n is an integer in the range of 0 to 24. For example, to initiate a job 5 hours after the current time:  /AFTER:+5 H

The date and time specifications used with the /AFTER option are subject to the following rules:

1. The date specifications (calendar, numeric, and relative) when used without a time specification, cause the job to begin after midnight of the specified date.
2. Only one specification of each type is allowed. For instance, a relative date cannot be used with a calendar date nor can a relative time be used with military time.
3. When a relative time of 0 is used with a relative date specification, it causes the job to begin after the current time on the date specified. For example, if the following option is used at 3 p.m. on September 12:

/AFTER:+ 2 DAYS:+0 HOURS

the job is initiated after 3 p.m. on September 14.

## NOTE

The filename options /CO:nn, /NH, /DE, and /BI may also be specified as job options. (These are described in Table 20-3.) Any filename option so specified is applied to each file in the job.

The external file specification of a file to be included in the job can have the following form:

device:filename.extension[proj,prog]/option/option...

A valid device is either a valid disk designator or null (which indicates the system device in the public structure). QUE interprets logical names if you make the assignment before QUE runs. Otherwise, an unassigned logical device name generates the ?ILLEGAL INPUT FILE error. See Section 4.4 for a description of logical names.

**Table 20-2: QUE Job Output Options**

Option Format	Description						
<b>/MODE:n</b>  <b>/LENGTH:nnn</b> <b>/CONVERT</b> <b>/TRUNCATE</b> <b>/LPFORM</b> <b>/UPPERCASE</b> <b>/SKIP</b>	<p>The value n specifies the MODE which the spooling program will use in its OPEN statement for the output device (see the <i>RSTS/E Programming Manual</i>). The following specific MODE options may be used instead of /MODE:n.</p> <p>The value nnn (1 to 127) sets form length in lines per page.</p> <p>Change character 0 (zero) to character O (oh).</p> <p>Truncate lines longer than unit's configured length.</p> <p>Enable software formatting.</p> <p>Translate lower- to upper-case characters.</p> <p>Skip 6 lines at bottom of each form.</p>						
<b>/AFTER:spec</b>	<p>The value in spec represents the date and time after which the queue manager will initiate the current request. The values in spec can be one or more of the following as described in Section 20.2:</p>						
<b>/PRIORITY:n</b>	<p>QUE sets the priority to the value n which can be between 0 and 255. If /PRIORITY:n is not specified, QUE assigns the value 128. A privileged user can specify a priority between 0 and 255. A non-privileged user can specify a priority between 0 and 127. The priority setting determines the job's place in the queue; the higher the number, the higher the priority. If two or more jobs have the same priority, they go to the spooler in order of job requests.</p>						
<b>/TYPE:xxx</b>	<p>Use the format specified by xxx for printing the file. Value may be:</p> <table style="margin-left: 40px;"> <tr> <td>FTN</td> <td>FORTRAN forms control.</td> </tr> <tr> <td>EMB</td> <td>Embedded forms control (equivalent to paper tape).</td> </tr> <tr> <td>IMP</td> <td>Implied forms control (LF and CR printed before each record).</td> </tr> </table>	FTN	FORTRAN forms control.	EMB	Embedded forms control (equivalent to paper tape).	IMP	Implied forms control (LF and CR printed before each record).
FTN	FORTRAN forms control.						
EMB	Embedded forms control (equivalent to paper tape).						
IMP	Implied forms control (LF and CR printed before each record).						
<b>/FORMS:&lt;form name&gt;</b>	<p>The string &lt;form name&gt; specifies the name of the form on which the job is printed; &lt;form name&gt; is up to 6 characters, of which the first character should not be a digit. The default form name is NORMAL.</p>						
<b>/JCOPIES:n</b>	<p>The value n is the number of job copies to be printed. If /JCOPIES:n is not specified, one copy of the job is printed.</p>						

The filename and extension may consist of ? and \* wildcard characters described in Section 11.3.1. A file specified with no extension is given the default extension .LST if it is in a line printer queue, or the default extension .CTL if it is in a BATCH queue. The [proj,prog] designation is the project-programmer number (account) under which the file to be printed is stored. The designation can be omitted if the file is stored under your current account. Note that wildcard characters (?) or (\*) cannot be used to designate project-programmer numbers. The /option designation is a slash character (/) followed by one of the Q command options. The options are described in Table 20-3.

**Table 20-3: Q Command Options**

Option	Format	Description
Copies	/CO:nn	Causes SPOOL to print the number of copies indicated by the decimal integer n. If /CO:n is not given, one copy is printed.
Noheader	/NH	Causes SPOOL to suppress printing of the file header.
Delete	/DE	Causes SPOOL to delete the file after it is printed if the protection code of the file permits. If /DE is not specified, the file is left intact. The QUEMAN program checks the protection code of the file.
More	/MORE	Used only at the end of a command line to indicate to QUE that you have more text to type on the next physical line.
Binary	/BI	Used to indicate a binary file to the RJ2780 program, which must be in TRANSPARENT mode.

The following command typed to QUE:

```
* Q ABC.DAT
*
```

causes the file ABC.DAT stored under your current account on the public structure to be sent to the queue manager for printing on the spooled line printer, unit 0. SPOOL generates one copy of the file under job header ABC.

To queue a file from a device other than the public disk, specify the device designator in the command. For example:

```
* Q DK1:A.BAS/CO:2
*
```

The indicated file on DK1: is queued for printing on line printer unit 0. SPOOL generates two copies of the file under the job header A.

To specify a job name for the request, type the name and the = character in the command. For example:

```
* Q SORT=DK1:FILEA,FILEB/CO:2
*
```

As a result, SPOOL prints SORT in the job header and generates one copy of DK1:FILEA.LST and two copies of DK1:FILEB.LST.

To specify a spooling program other than the one for line printer unit 0, type the appropriate designator and the = character in the command. For example:

```
*Q LP1:=DK1:FILOUT.001  
*
```

As a result, QUE creates a request for FILOUT.001 on line printer unit 1, with job name FILOUT.

To specify a request longer than one physical line, type the /MORE option as the last item on the line. For example:

```
*Q LP1:PRINT1=FILE1.001,ABCDEF.001,/MORE  
MORE > HELP.TXT,ACCT.SYS  
*
```

As a result, QUE prints, as a prompting indicator, the text MORE>, after which you can continue typing the request. QUE allows up to eleven files in one request. Type /MORE as many times as necessary. Note that /MORE is invisible to the parser; the completed command is interpreted as if it were typed as one line. Therefore, all punctuation (commas, colons, etc.) must appear in the appropriate places.

When a device designator is specified in the command, QUE uses that device (and not the system device) as a default for subsequent input file names on the physical line. For example, when the following command is executed:

```
*Q LP1:SORT=DK2:FILEA,FILEB
```

QUE sets the device to DK2: for both FILEA.LST and FILEB.LST. Because no device designator is specified for FILEB, DK2: becomes the default device.

To set a priority to a job when queuing a file, specify the /PR:n option with a value between 0 and 255. Without the /PR option, QUE assigns a priority of 128 to the job. The queue management program processes jobs with a priority of 128 or greater before processing jobs less than 128. Only privileged users can specify a priority greater than 128. With such a priority scheme, the queue manager can process important jobs before routine jobs and can delay processing less important jobs until routine jobs are completed. Note that because scheduling is a strict function of priority, a high priority compute-bound job can slow batch processing.

## 20.3 Using the L and S Commands

The L command lists, at your terminal, information about pending requests for a specified device. The following example illustrates the command and its results:

```
*L LP0:  
LP0 QUEUE LISTING      19-Feb-79      04:12 PM  
UNIT   JOB          S / P    FILES  
O      NOTICE[2,201]/SE:1038/FO:NORMAL  
                  SK/128/TY:EMB  
                  SY :[1,2]NOTICE.TXT  
O      BNFRN[2,201]/SE:1039/FO:NORMAL  
                  O /128/TY:EMB  
                  SY :[2,201]BNFRN.SIF  
                  SY :[2,201]BNFBLD.SIF  
                  SY :[2,201]PERCNT.BAS  
O      VISITS[2,201]/SE:1040/FO:NORMAL  
                  O /128/TY:EMB  
                  SY :[2,201]VISITS.LST
```

### NOTE

Because both QUE and QUEMAN can access a queue list simultaneously, QUE tests the linkage of the queue list to verify that the queue list has not been changed by QUEMAN during QUE's search. If QUE detects that the linkage has changed, QUE will display the following message and will restart the listing from the beginning of the queue list.

```
*****QUEUE CHANGING--WILL RESTART LISTING*****
```

The L command causes QUE to print two header lines for the specified spooled device. The first header line contains the device and unit designator of the specified device (LP0: by default), the current system date, and the time of day. The second header line contains headings that denote the unit queued to, the jobname and account number of the requester, and the status, priority, and full file specifications associated with each job. If no jobs are queued, QUE prints:

```
dev: QUEUE IS EMPTY
```

Jobs are listed in the order in which QUEMAN accesses them. In the first line of each job description, the jobname is followed by its account number and the sequence number (/SE:n) assigned the job by QUEMAN. Following the sequence number is the /FO[RMS] option and its argument, the forms name: the default NORMAL or the name you specified.

In the second line of each job description, the status appears under the heading S, and is indicated by one of the following:

- O Job is in queue waiting to be processed.
- S Job has been sent to spooler.
- A Job is waiting for an /AFTER date/time to expire.
- H Job is in hold status: it has been modified (by the M command) to be put into hold, or it was put into hold if, upon restarting, QUEMAN found that the job was previously sent to the spooler.
- K Job is in kill status.

Some status indicators may appear together; for example, SK means that the job was sent to the spooler, and that a K command was later issued for it.

Next on the second line appears the priority (0 to 255), under the heading P; 128 is the default. Following the priority are any job options you specified.

Appearing under the lines describing a job are the descriptions of individual files in that job--the full file specifications followed by any file options you specified.

To list jobs on all line printers, type the L LP: command. QUE prints the related unit number in the UNIT column for each queued job. Jobs queued for the BATCH processor are listed by typing the L BA: command; jobs queued for the RJ2780 program are listed by typing the L RJ: command.

To list only information about one request or several requests, include the = character followed by the jobname(s) in the L command. The following example illustrates:

```
*L LPO:=REW[200,57]
LPO QUEUE LISTING      09-Nov-78      03:22 PM
UNIT    JOB      S / P   FILES
 0      REW    [200,57]/SE:2775/FO:NORMAL
                  S /128/TY:EMB
                           SY :[200,57]EXPENS.BAS
                           SY :[200,57]VISITS.LST/C:3
```

QUE prints the header lines and the data for the jobname specified. If you specify more than one jobname, separate them by commas, as in the following example:

```
*L LPO:=REW[200,57],REF[200,57]
LPO QUEUE LISTING      09-Nov-78      03:22 PM
UNIT    JOB      S / P   FILES
 0      REW    [200,57]/SE:2775/FO:NORMAL
                  A /128/TY:EMB
                           SY :[200,57]EXPENS.BAS
                           SY :[200,57]VISITS.LST/C:3
 0      REF    [200,57]/SE:2776/FO:NORMAL
                  S /128/TY:EMB
                           SY :[200,57]CHOICE.BAS/C:3/N
                           SY :[200,57]CURZ .LST/C:7/D
```

The S command lists, in a shortened form at your terminal, information on pending requests for a specified device. The following example illustrates the command and its result:

```
#S LP0:  
LP0 SHORT QUEUE LISTING 02-Oct-78      11:25 AM  
UNIT    JOB          S / P  
0      NEWV05[1,253]/SE:1374/FO:NORMAL  S /128  
0      RMSTST[2,211]/SE:1375/FO:NORMAL  0 /128  
#^Z
```

Unlike the L command, the S command causes QUE to print a "short queue listing" header line followed by a single line description of each job in queue. The single line description contains the following information:

- Jobname
- Job's account number
- sequence number (/SE:)
- /FORMS option argument (/FO:)
- job status
- job priority
- /AFTER time if specified

## 20.4 Using the K Command

The K command removes a job or jobs from the queue for a specified device. To remove a job, specify the device designator and the = character followed by the jobname or jobnames separated by commas. For example:

```
*K LP1:=PRINT1,BATCH1  
*
```

As a result, each job for your account in the LP1: part of the queue with the name PRINT1 or BATCH1 is deleted if it is not currently being processed by the LP1: spooling program. In the case of matching jobnames, specify the job to be removed from the queue by including the /SE:nnnnn option in the K command. The argument nnnnn represents the job's sequence number, assigned to it by QUEMAN (see Section 20.3).

## 20.5 Using the M Command

The M command allows you to modify the parameters of a job that is already in the queue. By issuing M with appropriate options, you can modify the job parameters originally set in the Q command: priority, mode, date/time, number of job copies, form, and type. If, however, the job has already been sent to a spooling program, QUEMAN will not allow any modifications.

To specify modifications to the job, type the device designator followed by the jobname and the desired modification option(s) in the following format:

M device:jobname/option(s)

Each option must be preceded by a slash (/). The following list contains job modification options that may be specified with the M command. Except where otherwise indicated (for example, with /HOLD and /UNHOLD), these options correspond in name, function, and format with those of the Q command. Their full descriptions are in Table 20-2.

/HOLD	Halt the job's advancement in the queue, and do not send it to a spooling program until the /UNHOLD option is specified.
/UNHOLD	Continue the job's advancement in the queue.
/SE:nnnn	Modify only the job with sequence number nnnn (assigned by QUEMAN; see Section 20.3). Used in case of matching jobnames.
/PRIORITY:nnn	(See Table 20-2 for explanations of this and the remaining options listed here.)
/MODE:nnnn or	
/LENGTH:nnn	/UPPERCASE
/CONVERT	/SKIP
/TRUNCATE	
/LPFORM	
/AFTER:	{ dd-mmm-yy yy.mm.dd +n D[AYS] } { hh:mm hh:mm:am/pm +n H[OURS] }
/TYPE:xxx	
/FORMS:form name	
/JCOPIES:n	

## 20.6 Chaining to QUE from a User Program

To run the QUE program by a CHAIN operation, your program must first put in core common a string in the following format:

Program name	The filename specification of the program to which QUE passes control. The specification can include a project-programmer field but not an extension. This program need not be the calling program.
Delimiter	The delimiter must be CHR\$(13), the CR character.
Line number	An integer number in CVT%\$ format indicating the line number of the program to which QUE passes control. For example, CVT%\$ (28000%).
Command	The string QUE executes. It must have the same syntax as if it were typed at the terminal in response to the # character but must not include CR and LF characters. The option /MORE must not be used.

Delimiter	The delimiter must be CHR\$(13).
Text	Optional string which QUE places in core common when it completes processing and passes control. Its length is restricted by the number of bytes remaining in core common.

Secondly your program must execute a CHAIN "\$QUE" LINE 31000 statement, to actually pass control to QUE stored in the system library account.

Upon completing the request passed to it through core common, QUE places a string in core common and executes a CHAIN "program name" line number statement. The program name and line number are taken from the values passed to QUE in core common. The core common string QUE passes has the following format:

Error number	The QUE error code in CHR\$ format. For example, CHR\$(E%). See Section 20.7 for a description of the error codes generated by QUE. CHR\$ (0%) indicates no error code was generated.
Error message text CHR\$(13%)	If an error occurred, this string will be returned.
Text	The optional string which your program passes to QUE.

## 20.7 Error Messages and Codes

QUE reports an error condition by printing a message or by returning an error code. When running at a terminal, QUE prints a unique message which describes the error condition. No part of the typed command is executed. Retype the entire command in the correct format. Any RSTS/E system errors encountered are reported in the ?ILLEGAL INPUT FILE message. When running by means of a CHAIN operation from a user program, QUE reports an error by an error code when it passes control. Table 20-4 gives both the messages and codes and describes the related error conditions.

**Table 20-4: QUE Error Messages and Codes**

Message and Meaning	Error Code
?INVALID COMMAND-<text> The <text> indicated is an undefined command.	1
?SW ON WRONG SIDE OF COMMAND The option switch specified in the command line to QUE is reserved for the input side and was found on the output side, or vice-versa.	2
?INVALID SWITCH FORMAT The option contains an invalid request.	3

(continued on next page)

**Table 20-4: QUE Error Messages and Codes (Cont.)**

Message and Meaning	Error Code
<b>?UNDEFINED SWITCH</b> The command line contains an undefined option switch.	4
<b>?TOO MANY FILES</b> No more than eleven files can be given in the Q or K command.	5
<b>?NULL FILE SPEC</b> At least one file specification must be given in a Q or K command and none was found; or two commas occurred with no file specification between them.  Not used.	6 7
<b>?ILLEGAL INPUT FILE- &lt;text—filespec&gt;</b> The request is invalid because the file indicated by the file specification caused the RSTS error described in the text. Retype the request.	8
<b>?WILDCARDS NOT ALLOWED IN PPN</b> An asterisk is not allowed in the project-programmer field.	9
<b>?QUEUE FULL</b> The queue file is temporarily full. Try again when the spooling program has processed some pending requests.	10
<b>?BAD SWITCH VALUE-&lt;text&gt;</b> There is an invalid or improper number in the option indicated by <text>. For example, you specified /MODE:A or, being non-privileged, you specified /PR:129 or greater.	11
<b>?'MORE' REQUESTED ON A CHAIN</b> The /MORE option is not allowed when a program runs QUE by a CHAIN operation.	12
<b>?NOT A QUEUABLE DEVICE</b> You attempted to queue a request to a device which cannot handle queued requests. For example, Q LG:=JOB.  Not used.	13 14
<b>?QUEUE NOT INITIALIZED</b> The queue file does not exist in the system library account. A privileged user must run QUEMAN to initialize the queue.	15

(continued on next page)

**Table 20-4: QUE Error Messages and Codes (Cont.)**

Message and Meaning	Error Code
?QUEMAN NOT RUNNING - CAN'T QUE OR KILL You have attempted to queue or kill a job and, because the queue manager is not running on the system, QUE generates an error. You can, however, list jobs when QUEMAN is not running.	16
?QUEUEING DISABLED QUEMAN is in the shutdown process.	17
?ILLEGAL SWITCHES FOR CMD Specified options are illegal for commands used; e.g., /HOLD and /UNHOLD are illegal for Q.	18

## **20.8 Running QUE by CCL Commands**

You can run QUE by means of the Concise Command Language (CCL) command QUEUE, which may be minimally abbreviated to QU. The correct format is QUEUE/<command line> where <command line> is any valid QUE program command. If the QUE program does not find a / character immediately following the characters QUEUE, it defaults the command to a Q command. For example, the command:

```
QUE/Q JOB1 = ABC.DAT
```

has the same effect as the command:

```
QUE JOB1 = ABC.DAT
```

These commands are equivalent since both enter JOB1 in the queue for line printer 0. In either case, the QUE program runs and executes the command. If no error is detected, the system prints the READY message. An error in the command causes QUE to print the related error message on a subsequent line, after which the system prints READY.

To include an output option when queuing a job, begin the QUE command with /Q or else give an explicit jobname in the QUE command. For example:

```
QUE/Q/MODE:2048=DK1:DISK.BAS
READY
```

The / character preceding the Q character differentiates the Q command from the jobname Q.

To list or kill jobs, type QUE, followed by a slash (/) character and the appropriate command. For example:

```
QUE/L DISPLAY[1,253]
LPO QUEUE LISTING      09-Nov-78      03:22 PM
UNIT    JOB           S / P   FILES
O        DISPLAY[1,253]/SE:558/FO:NORMAL
                  0 /128/TY:EMB
SY :[1,253]DISPLAY.VGB
Ready
```

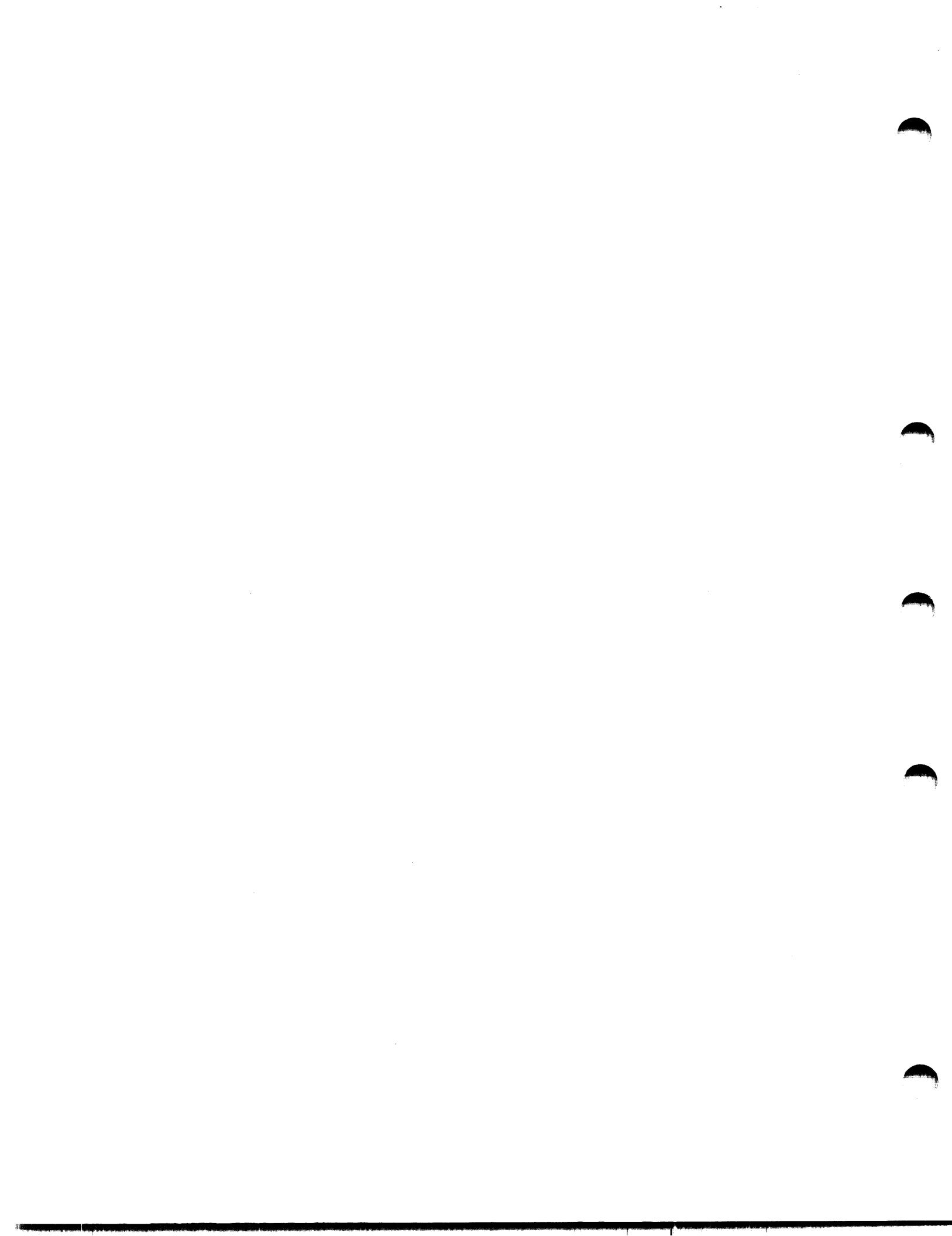
QUE runs and executes the command indicated following the / character, after which control is returned to system command level.

## 20.9 Running QUE at a Logged Out Terminal

The currently pending requests for a spooling program can be printed by typing the QUE/L dev: command. For example, to print the line printer unit 1 queue, type the following command:

```
QUE/L LP1:
```

QUE executes the L (LIST) command for the LP1: queue. After printing the pending requests, QUE exits to RSTS/E and prints BYE.



## **Chapter 21**

# **The Batch Processing Program: BATCH**

The ability of the Batch Processor to execute a stream of commands allows you to submit jobs to be run without terminal dialogue. Because Batch runs from a pseudo keyboard, your terminal is freed for other use during Batch processing. Batch processing is particularly useful in data processing operations that do not require interaction.

Batch input can be submitted from control files created with PIP or an editor on a random access device. For purposes of this description, input is dealt with as though it were on cards, where each card represents one record. Such input consists of elements of the batch control language and is collectively referred to as a batch stream. It is possible to execute multiple streams simultaneously by running multiple copies of the BATCH program. The capability to run more than a single batch stream is controlled by the system manager.

Sections 21.1 through 21.3 discuss the elements of the batch control language. Operating procedures are described in Section 21.4.

### **21.1 Control Statements**

Batch control language statements consist of a command field, specification field(s) and a comment field, in the following format:

```
$[command-field] [specification-field(s)] [!comment]
```

Fields must be separated by one or more spaces and/or tabs.

A command field must always be present and may contain switch modifiers to control or limit the command. When appropriate, the command field is followed by one or more specification fields. The ! character is a comment prefix signifying that the information between the ! character and the line terminating character is a comment. The system takes no action on comment information. Note that the comment character (!) only applies to Batch command

lines. If a comment line is included in data statements, BATCH treats it as data.

The hyphen (-) character can indicate a continuation of a command. If a hyphen is the last character in a command, the next line is treated as a continuation of the previous line and must begin with a \$ followed by a blank. The hyphen must appear before any comment.

A physical command line can have a maximum of 120 characters.

Quotation marks may be used in control statements to reproduce some text identically and override any special interpretation of characters by BATCH. For example, the exclamation point (!) in RSTS/E is the designator for the auxiliary library account [1,3] or for an installation defined account. In BATCH, the exclamation point signifies a comment. To prevent BATCH from misinterpreting the ! character given as an account designator, include quotation marks as shown in the following sample control statement:

```
$RUN " !UPDAT"
```

As a result, BATCH executes the program UPDAT from the auxiliary library account. Without the quotes in the preceding example, the current program is executed and the characters following the ! character are treated as comment characters.

### 21.1.1 Command Field

The command field consists of the following elements:

1. A \$ (dollar sign) character is in the first character position. The \$ character is the control statement recognition character.
2. The command name begins in the second character position and immediately follows the \$ character. For example, \$JOB. No blanks are allowed in the command field.
3. Valid switches follow the command name. No blank can appear between the command name and a switch. Switches are denoted by a slash (/). For example, /NAME=COMPL.
4. A blank (or horizontal tab) delimits the command field.

#### NOTE

Command names and switch names can be shortened to their first three characters. For example, BATCH interprets the command \$BAS as well as \$BASIC.

Multiple, adjacent blank characters are equivalent to one blank character. A horizontal tab is equivalent to one blank character. A blank character delimits a field; otherwise the blank character is ignored.

### **21.1.2 Specification Fields**

A list of specification fields immediately follows the command field delimiter. The following rules apply:

1. Specification fields are separated by blanks.
2. Specification fields are terminated by a ! character if followed by a comment, or are otherwise terminated by a line terminating character.
3. Depending on the command, a specification field consists of a device specification, a file specification, or an arbitrary ASCII string, any of which can be followed by appropriate switches. The / character signals the start of a switch. For example, XYZ.BAS/SOURCE. The switch indicates that the file is a source file.

### **21.1.3 Comments**

The following rules govern comment fields:

1. The start of a comment is defined by an ! character in the control statement.
2. Any character following an ! character and preceding the end-of-line terminator is treated as a comment and is otherwise ignored by the Batch processor. Comment lines with no text may force line spacing on the job log and thereby make the log more readable. To force line spacing, include lines consisting solely of \$! followed immediately by a carriage return/line feed.

### **21.1.4 Syntactical Rules**

The following are syntax rules for control language statements:

1. A control statement must have a command name (except in the case of the comment line "\$!"). If the command name is omitted, the command is ignored. An unrecognizable command name is illegal, and causes the Batch processor to print an error message.

Switches in the command field apply to the entire command. If a switch in the specification field contradicts a command field switch, an error results.

2. An asterisk is allowed in the filename or the file extension field of a file specification, subject to restrictions on individual commands. See Section 21.2 for the description of file specifications. An asterisk can refer only to files already created. An asterisk appearing in a specification of a file not yet created constitutes an invalid file specification.

The Batch processor uses the leftmost 6 characters from filename fields longer than 6 characters and uses the leftmost 3 characters from the file type fields longer than 3 characters.

- Switches can be used in the command field and specification fields of a control statement. Switches appearing in the command field are command qualifiers, and their function applies to the entire command. Switches appearing in specification fields apply only to the field in which they appear.

Unrecognizable switches invalidate the control statements in which they appear.

### 21.1.5 Syntax Example

The following are sample control statements which illustrate the syntax of Batch statements:

```
$JOB/NAME=SMYTHE !FIRST JOB
$!
$!COMPILE OF NEW SOURCE FILES
$!
$MESSAGE STARTING COMPILE
$BASIC XYZ/SOURCE XYZ.LIS/LIST XYZ/EXECUTE
$BASIC ABC/SOURCE ABC.LIS/LIST ABC/EXECUTE
$!
$MESSAGE STARTING LISTING OUTPUT
$!
$PRINT *.LIS! ALL LIST FILES
$!
$EOJ
```

**Table 21-1: BATCH Special Characters**

Character	Meaning
space	Separates fields in a control statement. Otherwise ignored unless embedded in a string delimited by quotation marks.
horizontal tab	Separates fields in a control statement. (Equivalent to one space (blank) character; otherwise ignored.)
hyphen(-)	As last nonblank character in a control statement, indicates that a continuation line follows. If the statement contains a comment, the hyphen must be last nonblank character before the exclamation point.
exclamation point (!)	Indicates a comment unless embedded in a string delimited by quotation marks.
dollar sign (\$)	Used as first character in first position of a control statement; causes control statement recognition.
slash (/)	Denotes a switch (separates specification field from switch name).
asterisk (*)	Indicates wildcard in filename or file type.
colon (:)	Separate switch name from argument.
equals (=)	
quotation marks ("")	Used to open and close a string to preserve embedded spaces or to pass a special character (such as !) without interpretation by BATCH.
single quotation mark ('')	
plus (+)	Indicates file concatenation in \$COPY command.
comma (,)	Separates file, device, and/or account specifications within a specification field which allows multiple elements.

## 21.2 File Specifications

A file specification appears in the specification field and is an alphanumeric string containing the following elements:

`filename.ext`

The Batch processor assigns default values if part or all of the file specification is optionally omitted.

### 21.2.1 Filename Specification

A filename specification is a string of one to six unique alphanumeric characters. An asterisk in place of a filename denotes all files of the specified type in the account designated. If necessary, the Batch processor generates a default filename related to the time of day as described in Section 21.2.3.

### 21.2.2 File Extension Specification

A file extension specification consists of a period, immediately followed by a string containing three unique alphanumeric characters.

The file extension reflects the nature of the file. For example, a BASIC source file has .BAS as its file extension. An asterisk in place of a file extension denotes all file extensions including files with no extensions specified.

Some standard file extensions are listed below:

.CTL	Batch control file.
.DAT	Data file.
.DIR	Directory file.
.BAS	BASIC-PLUS source file.
.LIS	List file.
.BAC	BASIC-PLUS compiled output file.
.OBJ	FORTRAN compiled output file.
.SRT	PDP-11 SORT-11 input, output or listing file.
.MAP	Map file.
.TMP	Temporary file.
.B2S	BASIC-PLUS-2 source file.
.TSK	BASIC-PLUS-2 executable file.
.TSK	BASIC-PLUS-2 task built executable file.
.FOR	FORTRAN source file.
.SAV	FORTRAN linked executable file.

These file extensions are the defaults when no file extension is specified. The default chosen is determined by the current operation and by the type of file expected. Table 21-2 summarizes the default file extensions that apply to particular batch commands.

**Table 21-2: BATCH Commands – Related Default File Extensions**

Command/ Section	Default Extension	Meaning
\$BASIC 21.3.3	.BAS .B2S .BAC .LIS	Input source file default (BASIC-PLUS) (BASIC-PLUS-2). Output executable file default. Listing file default.
\$CREATE 21.3.4.5	.DAT	The file generated as output by CREATE has a file extension of .DAT.
\$DIRECTORY 21.3.4.4	.DIR	The file in which the directory is to be recorded has a file extension of .DIR.
\$FORTRAN 21.3.12	.FOR .OBJ .LST .SAV	Input source file default. Output object file default. Listing file default. Executable linked file.
\$JOB 21.3.1	.CTL .LOG	Batch control file default; assumed when the Batch job is on a file-structured device. Batch output log file default.
\$PRINT 21.3.4.3	.LIS	Default of file to be printed.
\$RUN 21.3.5	.BAC	Default of file to be run.
\$SORT 21.3.11	.SRT	Input or output file default.

### 21.2.3 File Specification Defaults

Defaults are assigned to omitted filename and file extension elements as shown in Table 21-3.

**Table 21-3: File Specification Defaults**

Condition	Default	Example
Name specified but no file extension	Default assigned as appropriate to the current operation. For example, with the BATCH command BASIC, the default is .BAS.	ABC=ABC.type
Name, followed by dot, but no file extension	Default file extension is null. No file extension is assigned.	ABC.=ABC
File extension, but no name	Default filename is related to time of day.	.LIS=B2347P.LIS (created at 01:23:47 PM)
No file specification	Default filename (related to time of day) with default extension as appropriate to current operation.	Null=B2347P.ext

#### **21.2.4 Switch Specification**

Switches consist of a / character followed immediately by a name. If the switch takes an argument, the argument is separated from the switch name by a colon (:) or equal sign (=). If the switch takes an argument and subarguments, each subargument is separated from the argument and from other subarguments by a colon. For example:

```
/NAME=JOB3  
/VID="MY TAPE"
```

Switches accept arguments such as decimal constant, alphanumeric string, and date-time.

Switch values can be negated by putting the characters NO between the / character and the switch name. For example:

```
/NOOBJ
```

This switch indicates that no object file is to be produced. Its most frequent use is in conjunction with the \$BASIC command.

#### **NOTE**

The negation characters NO are not considered part of the switch name. Thus, a negated switch must contain at least five characters. For example:

```
/NOOBJ or /NOBJECT
```

is valid, but

```
/NOO
```

is invalid.

### **21.3 Batch Commands**

The BATCH command set consists of:

\$JOB	Begins a job.
\$EOJ	Ends a job.
\$BASIC	Executes the BASIC-PLUS or BASIC-PLUS-2 compiler.
system command or \$system command	Executes a system utility function.
\$RUN	Executes a program.
\$DATA	Begins data images.
\$EOD	Ends data images.
\$MESSAGE	Logs a message on the operator services console.

\$MOUNT	Assigns a device.
\$DISMOUNT	Deassigns a device.
\$SORT	Executes the PDP-11 Sort program SORT-11.
\$FORTRAN	Executes the FORTRAN compiler.
\$DELETE	Deletes files.
\$COPY	Copies files.
\$PRINT	Queues a file for the default line printer.
\$DIRECTORY	Lists a file directory.
\$CREATE	Creates a file from data in the input stream.

### NOTE

The \$BASIC, \$FORTRAN, and \$SORT commands are subsets of BASIC, FORTRAN, and SORT capabilities. Therefore, to use the full capabilities of BASIC, FORTRAN, or SORT, use the BATCH command \$RUN.

#### 21.3.1 \$JOB

This command marks the beginning of a job. The following command switches are allowed:

/NAME=jobname	Assigns a name to the job. Job names can be up to 6 characters long. This name overrides the control filename as the identifier of the job.
/NONAME	Indicates that no job name is defined. A default job name is assigned. The default job name is the name of the control file. The name appears in all messages to the system operator.
/LIMIT=nnn	Assigns an elapsed time limit to the job. The value of nnn, a decimal number, is interpreted as minutes. Note that the elapsed time taken to execute a job is heavily dependent on overall system loading.
/NOLIMIT	Gives the job an unlimited amount of elapsed time to complete. If neither /LIMIT:nn nor /NOLIMIT appear, the job is given 10 minutes elapsed time to complete execution before BATCH terminates it.
/CPU:nnn	Assigns a CPU time limit to the job. The value of nnn, a decimal number, is interpreted as seconds. If /CPU is specified, and /LIMIT is not specified, no elapsed time limit is enforced, and the only time limit is on CPU time. If both switches are specified, both limits are enforced. If no CPU time limit is specified, the allowable CPU time is infinite.
/NOCPU	Gives the job an unlimited amount of CPU time to complete.
/QUE	Queues the batch log file to LP0 (default) or the device specified at system start up.
/NOQUE	Suppresses printing of the batch log file.
/PRIORITY:n	Sets the RSTS/E job priority to n (or the next lowest multiple of 8) for the BATCH stream. For privileged users, n can be between -120 and +127; for non-privileged users, n is limited to a value between -120 and -8. Unless otherwise altered by the /PRIORITY:n switch, all jobs run at -8 priority.

/CCL	Allows the use of the system's interactive Concise Command Language. When this switch is specified, any of the system commands which do not conflict with existing Batch commands may follow the \$ character. The Batch processor ensures that the job is in the READY state before executing the command.
/ERROR:[operand]	Specifies the level of error which the Batch processor tolerates without terminating the job. The level is indicated by [operand], which may be FAT[AL], WAR[NING], or NON[E]. If FATAL, all errors are tolerated until completion. If WARNING, a fatal error terminates the job, but warning errors are tolerated. If NONE, any error terminates the job. If a job is to be terminated because the error level has been exceeded, termination occurs when the job next asks for input. A message is entered in the log file giving the reason for termination. The default error level for the BATCH stream is determined at start-up time.

#### NOTE

Refer to Table C-1 for the severity standards in error messages. Some programs (for example, the FORTRAN IV compiler, which operates under a run-time system other than BASIC-PLUS) do not use the standard severity characters in error messages. Therefore, compilation errors are not detected by Batch processing. User programs that are coded to run under Batch control must use standard severity characters.

The following specification field may be included:

[n,m]	To have the job executed on an account other than that under which it was queued, a specification field may indicate the account number desired. This feature can be used only by a privileged user.
-------	--

The following error conditions are possible:

- Unrecognized switch
- Illegal switch value
- Multiple conflicting specifications (switches)
- Different account specified by non-privileged user
- Higher priority desired by non-privileged user

#### 21.3.2 \$EOJ

This command marks the end of a job. The \$EOJ command automatically dismounts all devices mounted by the job. \$EOJ prints an appropriate message to the operator that the logical device should be dismounted. A logical deassignment is performed.

#### NOTE

1. The \$EOJ command is implied when BATCH encounters a physical end-of-file condition or another \$JOB control statement while processing a control file.
2. No switches are legal in the \$EOJ command.

### 21.3.3 \$BASIC

The \$BASIC command calls a BASIC compiler, which compiles a source program. The format of the \$BASIC command is:

```
$BASIC[switches][specification field [switch]][specification fields]
```

The following switches are valid in the command field:

- |      |  |
|------|--|
| /BP1 | Use the BASIC-PLUS compiler. If neither /BP1 nor /BP2 appears, /BP1 is used.   |
| /BP2 | Use the BASIC-PLUS-2 compiler. If neither /BP2 nor /BP1 appears, /BP1 is used. |

#### NOTE

When \$BASIC appears in the BATCH command file, the Batch processor assumes that BASIC-PLUS is the default Run-Time System. If this is not the case and the BASIC-PLUS compiler is desired, use the SWITCH program to switch to the BASIC-PLUS Run-Time System prior to the \$BASIC command in the BATCH command file.

/RUN	Execute (only) a previously compiled/task built program. If this switch is used, the entire command line must have one and only one file specification, and can have only one other switch: /EXECUTE.
/NORUN	Perform the compile/task build procedure, but do not execute the final file.
/OBJECT (legal only for BASIC-PLUS-2 runs; see /EXECUTE)	Create the object file filename.OBJ, where the filename is that of the source file. This switch implies /BP2, and therefore causes BASIC-PLUS-2 to be run; it also causes a task build operation.
/NOBJECT	Create the object file filename.TMP, where the filename is that of the source file. Delete this .TMP (temporary) file upon completing the command. This switch implies /BP2, and therefore causes BASIC-PLUS-2 to be run. Thus, like /OBJECT, it is legal only in a BASIC-PLUS-2 run.
/LIST	Produce the listing file filename.LST, where the filename is that of the source file. If neither /LIST nor /NOLIST appears, /NOLIST is used.
/NOLIST	Do not produce a listing file. If neither /NOLIST nor /LIST appears, /NOLIST is used.
/MAP	Create the task builder map file filename.MAP, where the filename is that of the source file. This switch implies /BP2, and therefore causes BASIC-PLUS-2 to be run; it also causes a task build operation. Thus, it is legal only in a BASIC-PLUS-2 run. If neither /MAP nor /NOMAP appears, /NOMAP is used.
/NOMAP	Do not create a map file. This switch implies /BP2, and therefore causes BASIC-PLUS-2 to be run. Thus, it is legal only in a BASIC-PLUS-2 run. If neither /MAP nor /NOMAP appears, /NOMAP is used.

/EXECUTE (replaces /OBJECT for BASIC-PLUS runs)	Create an executable file, whose filename is that of the source file. Choose its type according to the following rules:  If the language is BASIC-PLUS, give the file a .BAC type (filename.BAC).  If the language is BASIC-PLUS-2, give the file a .TSK type (filename.TSK).
/NOEXECUTE	If an executable file is needed, create a temporary one (.TMP), and delete it upon completion of \$BASIC processing.

One of the following switches may appear in the first specification field, described in the format guide at the start of this section:

/SOURCE /BASIC	Both switches have the same meaning: i.e., that this is the BASIC-PLUS or BASIC-PLUS-2 source file on which to operate.
/EXECUTE	This switch is legal only if /RUN appears in the command field, and means that this is an executable file.

In this field, any specification lacking a switch is assumed to be the input file for the command. Thus, only one file specification may appear without switches. This specification may not contain wildcards (neither asterisks nor question marks). If /NORUN appears in the command field, the lack of a switch here implies /BASIC. If /RUN appears in the command field, the lack of a switch here implies /EXECUTE. And if neither /RUN nor /NORUN appears in the command field, the lack of a switch implies /BASIC.

The optional specifications ending the format description (at the beginning of this section) define other files which may be needed in the operation. Any of the following switches may be used in the formats indicated. Each switch, however, may be used only once in the entire \$BASIC command line. Moreover, its negation cannot be used anywhere in the command line (/NOLIST and /LIST, for example, cannot appear together in a command line).

file specification/LIST	Define the listing file as specified.
file specification/OBJECT	Define the object file as specified (switch implies /BP2 and causes BASIC-PLUS-2 to be run, with task build; switch is legal only with BASIC-PLUS-2).
file specification/MAP	Define the map file as specified (switch implies /BP2 and causes BASIC-PLUS-2 to be run, with task build; switch is legal only with BASIC-PLUS-2).
file specification/EXECUTE	Define the executable file as specified.

If a source file is not specified, the \$BASIC command must be followed by a set of BASIC source statements, terminated by either \$EOD (see Section 21.3.7) or some other recognized Batch control statement. For example:

```
$BAS      LISTING/LIS
      BASIC
      Source
      Deck
$EOD
```

If a source file is explicitly specified, any source statements following this command are appended to the source program. Source statements following this command and having line numbers equal to those in the source program replace those in the source program. Source input must be provided, either through a file specification, or through source statements, or both.

If no listing file is specified, but the /LIST switch is present, the Batch processor creates the default listing file. If a file specification appears with the /LIST switch, the Batch processor uses that specification for the file. To print the file specified as part of the Batch job, supply a \$PRINT control statement described in Section 21.3.4.3.

If no executable file is specified with the /EXECUTE switch, a default executable file is created and is deleted after job completion. If an executable file is explicitly specified, it is preserved after job execution. Errors result from conflicting switch specifications such as both /BASIC and /SOURCE on different specification fields.

The default applied when a file is specified without a switch is /SOURCE.

The following error conditions are possible:

Unrecognized switch

Multiple conflicting specifications (switches)

File specification syntax error

#### 21.3.4 Utility BATCH Commands

The following utility functions are provided by the RSTS/E Batch processor:

\$DELETE	Deletes files.
\$COPY	Copies files.
\$PRINT	Prints a file on the system line printer by means of the line printer spooling program SPOOL.
\$DIRECTORY	Lists a file directory.
\$CREATE	Creates a file from data in the input stream.

**21.3.4.1 \$DELETE** — The \$DELETE command is used to delete specified files. It is issued in the following format:

```
$DELETE file1 [file2 ... filen]
```

The filename and file extension must be included. An asterisk is not valid in either the filename or the file extension field.

No switches are used with \$DELETE.

The following error conditions are possible:

No file specification

Syntax error in file specification

**21.3.4.2 \$COPY** — \$COPY is used to copy files. Use of the asterisk character in the file specification is invalid. The following are the valid switches:

/OUTPUT For new files to be created.

/INPUT For files to be copied.

The following is an example of the \$COPY command:

```
$COPY TER.LIS/OUTPUT TERRY.LIS/INPUT
```

The \$COPY command supports the use of + (plus sign) to indicate file concatenation. When used with \$COPY, file concatenation results in the creation of a single file, which consists of files connected together. The + character appears in the file specification field, between the specifications of files to be concatenated. If no switch is specified, /INPUT is assumed.

The following error conditions are possible:

No output specification

No input specification

Multiple conflicting specifications

Syntax error in file description

**21.3.4.3 \$PRINT** — The \$PRINT command prints the contents of files on the system line printer by means of the spooling program SPOOL. File specifications accept all switches available in the Q command (see Section 20.2). Asterisks can be used in file specifications. The \$PRINT command is issued in the following format:

```
$PRINT file1/switches [file2 ... filen]
```

The specification field contains the file or files to be printed.

The following error conditions are possible:

No file specification

Syntax error in file specification

**21.3.4.4 \$DIRECTORY** — \$DIRECTORY produces a directory listing of the file(s) in the specified account and is issued in the following format:

```
$DIRECTORY [specification field]
```

The specification field can contain file specifications. If no file specification appears, the \$DIR command lists the contents of the current account on the Batch log device. A file specification indicates the directory of a file or set of files and can contain an asterisk in either the filename field or file extension field. For example:

```
$DIR *.BAS
```

This command creates a directory listing of all files in the current account with the .BAS extension.

To create a directory in a disk file rather than on the Batch log device, specify a file and the /DIRECTORY switch. For example:

```
$DIR BAJOB.DIR/DIR
```

creates the directory listing in a file BAJOB.DIR on the system disk under the current account.

To create a directory in a disk file and to designate which files are to be listed, specify both the /DIRECTORY and /INPUT switches with the related file specification. For example:

```
$DIR BA.DIR/DIR *.BAC/INPUT
```

The \$DIR command in this example creates a directory listing of all compiled BASIC-PLUS files and stores the listing in the file BA.DIR on the system disk under the current account.

The following error conditions are possible:

Syntax error in file specification  
Multiple conflicting specifications

**21.3.4.5 \$CREATE** — The \$CREATE command creates an ASCII file as indicated in the specification field. The file consists of the data images following the \$CREATE command in the input stream. Data images must follow \$CREATE and must be terminated by \$EOD, or an error occurs. The data images must not be preceded by any other command because the \$CREATE function terminates on encountering a \$ in the first column of a data image.

Any previously existing file of the name specified is deleted at batch execution time, and replaced by the file created by the \$CREATE command.

The \$CREATE command has the following format:

```
$CREATE file
```

The following error conditions are possible:

Syntax error in file specification  
No filename specified  
Non-comment characters following file specification

### 21.3.5 \$RUN

The \$RUN command causes execution of system programs. For example, to run PIP, type:

```
$RUN $PIP
```

followed by appropriate PIP commands. The PIP program reads the commands as data images in the input stream. Execution of PIP is terminated when the next Batch control statement is read.

No switches can be specified. The general format of \$RUN is:

```
$RUN [file]
```

where file specifies the executable program. If file is omitted, the default current program is used.

The following error conditions are possible:

Syntax error in file specification

Non-comment characters following file specification

#### **21.3.6 \$DATA**

The \$DATA command provides a means of entering data to a program that is compiled and run by one of the language commands (e.g., \$BASIC, \$FORTRAN). \$DATA ensures that the program will be run, unless the /NORUN switch was specified. It also ensures that if the program does not use all of its data, the remaining data will be flushed from the stream.

The \$DATA command is issued without specification fields or switches, in the following format:

```
$DATA
```

#### **21.3.7 \$EOD**

The \$EOD command marks the end of data records included in the input stream following commands such as \$BASIC, \$CREATE, \$DATA, and \$RUN. For example:

```
$DATA  
:  
:  
:  
data  
:  
:  
:  
$EOD
```

#### **21.3.8 \$MESSAGE**

The \$MESSAGE command logs a message on the operator services console. It provides a way for the job to communicate with the operator. The command is issued in the following format:

```
$MESSAGE[/WAIT] message-string
```

The /WAIT switch can be used in the command field to indicate a pause to wait for operator action. The system pauses until the operator gives the appropriate command. For example, the following command halts the program until the operator takes action:

```
$MESSAGE/WAIT MOUNT SCRATCH TAPE ON DTO:
```

The WAIT condition remains in effect until the operator responds to the message on the operator services console. For information on operator response procedures, refer to the *RSTS/E System Manager's Guide*.

### 21.3.9 \$MOUNT

The \$MOUNT command causes a mount message to be printed on the control console, and effects a logical to physical device assignment. The physical device refers to a physical device type. The operator responds with the device and unit number in the standard format (e.g., MT1:). An automatic /WAIT occurs. Logical device names of up to six characters are used to specify logical devices.

The \$MOUNT command is issued in the following format:

```
$MOUNT devn:[/switch] devm:[/switch]
```

Both the logical device and the physical device must be specified. The colon is required as the terminator for each device specification. The following switches can be used for the physical device:

/PHYSICAL	Identifies the device specification to be the physical device (default).
/WRITE	Tells the operator to write-enable the device (or volume).
/NOWRITE	Tells the operator to write-protect the volume.
/VID:[string]	[string] is a visual identification which identifies the volume for the operator.
/DEN[SITY]:nnn	Specifies density for magnetic tape.
/PAR[ITY]:[ODD][EVEN]	Specifies odd or even parity for magnetic tape.

The following switch is used with the logical device:

/LOGICAL	Identifies the device specification to be the logical device name; this specification must correspond to the PACK ID for RSTS/E disks.
----------	--

The /VID switch on the physical device field is used to specify the volume identification. The value associated with /VID is the name physically attached to the volume. It is included to help the operator locate the volume.

If the name specified with /VID must contain blanks, it can be delimited by quotes. The following example illustrates:

```
/VID="FJM JT"
```

No blanks are allowed in a string not delimited by quotes. For example:

```
$MOU MT:/PHY/VID="MY TAPE" TAPE:/LOG
```

In this example, logical device name TAPE is assigned to a magnetic tape unit. The operator is told that the reel of tape to be physically mounted is labeled MY TAPE. The operator uses a PLEASE command (see the *RSTS/E System Manager's Guide*) to respond with the device and unit number on

which the tape is mounted. Thereafter, in the Batch command file, reference to the device TAPE: accesses the physical device on which the operator mounted the reel MY TAPE. If the physical device is a removable disk pack or cartridge, the logical device name must be the pack identification. The Batch processor logically mounts and unlocks private disks which the operator mounts as a result of \$MOUNT.

The valid physical devices that can be requested for mounting are:

CR: Card Reader  
DK: RK11/RK05 Disk Cartridge  
DP: RP11/RP03/RP02 disk pack  
DB: RH11/RP04/RP05/RP06 disk pack  
DM: RK611/RK06/RK07 disk pack  
DR: RM02/RM03 disk pack  
DF: RF11 disk  
DS: RH11/RS03/RS04 disk  
DL: RL01/RL02 disk  
DX: RX01/RX02 flexible diskette  
DT: TU56 DECTape  
DD: TU58 DECTape II  
LP: Line printer  
MT: TE10/TU10/TS03 magnetic tape  
MM: TE16/TU16/TU77/TU45 magnetic tape  
MS: TS11 magnetic tape  
PP: Paper tape punch  
PR: Paper tape reader  
SY: System device  
KB: Teletype (or terminal)

The following error conditions are possible:

Syntax error in device specification fields  
Invalid device name/unit  
Invalid logical device name specifications  
Unit number already assigned  
Both physical and logical names have not been specified

#### 21.3.10 \$DISMOUNT

The \$DISMOUNT command causes the logical to physical device assignment effected by the \$MOUNT command to be nullified. It also prints an operator message, requesting that the volume be dismounted. If a /WAIT switch is included in the command field, the job will not resume until a response, as with the \$MESSAGE command, is received from the operator. For example:

```
$DIS/WAI TAPE:
```

sends a message to the operator to dismount the magnetic tape that was mounted by the example in Section 21.3.9. As a result of the switch /WAI, BATCH pauses until the operator responds.

All devices are automatically dismounted at end-of-job (\$EOJ).

The following error conditions are possible:

- Syntax error in specification field
- Illegal switches
- Logical device not assigned

### 21.3.11 \$SORT

The \$SORT control statement causes the execution of the SORT-11 program, which is supplied on RSTS/E systems. For additional information on the SORT-11 program, refer to the *PDP-11 SORT Reference Manual*.

The format of the \$SORT control statement is as follows:

```
$SORT [job switches] [output [/OUTPUT]] [input [/INPUT]] [spec /SPEC]
```

Job switches define the sort process and can be abbreviated to the first three letters. When more than one switch is specified, they are separated by slashes (/). The following are valid job switches:

/ALL[OCATION]:n	For output, specifies the initial space allocation for the file. The value n is in the range of 0 to 65535 blocks. If no allocation is specified, the default depends on the sort process (see /PROCESS). If /PROCESS is Record or Tag, the default is the input file size. If /PROCESS is Index or Address Routing, the default is the number of records sorted.
/BLO[CKSIZE]:n	For magnetic tape input or output, specifies the tape block size. The default is a 512-byte block.
/BUC[KETSIZE]:n	For output, specifies the file's bucketsize. The default is the bucketsize of the input file.
/CON[TIGUOUS]	For output, specifies that the file will be contiguous. The default is non-contiguous.
/DEV[ICE]:x	For input, specifies the device to be used for scratch files, where x is a 1- to 4-character device name.
/FIL[ES]:n	For input, specifies the number of scratch files, where n is in the range of 3 to 10.
/FOR[MAT]:x:n	For input, specifies the file's record format (x) and maximum record size (n). The value x can be FIXED, STREAM, VARIABLE, or UNKNOWN and can be abbreviated to the first letter. This switch is required. The default format is VARIABLE; the record size must be specified.
/IND[EXEDSEQUENTIAL]:x	For input, specifies indexed file organization. The value x specifies the number of keys.
/KEY[S]:abm.n	For input, specifies the sorting key field. This switch is required if no specification file appears in the command line. A maximum of 10 keys (separated by colons) can be specified as follows:  /KEY: abm.n: abm.n: ... abm.n

/PRO[CESS]:x	For input, specifies the sorting process. The value x can be R (Record sort, the default), T (Tag sort), A (Address Routing), or I (Index sort).
/REL[ATIVE]	For output, specifies relative file organization.
/SEQ[UENTIAL]	For output, specifies sequential file organization.
/SIZ[E]:n	For output, specifies the file's cluster size.

A maximum of three file specifications (separated by spaces) can appear in the \$SORT control statement; an input file, an output file, and a specification file. To distinguish these files, the following switches, which can be abbreviated to the first three letters, are used:

/INP[UT]	The file to be sorted.
/OUT[PUT]	The file to contain the sorted data.
/SPE[CIFICATION]	The file which contains the control information for the sorting process.

A file specification without a switch is used as the file to be sorted. If the /SPECIFICATION switch is used, the /KEYS and /PROCESS switches must not appear in the command line. If a specification file is not given in the control statement, the /KEYS switch must be included in the command field to control the sorting process; the /FORMAT switch must always be included in the command field. If an extension is omitted from the file specification, BATCH uses .SRT as the extension.

### 21.3.12 \$FORTRAN

The \$FORTRAN command calls the FORTRAN Compiler, which compiles the source program and generates an object program. The format of the command is:

```
$FORTRAN[switches][specification field[switch]][spec fields[switch]]
```

The following switches are valid in the command field:

/RUN	Execute the previously compiled file. Only an object file can be specified.
/NORUN	Compile the source program but do not execute the object file.
/OBJECT	Create the compiled file filename.OBJ, where the filename is that of the source file. If neither /OBJECT nor /NOBJECT appears, /NOBJECT is used.
/NOBJECT	Do not create an object file. If neither /NOBJECT nor /OBJECT appears, /NOBJECT is used.
/LIST	Produce the listing file filename.LST, where the filename is that of the source file. If neither /LIST nor /NOLIST appears, /NOLIST is used.
/NOLIST	Do not produce a listing file. If neither /NOLIST nor /LIST appears, /NOLIST is used.

/MAP	Create the map file filename.MAP, where the filename is that of the source file. If neither /MAP nor /NOMAP is specified, /NOMAP is used.
/NOMAP	Do not create a map file. If neither /MAP nor /NOMAP is specified, /NOMAP is used.

One of the following switches may appear in the first specification field described in the format guide at the start of this section:

/FORTRAN /SOURCE	Both switches have the same meaning: i.e., that this is the source file on which to operate. If a file specification lacks a switch, it is assumed to be a source file.
---------------------	---

The optional specifications ending the format description define other files which may be needed in the operation. Any of the following switches may be used in the formats indicated. Each switch, however, may be used only once in the entire \$FORTRAN command line. Moreover, its negation cannot be used anywhere in the command line (/NOLIST and /LIST, for example, cannot appear together in a command line).

- file specification/LIST Define the listing file as specified.
- file specification/OBJECT Define the object file as specified.
- file specification/MAP Define the map file as specified.

Multiple specifications of the following form may appear anywhere in the command line, delimited by spaces:

file specification/LIBRARY

Each such specification is a library file that will be linked with the FORTRAN program.

## 21.4 Batch Operating Procedures

This section describes how you request Batch processing and how the Batch processor generates output.

### 21.4.1 Requesting a Batch Job Run

To request the running of a BATCH job, run the library program QUE and specify the Batch control file or files as follows:

```
RUN $QUE
QUE V7.0 RSTS V7.0 Timesharing
#Q BA:BATJOB=FILE1,FILE2,FILE3.DAT
#
```

You normally queue a BATCH job to device BA:. The job and log files in this example will be named BATJOB, and the files FILE1.CTL, FILE2.CTL, and FILE3.DAT will be concatenated to form the Batch control file. The log file

BATJOB.LOG will be printed after the job is complete. Note that QUE schedules jobs on a strict priority basis (see Section 20.2), thus a high priority compute bound job can slow processing.

The CCL command QUEUE (see Section 20.8), if available on the system, may also be used to submit a job for Batch processing. Also, the CCL command SUBMIT, if available on the system, can be used. However, unlike QUEUE, the SUBMIT CCL command only accepts filename specifications (separated by commas) and the Q command switches /DE and /MORE as described in Table 20-3.

#### 21.4.2 Batch Processing

As the Batch control file is read, it is checked for command sequence and syntactical validity. If an error is detected, an error message is printed in the log file. The job will not be run, but syntax checking will continue through the remainder of the file(s).

A \$MESSAGE/WAIT, a \$MOUNT, or a \$DISMOUNT/WAIT will cause the job to pause for an operator response. Until the operator takes action, no further commands will be sent to the pseudo keyboard.

If no errors are detected, the job is processed. A log is created, showing the sequence of Batch commands processed during the course of the job. If program output is directed to KB:, this output appears following the command that caused the program to execute. In the example that follows, a BATCH job named JOB1 has been run. The Batch control file contained the following sequence of commands:

```
$JOB/NAME=JOB1/LIMIT=4  
$CREATE SUB1.BAS  
  
source statements  
  
$EOD  
$BASIC/BP2 LISTING/LIS MAIN/OBJ  
  
source statements  
  
$DATA  
  
data  
  
$PRINT SUB1.BAS  
$EOJ
```

These commands have the following effect:

```
$JOB/NAME=JOB1/LIMIT=4
```

A job name of JOB1 is assigned to the job. This name appears on the job log along with the time and date of the job's execution. A time limit of four

minutes is set. If the job is not finished in four minutes from its start (actual elapsed time), the job is terminated, and the appropriate error message is printed in the log.

```
$CREATE SUB1.BAS
```

A BASIC source file named SUB1 is created from data records which must follow the \$CREATE command.

```
$EOD
```

The \$EOD command signals the end of SUB1.BAS.

```
$BASIC/BP2 LISTING/LIS MAIN/OBJ
```

The source statements following this command are compiled by the BASIC-PLUS-2 Compiler. A listing of the source statements is created in the file LISTIN.B2S, and the object data is placed in the file MAIN.OBJ. The temporary task built file has the extension .EXE. This file is executed.

```
$DATA
```

The data to be read during execution of MAIN.BAC follows this command.

```
$PRINT SUB1.BAS
```

The source file created by \$CREATE SUB1.BAS is printed. This command also has the effect of terminating data input to MAIN.BAC.

```
$E0J
```

This command signals the end of job JOB1.

#### 21.4.3 Error Procedures

When a syntax error is detected in a BATCH command, the job is not executed. Instead, an error log is printed listing all commands and data scanned along with the appropriate error message(s). The Batch log file always indicates all command lines scanned. If an error is found on a command line, the error message follows the command, marked with question marks (?????????????). Scanning of the control file continues, but the job will not be executed.

If no syntax errors occur, the time of output of lines will be indicated in the left margin of the log. All normal terminal interaction corresponding to the BATCH commands will appear in the log. Table 21-4 lists the BATCH error messages and their meanings.

**Table 21-4: Summary of BATCH Error Messages**

Message and Meaning
<b>BATCH BEING SHUT DOWN</b> The BATCH processor is going off-line and the job must be terminated.
<b>CANNOT USE THAT ACCOUNT</b> An account specification appeared in the \$JOB command but the request did not come from a privileged user.
<b>CANNOT INCREASE PRIORITY</b> A /PRIORITY:n switch appeared in the \$JOB command. The user was privileged but specified a value for n greater than 127. Or, the user was non-privileged and specified a value greater than -8.
<b>CONTINUATION MISSING</b> The hyphen (-) character was the last nonblank character in a control statement to continue the statement on the next line, but the following line did not begin with a dollar sign (\$) and a blank.
<b>DEVICE NOT MOUNTED</b> A \$DISMOUNT command was present but the device indicated had not been mounted.
<b>DISK MOUNT FAILURE</b> The volume to be mounted was not correct (pack IDs did not match) or the device was in use by another job.
<b>INVALID COMMAND</b> An undefined command name followed the \$ character in a statement but the /CCL command switch had not been specified in the \$JOB command.
<b>INVALID SPECIFICATION FIELD</b> The specification given in a control statement is in the wrong format.
<b>INVALID SWITCH</b> The switch used in the command field or in the specification field is undefined, in the wrong format, or is privileged.
<b>NO BATCH JOBS POSSIBLE AT THIS TIME</b> The Batch processor requires a pseudo keyboard to execute a job but one is not available. Requeue your request.
<b>NO SUCH ACCOUNT</b> The account specified in the \$JOB command or in a specification field could not be found on the device.
<b>SEQUENCE NOT SUPPORTED YET</b> The \$SEQUENCE command is not available with this version of BATCH.

(continued on next page)

**Table 21-4: Summary of BATCH Error Messages (Cont.)**

<b>Message and Meaning</b>
<b>TIME LIMIT EXCEEDED</b> Time specified in \$JOB command is insufficient to execute the job. Specify a larger limit by using /LIMIT=nnn or /NOLIMIT switch.
<b>TOO MANY MOUNTED DEVICES</b> The job has requested mounting of more devices than the maximum (12) allowed by Batch.
<b>UNABLE TO LOG IN BATCH JOB</b> To execute a request, the Batch processor logs a job into the system using the account under which the job was queued or the account specified in the \$JOB command. For some reason, the login procedure failed. For example, logins had been disabled. The job will be requeued for later execution.
<b>UNMATCHED PARENTHESES</b> An opening left parenthesis appears in a specification field but an accompanying closing parenthesis is not found.
<b>UNMATCHED QUOTATION MARKS</b> Quotation marks (and single quotation marks) must be paired in a control statement.

## **Chapter 22**

### **Formatting a Post-Mortem Dump: PMDUMP**

PMDUMP is a system library program that formats the contents of a post-mortem dump into human-readable form and allows you to copy it to a file on any RSTS/E device. The post-mortem dump is written in binary by the RSX Run-Time System or an RSX-based run-time system such as BASIC-PLUS-2. The information contained in the dump allows a system programmer or software specialist to view the actual state of the program at the time it aborted. When you invoke the PMDUMP program, it performs the following operations and writes the information to a user-specified output file:

1. Formats the binary dump and produces a human-readable dump of the contents of low-core memory.
2. Produces an octal dump of the user's task.

When the program completes its operation, it returns control to RSTS/E command level.

#### **22.1 When to Use PMDUMP**

Post-Mortem dumps are generated under control of the RSX Run-Time System and are used to determine the state of a task after that task has aborted. Post-Mortem dumps do not return an on-line snapshot of the state of a job. However, the following conditions must be true for PMDUMP to be successful in formatting the dump:

1. The Run-Time System that controls your job must be RSX or RSX-based.
2. The task to be dumped must have already aborted, i.e., terminated abnormally.
3. A request for a post-mortem dump must be specified when the program is input to the Task Builder, i.e., the /PM switch must be used as described in the *RSTS/E Task Builder Reference Manual*.

## 22.2 How to Invoke PMDUMP

To invoke PMDUMP, type:

RUN \$PMDUMP

or a CCL command, if one is installed.

If you invoke PMDUMP with the RUN command, the program prints a header line followed by a prompt (#) to indicate its readiness to accept command input.

In response to the prompt, type one of the following:

RETURN or LINE FEED	accepts a complete default.
CTRL/Z	terminates PMDUMP.
outfil.ext=infil.ext	specifies the input and output files to be used by the program.

If you do not specify an input file, the program defaults the filename to the following:

PMDnnn.PMD

Where nnn is the current job number. Note that this filename construct is identical to that of the dump file created by the RSX Run-Time System.

If you do not specify an output file, the program defaults the filename to that specified or defaulted for input. However, the program substitutes the extension .LST for the input extension of .PMD.

The program defaults for all elements of the file specification are as follows:

device	SY:
account	the current account
filename	the input filename
extension	.LST for output, .PMD for input

If you type the RETURN key in response to the prompt, the following defaults are applied:

1. The program places the files in the current account on the public disk structure (SY:).
2. The input filename is PMDnnn.PMD.
3. The output filename is PMDnnn.LST.

## 22.3 Contents of the Post-Mortem Dump

The following description of the contents of the PMDUMP listing is keyed to the example in Figure 22-1.

Item	Description
1	The name and account of the task being dumped and the date and time that the dump was generated. This information prints as a header line on each page of the dump listing.
2	The program name, or the name you assigned in the Task Builder TASK option, for the task being dumped. Partition name is GEN. or is the name you specified in the Task Builder PAR option.
3	The size of the task in octal and decimal words. The size is given for the task when it was initially loaded and also for when the dump was made.
4	The synchronous and asynchronous system traps in effect when the task was dumped.
5	The task status flags that were set at the time of the dump.
6	The entry point of the program, either by line number in Parameter Entry or by CCL command. CCL Entry also indicates whether a /DETACH or /SIZE switch is specified. Directive Status and Impure Area Pointers are used by the RSX emulation code.
7	The logical unit assignments in effect at the time of the dump. This information includes the device name, the status flags set for that device, and the allocated buffer space in octal and decimal words.
8	A breakdown of the task's keyword value. The last two octal characters of the value represent those set by the user program. The other flags are set by the run-time system and by the RSTS/E Monitor.
9	The contents; in octal, of the file request queue block (FIRQB) and the transfer request block (XRB). This information represents the Monitor requests for file and I/O operations at the time of the dump.
10	The contents of the job's core common area when the dump was made, in ASCII characters and octal byte values.
11	The assigned project-programmer number, default protection code, and logical name table entries assigned by the user job at the time the dump was made.
12	The program counter and processor status word at the time the task was loaded. This information represents initial conditions.
13	The general registers, stack pointer, and processor status at the time of the dump (contrast with item 12).
14	The task stack at the time of the dump.
15	An octal dump of the task. The first 1000 octal bytes reflect the status printed in the previous portion of the listing.  The information shown from octal byte 1000 to the value shown in the initial stack pointer (see item 13) represents the current contents of the stack region. In this example, the contents of the used portion of the stack are shown in the region from octal byte 1760 to octal byte 1777. That is, octal byte 001760 contains the value 003770, which is the stack pointer value shown in item 14. Note that the PDP-11 stack starts at the high address and works toward the lower address.  The octal bytes following the stack display the task image at the time of the dump. This display represents the state of the program at the time the abort occurred. It is especially useful for software specialists.

### Figure 22-1: Post-Mortem Dump

① Post-Mortem Dump of SY:[1,201]PMD030.PMD on 20-Jul-78 at 09:15 AM  
Formatted Dump of Low Memory

② Task Name : FOO  
Partition Name : GEN

Task Size (without Extension) : 002040 (1056.) words

③ Load Size of Task : 004000 (2048.) words  
Current Task Size : 004000 (2048.) words

ODT SST Vector Address : 000000  
ODT SST Vector Length : 000 (0.)

④ Task SST Vector Address : 000000  
Task SST Vector Length : 000 (0.)  
FPP AST Service Address : 100124  
CTRL/C AST Service Address : 000000

⑤ Task Flags : 010000 (4096.) Task Flags Set :  
Post-Mortem Dump Requested

User Parameter on Entry : 000000  
CCL Entry Flags : 000000  
Directive Status Word : 000001

⑥ FCS Impure Area Pointer : 000000  
OTS Impure Area Pointer : 002372  
Auto-Load Impure Area Pointer : 004044  
Extended Impure Area Pointer : 000000

⑦ LUN (Logical Unit Number) Table :

LUN #0 :  
Device Name : TI:  
Device Flags : 000007 Flags are :  
Record Oriented Device  
Carriage Control Device  
Terminal Device  
Buffer Size : 000200 (128.)

LUN #1 :  
Device Name : SY:  
Device Flags : 000010 Flags are :  
Directory Device  
Buffer Size : 001000 (512.)

LUN #2 :  
Device Name : SY:  
Device Flags : 000010 Flags are :  
Directory Device  
Buffer Size : 001000 (512.)

LUN #3 :  
Device Name : SY:  
Device Flags : 000010 Flags are :  
Directory Device  
Buffer Size : 001000 (512.)

LUN #4 :  
Device Name : SY:  
Device Flags : 000010 Flags are :  
Directory Device  
Buffer Size : 001000 (512.)

```
LUN #5 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #6 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #7 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #8 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #9 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #10 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #11 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)  
  
LUN #12 :  
    Device Name : SY:  
    Device Flags : 000010  Flags are :  
                           Directory Device  
    Buffer Size : 001000 (512.)
```

⑧ Job Keyword :  
 Value : 103300  
  
User Controlled : 00  
  
Run-Time System Controlled :  
 Post-Mortem Dump Enabled  
 Traps Enabled  
 No Temporary Privileges  
  
System Controlled :  
 JFFPP  
 JFPRIIV

⑨ Job's FIRQB :  
I0STS: 000 (0.)

FIRQB:  
000000  
010074  
000034  
000401  
006273  
034330  
021042  
000021  
001000  
000000  
100000  
024010  
041104  
176001  
013560  
000000

Job's XRB :  
000002  
000020  
000020  
000020  
000711  
177770  
000001  
000000  
000000

⑩ Job's Core Common :  
Length : 8  
T K B @ F O O  
124 113 102 040 100 106 117 117

⑪ User Assignable PPN : <None>  
User Default Protection : <None>  
User Logical Name Table :  
<Table Empty>

⑫ Initial PC : 002000  
Initial PS : 174017

⑬ Registers at time of DUMP :  
R0 000040  
R1 004020  
R2 121030  
R3 002372  
R4 002112  
R5 001774  
SP 001762  
PC 120150  
PS 000000

(14) Job's Stack :

Initial SP = 002000 Final SP = 001762

SP

!

V

001762 / 000000 121036 174000 101744 002112 000001 003770 004437

(15) Post-Mortem Dump of SY:[1,201]PMD030.PMD on 20-Jul-78 at 09:15 AM  
Memory Dump, in Octal

	00	02	04	06	10	12	14	16	
000000	/	023747	000000	026226	000000	000041	000000	000000	000000
000020	/	000000	100124	000000	010000	000000	000000	000100	000100
000040	/	000014	000000	000000	000001	000000	002372	004044	000000
000060	/	176347	002620	000450	000700	000000	010000	000170	166214
000100	/	174000	001740	003400	000000	000000	017112	112561	064143
000120	/	146264	023032	115635	061043	066063	013032	110664	061063
000140	/	106063	053072	115704	061063	006063	017112	112561	064243
000160	/	146264	023032	115635	061043	066063	000000	000000	000000
000200	/	044524	177777	000007	000200	054523	177777	000010	001000
000220	/	054523	177777	000010	001000	054523	177777	000010	001000
000240	/	054523	177777	000010	001000	054523	177777	000010	001000
000260	/	054523	177777	000010	001000	054523	177777	000010	001000
000300	/	054523	177777	000010	001000	054523	177777	000010	001000
000320	/	054523	177777	000010	001000	054523	177777	000010	001000
000340	/	054523	177777	000010	001000	000000	000000	000000	000000
000360	/	000000	000000	000000	054523	177777	000010	001000	

	00	02	04	06	10	12	14	16	
000400	/	103300	000000	010074	000034	000401	006273	034330	021042
000420	/	000021	001000	000000	100000	024010	041104	176001	013560
000440	/	000000	010000	010000	000000	000036	000000	000000	000000
000460	/	052010	041113	040040	047506	000117	042057	046525	027520
000500	/	047516	040515	000120	000000	000000	000000	000000	000000
000020	/	000000	000000	000000	000000	000000	000000	000000	000000
000540	/	000000	000000	000000	000000	000000	000000	000000	000000
000560	/	000000	000000	000000	000000	000000	000000	000000	000000
000600	/	000000	000000	000000	000000	000000	000000	000000	000000
000620	/	000000	000000	000000	000000	000000	000000	000000	000000
000640	/	000000	000000	000000	000000	000000	000000	000000	000000
000660	/	001742	000002	000020	000020	000020	000711	177770	000001
000700	/	000000	000000	000000	000000	000000	000000	000000	000000
000720	/	000000	140300	000000	001774	002000	174017	000000	000000
000740	/	000000	000000	000000	000000	000000	000000	000000	000000
000760	/	000000	000000	000000	000000	000000	000000	000000	000000

	00	02	04	06	10	12	14	16	
001000	/	000000	000000	000000	000000	000000	000000	000000	000000
001020	/	000000	000000	000000	000000	000000	000000	000000	000000
001040	/	000000	000000	000000	000000	000000	000000	000000	000000
001060	/	000000	000000	000000	000000	000000	000000	000000	000000
001100	/	000000	000000	000000	000000	000000	000000	000000	000000
001120	/	000000	000000	000000	000000	000000	000000	000000	000000
001140	/	000000	000000	000000	000000	000000	000000	000000	000000
001160	/	000000	000000	000000	000000	000000	000000	000000	000000
001200	/	000000	000000	000000	000000	000000	000000	000000	000000
001220	/	000000	000000	000000	000000	000000	000000	000000	000000
001240	/	000000	000000	000000	000000	000000	000000	000000	000000
001260	/	000000	000000	000000	000000	000000	000000	000000	000000
001300	/	000000	000000	000000	000000	000000	000000	000000	000000
001320	/	000000	000000	000000	000000	000000	000000	000000	000000
001340	/	000000	000000	000000	000000	000000	000000	000000	000000
001360	/	000000	000000	000000	000000	000000	000000	000000	000000

	00	02	04	06	10	12	14	16
001400	/	000000	000000	000000	000000	000000	000000	000000
001420	/	000000	000000	000000	000000	000000	000000	000000
001440	/	000000	000000	000000	000000	000000	000000	000000
001460	/	000000	000000	000000	000000	000000	000000	000000
001500	/	000000	000000	000000	000000	000000	000000	000000
001520	/	000000	000000	000000	000000	000000	000000	000000
001540	/	000000	000000	000000	000000	000000	000000	000000
001560	/	000000	000000	000000	000000	000000	000000	000000
001600	/	000000	000000	000000	000000	000000	000000	000000
001620	/	000000	000000	000000	000000	000000	000000	000000
001640	/	000000	000000	000000	000000	000000	000000	000000
001660	/	000000	000000	000000	000000	000000	000000	000000
001700	/	002322	000000	001726	004070	002016	000000	001722 100330
001720	/	174000	100330	174000	163046	002372	000071	000032 000170
001740	/	120230	000040	004020	121030	002372	002112	001774 001762
001760	/	120150	000000	121036	174000	101744	002112	000001 003770

	00	02	04	06	10	12	14	16
002000	/	004437	100212	002034	004012	001000	004070	002322 002120
002020	/	000002	000000	002122	000000	003762	000000	002066 003772
002040	/	003772	003762	000004	004014	000002	004020	002000 000000
002060	/	047506	020117	020040	117172	000012	144306	000040 003770
002100	/	144302	003770	101732	000001	121030	117172	077777 125630
002120	/	000000	000000	000000	000000	000000	141622	101550 101550
002140	/	101550	142064	101550	101550	101550	142252	101550 101550
002160	/	101550	142742	101550	101550	101550	141724	101550 101550
002200	/	101550	141724	101550	101550	101550	141622	101550 101550
002220	/	101550	141622	101550	101550	101550	101550	101550 101550

	00	02	04	06	10	12	14	16
007120	/	000000	000000	000000	000000	000000	000000	000000
007140	/	000000	000000	000000	000000	000000	000000	000000
007160	/	000000	000000	000000	000000	000000	000000	000000
007200	/	000000	000000	000000	000000	000000	000000	000000
007220	/	000000	000000	000000	000000	000000	000000	000000
007240	/	000000	000000	000000	000000	000000	000000	000000
007260	/	000000	000000	000000	000000	000000	000000	000000
007300	/	000000	000000	000000	000000	000000	000000	000000
007320	/	000000	000000	000000	000000	000000	000000	000000
007340	/	000000	000000	000000	000000	000000	000000	000000
007360	/	000000	000000	000000	000000	000000	000000	000000

	00	02	04	06	10	12	14	16
007400	/	000000	000000	000000	000000	000000	000000	000000
007420	/	000000	000000	000000	000000	000000	000000	000000
007440	/	000000	000000	000000	000000	000000	000000	000000
007460	/	000000	000000	000000	000000	000000	000000	000000
007500	/	000000	000000	000000	000000	000000	000000	000000
007520	/	000000	000000	000000	000000	000000	000000	000000
007540	/	000000	000000	000000	000000	000000	000000	000000
007560	/	000000	000000	000000	000000	000000	000000	000000
007600	/	000000	000000	000000	000000	000000	000000	000000
007620	/	000000	000000	000000	000000	000000	000000	000000
007640	/	000000	000000	000000	000000	000000	002000	000000 000000
007660	/	000000	000000	000000	000000	000000	000000	000000
007700	/	000000	000000	000000	000000	000000	000000	000000
007720	/	000000	000000	000000	000000	000000	000000	000000
007740	/	000000	000000	000000	000000	000000	000000	000000
007760	/	000000	000000	000000	000000	000000	00237B	000000

## **Chapter 23**

### **RSTS/E Peripheral Devices**

RSTS/E has several peripherals which are available to you. These devices include:

- user terminal (ASR-33, ASR-35, LA30 and LA36 DECwriters, VT05 and VT50 displays)
- high-speed paper tape reader/punch
- card reader
- line printer
- DECtape
- magnetic tape
- flexible diskette

While normal operation of a computer system is by programmed control, manual operation is necessary for some tasks. This chapter describes the manual control and operation of some of the common RSTS/E user peripherals.

Table 23-1 lists the RSTS/E peripherals according to their locations in Chapter 23.

**Table 23-1: A Guide to the RSTS/E Peripheral Devices**

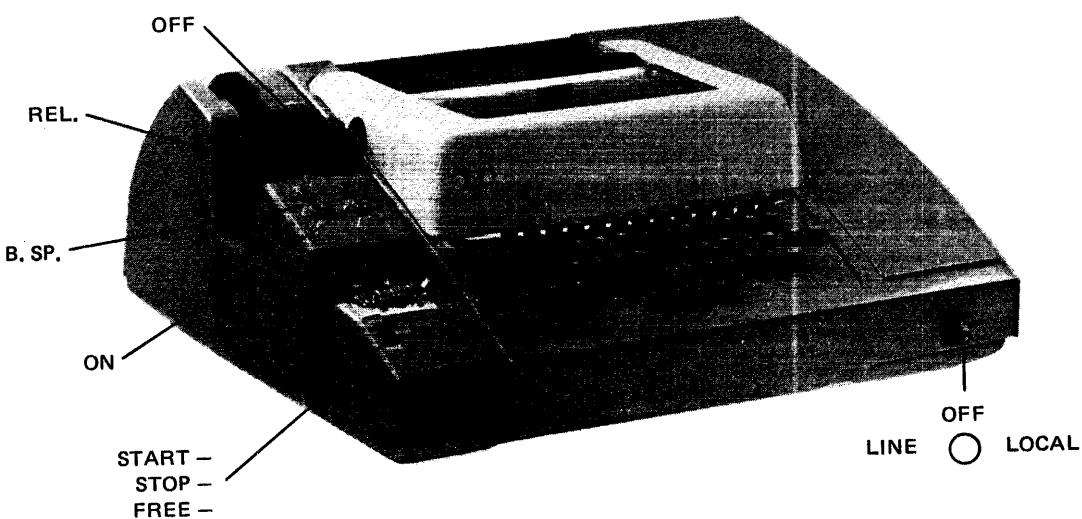
Device	Location
Card Reader (CR11)	23.3
DECtape Control & Transport	23.5
Flexible Diskette (RX11)	23.10
Line Printer (LP11)	23.4
Magnetic Tape Control & Transport (TM11/TU10 and TJU16)	23.6
Paper Tape Punch Low-Speed (ASR-33 Teletype*)	23.1.5
High-Speed	23.2.2
Paper Tape Reader Low-Speed (ASR-33 Teletype)	23.1.4
High-Speed	23.2.1
Terminals	
DECwriter II (LA36)	23.9
Teletype (ASR-33)	23.1
VT05 Display Terminal	23.7
2741 Communications Terminal	23.8

\*Teletype is a trademark of Teletype Corporation.

### 23.1 ASR-33 Teletype

The ASR-33 Teletype is an inexpensive, commonly employed user terminal. Major features are noted in Figure 23-1.

**Figure 23-1: ASR-33 Teletype Console**



The components of the Teletype unit and their functions are described below.

### 23.1.1 Control Knob

The control knob of the ASR-33 Teletype console has three positions:

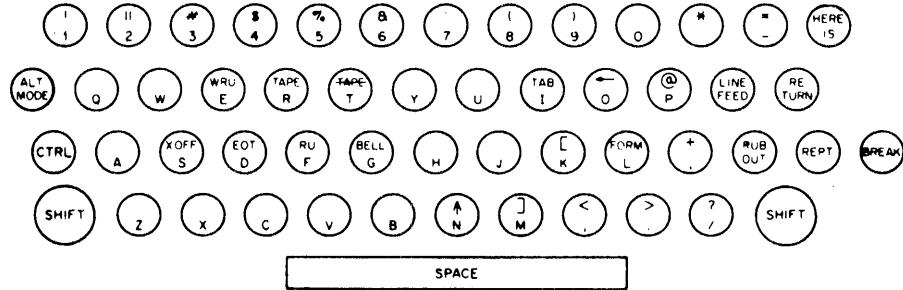
- LINE      The console has power and is connected to the system as an I/O device under RSTS/E control.
- OFF      The console does not have power.
- LOCAL    The console has power for off-line operation under control of the keyboard and switches only.

### 23.1.2 Keyboard

The Teletype keyboard shown in Figure 23-2 is similar to a typewriter keyboard, except that some nonprinting characters are included as upper case elements. For typing characters or symbols such as \$, %, or # which appear on the upper portion of numeric keys and some alphabetic keys, the SHIFT key is depressed while the desired key is typed.

Nonprinting operational functions are shown on the upper part of some alphabetic keys. By depressing the CTRL (control) key and typing the desired key, these functions are activated.\*

**Figure 23-2: Teletype Keyboard**



### 23.1.3 Printer

The Teletype printer provides a printed copy of input and output at ten characters per second, maximum rate. When the Teletype unit is on-line to the system (control knob turned to LINE), the copy is generated by the computer (including the echoing of the characters you type); when the Teletype unit is off-line (control knob turned to LOCAL), the copy is generated directly from the keyboard onto the printer as a key is struck.

\*Although not shown on most keyboards, SHIFT/L produces the backslash character (\) and SHIFT/K and SHIFT/M produce the left and right square brackets, respectively.

### **23.1.4 Low-Speed Paper Tape Reader**

The paper tape reader is used to read data punched on eight-channel perforated paper tape at a maximum rate of 10 characters per second. The reader controls are shown in Figure 23-1 and described below.

**START** The reader is activated; reader sprocket wheel is engaged and operative.

**STOP** The reader is deactivated; reader sprocket wheel is engaged but not operative.

**FREE** The reader is deactivated; reader sprocket wheel is disengaged.

The following procedure describes how to properly position paper tape in the low-speed reader.

1. Raise the tape retainer cover.
2. Set reader control to FREE.
3. Position the leader portion of the tape over the read pins with the sprocket (feed) holes over the sprocket (feed) wheel and with the arrow on the tape (printed or cut) pointing outward (forward).
4. Close the tape retainer cover.
5. Make sure that the tape moves freely (if the tape does not move back and forth freely, the paper feed holes are not properly positioned).
6. Set reader control to START, and the tape is ready.

### **23.1.5 Low-Speed Paper Tape Punch**

The paper tape punch is used to perforate eight-channel rolled oiled paper tape at a maximum rate of 10 characters per second. The punch controls are shown in Figure 23-1 and described below.

**RELEASE** Disengages the tape to allow tape removal or loading.

**B.SP** Backspaces the tape one space for each firm depression of the B.SP button.

**ON (LOCK ON)** Activates the punch.

**OFF (UNLOCK)** Deactivates the punch.

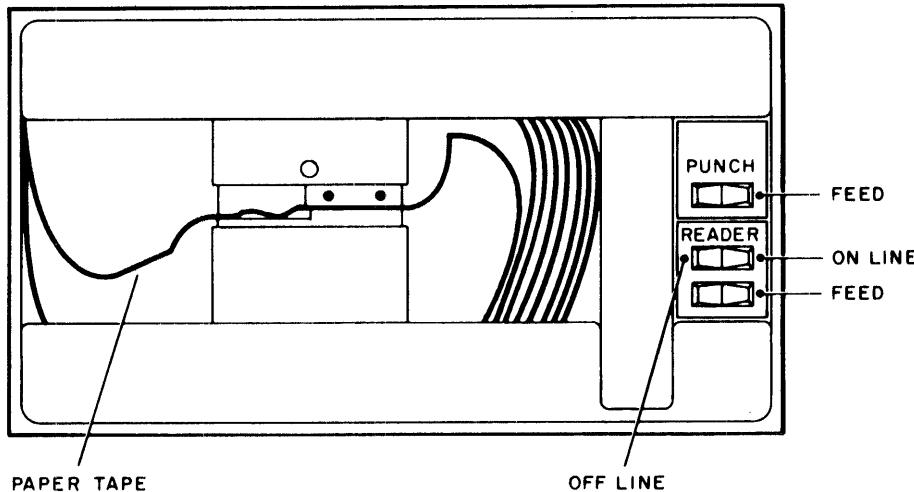
Blank leader/trailer tape is generated by:

1. Turning the control knob to LOCAL.
2. Turning the punch control to ON.
3. Typing the HERE IS key.
4. Turning the punch control to OFF.
5. Turning the control knob to LINE.

## 23.2 High-Speed Paper Tape Reader and Punch Units

One high-speed paper tape unit can be provided with each RSTS/E system. This unit is mounted on the central computer console. A high-speed paper tape unit is pictured in Figure 23-3 and descriptions of the reader and punch units follow.

Figure 23-3: High-Speed Paper Tape Reader/Punch



### 23.2.1 High-Speed Reader Unit

The high-speed paper tape reader is used to read data from eight-channel, fan-folded (non-oiled), perforated paper tape photoelectrically at a maximum rate of 300 characters per second.

#### CAUTION

Tape from the Teletype punch should not be used with the high-speed reader as the oil on the tape causes lint and dust to collect on the photoelectric cells.

Primary power is applied to the reader when the computer power is on.

In order to use the high-speed reader as an input device, turn the reader ON LINE/OFF LINE rocker switch to ON LINE. Load tape into the reader as follows:

1. Raise the tape retainer cover.
2. Place tape in right-hand bin with printed arrows pointing toward left-hand bin. (Channel one of the tape is toward the rear of the bin.)
3. Place several folds of blank tape past the reader and into the left-hand bin.
4. Place the tape over the reader head with feed holes engaged in the teeth of the sprocket wheel.
5. Close the tape retainer cover.
6. Depress the FEED rocker switch until leader tape is over the reader head.

The reader is capable of sensing whether a tape is in the reader. If an attempt is made to read a tape when the reader is either empty or OFF LINE, an error is generated.

### 23.2.2 High-Speed Punch Unit

The high-speed paper tape punch is used to record computer output on eight-channel, fan-folded, non-oiled paper tape at a maximum rate of 50 characters per second. All characters are punched under program control from the computer. Blank tape (feed holes only, no data), is produced by pressing the punch FEED rocker switch. The punch unit has power turned on whenever the computer has power; it does not require any additional on/off switch.

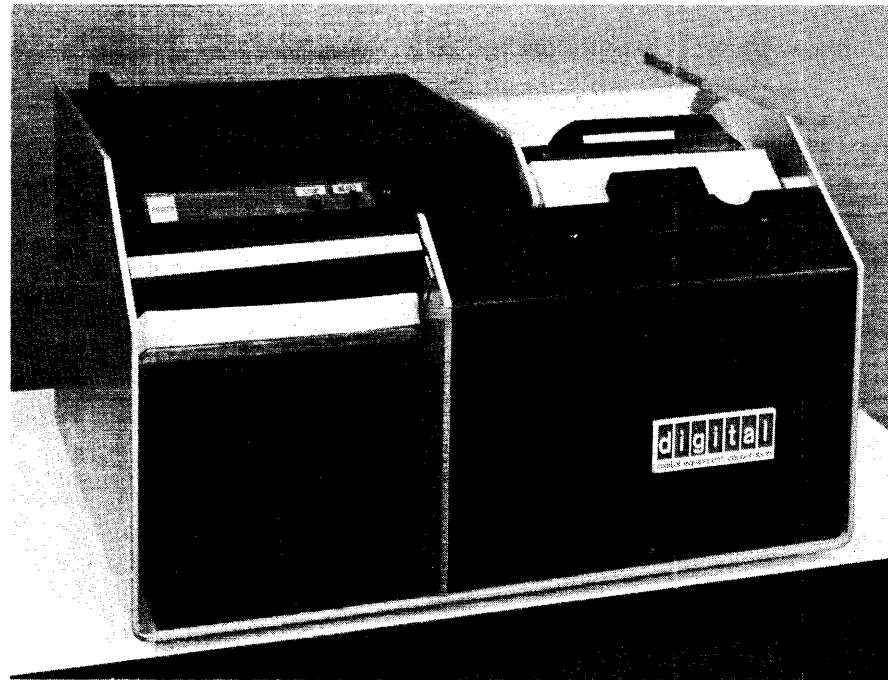
Fan-folded paper tape is generally grey in color. When a box of tape is nearly empty, purple tape is produced. Rather than risk running out of tape while punching, replace the box of paper tape at this point or notify the system manager, who will replace the tape.

## 23.3 CR11 Card Reader

The CR11 card reader allows the RSTS/E system to accept information from punched 80 column data cards. Cards can be read under program control at rates of up to 200 or 300 cards per minute.

Power to the reader, shown in Figure 23-4, is controlled by the ON/OFF switch on the upper left hand corner of the back panel. Two toggle switches are present on the back panel and should be set to AUTO and REMOTE for proper operation. The LAMP TEST button on the reader back panel can be depressed to check the operation of the various reader lights on the front panel.

Figure 23-4: CR11 Punched Card Reader



In order to use the card reader:

1. Remove card weight from input hopper. Place cards loosely in input hopper. The first card to be read is placed at the front of the deck, "9" edge down, column 1 to the left. Replace card weight on top of cards in input hopper. Cards should not be packed tightly.
2. Press green RESET button. Wait 4 seconds for RESET light to come on. The card deck is now able to be read under program control.
3. Cards may be loaded while the reader is operating provided tension is maintained on the front of the deck as cards are added to the rear. Additional cards should not be loaded until the hopper is 1/2 to 2/3 empty.
4. The output stacker bin can be unloaded while cards are being read. Care should be taken to maintain the order of the deck.

The various lights and switches (buttons) on the reader front panel and their significance are described in Table 23-2.

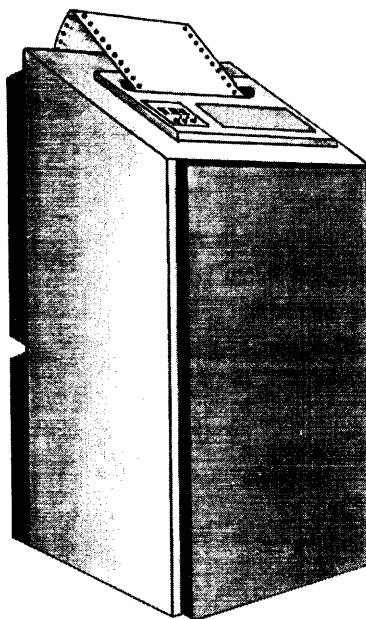
**Table 23-2: Card Reader Controls**

Light/Switch	Function
POWER	light indicates that there is power to the reader.
READ CHECK	light indicates a reading error, torn card, or card too long for reader. Reader stops and RESET light is out.
PICK CHECK	light indicates inability to remove card from input hopper. Reader stops and RESET light is out.
STACK CHECK	light indicates inability to remove card from input hopper. Reader stops and RESET light is out.
HOPPER CHECK	light indicates that there are no cards in input hopper or the output stacker is full. Condition must be manually corrected to allow further operation.
STOP	button, when depressed, causes red light to go on momentarily. RESET light goes out and reader operation stops as soon as the card currently in the read station has been read.
RESET	button, when depressed, causes green RESET light to go on and initializes card reader logic.

## **23.4 LP11 Line Printer**

The LP11 line printer, pictured in Figure 23-5, has an 80 column capacity, prints at a rate of 356 lines per minute at a full 80 columns, and can print 1100 lines per minute at 20 columns. These rates are based on a 64-character set. A 96-character set and 132-column version are also available. The print rate is dependent upon the data and number of columns to be printed.

**Figure 23-5: LP11 Line Printer System (80-column model)**



Characters are loaded into a 20-character printer memory by means of a Line Printer Buffer. When this buffer is full, the characters are automatically printed. This process continues until the 80 columns (four print zones) have been printed or a carriage return, line feed, or form feed character is recognized. The printer responds only to codes representing the character set and three control characters. All other codes are ignored.

#### **23.4.1 Line Printer Character Set**

The 64-character set consists of the 26 upper-case letters (A-Z), ten numerals (0-9), 27 special characters and the space character. The 96-character set contains all of the above plus 26 lower-case letters and 6 additional symbols. The character codes are 7-bit ASCII.

Characters are printed 10 characters per horizontal inch and 6 lines per vertical inch.

Line printers can use paper varying in width from 4 inches to 9-7/8 inches for the 80-column printer. Forms making up to six copies can be used when multiple copy printing is desired.

The special symbols available are as follows:

<b>64-character set</b>	<b>96-character set</b>
-------------------------	-------------------------

! " # \$ % & ' ( )	all of the 64-character
--------------------	-------------------------

* + , - . / ; : <	set symbols
-------------------	-------------

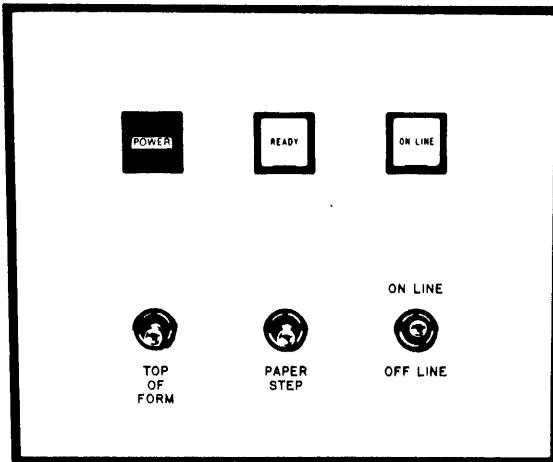
> = ? @ \ [ ] ↑	DEL
-----------------	-----

ASCII numeric equivalents for the various characters are contained in Appendix D.

### 23.4.2 Line Printer Operation

Figure 23-6 illustrates the line printer control panel on which are mounted three indicator lights and three toggle switches. Operation of these switches and the power switch, and the meaning of the lights is explained in Table 23-3.

**Figure 23-6: Line Printer Control Panel**



The following procedure is used when loading paper in the line printer.

1. Open front door of cabinet. POWER light should be on. Printer should be off line.
2. Lift control panel TOP OF FORM switch and release to move tractors to correct loading position.
3. Open drum gate by moving drum gate latch knob to left and up. Swing drum gate open.
4. Adjust right hand tractor paper width adjustment for proper paper width if necessary. (Loosen set screw on 80 column printer; user release mechanism on 132 column printer.) Tighten tractor after adjustment.
5. Open spring loaded pressure plates on both tractors.
6. Load paper so that the two red arrows point to a perforation. Paper should lie smoothly between tractors without wrinkling or tearing the feed holes.
7. Close spring-loaded pressure plates on both tractors.
8. Adjust COPIES CONTROL lever to proper number of copies to be made, if necessary. Set to 1 or 2 for single forms, set to 5 or 6 for six-part forms.
9. Close drum gate and lock in position with drum gate latch. After 10 seconds the READY indicator should light.
10. Lift TOP OF FORM switch several times to ensure paper is feeding properly.

- 11. Set printer to on line. ON LINE indicator should light. At this point printed matter can be aligned with the paper lines, if desired, by rotating the paper vertical adjustment knob.

**Table 23-3: Line Printer Controls**

Light/Switch	Function
POWER light	Glow red to indicate main power switch (located inside cabinet) is at ON position and power is available to the printer.
READY light	Glow white, shortly after the POWER light goes on to indicate that internal components have reached synchronous state, paper is loaded, and the printer is ready to operate.
ON LINE light	Glow white to indicate that ON LINE/OFF LINE toggle switch is in ON LINE position.
ON LINE/OFF LINE switch	This three-position toggle switch is spring-retumed to center. When momentarily positioned at ON LINE it logically connects the printer to the computer and causes the ON LINE light to glow. Positioned momentarily at OFF LINE, the logical connection to the computer is broken, the ON LINE light goes off, and the TOP OF FORM and PAPER STEP switches are enabled. If printer is switched to OFF LINE, the ON LINE light remains on until either PAPER STEP or TOP OF FORM switch is activated. The printer should again be turned ON LINE.
TOP OF FORM switch	This two-position toggle switch is tipped toward the rear of the cabinet to roll the form to the top of the succeeding page. It is spring-retumed to center position, and produces a single top-of-form operation each time it is actuated. The switch is effective when the printer is off line.
PAPER STEP switch	This two-position toggle switch operates similarly to TOP OF FORM but produces a single line step each time it is actuated. It is only effective when the printer is off line.
ON/OFF (main power) switch	This switch controls line current to the printer. To gain access to it, the printer front panel is unlatched, by pushing the circular button on the right hand edge, and opened to the left on its hinges. The switch is located to the left of center approximately fourteen inches below the top. If power is available, the red POWER light on the control panel will glow when the switch is positioned at ON.
	The switch is on when in the up position. The ON and OFF labels are printed on the stem of the switch. A group of two switches and three indicator lights, above the main power switch, are for the use of technicians in making initial adjustments to the printer.

## 23.5 TC11/TU56 Dectape Control and Transport

DECtape units are available on most RSTS/E systems. DECtape serves as an auxiliary magnetic tape storage facility to the system disk(s).

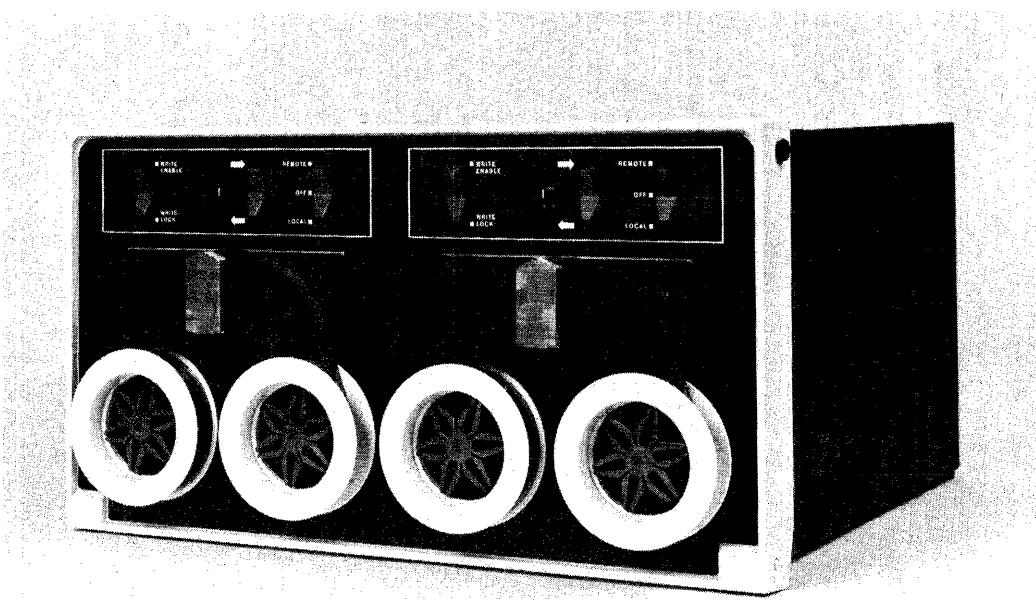
A DECtape peripheral unit consists of three components:

1. TU56 DECtape transport, pictured in Figure 23-7, which reads and/or writes information on magnetic tape.
2. TC11 Controller, the interface which controls information transfer. One controller serves up to four transports (up to 8 tape drives). The Controller is transparent to you.
3. DECtape, the recording medium used for data storage, consists of reel mounted magnetic tape formatted to permit read/write operations in either direction, error checking, block identification, and timing control.

DECtape stores and retrieves information at fixed positions on magnetic tape. The advantage of DECtape over conventional magnetic tape is that information at fixed positions can be addressed. Conventional magnetic tape stores information in sequential (not addressable directly), variable-length positions. DECtape incorporates timing and mark information to reference the fixed positions. The ten-channel DECtape records five channels of information: a timing channel, a mark channel, and three information channels. These five channels are duplicated on the five channels remaining to minimize any possibility of information loss from the other channels.

Each formatted (certified) DECtape contains 578 blocks of data consisting of 256 (16-bit) PDP-11 words per block. 562 blocks are available to you on each DECtape (several blocks are used for file directories).

**Figure 23-7: TU56 DECtape Transport**



Tape movement can be controlled by programmed instructions from the computer (i.e., through PIP) or by manual operation of switches located on the front panel of the transport. Data is transferred only under program control. The transport controls and lights are described in Table 23-4.

**Table 23-4: DECTape Controls**

Light/Switch	Function
REMOTE/OFF/LOCAL	<p>This three-position rocker switch determines control of the DECTape unit.</p> <p>REMOTE enables computer control of the transport unit.</p> <p>OFF disables the transport unit.</p> <p>LOCAL places the transport unit under operator control from the external transport switches.</p>
FORWARD/REVERSE	<p>This two-position rocker switch enables manual winding of tape when transport unit is under LOCAL control.</p> <p>FORWARD causes tape to feed onto right-hand spool.</p> <p>REVERSE causes tape to feed onto left-hand spool.</p>
Unit selector	<p>The value specified by this eight-position rotary switch identifies the transport to the computer. A unit selector value of 1 allows the tape on that unit to be accessed as device DT1:.</p>
WRITE ENABLE/WRITE LOCK	<p>This two-position rocker switch determines whether or not a tape can be written. Any tape can be searched and read; however when the WRITE LOCK switch is on, the tape is protected from accidental writing or program deletion.</p>
REMOTE light	<p>When lit, the tape unit is on-line. The tape unit is on-line when:</p> <ol style="list-style-type: none"> <li>1. REMOTE/OFF/LOCAL switch is in REMOTE position, and</li> <li>2. unit selector switch setting agrees with the DECTape unit currently being accessed.</li> </ol>
WRITE ENABLE light	<p>When lit, indicates that the WRITE ENABLE/WRITE LOCK switch is set to WRITE ENBALE, regardless of whether the REMOTE light is on or not.</p>

Operating procedures associated with DECTape units are described below. In order to mount a tape on a DECTape drive:

1. Set the REMOTE/OFF/LOCAL switch to OFF.
2. Place full DECTape reel on left spindle with label facing out. Press reel tightly onto spindle.

3. Pull tape leader over the two tape guides and the magnetic head until it reaches the take-up reel on the right hand side of the tape unit.
4. Wind loose tape end four turns around the empty right-hand reel by rotating right reel clockwise.
5. Set REMOTE/OFF/LOCAL switch to LOCAL. Verify that power is available to the tape unit.
6. Depress FORWARD/REVERSE switch in the FORWARD direction to wind about 15 turns onto the right-hand reel. This ensures that the tape is securely mounted.

In order to operate the DECTape unit on-line:

1. Set the REMOTE/OFF/LOCAL switch to either LOCAL or OFF and be sure power is available to the system.
2. Load the appropriate DECTape following the instructions above.
3. If the DECTape write operation is to be inhibited (to protect tape from accidental damage), set WRITE ENABLE/WRITE LOCK switch to WRITE LOCK. If the write operation is desired on this tape, set switch to WRITE ENABLE.
4. Dial the correct unit number on the unit selector. (No two active DECTape units should have the same unit number.)
5. Set REMOTE/OFF/LOCAL switch to REMOTE. Use of this DECTape unit is now under program control.
6. When on-line operation of this unit is to cease, set REMOTE/OFF /LOCAL switch to OFF or LOCAL. A moving tape can be stopped by quickly switching the REMOTE/OFF/LOCAL switch from REMOTE to LOCAL. The switch can be set to OFF when tape motion has stopped.

To remove a DECTape from the tape unit:

1. Set REMOTE/OFF/LOCAL switch to LOCAL.
2. Depress and hold REVERSE switch until all tape is wound onto the left-hand tape reel.
3. Set REMOTE/OFF/LOCAL switch to OFF.
4. Remove full reel from left-hand spindle.

## 23.6 TM11/TU10 and TJU16 Magnetic Tape Control and Transport

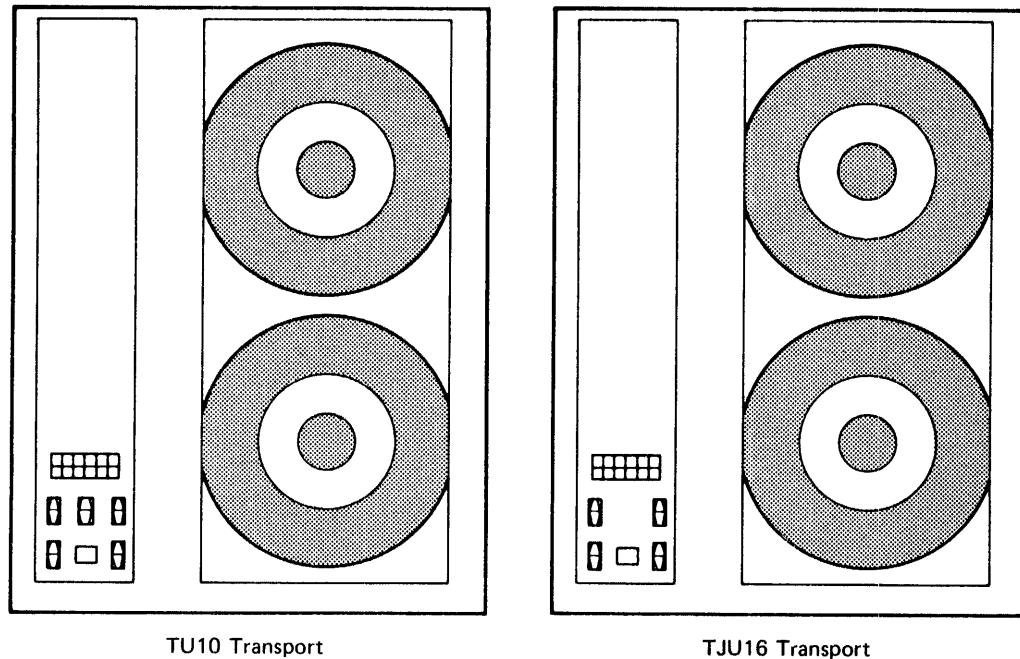
Magnetic tape is an optional addition to a RSTS/E system. Magnetic tape is used to provide storage for large volumes of data and programs. Writing, reading, and search operations are performed in a serial manner. Transfer of information can be made between RSTS/E and other computer systems because TJU16 and TM11 Controllers read and write information in industry-compatible format.

The basic magnetic tape system consists of:

1. The TU10 Transport, pictured in Figure 23-8, can read or write information on magnetic tape in seven or nine channels (tracks). The TJU16 Tape Transport reads and writes information on nine channels only. The TU10 reads and writes data at 800 bits per inch (BPI), and the TJU16 Tape Transport can be used to read and write magnetic tape either at 800 BPI or at 800 BPI and 1600 BPI.
2. The TM11 Controller is an interface between the TU10 Tape Transport and the PDP-11 system. The RH11 or RH70 is an interface between the PDP-11 and the TM02 Formatter/TU16 Transport. One controller, transparent to you, serves up to eight transport units.

Transfer rate is up to 36,000 characters per second for the TU10 and up to 72,000 characters per second for the TJU16. Ten and one half inch tape reels permit up to 2400 feet of tape per reel. Rewind time for a reel of 2400 feet is approximately 3 minutes, end to end.

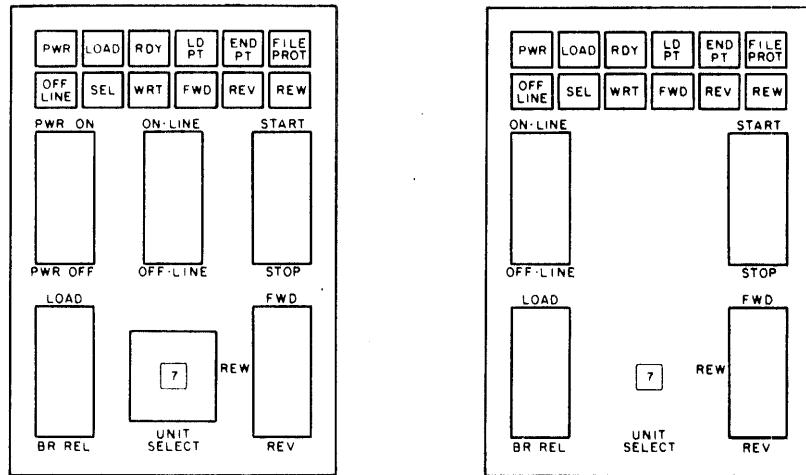
**Figure 23-8: Magnetic Tape System**



### **23.6.1 Magnetic Tape Control Panel**

The magnetic tape transport control panel is shown in Figure 23-9. This panel is located at the lower left of the TU10 and TJU16 front panel shown in Figure 23-8. Table 23-5 describes the tape transport controls and Table 23-6 describes the various tape transport indicators.

**Figure 23-9: Control Panels**



TU10 Control Panel

TJU16 Control Panel

**Table 23-5: Magnetic Tape Transport Controls**

Switch	Function
PWR ON/PWR OFF	This two-position switch applies power to the TU10 Transport. (This switch does not exist on the TJU16 Transport.)
ON-LINE/OFF-LINE	This two-position switch controls operation of the transport unit. ON-LINE allows system operation under program control; OFF-LINE allows manual operation. Tape unit cannot be remotely selected when switch is in OFF-LINE position.
START/STOP	This two-position switch controls starting and stopping of tape motion. STOP does not stop transport during a rewind operation.
LOAD/BR REL	This three-position switch energizes the vacuum system in the LOAD position (necessary for any operation). The BR REL position releases vacuum tension and allows reels to be manually rotated. Center position locks reel brakes.
UNIT SELECT	This eight-position rotary switch identifies the transport to the computer. A unit select value of 1 allows the tape on that unit to be accessed as device MT1:. No two transports should be set to the same number.
FWD/REW/REV	This three-position switch moves the tape in the selected direction, depending on activation of the START/STOP switch. FWD moves tape forward until BOT or EOT marker is sensed. REW rewinds tape onto the feed reel until BOT marker is sensed. REV rewinds tape until BOT marker is sensed; toggling the START/STOP switch again causes the tape to rewind off the reel.

**Table 23-6: Magnetic Tape Transport Indicators**

Light	Function
PWR light (power)	When lit, indicates that power is available to the transport unit.
LOAD light	When lit, indicates that vacuum system has been enabled, allowing either on-line or off-line commands.
RDY light (ready)	When lit, indicates that all I/O lines are enabled. Transport can accept processor commands provided SEL light is also lit.
LD PT light (load point)	When lit, indicates that BOT marker has been sensed; transport is ready for operation.
END PT light (end point)	When lit, indicates that EOT marker has been sensed; all tape motion stops to prevent tape from winding off reel.
FILE PROT light (file protection)	When lit, indicates that writing on the tape is inhibited. This is true if no file reel is mounted on feed reel hub or if a file reel is mounted without a write enable ring.
OFF-LINE light	When lit, indicates that transport can be operated manually and cannot be operated under program control.
SEL light (select)	When lit, indicates that transport has been selected and is completely on-line. Transport can read or write data.
WRT light (write)	When lit, indicates that write-enable ring has been installed on feed reel and transport can write on tape.
FWD light (forward)	When lit, indicates tape is moving in forward direction.
REV light (reverse)	When lit, indicates that tape is moving in reverse direction.
REW light (rewind)	When lit, indicates that tape is being rewound; Tape continues until BOT marker is sensed.

### 23.6.2 Magnetic Tape Operating Procedures

Whenever handling magnetic tapes and reels, observe the following precautions to prevent loss of data and/or damage to tape handling equipment:

1. Handle a tape reel by the hub hole only. Squeezing reel flanges can damage tape edges when winding or unwinding tape.
2. Never touch tape between BOT and EOT markers. Do not allow end of tape to drag on floor.
3. Never use a contaminated reel of tape; this spreads dirt to clean tape reels and can affect transport operation.

4. Always store tape reels inside containers. Keep empty containers closed so dust and dirt cannot collect.
5. Inspect tapes, reels, and containers for dust and dirt. Replace old or damaged take-up reels.
6. Do not smoke near transport or tape storage area. Smoke and ash are especially damaging to tape.
7. Do not place transport near a line printer or other device that produces paper dust.
8. Clean tape path frequently.

To mount a tape reel on the magnetic tape transport:

1. Apply power to the transport. (Depress the PWR ON switch on the TU10 transport.) Ensure that the LOAD/BR REL switch is in the center position and that the ON-LINE/OFF-LINE switch is in its OFF-LINE position.
2. Place a write-enable ring in the groove on the file reel if data is to be written on the tape. If writing is not required, be sure there is no ring in the groove.
3. Mount file reel onto lower hub with groove facing toward the back. Press reel tightly onto spindle; tighten center nut.
4. Install take-up reel (top reel), if necessary, as described above. The top reel is generally permanent and should not require installation by the user.
5. Place LOAD/BR REL switch to the BR REL position.
6. Unwind tape from the file reel and thread tape over tape guides and head assembly as shown in Figure 23-10. Wind about five turns of tape onto take-up reel.
7. Set LOAD/BR REL switch to LOAD position to draw tape into vacuum columns. As a result, the LOAD light comes on.
8. Select FWD and depress the START switch to advance the tape to the load point. When BOT marker is sensed, tape motion stops, the FWD indicator goes out and the LOAD PT indicator comes on.

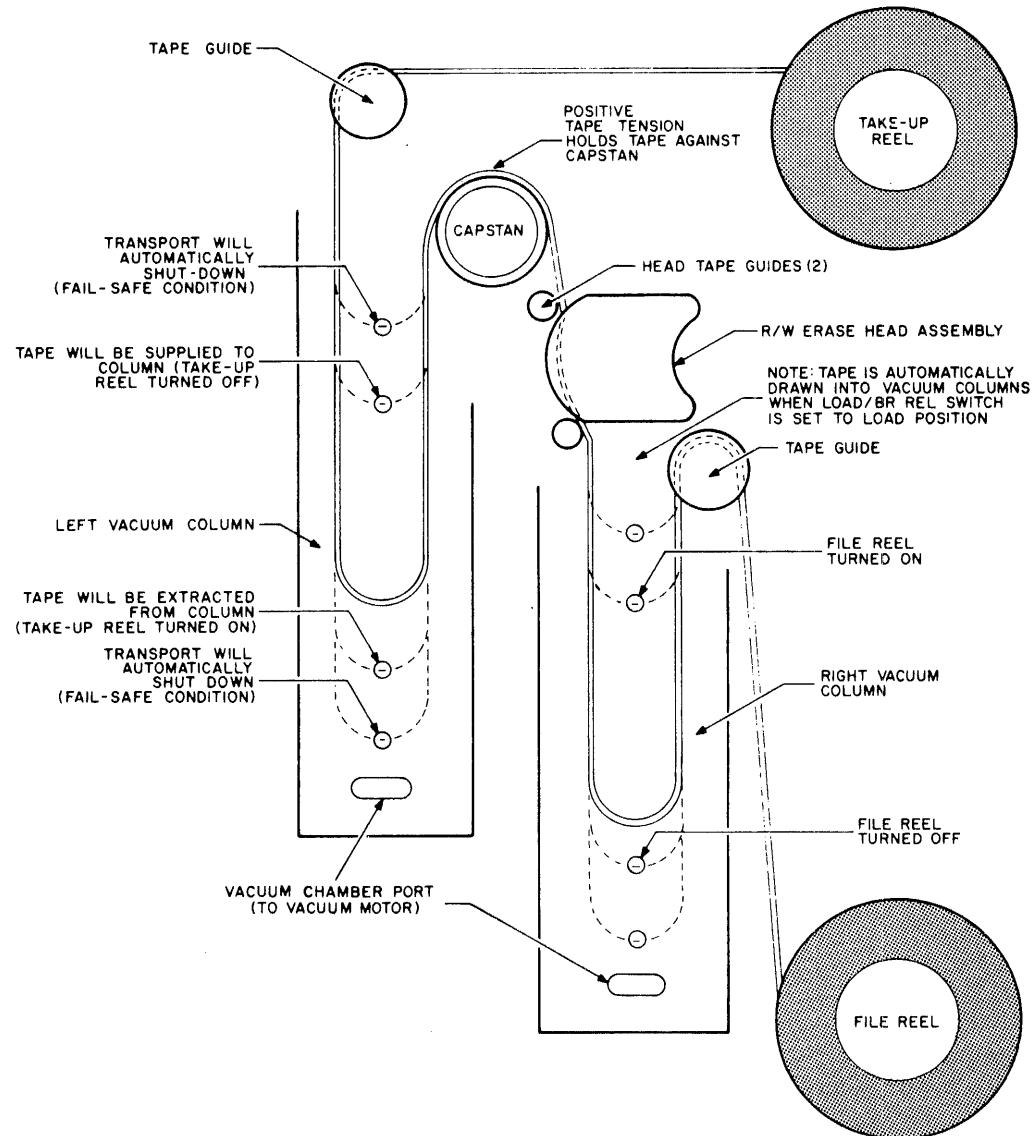
If tape motion continues for more than 10 seconds, depress STOP, select REV, and then depress START. The tape will advance to the BOT marker before stopping. (This may be necessary if, in winding the tape manually, the BOT marker has already been passed.)

Setting the ON-LINE/OFF-LINE switch to ON-LINE allows the transport to accept commands from the controller under program control. The transport is not fully on-line until the RDY and SEL indicators are lit.

To remove a tape from the transport unit:

1. Set ON-LINE/OFF-LINE switch to OFF/LINE position.
2. Set START/STOP switch to STOP position.
3. Set FWD/REW/REV switch to REW position.
4. Set START/STOP switch to START position. Tape rewinds until BOT marker is reached.
5. Set LOAD/BR REL switch to BR REL position to release brakes.
6. Gently hand wind the file reel in a counter clockwise direction until all of the tape is wound onto the reel. Do not jerk the reel. This may stretch or compress the tape which can damage data.
7. Remove the file reel from the hub assembly.

**Figure 23-10: Magnetic Tape Transport Threading Diagram**



## 23.7 VT05 Alphanumeric Display Terminal

The VT05 alphanumeric display terminal consists of a cathode ray tube (CRT) display and a self-contained keyboard. The VT05 keyboard operates as a typewriter keyboard except that no hard copy is produced. Each graphic character generated by typing at the keyboard is converted to a 5 by 7 dot matrix and displayed on a television-like screen. The full capacity of the screen is 20 lines, each containing 72 character positions for a total of 1440 characters.

When power is applied to the terminal and the CRT filament is warmed up, a blinking indicator called the cursor appears at the leftmost position of the top line (line number 1) of the screen. The blinking cursor indicates the position which the next generated character will occupy on the screen. The cursor can be moved up, down, left, or right by the use of various control characters generated by keys located to the right of the keyboard. When a displayable character is generated, its representation is displayed and the cursor automatically moves right to the next character location until the cursor reaches character position 72.

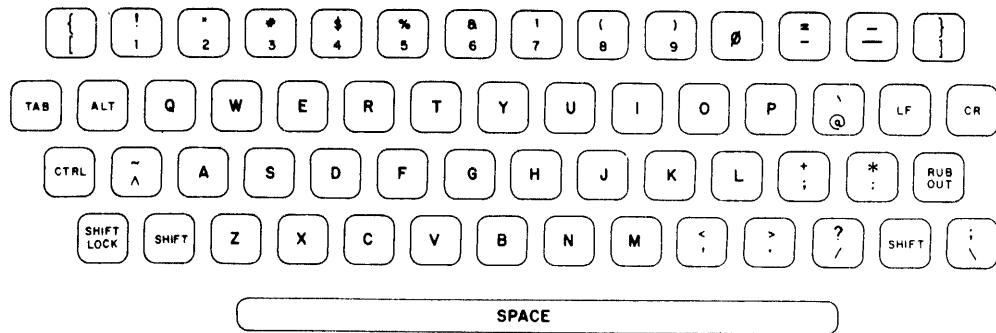
A speaker in the VT05 emits an audible tone or beep when the cursor reaches character position 65. This action warns the user that the cursor is within 8 spaces of the end of the line. (The speaker also beeps when the terminal or the computer generates the BEL character.) When the cursor reaches position 72 on a line, characters subsequently generated replace the character previously in position 72. Automatic carriage return or line feed is a hardware modification and terminals without the modification must be programmed to include these automatic operations.

The VT05 keyboard, shown in Figure 23-11, is similar to a typewriter keyboard except for the following operations keys:

ALT	Generates the ESCAPE character CHR\$(27).
CTRL	Control key is used to generate various control character combinations. See Section 4.9 for a description of control characters.
LF	Generates the LINE FEED character CHR\$(10).
CR	Generates the RETURN character CHR\$(13).
RUBOUT	Generates the DELETE character CHR\$(127).
TAB	Generates the HT character CHR\$(9) and causes the cursor to move to the right to the next tab stop. Tab stops are preset eight character spaces apart and are at locations 1, 9, 17, 25, 33, 41, 49, 57 and 65. Once the cursor reaches character position 65, the HT character moves the cursor right one position. Another HT character causes a RETURN and LINE FEED, leaving the cursor at position 1.

A line feed typed with the cursor in the bottom line (line 20) causes all displayed data on the screen to move up one line and any data in the top line to disappear. This action is termed automatic scrolling and is useful when a large amount of data is transmitted from the computer to the VT05 and is to be received and displayed. The user can employ the CTRL/S combination described in Section 4.9.4 to temporarily suspend transmission of such output to the screen and enable examination of data currently displayed on the screen. The CTRL/Q combination resumes output.

**Figure 23-11: VT05 Keyboard**



Four actions erase characters from the screen. A character is erased when the cursor is placed under it and a space character is generated. A character is replaced on the screen if the cursor is positioned under an existing character and another character is generated. The EOL and EOS special functions also erase characters from the screen.

The DELETE code, CHR\$(127), generated by typing the RUBOUT key, is ignored and no visual indication occurs on the display. When the RUBOUT key is typed on a VT05 terminal directly connected to a RSTS/E system, the system backspaces the cursor one character position, generates a space character in that position on the screen and in memory and backspaces one character position again. A RUBOUT key typed on a VT05 terminal connected to a RSTS/E system by a dial up line is treated as the RUBOUT key typed at a teleprinter unless the TTYSET SCOPE command is in effect for that line. Executing the SCOPE command causes the RUBOUT key to be treated as described above.

The ALT key typed at a VT05 generates the ESCAPE character. When received by the VT05, the ESCAPE character has no effect on the display. RSTS/E system programs such as GRIPE treat the ESCAPE character as a line terminating character; the system echoes the \$ character on the display when the ESCAPE character is received.

Controls to adjust the quality of the display are located on the right hand side of the terminal shown in Figure 23-12. The means by which you select a baud rate and mode of operation are on the rear panel of the device. At speeds above 300 baud, FILL characters for time delay as shown in Table 23-7 are required after some control characters are generated. The number of FILL characters required depends upon the baud rate at which the terminal operates.

To effect the proper number of FILL characters on a VT05 terminal which operates above 300 baud and whose characteristics are not permanently set, use the FILL command of the TTYSET system program.

**Table 23-7: FILL Characters Required for VT05**

Baud Rate	Number of FILL Characters
300	none
600	1
1200	2
2400	4

**Figure 23-12: VT05 Alphanumeric Display Terminal**



### **23.7.1 Controls and Operating Procedures**

The controls and switches for a VT05 terminal are listed and described in Table 23-8. To start the terminal connected by a direct line to the RSTS/E system, do the following:

1. Set the LOC/REM switch to the REM position.
2. Set the ON/OFF switch to the ON position. Allow approximately one minute for the filament to warm up, for the blinking cursor to appear in the home position (upper left corner of the display), and for the speaker to emit one beep.
3. If the cursor fails to appear as specified, press the HOME key. Ensure that the BRIGHTNESS control is not turned fully counterclockwise. If the cursor still fails to appear, place the ON/OFF switch in its OFF position and report the malfunction to the proper person.

4. To properly adjust the clarity of the display, turn the CONTRAST control counterclockwise to its minimum setting. Turn the BRIGHTNESS control counterclockwise until the characters are barely visible. Adjust the CONTRAST control to the optimum level.
5. When the operating session is completed, set the ON/OFF switch to its OFF position.

**Table 23-8: VT05 Controls and Switches**

Location	Label	Operation
Keyboard	ON/OFF LOC/REM	<p>When placed in its ON position, power is applied to the VT05 display and the refresh memory is cleared. When placed in its OFF position, the VT05 is inoperative and the screen is darkened.</p> <p>When placed in its LOC position, it breaks the electrical connection between the device and the computer. Local operation for maintenance and training is still allowed. In its REM position, it connects the terminal to the remote computer system. Data is simultaneously transmitted to and received from the computer.</p>
Right Side	BRIGHTNESS CONTRAST VERTICAL HORIZONTAL	<p>Turning this control in the clockwise direction increases the brightness of the screen. Turning the control counterclockwise decreases brightness. If turned fully counterclockwise, the display is completely darkened.</p> <p>This control increases and decreases the clarity of the characters displayed on the screen.</p> <p>This control synchronizes the display in the vertical position such that all 20 lines are visible on the screen.</p> <p>This control moves the display in the horizontal direction such that all 72 character positions are visible on the screen.</p>
Rear Panel	BAUD RATE FULL/HALF DUPLEX	<p>This 10-position selection switch determines the rates at which the terminal transmits and receives data.</p> <p>When at FULL, it allows the keyboard to transmit data to the computer and allows the display to simultaneously receive data from the computer. When at HALF, data is transmitted to the VT05 receiver logic as well as to the computer.</p>

To operate the VT05 terminal which is connected to the computer by a dial up line, perform the following steps:

1. Follow the procedures to start the VT05 terminal as if it were connected by a direct line to the RSTS/E system and set the BAUD RATE selector switch on the rear panel to its 110 position or to the position required by the permanent default characteristics established by the system manager for that line.

2. Dial the RSTS/E system and perform the log in procedures. If the line does not have permanent default characteristics for a VT05, continue with step 3. Otherwise, go to step 5.
3. Run the TTYSET system program and type the VT05 macro command or the SPEED command with the proper baud rate value. (The maximum baud rate allowed on a voice grade telephone line is 300.)
4. Set the BAUD RATE selector switch on the rear panel to the proper value, after which the VT05 operates at the proper baud rate.
5. When the operating session is completed, set the BAUD RATE selector switch to the 110 position and set the ON/OFF switch to its OFF position.

## 23.8 2741 Communications Terminals

RSTS/E systems allow the use of 2741 compatible\* terminals for time-sharing operations. 2741 terminals employ a Selectric typing mechanism which provides high quality copy in upper- and lower-case format. Such copy is especially suitable for documentation preparation and for output of the RUNOFF system program.

The 2741 terminal operates in half duplex mode and transmits and receives non-ASCII characters. The terminal echoes locally; the computer does not echo the characters. This action differs greatly from ASCII terminals which operate under RSTS/E in full duplex mode.

Half duplex operation of the 2741 terminal involves stricter intercommunication between the device and the computer. For example, when you type the RETURN key to enter a line, the keyboard locks until the computer accepts the line and sends an EOT character to unlock the keyboard. This locking and unlocking action is noticeable and may be annoying to the fast typist.

Many graphic code and keyboard arrangements are available for 2741 compatible terminals. RSTS/E supports the four most common codes: Correspondence, Extended Binary Coded Decimal, Binary Coded Decimal and CALL 360 BASIC. Since the system can be configured for any combination of the four codes, it is advisable to consult the system manager for information concerning which codes are available at the local installation.

2741 terminals do not generate all the characters normally used for time-sharing operations under RSTS/E. The system interprets certain keys in special ways described in the following subsections. The four supported keyboard arrangements are shown in Figures 23-13 through 23-16 at the end of the section.

---

\*The RSTS/E 2741 code has been tested with IBM, DATEL, and TREND DATA terminals. Digital Equipment Corporation makes no commitment to support 2741-type terminals made by other manufacturers.

### **23.8.1 The ATTN Key**

The 2741 terminal has an ATTN (Attention) key usually on the upper right hand portion of the keyboard. (Some terminals have an equivalent key called the BREAK key.) The key has several uses under RSTS/E depending upon whether the terminal is transmitting data to or receiving data from the computer.

The terminal is considered transmitting data to the computer when it is at BASIC-PLUS command level or in the program input state. The terminal is receiving characters whenever the system performs output to the device.

When the terminal is transmitting data, the ATTN key can have two effects. If pressed while the SHIFT key is in its upper case position, the key generates the effect of a CTRL/C combination as described in Section 4.9.1. The system echoes the transmission as either ^C or ↑C. If pressed while the SHIFT key is in its lower case position, the key generates the effect of a CTRL/U combination (erase line) as described in Section 8.1.3. The system echoes the lower case ATTN key as either ^U or ↑U.

When the terminal is receiving data, pressing the ATTN key with the SHIFT key in its upper case or lower case position generates the effect of a CTRL/C combination as described in Section 4.9.1. This action allows you to interrupt computer printout and return control to command level.

#### **NOTE**

The system reacts to the CTRL/C combination on a 2741 type terminal as if you had typed two such combinations in rapid succession. Programs executing the CTRL/C trap enable system function cannot trap a CTRL/C combination from a 2741 type terminal.

### **23.8.2 The RETURN Key**

The RETURN key generates one of two characters depending upon whether the SHIFT key is in its upper- or lower-case position. If typed as a lower-case character, the key generates a RETURN character (CHR\$(13)). If typed as an upper-case character, the key generates a LINE FEED character (CHR\$(10)). The system thus allows you to continue BASIC-PLUS statement lines when you enter source code from a 2741 terminal.

### **23.8.3 The BKSP Key**

The BKSP key generates one of two characters depending upon whether the SHIFT key is in its upper- or lower-case position. If typed as an upper-case character, the key backspaces the typing element one character position and generates the BACKSPACE character (CHR\$(8)). If typed as a lower-case character, the key backspaces the typing element one character position and generates the DELETE (RUBOUT) character (CHR\$(127)). In the latter case, the system deletes from memory the last character typed. This action is similar to the RUBOUT key typed at an ASCII terminal except that any replacement character typed is printed over the deleted character.

#### **23.8.4 Bracket Characters**

Since most 2741 terminals have no bracket characters, the system accepts ( and ) characters typed in place of [ and ] characters. This feature allows you to delimit a project-programmer number with open and close parenthesis characters rather than with bracket characters. For example, (100,100) is equivalent to [100,100]. System programs translate [ and ] characters to ( and ) characters before processing commands.

#### **23.8.5 Changing Codes**

If the system is configured to handle more than one transmission code, you can change the code recognized by the system. This feature is available because the system initializes each device to a default code when time-sharing operations begin. Therefore, if an individual terminal does not employ the default code, change the code to operate on the system.

To enable the system to recognize codes generated by one of the keyboards shown in Figures 23-13 through 23-16, type the numeral 1 followed by the upper case ATTN key. (See the description of the ATTN key in Section 23.8.1). For example:

1 ^C

E963

Ready

As a result, the system changes the code conversion table for the device. It prints an identifier which associates both the code and its required typing sphere and prints the Ready message. Table 23-9 shows the identifiers and related reference information.

**Table 23-9: 2741 Transmission Code Identifiers**

Identifier	Keyboard Code	Figure	Typing Sphere
C029	Correspondence	23-13	1167-029
E963	EBCD	23-14	1167-963
B938	BCD	23-15	1167-938
C087	CALL/360 BASIC	23-16	1167-087

The code table the system uses and the typing sphere the device uses must match the particular keyboard. If either the typing sphere or the code table is incorrect, the terminal produces garbled output. For example:

```
1 ' s  
X92*  
NUPVI
```

To verify the typing sphere, compare the typing sphere number shown in Table 23-9 with that stamped on the top edge of the type ball under the retaining clamp lever. To verify the keyboard, compare it with the one shown in the figure referred to in Table 23-9. If the typing sphere does not match the keyboard, install the correct type ball. Otherwise, continue changing codes until the system prints a recognizable identifier and Ready message. If you change codes four times and the system does not print a recognizable message, the terminal cannot be used unless the system is regenerated to support that code and keyboard arrangement.

The graphics of the typing spheres for each keyboard are compatible with conventional system graphics except for certain special characters. Where differences exist, the conventional character is shown in the related keyboard layout above the graphic printed by the sphere. All system programs accept lower-case characters by performing a conversion to upper case with the CVT\$\$ function described in the *BASIC-PLUS Language Manual*.

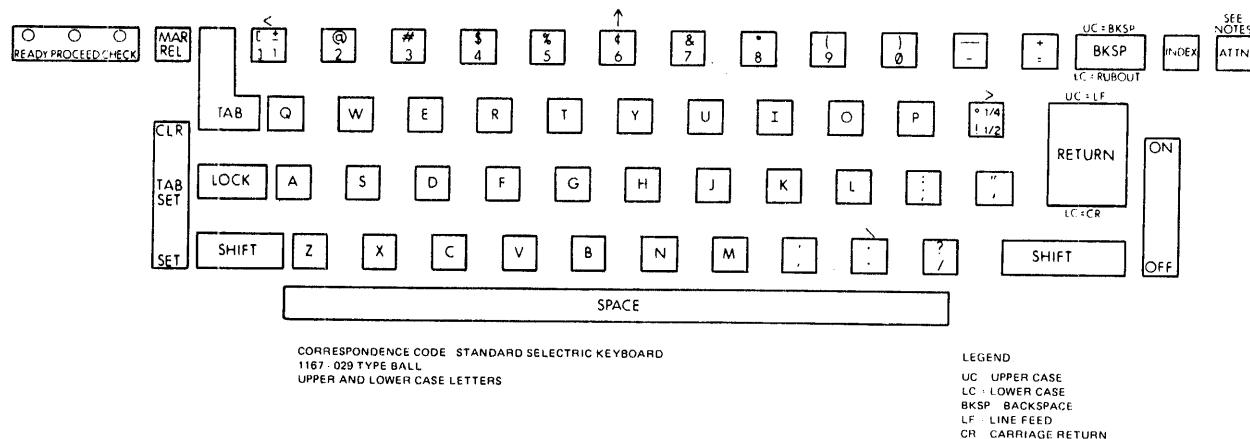
## Correspondence Code Keyboard

The following special characters and system actions are produced on this keyboard.

Character or Action	2741 Method
LF (Line Feed)	RETURN key in upper case.
RET (RETURN)	RETURN key in lower case.
DEL (RUBOUT)	BKSP key in lower case.
BKSP (BACKSPACE)	BKSP key in upper case.
CTRL/C combination	ATTN key in upper case.
CTRL/U combination	ATTN key in lower case.
CTRL/O combination	None.
CTRL/Z combination	None.
Change Code	1 followed by the upper case ATTN key.

Figure 23-13 shows the correspondence keyboard layout and specifies the required typing sphere. Special system characters such as ^, and \, which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints the ↑ character to designate the ^ character. Abbreviations above or below keys are defined in the legend.

Figure 23-13: Correspondence Code Keyboard



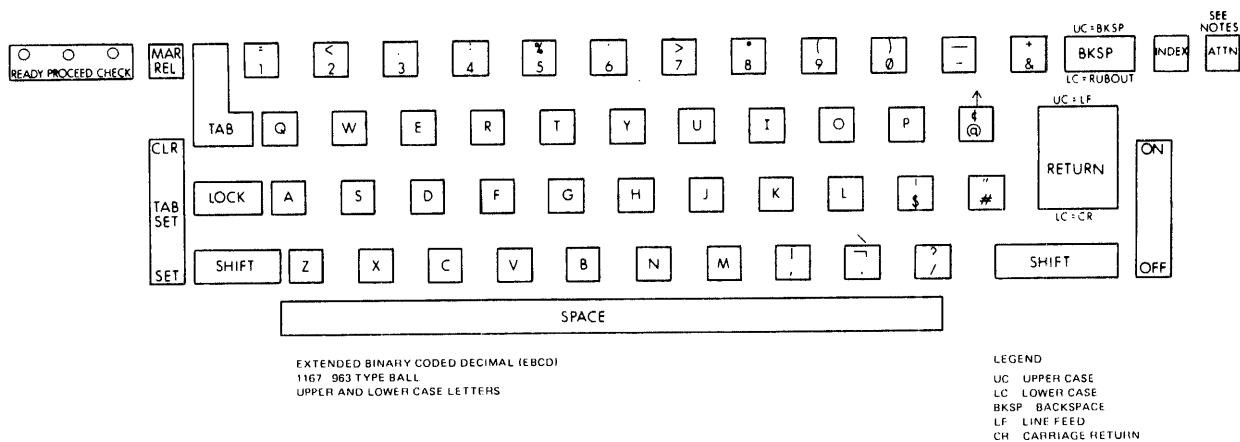
## EBCD Keyboard

The following special characters and system actions are produced on this keyboard.

Character or Action	2741 Method
<b>LF</b> (LINE FEED)	RETURN key in upper case.
<b>RET</b> (RETURN)	RETURN key in lower case.
DEL (RUBOUT)	BKSP key in lower case.
BKSP (BACKSPACE)	BKSP key in upper case.
CTRL/C combination	ATTN key in upper case.
CTRL/U combination	ATTN key in lower case.
CTRL/O combination	None.
CTRL/Z combination	None.
Change Code	1 followed by the upper case ATTN key.

Figure 23-14 shows the EBCD keyboard layout and specifies the required typing sphere. Special system characters such as ^, and \ which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints a ^ character in place of the ^ character. Abbreviations above or below keys are defined in the legend.

**Figure 23-14: EBCD Keyboard**



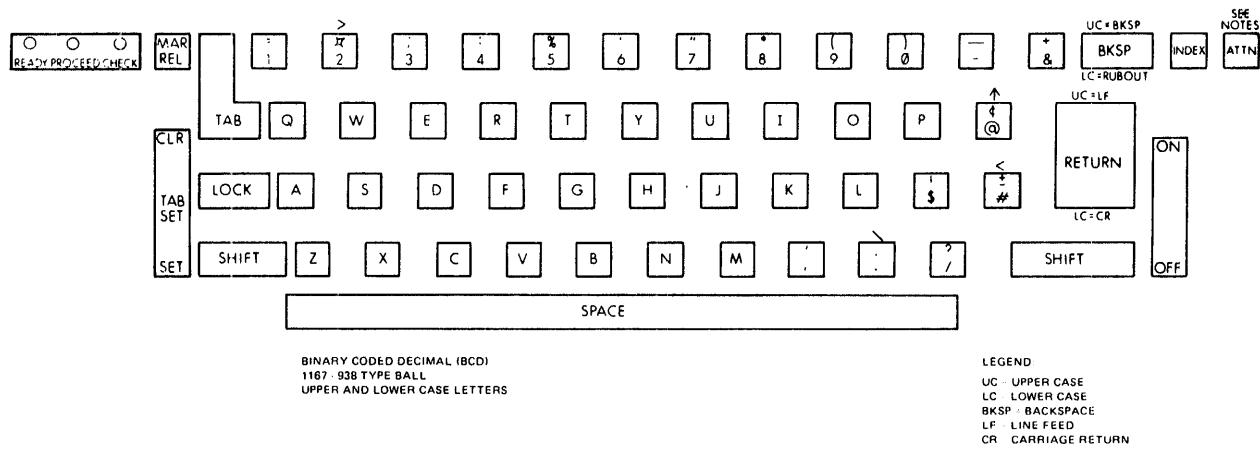
## BCD Keyboard

The following special characters and system actions are produced on this keyboard.

Character Action	2741 Method
LF (LINE FEED)	RETURN key in upper case.
REJ (RETURN)	RETURN key in lower case.
DEL (RUBOUT)	BKSP key in lower case.
BKSP (BACKSPACE)	BKSP key in upper case.
CTRL/C combination	ATTN key in upper case.
CTRL/U combination	ATTN key in lower case.
CTRL/O combination	None.
CTRL/Z combination	None.
Change Code	1 followed by the upper case ATTN key.

Figure 23-15 shows the BCD keyboard layout, and specifies the required typing sphere. Special system characters such as ^, and \, which do not have a graphic equivalent are shown in the area above the key that is used as a replacement. For example, the terminal prints the † character to designate the ^ character. Abbreviations above or below keys are defined in the legend.

**Figure 23-15:** BCD Keyboard



## CALL/360 BASIC Keyboard

The following special characters and system action are produced on this keyboard.

Character or Action	2741 Method
<b>L</b> (LINE FEED)	RETURN key in upper case.
<b>R</b> ( <b>E</b> ) <b>(R</b> <b>E</b> <b>T</b> <b>N</b> )	RETURN key in lower case.
<b>D</b> <b>E</b> <b>L</b> ( <b>R</b> <b>U</b> <b>B<b>O<b>U</b><b>T</b>)</b></b>	BKSP key in lower case.
<b>B</b> <b>K</b> <b>S</b> <b>P</b> ( <b>B</b> <b>A</b> <b>C</b> <b>K</b> <b>S</b> <b>P</b> )	BKSP key in upper case.
<b>C</b> <b>T</b> <b>R</b> <b>L</b> / <b>C</b> combination	ATTN key in upper case.
<b>C</b> <b>T</b> <b>R</b> <b>L</b> / <b>U</b> combination	ATTN key in lower case.
<b>C</b> <b>T</b> <b>R</b> <b>L</b> / <b>O</b> combination	None.
<b>C</b> <b>T</b> <b>R</b> <b>L</b> / <b>Z</b> combination	None.
Change Code	1 followed by the upper case ATTN key.

## 23.9 LA36 DECwriter II Operator Controls

DECwriter II offers fast, reliable operation and can be easily interfaced as a remote terminal or local computer I/O device. This terminal prints 30 characters per second and up to 132 characters per line. Characters are formed from a 7 x 7 dot matrix. Character spacing is 10 characters per inch, horizontal and 6 lines per inch, vertical. An original and up to five copies can be printed, and forms can be any width from 3 inches to 15 inches wide.

Table 23-10 describes the purpose of each operator control on the DECwriter II.

**Table 23-10: DECwriter II Operator Controls**

Control	Meaning
Power On-Off	Applies and removes AC power to entire machine.
Line/Local	Selects on-line or local operation.
Baud-rate - 110, 150, 300	3-position switch selects the baud rate clock frequency for communications line operation.
Forms thickness adjustment	Located on right side of print head carriage. Selects proper gap for 1- through 6-part form. Approximately 1 click for each part.
Right Tractor Adjustment	Thumb screw may be loosened to allow movement of right tractor for various forms widths.
Fine Vertical Tractor Release	Line feed knob may be depressed inward and rotated in the approximate direction for precise location of printing with respect to vertical zones.

## 23.10 RX11 Diskette

The RX11 flexible diskette system provides a low cost, random access, mass memory device capable of storing up to 256, 8-bit bytes of data in a non-file structured format. The diskette itself is a thin flexible, oxide-coated disk, similar in size to a 45 RPM phonograph record. The diskette is recorded on one side and is housed in an 8 inch square flexible envelope. The envelope has a large center hole for the drive spindle, a small hole for track index sensing, and a larger slit for the read/write head.

Each RX11 flexible diskette controller handles one or two drives, mounted side-by-side horizontally. The lower numbered unit is on the left; the higher numbered unit on the right. RSTS/E systems support up to four controllers (and eight drives).

Once power is applied to the flexible diskette system, raise the door of the desired drive by pulling up on the centrally located latch. Then simply insert the diskette (still housed in its square envelope) into the drive, label-side up, and read/write head slit first. Close the door firmly. The diskette is now ready to use; the diskette rotates at its operating speed immediately.

To remove the diskette, simply open the door and pull the envelope out. Once again, the diskette stops rotating as soon as the door is opened.



# **Appendix A**

## **BASIC-PLUS Language Summary**

### **A.1 Summary of Variable Types**

Type	Variable Name	Examples
Floating Point	Single letter optionally followed by a single digit.	A I X3
Integer	Any floating point variable name followed by a % character.	B% D7%
Character String	Any floating point variable name followed by a \$ character.	M\$ R1\$
Floating Point Matrix	Any floating point variable name followed by one or two dimension elements in parentheses.	S(4) N2(8)      E(5,1) V8(3,3)
Integer Matrix	Any integer variable name followed by one or two dimension elements in parentheses.	A%(2) E3%(4)      I%(3,5) R2%(2,1)
Character String Matrix	Any character string variable name followed by one or two dimension elements in parentheses.	C\$(1) A2\$(8)      S\$(8,5) V1\$(4,2)

## A.2 Summary of Operators

Operator	Type	Operates Upon
Arithmetic		
-	Unary minus	Numeric variables and constants
<sup>a</sup>	Exponentiation	
*, /	Multiplication, division	
+, -	Addition, subtraction	
Relational		
=	Equals	String or numeric variables and constants
<	Less than	Numeric variables
<=	Less than or equal to	
>	Greater than	String variables
>=	Greater than or equal to	
<>	Not equal to	
==	Numeric approximately equal to	
==	String exactly equal to	
Logical		
NOT	Logical negation	Integer variables and integer-valued expressions
AND	Logical product	
OR	Logical sum	
XOR	Logical exclusive or	
IMP	Logical implication	
EQV	Equivalence	
String	+	String constants and variables
Matrix		
+,-	Addition and subtraction of matrices or equal dimensions, one operation per statement	Dimensioned variables.
*	Multiplication of conformable matrices	
*	Scalar multiplication of a matrix	

## A.3 Summary of Functions

Under the Function column, the function is shown as:

`Y=function`

where the characters % and \$ are appended to Y if the value returned is an integer or character string.

A floating value (X), where specified, can always be replaced by an integer value. An integer value (N%) can always be replaced by a floating value (an implied FIX is done) except in the CVT%\$ and MAGTAPE functions (the symbol I% is used to indicate the necessity for an integer value).

Type	Function	Explanation
Mathematical	Y=ABS(X)	Returns the absolute value of X.
	Y=ATN(X)	Returns the arctangent of X, where X is in radians.
	Y=COS(X)	Returns the cosine of X, where X is in radians.
	Y=EXP(X)	Returns the value of e^X, where e=2.71828.
	Y=FIX(X)	Returns the truncated value of X, SGN(X)*INT(ABS(X)).
	Y=INT(X)	Returns the greatest integer in X which is less than or equal to X.
	Y=LOG(X)	Returns the natural logarithm of X, log(e)X.
	Y=LOG10(X)	Returns the common logarithm of X, log(10)X.
	Y=PI	Has a constant value of 3.14159.
	Y=RND	Returns a random number between 0 and 1.
	Y=RND(X)	Returns a random number between 0 and 1.
	Y=SGN(X)	Returns the sign function of X, a value of 1 preceded by the sign of X.
	Y=SIN(X)	Returns the sine of X, where X is in radians.
	Y=SQR(X)	Returns the square root of X.
	Y=TAN(X)	Returns the tangent of X, where X is in radians.
Print	Y%=POS(X%) or Y%=CCPOS(X%)	Returns the current position of the print head for I/O channel X, 0 is your terminal. (This value is imaginary for disk files.)
	Y\$=TAB(X%)	Moves print head to position X in the current print record, or is disregarded if the current position is beyond X. The first position is counted as 0.
String	Y%=ASCII(A\$)	Returns the ASCII value of the first character in the string A\$.
	Y\$=CHR\$(X%)	Returns a character string having the ASCII value of X. Only one character is generated.
	Y\$=CVT%\$(I%)	Maps integer into 2-character string.
	Y\$=CVTF\$(X)	Maps floating-point number into 4- or 8-character string.
	Y%=CVT%\$(A\$)	Maps first 2 characters of string A\$ into an integer.
	Y=CVT\$F(A\$)	Maps first 4 or 8 characters of string A\$ into a floating-point number.
	Y\$=CVT\$\$(A\$,I%)	Converts string A\$ to string Y\$ according to value of I%.
	Y\$=RAD\$(N%)	Converts an integer value to a 3-character string and is used to convert from Radix-50 format back to ASCII. See Appendix D.
	Y%=SWAP%(N%)	Causes a byte swap operation on the two bytes in the integer variable N%.

Type	Function	Explanation
	Y\$=STRING(N1%,N2%)	Creates string Y\$ of length N1 and characters whose ASCII decimal value is N2.
	Y\$=LEFT(A\$,N%)	Returns a substring of the string A\$ from the first character to the Nth character (the leftmost N characters).
	Y\$=RIGHT(A\$,N%)	Returns a substring of the string A\$ from the Nth to the last character; the rightmost characters of the string starting with the Nth character.
	Y\$=MID(A\$,N1%,N2%)	Returns a substring of the string A\$ starting with the N1 and being N2 characters long (the characters between and including the N1 to N1+N2-1 characters).
	Y%=LEN(A\$)	Returns the number of characters in the string A\$, including trailing blanks.
	Y%=INSTR(N1%,A\$,B\$)	Indicates a search for the substring B\$ within the string A\$ beginning at character position N1. Returns a value 0 if B\$ is not in A\$, and the character position of B\$ if B\$ is found to be in A\$ (character position is measured from the start of the string).
	Y\$=SPACE\$(N%)	Indicates a string of N spaces, used to insert spaces within a character string.
	Y\$=NUM\$(N%)	Indicates a string of numeric characters representing the value of N as it would be output by a PRINT statement. For example: NUM\$(1,0000)=(space)1(space) and NUM\$(-1,0000)=1(space).
	Y\$=NUM1\$(N)	Returns a string of characters representing the value of N. This is similar to the function NUM\$, except that it does not return spaces or E-format results.
	Y=VAL(A\$)	Computes the numeric value of the string of numeric characters A\$. If A\$ contains any character not acceptable as numeric input with the INPUT statement, an error results. For example: VAL("15")=15
	Y\$=XLATE(A\$,B\$)	Translate A\$ to the new string Y\$ by means of the table string B\$.
	Y\$=SUM\$(A\$,B\$)	Returns a numeric string equal to the arithmetic sum of numeric strings A\$ and B\$.
	Y\$=DIFF\$(A\$,B\$)	Returns a numeric string equal to the arithmetic difference A\$-B\$ of numeric strings A\$ and B\$.
	Y\$=PROD\$(A\$,B\$,P%)	Returns a numeric string equal to the product of numeric strings A\$ and B\$, rounded or truncated to P% places.
	Y\$=QUO\$(A\$,B\$,P%)	Returns a numeric string equal to the arithmetic quotient A\$/B\$ of numeric strings A\$ and B\$, rounded or truncated to P% places.
	Y\$=PLACE\$(A\$,P%)	Returns a numeric string equal to the numeric string A, rounded or truncated to P% places.

Type	Function	Explanation
	T% = COMP%(A\$,B\$)	Returns a value reflecting the result of an arithmetic comparison between numeric strings A\$ and B\$; T%=-1 for A<B, 0 for A=B and 1 for A>B.
System	Y\$=DATE\$(0%)	Returns the current date in the following format: 02-Mar-79
	Y\$=DATE\$(N%)	Returns a character string corresponding to a calendar date as follows:  N=(dayofyear)+[(numberofyearssince1970)*1000] DATE\$(1) = "01-Jan-70" DATE\$(125) = "05-May-70" DATE\$(4125) = "05-May-74"
	Y\$=TIME\$(0%)	Returns the current time of day as a character string as follows:  TIME\$(0) = "05:30 PM" or "17:30 "
	Y\$=TIME\$(N%)	Returns a string corresponding to the time at N minutes before midnight, for example:  TIME\$(1) = "11:59 PM" or "23:59 " TIME\$(1440) = "12:00 PM" or "00:00 " TIME\$(721) = "11:59 AM" or "11:59 "
	Y=TIME(0%)	Returns the clock time in seconds since midnight, as a floating-point number.
	Y=TIME(1%)	Returns the central processor time used by the current job in tenths of seconds.
	Y=TIME(2%)	Returns the connect time (during which the user is logged into the system) for the current job in minutes.
	Y=TIME(3%)	Returns to Y the decimal number of kilo-core ticks (kct's) used by this job.
	Y=TIME(4%)	Returns to Y the decimal number of minutes of device time used by this job.
	Y% = STATUS	Returns to Y% the status of a channel as of the most recent OPEN statement executed in the program.
	Y% = BUFSIZ(N)	Returns to Y% the buffer size of the device or file open on channel N.
	Y% = LINE	Returns to Y% the line number of the statement being executed at the time of an interrupt.
	Y% = ERR	Returns value associated with the last encountered error if an ON ERROR GOTO statement appears in the program.
	Y% = ERL	Returns the line number at which the last error occurred if an ON ERROR GOTO statement appears in the program.
Matrix	MAT Y=TRN(X)	Returns the transpose of the matrix X.
	MAT Y=INV(X)	Returns the inverse of the matrix X.

Type	Function	Explanation
	Y=DET	Following an INV(X) function evaluation, the variable DET is equivalent to the determinant of X.
	Y%=NUM	Following input of a matrix, NUM contains the number of rows input, or, in the case of a dimensional matrix, the number of elements entered.
	Y%=NUM2	Following input of a matrix, NUM2 contains the number of elements entered in that row.
Input/Output	Y%=RECOUNT	Returns the number of characters read following every input operation. Used primarily with non-file structured devices.

## A.4 Summary of BASIC-PLUS Statements

The following summary of statements available in the BASIC-PLUS language defines the general format for the statement as a line in a BASIC program. If more detailed information is needed, refer to the *BASIC-PLUS Language Manual*.

In these definitions, elements in angle brackets are necessary elements of the statement. Elements in square brackets are necessary elements of which the statement may contain one. Elements in braces are optional elements of the statement.

Where the term line number ({line number}) is shown in braces, this statement can be used in immediate mode.

The various elements and their abbreviations are described below:

variable or var	Any legal BASIC variable.
line number	Any legal BASIC line number.
expression or exp	Any legal BASIC expression.
message	Any combination of characters.
condition or cond	Any logical condition.
constant	Any acceptable integer constant (need not contain a % character).
argument(s) or arg	Dummy variable names.
statement	Any legal BASIC-PLUS statement.
string	Any legal string constant or variable.
protection	Any legal protection code.
value(s)	Any floating point, integer, or character string constant.
list	The legal list for that particular statement.
dimension(s)	One or two dimensions of a matrix, the maximum dimension(s) for that particular statement.

## Statement Formats and Examples

### REM

{line number} REM <message>  
{line number} {<statement>} !<message>

```
100      REM      THIS IS A COMMENT
110      ! THIS IS ANOTHER FORM OF COMMENT
120      PRINT          ! PERFORM A CR/LF
```

### LET

{line number} {LET}<var>{,<var>,<var>...}=<exp>

```
110      LET A% = 40% \ B=22
120      C,F1,V(0) = 0           ! MULTIPLE ASSIGNMENT
```

### DIM

line number DIM<var(dimension(s))>

```
30      DIM A(20), B$(6,5), C%(99)
```

line number DIM#<constant>,<var(dimension(s))>=<constant>

```
70      DIM *4, A$(100) = 32, B(50,50)
```

### RANDOMIZE

line number RANDOM{IZE}

```
40      RANDOM
100     RANDOMIZE
```

### IF-THEN, IF-GOTO

line number IF <cond> THEN<statement>  
THEN<line number>  
GOTO<line number>

```
55      IF A>B OR B>C THEN PRINT 'NO'
65      IF FNA(R) = B THEN 250
90      IF L <X^2% AND L<> 0, GO TO 345
```

### IF-THEN-ELSE

line number IF <cond> THEN<statement>  
THEN<line number> }  
 } ELSE<statement>  
 } ELSE <line number>

```
100     IF B = A THEN PRINT 'EQUAL' ELSE PRINT 'NOT EQUAL'
110     IF A == N THEN 200 ELSE PRINT A \ STOP
120     IF FNA(R) = B
          THEN GO TO 260
          ELSE LET B = B+FNA(R1)
          \ GO TO 370
```

### FOR

line number FOR<var>=<exp>TO<exp>{STEP<exp>}

```
200     FOR I=2 TO 40 STEP 2
320     FOR T%=0% TO TG% STEP 1%
410     FOR N=A TO (C + S1)/A
```

## Statement Formats and Examples

### NEXT

line number NEXT<var>

```
460      NEXT I  
465      NEXT N%
```

### FOR-WHILE, FOR-UNTIL

line number FOR<var>=<exp>{STEP <exp>} WHILE  
UNTIL <cond>

```
450      FOR I=1, STEP 3, WHILE I<X  
470      FOR N% = 2% STEP 4% UNTIL N%>A% OR N%=B%  
500      FOR B = 0, STEP B=1, UNTIL B>B1
```

### EXTEND

line number EXTEND

```
10       EXTEND ! PROGRAM IN EXTEND MODE
```

### NO EXTEND

line number NO EXTEND

```
110      NOEXTEND  
110      NO EXTEND
```

### DEF\*, single line

line number DEF\* FN<var>(arg)=<exp(arg)>

```
120      DEF* FNA(X,Y,Z) = SQR(X^2% + Y^2% + Z^2%)
```

BASIC-PLUS supports DEF as well as DEF\* in single- and multiple-line statements, for compatibility with earlier versions. However, DEF\* is preferred for compatibility with BASIC-PLUS-2.

### DEF\*, multiple line

line number DEF\* FN<var>(arg)

<statements>

line number FN<var>=<exp>

line number FNEND

```
300      DEF*FNF(M%)      !FACTORIAL FUNCTION  
310      IF M%=0% OR M%=1%  
                  THEN FNF=1%  
                  ELSE FNF=M%*FNF(M%-1%)  
320      FNEND
```

### GOTO

line number GOTO<line number>

```
100     GOTO 150
```

## Statement Formats and Examples

### ON-GOTO

line number ON<exp>GOTO<list of line numbers>

150        ON X% GOTO 170,570,430,300

### GOSUB

line number GOSUB<line number>

190        GOSUB 2000

### ON-GOSUB

line number ON<exp>GOSUB<list of line numbers>

230        ON FNC(M) GOSUB 2000,2400,3000

### RETURN

line number RETURN

370        RETURN

### CHANGE

{line number} CHANGE [ <array name> ] TO [ <string var> ] [ <array name> ]

300        CHANGE A\$ TO X

450        CHANGE F1 TO F1\$

### OPEN

{line number} OPEN<string> {FOR [ INPUT ] [ OUTPUT ] } AS FILE<exp>  
{,RECORDSIZE<exp>} {,CLUSTERSIZE<exp>} {,MODE<exp>}

100        OPEN 'PP:' FOR OUTPUT AS FILE B1%

120        OPEN 'FOO' AS FILE 3

160        OPEN 'DT4:DATA,TR' FOR INPUT AS FILE 10

### CLOSE

{line number} CLOSE{<}<list of exp>

780        CLOSE 2%

920        CLOSE 11, 3, N1

### READ

line number READ<list of variables>

100        READ A, B\$, F1%, B(1%), R2

### DATA

line number DATA <list of variables>

1300        DATA 4.3, "STRING", 10,1000,1,45E9

## Statement Formats and Examples

### RESTORE

{line number} RESTORE

130 RESTORE

### PRINT

{line number} PRINT {#<exp>,<list>}

130 PRINT !GENERATES CR/LF  
170 PRINT 'BEGINNING OF OUTPUT:' ;I,A\*I  
240 PRINT #4, 'OUTPUT TO DEVICE';N%  
260 PRINT 'TITLE: ' ;T\$, 'REF, \*';R\$

### PRINT USING

{line number} PRINT {#<exp>,<list>}USING<string>

540 PRINT USING '##.##',AA  
680 PRINT #7, USING B\$,A,B,C

### INPUT

{line number} INPUT {#<exp>,<list>}

140 INPUT 'TYPE YOUR NAME ',A\$  
180 INPUT #8, A,N, B\$

### INPUT LINE

{line number} INPUT LINE {#<exp>,<string>}

170 INPUT LINE R\$  
190 INPUT LINE #1,E\$

### NAME-AS

{line number} NAME <string> AS <string>

455 NAME 'NONAME' AS 'FILE1<48>'  
990 NAME "DT4:MATRIX" AS 'MATA1<48>'

### KILL

{line number} KILL <string>

190 KILL 'NONAME'

### ON ERROR GOTO

line number ON ERROR GOTO {<line number>}

100 ON ERROR GOTO 9000  
110 ON ERROR GO TO !DISABLES ERROR ROUTINE  
111 ON ERROR GOTO 0 !DISABLES ERROR ROUTINE

### RESUME

line number RESUME {<line number>}

1000 RESUME !EQUIVALENT TO RESUME 0  
650 RESUME 200 !SPECIFY RESUMPTION POINT

## Statement Formats and Examples

### CHAIN

line number CHAIN <string>{LINE}{<exp>}

```
100      CHAIN 'PROG3' LINE 75
200      CHAIN "PROG3" LINE A
```

The following is also legal for compatibility with earlier versions of BASIC-PLUS.

```
440      CHAIN 'PROG2'
550      CHAIN 'PROG3' 75
670      CHAIN "PROG3" A
```

### STOP

line number STOP

```
450      STOP
```

### END

line number END

```
32767    END
```

## Matrix Statements

### MAT READ

{line number} MAT READ <list of matrices>

```
150      DIM A(20), B$(32), C%(15,10)
290      MAT READ A, B$(25), C%
```

### MAT PRINT

{line number} MAT PRINT{#<exp>,{<matrix name>}

```
150      DIM A(20), B%(15,30)
190      MAT PRINT A,      !PRINT 20 ELEMENTS FIVE TO A LINE
220      MAT PRINT B%(10,25);   !PRINT 10-BY-25 SUBSET
          OF B%, PACKED
270      MAT PRINT #2, A;      !PRINT ON OUTPUT CHANNEL 2
```

### MAT INPUT

{line number} MAT INPUT {#<exp>,{<list of matrices>}

```
100      DIM B$(40), F1%(35)
110      OPEN 'DT3:FOO' FOR INPUT AS FILE 3%
120      MAT INPUT #3, B4, F1%
```

### MAT Initialization

{line number} MAT <matrix name>= 
$$\begin{bmatrix} \text{ZER} \\ \text{CON} \\ \text{IDN} \end{bmatrix}$$
 {dimension(s)}

```
100      DIM B(15,10), A(10), C%(5)
110      MAT C% = CON
120      MAT B = IDN(10,10)
130      MAT B = ZER(N,M)
```

## Statement Formats and Examples

### Statement Modifiers (can be used in immediate mode)

IF

<statement> IF <condition>

510 PRINT T% IF T%>T1%

UNLESS

<statement> UNLESS <condition>

340 PRINT A\$ UNLESS Y%<0%

FOR

<statement> FOR <var> = <exp> TO <exp> {STEP <exp>}

175 LET B\$(I%) = C\$(I%) FOR I% = 1% TO J1%  
190 READ A(I%) FOR I% = 0% TO 20% STEP 15%

WHILE

<statement> WHILE <condition>

230 LET A(I%) = FNX(I%) WHILE A<45.5

UNTIL

<statement> UNTIL <condition>

1060 IF B <> 0 THEN A(I%) = B UNTIL I% > K

### System Statements

SLEEP

line number SLEEP<expression>

260 SLEEP 20 !DISMISS JOB FOR 20 SECONDS

WAIT

line number WAIT <expression>

520 WAIT A%

### Block I/O Statements

LSET

line number LSET <string var>{,<string var>} = <string>

900 LSET B\$ = 'XYZ'

RSET

line number RSET <string var>{,<string var>} = <string>

250 RSET C\$ = "67890"

## Statement Formats and Examples

### **FIELD**

line number FIELD#<expr>,<expr> AS <string var>{,<expr> AS <string var>}

710       FIELD #2%, 10% AS A\$, 20% AS B\$

### **GET**

line number GET#<expr>,{RECORD<expr>  
BLOCK<expr>},COUNT<expr>,USING<expr>

140       GET #1%, RECORD 99%

### **PUT**

line number PUT#<expr>,{RECORD<expr>  
BLOCK<expr>},COUNT<expr>,USING<expr>

390       PUT #1%, COUNT 80%

### **UNLOCK**

line number UNLOCK #<expression>

490       UNLOCK #5%



## **Appendix B**

### **BASIC-PLUS Command Summary**

<b>Command</b>	<b>Explanation</b>	<b>Section</b>
APPEND	Used to include contents of a previously saved source program in current program.	8.1.5
ASSIGN	Used to reserve an I/O device for the use of the individual issuing the command. The specified device can then be given commands only from the terminal which issued the ASSIGN. Also establishes a logical name for a device, establishes an account for the @ character, and assigns a default protection code.	4.1
BYE	Indicates to RSTS/E that you wish to leave the terminal. Closes and saves any files remaining open.	Chapter 13
CAT CATALOG	Returns your file directory. Unless another device is specified following the term CAT or CATALOG, the disk is the assumed device.	7.9
CCONT	For privileged users. Same as CONT command but detaches job from terminal.	9.2.3
COMPILE	Allows you to store a compiled version of your BASIC program. The file is stored on disk with the current name and the extension .BAC. Or, you can specify a new file name and extension.	7.4.3
CONT	Allows you to continue execution of the program currently in memory following the execution of a STOP statement.	9.2.2
DEASSIGN	Used to release the specified device for use by others. If no device is specified, all devices assigned to that terminal are released. An automatic DEASSIGN is performed when the BYE command is given. Also releases any logical name for a device.	4.2, 4.4, 4.7

Command	Explanation	Section
DELETE	Allows you to remove one or more lines from the program currently in memory. Following the word DELETE, type the line number of the single line to be deleted or two line numbers separated by a dash (-) indicating the first and last line of the section of code to be removed. Several single lines or line sections can be indicated by separating the line numbers, or line number pairs, with a comma.	8.1.2
EXIT	Returns you to your private default run-time system.	5.3
HELLO	Indicates to RSTS/E that you wish to enter a project-programmer number and password. Also attaches a detached job to the current terminal or changes accounts without having to log off the system.	Chapter 13
KEY	Used to re-enable the echo feature on the terminal following the issue of a TAPE command. Enter with LINE FEED or ESCAPE key.	4.8.2
LENGTH	Returns the length of the current program in memory, in 1K increments.	7.8
LIST	Allows you to obtain printed listing at the terminal of the program currently in memory, or one or more lines of that program. The word LIST by itself will cause the listing of the entire program. LIST followed by one line number will list that line; and LIST followed by two line numbers separated by a dash (-) will list the lines between and including the lines indicated. Several single lines or line sections can be indicated by separating the line numbers, or line number pairs, with a comma.	8.1.1
LISTNH	Same as LIST, but does not print header containing the program name and current date.	8.1.1
LOGIN	Same as HELLO.	Chapter 13
NEW	Clears your area in memory and allows you to input a new program from the terminal. A program name can be indicated following the word NEW or when the system requests it.	7.1.1
OLD	Clears your area in memory and allows you to recall a saved program from a storage device. You can indicate a program name following the word OLD or when the system requests it. If no device name is given, the file is assumed to be on the system disk. A device specification without a filename will cause a program to be read from an input-only device (such as high speed reader, card reader).	7.3

Command	Explanation	Section
REASSIGN	Transfers control of a device to another job.	4.3
RENAME	Causes the name of the program currently in memory to be changed to the name specified after the word RENAME.	7.5
REPLACE	Same as SAVE, but allows you to substitute a new program with the same name for an old program, erasing the old program.	7.6
RUN	Allows you to begin execution of the program currently in memory. The word RUN can be followed by a filename in which case the file is loaded from the system disk, compiled, and run, alternatively, the device and filename can be indicated if the file is not on the system disk. A device specification without a filename will cause a program to be read from an input only device (such as high-speed reader, card reader).	7.4.1
RUNNH	Causes execution of the program currently in memory but header information containing the program name and current date is not printed. If a filename is used, the command is executed as if no filename were given.	7.4.1
SAVE	Causes the program currently in memory to be saved on the system disk under its current filename with the extension .BAS. Where the word SAVE is followed by a filename or a device and a filename, the program in memory is saved under the name given and on the device specified. A device specification without a filename will cause the program to be output to any output only device (line printer, high-speed punch).	7.2
SCALE	Sets the scale factor to a designated value or prints the value(s) currently in effect if no value is designated.	7.10
TAPE	Used to disable the echo feature on the terminal while reading paper tape via the low-speed reader.	4.8.1
UNSAVE	The word UNSAVE is followed by the filename and, optionally, the extension of the file to be removed. The UNSAVE command cannot remove files without an extension. If no extension is specified, the source (.BAS) file is deleted. If no device is specified, the disk is assumed.	8.1.4

## Special Control Character Summary

Control Character	Explanation	Section
CTRL/C	Causes the system to return to BASIC command mode to allow for issuing of further commands or editing. Echoes on terminal as ^C.	4.9.1
CTRL/O	Used as a switch to suppress/enable output of a program on the terminal. Echoes as ^O.	4.9.2
CTRL/R	Causes system to print contents of buffer. Used to check contents of hard-copy terminal line that has been edited with RUBOUT.	4.9.3
CTRL/Q	When generated by a device on which a CTRL/S has interrupted output, causes computer to resume output at the next character.	4.9.4
CTRL/S	When generated by a device for which SCOPE characteristics are set, interrupts computer output on the device until either CTRL/Q or another character is generated.	4.9.4
CTRL/U	Deletes the current typed line, echoes as ^U and performs a carriage return/line feed.	8.1.3
CTRL/Z	Used as an end-of-file character.	4.9.5
CTRL/T	Causes the system to print an abbreviated status report that describes the job number, keyboard, current run-time system, program state, job and run-time system size, and processing time. To have effect, this option must be installed during system generation.	4.9.8
ESCAPE or ALT MODE Key	Enters a typed line to the system, echoes on the terminal as a \$ character and does not cause a carriage return/line feed.	4.9.7
LINE FEED Key	Used to continue the current logical line on an additional physical line. Performs a carriage return/line feed operation.	8.2.2.2
RETURN Key	Enters a typed line to the system, results in a carriage return/line feed operation at the terminal.	4.9.6
RUBOUT Key	Deletes the last character typed on that physical line. Erased characters are shown on the teleprinter between back slashes.	8.1.3
TAB or CTRL/I	Performs a tabulation to the next of nine tab stops (eight spaces apart) which form the terminal printing line.	8.2.2.3
CTRL/L	Generates FORM FEED character and results in four line feed operations at the terminal.	8.2.2

## **Appendix C**

### **Error Messages**

Messages in RSTS/E are generated for BASIC-PLUS errors\* and RSTS/E errors. To avoid confusion, both types of messages are called RSTS/E error messages and are described as one set. The BASIC-PLUS errors cover compiler and run time conditions such as a violation of the syntax rules (?SYNTAX ERROR) and referencing an element of an array beyond the defined limits (?SUBSCRIPT OUT OF RANGE). The RSTS/E errors involve operating system conditions such as failing to locate the file or account specified (?CAN'T FIND FILE OR ACCOUNT) and requesting the hardware to perform a function for which it is not ready (?DEVICE HUNG OR WRITE LOCKED).

In most cases, if no error trapping is being done (that is, an ON ERROR GOTO statement is not in effect), BASIC-PLUS stops running the program. It prints the error message and the line number of the BASIC-PLUS statement that was being executed when the error occurred. The following sample printout shows the procedure:

```
10      OPEN 'Z' FOR INPUT AS FILE 1%
RUNNH
?CAN'T FIND FILE OR ACCOUNT AT LINE 10
READY
```

As the READY message indicates, control returns to the system.

---

\*Different messages are generated while a job is operating under run-time systems other than BASIC-PLUS. Such run-time systems are those for BASIC-PLUS-2 and FORTRAN-IV. For these error messages, consult the appropriate User's Guides.

An exception to this procedure occurs when an INPUT statement is being executed at the job's console terminal and error trapping is not in effect. The system generates the error message and executes the statement again as shown in the sample printout below:

```
10      ON ERROR GOTO 0 \  INPUT 'INTEGER VALUE';A%
RUNNH
INTEGER VALUE? C
%DATA FORMAT ERROR AT LINE 10
INTEGER VALUE?
```

With error trapping disabled at line 10, an invalid response to the INPUT statement causes the system to print the error message, clear the error condition, and execute the statement again.

Associated with each message is an error variable called ERR. Whenever an error occurs with trapping in effect, the system checks the error variable which is a decimal number in the range 0 to 127. An error with a number between 1 and 70 causes the system to transfer control to the line number indicated in the ON ERROR GOTO statement. The system does not print the error message. Your program is able to check the ERR variable and perform a recovery procedure. If the error number is between 71 and 127, the system does not transfer control to the recovery routine but prints the message and returns control to the system. (Error number 0 is reserved to identify the system installation name.) Note that the BASIC-PLUS-2 Run-Time System uses the error numbers 128 through 255; refer to the *RSTS/E BASIC-PLUS-2 User's Guide*.

Because a BASIC-PLUS program can recover from certain errors, this appendix lists errors in two categories - recoverable and non-recoverable. The recoverable error messages are listed in ascending order of their related error numbers. A program can use these error numbers to differentiate errors. Non-recoverable errors are in alphabetical order without error numbers because a program can not use these numbers in an error handling routine.

The first character position of each message indicates the severity of the error. Table C-1 describes this standard.

**Table C-1: Severity Standard in Error Messages**

Character	Severity	Meaning
%	Warning	Execution of the program can continue but may not generate the expected results.
?	Fatal	Execution cannot continue unless you remove the cause of the error. No space or tab is allowed after the ?.
	Information	A message beginning with neither a question mark nor a percent is for information only.

The severity indication is examined by BATCH to determine the severity of the error.

In the descriptions of error messages, certain abbreviations, as shown in Table C-2, denote special characteristics of the error.

**Table C-2: Special Abbreviations for Error Descriptions**

Abbreviation	Meaning
(C)	Continue. If an ON ERROR GOTO statement is not in effect, execution continues but with the conditions described.
(SPR)	Software Performance Report. This error should occur only under the conditions described. If it occurs under any other conditions, submit an SPR to DIGITAL and document the conditions under which the error occurred. Instructions for filling out the SPR are given in Section C.3.

An error whose description is accompanied by the abbreviation (C) indicates an exception to the error trapping procedure. If such an error occurs in a program with no error trapping in effect, BASIC-PLUS prints the error message and line number but continues running the program. The following sample printout shows the procedure:

```
100      ON ERROR GOTO 0 \ A% = 32768.
200      PRINT A%
RUNNH
%INTEGER ERROR AT LINE 100
 0
READY
```

The INTEGER ERROR is generated at line 100 by the attempt to compute a value outside the range for integers. After the error message is printed, processing continues but with the conditions described in the error meaning. 0 is substituted for the erroneously computed value.

The number of RSTS/E error messages is restricted to 127. Because of this restriction, certain error messages have multiple meanings. The specific meaning of an error message depends on the operation being performed when the error condition occurs. For example, if the system attempts a file access and the designated file can not be located, RSTS/E generates the ?CAN'T FIND FILE OR ACCOUNT error (ERR=5). That same error condition, however, applies to other, generically similar access operations. Thus, if a program attempts to send a message to another program and the proper entry is not found in the system table of eligible receivers, RSTS/E returns error number 5. Though the second failure does not involve a file access error, it too is classified as an access failure.

Certain RSTS/E errors, although classified as user recoverable, are not capable of being trapped by a program. Table C-3 lists such errors.

These errors involve special conditions which a program cannot control and which ought not to occur on a normal system. For example, the ?DISK ERROR DURING SWAP error indicates a hardware problem. The system

does not return control to the program. The error condition itself, however, can be either transient or recurring. Such errors should be brought to the attention of the system manager for further investigation. The errors are recoverable in the strict sense that the Monitor can take corrective action but the BASIC-PLUS Run-Time System does not return control to the user program.

**Table C-3: Non-Trappable Errors in Recoverable Class**

ERR	Message Printed
34	?RESERVED INSTRUCTION TRAP
36	?SP STACK OVERFLOW
37	?DISK ERROR DURING SWAP
38	?MEMORY PARITY FAILURE

## C.1 User Recoverable

Message and Meaning	Error Code
(SYSTEM INSTALLATION NAME) The error code 0 is associated with the system installation name and is used by system programs to print identification lines.	0
?BAD DIRECTORY FOR DEVICE The directory of the device referenced is in an unreadable format. The magnetic tape label format on tape differs from the system-wide default format, the current job default format, or the format specified in the OPEN statement. Use the ASSIGN command to set the correct format default or change the format specification in the MODE option of the OPEN statement.	1
?ILLEGAL FILE NAME The filename specified is not acceptable. It contains unacceptable characters or the filename specification format has been violated. The CCL command to be added begins with a number or contains a character other than A through Z, 0 through 9 and commercial at (@).	2

Message and Meaning	Error Code
?ACCOUNT OR DEVICE IN USE Reassigning or dismounting of the device cannot be done because the device is open or has one or more open files. The account to be deleted has one or more files and must be zeroed before being deleted. The runtime system to be deleted is currently loaded in memory and in use. Output to a pseudo keyboard cannot be done unless the device is in KB wait state. An echo control field cannot be declared while another field is currently active. The CCL command to be added already exists.	3
?NO ROOM FOR USER ON DEVICE Storage space allowed for the current user on the device specified has been used, or the device as a whole is too full to accept further data, or a contiguous file was specified and there is insufficient contiguous space.	4
?CAN'T FIND FILE OR ACCOUNT The file or account number specified was not found on the device specified. The CCL command to be deleted does not exist.	5
?NOT A VALID DEVICE The device specification supplied is not valid for one of the following reasons. The unit number or its type is not configured on the system. The specification is logical and untranslatable because a physical device is not associated with it.	6
?I/O CHANNEL ALREADY OPEN An attempt was made to open one of the twelve I/O channels which had already been opened by the program. (SPR)	7
?DEVICE NOT AVAILABLE The specified device exists on the system but an attempt to ASSIGN or OPEN it is prohibited for one of the following reasons. The device is currently reserved by another job. The device requires privileges for ownership and you do not have privilege. The device or its controller has been disabled by the system manager. The device is a keyboard line for pseudo keyboard use only.	8
?I/O CHANNEL NOT OPEN An attempt was made to perform I/O on one of the twelve channels which has not been previously opened in the program.	9

Message and Meaning	Error Code
?PROTECTION VIOLATION You were prohibited from performing the requested operation because the kind of operation was illegal (such as input from a line printer) or because you did not have the privileges necessary (such as deleting a protected file).	10
?END OF FILE ON DEVICE Attempt to perform input beyond the end of a data file; or a BASIC source file is called into memory and is found to contain no END statement.	11
?FATAL SYSTEM I/O FAILURE An I/O error has occurred on the system level. You have no guarantee that the last operation has been performed. This error is caused by a hardware condition. Report such occurrences to the system manager. (See the discussion at the beginning of Appendix C.)	12
?USER DATA ERROR ON DEVICE One or more characters may have been transmitted incorrectly due to a parity error, bad punch combination on a card, or similar error.	13
?DEVICE HUNG OR WRITE LOCKED Check hardware condition of device requested. Possible causes of this error include a line printer out of paper or high-speed reader being off-line.	14
?KEYBOARD WAIT EXHAUSTED Time requested by WAIT statement has been exhausted with no input received from the specified keyboard.	15
?NAME OR ACCOUNT NOW EXISTS An attempt was made to rename a file with the name of a file which already exists, or an attempt was made by the system manager to insert an account number which is already within the system.	16
?TOO MANY OPEN FILES ON UNIT Only one open DECTape output file is permitted per DECTape drive. Only one open file per magnetic tape drive is permitted.	17
?ILLEGAL SYS( ) USAGE Illegal use of the SYS system function.	18
?DISK BLOCK IS INTERLOCKED The requested disk block segment is already in use (locked) by some other user.	19

Message and Meaning	Error Code
?PACK IDS DON'T MATCH The identification code for the specified disk pack does not match the identification code already on the pack.	20
?DISK PACK IS NOT MOUNTED No disk pack is mounted on the specified disk drive.	21
?DISK PACK IS LOCKED OUT The disk pack specified is mounted but temporarily disabled.	22
?ILLEGAL CLUSTER SIZE The specified cluster size is unacceptable. The cluster size must be a power of 2. For a file cluster, the size must be equal to or greater than the pack cluster size and must not be greater than 256. For a pack cluster, the size must be equal to or greater than the device cluster size and must not be greater than 16. The device cluster size is fixed by type.	23
?DISK PACK IS PRIVATE You do not have access to the specified private disk pack.	24
?DISK PACK NEEDS 'CLEANING' Non-fatal disk mounting error; use the CLEAN operation in UTILITY or the ONLCLN (On Line Clean) program.	25
?FATAL DISK PACK MOUNT ERROR Fatal disk mounting error. Disk cannot be successfully mounted.	26
?I/O TO DETACHED KEYBOARD I/O was attempted to a hung up dataset or to the previous, but now detached, console keyboard for the job.	27
?PROGRAMMABLE ^C TRAP A CTRL/C combination was typed while an ON ERROR GOTO statement was in effect and programmable CTRL/C trapping was enabled.	28
?CORRUPTED FILE STRUCTURE Fatal error in CLEAN operation.	29
?DEVICE NOT FILE STRUCTURED An attempt is made to access a device, other than a disk, DECTape, or magnetic tape device, as a file structured device. This error occurs, for example, when you attempt to gain a directory listing of a non-directory device.	30

Message and Meaning	Error Code
---------------------	------------

**?ILLEGAL BYTE COUNT FOR I/O**

31

The buffer size specified in the RECORDSIZE option of the OPEN statement or in the COUNT option of the PUT statement is not a multiple of the block size of the device being used for I/O, or is illegal for the device. An attempt is made to run a compiled file which has improper size due to incorrect transfer procedure.

**?NO BUFFER SPACE AVAILABLE**

32

You access a file and the Monitor requires one small buffer to complete the request but one is not currently available. If the program is sending messages, two conditions are possible. The first occurs when a program sends a message and the receiving program has exceeded the pending message limit. The second occurs when a sending program attempts to send a message and a small buffer is not available for the operation.

**?ODD ADDRESS TRAP**

33

This error occurs when an attempt is made to reference a non-existent address, reference an odd address using a word instruction, or perform a PEEK function with an odd or non-existent parameter.

**?RESERVED INSTRUCTION TRAP**

34

An attempt is made to execute an illegal or reserved instruction or an FPP instruction when floating-point hardware is not available. (See discussion at beginning of Appendix C.)

**?MEMORY MANAGEMENT VIOLATION**

35

This hardware error occurs when an illegal Monitor address is specified using the PEEK function. Generation of the error message in situations other than using PEEK is cause for an SPR.

**?SP STACK OVERFLOW**

36

An attempt to extend the hardware stack beyond its legal size is encountered. (See discussion at beginning of Appendix C.) (SPR)

**?DISK ERROR DURING SWAP**

37

A hardware error occurs when your job is swapped into or out of memory. The contents of the job area are lost but the job remains logged into the system and is reinitialized to run the NONAME program. Report such occurrences to the system manager. (See discussion at beginning of Appendix C.)

Message and Meaning	Error Code
?MEMORY PARITY FAILURE A parity error was detected in the memory occupied by this job. (See discussion at beginning of Appendix C).	38
?MAGTAPE SELECT ERROR When access to a magnetic tape drive was attempted, the selected unit was found to be off line.	39
?MAGTAPE RECORD LENGTH ERROR When performing input from magnetic tape, the record on tape was found to be longer than the buffer designated to handle the record.	40
?NON-RES RUN-TIME SYSTEM The run-time system referenced has not been loaded into memory and is therefore non-resident.	41
?VIRTUAL BUFFER TOO LARGE Virtual core buffers must be 512 bytes long.	42
?VIRTUAL ARRAY NOT ON DISK A non-disk device is open on the channel upon which the virtual array is referenced.	43
?MATRIX OR ARRAY TOO BIG In-core array size is too large.	44
?VIRTUAL ARRAY NOT YET OPEN An attempt was made to use a virtual array before opening the corresponding disk file.	45
?ILLEGAL I/O CHANNEL An attempt was made to open a file on an I/O channel outside the range of the integer numbers 1 to 12.	46
?LINE TOO LONG An attempt was made to input a line longer than 255 characters (which includes a line terminator). Buffer overflows.	47
%FLOATING POINT ERROR An attempt was made to use a computed floating-point number outside the range 1E-38 <n>1E38 excluding zero. If no transfer to an error handling routine is made, zero is returned as the floating-point value. (C)	48
%ARGUMENT TOO LARGE IN EXP Acceptable arguments are within the approximate range -89<arg<+88. The value returned is zero. (C)	49

Message and Meaning	Error Code
<b>%DATA FORMAT ERROR</b> A READ or INPUT statement detected data in an illegal format. For example, 1..2 is an improperly formed number, and 1.3 is an improperly formed integer, and X“ is an illegal string. (C)	50
<b>%INTEGER ERROR</b> An attempt was made to use a computed integer outside the range -32768<n<32767. For example, an attempt is made to assign to an integer variable a floating-point number outside the integer range. If no transfer to an error handling routine is made, zero is returned as the integer value. (C)	51
<b>?ILLEGAL NUMBER</b> Integer overflow or underflow or floating-point overflow. The range for integers is -32768 to +32767; for floating-point numbers, the upper limit is 1E38. (For floating-point underflow, the ?FLOATING POINT ERROR (ERR=48) is generated.)	52
<b>%ILLEGAL ARGUMENT IN LOG</b> Negative or zero argument to LOG function. Value returned is the argument as passed to the function. (C)	53
<b>%IMAGINARY SQUARE ROOTS</b> An attempt was made to take square root of a number less than zero. The value returned is the square root of the absolute value of the argument. (C)	54
<b>?SUBSCRIPT OUT OF RANGE</b> An attempt was made to reference an array element beyond the number of elements created for the array when it was dimensioned.	55
<b>?CAN'T INVERT MATRIX</b> An attempt was made to invert a singular or nearly singular matrix.	56
<b>?OUT OF DATA</b> The DATA list was exhausted and a READ requested additional data.	57
<b>?ON STATEMENT OUT OF RANGE</b> The index value in an ON GOTO or ON GOSUB statement is less than one or greater than the number of line numbers in the list.	58
<b>?NOT ENOUGH DATA IN RECORD</b> An INPUT statement did not find enough data in one line to satisfy all the specified variables.	59

Message and Meaning	Error Code
?INTEGER OVERFLOW, FOR LOOP The integer index in a FOR loop attempted to go beyond 32766 or below -32767.	60
%DIVISION BY 0 Your program attempted to divide some quantity by zero. If no transfer is made to an error handler routine, a 0 is returned as the result. (C)	61
?NO RUN-TIME SYSTEM The run-time system referenced has not been added to the system list of run-time systems.	62
?FIELD OVERFLOWS BUFFER An attempt was made to use FIELD to allocate more space than exists in the specified buffer.	63
?NOT A RANDOM ACCESS DEVICE An attempt was made to perform random access I/O to a non-random access device.	64
?ILLEGAL MAGTAPE ( ) USAGE Improper use of the MAGTAPE function.	65
?MISSING SPECIAL FEATURE Your program employs a BASIC-PLUS feature not present on the given installation.	66
?ILLEGAL SWITCH USAGE A CCL command contains an error in an otherwise valid CCL switch. (For example, the /SI:n switch was used without a value for n or a colon; or more than one of the same type of CCL switch was specified.) A file specification switch is not the last element in a file specification or is missing a colon or an argument.	67

## C.2 Non-Recoverable

Message and Meaning
?ARGUMENTS DON'T MATCH Arguments in a function call do not match, in number or in type, the arguments defined for the function.
?BAD LINE NUMBER PAIR Line numbers specified in a LIST or DELETE command were formatted incorrectly.
?BAD NUMBER IN PRINT-USING Format specified in the PRINT-USING string cannot be used to print one or more values.

## **Message and Meaning**

### **?CAN'T CONTINUE**

Program was stopped or ended at a spot from which execution cannot be resumed.

### **?DATA TYPE ERROR**

Incorrect usage of floating-point, integer, or character string format variable or constant where some other data type was necessary.

### **?DEF WITHOUT FNEND**

A second DEF statement was encountered in the processing of a user function without an FNEND statement terminating the first user function definition.

### **?END OF STATEMENT NOT SEEN**

Statement contains too many elements to be processed correctly.

### **?ERROR TEXT LOOKUP FAILURE**

An I/O error occurred while the system was attempting to retrieve an error message. Possible cause could be the device containing the system error file (ERR.SYS) is off-line, or the system error file contains a bad block.

### **?EXECUTE ONLY FILE**

Attempt was made to add, delete or list a statement in a compiled (.BAC) format file.

### **?EXPRESSION TOO COMPLICATED**

This error usually occurs when parentheses have been nested too deeply. The depth allowable is dependent on the individual expression.

### **?FILE EXISTS-RENAME/REPLACE**

A file of the name specified in a SAVE command already exists. In order to save the current program under the name specified, use REPLACE, or use RENAME followed by SAVE.

### **?FNEND WITHOUT DEF**

An FNEND statement was encountered in your program without a previous function call having been executed.

### **?FNEND WITHOUT FUNCTION CALL**

A FNEND statement was encountered in your program without a previous DEF statement being seen.

### **?FOR WITHOUT NEXT**

A FOR statement was encountered in your program without a corresponding NEXT statement to terminate the loop.

### **?ILLEGAL CONDITIONAL CLAUSE**

Incorrectly formatted conditional expression.

### **?ILLEGAL DEF NESTING**

The range of one function definition crosses the range of another function definition.

## **Message and Meaning**

### **?ILLEGAL DUMMY VARIABLE**

One of the variables in the dummy variable list of a user-defined function is not a legal variable name.

### **?ILLEGAL EXPRESSION**

Double operators, missing operators, mismatched parentheses, or some similar error has been found in an expression.

### **?ILLEGAL FIELD VARIABLE**

The FIELD variable specified is unacceptable.

### **?ILLEGAL FN REDEFINITION**

An attempt was made to redefine a user function.

### **?ILLEGAL FUNCTION NAME**

An attempt was made to define a function with a function name not subscribing to the established format.

### **?ILLEGAL IF STATEMENT**

Incorrectly formatted IF statement.

### **?ILLEGAL IN IMMEDIATE MODE**

You issued a statement for execution in immediate mode which can only be performed as part of a program.

### **?ILLEGAL LINE NUMBER(S)**

Line number reference outside the range  $1 < n < 32767$ .

### **?ILLEGAL MODE MIXING**

String and numeric operations cannot be mixed.

### **?ILLEGAL STATEMENT**

An attempt was made to execute a statement that did not compile without errors.

### **?ILLEGAL SYMBOL**

An unrecognizable character was encountered. For example, a line consisting of a # character can cause this error.

### **?ILLEGAL VERB**

The BASIC verb portion of the statement cannot be recognized.

### **%INCONSISTENT FUNCTION USAGE**

A function is defined with a certain number of arguments but is elsewhere referenced with a different number of arguments. Fix the reference to match the definition and reload the program to reset the function definition.

### **%INCONSISTENT SUBSCRIPT USE**

A subscripted variable is being used with a different number of dimensions from the number with which it was originally defined.

## **Message and Meaning**

### **X(Y)K OF MEMORY USED**

Message printed by the LENGTH command. The value for x is the current size, to the nearest 1K-word increment, of the program in memory. The value for y is the size to which the program can expand, given the runtime system being used and the job's private memory size maximum set by the system manager.

### **?LITERAL STRING NEEDED**

A variable name was used where a numeric or character string was necessary.

### **?MATRIX DIMENSION ERROR**

An attempt was made to dimension a matrix to more than two dimensions, or an error was made in the syntax of a DIM statement.

### **?MATRIX OR ARRAY WITHOUT DIM**

A matrix or array element was referenced beyond the range of an implicitly dimensioned matrix.

### **?MAXIMUM MEMORY EXCEEDED**

During an OLD operation, the job's private memory size maximum was reached. While running a program, the system required more memory for string or I/O buffer space and the job's private memory size maximum or the system maximum (16K words for BASIC-PLUS) was reached.

### **?MODIFIER ERROR**

An attempt was made to use one of the statement modifiers (FOR, WHILE, UNTIL, IF, or UNLESS) incorrectly. An OPEN statement modifier, such as a RECORDSIZE, CLUSTERSIZE, FILESIZE, or MODE option, is not in the correct order.

### **?NEXT WITHOUT FOR**

A NEXT statement was encountered in your program without a previous FOR statement having been seen.

### **?NO LOGINS**

Message printed if the system is full and cannot accept additional users or if further logins are disabled by the system manager.

### **?NOT ENOUGH AVAILABLE MEMORY**

An attempt is made to load a non-privileged compiled program which is too large to run, given the job's private memory size maximum. The program must be made privileged to allow it to expand above a private memory size maximum; or the system manager must increase the job's private memory size maximum to accommodate the program.

### **?NUMBER IS NEEDED**

A character string or variable name was used where a number was necessary.

### **?1 OR 2 DIMENSIONS ONLY**

An attempt was made to dimension a matrix to more than two dimensions.

## **Message and Meaning**

### **?ON STATEMENT NEEDS GOTO**

A statement beginning with ON does not contain a GOTO or GOSUB clause.

### **?PLEASE SAY HELLO**

Message printed by the LOGIN system program. While not logged into the system, you typed something other than a legal, logged-out command to the system.

### **?PLEASE USE THE RUN COMMAND**

A transfer of control (as in a GOTO, GOSUB or IF GOTO statement) cannot be performed from immediate mode.

### **?PRINT-USING BUFFER OVERFLOW**

Format specified contains a field too large to be manipulated by the PRINT-USING statement.

### **?PRINT-USING FORMAT ERROR**

An error was made in the construction of the string used to supply the output format in a PRINT-USING statement.

### **?PROGRAM LOST-SORRY**

A fatal system error has occurred which caused your program to be lost. This error can indicate hardware problems or use of an improperly compiled program. Consult the system manager or the discussion of such errors in the *RSTS/E System Manager's Guide*.

### **?REDIMENSIONED ARRAY**

Usage of an array or matrix within your program has caused BASIC-PLUS to redimension the array implicitly.

### **?RESUME AND NO ERROR**

A RESUME statement was encountered where no error had occurred to cause a transfer into an error handling routine via the ON ERROR GOTO statement.

### **?RETURN WITHOUT GOSUB**

RETURN statement encountered in your program without a previous GOSUB statement having been executed.

### **%SCALE FACTOR INTERLOCK**

An attempt was made to execute a program or source statement with the current scale factor. The program runs but the system uses the scale factor of the program in memory rather than the current scale factor. Use REPLACE and OLD or recompile the program to run with the current scale factor. (C)

### **?STATEMENT NOT FOUND**

Reference is made within the program to a line number which is not within the program.

## **Message and Meaning**

### **STOP**

STOP statement was executed. You can usually continue program execution by typing CONT and the RETURN key.

### **?STRING IS NEEDED**

A number or variable name was used where a character string was necessary.

### **?SYNTAX ERROR**

BASIC-PLUS statement was incorrectly formatted.

### **?TOO FEW ARGUMENTS**

The function has been called with a number of arguments not equal to the number defined for the function.

### **?TOO MANY ARGUMENTS**

A user-defined function may have up to five arguments.

### **?UNDEFINED FUNCTION CALLED**

BASIC-PLUS interpreted some statement component as a function call for which there is no defined function (system or user).

### **?WHAT?**

A command or immediate mode statement entered to BASIC-PLUS could not be processed. Illegal verb or improper format usually causes this error.

### **?WRONG MATH PACKAGE**

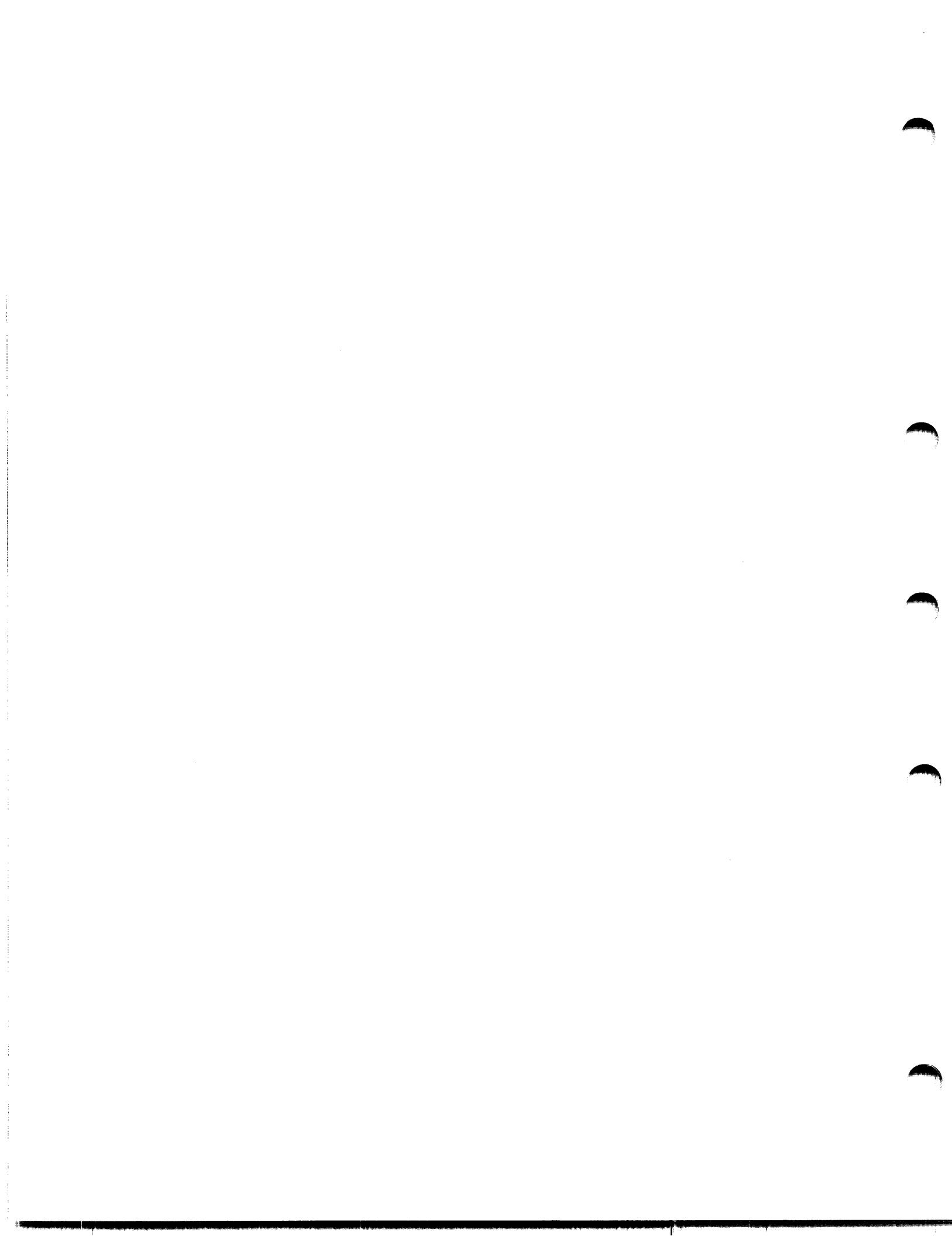
Program was compiled on a system with either the 2-word or 4-word math package and an attempt is made to run the program on a system with the opposite math package. Recompile the program using the math package of the system on which it will be run.

## **C.3 Software Performance Report Guidelines**

The Software Performace Report (SPR) forms enable you to report problems with DIGITAL software. Note that prior to submitting an SPR, ensure that the problem has not been corrected in the Release Notes or the Software Dispatch. To speed response and prevent processing delays with an SPR, you should describe the problem as completely as possible. The following list contains the minimal information that you should include on the SPR.

- Complete hardware configuration; including CPU type, system disk, amount and type of memory, hardware option (such as floating-point processor), and system peripherals.
- Monitor options present on the system, including math package and BASIC-PLUS options.

- Program name, version number, and edit level (generally found on line 1010 of the program listing), and any optional patches included in the program. Also note the account(s) under which the program failed and whether it is privileged or non-privileged.
- The PRIORITY and SWAP MAX under which the program was running.
- A terminal printout of any relevant command strings and input data.
- A list of any modifications made to the program.
- A listing of any applicable log files.
- If a crash dump is submitted, it must include system load map listings, "CORE DUMP OF MONITOR" output of the crash dump analysis program, and the CRASH.SYS file and installed Monitor .SIL file in machine-readable format.



## **Appendix D**

### **BASIC-PLUS Character Set**

#### **D.1 BASIC-PLUS Character Set**

Program statements are composed of individual characters. Allowable characters come from the following character set:

A THROUGH Z    SPACE

0 THROUGH 9    TAB

and the following special symbols and keys:

Key	Use
\$	Used in specifying string variables, or as the system library file designator.
%	Used in specifying integer variables. Also denotes account [1,4].
' "	Used to delimit string constants, i.e., text strings.
!	Begins comment part of a line. Also denotes account [1,3].
:	Separates multiple statements on one line.
\	Separates multiple statements on one line as the colon (:) also does.
#	Denotes a device or file # name, or is used as an output format effector. Also denotes account number using current project number with a programmer number of 0.
,	Output format effector and list terminator.
;	Output format effector.
&	Denotes account [1,5].
@	Denotes the assignable account.
LF	When used at the end of a line, indicates that the current statement is continued on the next line.

Key	Use
()	Used to group arguments in an arithmetic expression. Also may be used to group project-programmer number.
[]	Used to group project-programmer number.
<>	Used to delimit file protection codes.
+*/	Arithmetic operators.
=	Replacement operator. Logical equivalence operator.
<	Logical "less than" operator.
>	Logical "greater than" operator.
==	Logical "approximately equal to" operator.

## D.2 ASCII Character Codes

Decimal Value	ASCII Character	RSTS Usage	Decimal Value	ASCII Character	RSTS Usage	Decimal Value	ASCII Character	RSTS Usage
0	NUL	FILL character	43	+		86	V	
1	SOH		44	,	COMMA	87	W	
2	STX		45	-		88	X	
3	ETX	CTRL/C	46	.		89	Y	
4	EOT		47	/		90	Z	
5	ENQ		48	0		91	[	
6	ACK		49	1		92	\	Backslash
7	BEL	BELL	50	2		93	]	
8	BS		51	3		94	^ or t	
9	HT	HORIZ.TAB	52	4		95	— or ←	Grave accent
10	LF	LINE FEED	53	5		96	a	
11	VT	VERT.TAB	54	6		97	b	
12	FF	FORM FEED	55	7		98	c	
13	CR	CAR.RET	56	8		99	d	
14	SO		57	9		100	e	
15	SI	CTRL/O	58	:		101	f	
16	DLE		59	;		102	g	
17	DC1	CTRL/Q	60	<		103	h	
18	DC2		61	=		104	i	
19	DC3	CTRL/S	62	>		105	j	
20	DC4		63	?		106	k	
21	NAK	CTRL/U	64	@		107	l	
22	SYN		65	A		108	m	
23	ETB		66	B		109	n	
24	CAN		67	C		110	o	
25	EM		68	D		111	p	
26	SUB	CTRL/Z	69	E		112	q	
27	ESC	ESCAPE*	70	F		113	r	
28	FS		71	G		114	s	
29	GS		72	H		115	t	
30	RS		73	I		116	u	
31	US		74	J		117	v	
32	SP	SPACE	75	K		118	w	
33	!		76	L		119	x	
34	"		77	M		120	y	
35	#		78	N		121	z	
36	\$		79	O		122	{	
37	%		80	P		123		Vertical Line
38	&		81	Q		124	~	Tilde
39	,	APOSTROPHE	82	R		125		
40	(		83	S		126		
41	)		84	T		127	DEL RUBOUT	
42	*		85	U				

\*ALTMODE (ASCII 125) or PREFIX (ASCII 126) keys which appear on some terminals are translated internally into ESCAPE.

### D.3 Radix-50 Character Set

Character	ASCII Octal Equivalent	Radix-50 Equivalent
space	40	0
A-Z	101-132	1-32
\$	44	33
.	56	34
unused		35
0-9	60-71	36-47

The maximum Radix-50 value is:

$$47*50^2 + 47*50 + 47 = 174777$$

The following table provides a convenient means of translating between the ASCII character set and its Radix-50 equivalents. For example, given the ASCII string X2B, the Radix-50 equivalent is (arithmetic is performed in octal):

$$\begin{array}{ll} X & = 113000 \\ 2 & = 002400 \\ B & = 000002 \\ \hline X2B & = 115402 \end{array}$$

Radix-50 Character/Position Table

Single Character or First Character	Second Character	Third Character
A 003100	A 000050	A 000001
B 006200	B 000120	B 000002
C 011300	C 000170	C 000003
D 014400	D 000240	D 000004
E 017500	E 000310	E 000005
F 022600	F 000360	F 000006
G 025700	G 000430	G 000007
H 031000	H 000500	H 000010
I 034100	I 000550	I 000011
J 037200	J 000620	J 000012
K 042300	K 000670	K 000013

(continued on next page)

<b>Single Character or First Character</b>	<b>Second Character</b>		<b>Third Character</b>
L 045400	L 000740	L 000014	
M 050500	M 001010	M 000015	
N 053600	N 001060	N 000016	
O 056700	O 001130	O 000017	
P 062000	P 001200	P 000020	
Q 065100	Q 001250	Q 000021	
R 070200	R 001320	R 000022	
S 073300	S 001370	S 000023	
T 076400	T 001440	T 000024	
U 101500	U 001510	U 000025	
V 104600	V 001560	V 000026	
W 107700	W 001630	W 000027	
X 113000	X 001700	X 000030	
Y 116100	Y 001750	Y 000031	
X 121200	Z 002020	Z 000032	
\$ 124300	\$ 002070	\$ 000033	
.	.	.	000034
unused 132500	unused 002210	unused 000035	
0 135600	0 002260	0 000036	
1 140700	1 002330	1 000037	
2 144000	2 002400	2 000040	
3 147100	3 002450	3 000041	
4 152200	4 002520	4 000042	
5 155300	5 002570	5 000043	
6 160400	6 002640	6 000044	
7 163500	7 002710	7 000045	
8 166600	8 002760	8 000046	
9 171700	9 003030	9 000047	

# Index

Where more than one page number appears for an entry, the defining entry is in bold type.

## A

Account  
associate with device and logical name, 4-6  
associate with physical device, 4-4  
cancel logical assignment of, 4-10  
logical assignment of, 4-9  
restore files to another, 17-15  
zero directory with PIP, 16-27

Account, user  
obtaining data on, 14-10

Account characters  
special, 11-4

Account number, 1-1, 11-3.  
*See also Project-programmer number*

Accounts  
designated by special characters, 11-4

ALTMODE key, 4-13

ANSI labels, 5-2  
initialize with PIP, 16-27

APPEND command, 8-6  
file specification in, 8-7  
merge two programs, 8-6

Arithmetic, scaled  
using, 7-14

ASCII, formatted, 16-19

ASCII character codes, D-2

ASCII character set, terminal with 1968, 19-5

ASCII characters  
transmit on terminal, 19-3

ASCII files  
translate to RMS, 16-21

ASCII mode, translate to EBCDIC, 18-4

ASSIGN command, 4-1  
assigning logical name, 4-4  
assigning multiple logical names, 4-4  
assigning physical name, 4-5  
associate device and account, 4-6  
changing default protection code, 5-1  
changing tape label default, 5-2  
disk pack logical name, 4-9  
format of, 4-1  
logical account assignment, 4-9  
reserve logically named device, 4-5  
reserving a device, 4-1

Asterisk  
as wildcard, 11-6

## At sign

account association, 4-10  
cancel account association, 4-10

ATTACH command, 13-3  
at a logged in terminal, 13-4

Attributes, file, 11-11, 11-12.  
*See also File attributes*

octal, 11-13  
symbolic, 11-12

## B

BACKUP dialogue, 17-1

Backup dialogue, 17-7, 17-8  
backup mode, 17-9  
in batch control file, 17-7  
default answers to, 17-7  
obtaining prompt explanations, 17-7  
prompts, 17-7

BACKUP errors  
bad block, 17-30  
skip over, 17-30

Backup file specification, 17-5  
access field, 17-5  
comments in, 17-6  
continuing, 17-6  
conventions in, 17-6  
creation field, 17-5  
spacing, 17-6  
time of day in, 17-5

BACKUP interruption commands, 17-19t

Backup job  
running a, 17-7

Backup labels, 17-23

Backup listing file, 17-10

Backup mode, 17-1  
bad block during, 17-30  
dialogue summary, 17-9  
file greater than 65535 blocks, 17-11  
listing file, 17-10  
placed files, 17-11  
primary index file, 17-10  
selecting, 17-8

Backup operation  
exclude files from, 17-5

BACKUP program, 17-1  
in a batch stream, 17-1

BACKUP program, (Cont.)  
account location, 17-4  
backup mode, 17-1  
under batch, 17-4  
CREATION keyword, 17-5  
deletion errors, 17-30  
dialogue command errors, 17-25  
error handling, 17-25  
error handling routines, 17-30  
EXCEPT keyword, 17-6  
files to disk, 17-2  
identification header, 17-4  
indirect command file, 17-8  
informational messages, 17-31  
install the, 17-4  
interruption command errors, 17-25, 17-27  
interruption commands, 17-19  
list dialogue, 17-17  
list mode, 17-4  
listing errors, 17-30  
loadindex dialogue, 17-16  
loadindex mode, 17-2, 17-3  
mode prompt, 17-4  
modes, 17-4  
mount procedures, 17-24  
primary index file, 17-2  
printing help file, 17-8  
processing errors, 17-25, 17-29  
restore dialogue, 17-12  
restore mode, 17-2  
running, 17-4  
running from indirect command file, 17-20  
running under batch, 17-7  
/SAVE option, 17-8  
secondary index file, 17-2  
selection errors, 17-29  
transfer errors, 17-30  
transfer process, 17-3  
volume mount errors, 17-25, 17-27  
work file, 17-1  
BACKUP program, file specification in, 17-4  
Backup set, 17-1  
creating separate index file, 17-3  
directory from index, 17-18  
directory of, 17-2  
expiration date of, 17-23  
identifier for, 17-23  
index file, 17-2  
index volume, 17-3  
list index, 17-4  
Backup set index  
permanent copy of, 17-10  
Backup volumes  
dismounting, 17-23

Backup volumes, (Cont.)  
mount procedures, 17-24  
mounting, 17-23  
Backup work file, 17-10  
estimating length of, 17-10  
Bad block files, RT-11  
transfer with FIT program, 16-35  
Bad blocks  
in COPY operation, 16-29  
BAS programs, 6-1  
BASIC-PLUS  
command summary, B-1 to B-3  
control scaled arithmetic, 7-14  
errors, C-1  
function summary, A-2 to A-6  
immediate mode, 9-1  
language summary, A-1  
operator summary, A-2  
statement summary, A-6 to A-13  
system commands, 6-1, 6-2t  
variable type summary, A-1  
BASIC-PLUS, character set, D-1  
BASIC-PLUS-2 compiler, calling with BATCH, 21-10  
BASIC-PLUS command  
as CCL command, 12-1  
BASIC-PLUS commands  
overview of, 6-2  
BASIC-PLUS compiler, calling with BATCH, 21-10  
BASIC-PLUS translator, 12-1  
Batch  
as spooling device, 20-1  
BACKUP program under, 17-4  
command field, 21-2  
command field delimiter, 21-3  
running BACKUP under, 17-7  
special characters, 21-4  
specification field, 21-3  
Batch command, \$BASIC, 21-7, 21-10  
call BASIC compiler, 21-10  
switches, 21-10  
Batch command, \$COPY, 21-8, 21-13  
copy files, 21-13  
Batch command, \$CREATE, 21-8, 21-14  
create ASCII file, 21-14  
Batch command, \$DATA, 21-7, 21-15  
enter data to program, 21-15  
Batch command, \$DELETE, 21-8, 21-12  
delete specified files, 21-12  
Batch command, \$DIRECTORY, 21-8, 21-13  
directory listing, 21-13  
Batch command, \$DISMOUNT, 21-8, 21-17  
nullify device assignment, 21-17

Batch command, \$EOD, 21-7, 21-15  
end of data, 21-15

Batch command, \$EOJ, 21-7, 21-9  
mark end of job, 21-9

Batch command, \$FORTRAN, 21-8, 21-19  
execute FORTRAN, 21-19  
switches, 21-19

Batch command, \$JOB, 21-7, 21-8  
begin a job, 21-8  
switches, 21-8

Batch command, \$MESSAGE, 21-7, 21-15  
message to operator, 21-15

Batch command, \$MOUNT, 21-8, 21-16  
device specifications, 21-17  
mount message to operator, 21-16  
mountable devices, 21-17  
switches, 21-16

Batch command, \$PRINT, 21-8, 21-13  
print file contents, 21-13

Batch command, \$RUN, 21-7, 21-14  
execute system program, 21-14

Batch command, \$SORT, 21-8, 21-18  
execute SORT program, 21-18  
file specifications, 21-18  
switches, 21-18

Batch commands  
default file extensions, 21-6  
syntax errors, 21-22  
utility, 21-12

Batch control file  
BACKUP dialogue in, 17-7

Batch control language  
comments, 21-4  
continuation lines, 21-4  
elements of, 21-1  
file extensions, 21-5  
file specification defaults, 21-6  
file specifications, 21-5  
separate fields in, 21-4  
statements, 21-1, 21-3  
syntax rules, 21-3

Batch input, 21-1

Batch job, 21-20

Batch log file, 21-22

Batch operating procedures, 21-20

Batch processing, 21-1, 21-21

Batch processor, 20-4, 21-1  
backup job under, 17-7  
file specification, 21-5  
utility functions of, 21-12

BATCH program, 21-1  
assign a device, 21-8  
begin a job, 21-7

Batch program, (Cont.)  
begin data, 21-7  
copy files, 21-8  
create a file, 21-8  
deassign a device, 21-8  
delete files, 21-8  
end a job, 21-7  
end data, 21-7  
error messages, 21-23  
error procedures, 21-22  
execute a program, 21-7  
execute BASIC, 21-7  
execute FORTRAN, 21-8  
execute SORT, 21-8  
execute system function, 21-7  
list directory, 21-8  
queue a file, 21-8  
requesting job run, 21-20  
send a message, 21-7

BCD keyboard, 23-29

Binary, formatted, 16-19

Binary mode  
terminal, 19-4

Blank extension, 11-6

Block size  
specifying in PIP file transfer, 16-14

Blocking factor, 18-5

BPCREF program, 15-18  
command formats, 15-18t  
error messages, 15-21  
header line, 15-18  
listing global variable references, 15-20  
listing local variable references, 15-21  
/NOHEADER switch, 15-20  
output file, 15-20  
output listing contents, 15-19  
queue an output file, 15-19  
response formats, 15-18  
restrictions on, 15-18  
running, 15-18  
statistical data, 15-20  
switches, 15-19  
variable references, 15-20

BREAK key  
as CTRL/C, 4-12

Bucket size  
determining file attribute, 11-12

Buffer  
line printer, 23-8  
status, 14-6

BYE command, 1-4, 13-7  
file deletion, 1-4  
response to, 13-7

## C

Caching  
  DIRECT program, 15-3  
CALL/360 BASIC keyboard, 23-30  
Card punch, 2-7  
Card reader, 2-7, 23-1, 23-6  
  controls, 23-7  
  programmed control, 23-6  
  use of, 23-7  
Cards, 2-7  
  advantages of, 2-7  
  disadvantages of, 2-7  
Carriage control  
  determining file attribute, 11-12  
  FORTRAN, 16-19  
CATALOG command, 7-13  
  directory printed with, 7-13  
  list files in directory, 7-13  
Cathode Ray Tube (CRT), 23-19  
CCL (Concise Command Language), 12-1  
CCL command  
  advantages of, 12-1  
  BASIC-PLUS command as, 12-1  
  cautions on typing, 12-1  
  DIRECT as, 15-1, 15-6  
  embedded spaces in, 12-1  
  FLINT as, 18-12  
  HELP as, 12-3  
  PIP as, 16-1  
  programs run by, 12-1  
  QUEUE as, 21-21  
  running QUE program, 20-14  
  standard, 12-1  
  SUBMIT as, 21-21  
  SYSTAT as, 14-9  
  TTYSET as, 19-12  
CCL commands  
  summary of, 10-3  
CCL translator, 12-1  
CCONT command, 9-3  
Central Processing Unit. *See CPU*  
CHAIN statement, 8-7  
  to access QUE, 20-11  
filename and extension in, 8-8  
format of, 8-8  
in immediate mode, 8-8  
with line number specification, 8-8  
transfer program control, 8-7  
Character codes, ASCII, D-2  
Character set  
  BASIC-PLUS, D-1  
  line printer, 23-8  
  Radix-50, D-3

Characters  
  erase incorrect, 7-3  
Cluster size  
  establish minimum, 11-9  
  recommended, 11-9  
  specifying in PIP file transfer, 16-15  
  specifying negative, 11-9  
CLUSTERSIZE option, 11-9  
  establish minimum cluster size, 11-9  
  format of, 11-9  
  negative cluster size, 11-9  
Command  
  APPEND, 8-6  
  ASSIGN, 4-1  
  ATTACH, 13-3  
  BYE, 1-4, 13-7  
  CATALOG, 7-13  
  CCL, 12-1  
  CCONT (privileged), 9-3  
  COMPILE, 7-8  
  CONT, 9-3  
  DEASSIGN, 4-3  
  DELETE, 8-3  
  DIRECT, 1-5  
  EXIT, 5-3  
  EXTEND, 8-9  
  HELLO, 1-2  
  KEY, 4-11  
  LENGTH, 7-12  
  LIST, 8-1  
  LISTNH, 8-2  
  MOUNT, 4-8, 19-13  
  NEW, 7-1  
  NOEXTEND, 8-9  
  OLD, 7-5  
  REASSIGN, 4-3  
  RENAME, 7-10  
  REPLACE, 7-11  
  RUN, 7-6  
  RUNNH, 7-6  
  SAVE, 7-3  
  SCALE, 7-14  
  TAPE, 4-10  
  UNSAVE, 8-5  
Command level, 1-3  
Commands  
  affecting system resources, 3-2t  
  BACKUP interruption, 17-19  
  batch, 21-2  
  batch utility, 21-12  
  DISMOUNT, 19-17  
  summary of, B-1 to B-3  
Commands, BASIC-PLUS  
  overview of, 6-2

Commands, system  
  BASIC-PLUS, 6-1

Comments  
  EXTEND/NOEXTEND format, 8-10

COMPILE command, 7-8  
  with file specification, 7-10  
  program on specific device, 7-10  
  purpose of, 7-8  
  result of, 7-9  
  on saved program, 7-9  
  specify protection code in, 7-10

Compiled program, 6-2  
  default protection code, 7-10  
  minimum size requirement, 7-9  
  save a, 7-9

Concise Command Language. *See CCL*

CONFIRM prompt, 1-4  
  possible replies, 1-4

CONT command, 9-3  
  continue program execution, 9-3

Continuation lines  
  EXTEND/NOEXTEND format, 8-10

Control characters, 4-11  
  ALTMODE key, 4-13  
  CTRL/C, 4-12, 9-3  
  CTRL/O, 4-12, 9-4  
  CTRL/Q, 4-13, 9-4  
  CTRL/R, 4-12  
  CTRL/S, 4-13, 9-4  
  CTRL/T, 4-13  
  CTRL/U, 7-3, 8-4  
  CTRL/Z, 4-13  
  ESCAPE key, 4-13  
  RETURN key, 4-13  
  summary of, B-3, B-4

COPY program, 16-29  
  block-by-block verification, 16-30  
  blocksize switch, 16-30  
  density switch, 16-31  
  error messages, 16-31  
  fast copy switch, 16-29  
  help message, 16-29  
  no copy switch, 16-30  
  parity switch, 16-31  
  presence of bad blocks, 16-29  
  restrictions on, 16-29  
  verify switch, 16-29

Core common, 20-11

Correspondence code keyboard, 23-27

CPU, 2-3

CPU time  
  finding job usage, 4-14

CREATION keyword  
  in BACKUP, 17-5

Cross-reference listing  
  generating a, 15-18

CTRL/C, 4-12  
  BREAK key as, 4-12  
  halt program execution, 9-3  
  integer variable LINE, 9-4

CTRL/O, 4-12  
  to continue terminal output, 9-4  
  suppress output to terminal, 9-4

CTRL/Q, 4-13

CTRL/R, 4-12  
  disable retype facility, 19-2  
  enable retype facility, 19-2

CTRL/S, 4-13  
  suppress terminal output with, 9-4

CTRL/T, 4-13  
  job states reported by, 4-14  
  status report format, 4-13

CTRL/U, 7-3, 8-4  
  delete program line with, 8-4

CTRL/Z, 4-13  
  mark the end-of-file, 4-13

Cursor, terminal, 23-19

## D

Data  
  mark the end of recorded, 4-13

Data, RSTS/E  
  transfer of with FLINT, 18-8

Data files  
  degrees of protection, 11-7

Data set  
  specifying known diskettes of, 18-6

Data Set Name, DSN, 18-3

DEASSIGN command, 4-3  
  cancel logical account assignment, 4-10  
  release logically named device, 4-5  
  releasing a device, 4-3

Debugging  
  example of program, 9-4  
  role of immediate mode in, 9-2  
  with STOP statements, 9-2

DECpack cartridge  
  logically mounting a, 4-8

DECtape, 2-7, 23-1, 23-11  
  advantages of, 2-7  
  controls, 23-12  
  copying information on, 16-29  
  disadvantages of, 2-7  
  mount a tape, 23-12  
  operating procedures, 23-12  
  transport, 23-11f

DECtape drive, 2-8

DECtape II, 2-8  
  advantages of, 2-8  
  disadvantages of, 2-8  
  list the directory of, 16-31

DECwriter  
  operator controls, 23-31

DELETE command, 8-3  
  cautions, 8-3  
  delete current program, 8-3  
  delete program section, 8-3  
  format of, 8-3

DELETE key, 7-3, 8-3

Delimiter, private, 19-4, 19-11.  
  *See also Private delimiter*

Density, magnetic tape  
  change with PIP, 16-27

Detached job  
  attach to, 13-5, 13-6  
  attach to terminal, 13-6

Device  
  associate with account and logical name, 4-6  
  associating multiple logical names, 4-4  
  associating physical name with, 4-5  
  initialize with PIP, 16-27  
  not available for use, 4-1  
  not configured on system, 4-2  
  queue a file from, 20-6  
  release logically named, 4-5  
  releasing an assigned, 4-3  
  remove file from, 8-5  
  reserve a logically named, 4-5  
  reserved to a job, 4-2  
  reserving a, 4-1  
  RT-11 directory structured, 16-34  
  running program from, 7-7  
  searching for specified, 4-9  
  specification, 2-9  
  specify in SAVE command, 7-4  
  status information, 14-6  
  transfer between two jobs, 4-3  
  transfer control of, 4-3

Device, physical  
  associate account with, 4-4  
  associate logical name with, 4-4  
  pack ID, 4-9

Device cluster number, 11-10

Device designator, 11-1  
  in file specification, 11-1  
  for non-file structured devices, 11-1

Device designators  
  list of, 11-2

Device directory  
  listing, 7-13

Device name  
  in a file specification, 2-10  
  logical, 2-10, 11-3  
  physical, 2-9, 11-3

Device specification, 11-2  
  in FIT program, 16-32  
  precede with underscore, 4-5

Device transfer, PIP, 16-1

Devices  
  assignable, 2-5  
  copying between, 16-29  
  degrees of accessibility, 2-5  
  file structured, 2-6  
  file transfer between, 16-31  
  input only, 2-2  
  input/output, 2-1  
  non-file structured, 2-6  
  output only, 2-2  
  peripheral, 23-1  
  releasing all assigned, 4-3  
  table of assignable, 4-2

Devices, non-file structured  
  designators for, 11-1

Devices, spooling, 20-1

Dial-up line  
  terminal connected to, 13-1

DIRECT command, 1-5

DIRECT program, 15-1  
  as a CCL command, 15-6  
  benefits of, 15-1  
  caching status, 15-3  
  error messages, 15-6  
  file attributes, 15-3  
  file placement, 15-2  
  files marked for deletion, 15-3  
  format of, 15-1  
  format of report, 15-5  
  header line, 15-1  
  list at line printer, 15-5  
  list several accounts, 15-5  
  list to file, 15-5  
  running by CCL command, 15-1  
  table of options, 15-2  
  wildcard specifications, 15-2

Directory, 1-5  
  from backup set index, 17-18  
  of backup set, 17-2  
  disk, 15-1  
  DOS disk listing, 16-37  
  elements of, 1-6  
  of IBM diskette, 18-1  
  listing, 15-1  
  listing device, 7-13

Directory, (Cont.)
 

- listing multi-volume magnetic tape, 16-17
- listing with batch, 21-13
- PIP to print listing, 16-25
- printed with CATALOG command, 7-13
- structure on RT-11 disk, 16-31
- zero for single density diskette, 18-8
- zero with PIP, 16-26

**Disk**, 2-9
 

- advantages of, 2-9
- backup files to, 17-2
- copying information on, 16-29
- determine file location, 11-10
- disadvantages of, 2-9
- private, 2-4
- public, 2-4
- public structure, 2-3
- quota report, 14-10
- specify file location on, 11-10
- status information, 14-6
- store current program on, 7-3
- system, 2-3

**Disk, directory**

- listing, 7-13

**Disk, private**

- contents of, 2-4

**Disk, public**

- contents of, 2-4

**Disk, RSTS/E**

- format of, 18-6
- transfer to IBM diskette, 18-2, 18-7

**Disk, system**

- contents of, 2-3

**Disk access**

- by logical name, 4-9
- by pack identification, 4-8

**Disk directory**, 15-1

**Disk drive**, 2-9

**Disk pack**

- logically mounting a, 4-8

**Disk pack, private**, 19-14
 

- in LOCK state, 19-14
- logical name, 19-14
- logically dismounting, 19-12, 19-16
- logically mounting, 19-12
- MOUNT command options, 19-13
- mount read only, 19-14
- no logical name, 19-14
- steps to logically mount, 19-12

**Diskette**

- erasing data on, 18-2
- erasing with FLINT, 18-8, 18-9
- flexible, 23-31
- initializing directory, 18-2

**Diskette, (Cont.)**

- initializing with FLINT, 18-8
- multi-volume data set transfer, 18-6
- operating procedures, 23-31
- single density, 18-8

**Diskette, flexible**, 2-8

**Diskette, IBM**

- directory headings, 18-3
- format of directory, 18-3
- listing directory, 18-2
- obtain directory of, 18-1
- RSTS/E data transfer, 18-3
- RSTS/E disk transfer, 18-2
- transfer data to RSTS/E disk, 18-1

**Diskette transfer**, 18-1

**DISMOUNT command**

- format of, 19-17
- /UNLOAD option, 19-17

**DOS disk**

- directory listing, 16-37
- list the directory of, 16-31

**DOS labels**, 5-2

**Dump, post-mortem**, 22-1

## E

**EBCD keyboard**, 23-28

**EBCDIC mode**, translate to ASCII, 18-4

**Editing, program**

- delete current line, 7-3
- erase incorrect characters, 7-3

**Editing tool**

- CTRL/U, 7-3
- RUBOUT, 7-3

**ERR variable**, C-2

**Error, user data**

- ignore on PIP transfer, 16-15

**Error codes**

- QUE program, 20-12

**Error handling**

- BACKUP program, 17-25
- BACKUP routines, 17-30

**Error messages**

- BACKUP interruption commands, 17-27
- BACKUP program dialogue, 17-25, 17-26
- BACKUP selection, 17-29
- BACKUP volume mount, 17-28
- BATCH program, 21-23
- BPCREF program, 15-21
- COPY program, 16-31
- DIRECT program, 15-6
- FILCOM program, 15-17
- FLINT program, 18-10, 18-11
- multiple meanings, C-3
- number of RSTS/E, C-3

Error messages, (Cont.)  
  PIP program, 16-28  
  QUE program, 20-12  
  severity standard in, C-2  
  SWITCH program, 5-4  
  TTYSET program, 19-8  
  UMOUNT program, 19-18  
Error messages, non-recoverable, C-2,  
  C-11 to C-16  
Error messages, recoverable, C-2,  
  C-4 to C-11  
Error trapping, C-1  
  exceptions to, C-3  
Error variable (ERR), C-2  
Errors  
  BACKUP program, 17-25  
Errors, BASIC-PLUS, C-1  
Errors, system, C-1  
ESC character, 19-4  
ESCAPE key, 4-13  
EXCEPT keyword  
  in BACKUP, 17-6  
EXIT command, 5-3  
EXTEND command, 8-9  
  format of, 8-10  
EXTEND format  
  comments, 8-10  
  continuation lines, 8-10  
  description of, 8-9  
  spaces and tabs, 8-10  
  variable/function names, 8-10  
EXTEND mode, 8-9  
Extension, 1-6, 11-4  
  .BAC, 6-2  
  .BAS, 6-1  
  .BAS default, 7-4  
BASIC-PLUS source program, 7-5  
blank, 11-6  
in CHAIN statement, 8-8  
default, 1-6  
maximum characters in, 1-6  
null, 11-6  
Extensions  
  batch standard, 21-5  
  list of common, 11-5

## F

FILCOM program, 15-7  
  compare ASCII files, 15-7  
  compare blank lines, 15-10  
  /COMPARE switch, 15-12  
  continuation lines, 15-9  
  default comparisons, 15-11

FILCOM program, (Cont.)  
  error messages, 15-17  
  examples, 15-14  
  identification header, 15-7  
  indirect command file, 15-8  
  /LIMIT switch, 15-12  
  multiple file compares, 15-13  
  prompts, 15-7, 15-8  
  single command line, 15-8, 15-10  
  specifying amount of comparison, 15-9  
  specifying file for comparison, 15-9  
  summary report, 15-13  
  switches, 15-11  
  wildcard specifications, 15-13  
File  
  block mode transfer with PIP, 16-13  
  change date of last access, 16-12  
  creating NONAME, 7-2  
  degrees of protection, 11-7  
  delete with KILL statement, 8-6  
  deletion during logout, 1-4  
  deletion of large protected, 11-8  
  deletion through LOGOUT, 13-7  
  determine location on disk, 11-10  
  extending with PIP, 16-13  
  greater than 65535 blocks, 17-11  
  indirect command, 12-2  
  level of protection, 1-7  
  move between systems, 11-15  
  pre-extending a, 11-8  
  printing with QUE, 20-6  
  queue from a device, 20-6  
  queueing a, 20-7  
  remove from device with UNSAVE, 8-5  
  report on open, 14-6  
  salvaging from damaged media, 16-15  
  scratch, 7-2  
  size, 1-10  
  specify location on disk, 11-10  
  temporary, 7-2  
  transfer with PIP, 16-10  
  zeroing with PIP, 16-27  
File, ASCII  
  line-by-line comparison, 15-7  
File, attributes  
  octal, 11-13  
  symbolic, 11-13  
File, cluster size  
  establish minimum, 11-9  
  for large files, 11-9  
  recommended, 11-9  
File, compiled  
  minimum size of, 7-9

File, data  
  CTRL/Z to mark the end of, 4-13

File, data format  
  translation of with PIP, 16-21

File, directory  
  with PIP, 16-25

File, privileged  
  deleting with PIP, 16-24

File, protected  
  overwrite with zeroes on delete, 1-9

File, queue  
  accessing, 20-2

File, RMS-11  
  transfer between RSTS/E systems, 16-13

File, temporary  
  subject to deletion, 11-4

File, transfer  
  BACKUP program, 17-3  
  block mode, 16-10, 16-13  
  date of creation on, 16-15  
  date of last access on, 16-15  
  disk and flexible diskette, 16-31  
  disk and TU58 DECTape, 16-31  
  DOS disk and RSTS disk, 16-31  
  with FIT, 16-32  
  flexible diskette, 18-1  
  IBM to RSTS/E, 18-1  
  image mode, 16-14  
  modify with PIP, 16-10  
  multi-volume magnetic tape, 16-17  
  with PIP, 16-10  
  preserving label formats, 16-10  
  protect file from deletion, 16-16  
  RSTS disk and RT-11 DECTape, 16-31  
  RSTS disk and RT-11 disk, 16-31  
  RSTS to non-RSTS procedures, 16-11  
  RSTS to RSTS procedures, 16-10  
  RSTS/E to IBM, 18-1, 18-7  
  specifying cluster size, 16-15  
  supress attributes, 16-16  
  tape to disk, 16-11  
  update on, 16-16

File attributes, 11-11  
  blocks in use, 11-12  
  bucket size, 11-12  
  carriage control, 11-12  
  DIRECT program, 15-3  
  file extension, 11-12  
  file organization, 11-12  
  header size, 11-12  
  octal, 11-13, 11-14t  
  PIP operations, 16-19  
  record format, 11-12  
  record size, 11-12

File attributes, (Cont.)  
  spanning, 11-12  
  supress on file transfer, 16-16  
  symbolic, 11-12

File characters  
  automatic generation, 19-9  
  generalized, 19-9

File deletion  
  with batch, 21-12

File deletion, on RT-11 device, 16-35

File directory, 1-5  
  elements of, 1-6

File extension  
  .BAC, 6-2  
  .BAS, 6-1

File organization  
  determining file attribute, 11-12

File ownership, 1-10

File placement  
  DIRECT program, 15-3

File protection  
  criteria for, 1-8

File protection codes, 11-7

File request queue block, 22-3

File specification, 1-6, 10-2t  
  in APPEND command, 8-7  
  batch processor, 21-5  
  change, 7-11  
  in COMPILE command, 7-10  
  contents of, 11-1  
  device designator in, 11-1  
  device name in, 2-10  
  elements of, 11-1  
  extension in, 11-4  
  filename in, 11-4  
  in FIT program, 16-32  
  in OLD command, 7-6  
  options, 11-8  
  PIP defaults, 16-4  
  project-programmer number in, 11-3  
  protection code in, 11-7  
  Q command jobname, 20-4  
  Q command options, 20-4  
  in REPLACE command, 7-11  
  rules for, 1-7  
  in RUN command, 7-7  
  in SAVE command, 7-4  
  system defaults, 11-1  
  wildcards in, 11-6

File specification options  
  /CLUSTERSIZE, 11-9  
  /FILESIZE, 11-8  
  format of, 11-8  
  /MODE, 11-11

File specification options, (Cont.)  
  /POSITION, 11-10  
  /ONLY, 11-11

File specifications  
  in BACKUP, 17-4  
  changing with PIP, 16-24

Filename, 1-6, 11-4  
  in a file specification, 11-4  
  in CHAIN statement, 8-8  
  maximum characters in, 1-6

Files  
  comparing with FILCOM, 15-7  
  copy with batch, 21-13  
  listing directory of, 15-1  
  off-line copies, 17-1  
  preserving with BACKUP, 17-1  
  renaming with PIP, 16-24  
  RMS-11, 11-13  
  RMS/ASCII translation, 16-21  
  storing off-line, 17-1

Files, large  
  selectively backup, 16-12

FILESIZE option, 11-8  
  arguments for, 11-8  
  with large files, 11-9  
  pre-extending a file, 11-8

Fill characters, 19-3  
  generalized, 19-10  
  table of, 19-10

FIT program, 16-31  
  compress RT-11 data, 16-35  
  device specification, 16-32  
  exit from, 16-32  
  file specifications, 16-32  
  help file, 16-32  
  initialize RT-11 device, 16-35  
  input devices, 16-33  
  invoking the, 16-31  
  output devices, 16-33  
  prompt, 16-32  
  RT-11 directory, 16-35  
  RT-11 file deletion, 16-35  
  switch specification, 16-33  
  transfer RT-11 .BAD files, 16-35  
  transferring files with, 16-32  
  watch switch, 16-32

FIT switches  
  /DI directory, 16-36  
  /DOS DOS format, 16-33  
  /LI directory, 16-36  
  /N directory segments, 16-36  
  /RSTS RSTS device, 16-33  
  RT-11 device, 16-35  
  /RT11 RT-11 directory, 16-37

FIT switches, (Cont.)  
  /RT11 RT-11 structure, 16-33  
  /SQ compress, 16-37  
  /ZE zero, 16-35

Flexible diskette, 2-8, 23-1, 23-31  
  advantages of, 2-8  
  controller, 23-31  
  disadvantages of, 2-8  
  list directory of, 16-31

FLINT program  
  erasing diskette, 18-9

FLINT program, 18-1  
  ASCII to EBCDIC translation, 18-7  
  blocking factor, 18-5  
  blocking factor computation, 18-8  
  error messages, 18-10, 18-11  
  IBM diskette to RSTS/E disk, 18-1  
  IBM transfer to RSTS/E, 18-3  
  image mode transfer, 18-7  
  initializing diskette, 18-9  
  initiation commands, 18-2  
  listing IBM diskette directory, 18-2  
  multi-volume data set transfer, 18-6  
  output file header record, 18-6  
  record size, 18-5  
  RSTS/E to IBM transfer, 18-7  
  running, 18-2  
  running with CCL command, 18-12

FLINT switches  
  /NH no logical header, 18-4

Form feed  
  disable, 19-2  
  enable, 19-2

Formatted ASCII, 16-19  
  PIP data format translation, 16-21

Formatted binary, 16-19  
  PIP data format translation, 16-21

FORTRAN carriage control, 16-19  
  PIP data format translation, 16-21

FORTRAN compiler  
  execute with batch, 21-19

Function names  
  EXTEND/NOEXTEND format, 8-10

Functions, BASIC-PLUS  
  summary of, A-2 to A-6

## G

GOTO statement, 9-3  
  resume program execution, 9-3

GRIPE program, 14-11  
  communicate with system manager, 14-11  
  identification header, 14-12  
  query line, 14-12  
  running the, 14-11

## H

HELLO command, 1-2  
Help file, 12-2  
obtaining for system program, 12-2  
requesting a, 12-2  
HELP program, 12-3  
CCL command, 12-3  
information on system programs, 12-3  
information on system resources, 12-3  
invoking, 12-3  
responses to, 12-3

## I

I/O devices, 2-1  
Immediate mode  
CHAIN statement in, 8-8  
executing statements in, 9-1  
illegal statements in, 9-2  
limitations of, 9-2  
PRINT LINE in, 9-4  
role in program debugging, 9-2  
Index file  
created with loadindex, 17-14  
Indirect command file, 12-2  
BACKUP program, 17-8  
create a, 12-2  
in FILCOM program, 15-8  
in PIP, 16-5  
running BACKUP from, 17-20  
running system program, 12-2  
INPUT statement, C-2  
Intermediate code, 6-2  
INUSE program, 14-12  
printout from, 14-12

## J

Job  
attach detached job to, 13-6  
attach to detached, 13-5, 13-6  
attach to terminal, 13-2  
changing default run-time system, 5-2  
core common area, 22-3  
current condition of, 14-5  
current size of, 14-5  
device reserved to, 4-2  
device transfer, 4-3  
executed by spooling program, 20-1  
finding CPU time, 4-14  
finding current operation of, 4-13  
finding current state of, 4-14  
generate status report on, 4-13  
login sequence, 13-3  
maximum logical assignments, 4-4

## Job, (Cont.)

modify parameters in queue, 20-10  
modify parameters of in queue, 20-3  
remove from queue, 20-3  
run from detached keyboard, 14-5  
run-time system, 14-6  
running from pseudo keyboard, 14-5  
running without terminal dialogue, 21-1  
sent to spooler, 20-9  
temporarily dropped temporary privilege, 14-5  
with temporary privilege, 14-5  
waiting in queue, 20-9

Job number, 1-2  
finding current, 4-13

Jobs  
list of all active, 14-5

## K

KEY command, 4-11  
enabling terminal echo, 4-11  
format of, 4-11  
Keyboard  
BCD, 23-29  
CALL/360 BASIC, 23-30  
correspondence code, 23-27  
EBCD, 23-28  
Keyboard, detached  
job running from, 14-5  
Keyboard, pseudo, 21-1  
Keyboard number  
finding current, 4-13  
KILL statement, 8-6  
delete file from directory, 8-6

## L

LENGTH command, 7-12  
finding program length, 7-12  
finding program maximum length, 7-12  
Library programs, 10-1  
LINE FEED key  
in tape mode, 4-11  
Line number  
specified in CHAIN statement, 8-8  
Line printer, 2-6, 23-1, 23-7, 23-8f  
advantages of, 2-6  
as spooling device, 20-1  
buffer, 23-8  
character set, 23-8  
control panel, 23-9  
controls, 23-10t  
disadvantages of, 2-6  
listing with SAVE command, 7-4

Line printer, (Cont.)  
load paper into, 23-9  
operation, 23-9  
Line termination, 4-13  
LIST command  
format of, 8-2  
print current program, 8-1  
print entire program, 8-1  
print program lines, 8-1  
summary of, 8-2  
LIST command, 8-1  
List mode, 17-4  
dialogue summary, 17-17  
LISTNH (no header) command, 8-2  
Loadindex mode, 17-2, 17-3  
dialogue, 17-3, 17-16  
dialogue summary, 17-16  
index file created with, 17-14  
Logical device name, 2-10, 4-4  
in ASSIGN command, 4-4  
associate with device and account, 4-6  
associate with physical device, 4-4  
disk access by, 4-9  
maximum assignments, 4-4  
pack ID as system-wide, 4-8  
releasing, 4-5  
reserving, 4-5  
system-wide, 2-11  
use of, 2-11  
Logical device names, 11-3  
account of system-wide, 4-7  
associating multiple, 4-4  
override system-wide account, 4-7  
system-wide, 4-7  
Logical unit assignments, 22-3  
Login procedure, 1-2  
LOGIN program, 13-1  
at a logged in terminal, 13-4  
from a logged out terminal, 13-1  
altering the, 13-1  
jobs running detached, 13-2  
login procedure, 13-2  
modifying the, 13-7  
LOGOUT program, 13-7  
CONFIRM responses, 13-8  
file deletion mode, 13-7  
file deletion responses, 13-7  
logout sequence, 13-8  
Logout sequence, 1-4  
Lower-case characters, 19-5

## M

Macro command, TTYSET, 19-5  
defining, 19-5

Magnetic tape, 2-8, 23-1, 23-13  
advantages of, 2-8  
ANSI label file transfer, 16-10  
ANSI labels, 5-2  
change density with PIP, 16-27  
change parity with PIP, 16-28  
changing labeling default, 5-1  
characteristics in REASSIGN, 4-3  
control panel, 23-14  
controller, 23-14  
copying information on, 16-29  
density settings in copy, 16-31  
disadvantages of, 2-8  
DOS labels, 5-2  
file transfer to disk, 16-11  
initialize an ANSI label, 16-27  
initialize with PIP, 16-27  
labeling default, 5-1  
logically dismounting, 19-12, 19-16  
logically mounting, 19-12  
mounting a, 19-15  
mounting read only, 19-16  
operating procedures, 23-16  
override label check on MOUNT, 19-15  
parity settings in copy, 16-31  
rewind and take off line, 19-17  
set density with MOUNT, 19-15  
set labeling default, 19-15  
set parity with MOUNT, 19-15  
specifying block size in file transfer, 16-14  
steps to logically mount, 19-15  
suppress rewind in PIP, 16-23  
threading diagram, 23-18f  
transport, 23-14  
transport controls, 23-15t  
wildcard searches with PIP, 16-23  
Magnetic tape, multi-volume  
directory listing on, 16-17  
file transfer, 16-17  
initiate transfer, 16-17  
PIP transfer switches, 16-17  
Magnetic tape drive, 2-8  
Magnetic tape labels, 5-2  
Magtape. *See Magnetic tape*  
Memory  
effect of NEW command, 7-2  
program currently in, 6-2  
Memory allocation  
SYSTAT option, 14-9  
MODE option, 11-11  
determine data transfer mode, 11-11  
format of, 11-11  
MONEY program, 14-10  
output from, 14-11  
report format, 14-11

MOUNT command, 4-8, 19-13  
/ANSI option, 19-15  
/DENSITY option, 19-15  
/DOS option, 19-15  
format of, 4-8, 19-13  
/JOB option, 19-16  
/LOCK option, 19-14  
/LOGICAL option, 19-14  
/NOCHECK option, 19-15  
options, 19-13  
/PARITY option, 19-15  
/PRIVATE option, 19-14  
/READ ONLY option, 19-14, 19-16  
Mounting, logically  
DECpack cartridge, 4-8  
disk pack, 4-8, 19-12  
magnetic tape, 19-12, 19-15

## N

NAME AS statement, 7-11  
to change protection code, 7-12  
format of, 7-11  
NEW command, 7-1  
effects in memory, 7-2  
NEW FILE NAME  
prompt, 7-1  
Nine thousand remote  
as spooling device, 20-1  
NOEXTEND command, 8-9  
format of, 8-10  
NOEXTEND format, 8-9  
comments, 8-10  
continuation lines, 8-10  
multiple statements on one line, 8-11  
spaces and tabs, 8-10  
variable/function names, 8-10  
NONAME file  
create a, 7-2  
restrictions on, 7-2  
NOTICE.TXT file  
system library, 13-3  
Null extension, 11-6

## O

OLD command, 7-5  
with file specification, 7-6  
format of, 7-5  
retrieve source file, 7-5  
OLD FILE NAME prompt, 7-5, 8-7  
ON ERROR GOTO statement, C-1  
Operator services, 20-1  
message to with batch, 21-15  
mount message to, 21-16

Operators, BASIC-PLUS  
summary of, A-2  
OPSER, operator services, 20-1

## P

Pack ID  
disk access by, 4-8  
physical device, 4-9  
system-wide logical name, 4-8  
Pack identification label. *See Pack ID*  
Paper tape, 2-7  
advantages of, 2-7  
as spooling device, 20-1  
disadvantages of, 2-7  
punch with SAVE command, 7-5  
Paper tape punch, 2-7, 23-1  
controls, 23-4  
high speed, 23-5, 23-6  
low speed, 23-4  
Paper tape reader, 2-7, 23-1  
controls, 23-4  
high speed, 23-5  
load tape into, 23-5  
low speed, 23-4  
position paper tape in, 23-4  
read source program from, 7-8  
Parity, magnetic tape  
change with PIP, 16-28  
Parity bit  
output, 19-11  
software conditioned to send, 19-11  
terminal, 19-4  
Partition name, 22-3  
Password, 1-1, 1-2  
prompt for, 1-1  
Peripheral devices, 23-1  
Physical device name, 2-9  
associating with device, 4-5  
preceded by underscore, 2-11  
Physical device names, 11-3  
PIP command  
/CL switch in, 16-15  
default device specification, 16-2  
default input file specification, 16-3  
default output file specification, 16-2  
default protection code, 16-2  
file specification defaults, 16-4  
indirect command file, 16-5  
input file restrictions, 16-4  
nested indirect commands, 16-5  
terminate a, 16-2  
wildcard specifications, 16-3  
PIP errors  
during file processing, 16-5

- PIP file transfer  
     based on date of creation, 16-21  
     based on date of last access, 16-21  
     excluding files from, 16-22  
     including files in, 16-22
- PIP program, 16-1  
     appending data to files, 16-13  
     ASCII data files, 16-13  
     assigning run-time system name, 16-16  
     block mode transfer, 16-10, 16-13  
     CCL command, 16-1  
     change RTS name during transfer, 16-16  
     change tape density, 16-27  
     change tape parity, 16-28  
     changing file specifications, 16-24  
     changing protection code, 16-25  
     command line specifications, 16-2  
     controlling tape wildcard search, 16-23  
     data format transfers, 16-20  
     date of creation on file transfer, 16-15  
     date of last access on file transfer, 16-15  
     deleting privileged file, 16-24  
     directory listing options, 16-25, 16-26t  
     directory listing switches, 16-25  
     error messages, 16-28  
     error recovery, 16-28  
     exit from, 16-2  
     extending files, 16-13  
     file attributes, 16-19  
     file data format translation, 16-21  
     file deletion, 16-24  
     file transfer operations, 16-10  
     file update on transfer, 16-16  
     halt switch, 16-23  
     ignore user data error, 16-15  
     initializing tapes, 16-27  
     input file specification, 16-3  
     invocation, 16-1  
     locking in memory, 16-28  
     multi-volume tape transfer, 16-17  
     output file specification, 16-2  
     prevent overwrite on transfer, 16-16  
     printing execution report, 16-23  
     printing help file, 16-9  
     privileged switches, 16-28  
     prompt, 16-1  
     rename files, 16-24  
     restrictions on zeroing files, 16-27  
     RMS to ASCII transfer, 16-21  
     RTS file transfer, 16-10, 16-11  
     run-time system assignment, 16-14  
     selective execution, 16-23  
     selectively backup large files, 16-12  
     setting priority, 16-28
- PIP program, (Cont.)  
     specifying block size, 16-14  
     specifying cluster size, 16-15  
     specifying mode, 16-15  
     suppress file attributes on transfer, 16-16  
     suppress tape rewind, 16-23  
     version identification, 16-23  
     wildcard specifications, 16-12  
     zeroing directory, 16-26
- PIP switches, 16-6  
     /AC change access date, 16-12  
     /AF after, 16-22  
     /AP append file, 16-13  
     /AS ASCII data files, 16-13  
     /BE before, 16-22  
     /BL block mode transfer, 16-13  
     /BR brief directory, 16-25  
     /BSIZE block size, 16-14  
     /CL cluster size, 16-15  
     /CRE date of creation, 16-21  
     date related, 16-21  
     /DE delete, 16-24  
     /DI directory, 16-25  
     /DLA date of last access, 16-21  
     /ER erase, 16-24  
     /EX extend file, 16-13  
     /F fast directory, 16-25  
     /GO ignore transfer error, 16-15  
     /HA halt, 16-23  
     /HE information, 16-9  
     /ID version, 16-23  
     /IG ignore transfer error, 16-15  
     /IN inspect, 16-23  
     /LI directory, 16-25  
     /LO log, 16-23  
     /LOCK privileged, 16-28  
     /MO mode specification, 16-15  
     on multi-volume tape transfer, 16-17  
     /NE new date of last access, 16-15  
     /NO no rewind, 16-23  
     /NOA no file attributes, 16-16  
     /NOL no log, 16-23  
     /ON, 16-22  
     /PR protect from deletion, 16-16  
     /PRIOR privileged, 16-28  
     /RE rename, 16-24  
     /RET retain date of last access, 16-15  
     /RMS file attribute transfer, 16-20  
     /RTS run-time system name, 16-16  
     /RW:NO no rewind, 16-23  
     /S directory, 16-25  
     /SIN since, 16-22  
     specifying, 16-3  
     table of, 16-6

PIP switches, (Cont.)  
  /TO today, 16-22  
  /UN until, 16-22  
  /UP file update, 16-16  
  /VE version, 16-23  
  /WA watch, 16-23  
  /WIPE wipeout, 16-24  
  /WO wipeout, 16-24  
  /ZE zero directory, 16-26

Placed files, 17-11

PMDUMP program, 22-1  
  conditions of, 22-1  
  invoking, 22-1, 22-2  
  output of, 22-2  
  running the, 22-2  
  when to use, 22-1

POSITION option, 11-10  
  arguments to, 11-10  
  device cluster number in, 11-10  
  device cluster size in, 11-10  
  format of, 11-10  
  specify file location on disk, 11-10

Post-mortem dump, 22-4f  
  contents of, 22-1, 22-2  
  formatting a, 22-1  
  generated under RSX, 22-1

Primary index file, 17-10  
  restore mode, 17-14

Private delimiter, 19-4, 19-11  
  creating a, 19-11  
  disable the, 19-4, 19-11  
  limitations on, 19-12  
  non-printable character as, 19-4, 19-11  
  printable character as, 19-11

Private disk  
  contents of, 2-4

Privilege, temporary, 14-5

Privileged file  
  deleting with PIP, 16-24

Program  
  BAC, 6-2  
  call existing, 7-2, 7-5  
  cause a temporary halt, 9-2  
  chaining to QUE from, 20-11  
  changing file specification, 7-11  
  compile on specific device, 7-10  
  compiled, 6-2  
  compiling a, 7-8  
  compiling a saved, 7-9  
  correcting typographical errors, 8-4  
  create a, 7-1, 7-2  
  current, 6-2  
  currently in memory, 6-2  
  debugging a, 9-1

Program, (Cont.)  
  delete current, 8-3  
  delete line with CTRL/U, 8-4  
  deleting lines, 8-3  
  editing lines with RUBOUT, 8-4  
  enter data to with batch, 21-15  
  entry point of, 22-3  
  finding current length, 7-12  
  finding maximum length, 7-12  
  formatting a, 8-9  
  input of new, 7-2, 7-3  
  list current, 8-1  
  name a, 7-1  
  rename the current, 7-10  
  replace a saved, 7-11  
  run-only, 6-2  
  running from private device, 7-7  
  running from public structure, 7-6  
  running from specific device, 7-7  
  save under different name, 7-4  
  saving the, 7-3  
  source, 6-1  
  store current on disk, 7-3  
  system library, 10-1  
  transfer control with CHAIN, 8-7  
  writing a, 7-1

Program, debugging  
  example of, 9-4

Program, line  
  delete current, 7-3

Program, privileged  
  overwrite with zeroes on delete, 1-9

Program, size  
  finding current, 4-14  
  finding maximum, 4-14

Program development  
  debugging phase, 9-1

Program execution  
  priority of, 7-7

Program header, 7-6, 8-2  
  system program, 12-2

Programs  
  .BAS, 6-1  
  merging, 8-6  
  run by CCL command, 12-1

Project-programmer number, 1-1, 11-3  
  in file specification, 11-3

Prompt  
  CONFIRM, 1-4  
  NEW FILE NAME, 7-1  
  OLD FILE NAME, 7-5, 8-7

Prompt, (Cont.)  
  READY, 1-3  
  system, 1-2  
Protection, file  
  degrees of, 11-7  
Protection code, 1-7, 11-7  
  changing default, 5-1  
  changing with NAME AS, 7-12  
  changing with PIP, 16-25  
  default for compiled program, 7-10  
  in file specification, 11-7  
  function of, 1-7  
  specify in COMPILE command, 7-10  
Protection codes  
  level of, 1-7  
  table of, 1-8  
Pseudo keyboard, 21-1  
  job running from, 14-5  
Public disk, 2-4  
  contents of, 2-4  
Public disk structure, 2-3  
Public structure  
  running program from, 7-6

## Q

QUE command, K  
  remove a job, 20-10  
QUE command, L, 20-8  
  listing, 20-10  
QUE command, M  
  modify job parameters, 20-10  
QUE command, Q  
  /AFTER option, 20-4  
  delete file after printing, 20-6  
  file specification, 20-4  
  format of, 20-3  
  no file header, 20-6  
  options, 20-6  
  priority scheme, 20-7  
  set job priority, 20-7  
  specify device, 20-7  
  specify number of copies, 20-6  
  specifying date, 20-4  
  specifying time, 20-4  
QUE command, S, 20-8  
  short listing, 20-10  
QUE commands, 20-2  
QUE program, 20-1  
  abort all requests, 20-3  
  chaining to from user program, 20-11  
  error codes, 20-12  
  error messages, 20-12  
  header, 20-2

QUE program, (Cont.)  
  job output options, 20-5  
  listing format, 20-8  
  priority scheme, 20-7  
  reporting error condition, 20-12  
  requests to, 20-1  
  running at logged out terminal, 20-15  
  running at terminal, 20-2  
  running by CCL command, 20-14  
  shortened listing, 20-10  
  specify job priority, 20-4  
  specifying relative date, 20-4  
  specifying relative time, 20-4  
  standard spooling programs, 20-1  
  terminate the, 20-2, 20-3  
QUEMAN, queue manager, 20-1  
Question mark  
  as wildcard, 11-6  
Queue  
  identify request in, 20-4  
  remove a job from, 20-10  
QUEUE CCL command, 21-21  
Queue file  
  accessing the, 20-2  
Queue manager, 20-1  
QUOLST program, 14-10  
  column headings, 14-10  
  output from, 14-10

## R

Radix-50 character set, D-3  
READY prompt, 1-3  
REASSIGN command, 4-3  
  magnetic tape characteristics, 4-3  
  transferring a device, 4-3  
Record format  
  determining file attribute, 11-12  
Record size  
  IBM, 18-5  
  RSTS/E, 18-5  
RENAME command, 7-10  
  change program name, 7-10  
REPLACE command, 7-11  
  exchange program of same name, 7-11  
  with file specification, 7-11  
Resident library  
  memory allocation, 14-9  
  status report, 14-6  
Restore  
  multiple operations, 17-2  
Restore mode, 17-2  
  bad block during, 17-30  
  dialogue, 17-2, 17-12

Restore mode, (Cont.)  
  dialogue summary, 17-12  
  files to another account, 17-15  
  index file, 17-14  
  listing file, 17-14  
RETURN key, 4-13  
  effect of, 1-3  
  enter line to system, 4-13  
  line termination, 4-13  
  in tape mode, 4-11, 4-13  
RJ2780  
  as spooling device, 20-1  
RMS (Record Management Services), 16-20  
RMS-11 software, 11-11  
RMS data files, 16-20  
RMS files  
  concatenate, 16-20  
  transfer sequential, 16-20  
  translate to ASCII, 16-21  
RMS format  
  PIP data format translation, 16-21  
RMS records, 16-20  
RMSCNV, RMS utility, 16-20  
RONLY option, 11-11  
  format of, 11-11  
  setting read only mode, 11-11  
RSTS/E file structure, 16-34  
RSTS/E system  
  gaining access to, 1-1  
RSX run-time system  
  post-mortem dump, 22-1  
RT-11  
  build directory structure, 16-36  
  directory structure, 16-31  
  directory structured device, 16-34  
RT-11 device  
  compress data on, 16-34, 16-37  
  delete a file on, 16-34, 16-35  
  initialize a, 16-34  
  list directory of, 16-36  
  obtain a directory, 16-34  
RT-11 disk  
  list the directory of, 16-31  
RT-11 file structure, 16-34  
  FIT program errors, 16-34  
RT-11 format  
  maintaining, 16-34  
RUBOUT key, 7-3, 8-3  
  editing program lines, 8-4  
  in tape mode, 4-11  
RUN command, 7-6  
  execute a program, 7-6  
  execute from specific device, 7-7  
  with file specification, 7-7

Run-time system  
  change name during PIP transfer, 16-16  
  change with EXIT, 5-3  
  change with SWITCH, 5-3  
  changing default for job, 5-2  
  finding current, 4-14  
  finding size of, 4-14  
  job, 14-6  
  memory allocation, 14-9  
  private default, 5-3  
  return to private default, 5-3  
  RSX, 22-1  
  status report, 14-6  
RUNNH (no header) command, 7-6

## S

SAVE command, 7-3  
  with file specification, 7-4  
  function of, 7-3  
  including filename and extension, 7-4  
  obtain line printer listing, 7-4  
  punch a paper tape, 7-5  
  specify a storage device, 7-4  
  store program under different name, 7-4  
SCALE command, 7-14  
  control scale factor, 7-14  
  control scaled arithmetic, 7-14  
  determine current scale factor, 7-15  
  disable scaled arithmetic, 7-15  
  specify scale factor, 7-15  
Scale factor, 7-14  
  determine current, 7-15  
  specify a, 7-15  
Scaled arithmetic, 7-14  
  disable, 7-15  
  using, 7-14  
Scratch file, 7-2  
Software Performance Report. *See SPR*  
SORT-11 program  
  batch execution, 21-18  
Source program, 6-1  
  BASIC-PLUS extension, 7-5  
  print a, 8-1  
  read from paper tape, 7-8  
  retrieve with OLD command, 7-5  
Spaces  
  EXTEND/NOEXTEND format, 8-10  
Spooler  
  job in hold status, 20-9  
  job in kill status, 20-9  
  job sent to, 20-9  
  status indicators, 20-9  
Spooling devices, 20-1  
  batch BA, 20-1

Spooling devices, (Cont.)  
line printer LP, 20-1  
nine thousand remote NR, 20-1  
paper tape PP, 20-1  
RJ2780 RJ, 20-1

Spooling program  
create request for, 20-3  
job priority, 20-4  
pending request for, 20-2  
specify a, 20-7  
specify MODE, 20-4

Spooling programs, standard, 20-1

Spooling services  
job executed by, 20-1  
using the, 20-1

SPR, C-16  
forms, C-16  
minimal information on, C-16  
preventing processing delays, C-16

Statement  
CHAIN, 8-7  
GOTO, 9-3  
INPUT, C-2  
KILL, 8-6  
NAME AS, 7-11  
ON ERROR GOTO, C-1  
STOP, 9-2

Statement separators, 8-11

Statements  
batch control, 21-1  
executing in immediate mode, 9-1  
illegal in immediate mode, 9-2

Statements, BASIC-PLUS  
summary of, A-6 to A-13

Status report  
printing for system, 14-1

Status report, system  
contents of, 14-4

STOP AT LINE message, 9-3

STOP statement, 9-2  
CONT command, 9-3  
debugging with, 9-2

SUBMIT CCL command, 21-21

SWITCH program  
change default run-time system, 5-3  
error messages, 5-4

SYSTAT  
as a CCL command, 14-9  
SYSTAT program, 14-1  
abbreviations used in, 14-6, 14-7t  
buffer status, 14-6  
busy device status information, 14-6  
current system information, 14-1  
disk status information, 14-6

SYSTAT program, (Cont.)  
job status information, 14-5  
large file systems, 14-9  
on large file systems, 14-3  
memory allocation, 14-9  
message receiver information, 14-6  
multiple options, 14-3  
open file report, 14-6  
output file created by, 14-2  
output responses, 14-1  
privileged options, 14-3  
privileged user running, 14-9  
resident library information, 14-6  
run-time system information, 14-6  
running logged into system, 14-1  
running not logged into system, 14-1  
sample output, 14-4  
status report contents, 14-4  
table of options, 14-2

System  
command level, 1-3  
enter line to, 4-13  
entering the, 1-3, 13-1  
halting the, 4-12  
job login sequence, 13-3  
leave the, 1-4, 13-7  
spooling services, 20-1

System, commands  
BASIC-PLUS, 6-1

System, large file  
DIRECT program, 15-2  
SYSTAT, 14-9  
SYSTAT on, 14-3

System, RSTS/E  
gaining access to, 1-1

System, status report  
printing a, 14-1

System disk, 2-3

System errors, C-1

System identification line, 13-1

System library  
NOTICE.TXT file, 13-3

System library programs, 10-1  
overview of, 10-2

System manager  
sending message to, 14-11

System message  
suppress printing, 1-3

System notices, 13-3

System program  
BACKUP, 17-1  
BATCH, 21-1  
BPCREF, 15-18  
COPY, 16-29

System program, (Cont.)
   
     DIRECT, 15-1
   
     execute with batch, 21-14
   
     FILCOM, 15-7
   
     FIT, 16-31
   
     FLINT, 18-1
   
     GRIPE, 14-11
   
     HELP, 12-3
   
     INUSE, 14-12
   
     LOGIN, 13-1
   
     LOGOUT, 13-7
   
     MONEY, 14-10
   
     PIP, 16-1
   
     PMDUMP, 22-1
   
     QUE, 20-1
   
     QUOLST, 14-10
   
     run by indirect command file, 12-2
   
     SWITCH, 5-3
   
     SYSTAT, 14-1
   
     TTYSET, 19-1
   
     UMOUNT, 19-12

System programs
   
     information on, 12-3
   
     obtaining help files for, 12-2
   
     program header, 12-2

System prompt, 1-2

System resources
   
     commands, 3-2

System status report
   
     contents of SYSTAT, 14-4

System traps
   
     asynchronous, 22-3
   
     synchronous, 22-3

System user
   
     classes of, 1-7

Systems
   
     move files between, 11-15

T

Tab control
   
     disable, 19-2
   
     enable, 19-2

Tabs
   
     EXTEND/NOEXTEND format, 8-10

TAPE command, 4-10, 4-11.
   
     *See also Tape mode*
  
     disable terminal echo, 4-10
   
     format of, 4-10

Tape mode, 4-11
   
     LINE FEED key in, 4-11
   
     RETURN key in, 4-11, 4-13
   
     RUBOUT key in, 4-11

Task
   
     octal dump of, 22-3

Task, (Cont.)
   
     size of, 22-3
   
     status flags, 22-3

Task builder
   
     /PM switch, 22-1
   
     post-mortem dump, 22-1
   
     TASK option, 22-3
   
     translate .MAP file to ASCII, 16-21

Teletype, 23-2

Teletype, keyboard, 23-3

Teletype, printer, 23-3

Teletype console, 23-2f
   
     components of, 23-3
   
     control knob, 23-3

Temporary file, 7-2
   
     subject to deletion, 11-4

Terminal, 2-6, 23-1
   
     with 1968 ASCII character set, 19-5
   
     advantages of, 2-6
   
     attach a job to, 13-2
   
     attach detached job to, 13-6
   
     binary mode, 19-4
   
     bringing on-line, 1-2
   
     connected to dial-up line, 13-1
   
     continue output, 9-4
   
     controls, 23-21
   
     cursor, 23-19
   
     declaring in use, 14-12
   
     disadvantages of, 2-6
   
     display, 23-19, 23-21f
   
     establish characteristics for, 19-1
   
     finding keyboard number of, 4-13
   
     list current characteristics, 19-2
   
     lower-case characters, 19-2, 19-5
   
     operating procedures, 23-21
   
     parity bit, 19-4
   
     print buffered input, 4-12
   
     resume output to, 4-13
   
     running QUE program at, 20-2
   
     set print line width, 19-2
   
     suppress output to, 4-12, 9-4
   
     suspend output to, 4-13
   
     transmit full character set, 19-3
   
     upper-case characters, 19-5

Terminal, 2741, 23-23
   
     ATTN key, 23-24
   
     BKSP key, 23-24
   
     bracket characters, 23-25
   
     full duplex mode, 23-23
   
     half duplex mode, 23-23
   
     RETURN key, 23-24

Terminal, logged out
   
     running QUE program, 20-15

Terminal characteristics, setting, 19-1

Terminal echo  
disabling, 4-10  
enabling, 4-11  
settings, 4-10  
Terminal interface, legal speed for, 19-3  
TTYSET command  
macro, 19-5, 19-6  
TTYSET commands, 19-1  
FILL, 19-9  
LC INPUT, 19-9  
LC OUTPUT, 19-9  
NO FILL, 19-9  
NO LC INPUT, 19-9  
NO LC OUTPUT, 19-9  
NO XON, 19-11  
SET DELIMITER, 19-11  
table of, 19-2  
XON, 19-11  
TTYSET error messages, 19-8  
TTYSET program, 19-1  
ALTMODE characters, 19-5  
default single characteristics, 19-6  
ESCAPE characters, 19-5  
fill characters, 19-9  
format, 19-1  
lower-case characters, 19-5  
macro commands, 19-5  
parity bit, 19-11  
PREFIX characters, 19-5  
private delimiters, 19-11  
run by CCL command, 19-12  
set terminal characteristics, 19-1  
table of fill characters, 19-10  
upper-case characters, 19-5  
user-defined macro commands, 19-5  
XON/XOFF commands, 19-9

## U

UMOUNT program, 19-12  
error messages, 19-18  
MOUNT command, 19-13  
MOUNT command options, 19-13  
table of options, 19-17  
UMOUNT program, DISMOUNT command,  
19-17  
/UNLOAD option, 19-17  
UMOUNT program, MOUNT command  
/ANSI option, 19-15  
/DENSITY option, 19-15  
/DOS option, 19-15  
/JOB option, 19-16

UMOUNT program, MOUNT command, (Cont.)  
/LOCK option, 19-14  
/LOGICAL option, 19-14  
/NOCHECK option, 19-15  
/PARITY option, 19-15  
/PRIVATE option, 19-14  
/READ ONLY option, 19-14, 19-16  
Underscore, in physical device name, 2-11  
UNSAVE command, 8-5  
format of, 8-5  
remove file from device, 8-5  
Upper-case characters, 19-5  
User Identification Code (UIC). *See*  
*Project-programmer number*

## V

Variable names  
EXTEND/NOEXTEND format, 8-10  
Variables, BASIC-PLUS  
summary of, A-1

## W

Wildcard  
asterisk, 11-6  
combining asterisk and question mark, 11-6  
controlling tape search with PIP, 16-23  
in FILCOM program, 15-13  
in PIP command line, 16-3  
question mark, 11-6  
Wildcard characters, 11-6  
Wildcard operations  
selective PIP execution, 16-23  
Wildcards  
in DIRECT program, 15-2  
in file specifications, 11-6  
in PIP command line, 16-12

Work file  
backup, 17-10  
BACKUP program, 17-1  
default filename, 17-10  
estimating length, 17-10  
Work file, backup, 17-10

## X

XOFF character, 19-2  
XON character, 19-2  
XON/XOFF  
remote reader control, 19-9  
TTYSET commands, 19-11  
XON/XOFF processing, 19-2

RSTS/E  
System User's Guide  
AD-5133B-T1

**READER'S COMMENTS**

Your evaluation will help us improve this manual in future revisions. Please take a few minutes to complete this comment form.

1. What is your computer application? \_\_\_\_\_
2. How do you rate this manual, compared with others you have read? Mark a point on the scale below.



3. Did you find errors in the manual? If so, specify by page. \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

4. Could some of the terms and concepts have been explained better? Please specify. \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

5. Where would more (or better) examples have helped clarify the text? \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

6. Is the information presented in the most logical order for your needs? Can you suggest improvements in the format or organization? \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

7. Did the table of contents, index, and other aids help you to easily retrieve information? If not, please specify problem areas. \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

8. Please indicate the type of user that you most nearly represent:

- Experienced programmer       User with little programming  
 Student programmer      experience  
 Non-programmer interested in computers

9. What improvements to the manual do you recommend? \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_ Position \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code or Country \_\_\_\_\_

Do you require a response?  Yes  No

-----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